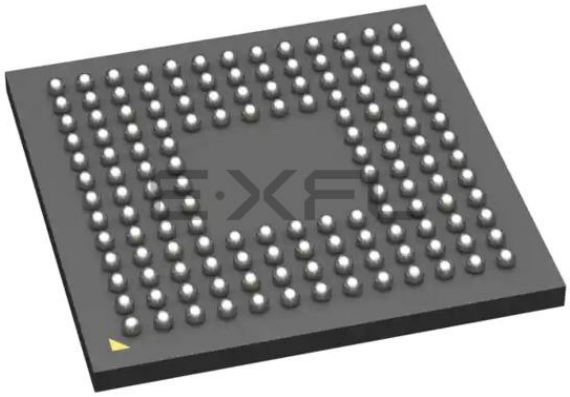


Welcome to E-XFL.COM

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	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	75
Program Memory Size	2MB (2M 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	144-UFBGA
Supplier Device Package	144-UFBGA (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsams70q21b-cfnt

LQFP Pin	QFN Pin (11)	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral CDir		PIO Peripheral DDir		Reset State
				Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
55	55	VDDIO	GPIO	PD3	I/O	–	–	GTX1	O	PWMC1_PWMH1	O	UTXD4	O	RI0	I	PIO, I, PU, ST
54	54	VDDIO	GPIO_CLK	PD4	I/O	–	–	GRXDV	I	PWMC1_PWML2	O	TRACED0	O	–	–	PIO, I, PU, ST
53	53	VDDIO	GPIO_CLK	PD5	I/O	–	–	GRX0	I	PWMC1_PWMH2	O	TRACED1	O	–	–	PIO, I, PU, ST
51	51	VDDIO	GPIO_CLK	PD6	I/O	–	–	GRX1	I	PWMC1_PWML3	O	TRACED2	O	–	–	PIO, I, PU, ST
50	50	VDDIO	GPIO_CLK	PD7	I/O	–	–	GRXER	I	PWMC1_PWMH3	O	TRACED3	O	–	–	PIO, I, PU, ST
49	49	VDDIO	GPIO_CLK	PD8	I/O	–	–	GMDC	O	PWMC0_PWMFI1	I	–	–	TRACECLK	O	PIO, I, PU, ST
48	48	VDDIO	GPIO_CLK	PD9	I/O	–	–	GMDIO	I/O	PWMC0_PWMFI2	I	AFE1_ADTRG	I	–	–	PIO, I, PU, ST
44	44	VDDIO	GPIO_MLB	PD10	I/O	–	–	GCRS	I	PWMC0_PWML0	O	TD	O	MLBSIG -	I/O -	PIO, I, PD, ST
43	43	VDDIO	GPIO_AD	PD11	I/O	–	–	GRX2	I	PWMC0_PWMH0	O	GTSUCOMP	O	ISI_D5	I	PIO, I, PU, ST
41	41	VDDIO	GPIO_AD	PD12	I/O	–	–	GRX3	I	CANTX1	O	SPI0_NPCS2	O	ISI_D6	I	PIO, I, PU, ST
26	26	VDDIO	GPIO_AD	PD21	I/O	–	–	PWMC0_PWMH1	O	SPI0_MOSI	I/O	TIOA11	I/O	ISI_D1	I	PIO, I, PU, ST
25	25	VDDIO	GPIO_AD	PD22	I/O	–	–	PWMC0_PWMH2	O	SPI0_SPCK	O	TIOB11	I/O	ISI_D0	I	PIO, I, PU, ST
22	22	VDDIO	GPIO_AD	PD24	I/O	–	–	PWMC0_PWML0	O	RF	I/O	TCLK11	I	ISI_HSYNC	I	PIO, I, PU, ST
20	20	VDDIO	GPIO_AD	PD25	I/O	–	–	PWMC0_PWML1	O	SPI0_NPCS1	I/O	URXD2	I	ISI_VSYNC	I	PIO, I, PU, ST
21	21	VDDIO	GPIO	PD26	I/O	–	–	PWMC0_PWML2	O	TD	O	UTXD2	O	UTXD1	O	PIO, I, PU, ST
2	3	VDDIO	GPIO_AD	PD31	I/O	–	–	QIO3	I/O	UTXD3	O	PCK2	O	ISI_D11	I	PIO, I, PU, ST
3	4	VDDOUT	Power	VDDOUT	–	–	–	–	–	–	–	–	–	–	–	–
4	5	VDDIN	Power	VDDIN	–	–	–	–	–	–	–	–	–	–	–	–
5	6	VDDIO	Reference	VREFP	I	–	–	–	–	–	–	–	–	–	–	–
36	36	VDDIO	RST	NRST	I/O	–	–	–	–	–	–	–	–	–	–	PIO, I, PU
37	37	VDDIO	TEST	TST	I	–	–	–	–	–	–	–	–	–	–	I, PD
10, 42, 58	10,42,58	VDDIO	Power	VDDIO	–	–	–	–	–	–	–	–	–	–	–	–
45	45	VDDIO	TEST	JTAGSEL	I	–	–	–	–	–	–	–	–	–	–	I, PD
13, 24, 61	13,24,61	VDDCORE	Power	VDDCORE	–	–	–	–	–	–	–	–	–	–	–	–
52	52	VDDPLL	Power	VDDPLL	–	–	–	–	–	–	–	–	–	–	–	–
59	59	VDDUTMII	USBHS	DM	I/O	–	–	–	–	–	–	–	–	–	–	–
60	60	VDDUTMII	USBHS	DP	I/O	–	–	–	–	–	–	–	–	–	–	–
14, 31	14,31	GND	Ground	GND	–	–	–	–	–	–	–	–	–	–	–	–
6	-	GND	Ground	GND	-	-	-	-	-	-	-	-	-	-	-	-
64	1	VDDPLLUSB	Power	VDDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–
--	62	--	VBG	VBG	I	-	-	-	-	-	-	-	-	-	-	-

Note:

1. To select this extra function, refer to the 32.5.14 Parallel Capture Mode section in the Parallel Input/Output Controller (PIO) chapter.

After reset, depending on the I/O, pullup or pulldown can be set.

32.5.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the Enable Register (PIO_PER) and the Disable Register (PIO_PDR). The Status Register (PIO_PSR) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of zero indicates that the pin is controlled by the corresponding on-chip peripheral selected in the Peripheral ABCD Select registers (PIO_ABCDSR1 and PIO_ABCDSR2). A value of one indicates the pin is controlled by the PIO Controller.

If a pin is used as a general-purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns a one for the corresponding bit.

After reset, the I/O lines are controlled by the PIO Controller, i.e., PIO_PSR resets at one. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset, or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level and depends on the multiplexing of the device.

32.5.3 Peripheral A or B or C or D Selection

The PIO Controller provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by writing PIO_ABCDSR1 and PIO_ABCDSR2.

For each pin:

- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral A is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral B is selected.
- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral C is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral D is selected.

Note that multiplexing of peripheral lines A, B, C and D only affects the output line. The peripheral input lines are always connected to the pin input (refer to Figure 32-2).

Writing in PIO_ABCDSR1 and PIO_ABCDSR2 manages the multiplexing regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in PIO_ABCDSR1 and PIO_ABCDSR2 in addition to a write in PIO_PDR.

After reset, PIO_ABCDSR1 and PIO_ABCDSR2 are zero, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O Line mode.

If the software selects a peripheral A, B, C or D which does not exist for a pin, no alternate functions are enabled for this pin and the selection is taken into account. The PIO Controller does not carry out checks to prevent selection of a peripheral which does not exist.

32.5.4 Output Control

When the I/O line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at zero, the drive of the I/O line is controlled by the peripheral. Peripheral A or B or C or D depending on the value in PIO_ABCDSR1 and PIO_ABCDSR2 determines whether the pin is driven or not.

The size of the data which can be read in PIO_PCRHR can be programmed using the DSIZE field in PIO_PCMR. If this data size is larger than 8 bits, then the Parallel Capture mode samples several sensor data to form a concatenated data of size defined by DSIZE. Then this data is stored in PIO_PCRHR and the flag DRDY is set to one in PIO_PCISR.

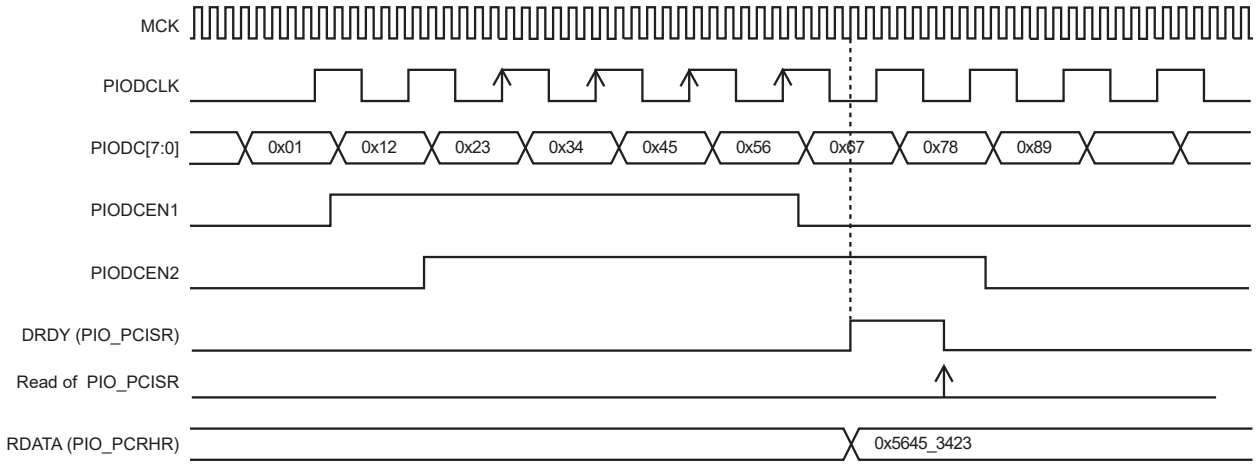
The Parallel Capture mode can be associated with a reception channel of the DMA Controller. This performs reception transfer from Parallel Capture mode to a memory buffer without any intervention from the CPU.

The Parallel Capture mode can take into account the sensor data enable signals or not. If the bit ALWYS is set to zero in PIO_PCMR, the Parallel Capture mode samples the sensor data at the rising edge of the sensor clock only if both data enable signals are active (at one). If the bit ALWYS is set to one, the Parallel Capture mode samples the sensor data at the rising edge of the sensor clock whichever the data enable signals are.

The Parallel Capture mode can sample the sensor data only one time out of two. This is particularly useful when the user wants only to sample the luminance Y of a CMOS digital image sensor which outputs a YUV422 data stream. If the HALFS bit is set to zero in PIO_PCMR, the Parallel Capture mode samples the sensor data in the conditions described above. If the HALFS bit is set to one in PIO_PCMR, the Parallel Capture mode samples the sensor data in the conditions described above, but only one time out of two. Depending on the FRSTS bit in PIO_PCMR, the sensor can either sample the even or odd sensor data. If sensor data are numbered in the order that they are received with an index from zero to n, if FRSTS equals zero then only data with an even index are sampled. If FRSTS equals one, then only data with an odd index are sampled. If data is ready in PIO_PCRHR and it is not read before a new data is stored in PIO_PCRHR, then an overrun error occurs. The previous data is lost and the OVRE flag in PIO_PCISR is set to one. This flag is automatically reset when PIO_PCISR is read (reset after read).

The flags DRDY and OVRE can be a source of the PIO interrupt.

Figure 32-10. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 0)



32.6.1.5 PIO Output Disable Register

Name: PIO_ODR
Offset: 0x0014
Property: Write-only

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

Bit	31	30	29	28	27	26	25	24
	P31	P30	P29	P28	P27	P26	P25	P24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	P23	P22	P21	P20	P19	P18	P17	P16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	P15	P14	P13	P12	P11	P10	P9	P8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	P7	P6	P5	P4	P3	P2	P1	P0
Access								
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, 24, 25, 26, 27, 28, 29, 30, 31 – P PIO Output Disable

Value	Description
0	No effect.
1	Disables the output on the I/O line.

Ethernet frames are normally stored in DMA memory complete with the FCS. Setting the FCS remove bit in the network configuration (bit 17) causes frames to be stored without their corresponding FCS. The reported frame length field is reduced by four bytes to reflect this operation.

The receive block signals to the register block to increment the alignment, CRC (FCS), short frame, long frame, jabber or receive symbol errors when any of these exception conditions occur.

If bit 26 is set in the network configuration, CRC errors will be ignored and CRC errored frames will not be discarded, though the Frame Check Sequence Errors statistic register will still be incremented. Additionally, if not enabled for jumbo frames mode, then bit[13] of the receiver descriptor word 1 will be updated to indicate the FCS validity for the particular frame. This is useful for applications such as EtherCAT whereby individual frames with FCS errors must be identified.

Received frames can be checked for length field error by setting the length field error frame discard bit of the Network Configuration register (bit-16). When this bit is set, the receiver compares a frame's measured length with the length field (bytes 13 and 14) extracted from the frame. The frame is discarded if the measured length is shorter. This checking procedure is for received frames between 64 bytes and 1518 bytes in length.

Each discarded frame is counted in the 10-bit length field error statistics register. Frames where the length field is greater than or equal to 0x0600 hex will not be checked.

38.6.6 Checksum Offload for IP, TCP and UDP

The GMAC can be programmed to perform IP, TCP and UDP checksum offloading in both receive and transmit directions, which is enabled by setting bit 24 in the Network Configuration register for receive and bit 11 in the DMA Configuration register for transmit.

IPv4 packets contain a 16-bit checksum field, which is the 16-bit 1’s complement of the 1’s complement sum of all 16-bit words in the header. TCP and UDP packets contain a 16-bit checksum field, which is the 16-bit 1’s complement of the 1’s complement sum of all 16-bit words in the header, the data and a conceptual IP pseudo header.

To calculate these checksums in software requires each byte of the packet to be processed. For TCP and UDP this can use a large amount of processing power. Offloading the checksum calculation to hardware can result in significant performance improvements.

For IP, TCP or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

38.6.6.1 Receiver Checksum Offload

When receive checksum offloading is enabled in the GMAC Network Configuration Register (NCFGR.RXCOEN), the IPv4 header checksum is checked as per RFC 791, where the packet meets the following criteria:

- If present, the VLAN header must be four octets long and the CFI bit must not be set.
- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding.
- IPv4 packet
- IP header is of a valid length

The GMAC also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)

Offset	Name	Bit Pos.								
		23:16	NFRX[23:16]							
		31:24	NFRX[31:24]							
0x0180	GMAC_TMXBFR	7:0	NFRX[7:0]							
		15:8	NFRX[15:8]							
		23:16	NFRX[23:16]							
		31:24	NFRX[31:24]							
0x0184	GMAC_UFR	7:0	UFRX[7:0]							
		15:8							UFRX[9:8]	
		23:16								
		31:24								
0x0188	GMAC_OFR	7:0	OFRX[7:0]							
		15:8							OFRX[9:8]	
		23:16								
		31:24								
0x018C	GMAC_JR	7:0	JRX[7:0]							
		15:8							JRX[9:8]	
		23:16								
		31:24								
0x0190	GMAC_FCSE	7:0	FCKR[7:0]							
		15:8							FCKR[9:8]	
		23:16								
		31:24								
0x0194	GMAC_LFFE	7:0	LFER[7:0]							
		15:8							LFER[9:8]	
		23:16								
		31:24								
0x0198	GMAC_RSE	7:0	RXSE[7:0]							
		15:8							RXSE[9:8]	
		23:16								
		31:24								
0x019C	GMAC_AE	7:0	AER[7:0]							
		15:8							AER[9:8]	
		23:16								
		31:24								
0x01A0	GMAC_RRE	7:0	RXRER[7:0]							
		15:8	RXRER[15:8]							
		23:16							RXRER[17:16]	
		31:24								
0x01A4	GMAC_ROE	7:0	RXOVR[7:0]							
		15:8							RXOVR[9:8]	
		23:16								
		31:24								
0x01A8	GMAC_IHCE	7:0	HCKER[7:0]							
		15:8								
		23:16								
		31:24								

39. USB High-Speed Interface (USBHS)

39.1 Description

The USB High-Speed Interface (USBHS) complies with the Universal Serial Bus (USB) 2.0 specification in all speeds.

Each pipe/endpoint can be configured in one of several USB transfer types. It can be associated with one, two or three banks of a DPRAM used to store the current data payload. If two or three banks are used, then one DPRAM bank is read or written by the CPU or the DMA, while the other is read or written by the USBHS core. This feature is mandatory for isochronous pipes/endpoints.

The following table describes the hardware configuration of the USB MCU device.

Table 39-1. Description of USB Pipes/Endpoints

Pipe/ Endpoint	Mnemonic	Max. Number Banks	DMA	High Band Width	Max. Pipe/ Endpoint Size	Type
0	PEP_0	1	N	N	64	Control
1	PEP_1	3	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
2	PEP_2	3	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
3	PEP_3	2	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
4	PEP_4	2	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
5	PEP_5	2	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
6	PEP_6	2	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
7	PEP_7	2	Y	Y	1024	Isochronous/Bulk/ Interrupt/Control
8	PEP_8	2	N	Y	1024	Isochronous/Bulk/ Interrupt/Control
9	PEP_9	2	N	Y	1024	Isochronous/Bulk/ Interrupt/Control

39.2 Embedded Characteristics

- Compatible with the USB 2.0 Specification
- Supports High-Speed (480 Mbps), Full-Speed (12 Mbps) and Low-Speed (1.5 Mbps) Communication

4. USBHS_DEVCTRL.RMWKUP is cleared at the end of the upstream resume.
5. When the controller detects a valid “End of Resume” signal from the host, the End of Resume (USBHS_DEVISR.EORSM) interrupt is set.

39.5.2.10 STALL Request

For each endpoint, the STALL management is performed using:

- the STALL Request (USBHS_DEVEPTIMRx.STALLRQ) bit to initiate a STALL request,
- the STALLed Interrupt (USBHS_DEVEPTISRx.STALLEDI) bit, which is set when a STALL handshake has been sent.

To answer the next request with a STALL handshake, USBHS_DEVEPTIMRx.STALLRQ has to be set by writing a one to the STALL Request Set (USBHS_DEVEPTIERx.STALLRQS) bit. All following requests are discarded (USBHS_DEVEPTISRx.RXOUTI, etc. is not be set) and handshaked with a STALL until the USBHS_DEVEPTIMRx.STALLRQ bit is cleared, which is done when a new SETUP packet is received (for control endpoints) or when the STALL Request Clear (USBHS_DEVEPTIMRx.STALLRQC) bit is written to one.

Each time a STALL handshake is sent, the USBHS_DEVEPTISRx.STALLEDI bit is set by the USBHS and the PEP_x interrupt is set.

Special Considerations for Control Endpoints

If a SETUP packet is received into a control endpoint for which a STALL is requested, the Received SETUP Interrupt (USBHS_DEVEPTISRx.RXSTPI) bit is set and USBHS_DEVEPTIMRx.STALLRQ and USBHS_DEVEPTISRx.STALLEDI are cleared. The SETUP has to be ACKed.

This simplifies the enumeration process management. If a command is not supported or contains an error, the user requests a STALL and can return to the main task, waiting for the next SETUP request.

STALL Handshake and Retry Mechanism

The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the USBHS_DEVEPTIMRx.STALLRQ bit is set and if no retry is required.

39.5.2.11 Management of Control Endpoints

Overview

A SETUP request is always ACKed. When a new SETUP packet is received, the USBHS_DEVEPTISRx.RXSTPI is set; the Received OUT Data Interrupt (USBHS_DEVEPTISRx.RXOUTI) bit is not.

The FIFO Control (USBHS_DEVEPTIMRx.FIFOCON) bit and the Read/Write Allowed (USBHS_DEVEPTISRx.RWALL) bit are irrelevant for control endpoints. The user never uses them on these endpoints. When read, their values are always zero.

Control endpoints are managed using:

- the USBHS_DEVEPTISRx.RXSTPI bit, which is set when a new SETUP packet is received and which is cleared by firmware to acknowledge the packet and to free the bank;
- the USBHS_DEVEPTISRx.RXOUTI bit, which is set when a new OUT packet is received and which is cleared by firmware to acknowledge the packet and to free the bank;
- the Transmitted IN Data Interrupt (USBHS_DEVEPTISRx.TXINI) bit, which is set when the current bank is ready to accept a new IN packet and which is cleared by firmware to send the packet.
  Control Write

39.6.4 General Status Set Register

Name: USBHS_SFR
Offset: 0x080C
Property: Write-only

This register always reads as zero.

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
							VBUSRQS	
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				RDERRIS				
Access								
Reset								

Bit 9 – VBUSRQS VBUS Request Set
Must be set to ‘1’.

Value	Description
0	No effect.
1	Sets the VBUSRQ bit in USBHS_SR.

Bit 4 – RDERRIS Remote Device Connection Error Interrupt Set

Value	Description
0	No effect.
1	Sets the RDERRI bit in USBHS_SR, which may be useful for test or debug purposes.

39.6.20 Device Endpoint Interrupt Set Register (Isochronous Endpoints)

Name: USBHS_DEVEPTIFRx (ISOENPT)
Offset: 0x0190 + x*0x04 [x=0..9]
Reset: 0
Property: Read/Write

This register view is relevant only if EPTYPE = 0x1 in "Device Endpoint x Configuration Register".

For additional information, see "Device Endpoint x Status Register (Isochronous Endpoints)".

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

0: No effect.

1: Sets the corresponding bit in USBHS_DEVEPTISRx, which may be useful for test or debug purposes.

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
				NBUSYBKS				
Access								
Reset				0				

Bit	7	6	5	4	3	2	1	0
	SHORTPACKETS	CRCERRIS	OVERFIS	HBISOFLUSHIS	HBISOINERRIS	UNDERFIS	RXOUTIS	TXINIS
Access								
Reset	0	0	0	0	0	0	0	0

Bit 12 – NBUSYBKS Number of Busy Banks Interrupt Set

Bit 7 – SHORTPACKETS Short Packet Interrupt Set

Bit 6 – CRCERRIS CRC Error Interrupt Set

Bit 5 – OVERFIS Overflow Interrupt Set

Bit 4 – HBISOFLUSHIS High Bandwidth Isochronous IN Flush Interrupt Set

Bit 3 – HBISOINERRIS High Bandwidth Isochronous IN Underflow Error Interrupt Set

Bit 2 – UNDERFIS Underflow Interrupt Set

Bit 5 – END_BF_ST End of Channel Buffer Status
Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

Value	Description
0	Cleared automatically when read by software.
1	Set by hardware when the BUFF_COUNT count-down reaches zero.

Bit 4 – END_TR_ST End of Channel Transfer Status
Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

Value	Description
0	Cleared automatically when read by software.
1	Set by hardware when the last packet transfer is complete, if the USBHS device has ended the transfer.

Bit 1 – CHANN_ACT Channel Active Status
When a packet transfer is ended, this bit is automatically reset.

When a packet transfer cannot be completed due to an END_BF_ST, this flag stays set during the next channel descriptor load (if any) and potentially until completion of a USBHS packet transfer, if allowed by the new descriptor.

Value	Description
0	The DMA channel is no longer trying to source the packet data.
1	The DMA channel is currently trying to source packet data, i.e., selected as the highest-priority requesting channel.

Bit 0 – CHANN_ENB Channel Enable Status
When any transfer is ended either due to an elapsed byte count or to completion of a USBHS device-initiated transfer, this bit is automatically reset.

This bit is normally set or cleared by writing into the USBHS_DEVDMACONTROLx.CHANN_ENB bit field either by software or descriptor loading.

If a channel request is currently serviced when the USBHS_DEVDMACONTROLx.CHANN_ENB bit is cleared, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

Value	Description
0	If cleared, the DMA channel no longer transfers data, and may load the next descriptor if the USBHS_DEVDMACONTROLx.LDNXT_DSC bit is set.
1	If set, the DMA channel is currently enabled and transfers data upon request.

39.6.31 Host General Control Register

Name: USBHS_HSTCTRL
Offset: 0x0400
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
			SPDCONF[1:0]			RESUME	RESET	SOFE
Access								
Reset			0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 13:12 – SPDCONF[1:0] Mode Configuration
This field contains the host speed capability:.

Value	Name	Description
0	NORMAL	The host starts in Full-speed mode and performs a high-speed reset to switch to High-speed mode if the downstream peripheral is high-speed capable.
1	LOW_POWER	For a better consumption, if high speed is not needed.
2	HIGH_SPEED	Forced high speed.
3	FORCED_FS	The host remains in Full-speed mode whatever the peripheral speed capability.

Bit 10 – RESUME Send USB Resume
This bit is cleared when the USB Resume has been sent or when a USB reset is requested.

This bit should be written to one only when the start of frame generation is enabled (SOFE = 1).

Value	Description
0	No effect.
1	Generates a USB Resume on the USB bus.

– The SPCK pin is driven by the transmitter to synchronize the receiver.
– The NPCS0 pin becomes an input, and is used as a slave select signal (NSS)
– The NPCS1 to NPCS3 pins are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operation. The baud rate generator is activated only in Master mode.

41.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the SPI Chip Select registers (SPI_CSRx). The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on 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. Consequently, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are connected and require different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

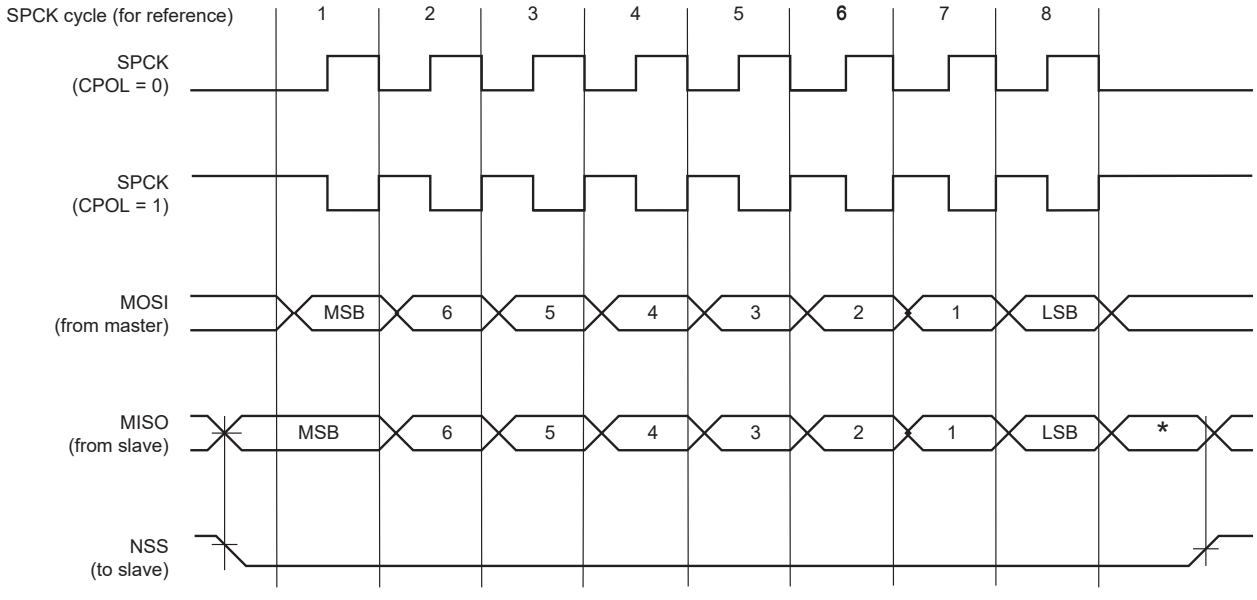
The table below shows the four modes and corresponding parameter settings.

Table 41-2. SPI Bus Protocol Modes

SPI Mode	CPOL	NCPHA	Shift SPCK Edge	Capture SPCK Edge	SPCK Inactive Level
0	0	1	Falling	Rising	Low
1	0	0	Rising	Falling	Low
2	1	1	Rising	Falling	High
3	1	0	Falling	Rising	High

The following figures show examples of data transfers.

Figure 41-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined.

41.8.4 SPI Transmit Data Register

Name: SPI_TDR
Offset: 0x0C
Reset: –
Property: Write-only

Bit	31	30	29	28	27	26	25	24
								LASTXFER
Access								W
Reset								–

Bit	23	22	21	20	19	18	17	16
					PCS[3:0]			
Access					W	W	W	W
Reset					–	–	–	–

Bit	15	14	13	12	11	10	9	8
	TD[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	–	–	–	–	–	–	–	–

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

Bit 24 – LASTXFER Last Transfer
This field is only used if variable peripheral select is active (SPI_MR.PS = 1).

Value	Description
0	No effect
1	The current NPCS is deasserted after the transfer of the character written in TD. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Bits 19:16 – PCS[3:0] Peripheral Chip Select
This field is only used if variable peripheral select is active (SPI_MR.PS = 1).

If SPI_MR.PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don’t care)

Value	Description
0	No effect.
1	Disables the QSPI.

Bit 0 – QSPIEN QSPI Enable

Value	Description
0	No effect.
1	Enables the QSPI to transfer and receive data.

Figure 44-17. Transmit Frame Format in Continuous Mode (STTDLY = 0)

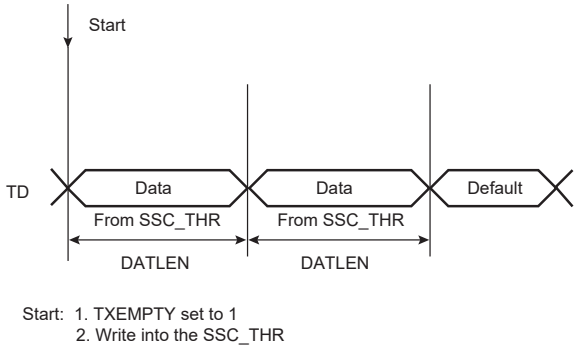
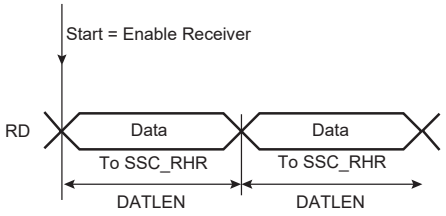


Figure 44-18. Receive Frame Format in Continuous Mode (STTDLY = 0)



44.8.8 Loop Mode

The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in the SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

44.8.9 Interrupt

Most bits in the SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing the Interrupt Enable Register (SSC_IER) and Interrupt Disable Register (SSC_IDR). These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in the Interrupt Mask Register (SSC_IMR), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

46.7.1 USART Control Register

Name: US_CR
Offset: 0x0000
Property: Write-only

For SPI control, see “USART Control Register (SPI_MODE)”.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								
			LINWKUP	LINABT	RTSDIS	RTSEN	DTRDIS	DTREN

Bit	15	14	13	12	11	10	9	8
Access								
Reset								
	RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

Bit	7	6	5	4	3	2	1	0
Access								
Reset								
	TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX		

Bit 21 – LINWKUP Send LIN Wakeup Signal

Value	Description
0	No effect.
1	Sends a wakeup signal on the LIN bus.

Bit 20 – LINABT Abort LIN Transmission

Value	Description
0	No effect.
1	Abort the current LIN transmission.

Bit 19 – RTSDIS Request to Send Pin Control

Value	Description
0	No effect.
1	Drives RTS pin to 0 if US_MR.USART_MODE field = 2, else drives RTS pin to 1 if US_MR.USART_MODE field = 0.

Bit 18 – RTSEN Request to Send Pin Control

Bit 21 – DSR Image of DSR Input

Value	Description
0	DSR input is driven low.
1	DSR input is driven high.

Bit 20 – RI Image of RI Input

Value	Description
0	RI input is driven low.
1	RI input is driven high.

Bit 19 – CTSIC Clear to Send Input Change Flag (cleared on read)

Value	Description
0	No input change has been detected on the CTS pin since the last read of US_CSR.
1	At least one input change has been detected on the CTS pin since the last read of US_CSR.

Bit 18 – DCDIC Data Carrier Detect Input Change Flag (cleared on read)

Value	Description
0	No input change has been detected on the DCD pin since the last read of US_CSR.
1	At least one input change has been detected on the DCD pin since the last read of US_CSR.

Bit 17 – DSRIC Data Set Ready Input Change Flag (cleared on read)

Value	Description
0	No input change has been detected on the DSR pin since the last read of US_CSR.
1	At least one input change has been detected on the DSR pin since the last read of US_CSR.

Bit 16 – RIIC Ring Indicator Input Change Flag (cleared on read)

Value	Description
0	No input change has been detected on the RI pin since the last read of US_CSR.
1	At least one input change has been detected on the RI pin since the last read of US_CSR.

Bit 13 – NACK Non Acknowledge Interrupt (cleared by writing a one to bit US_CR.RSTNACK)

Value	Description
0	Non acknowledge has not been detected since the last RSTNACK.
1	At least one non acknowledge has been detected since the last RSTNACK.

Bit 10 – ITER Max Number of Repetitions Reached (cleared by writing a one to bit US_CR.RSTIT)

Value	Description
0	Maximum number of repetitions has not been reached since the last RSTIT.
1	Maximum number of repetitions has been reached since the last RSTIT.

Bit 9 – TXEMPTY Transmitter Empty (cleared by writing US_THR)

Value (see Note)	RxStatus	Description
		channel format or was out of sequence. Only allowed on control and asynchronous channels.
74h... 7Eh	rsvd	Reserved
System Responses (Rx Device response in System Channel):		
00h	DeviceNotPresent	
80h	DevicePresent	
82h	DeviceServiceRequest	Device response to DeviceAddress scan (MLBScan), where the scanned Device needs some or all its ChannelAddresses configured.
84h...FEh	rsvd	Reserved

Note: All odd values (LSB set) are reserved.

48.6.1.5 System Commands

The Controller sends out System commands in the physical channel associated with the FRAMESYNC MediaLB frame alignment ChannelAddress (PC0). The NoData command indicates no command exists on the System Channel for this frame. All System commands are optional and may or may not be implemented on the MediaLB Controller. Additionally, System responses (including dynamic configuration) are optional and may or may not be implemented on a specific MediaLB Device.

The MOSTLock and MOSTUnlock commands indicate the status of the Controller relative to the MOST Network. When the Controller is not locked to the MOST Network (MOSTUnlock), all MediaLB data being transferred to or from the MOST Network must also stop. Buffers in the Controller could delay the stopping point to beyond when MOSTUnlock shows up on MediaLB.

The MLBReset command is designed to place the MediaLB interface in one or all Devices in a known state. When a MediaLB Device receives the MLBReset command, it will look at the corresponding first two received (most significant) data bytes on the MLBD line:

- If the first two bytes are zero, then all MediaLB Devices must reset their MediaLB interface to an initialized known state (broadcast reset to all Devices).
- If the first two bytes match the local DeviceAddress, then only the Device with the matching DeviceAddress will reset its MediaLB interface to an initialized known state (reset targeted to only one Device).

The MLBSubCmd command is used for configuration and status information from the Controller to Devices. A sub-command is contained in the first byte of the MLBD quadlet. When MediaLB Device interfaces receive the MLBSubCmd command, they will store the command and corresponding data quadlet (sub-command). Currently, only one sub-command is defined (scSetCA) and is used in dynamic configuration.

MediaLB Devices and ChannelAddresses can be configured using two methods: static or dynamic. When the EHC MediaLB Device uses the dynamic method, it instructs the Controller to scan for other MediaLB

Bits 21:16 – F1PI[5:0] Receive FIFO 1 Put Index
Receive FIFO 1 write index pointer, range 0 to 63.

Bits 13:8 – F1GI[5:0] Receive FIFO 1 Get Index
Receive FIFO 1 read index pointer, range 0 to 63.

Bits 6:0 – F1FL[6:0] Receive FIFO 1 Fill Level
Number of elements stored in Receive FIFO 1, range 0 to 64.