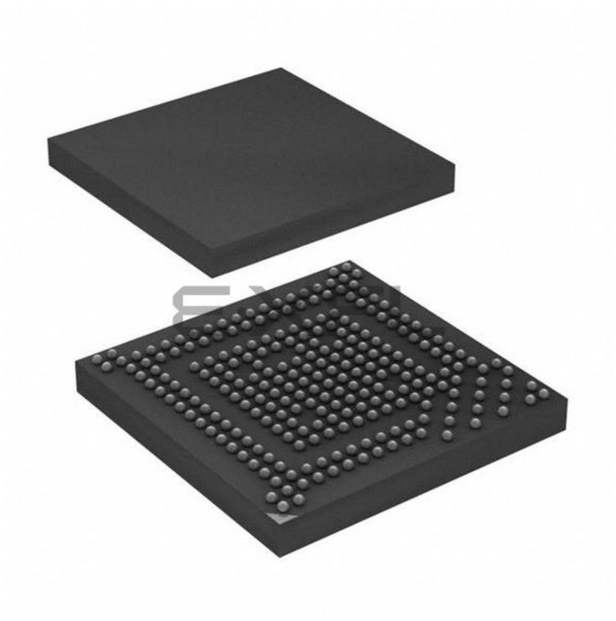


Welcome to E-XFL.COM

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	Obsolete
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	400MHz
Co-Processors/DSP	-
RAM Controllers	LPDDR, LPDDR2, DDR2, SDR, SRAM
Graphics Acceleration	No
Display & Interface Controllers	LCD, Touchscreen
Ethernet	-
SATA	-
USB	USB 2.0 (2)
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/at91sam9cn12-cfur

The multi-master bus architecture has a number of benefits:

- It allows the development of multi-master systems with an increased bus bandwidth and a flexible architecture.
- Each AHB layer becomes simple because it only has one master, so no arbitration or master-to-slave muxing is required. AHB layers, implementing AHB-Lite protocol, do not have to support request and grant, nor do they have to support retry and split transactions.
- The arbitration becomes effective when more than one master wants to access the same slave simultaneously.

8.8.1 Supported Transfers

The Arm926EJ-S processor performs all AHB accesses as single word, bursts of four words, or bursts of eight words. Any Arm9EJ-S core request that is not 1, 4, 8 words in size is split into packets of these sizes. Note that the Microchip bus is AHB-Lite protocol compliant, hence it does not support split and retry requests.

Table 8-7 gives an overview of the supported transfers and different kinds of transactions they are used for.

Table 8-7: Supported Transfers

HBurst[2:0]	Description	
SINGLE	Single transfer	Single transfer of word, half word, or byte: ● data write (NCNB, NCB, WT, or WB that has missed in DCache) ● data read (NCNB or NCB) ● NC instruction fetch (prefetched and non-prefetched) ● page table walk read
INCR4	Four-word incrementing burst	Half-line cache write-back, Instruction prefetch, if enabled. Four-word burst NCNB, NCB, WT, or WB write.
INCR8	Eight-word incrementing burst	Full-line cache write-back, eight-word burst NCNB, NCB, WT, or WB write.
WRAP8	Eight-word wrapping burst	Cache linefill

8.8.2 Thumb Instruction Fetches

All instructions fetches, regardless of the state of Arm9EJ-S core, are made as 32-bit accesses on the AHB. If the Arm9EJ-S is in Thumb state, then two instructions can be fetched at a time.

8.8.3 Address Alignment

The Arm926EJ-S BIU performs address alignment checking and aligns AHB addresses to the necessary boundary. 16-bit accesses are aligned to halfword boundaries, and 32-bit accesses are aligned to word boundaries.

Before performing the jump to the application in internal SRAM, all the PIOs and peripherals used in the boot program are set to their reset state.

Table 11-4: PIO Driven during Boot Program Execution

NVM Bootloader	Peripheral	Pin	PIO Line
NAND	EBI CS3 SMC	NANDOE	PIOD0
	EBI CS3 SMC	NANDWE	PIOD1
	EBI CS3 SMC	NANDCS	PIOD4
	EBI CS3 SMC	NAND ALE	A21
	EBI CS3 SMC	NAND CLE	A22
	EBI CS3 SMC	Cmd/Addr/Data	D[16:0]
SD Card	MCI0	MCI0_CK	PIOA17
	MCI0	MCI0_D0	PIOA15
	MCI0	MCI0_D1	PIOA18
	MCI0	MCI0_D2	PIOA19
	MCI0	MCI0_D3	PIOA20
SPI Flash	SPI0	MOSI	PIOA10
	SPI0	MISO	PIOA11
	SPI0	SPCK	PIOA13
	SPI0	NPCS0	PIOA14
	SPI0	NPCS1	PIOA7
TWI0 EEPROM	TWI0	TWD0	PIOA30
	TWI0	TWCK0	PIOA31
SAM-BA Monitor	DBGU	DRXD	PIOA9
	DBGU	DTXD	PIOA10

11.2.5 SAM-BA Monitor

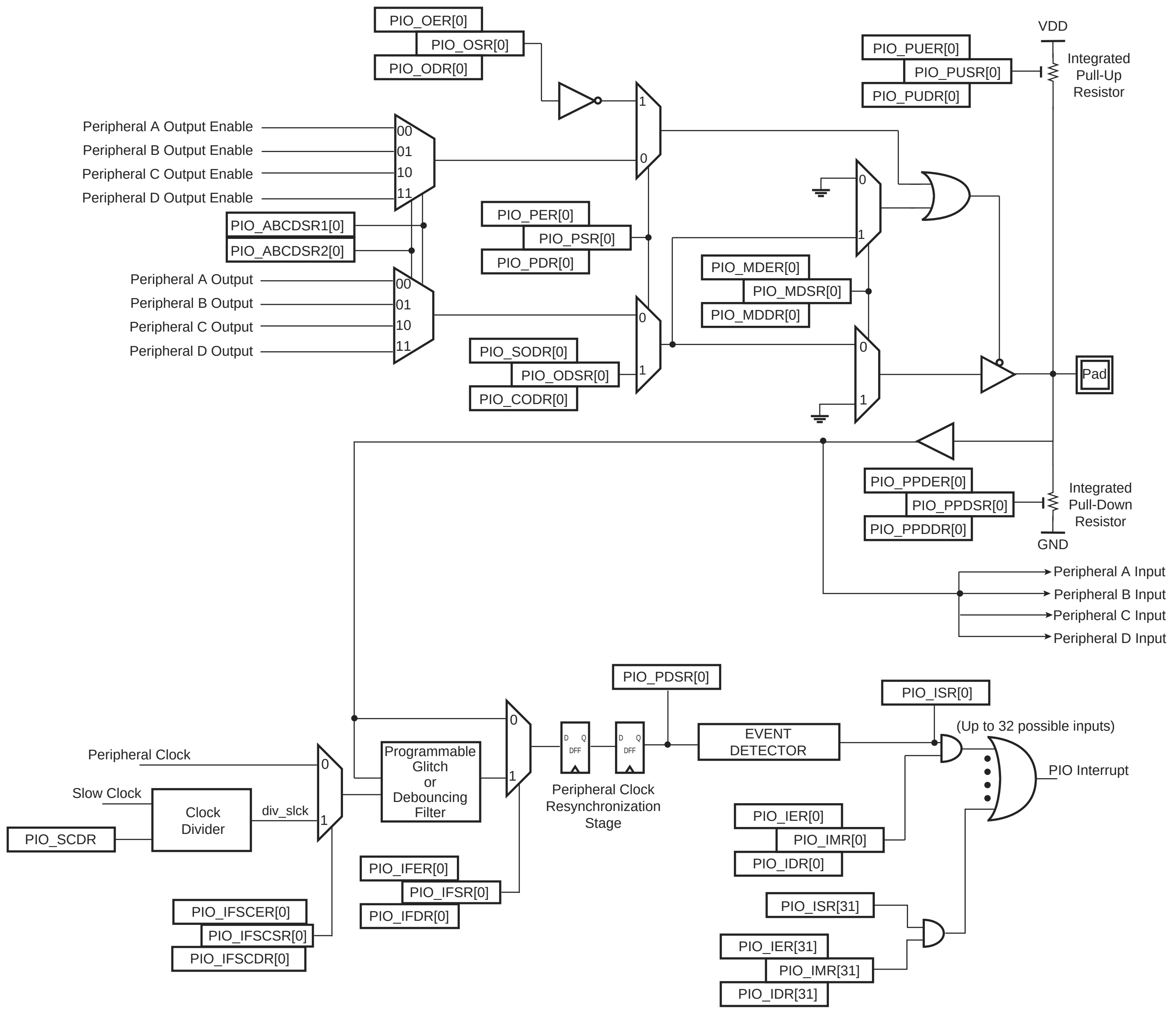
If no valid code has been found in NVM during the NVM bootloader sequence, the SAM-BA Monitor program is launched.

The SAM-BA Monitor principle is to:

- Initialize DBGU and USB
- Check if USB Device enumeration has occurred
- Check if characters have been received on the DBGU

Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as listed in Table 11-5.

Figure 22-2: I/O Line Control Logic



22.5.1 Pull-up and Pull-down Resistor Control

Each I/O line is designed with an embedded pull-up resistor and an embedded pull-down resistor. The pull-up resistor can be enabled or disabled by writing to the Pull-up Enable Register (PIO_PUER) or Pull-up Disable Register (PIO_PUDR), respectively. Writing to these registers results in setting or clearing the corresponding bit in the Pull-up Status Register (PIO_PUSR). Reading a one in PIO_PUSR means the pull-up is disabled and reading a zero means the pull-up is enabled. The pull-down resistor can be enabled or disabled by writing the Pull-down Enable Register (PIO_PPDER) or the Pull-down Disable Register (PIO_PPDDR), respectively. Writing in these registers results in setting or clearing the corresponding bit in the Pull-down Status Register (PIO_PPDSR). Reading a one in PIO_PPDSR means the pull-up is disabled and reading a zero means the pull-down is enabled.

Enabling the pull-down resistor while the pull-up resistor is still enabled is not possible. In this case, the write of PIO_PPDER for the relevant I/O line is discarded. Likewise, enabling the pull-up resistor while the pull-down resistor is still enabled is not possible. In this case, the write of PIO_PUER for the relevant I/O line is discarded.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, depending on the I/O, pull-up or pull-down can be set.

22.6.47 PIO Schmitt Trigger Register

Name:PIO_SCHMITT

Address:0xFFFFF500 (PIOA), 0xFFFFF700 (PIOB), 0xFFFFF900 (PIOC), 0xFFFFFB00 (PIOD)

Access: Read/Write

31	30	29	28	27	26	25	24
SCHMITT31	SCHMITT30	SCHMITT29	SCHMITT28	SCHMITT27	SCHMITT26	SCHMITT25	SCHMITT24
23	22	21	20	19	18	17	16
SCHMITT23	SCHMITT22	SCHMITT21	SCHMITT20	SCHMITT19	SCHMITT18	SCHMITT17	SCHMITT16
15	14	13	12	11	10	9	8
SCHMITT15	SCHMITT14	SCHMITT13	SCHMITT12	SCHMITT11	SCHMITT10	SCHMITT9	SCHMITT8
7	6	5	4	3	2	1	0
SCHMITT7	SCHMITT6	SCHMITT5	SCHMITT4	SCHMITT3	SCHMITT2	SCHMITT1	SCHMITT0

SCHMITTx [x=0..31]: Schmitt Trigger Control

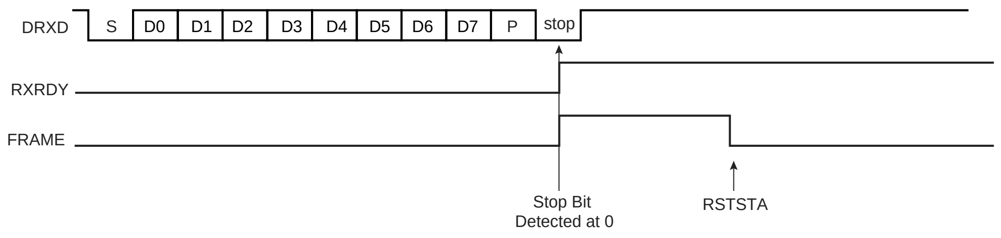
0: Schmitt trigger is enabled.

1: Schmitt trigger is disabled.

23.5.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time as the RXRDY bit is set. The bit FRAME remains high until a one is written to the RSTSTA bit in the DBGU_CR.

Figure 23-9: Receiver Framing Error



23.5.3 Transmitter

23.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing a one to the TXEN bit in DBGU_CR. From this command, the transmitter waits for a character to be written in the Transmit Holding register (DBGU_THR) before actually starting the transmission.

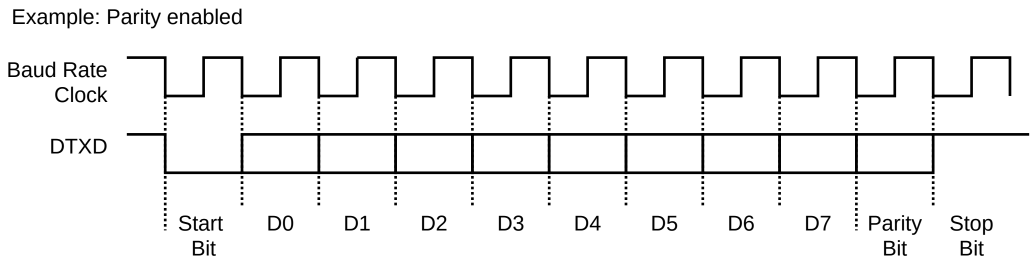
The programmer can disable the transmitter by writing a one to the TXDIS bit in the DBGU_CR. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing a one to the RSTTX bit in the DBGU_CR. This immediately stops the transmitter, whether or not it is processing characters.

23.5.3.2 Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in DBGU_MR and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field PARE in DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 23-10: Character Transmission



23.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in DBGU_SR. The transmission starts when the programmer writes in DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

23.6.4 Debug Unit Interrupt Disable Register

Name:DBGU_IDR

Address:0xFFFFF20C

Access: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

RXRDY: Disable RXRDY Interrupt

TXRDY: Disable TXRDY Interrupt

OVRE: Disable Overrun Error Interrupt

FRAME: Disable Framing Error Interrupt

PARE: Disable Parity Error Interrupt

TXEMPTY: Disable TXEMPTY Interrupt

COMMTX: Disable COMMTX (from Arm) Interrupt

COMMRX: Disable COMMRX (from Arm) Interrupt

0: No effect.

1: Disables the corresponding interrupt.

23.6.7 Debug Unit Receive Holding Register

Name: DBGU_RHR

Address: 0xFFFFF218

Access: Read-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**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCHR							

RXCHR: Received Character

Last received character if RXRDY is set.

23.6.8 Debug Unit Transmit Holding Register

Name:DBGU_THR

Address:0xFFFFF21C

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**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCHR							

TXCHR: Character to be Transmitted

Next character to be transmitted after the current character if TXRDY is not set.

24.5.4 Fuse Data Register

Name: FUSE_DR

Address: 0xFFFFDC0C

Access: Read/Write

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

DATA: Data to Program

Data to program. Only bits of with a value of “1” will be programmed.

32. USB Device Port (UDP)

32.1 Description

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) 2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1 Mbyte/s) by working with endpoints with two banks of DPR.

Table 32-1: USB Endpoint Description

Endpoint No.	Mnemonic	Dual-Bank(1)	Max. Endpoint Size	Endpoint Type
0	EP0	No	64	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	512	Bulk/Iso/Interrupt
5	EP5	Yes	512	Bulk/Iso/Interrupt

Note 1: The Dual-Bank function provides two banks for an endpoint. This feature is used for ping-pong mode.

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wakeup request to the USB host controller.

32.2 Embedded Characteristics

- USB 2.0 Full-speed Compliant, 12 Mbit/s
- Embedded USB 2.0 Full-speed Transceiver
- Integrated Pull-up on DDP
- 6 Endpoints
- Embedded Dual-port RAM for Endpoints
- Suspend/Resume Logic
- Ping-pong Mode (2 Memory Banks) for Isochronous and Bulk Endpoints

32.7.13 UDP Transceiver Control Register

Name:UDP_TXVC

Address:0xF803C074

Access: Read/Write

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

WARNING: The UDP peripheral clock in the PMC must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

TXVDIS: Transceiver Disable

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS bit.

To enable the transceiver, TXVDIS must be cleared.

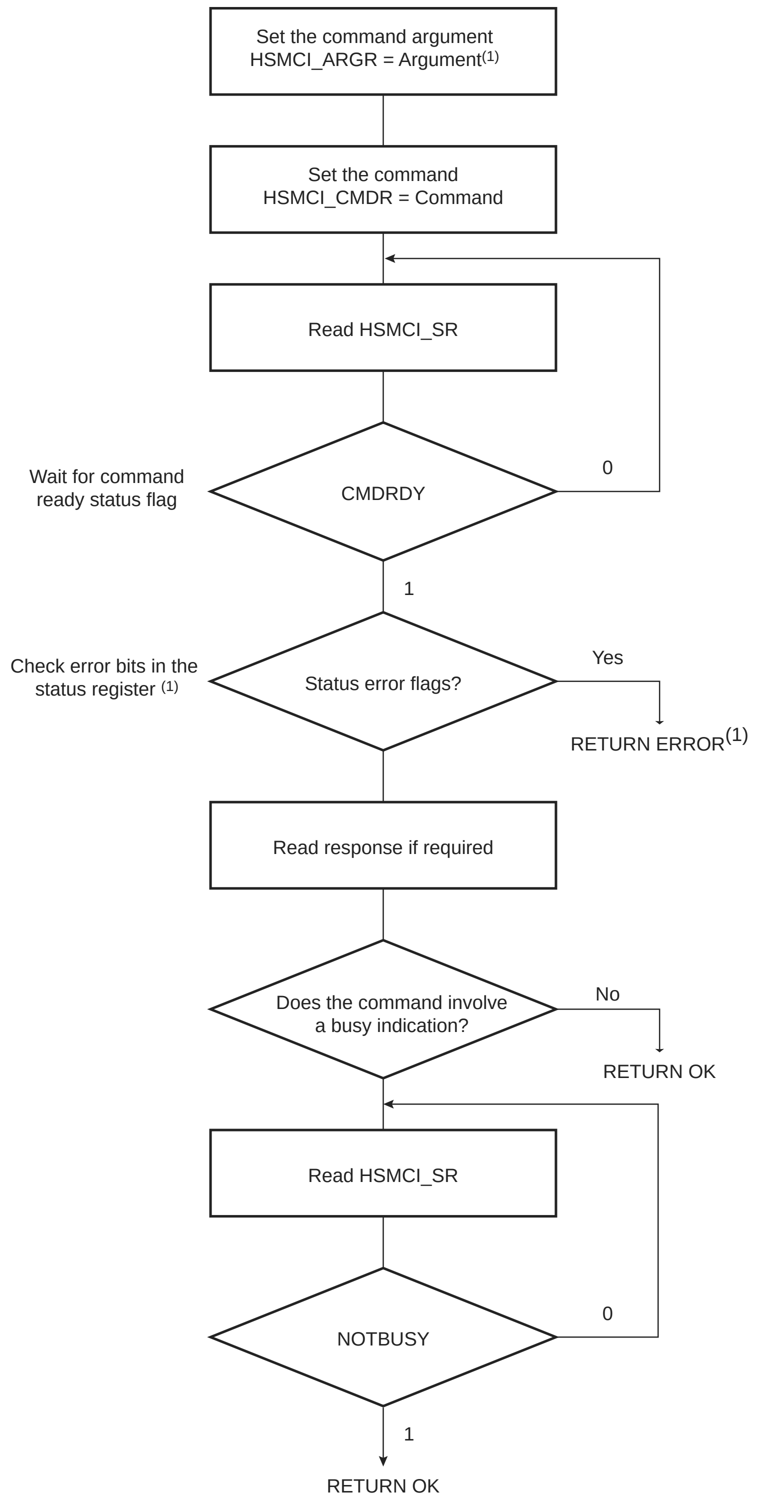
PUON: Pull-up On

0: The 1.5KΩ integrated pull-up on DDP is disconnected.

1: The 1.5 KΩ integrated pull-up on DDP is connected.

NOTE: If the USB pull-up is not connected on DDP, the user should not write in any UDP register other than the UDP_TXVC register. This is because if DDP and DDM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

Figure 34-7: Command/Response Functional Flow Diagram



Note: If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the High Speed MultiMedia Card specification).

34.9 SD/SDIO Card Operation

The High Speed MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the MultiMedia Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the High Speed MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the High Speed MultiMedia Card is the initialization process.

The SD/SDIO Card Register (HSMCI_SDCR) allows selection of the Card Slot and the data bus width.

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

34.9.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The TRTYP field in the HSMCI Command Register (HSMCI_CMDR) allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the HSMCI Block Register (HSMCI_BLKR). In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field (IOSPCMD) in the HSMCI Command Register.

34.9.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled through the HSMCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

34.10 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, 8-bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

34.10.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8 KB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are configured to 0.

35.6.3 Interrupt

The SPI interface has an interrupt line connected to the interrupt controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

Table 35-3: Peripheral IDs

Instance	ID
SPI0	13
SPI1	14

35.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to the relevant section.

35.7 Functional Description

35.7.1 Modes of Operation

The SPI operates in Master mode or in Slave mode.

- The SPI operates in Master mode by setting the MSTR bit in the SPI Mode Register (SPI_MR):
  - Pins NPCS0 to NPCS3 are all configured as outputs
  - The SPCK pin is driven
  - The MISO line is wired on the receiver input
  - The MOSI line is driven as an output by the transmitter.
- The SPI operates in Slave mode if the MSTR bit in the SPI_MR is written to 0:
  - The MISO line is driven by the transmitter output
  - The MOSI line is wired on the receiver input
  - 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)
  - NPCS1 to NPCS3 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.

35.7.2 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.

Table 35-4 shows the four modes and corresponding parameter settings.

Table 35-4: 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

Figure 35-3 and Figure 35-4 show examples of data transfers.

38.6 Product Dependencies

38.6.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor. When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the user must program the PIO Controller to dedicate TWD and TWCK as peripheral lines.

The user must not program TWD and TWCK as open-drain. This is already done by the hardware.

Table 38-4: I/O Lines

Instance	Signal	I/O Line	Peripheral
TWI0	TWCK0	PA31	A
TWI0	TWD0	PA30	A
TWI1	TWCK1	PC1	C
TWI1	TWD1	PC0	C

38.6.2 Power Management

The TWI may be clocked through the Power Management Controller (PMC), thus the user must first configure the PMC to enable the TWI clock.

38.6.3 Interrupt Sources

The TWI has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWI.

Table 38-5: Peripheral IDs

Instance	ID
TWI0	9
TWI1	10

38.7 Functional Description

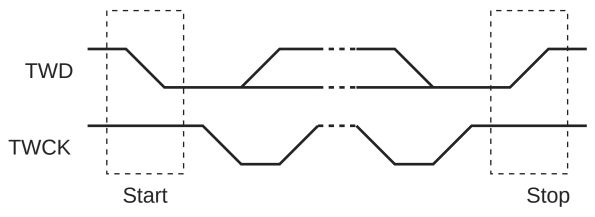
38.7.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 38-3).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 38-2).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines the STOP condition.

Figure 38-2: START and STOP Conditions



39.5 Product Dependencies

39.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

Table 39-2: I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PA3	A
USART0	RTS0	PA2	A
USART0	RXD0	PA1	A
USART0	SCK0	PA4	A
USART0	TXD0	PA0	A
USART1	CTS1	PC28	C
USART1	RTS1	PC27	C
USART1	RXD1	PA6	A
USART1	SCK1	PC29	C
USART1	TXD1	PA5	A
USART2	CTS2	PB1	B
USART2	RTS2	PB0	B
USART2	RXD2	PA8	A
USART2	SCK2	PB2	B
USART2	TXD2	PA7	A
USART3	CTS3	PC25	B
USART3	RTS3	PC24	B
USART3	RXD3	PC23	B
USART3	SCK3	PC26	B
USART3	TXD3	PC22	B

39.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

39.5.3 Interrupt Sources

The USART interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the USART interrupt requires the Interrupt Controller to be programmed first.

Table 39-3: Peripheral IDs

Instance	ID
USART0	5

39.7.25 USART Manchester Configuration Register

Name: US_MAN

Address: 0xF801C050 (0), 0xF8020050 (1), 0xF8024050 (2), 0xF8028050 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	ONE	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

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

TX_PL: Transmitter Preamble Length

0: The transmitter preamble pattern generation is disabled

1–15: The preamble length is TX_PL × Bit Period

TX_PP: Transmitter Preamble Pattern

The following values assume that TX_MPOL field is not set:

Value	Name	Description
0	ALL_ONE	The preamble is composed of ‘1’s
1	ALL_ZERO	The preamble is composed of ‘0’s
2	ZERO_ONE	The preamble is composed of ‘01’s
3	ONE_ZERO	The preamble is composed of ‘10’s

TX_MPOL: Transmitter Manchester Polarity

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

RX_PL: Receiver Preamble Length

0: The receiver preamble pattern detection is disabled

1–15: The detected preamble length is RX_PL × Bit Period

RX_PP: Receiver Preamble Pattern detected

The following values assume that RX_MPOL field is not set:

Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s
01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

44.5.8 AES Input Data Register x

Name: AES_IDATARx [x=0..3]

Address:0xF000C040

Access: Write-only

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

IDATA: Input Data Word

The four 32-bit Input Data registers set the 128-bit data block used for encryption/decryption.

AES_IDATAR0 corresponds to the first word of the data to be encrypted/decrypted, and AES_IDATAR3 to the last one.

These registers are write-only to prevent the input data from being read by another application.

46.6.6 TRNG Output Data Register

Name: TRNG_ODATA

Address:0xF8048050

Access: Read-only

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

ODATA: Output Data

The 32-bit Output Data register contains the 32-bit random data.

Doc. Rev 11063J	Comments (Continued)	Change Request Ref.(1)
	EBI: Added titles to figures in Section 26.7.4 “Power Supplies” and Section 26.8 “Implementation Examples”. Added Figure 26-13 "16-bit NAND Flash with NFD0_ON_D16 = 1".	rfo
	PMECC: Section 27.2 ”Embedded Characteristics”, added a line on supporting 8-bit Nand Flash data bus. Section 27.6.11 ”PMECC Interrupt Status Register”, fixed a typo in the register table: - replaced duplicate bits 31 - 24 by missing 7 - 0	8403
	SMC: Replaced ‘turned out’ with ‘switched to output mode ‘ in Section 29.9.4.1 ”Write is Controlled by NWE (WRITE_MODE = 1)” and Section 29.9.4.2 ”Write is Controlled by NCS (WRITE_MODE = 0)”.	7925
	DDRSDRC: Replaced TCSR with TCR when related to Section 30.7.7 ”DDRSDRC Low-power Register” to prevent confusion between JEDEC naming TCSR and the associated bitfield TCR. Replaced binary configuration values with decimal values throughout the document. Section 30.7 ”DDR SDR SDRAM Controller (DDRSDRC) User Interface”: - Table 30-16 ”Register Mapping”, updated offset values for reserved registers (0x54-0xE0, 0xEC-0xFC) - Section 30.7.3 ”DDRSDRC Configuration Register”: - DIC: updated the bitfield name from DIC/DIS to DIC and revised the description content - Section 30.7.11 ”DDRSDRC Write Protect Mode Register”: - WPKEY: replaced the bitfield description with a table - described with a table and/or updated data presentation in: - Section 30.7.1 ”DDRSDRC Mode Register” (MODE) - Section 30.7.3 ”DDRSDRC Configuration Register” (NC, NR, OCD, NB, DECOD) - Section 30.7.4 ”DDRSDRC Timing Parameter 0 Register” (TWTR) - Section 30.7.7 ”DDRSDRC Low-power Register”(LPCB, TIMEOUT, APDE, UPD_MR) - Section 30.7.8 ”DDRSDRC Memory Device Register” (MD, DBW)	8592 rfo 8968 8592 8883 8883
	DMAC: Added Section 31.3 ”DMA Controller Peripheral Connections” and moved Table 31-1 ”DMA Channel Definition” from Section 31.2 ”Embedded Characteristics” to this new section. Section 31.8.1 ”DMAC Global Configuration Register”: - KEY: replaced the bitfield description with a table Section 31.8.15 ”DMAC Channel x [x = 0..7] Descriptor Address Register”: - DSCR_IF bitfield description table: added DMA Master Interface references Section 31.8.17 ”DMAC Channel x [x = 0..7] Control B Register”: - SIF bitfield description table: added DMA Master Interface references - DIF bitfield description table: added DMA Master Interface references Section 31.8.21 ”DMAC Write Protect Mode Register”: - WPKEY: replaced the bitfield description with a table	8955 8835 rfo 8834