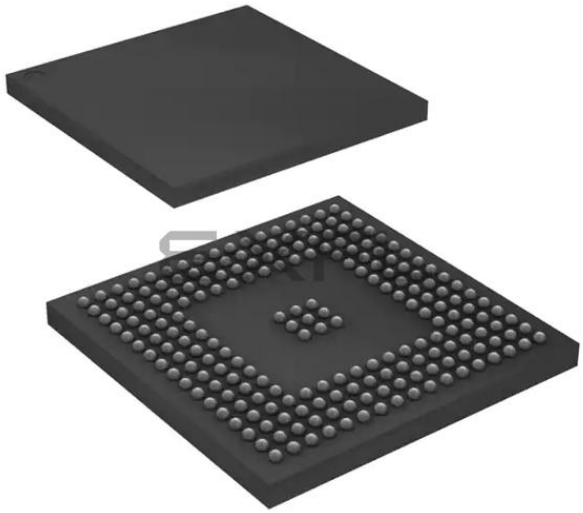


Welcome to E-XFL.COM

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	LCD, Touchscreen
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	217-LFBGA
Supplier Device Package	217-LFBGA (15x15)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91sam9g35-cu

Exception Modes and Handling

Exceptions arise whenever the normal flow of a program must be halted temporarily, for example, to service an interrupt from a peripheral.

When handling an ARM exception, the ARM9EJ-S core performs the following operations:

1. Preserves the address of the next instruction in the appropriate Link Register that corresponds to the new mode that has been entered. When the exception entry is from:
   - ARM and Jazelle states, the ARM9EJ-S copies the address of the next instruction into LR (current PC(r15) + 4 or PC + 8 depending on the exception).
   - THUMB state, the ARM9EJ-S writes the value of the PC into LR, offset by a value (current PC + 2, PC + 4 or PC + 8 depending on the exception) that causes the program to resume from the correct place on return.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value that depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The register r13 is also banked across exception modes to provide each exception handler with private stack pointer.

The ARM9EJ-S can also set the interrupt disable flags to prevent otherwise unmanageable nesting of exceptions.

When an exception has completed, the exception handler must move both the return value in the banked LR minus an offset to the PC and the SPSR to the CPSR. The offset value varies according to the type of exception. This action restores both PC and the CPSR.

The fast interrupt mode has seven private registers r8 to r14 (banked registers) to reduce or remove the requirement for register saving which minimizes the overhead of context switching.

The Prefetch Abort is one of the aborts that indicates that the current memory access cannot be completed. When a Prefetch Abort occurs, the ARM9EJ-S marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the Execute stage in the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

The breakpoint (BKPT) instruction is a new feature of ARM9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.
A breakpoint instruction does not cause the ARM9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

8.4.8 ARM Instruction Set Overview

The ARM instruction set is divided into:

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load and Store instructions
- Coprocessor instructions
- Exception-generating instructions

ARM instructions can be executed conditionally. Every instruction contains a 4-bit condition code field (bits[31:28]).

For further details, see the ARM Technical Reference Manual.

8.4.9 New ARM Instruction Set

Table 8-3. New ARM Instruction Mnemonic List

Mnemonic	Operation
BXJ	Branch and exchange to Java
BLX (1)	Branch, Link and exchange
SMLAxy	Signed Multiply Accumulate 16 * 16 bit
SMLAL	Signed Multiply Accumulate Long
SMLAWy	Signed Multiply Accumulate 32 * 16 bit
SMULxy	Signed Multiply 16 * 16 bit
SMULWy	Signed Multiply 32 * 16 bit
QADD	Saturated Add
QDADD	Saturated Add with Double
QSUB	Saturated subtract
QDSUB	Saturated Subtract with double

Mnemonic	Operation
MRRC	Move double from coprocessor
MCR2	Alternative move of ARM reg to coprocessor
MCRR	Move double to coprocessor
CDP2	Alternative Coprocessor Data Processing
BKPT	Breakpoint
PLD	Soft Preload, Memory prepare to load from address
STRD	Store Double
STC2	Alternative Store from Coprocessor
LDRD	Load Double
LDC2	Alternative Load to Coprocessor
CLZ	Count Leading Zeroes

Notes: 1. A Thumb BLX contains two consecutive Thumb instructions, and takes four cycles.

8.4.10 Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the ARM instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store multiple instructions
- Exception-generating instruction

For further details, see the ARM Technical Reference Manual.

Table 8-4 gives the Thumb instruction mnemonic list.

10.5 SAM-BA Monitor

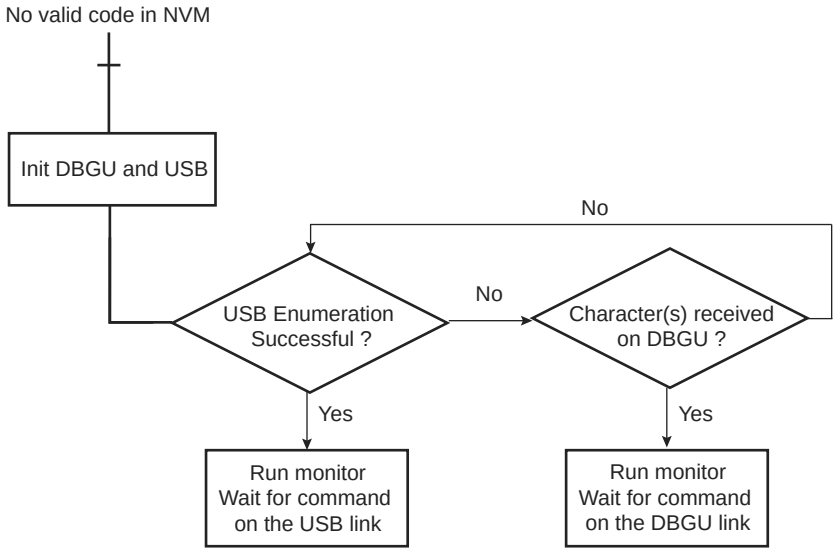
If no valid code has been found in NVM during the NVM bootloader sequence, the SAM-BA Monitor program is launched.

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 10-5.

Figure 10-10. SAM-BA Monitor Diagram



10.5.1 Command List

Table 10-5. Commands Available Through the SAM-BA Monitor

Command	Action	Argument(s)	Example
N	set Normal mode	No argument	**N**#
T	set Terminal mode	No argument	**T**#
O	write a byte	Address, Value#	**O**200001,CA#
o	read a byte	Address,#	**o**200001,#
H	write a half word	Address, Value#	**H**200002,CAFE#
h	read a half word	Address,#	**h**200002,#
W	write a word	Address, Value#	**W**200000,CAFEDECA#
w	read a word	Address,#	**w**200000,#
S	send a file	Address,#	**S**200000,#
R	receive a file	Address, NbOfBytes#	**R**200000,1234#
G	go	Address#	**G**200200#
V	display version	No argument	**V**#

17.6 Functional Description

The Shutdown Controller manages the main power supply. To do so, it is supplied with VDDBU and manages wake-up input pins and one output pin, SHDN.

A typical application connects the pin SHDN to the shutdown input of the DC/DC Converter providing the main power supplies of the system, and especially VDDCORE and/or VDDIO. The wake-up inputs (WKUP0) connect to any push-buttons or signal that wake up the system.

The software is able to control the pin SHDN by writing the Shutdown Control Register (SHDW_CR) with the bit SHDW at 1. The shutdown is taken into account only two slow clock cycles after the write of SHDW_CR. This register is password-protected and so the value written should contain the correct key for the command to be taken into account. As a result, the system should be powered down.

17.6.1 Wake-up Inputs

A level change on WKUP0 is used as a wake-up. Wake-up is configured in the Shutdown Mode Register (SHDW_MR). The transition detector can be programmed to detect either a positive or negative transition or any level change on WKUP0. The detection can also be disabled. Programming is performed by defining WKMODE0

Moreover, a debouncing circuit can be programmed for WKUP0. The debouncing circuit filters pulses on WKUP0 shorter than the programmed number of 16 SLCK cycles in CPTWK0 of the SHDW_MR. If the programmed level change is detected on a pin, a counter starts. When the counter reaches the value programmed in the corresponding field, CPTWK0, the SHDN pin is released. If a new input change is detected before the counter reaches the corresponding value, the counter is stopped and cleared. WAKEUP0 of the Status Register (SHDW_SR) reports the detection of the programmed events on WKUP0 with a reset after the read of SHDW_SR.

The Shutdown Controller can be programmed so as to activate the wake-up using the RTC alarm (the detection of the rising edge of the RTC alarm is synchronized with SLCK). This is done by writing the SHDW_MR using the RTCWKEN field. When enabled, the detection of RTC alarm is reported in the RTCWK bit of the SHDW_SR. They are reset after the read of SHDW_SR. When using the RTC alarm to wake up the system, the user must ensure that RTC alarm status flag is cleared before shutting down the system. Otherwise, no rising edge of the status flags may be detected and the wake-up will fail.

22.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Address: 0xFFFFF454 (PIOA), 0xFFFFF654 (PIOB), 0xFFFFF854 (PIOC), 0xFFFFFA54 (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

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

- **P0–P31: Multi-drive Disable**

0: No effect.

1: Disables multi-drive on the I/O line.

27.5.5 Error Location Status Register

Name: PMERRLOC_ELSR

Address: 0xFFFFE610

Access: Read-write

Reset: 0x00000000

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
–	–	–	–	–	–	–	BUSY

- **BUSY: Error Location Engine Busy**

29.7.12 DDRSDRC Write Protection Status Register

Name: DDRSDRC_WPSR

Address: 0xFFFFE8E8

Access: Read-only

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the DDRSDRC_WPSR.

1: A write protection violation has occurred since the last read of the DDRSDRC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading DDRSDRC_WPSR automatically clears all fields.

30.6.2 Memory Peripherals

Figure 30-3 on page 474 shows the DMAC transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level handshaking interface is to allow the DMAC to attempt AMBA transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AMBA transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the DMAC that it is ready to transmit/receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

30.6.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or chunk transfers. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer/accept data over the AMBA bus. A non-memory peripheral can request a DMAC transfer through the DMAC using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

30.6.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMAC transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMAC transaction. These software registers are used to implement the software handshaking interface.

The SRC_H2SEL/DST_H2SEL bit in the Channel Configuration Register (DMAC_CFGx) must be cleared to enable software handshaking.

When the peripheral is not the flow controller, then the Software Last Transfer Flag Register (DMAC_LAST) is not used, and the values in these registers are ignored.

Chunk Transactions

Writing a ‘1’ to the Software Chunk Transfer Request Register (DMAC_CREQ[2x]) starts a source chunk transaction request, where x is the channel number. Writing a ‘1’ to the DMAC_CREQ[2x+1] register starts a destination chunk transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_CREQ[*2x*] or DMAC_CREQ[2x+1].

Single Transactions

Writing a ‘1’ to the Software Single Request Register (DMAC_SREQ[2x]) starts a source single transaction request, where x is the channel number. Writing a ‘1’ to the DMAC_SREQ[2x+1] register starts a destination single transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_SREQ[x] or DMAC_SREQ[2x+1].

The software can poll the relevant channel bit in the DMAC_CREQ[2x]/DMAC_CREQ[2x+1] and DMAC_SREQ[x]/DMAC_SREQ[2x+1] registers. When both are 0, then either the requested chunk or single transaction has completed.

33.14.1 HSMCI Control Register

Name: HSMCI_CR

Address: 0xF0008000 (0), 0xF000C000 (1)

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
SWRST	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

- **MCIEN: Multi-Media Interface Enable**

0: No effect.

1: Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0: No effect.

1: Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0: No effect.

1: Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field of the HSMCI_MR.

- **PWSDIS: Power Save Mode Disable**

0: No effect.

1: Disables the Power Saving Mode.

- **SWRST: Software Reset**

0: No effect.

1: Resets the HSMCI. A software triggered hardware reset of the HSMCI is performed.

- **DCRCE: Data CRC Error (cleared on read)**

0: No error.

1: A CRC16 error has been detected in the last data block.

- **DTOE: Data Time-out Error (cleared on read)**

0: No error.

1: The data time-out set by DTOCYC and DTOMUL in HSMCI_DTOR has been exceeded.

- **CSTOE: Completion Signal Time-out Error (cleared on read)**

0: No error.

1: The completion signal time-out set by CSTOCYC and CSTOMUL in HSMCI_CSTOR has been exceeded.

- **BLKOVRE: DMA Block Overrun Error (cleared on read)**

0: No error.

1: A new block of data is received and the DMA controller has not started to move the current pending block, a block overrun is raised.

- **DMADONE: DMA Transfer Done (cleared on read)**

0: DMA buffer transfer has not completed since the last read of the HSMCI_SR.

1: DMA buffer transfer has completed since the last read of the HSMCI_SR.

- **FIFOEMPTY: FIFO empty flag**

0: FIFO contains at least one byte.

1: FIFO is empty.

- **XFRDONE: Transfer Done flag**

0: A transfer is in progress.

1: Command Register is ready to operate and the data bus is in the idle state.

- **ACKRCV: Boot Operation Acknowledge Received (cleared on read)**

0: No Boot acknowledge received since the last read of the HSMCI_SR.

1: A Boot acknowledge signal has been received since the last read of HSMCI_SR.

- **ACKRCVE: Boot Operation Acknowledge Error (cleared on read)**

0: No boot operation error since the last read of HSMCI_SR

1: Corrupted Boot Acknowledge signal received since the last read of HSMCI_SR.

- **OVRE: Overrun (if FERRCTRL = 1, cleared by writing in HSMCI_CMDR or cleared on read if FERRCTRL = 0)**

0: No error.

1: At least one 8-bit received data has been lost (not read).

If FERRCTRL = 1 in HSMCI_CFG, OVRE is cleared on read.

If FERRCTRL = 0 in HSMCI_CFG, OVRE is cleared by writing HSMCI_CMDR.

35.7.4 TC Stepper Motor Mode Register

Name: TC_SMMRx [x=0..2]

Address: 0xF8008008 (0)[0], 0xF8008048 (0)[1], 0xF8008088 (0)[2], 0xF800C008 (1)[0], 0xF800C048 (1)[1], 0xF800C088 (1)[2]

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DOWN	GCEN

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

- **GCEN: Gray Count Enable**

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit gray counter.

- **DOWN: Down Count**

0: Up counter.

1: Down counter.

38.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - Digital Filter on Receive Line
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Oversampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 kbit/s
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to $f_{peripheral\ clock}/6$
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 SPECIFICATIONS
  - Master or Slave
  - Processing of Frames with Up to 256 Data Bytes
  - Response Data Length can be Configurable or Defined Automatically by the Identifier
  - Self-synchronization in Slave Node Configuration
  - Automatic Processing and Verification of the “Synch Break” and the “Synch Field”
  - “Synch Break” Detection Even When Partially Superimposed with a Data Byte
  - Automatic Identifier Parity Calculation/Sending and Verification
  - Parity Sending and Verification Can be Disabled
  - Automatic Checksum Calculation/sending and Verification
  - Checksum Sending and Verification Can be Disabled
  - Support Both “Classic” and “Enhanced” Checksum Types
  - Full LIN Error Checking and Reporting
  - Frame Slot Mode: Master Allocates Slots to the Scheduled Frames Automatically
  - Generation of the Wakeup Signal
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two DMA Controller Channels (DMAC)
- Offers Buffer Transfer without Processor Intervention
- Register Write Protection

The USART cannot operate concurrently in both Receiver and Transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value.

38.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 38-30.

If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in Figure 38-31. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding register (US_RHR). It appropriately sets the PARE bit in the Status register (US_SR) so that the software can handle the error.

Figure 38-30. T = 0 Protocol without Parity Error

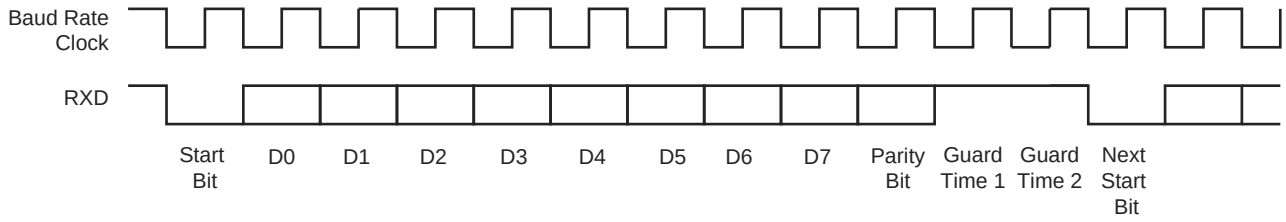
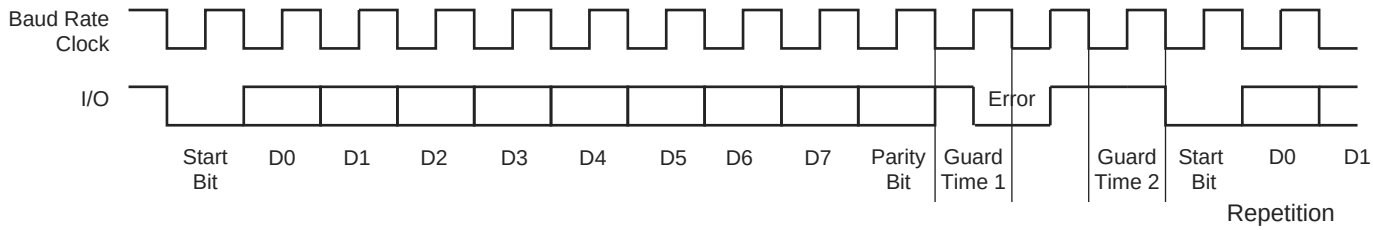


Figure 38-31. T = 0 Protocol with Parity Error



Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in US_MR. If INACK is to 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding register, as if no error occurred and the RXRDY bit does rise.

Transmit Character Repetition

38.7.9 USART Interrupt Disable Register (SPI_MODE)

Name: US_IDR (SPI_MODE)

Address: 0xF801C00C (0), 0xF802000C (1), 0xF802400C (2)

Access: Write-only

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

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

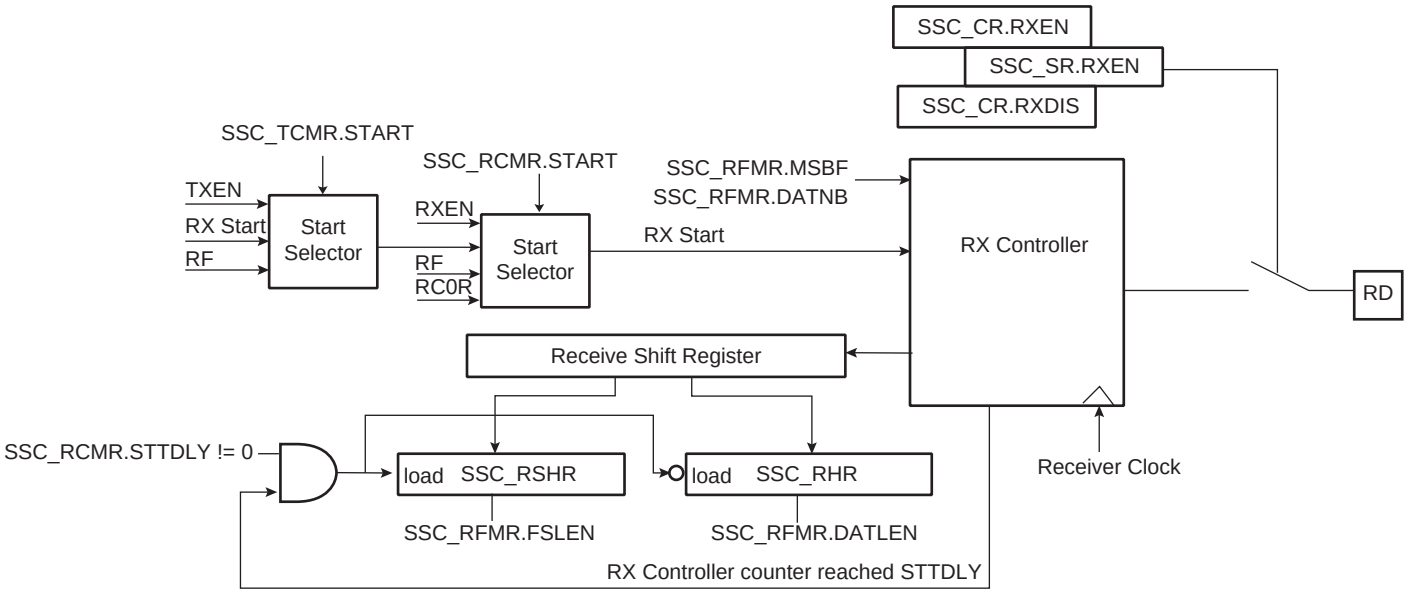
The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: SPI Underrun Error Interrupt Disable**
- **NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Disable**

Figure 42-12. Receiver Block Diagram



42.8.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (SSC_RCMR/SSC_TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (SSC_TFMR/SSC_RFMR).

Table 44-55. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00000108	Overlay 1 Channel Status Register	LCDC_OVRCHSR1	Read-only	0x00000000
0x0000010C	Overlay 1 Interrupt Enable Register	LCDC_OVRIER1	Write-only	–
0x00000110	Overlay 1 Interrupt Disable Register	LCDC_OVRIDR1	Write-only	–
0x00000114	Overlay 1 Interrupt Mask Register	LCDC_OVRIMR1	Read-only	0x00000000
0x00000118	Overlay 1 Interrupt Status Register	LCDC_OVRISR1	Read-only	0x00000000
0x0000011C	Overlay 1 DMA Head Register	LCDC_OVRHEAD1	Read/Write	0x00000000
0x00000120	Overlay 1 DMA Address Register	LCDC_OVRADDR1	Read/Write	0x00000000
0x00000124	Overlay1 DMA Control Register	LCDC_OVRCTRL1	Read/Write	0x00000000
0x00000128	Overlay1 DMA Next Register	LCDC_OVRNEXT1	Read/Write	0x00000000
0x0000012C	Overlay 1 Configuration 0 Register	LCDC_OVR1CFG0	Read/Write	0x00000000
0x00000130	Overlay 1 Configuration 1 Register	LCDC_OVR1CFG1	Read/Write	0x00000000
0x00000134	Overlay 1 Configuration 2 Register	LCDC_OVR1CFG2	Read/Write	0x00000000
0x00000138	Overlay 1 Configuration 3 Register	LCDC_OVR1CFG3	Read/Write	0x00000000
0x0000013C	Overlay 1 Configuration 4 Register	LCDC_OVR1CFG4	Read/Write	0x00000000
0x00000140	Overlay 1 Configuration 5 Register	LCDC_OVR1CFG5	Read/Write	0x00000000
0x00000144	Overlay 1 Configuration 6 Register	LCDC_OVR1CFG6	Read/Write	0x00000000
0x00000148	Overlay 1 Configuration 7 Register	LCDC_OVR1CFG7	Read/Write	0x00000000
0x0000014C	Overlay 1 Configuration 8 Register	LCDC_OVR1CFG8	Read/Write	0x00000000
0x00000150	Overlay 1 Configuration 9 Register	LCDC_OVR1CFG9	Read/Write	0x00000000
0x154–0x27C	Reserved	–	–	–
0x00000280	High End Overlay Channel Enable Register	LCDC_HEOCHER	Write-only	–
0x00000284	High End Overlay Channel Disable Register	LCDC_HEOCHDR	Write-only	–
0x00000288	High End Overlay Channel Status Register	LCDC_HEOCHSR	Read-only	0x00000000
0x0000028C	High End Overlay Interrupt Enable Register	LCDC_HEOIER	Write-only	–
0x00000290	High End Overlay Interrupt Disable Register	LCDC_HEOIDR	Write-only	–
0x00000294	High End Overlay Interrupt Mask Register	LCDC_HEOIMR	Read-only	0x00000000
0x00000298	High End Overlay Interrupt Status Register	LCDC_HEOISR	Read-only	0x00000000
0x0000029C	High End Overlay DMA Head Register	LCDC_HEOHEAD	Read/Write	0x00000000
0x000002A0	High End Overlay DMA Address Register	LCDC_HEOADDR	Read/Write	0x00000000
0x000002A4	High End Overlay DMA Control Register	LCDC_HEOCTRL	Read/Write	0x00000000
0x000002A8	High End Overlay DMA Next Register	LCDC_HEONEXT	Read/Write	0x00000000
0x000002AC	High End Overlay U DMA Head Register	LCDC_HEOUHEAD	Read/Write	0x00000000
0x000002B0	High End Overlay U DMA Address Register	LCDC_HEOUADDR	Read/Write	0x00000000
0x000002B4	High End Overlay U DMA Control Register	LCDC_HEOUCTRL	Read/Write	0x00000000
0x000002B8	High End Overlay U DMA Next Register	LCDC_HEOUNEXT	Read/Write	0x00000000
0x000002BC	High End Overlay V DMA Head Register	LCDC_HEOVHEAD	Read/Write	0x00000000
0x000002C0	High End Overlay V DMA Address Register	LCDC_HEOVADDR	Read/Write	0x00000000

44.7.45 Overlay 1 Layer Configuration 3 Register

Name: LCDC_OVR1CFG3

Address: 0xF8038138

Access: Read/Write

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

- **XSIZE: Horizontal Window Size**

Overlay 1 window width in pixels. The window width is set to (XSIZE + 1).

The following constraint must be met: XPOS + XSIZE ≤PPL

- **YSIZE: Vertical Window Size**

Overlay 1 window height in pixels. The window height is set to (YSIZE + 1).

The following constrain must be met: YPOS + YSIZE ≤RPF

44.7.51 Overlay1 Layer Configuration 9 Register

Name: LCDC_OVR1CFG9

Address: 0xF8038150

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
GA							
15	14	13	12	11	10	9	8
–	–	–	–	–	DSTKEY	REP	DMA
7	6	5	4	3	2	1	0
OVR	LAEN	GAEN	REVALPHA	ITER	ITER2BL	INV	CRKEY

- **CRKEY: Blender Chroma Key Enable**

0: Chroma key matching is disabled.

1: Chroma key matching is enabled.

- **INV: Blender Inverted Blender Output Enable**

0: Iterated pixel is the blended pixel.

1: Iterated pixel is the inverted pixel.

- **ITER2BL: Blender Iterated Color Enable**

0: Final adder stage operand is set to 0.

1: Final adder stage operand is set to the iterated pixel value.

- **ITER: Blender Use Iterated Color**

0: Pixel difference is set to 0.

1: Pixel difference is set to the iterated pixel value.

- **REVALPHA: Blender Reverse Alpha**

0: Pixel difference is multiplied by alpha.

1: Pixel difference is multiplied by 1 - alpha.

- **GAEN: Blender Global Alpha Enable**

0: Global alpha blending coefficient is disabled.

1: Global alpha blending coefficient is enabled.

- **LAEN: Blender Local Alpha Enable**

0: Local alpha blending coefficient is disabled.

1: Local alpha blending coefficient is enabled.

- **OVR: Blender Overlay Layer Enable**

0: Overlay pixel color is set to the default overlay pixel color.

1: Overlay pixel color is set to the DMA channel pixel color.

44.7.62 High End Overlay Layer Next Register

Name: LCDC_HEONEXT

Address: 0xF80382A8

Access: Read/Write

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

- **NEXT: DMA Descriptor Next Address**

DMA Descriptor next address, this address must be word aligned.

Doc. Rev. 11053F	Comments
31-Aug-15	Section 23. “Debug Unit (DBGU)”
	Instances of “Master clock” or “MCK” replaced with “Peripheral clock”
	Updated Section 23.2 “Embedded Characteristics”
	Updated Figure 23-1 “Debug Unit Functional Block Diagram”
	Section 23.6.10 “Debug Unit Chip ID Register”: changed name and description of value 0xA5 for ARCH field (was reserved; is now ATSAMA5xx / ATSAMA5xx Series)
	Section 28. “Static Memory Controller (SMC)”
	Added Table 28-3 “I/O Lines”
	Section 28.9 “Standard Read and Write Protocols”: deleted subsection “Write Protected Registers”
	Updated Section 28.9.1.3 “Read Cycle”
	Updated Section 28.9.3.3 “Write Cycle”
	Section 28.14.2 “Byte Access Type in Page Mode”: “SMC_REGISTER” corrected to “SMC Mode Register (SMC_MODE)”
	Removed section 29.15 “Programmable IO Delays”
	Added Section 28.15 “Register Write Protection”
	Table 28-9 “Register Mapping”: removed registers SMC_DELAY1–SMC_DELAY8 (offset range 0xC0–0xDC now reserved)
	Section 28.16.1 “SMC Setup Register”: added sentence about write protection
	Section 28.16.2 “SMC Pulse Register”: added sentence about write protection
	Section 28.16.3 “SMC Cycle Register”: added sentence about write protection
	Section 28.16.4 “SMC Mode Register”:
	- added sentence about write protection
	- added sentence about confirming the SMC configuration
	- updated descriptions of bits/fields READ_MODE, WRITE_MODE, EXNW_MODE, BAT, DBW, and PS
	Removed section 29.16.5 “SMC DELAY I/O Register”
	Section 28.16.5 “SMC Write Protection Mode Register”: removed “Reset” line; updated WPEN and WPKEY field descriptions
	Section 28.16.6 “SMC Write Protection Status Register”: removed “Reset” line; updated WPVS and WPVSRC field descriptions
	Section 29. “DDR SDR SDRAM Controller (DDRSDRC)”
	Removed instances of or references to “temperature compensated self refresh”, “TCR” field, and acronym “TCSR”
	Section 29.4.2 “Low-power DDR1-SDRAM Initialization”: added “Low-power” to title and modified step 6
	Section 29.5.1 “SDRAM Controller Write Cycle”: added note defining TWRD
	Figure 29-12 “Single Read Access, Row Closed, Latency = 3, DDR2-SDRAM Device”: modified diagram to add one cycle and corrected “Latency = 2” to “Latency = 3”
	Figure 29-16 “Burst Read Access, Latency = 2, SDR-SDRAM Devices”: removed DQS[1:0] waveform
	Section 29.5.4 “Power Management”: added note specifying that possible SDRAM constraint of 4K cycles of burst auto-refresh is not supported
	Updated Section 29.5.6 “Register Write Protection”
	Section 29.6.3 “SDR-SDRAM Address Mapping for 32-bit Memory Data Bus Width”: updated footnote 2
	Table 29-16 “Register Mapping”: added row for reserved offset 0x28; added row for reserved offset range 0xEC–0xFC
	Removed “Reset” line from individual register descriptions (reset values are provided in Table 29-16 “Register Mapping”)