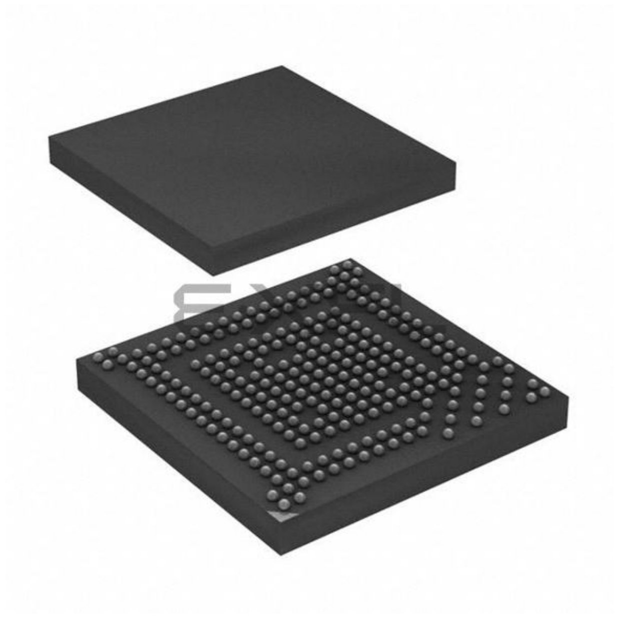
Microchip Technology - AT91SAM9G25-CFU-999 Datasheet



Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Active
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	400MHz
Co-Processors/DSP	-
RAM Controllers	DDR2, SDRAM, SRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10/100Mbps
SATA	-
USB	USB 2.0 (3)
Voltage - I/O	1.8V, 3.3V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	247-LFBGA
Supplier Device Package	247-BGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91sam9g25-cfu-999

Figure 8-2 shows the status register format, where:

- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- The Sticky Overflow (Q) flag can be set by certain multiply and fractional arithmetic instructions like QADD, QDADD, QSUB, QDSUB, SMLAxy, and SMLAWy needed to achieve DSP operations.
  The Q flag is sticky in that, when set by an instruction, it remains set until explicitly cleared by an MSR instruction writing to the CPSR. Instructions cannot execute conditionally on the status of the Q flag.
- The J bit in the CPSR indicates when the ARM9EJ-S core is in Jazelle state, where:
  - J = 0: The processor is in ARM or Thumb state, depending on the T bit
  - J = 1: The processor is in Jazelle state.
- Mode: five bits to encode the current processor mode

8.4.7.2 Exceptions

Exception Types and Priorities

The ARM9EJ-S supports five types of exceptions. Each type drives the ARM9EJ-S in a privileged mode. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Data and Prefetched aborts (Abort)
- Undefined instruction (Undefined)
- Software interrupt and Reset (Supervisor)

When an exception occurs, the banked version of R14 and the SPSR for the exception mode are used to save the state.

More than one exception can happen at a time, therefore the ARM9EJ-S takes the arisen exceptions according to the following priority order:

- Reset (highest priority)
- Data Abort
- FIQ
- IRQ
- Prefetch Abort
- BKPT, Undefined instruction, and Software Interrupt (SWI) (Lowest priority)

The BKPT, or Undefined instruction, and SWI exceptions are mutually exclusive.

Note that there is one exception in the priority scheme: when FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the ARM9EJ-S core enters the Data Abort handler, and proceeds immediately to FIQ vector. A normal return from the FIQ causes the Data Abort handler to resume execution. Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection.

- Mode commands:
  - Normal mode configures SAM-BA Monitor to send / receive data in binary format,
  - Terminal mode configures SAM-BA Monitor to send / receive data in ascii format.
- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  - *Address*: Address in hexadecimal.
  - *Value*: Byte, halfword or word to write in hexadecimal.
  - *Output*: '>'
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  - *Address*: Address in hexadecimal.
  - *Output*: The byte, halfword or word read in hexadecimal followed by '>'
- Send a file (**S**): Send a file to a specified address.
  - *Address*: Address in hexadecimal.
  - *Output*: '>'

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal.
  - *NbOfBytes*: Number of bytes in hexadecimal to receive.
  - *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code.
  - *Address*: Address to jump in hexadecimal.
  - *Output*: '>'once returned from the program execution. If the executed program does not handle the link register at its entry and does not return, the prompt will not be displayed.
- Get Version (**V**): Return the Boot Program version.
  - *Output*: version, date and time of ROM code followed by '>'.

10.5.2 DBGU Serial Port

Communication is performed through the DBGU serial port initialized to 115,200 baud, 8 bits of data, no parity, 1 stop bit.

10.5.2.1 Supported External Crystal/External Clocks

The SAM-BA monitor supports a frequency of 12 MHz to allow DBGU communication for both external crystal and external clock.

10.5.2.2 Xmodem Protocol

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory in order to work.

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC16 to guarantee detection of a maximum bit error.

12.9.19 AIC Write Protection Mode Register

Name: AIC_WPMR

Address: 0xFFFFF1E4

Access: Read/Write

Reset: See Table 12-3

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

- **WPEN: Write Protection Enable**

0: Disables write protection if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

1: Enables write protection if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

See Section 12.8.8 “Register Write Protection” for list of write-protected registers.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x414943	PASSWD	Writing any other value in this field aborts the write operation of bit WPEN. Always reads as 0.

17.7.3 Shutdown Status Register

Name: SHDW_SR

Address: 0xFFFFFE18

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

- **WAKEUP0: Wake-up 0 Status**

0: No wake-up event occurred on WKUP0 input since the last read of SHDW_SR.

1: At least one wake-up event occurred on WKUP0 input since the last read of SHDW_SR.

- **RTCWK: Real-time Clock Wake-up**

0: No wake-up alarm from the RTC occurred since the last read of SHDW_SR.

1: At least one wake-up alarm from the RTC occurred since the last read of SHDW_SR.

19.4 Functional Description

The bits RCEN, OSC32EN, OSCSEL, and OSC32BYP are located in the Slow Clock Controller Configuration Register (SCKC_CR) located at the address 0xFFFFFE50 in the backed-up part of the System Controller and, thus, they are preserved while VDDBU is present.

The embedded 32 kHz (typical) RC oscillator and the 32.768 kHz crystal oscillator can be enabled by setting to 1, respectively, the RCEN and OSC32EN bits. The Slow Clock Selector command (OSCSEL bit) selects the slow clock source.

The 32.768 kHz crystal oscillator can be bypassed by setting the OSC32BYP bit to accept an external slow clock on XIN32.

After the VDDBU power-on reset, the default configuration is RCEN = 1, OSC32EN = 0 and OSCSEL = 0, allowing the system to start on the embedded 32 kHz (typical) RC oscillator.

The programmer controls the slow clock switching by software and so must take precautions during the switching phase.

19.4.1 Switching from Embedded 32 kHz RC Oscillator to 32.768 kHz Crystal Oscillator

The sequence to switch from the embedded 32 kHz (typical) RC oscillator to the 32.768 kHz crystal oscillator is the following:

1. Switch the master clock to a source different from slow clock (PLL or Main Oscillator) through the Power Management Controller.
2. Enable the 32.768 kHz crystal oscillator by writing a 1 to the OSC32EN bit.
3. Wait for the 32.768 kHz crystal oscillator to stabilize (software loop).
4. Switch from the embedded 32 kHz (typical) RC oscillator to the 32.768 kHz crystal oscillator by writing a 1 to the OSCSEL bit.
5. Wait 5 slow clock cycles for internal resynchronization.
6. Disable the 32 kHz (typical) RC oscillator by writing a 0 to the RCEN bit.

19.4.2 Bypassing the 32.768 kHz Crystal Oscillator

The sequence to bypass the 32.768 kHz crystal oscillator is the following:

1. An external clock must be connected on XIN32.
2. Enable the bypass path by writing a 1 to the OSC32BYP bit.
3. Disable the 32.768 kHz crystal oscillator by writing a 0 to the OSC32EN bit.

19.4.3 Switching from 32.768 kHz Crystal Oscillator to Embedded 32 kHz RC Oscillator

The sequence to switch from the 32.768 kHz crystal oscillator to the embedded 32 kHz (typical) RC oscillator is the following:

1. Switch the master clock to a source different from slow clock (PLL or Main Oscillator).
2. Enable the embedded 32 kHz (typical) RC oscillator for low power by writing a 1 to the RCEN bit.
3. Wait for the embedded 32 kHz (typical) RC oscillator to stabilize (software loop).
4. Switch from the 32.768 kHz crystal oscillator to the embedded RC oscillator by writing a 0 to the OSCSEL bit.
5. Wait 5 slow clock cycles for internal resynchronization.
6. Disable the 32.768 kHz crystal oscillator by writing a 0 to the OSC32EN bit.

26.6.12 PMECC ECC x Register

Name:	PMECC_ECCx [x=0..10] [sec_num=0..7]
Address:	0xFFFFE040 [0][0] .. 0xFFFFE068 [10][0] 0xFFFFE080 [0][1] .. 0xFFFFE0A8 [10][1] 0xFFFFE0C0 [0][2] .. 0xFFFFE0E8 [10][2] 0xFFFFE100 [0][3] .. 0xFFFFE128 [10][3] 0xFFFFE140 [0][4] .. 0xFFFFE168 [10][4] 0xFFFFE180 [0][5] .. 0xFFFFE1A8 [10][5] 0xFFFFE1C0 [0][6] .. 0xFFFFE1E8 [10][6] 0xFFFFE200 [0][7] .. 0xFFFFE228 [10][7]
Access:	Read-only
Reset:	0x00000000

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

- **ECC: BCH Redundancy**

This register contains the remainder of the division of the codeword by the generator polynomial.

26.6.13 PMECC Remainder x Register

Name:	PMECC_REMx [x=0..11] [sec_num=0..7]
Address:	0xFFFFE240 [0][0] .. 0xFFFFE26C [11][0] 0xFFFFE280 [0][1] .. 0xFFFFE2AC [11][1] 0xFFFFE2C0 [0][2] .. 0xFFFFE2EC [11][2] 0xFFFFE300 [0][3] .. 0xFFFFE32C [11][3] 0xFFFFE340 [0][4] .. 0xFFFFE36C [11][4] 0xFFFFE380 [0][5] .. 0xFFFFE3AC [11][5] 0xFFFFE3C0 [0][6] .. 0xFFFFE3EC [11][6] 0xFFFFE400 [0][7] .. 0xFFFFE42C [11][7]
Access:	Read-only
Reset:	0x00000000

31	30	29	28	27	26	25	24
–	–	REM2NP3					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
REM2NP3							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	REM2NP1					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REM2NP1							

- **REM2NP1: BCH Remainder 2 * N + 1**

When sector size is set to 512 bytes, bit REM2NP1[13] is not used and read as zero.

If bit *i* of the REM2NP1 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

- **REM2NP3: BCH Remainder 2 * N + 3**

When sector size is set to 512 bytes, bit REM2NP3[29] is not used and read as zero.

If bit *i* of the REM2NP3 field is set to one then the coefficient of the X ^ *i* is set to one, otherwise the coefficient is zero.

Figure 28-23. TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

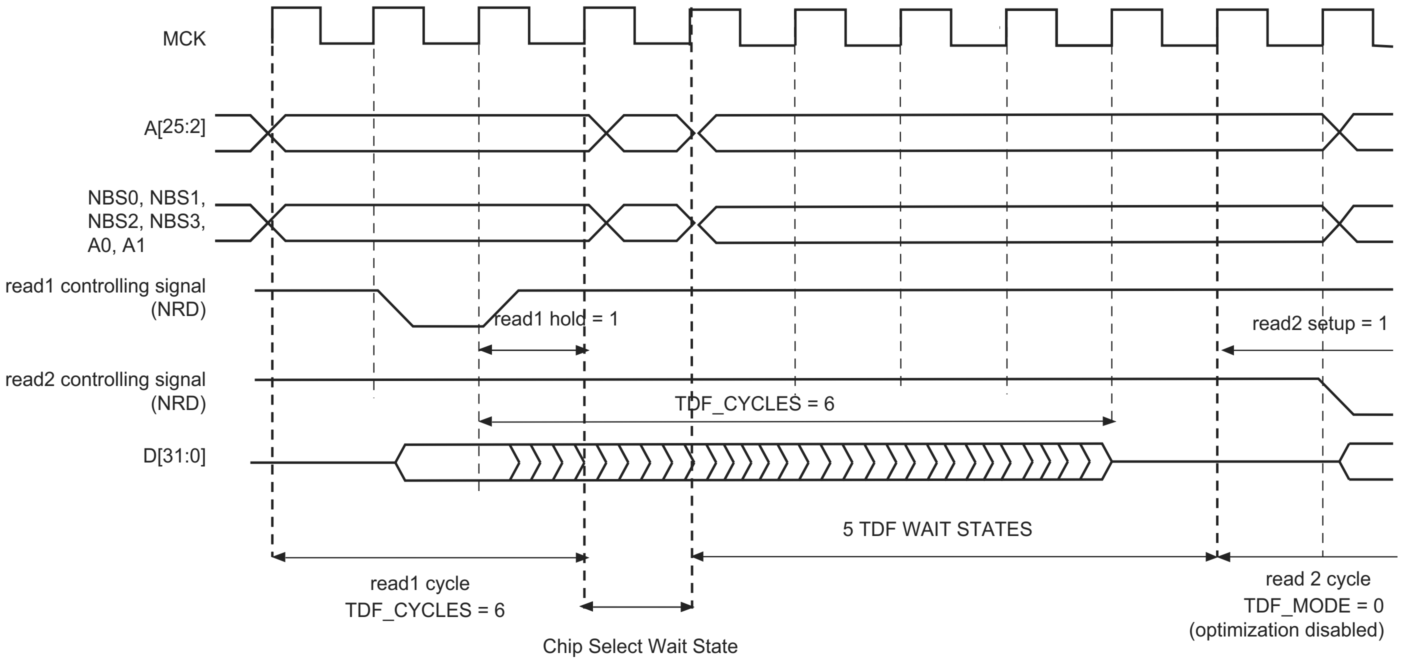
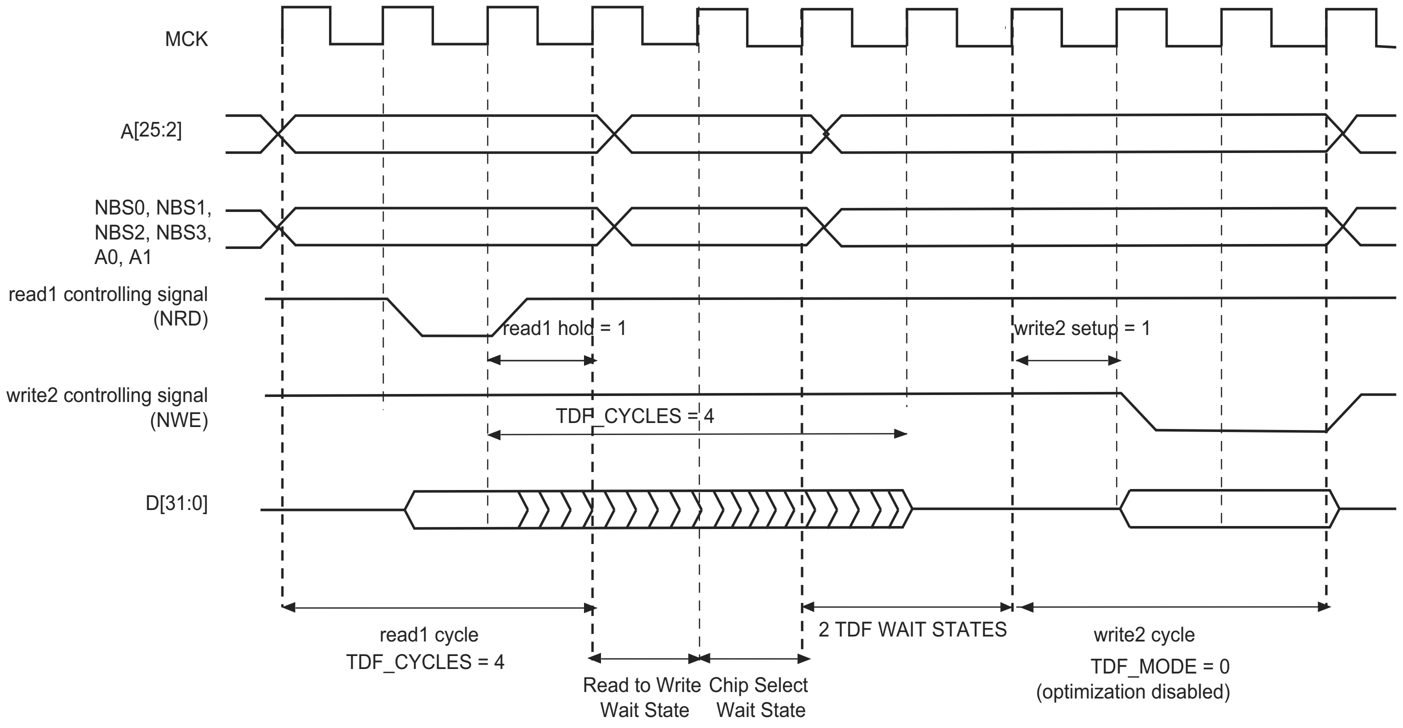


Figure 28-24. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects



28.15 Register Write Protection

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

If a write access to a write-protected register is detected, the WPVS flag in the SMC Write Protection Status Register (SMC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the SMC_WPSR.

The following registers can be write-protected:

- SMC Setup Register
- SMC Pulse Register
- SMC Cycle Register
- SMC Mode Register

29.4 Initialization Sequence

The addresses given are for example purposes only. The real address depends on implementation in the product.

29.4.1 SDR-SDRAM Initialization

The initialization sequence is generated by software. The following sequence initializes SDR-SDRAM devices:

1. Program the memory device type in the Memory Device Register (see Section 29.7.8 on page 471).
2. Program the features of the SDR-SDRAM device in the Timing Register (asynchronous timing (trc, tras, etc.)), and in the Configuration Register (number of columns, rows, banks, CAS latency) (see Section 29.7.3 on page 462, Section 29.7.4 on page 465 and Section 29.7.5 on page 467).
3. For low-power SDRAM, drive strength (DS) and partial array self refresh (PASR) must be set in the Low-power Register (see Section 29.7.7 on page 469).

A minimum pause of 200 µs is provided to precede any signal toggle.

4. A NOP command is issued to the SDR-SDRAM. To program the NOP command, the application must configure the MODE field value to 1 in the Mode Register (see Section 29.7.1 on page 460) and perform a write access to any SDR-SDRAM address to acknowledge this command. Now the clock which drives SDR-SDRAM device is enabled.
5. An All Banks Precharge command is issued to the SDR-SDRAM. To program All Banks Precharge command, the application must configure the MODE field value to 2 in the Mode Register (see Section 29.7.1 on page 460) and perform a write access to any SDR-SDRAM address to acknowledge this command.
6. Eight CAS before RAS (CBR) auto-refresh cycles are provided. To program the auto refresh command (CBR), the application must configure the MODE field value to 4 in the Mode Register (see Section 29.7.1 on page 460) and perform a write access to any SDR-SDRAM location eight times to acknowledge these commands.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDR-SDRAM devices, in particular CAS latency and burst length. The application must configure the MODE field value to 3 in the Mode Register (see Section 29.7.1 on page 460) and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20000000.

Note: This address is for example purposes only. The real address is dependent on implementation in the product.

8. For low-power SDR-SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDR-SDRAM parameters (PASR, DS acronyms in JEDEC datasheet). The application must configure the MODE field value to 5 in the Mode Register (see Section 29.7.1 on page 460) and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1] is set to 1 and BA[0] is set to 0. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be performed at the address 0x20800000.
9. The application must go into Normal mode by configuring the MODE field value to 0 in the Mode Register (see Section 29.7.1 on page 460) and performing a write access at any location in the SDRAM to acknowledge this command.
10. Write the refresh rate into the COUNT field in the DDRSDRC Refresh Timer Register (DDRSDRC_RTR). (Refresh rate = delay between refresh cycles). The SDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, DDRSDRC_RTR.COUNT must be configured to 15.625 $\times$ 100 MHz = 1562 (0x061A) or 7.81 $\times$ 100 MHz = 781 (0x030D).

After initialization, the SDR-SDRAM device is fully functional.

- If no bank is validated yet, the default DATA0 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). Then, no data bank is flushed at microframe end.
- If no data bank has been validated at the time when a response should be made for the second transaction of NB_TRANS = 3 transactions microframe, a DATA1 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). If and only if remaining untransmitted banks for that microframe are available at its end, they are flushed and an error condition is flagged (ERR_FLUSH is set in UDPHS_EPTSTAx).
- If no data bank has been validated at the time when a response should be made for the last programmed transaction of a microframe, a DATA0 ZLP is answered and underflow is flagged (ERR_FL_ISO is set in UDPHS_EPTSTAx). If and only if the remaining untransmitted data bank for that microframe is available at its end, it is flushed and an error condition is flagged (ERR_FLUSH is set in UDPHS_EPTSTAx).
- If at the end of a microframe no valid token IN has been recognized, no data bank is flushed and no error condition is reported.

At the end of a microframe in which at least one data bank has been transmitted, if less than NB_TRANS banks have been validated for that microframe, an error condition is flagged (ERR_TRANS is set in UDPHS_EPTSTAx).

Cases of Error (in UDPHS_EPTSTAx)

- ERR_FL_ISO: There was no data to transmit inside a microframe, so a ZLP is answered by default.
- ERR_FLUSH: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of transactions actually validated (TXRDY_TRER) and likewise with the NB_TRANS programmed.
- ERR_TRANS: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of programmed NB_TRANS transactions and the packets not requested were not validated.
- ERR_FL_ISO + ERR_FLUSH: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN.
- ERR_FL_ISO + ERR_TRANS: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN and the data can be discarded at the microframe end.
- ERR_FLUSH + ERR_TRANS: The first token IN has been answered and it was the only one received, a second bank has been validated but not the third, whereas NB_TRANS was waiting for three transactions.
- ERR_FL_ISO + ERR_FLUSH + ERR_TRANS: The first token IN has been treated, the data for the second Token IN was not available in time, but the second bank has been validated before the end of the microframe. The third bank has not been validated, but three transactions have been set in NB_TRANS.

31.6.10.4 Data OUT

Bulk OUT or Interrupt OUT

Like data IN, data OUT packets are sent by the host during the data or the status stage of control transfer or during an interrupt/bulk/isochronous OUT transfer. Data buffers are sent packet by packet under the control of the application or under the control of the DMA channel.

Bulk OUT or Interrupt OUT: Receiving a Packet Under Application Control (Host to Device)

Algorithm Description for Each Packet:

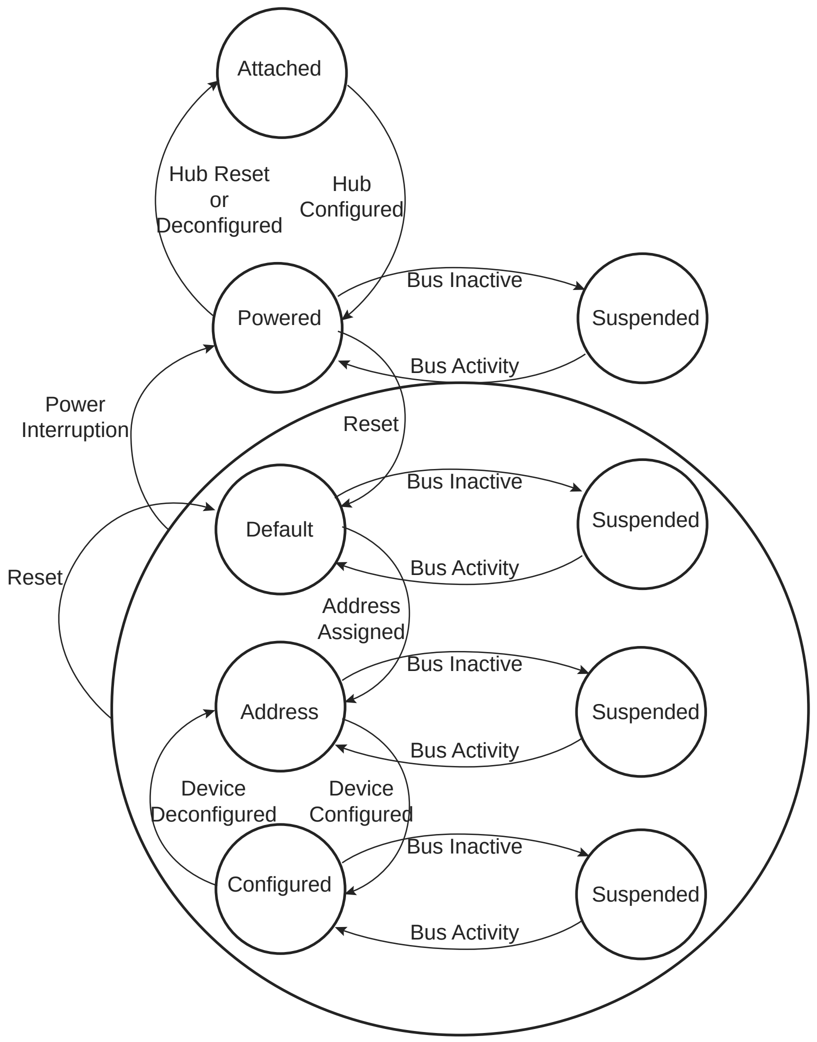
- The application enables an interrupt on RXRDY_TXKL.
- When an interrupt on RXRDY_TXKL is received, the application knows that UDPHS_EPTSTAx register BYTE_COUNT bytes have been received.
- The application reads the BYTE_COUNT bytes from the endpoint.
- The application clears RXRDY_TXKL.

31.6.14 Power Modes

31.6.14.1 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 (USB Device Framework) of the Universal Serial Bus Specification, Rev 2.0.

Figure 31-20. UDPHS Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend mode are very strict for bus-powered applications; devices may not consume more than 500 µA on the USB bus.

While in Suspend mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake-up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake-up feature is not mandatory for all devices and must be negotiated with the host.

- **LDBDIS: Counter Clock Disable with RB Loading**

0: Counter clock is not disabled when RB loading occurs.

1: Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0: TIOB is used as an external trigger.

1: TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0: RC Compare has no effect on the counter and its clock.

1: RC Compare resets the counter and starts the counter clock.

- **WAVE: Waveform Mode**

0: Capture mode is enabled.

1: Capture mode is disabled (Waveform mode is enabled).

- **LDRA: RA Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

- **LDRB: RB Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

- **ETRGS: External Trigger Status (cleared on read)**

0: External trigger has not occurred since the last read of the Status Register.

1: External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0: Clock is disabled.

1: Clock is enabled.

- **MTIOA: TIOA Mirror**

0: TIOA is low. If TC_CMRx.WAVE = 0, this means that TIOA pin is low. If TC_CMRx.WAVE = 1, this means that TIOA is driven low.

1: TIOA is high. If TC_CMRx.WAVE = 0, this means that TIOA pin is high. If TC_CMRx.WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0: TIOB is low. If TC_CMRx.WAVE = 0, this means that TIOB pin is low. If TC_CMRx.WAVE = 1, this means that TIOB is driven low.

1: TIOB is high. If TC_CMRx.WAVE = 0, this means that TIOB pin is high. If TC_CMRx.WAVE = 1, this means that TIOB is driven high.

The main properties of the LIN bus are:

- Single master/multiple slaves concept
- Low-cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine.
- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN mode enables processing LIN frames with a minimum of action from the microprocessor.

38.6.8.1 Modes of Operation

The USART can act either as a LIN master node or as a LIN slave node.

The node configuration is chosen by setting the USART_MODE field in the USART Mode register (US_MR):

- LIN master node (USART_MODE = 0xA)
- LIN slave node (USART_MODE = 0xB)

In order to avoid unpredictable behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See Section 38.6.8.3.)

38.6.8.2 Baud Rate Configuration

See Section 38.6.1.1 ”Baud Rate in Asynchronous Mode”

The baud rate is configured in US_BRGR.

38.6.8.3 Receiver and Transmitter Control

See Section 38.6.2 ”Receiver and Transmitter Control”

38.6.8.4 Character Transmission

See Section 38.6.3.1 ”Transmitter Operations”.

38.6.8.5 Character Reception

See Section 38.6.3.7 ”Receiver Operations”.

38.6.8.6 Header Transmission (Master Node Configuration)

All the LIN frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in master node configuration, the frame handling starts with the sending of the header.

The header is transmitted as soon as the identifier is written in the LIN Identifier register (US_LINIR). At this moment the flag TXRDY falls.

The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier register (US_LINIR). The Identifier parity bits can be automatically computed and sent (see Section 38.6.8.9 ”Identifier Parity”).

The flag TXRDY rises when the identifier character is transferred into the Shift register of the transmitter.

As soon as the Synch Break Field is transmitted, the flag LINBK in US_CSR is set to 1. Likewise, as soon as the Identifier Field is sent, the flag bit LINID in the US_CSR is set to 1. These flags are reset by writing a 1 to the bit RSTSTA in US_CR.

38.7.21 USART Transmitter Timeguard Register

Name: US_TTGR

Address: 0xF801C028 (0), 0xF8020028 (1), 0xF8024028 (2), 0xF8028028 (3)

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**
TG							

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

- **TG: Timeguard Value**

0: The transmitter timeguard is disabled.

1–255: The transmitter timeguard is enabled and TG is Timeguard Delay / Bit Period.

- **CRC_ERR: Embedded Synchronization CRC Error Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **FR_OVR: Frame Rate Overflow Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

For a properly working Ethernet system, there should be no excessively long frames or frames greater than 128 bytes with CRC/FCS errors. Collision fragments are less than 128 bytes long. Therefore, it is a rare occurrence to find a frame fragment in a receive buffer.

If bit 0 is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit 0. In this case, the DMA block sets the 'Buffer Not Available' bit in the Receive Status Register (EMAC_RSR) and triggers an interrupt.

If bit 0 is set when the receive buffer manager reads the location of the receive buffer and a frame is being received, the frame is discarded and the Receive Resource Errors Register (EMAC_RRE) is incremented.

A receive overrun condition occurs when bus was not granted in time or because HRESP was not OK (bus error). In a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame received with an address that is recognized reuses the buffer.

If bit 17 of the EMAC_NCFGR is set, the FCS of received frames shall not be copied to memory. The frame length indicated in the receive status field shall be reduced by four bytes in this case.

44.4.2.3 Transmit Buffer

Frames to be transmitted are stored in one or more transmit buffers. Transmit buffers can be between 0 and 2047 bytes long, so it is possible to transmit frames longer than the maximum length specified in IEEE Standard 802.3. Zero length buffers are allowed. The maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the Transmit Buffer Queue Pointer Register (EMAC_TBQP). Each list entry consists of two words, the first being the byte address of the transmit buffer and the second containing the transmit control and status. Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad is also automatically generated to take frames to a minimum length of 64 bytes. Table 44-2 on page 1053 defines an entry in the transmit buffer descriptor list. To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits 31 to 0 in the first word of each list entry. The second transmit buffer descriptor is initialized with control information that indicates the length of the buffer, whether or not it is to be transmitted with CRC and whether the buffer is the last buffer in the frame.

After transmission, the control bits are written back to the second word of the first buffer along with the 'used' bit and other status information. Bit 31 is the 'used' bit which must be zero when the control word is read if transmission is to happen. It is written to one when a frame has been transmitted. Bits 27, 28 and 29 indicate various transmit error conditions. Bit 30 is the 'wrap' bit which can be set for any buffer within a frame. If no wrap bit is encountered after 1024 descriptors, the queue pointer rolls over to the start in a similar fashion to the receive queue.

The EMAC_TBQP register must not be written while transmit is active. If a new value is written to the EMAC_TBQP register, the queue pointer resets itself to point to the beginning of the new queue. If transmit is disabled by writing to bit 3 of the EMAC_NCR, the EMAC_TBQP register resets to point to the beginning of the transmit queue. Note that disabling receive does not have the same effect on the receive queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to bit 9, the 'Start Transmission' bit of the EMAC_NCR. Transmit is halted when a buffer descriptor with its 'used' bit set is read, or if a transmit error occurs, or by writing to the 'Transmit Halt' bit of the EMAC_NCR. (Transmission is suspended if a pause frame is received while the 'Pause Enable' bit is set in the EMAC_NCFGR.) Rewriting the start bit while transmission is active is allowed.

44.6.24 Type ID Checking Register

Name: EMAC_TID

Address: 0xF802C0B8

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
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID Checking**

For use in comparisons with received frames TypeID/Length field.

Table of Contents