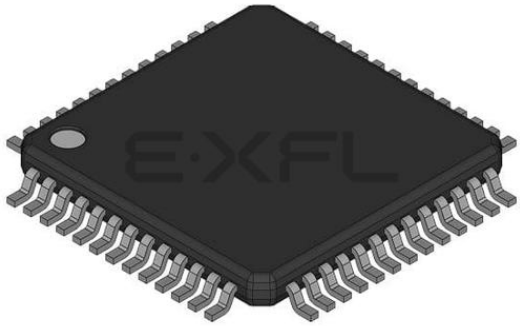

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HCS12
Core Size	16-Bit
Speed	25MHz
Connectivity	EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	35
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-QFP (10x10)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12gc128mpbe

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x005F	TC7 (lo)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0060	PACTL	Read:	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
		Write:								
0x0061	PAFLG	Read:	0	0	0	0	0	0	PAOVF	PAIF
		Write:								
0x0062	PACNT (hi)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
0x0063	PACNT (lo)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
0x0064	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0065	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0066	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0067	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0068	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0069	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006A	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006B	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006C	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006D	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006E	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x006F	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

0x0070–0x007F Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0070–0x007F	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

0x0180–0x023F Reserved

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0180–0x023F	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								

0x0240–0x027F PIM (Port Interface Module) (Sheet 1 of 3)

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0240	PTT	Read:	PTT7	PTT6	PTT5	PTT4	PTT3	PTT2	PTT1	PTT0
		Write:								
0x0241	PTIT	Read:	PTIT7	PTIT6	PTIT5	PTIT4	PTIT3	PTIT2	PTIT1	PTIT0
		Write:								
0x0242	DDRT	Read:	DDRT7	DDRT7	DDRT5	DDRT4	DDRT3	DDRT2	DDRT1	DDRT0
		Write:								
0x0243	RDRT	Read:	RDRT7	RDRT6	RDRT5	RDRT4	RDRT3	RDRT2	RDRT1	RDRT0
		Write:								
0x0244	PERT	Read:	PERT7	PERT6	PERT5	PERT4	PERT3	PERT2	PERT1	PERT0
		Write:								
0x0245	PPST	Read:	PPST7	PPST6	PPST5	PPST4	PPST3	PPST2	PPST1	PPST0
		Write:								
0x0246	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0247	MODRR	Read:	0	0	0	MODRR4	MODRR3	MODRR2	MODRR1	MODRR0
		Write:								
0x0248	PTS	Read:	0	0	0	0	PTS3	PTS2	PTS1	PTS0
		Write:								
0x0249	PTIS	Read:	0	0	0	0	PTIS3	PTIS2	PTIS1	PTIS0
		Write:								
0x024A	DDRS	Read:	0	0	0	0	DDRS3	DDRS2	DDRS1	DDRS0
		Write:								
0x024B	RDRS	Read:	0	0	0	0	RDRS3	RDRS2	RDRS1	RDRS0
		Write:								
0x024C	PERS	Read:	0	0	0	0	PERS3	PERS2	PERS1	PERS0
		Write:								
0x024D	PPSS	Read:	0	0	0	0	PPSS3	PPSS2	PPSS1	PPSS0
		Write:								
0x024E	WOMS	Read:	0	0	0	0	WOMS3	WOMS2	WOMS1	WOMS0
		Write:								
0x024F	Reserved	Read:	0	0	0	0	0	0	0	0
		Write:								
0x0250	PTM	Read:	0	0	PTM5	PTM4	PTM3	PTM2	PTM1	PTM0
		Write:								
0x0251	PTIM	Read:	0	0	PTIM5	PTIM4	PTIM3	PTIM2	PTIM1	PTIM0
		Write:								
0x0252	DDRM	Read:	0	0	DDRM5	DDRM4	DDRM3	DDRM2	DDRM1	DDRM0
		Write:								

3.3.2.7 Memory Size Register 0 (MEMSIZ0)

Module Base + 0x001C
Starting address location affected by INITRG register setting.

	7	6	5	4	3	2	1	0
R	REG_SW0	0	EEP_SW1	EEP_SW0	0	RAM_SW2	RAM_SW1	RAM_SW0
W								
Reset	—	—	—	—	—	—	—	—

= Unimplemented or Reserved

Figure 3-9. Memory Size Register 0 (MEMSIZ0)

Read: Anytime

Write: Writes have no effect

Reset: Defined at chip integration, see device overview section.

The MEMSIZ0 register reflects the state of the register, EEPROM and RAM memory space configuration switches at the core boundary which are configured at system integration. This register allows read visibility to the state of these switches.

Table 3-7. MEMSIZ0 Field Descriptions

Field	Description
7 REG_SW0	**Allocated System Register Space** 0 Allocated system register space size is 1K byte 1 Allocated system register space size is 2K byte
5:4 EEP_SW[1:0]	**Allocated System EEPROM Memory Space** — The allocated system EEPROM memory space size is as given in Table 3-8.
2 RAM_SW[2:0]	**Allocated System RAM Memory Space** — The allocated system RAM memory space size is as given in Table 3-9.

Table 3-8. Allocated EEPROM Memory Space

eep_sw1:eep_sw0	Allocated EEPROM Space
00	0K byte
01	2K bytes
10	4K bytes
11	8K bytes

Table 3-9. Allocated RAM Memory Space

ram_sw2:ram_sw0	Allocated RAM Space	RAM Mappable Region	INITRM Bits Used	RAM Reset Base Address[1]
000	2K bytes	2K bytes	RAM[15:11]	0x0800
001	4K bytes	4K bytes	RAM[15:12]	0x0000
010	6K bytes	8K bytes[2]	RAM[15:13]	0x0800

9.3.2.10 Reserved Register (FORBYP)

NOTE

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special modes can alter the CRG's functionality.

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-13. Reserved Register (FORBYP)

Read: always read 0x0000 except in special modes

Write: only in special modes

9.3.2.11 Reserved Register (CTCTL)

NOTE

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in special test modes can alter the CRG's functionality.

Module Base + 0x000A

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-14. Reserved Register (CTCTL)

Read: always read 0x0080 except in special modes

Write: only in special modes

NOTE

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

10.3.2.15 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-18. MSCAN Transmit Error Counter (CANTXERR)

Read: Only when in sleep mode (SLPRQ = 1 and SLPAK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

Write: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 10.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 10.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

10.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 10-38). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 10-38). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 10.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 10.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 10.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO[2], sets the RXF flag, and generates a receive interrupt (see Section 10.4.7.3, "Receive Interrupt") to the CPU[3]. The user's receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.
2. Only if the RXF flag is not set.
3. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

If the bus clock is generated from a PLL, it is recommended to select the oscillator clock rather than the bus clock due to jitter considerations, especially at the faster CAN bus rates.

For microcontrollers without a clock and reset generator (CRG), CANCLK is driven from the crystal oscillator (oscillator clock).

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

Eqn. 10-2

$$Tq = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments as described in the Bosch CAN specification. (see Figure 10-43):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

Eqn. 10-3

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

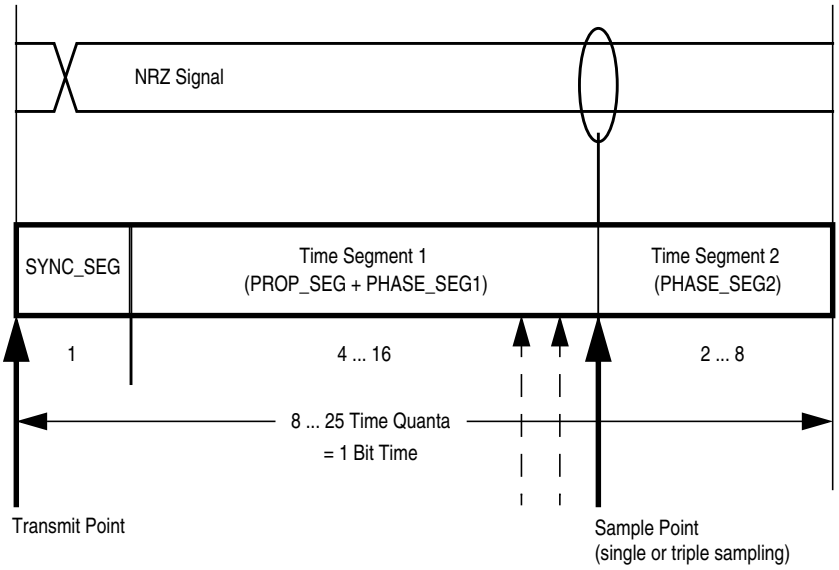

Figure 10-43. Segments within the Bit Time

12.3.2.12 PWM Channel Counter Registers (PWMCNTx)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source. The counter can be read at any time without affecting the count or the operation of the PWM channel. In left aligned output mode, the counter counts from 0 to the value in the period register – 1. In center aligned output mode, the counter counts from 0 up to the value in the period register and then back down to 0.

Any value written to the counter causes the counter to reset to 0x0000, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. The counter is also cleared at the end of the effective period (see Section 12.4.2.5, “Left Aligned Outputs,” and Section 12.4.2.6, “Center Aligned Outputs,” for more details). When the channel is disabled (PWMEx = 0), the PWMCNTx register does not count. When a channel becomes enabled (PWMEx = 1), the associated PWM counter starts at the count in the PWMCNTx register. For more detailed information on the operation of the counters, reference Section 12.4.2.4, “PWM Timer Counters.”

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low- or high-order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

Figure 12-15. PWM Channel Counter Registers (PWMCNT0)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

Figure 12-16. PWM Channel Counter Registers (PWMCNT1)

13.3.2.1 SCI Baud Rate Registers (SCIBDH and SCHBDL)

Module Base + 0x_0000

	7	6	5	4	3	2	1	0
R	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
W								
Reset	0	0	0	0	0	0	0	0

Module Base + 0x_0001

	7	6	5	4	3	2	1	0
R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
W								
Reset	0	0	0	0	0	1	0	0

= Unimplemented or Reserved

Figure 13-3. SCI Baud Rate Registers (SCIBDH and SCIBDL)

The SCI Baud Rate Register is used by the counter to determine the baud rate of the SCI. The formula for calculating the baud rate is:

SCI baud rate = SCI module clock / (16 x BR)

where:

BR is the content of the SCI baud rate registers, bits SBR12 through SBR0. The baud rate registers can contain a value from 1 to 8191.

Read: Anytime. If only SCIBDH is written to, a read will not return the correct data until SCIBDL is written to as well, following a write to SCIBDH.

Write: Anytime

Table 13-1. SCIBDH AND SCIBDL Field Descriptions

Field	Description
4–0 7–0 SBR[12:0]	**SCI Baud Rate Bits** — The baud rate for the SCI is determined by these 13 bits. **Note:** The baud rate generator is disabled until the TE bit or the RE bit is set for the first time after reset. The baud rate generator is disabled when BR = 0. Writing to SCIBDH has no effect without writing to SCIBDL, since writing to SCIBDH puts the data in a temporary location until SCIBDL is written to.

13.4 Functional Description

This section provides a complete functional description of the SCI block, detailing the operation of the design from the end user perspective in a number of subsections.

Figure 13-9 shows the structure of the SCI module. The SCI allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The SCI transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

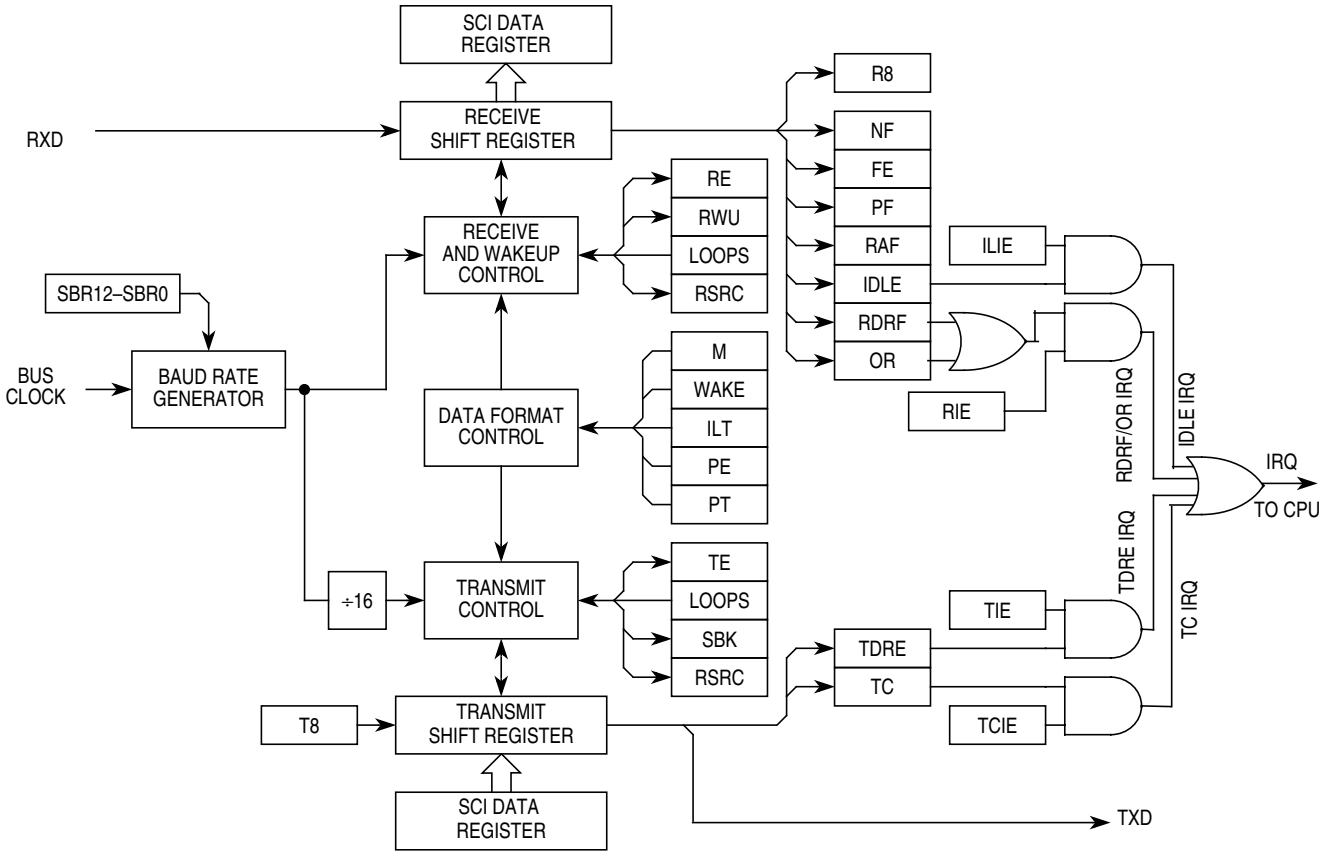

Figure 13-9. SCI Block Diagram

13.4.3 Transmitter

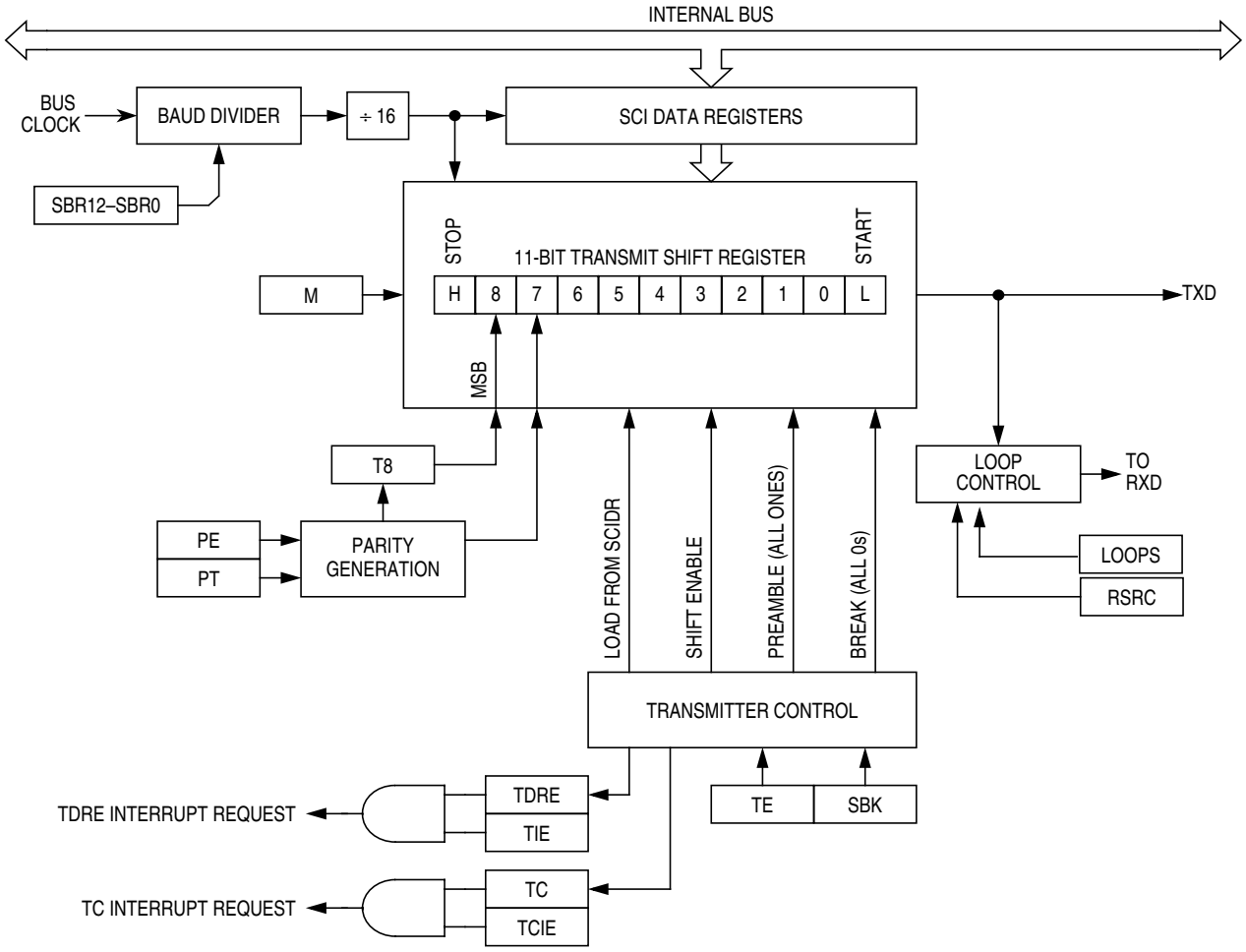

Figure 13-11. Transmitter Block Diagram

13.4.3.1 Transmitter Character Length

The SCI transmitter can accommodate either 8-bit or 9-bit data characters. The state of the M bit in SCI control register 1 (SCICR1) determines the length of data characters. When transmitting 9-bit data, bit T8 in SCI data register high (SCIDRH) is the ninth bit (bit 8).

13.4.3.2 Character Transmission

To transmit data, the MCU writes the data bits to the SCI data registers (SCIDRH/SCIDRL), which in turn are transferred to the transmitter shift register. The transmit shift register then shifts a frame out through the **Tx output** signal, after it has prefaced them with a start bit and appended them with a stop bit. The SCI data registers (SCIDRH and SCIDRL) are the write-only buffers between the internal data bus and the transmit shift register.

The SCI also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDRH/L) to the transmitter shift register.The transmit driver routine may respond to this flag by

In Figure 13-14 the verification samples RT3 and RT5 determine that the first low detected was noise and not the beginning of a start bit. The RT clock is reset and the start bit search begins again. The noise flag is not set because the noise occurred before the start bit was found.

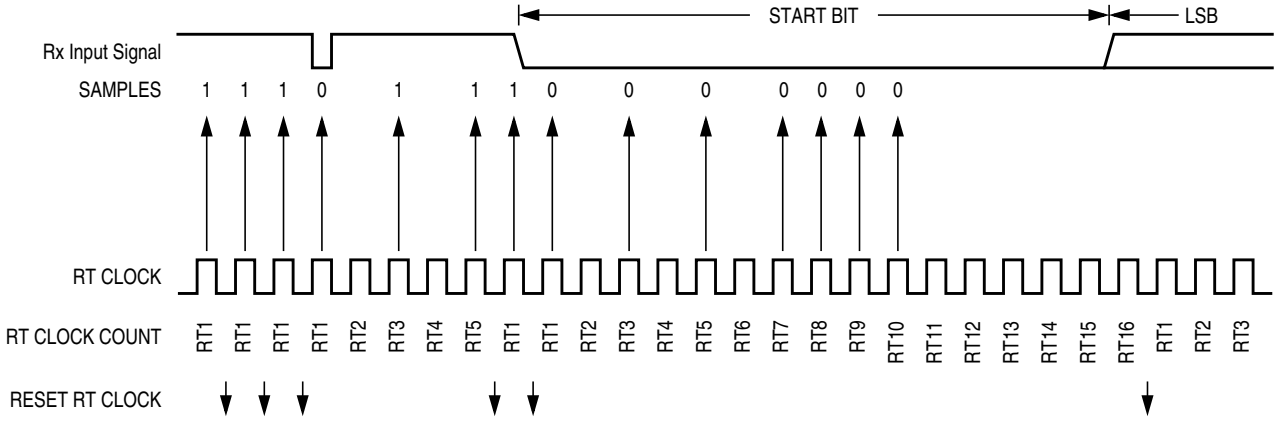

Figure 13-14. Start Bit Search Example 1

In Figure 13-15, verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

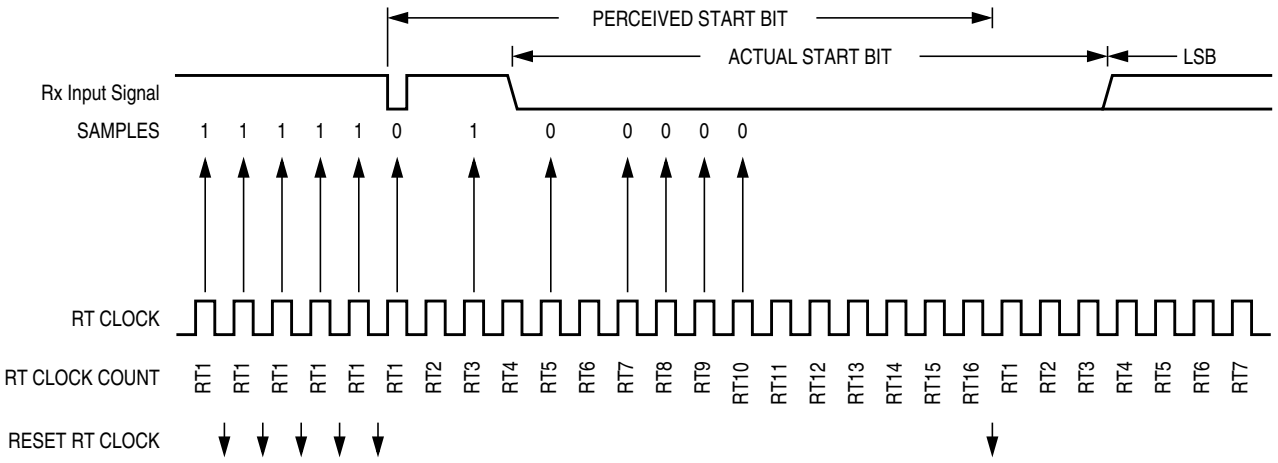

Figure 13-15. Start Bit Search Example 2

To change the Flash protection that will be loaded on reset, the upper sector of the Flash array must be unprotected, then the Flash protection byte located at Flash address 0xFF0D must be written to.

A protected Flash sector is disabled by FPHDIS while the size of the protected sector is defined by FPHS[1:0] in the FPROT register.

Trying to alter any of the protected areas will result in a protect violation error and the PVIOL flag will be set in the FSTAT register (see Section 17.3.2.6). A mass erase of the whole Flash array is only possible when protection is fully disabled by setting the FPOPEN and FPHDIS bits. An attempt to mass erase a Flash array while protection is enabled will set the PVIOL flag in the FSTAT register.

Table 17-8. FPROT Field Descriptions

Field	Description
7 FPOPEN	**Protection Function for Program or Erase** — The FPOPEN bit is used to either select an address range to be protected using the FPHDIS and FPHS[1:0] bits or to select the same address range to be unprotected as shown in Table 17-9. 0 The FPHDIS bit allows a Flash address range to be unprotected 1 The FPHDIS bit allows a Flash address range to be protected
6 NV6	**Nonvolatile Flag Bit** — The NV6 bit should remain in the erased state for future enhancements.
5 FPHDIS	**Flash Protection Higher Address Range Disable** — The FPHDIS bit determines whether there is a protected/unprotected area in the higher space of the Flash address map. 0 Protection/unprotection enabled 1 Protection/unprotection disabled
4–3 FPHS[1:0]	**Flash Protection Higher Address Size** — The FPHS[1:0] bits determine the size of the protected/unprotected sector as shown in Table 17-10. The FPHS[1:0] bits can only be written to while the FPHDIS bit is set.
2–0 NV[2:0]	**Nonvolatile Flag Bits** — The NV[2:0] bits should remain in the erased state for future enhancements.

Table 17-9. Flash Protection Function

FPOPEN	FPHDIS	FPHS1	FPHS0	Function(1)
1	1	x	x	No protection
1	0	x	x	Protect high range
0	1	x	x	Full Flash array protected
0	0	x	x	Unprotected high range

1. For range sizes refer to Table 17-10.

Table 17-10. Flash Protection Higher Address Range

FPHS[1:0]	Address Range	Range Size
00	0xF800–0xFFFF	2 Kbytes
01	0xF000–0xFFFF	4 Kbytes
10	0xE000–0xFFFF	8 Kbytes
11	0xC000–0xFFFF	16 Kbytes

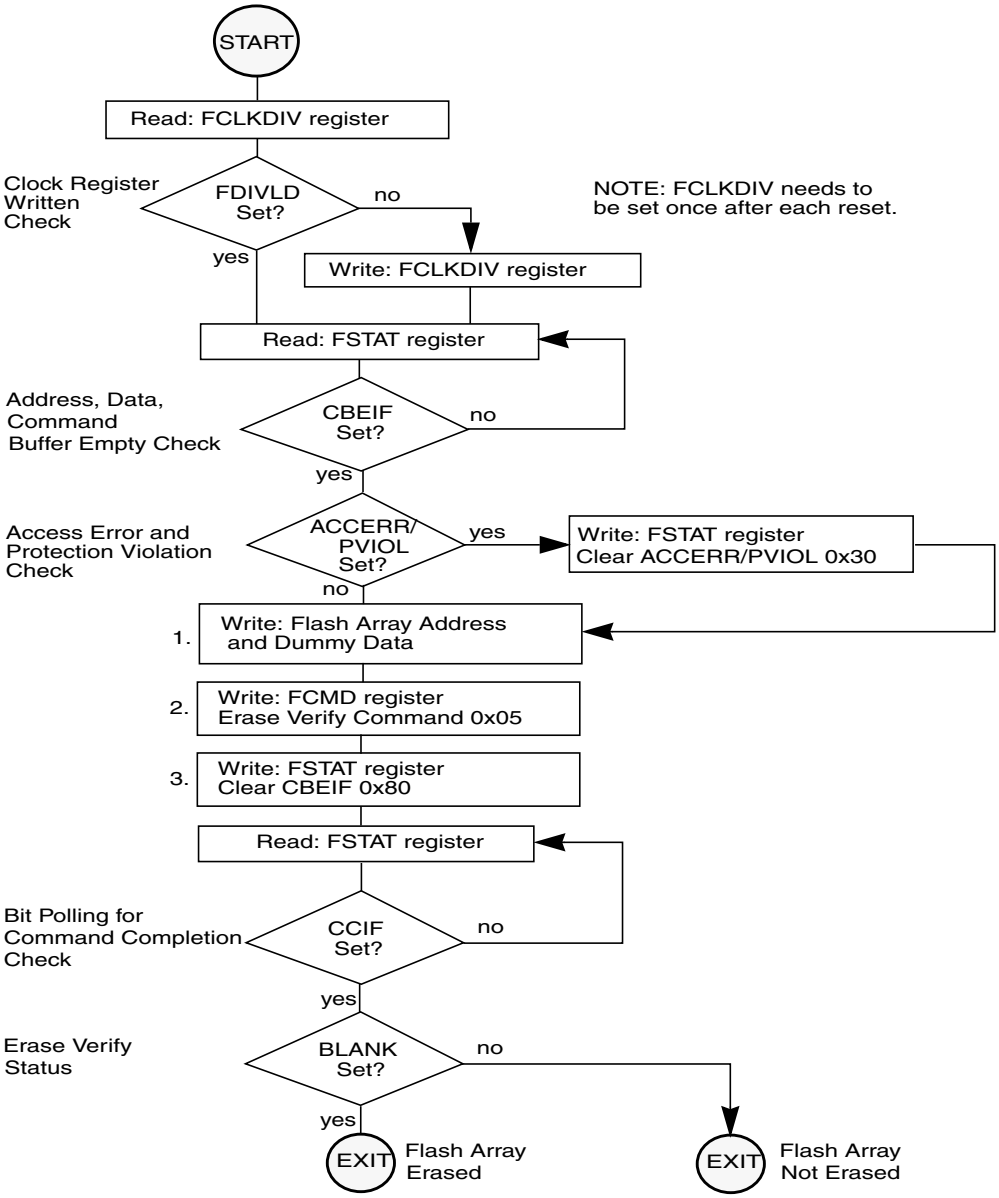

Figure 18-22. Example Erase Verify Command Flow

Table 19-13. Flash Protection Scenario Transitions

From Protection Scenario	To Protection Scenario[1]							
	0	1	2	3	4	5	6	7
6		X		X	X		X	
7	X	X	X	X	X	X	X	X

1. Allowed transitions marked with X.

19.3.2.6 Flash Status Register (FSTAT)

The FSTAT register defines the status of the Flash command controller and the results of command execution.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	FAIL	DONE
W								
Reset	1	1	0	0	0	0	0	1

= Unimplemented or Reserved

Figure 19-12. Flash Status Register (FSTAT)

In normal modes, bits CBEIF, PVIOL, and ACCERR are readable and writable, bits CCIF and BLANK are readable and not writable, remaining bits, including FAIL and DONE, read 0 and are not writable. In special modes, FAIL is readable and writable while DONE is readable but not writable. FAIL must be clear in special modes when starting a command write sequence.

Table 19-14. FSTAT Field Descriptions

Field	Description
7 CBEIF	**Command Buffer Empty Interrupt Flag** — The CBEIF flag indicates that the address, data and command buffers are empty so that a new command write sequence can be started. The CBEIF flag is cleared by writing a 1 to CBEIF. Writing a 0 to the CBEIF flag has no effect on CBEIF. Writing a 0 to CBEIF after writing an aligned word to the Flash address space but before CBEIF is cleared will abort a command write sequence and cause the ACCERR flag in the FSTAT register to be set. Writing a 0 to CBEIF outside of a command write sequence will not set the ACCERR flag. The CBEIF flag is used together with the CBEIE bit in the FCNFG register to generate an interrupt request (see Figure 19-29). 0 Buffers are full 1 Buffers are ready to accept a new command
6 CCIF	**Command Complete Interrupt Flag** — The CCIF flag indicates that there are no more commands pending. The CCIF flag is cleared when CBEIF is clear and sets automatically upon completion of all active and pending commands. The CCIF flag does not set when an active commands completes and a pending command is fetched from the command buffer. Writing to the CCIF flag has no effect. The CCIF flag is used together with the CCIE bit in the FCNFG register to generate an interrupt request (see Figure 19-29). 0 Command in progress 1 All commands are completed

19.4.1.3.3 Sector Erase Command

The sector erase operation will erase all addresses in a 1024 byte sector of the Flash array using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 19-27. The sector erase command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the sector erase command. The Flash address written determines the sector to be erased while MCU address bits [9:0] and the data written are ignored.
2. Write the sector erase command, 0x40, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the sector erase command.

If a Flash sector to be erased is in a protected area of the Flash array, the PVIOL flag in the FSTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the CCIF flag in the FSTAT register will set after the sector erase operation has completed unless a new command write sequence has been buffered.

19.4.1.3.4 Mass Erase Command

The mass erase operation will erase all addresses in a Flash array using an embedded algorithm.

An example flow to execute the mass erase operation is shown in Figure 19-28. The mass erase command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the mass erase command. The address and data written will be ignored.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the mass erase command.

If a Flash array to be erased contains any protected area, the PVIOL flag in the FSTAT register will set and the mass erase command will not launch. Once the mass erase command has successfully launched, the CCIF flag in the FSTAT register will set after the mass erase operation has completed unless a new command write sequence has been buffered.

Table 20-3. Flash Array Memory Map Summary

MCU Address Range	PPAGE	Protectable Low Range	Protectable High Range	Array Relative Address[1]
0x0000–0x3FFF[2]	Unpaged (0x3D)	N.A.	N.A.	0x14000–0x17FFF
0x4000–0x7FFF	Unpaged (0x3E)	0x4000–0x43FF 0x4000–0x47FF 0x4000–0x4FFF 0x4000–0x5FFF	N.A.	0x18000–0x1BFFF
0x8000–0xBFFF	0x3A	N.A.	N.A.	0x08000–0x0BFFF
	0x3B	N.A.	N.A.	0x0C000–0x0FFFF
	0x3C	N.A.	N.A.	0x10000–0x13FFF
	0x3D	N.A.	N.A.	0x14000–0x17FFF
	0x3E	0x8000–0x83FF 0x8000–0x87FF 0x8000–0x8FFF 0x8000–0x9FFF	N.A.	0x18000–0x1BFFF
	0x3F	N.A.	0xB800–0xBFFF 0xB000–0xBFFF 0xA000–0xBFFF 0x8000–0xBFFF	0x1C000–0x1FFFF
0xC000–0xFFFF	Unpaged (0x3F)	N.A.	0xF800–0xFFFF 0xF000–0xFFFF 0xE000–0xFFFF 0xC000–0xFFFF	0x1C000–0x1FFFF

1. Inside Flash block.
2. If allowed by MCU.

Table 20-15. FCMD Field Descriptions

Field	Description
6, 5, 2, 0 CMDB[6:5] CMDB[2] CMDB[0]	Valid Flash commands are shown in Table 20-16. An attempt to execute any command other than those listed in Table 20-16 will set the ACCERR bit in the FSTAT register (see Section 20.3.2.6).

Table 20-16. Valid Flash Command List

CMDB	NVM Command
0x05	Erase verify
0x20	Word program
0x40	Sector erase
0x41	Mass erase

20.3.2.8 RESERVED2

This register is reserved for factory testing and is not accessible to the user.

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 20-14. RESERVED2

All bits read 0 and are not writable.

20.3.2.9 Flash Address Register (FADDR)

FADDRHI and FADDRLO are the Flash address registers.

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R	FABHI							
W								
Reset	0	0	0	0	0	0	0	0

Figure 20-15. Flash Address High Register (FADDRHI)

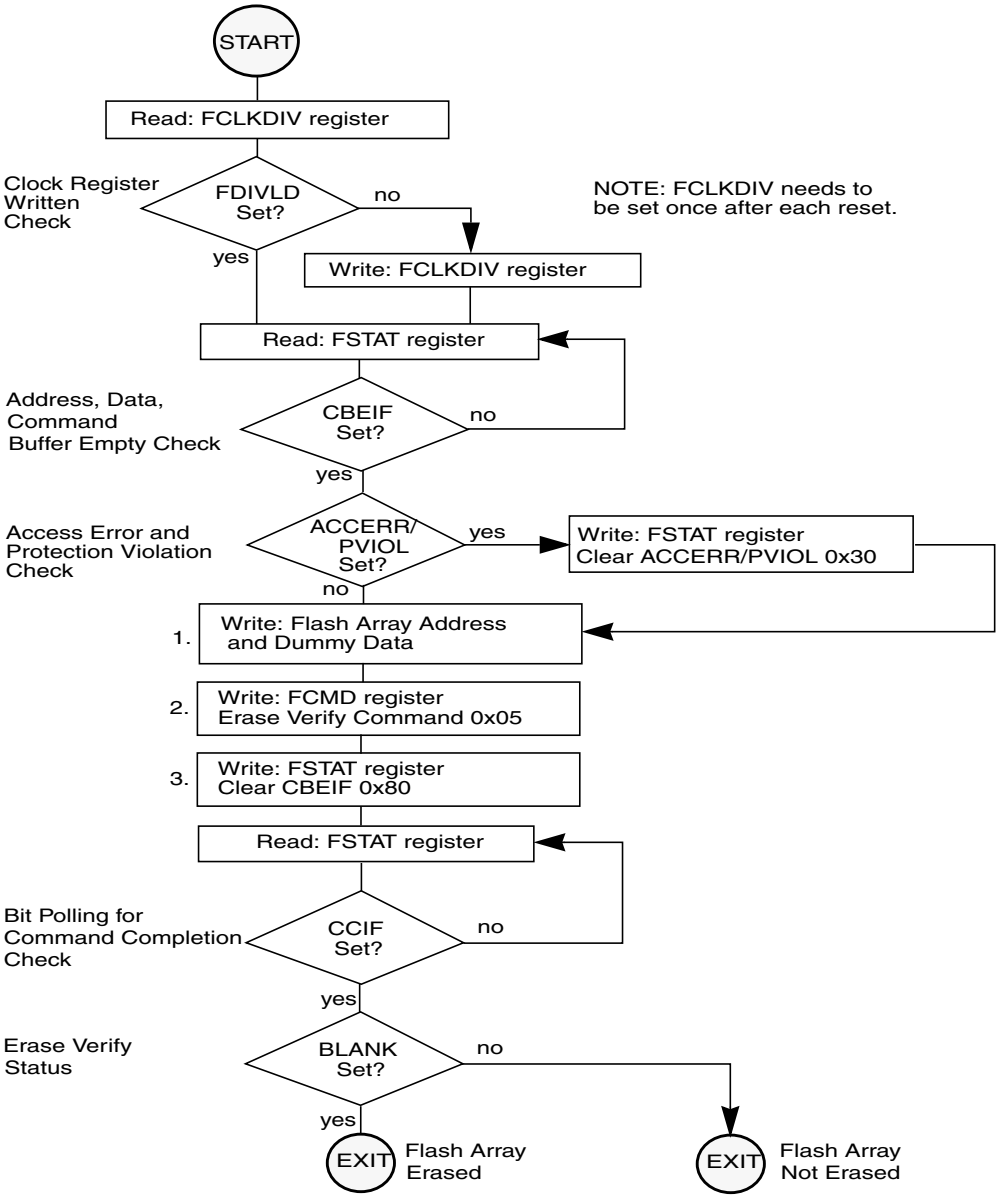

Figure 21-22. Example Erase Verify Command Flow