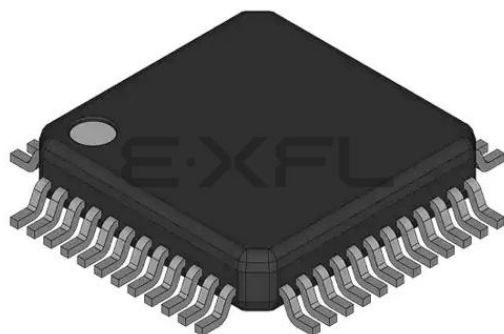




Welcome to E-XFL.COM

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	31
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K 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	48-LQFP
Supplier Device Package	48-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12gc16mfae

15.2.8	IOC0 — Input Capture and Output Compare Channel 0 Pin	439
15.3	Memory Map and Register Definition	439
15.3.1	Module Memory Map	439
15.3.2	Register Descriptions	441
15.4	Functional Description	457
15.4.1	Prescaler	458
15.4.2	Input Capture	459
15.4.3	Output Compare	459
15.4.4	Pulse Accumulator	460
15.4.5	Event Counter Mode	460
15.4.6	Gated Time Accumulation Mode	461
15.5	Resets	461
15.6	Interrupts	461
15.6.1	Channel [7:0] Interrupt (C[7:0]F)	461
15.6.2	Pulse Accumulator Input Interrupt (PAOVI)	462
15.6.3	Pulse Accumulator Overflow Interrupt (PAOVF)	462
15.6.4	Timer Overflow Interrupt (TOF)	462

Chapter 16

Dual Output Voltage Regulator (VREG3V3V2)

Block Description

16.1	Introduction	463
16.1.1	Features	463
16.1.2	Modes of Operation	463
16.1.3	Block Diagram	464
16.2	External Signal Description	465
16.2.1	V _{DDR} — Regulator Power Input	465
16.2.2	V _{DDA} , V _{SSA} — Regulator Reference Supply	465
16.2.3	V _{DD} , V _{SS} — Regulator Output1 (Core Logic)	466
16.2.4	V _{DDPLL} , V _{SSPLL} — Regulator Output2 (PLL)	466
16.2.5	V _{REGEN} — Optional Regulator Enable	466
16.3	Memory Map and Register Definition	466
16.3.1	Module Memory Map	466
16.3.2	Register Descriptions	467
16.4	Functional Description	467
16.4.1	REG — Regulator Core	468
16.4.2	Full-Performance Mode	468
16.4.3	Reduced-Power Mode	468
16.4.4	LVD — Low-Voltage Detect	468
16.4.5	POR — Power-On Reset	468
16.4.6	LVR — Low-Voltage Reset	468
16.4.7	CTRL — Regulator Control	468
16.5	Resets	469
16.5.1	Power-On Reset	469
16.5.2	Low-Voltage Reset	469



4.1.2 Modes of Operation

- Normal expanded wide mode
Ports A and B are configured as a 16-bit multiplexed address and data bus and port E provides bus control and status signals. This mode allows 16-bit external memory and peripheral devices to be interfaced to the system.
- Normal expanded narrow mode
Ports A and B are configured as a 16-bit address bus and port A is multiplexed with 8-bit data. Port E provides bus control and status signals. This mode allows 8-bit external memory and peripheral devices to be interfaced to the system.
- Normal single-chip mode
There is no external expansion bus in this mode. The processor program is executed from internal memory. Ports A, B, K, and most of E are available as general-purpose I/O.
- Special single-chip mode
This mode is generally used for debugging single-chip operation, boot-strapping, or security related operations. The active background mode is in control of CPU execution and BDM firmware is waiting for additional serial commands through the BKGD pin. There is no external expansion bus after reset in this mode.
- Emulation expanded wide mode
Developers use this mode for emulation systems in which the users target application is normal expanded wide mode.
- Emulation expanded narrow mode
Developers use this mode for emulation systems in which the users target application is normal expanded narrow mode.
- Special test mode
Ports A and B are configured as a 16-bit multiplexed address and data bus and port E provides bus control and status signals. In special test mode, the write protection of many control bits is lifted so that they can be thoroughly tested without needing to go through reset.
- Special peripheral mode
This mode is intended for Freescale Semiconductor factory testing of the system. The CPU is inactive and an external (tester) bus master drives address, data, and bus control signals.

4.2 External Signal Description

In typical implementations, the MEBI sub-block of the core interfaces directly with external system pins. Some pins may not be bonded out in all implementations.

Table 4-1 outlines the pin names and functions and gives a brief description of their operation reset state of these pins and associated pull-ups or pull-downs is dependent on the mode of operation and on the integration of this block at the chip level (chip dependent).

Table 4-7. MODE Field Descriptions

Field	Description
7:5 MOD[C:A]	Mode Select Bits — These bits indicate the current operating mode. If MODA = 1, then MODC, MODB, and MODA are write never. If MODC = MODA = 0, then MODC, MODB, and MODA are writable with the exception that you cannot change to or from special peripheral mode If MODC = 1, MODB = 0, and MODA = 0, then MODC is write never. MODB and MODA are write once, except that you cannot change to special peripheral mode. From normal single-chip, only normal expanded narrow and normal expanded wide modes are available. See Table 4-8 and Table 4-16.
3 IVIS	Internal Visibility (for both read and write accesses) — This bit determines whether internal accesses generate a bus cycle that is visible on the external bus. Normal: write once Emulation: write never Special: write anytime 0 No visibility of internal bus operations on external bus. 1 Internal bus operations are visible on external bus.
1 EMK	Emulate Port K Normal: write once Emulation: write never Special: write anytime 0 PORTK and DDRK are in the memory map so port K can be used for general-purpose I/O. 1 If in any expanded mode, PORTK and DDRK are removed from the memory map. In single-chip modes, PORTK and DDRK are always in the map regardless of the state of this bit. In special peripheral mode, PORTK and DDRK are never in the map regardless of the state of this bit.
0 EME	Emulate Port E Normal and Emulation: write never Special: write anytime 0 PORTE and DDRE are in the memory map so port E can be used for general-purpose I/O. 1 If in any expanded mode or special peripheral mode, PORTE and DDRE are removed from the memory map. Removing the registers from the map allows the user to emulate the function of these registers externally. In single-chip modes, PORTE and DDRE are always in the map regardless of the state of this bit.

4.4.3.2 Special Operating Modes

There are two special operating modes that correspond to normal operating modes. These operating modes are commonly used in factory testing and system development.

4.4.3.2.1 Special Single-Chip Mode

When the MCU is reset in this mode, the background debug mode is enabled and active. The MCU does not fetch the reset vector and execute application code as it would in other modes. Instead the active background mode is in control of CPU execution and BDM firmware is waiting for additional serial commands through the BKGD pin. When a serial command instructs the MCU to return to normal execution, the system will be configured as described below unless the reset states of internal control registers have been changed through background commands after the MCU was reset.

There is no external expansion bus after reset in this mode. Ports A and B are initially simple bidirectional I/O pins that are configured as high-impedance inputs with internal pull resistors disabled; however, writing to the mode select bits in the MODE register (which is allowed in special modes) can change this after reset. All of the Port E pins (except PE4/ECLK) are initially configured as general purpose high-impedance inputs with internal pull resistors enabled. PE4/ECLK is configured as the E clock output in this mode.

The pins associated with Port E bits 6, 5, 3, and 2 cannot be configured for their alternate functions IPIPE1, IPIPE0, $\overline{\text{LSTRB}}$, and $\text{R}/\overline{\text{W}}$ while the MCU is in single chip modes. In single chip modes, the associated control bits PIPOE, LSTRE and RDWE are reset to zero. Writing the opposite value into these bits in single chip mode does not change the operation of the associated Port E pins.

Port E, bit 4 can be configured for a free-running E clock output by clearing NECLK=0. Typically the only use for an E clock output while the MCU is in single chip modes would be to get a constant speed clock for use in the external application system.

4.4.3.2.2 Special Test Mode

In expanded wide modes, Ports A and B are configured as a 16-bit multiplexed address and data bus and Port E provides bus control and status signals. In special test mode, the write protection of many control bits is lifted so that they can be thoroughly tested without needing to go through reset.

4.4.3.3 Test Operating Mode

There is a test operating mode in which an external master, such as an I.C. tester, can control the on-chip peripherals.

4.4.3.3.1 Peripheral Mode

This mode is intended for factory testing of the MCU. In this mode, the CPU is inactive and an external (tester) bus master drives address, data and bus control signals in through Ports A, B and E. In effect, the whole MCU acts as if it was a peripheral under control of an external CPU. This allows faster testing of on-chip memory and peripherals than previous testing methods. Since the mode control register is not accessible in peripheral mode, the only way to change to another mode is to reset the MCU into a different

NOTE

16-bit misaligned reads and writes are not allowed. If attempted, the BDM will ignore the least significant bit of the address and will assume an even address from the remaining bits.

For hardware data read commands, the external host must wait 150 bus clock cycles after sending the address before attempting to obtain the read data. This is to be certain that valid data is available in the BDM shift register, ready to be shifted out. For hardware write commands, the external host must wait 150 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed. The 150 bus clock cycle delay in both cases includes the maximum 128 cycle delay that can be incurred as the BDM waits for a free cycle before stealing a cycle.

For firmware read commands, the external host should wait 44 bus clock cycles after sending the command opcode and before attempting to obtain the read data. This includes the potential of an extra 7 cycles when the access is external with a narrow bus access (+1 cycle) and / or a stretch (+1, 2, or 3 cycles), (7 cycles could be needed if both occur). The 44 cycle wait allows enough time for the requested data to be made available in the BDM shift register, ready to be shifted out.

NOTE

This timing has increased from previous BDM modules due to the new capability in which the BDM serial interface can potentially run faster than the bus. On previous BDM modules this extra time could be hidden within the serial time.

For firmware write commands, the external host must wait 32 bus clock cycles after sending the data to be written before attempting to send a new command. This is to avoid disturbing the BDM shift register before the write has been completed.

The external host should wait 64 bus clock cycles after a TRACE1 or GO command before starting any new serial command. This is to allow the CPU to exit gracefully from the standard BDM firmware lookup table and resume execution of the user code. Disturbing the BDM shift register prematurely may adversely affect the exit from the standard BDM firmware lookup table.

NOTE

If the bus rate of the target processor is unknown or could be changing, it is recommended that the ACK (acknowledge function) be used to indicate when an operation is complete. When using ACK, the delay times are automated.

Figure 6-6 represents the BDM command structure. The command blocks illustrate a series of eight bit times starting with a falling edge. The bar across the top of the blocks indicates that the BKGD line idles in the high state. The time for an 8-bit command is 8×16 target clock cycles.¹

1. Target clock cycles are cycles measured using the target MCU's serial clock rate. See [Section 6.4.6, "BDM Serial Interface,"](#) and [Section 6.3.2.1, "BDM Status Register \(BDMSTS\),"](#) for information on how serial clock rate is selected.

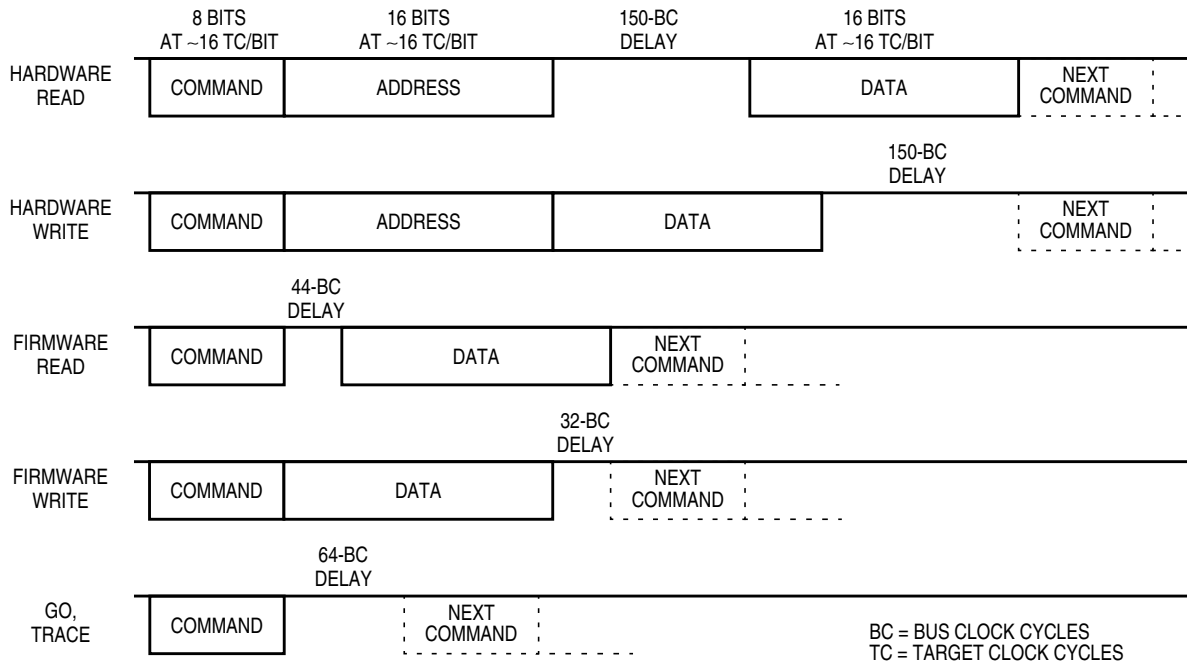


Figure 6-6. BDM Command Structure

6.4.6 BDM Serial Interface

The BDM communicates with external devices serially via the BKGD pin. During reset, this pin is a mode select input which selects between normal and special modes of operation. After reset, this pin becomes the dedicated serial interface pin for the BDM.

The BDM serial interface is timed using the clock selected by the CLKSW bit in the status register see [Section 6.3.2.1, “BDM Status Register \(BDMSTS\).”](#) This clock will be referred to as the target clock in the following explanation.

The BDM serial interface uses a clocking scheme in which the external host generates a falling edge on the BKGD pin to indicate the start of each bit time. This falling edge is sent for every bit whether data is transmitted or received. Data is transferred most significant bit (MSB) first at 16 target clock cycles per bit. The interface times out if 512 clock cycles occur between falling edges from the host.

The BKGD pin is a pseudo open-drain pin and has an weak on-chip active pull-up that is enabled at all times. It is assumed that there is an external pull-up and that drivers connected to BKGD do not typically drive the high level. Because R-C rise time could be unacceptably long, the target system and host provide brief driven-high (speedup) pulses to drive BKGD to a logic 1. The source of this speedup pulse is the host for transmit cases and the target for receive cases.

The timing for host-to-target is shown in [Figure 6-7](#) and that of target-to-host in [Figure 6-8](#) and [Figure 6-9](#). All four cases begin when the host drives the BKGD pin low to generate a falling edge. Because the host and target are operating from separate clocks, it can take the target system up to one full clock cycle to recognize this edge. The target measures delays from this perceived start of the bit time while the host measures delays from the point it actually drove BKGD low to start the bit up to one target clock cycle

8.3 Memory Map and Registers

This section provides a detailed description of all registers accessible in the ATD10B8C.

8.3.1 Module Memory Map

Figure 8-2 gives an overview on all ATD10B8C registers.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000	ATDCTL0	R	0	0	0	0	0	0	0	0
		W								
0x0001	ATDCTL1	R	0	0	0	0	0	0	0	0
		W								
0x0002	ATDCTL2	R	ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
		W								
0x0003	ATDCTL3	R	0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
		W								
0x0004	ATDCTL4	R	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
		W								
0x0005	ATDCTL5	R	DJM	DSGN	SCAN	MULT	0	CC	CB	CA
		W								
0x0006	ATDSTAT0	R	SCF	0	ETORF	FIFOR	0	CC2	CC1	CC0
		W								
0x0007	Unimplemented	R	0	0	0	0	0	0	0	0
		W								
0x0008	ATDTEST0	R	U	U	U	U	U	U	U	U
		W								
0x0009	ATDTEST1	R	U	U	U	U	U	U	U	SC
		W								
0x000A	Unimplemented	R	0	0	0	0	0	0	0	0
		W								
0x000B	ATDSTAT1	R	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
		W								
0x000C	Unimplemented	R	0	0	0	0	0	0	0	0
		W								
0x000D	ATDDIEN	R	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
		W								
0x000E	Unimplemented	R	0	0	0	0	0	0	0	0
		W								
0x000F	PORTAD	R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
		W								

 = Unimplemented or Reserved

Figure 8-2. ATD Register Summary (Sheet 1 of 4)

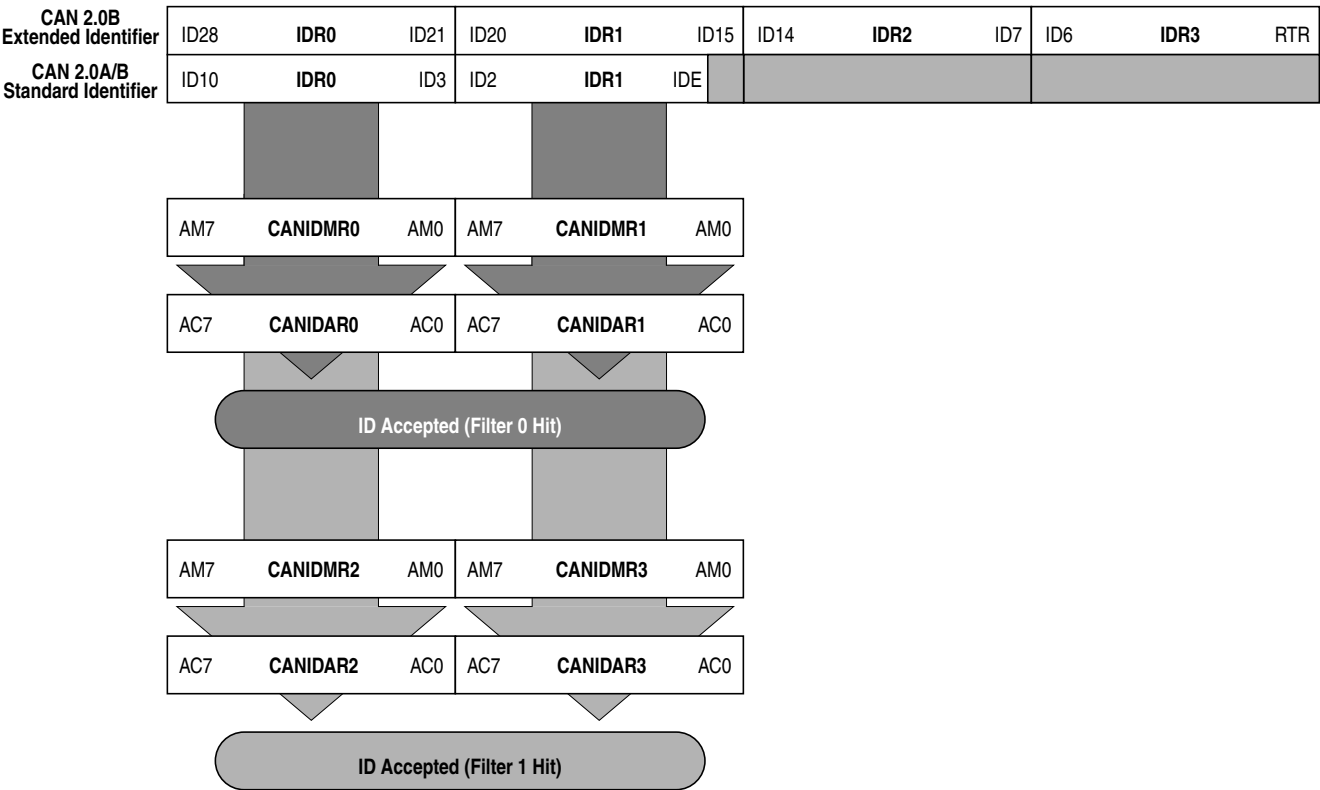


Figure 10-40. 16-bit Maskable Identifier Acceptance Filters

11.3 Memory Map and Register Definition

The CRG contains the registers and associated bits for controlling and monitoring the OSCV2 module.

11.4 Functional Description

The OSCV2 block has two external pins, EXTAL and XTAL. The oscillator input pin, EXTAL, is intended to be connected to either a crystal or an external clock source. The selection of Colpitts oscillator or Pierce oscillator/external clock depends on the XCLKS signal which is sampled during reset. The XTAL pin is an output signal that provides crystal circuit feedback.

A buffered EXTAL signal, OSCCLK, becomes the internal reference clock. To improve noise immunity, the oscillator is powered by the V_{DDPLL} and V_{SSPLL} power supply pins.

The Pierce oscillator can be used for higher frequencies compared to the low power Colpitts oscillator.

11.4.1 Amplitude Limitation Control (ALC)

The Colpitts oscillator is equipped with a feedback system which does not waste current by generating harmonics. Its configuration is “Colpitts oscillator with translated ground.” The transconductor used is driven by a current source under the control of a peak detector which will measure the amplitude of the AC signal appearing on EXTAL node in order to implement an amplitude limitation control (ALC) loop. The ALC loop is in charge of reducing the quiescent current in the transconductor as a result of an increase in the oscillation amplitude. The oscillation amplitude can be limited to two values. The normal amplitude which is intended for non power saving modes and a small amplitude which is intended for low power operation modes. Please refer to the CRG block description chapter for the control and assignment of the amplitude value to operation modes.

11.4.2 Clock Monitor (CM)

The clock monitor circuit is based on an internal resistor-capacitor (RC) time delay so that it can operate without any MCU clocks. If no OSCCLK edges are detected within this RC time delay, the clock monitor indicates a failure which asserts self clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated. The clock monitor function is enabled/disabled by the CME control bit, described in the CRG block description chapter.

11.5 Interrupts

OSCV2 contains a clock monitor, which can trigger an interrupt or reset. The control bits and status bits for the clock monitor are described in the CRG block description chapter.

13.4.3 Transmitter

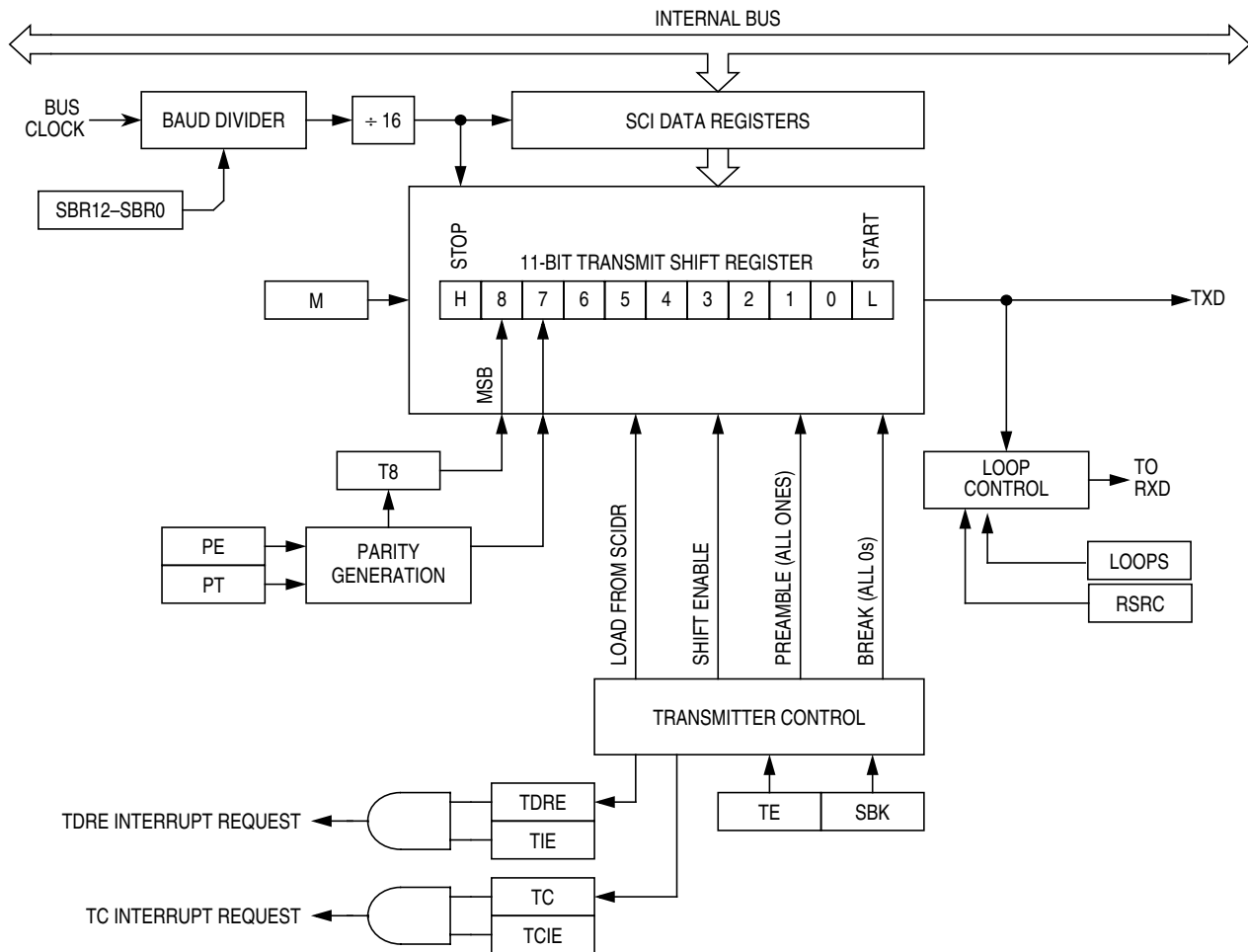


Figure 13-11. Transmitter Block Diagram

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

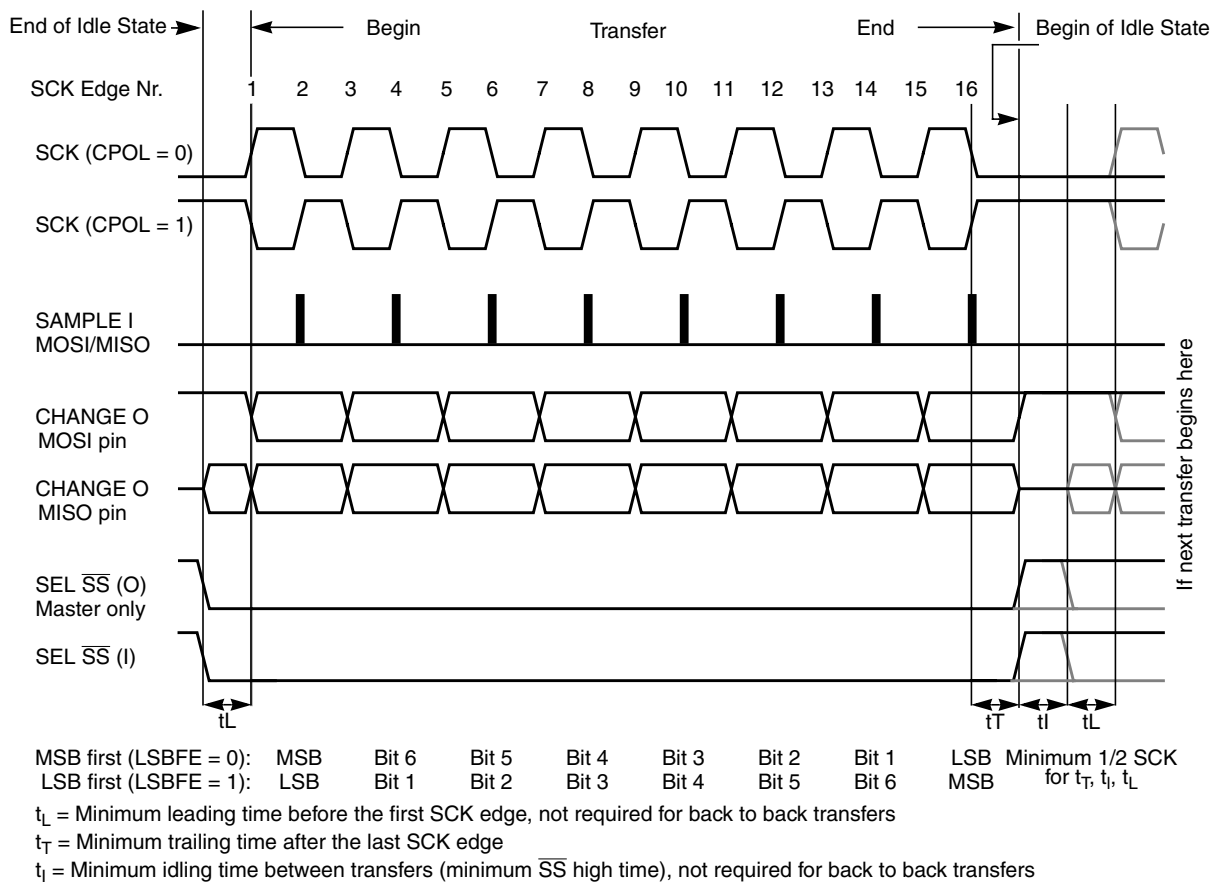


Figure 14-10. SPI Clock Format 1 (CPHA = 1)

14.4.4 SPI Baud Rate Generation

Baud rate generation consists of a series of divider stages. Six bits in the SPI Baud Rate register (SPPR2, SPPR1, SPPR0, SPR2, SPR1, and SPR0) determine the divisor to the SPI module clock which results in the SPI baud rate.

The SPI clock rate is determined by the product of the value in the baud rate preselection bits (SPPR2–SPPR0) and the value in the baud rate selection bits (SPR2–SPR0). The module clock divisor equation is shown in [Figure 14-11](#)

When all bits are clear (the default condition), the SPI module clock is divided by 2. When the selection bits (SPR2–SPR0) are 001 and the preselection bits (SPPR2–SPPR0) are 000, the module clock divisor becomes 4. When the selection bits are 010, the module clock divisor becomes 8 etc.

When the preselection bits are 001, the divisor determined by the selection bits is multiplied by 2. When the preselection bits are 010, the divisor is multiplied by 3, etc. See [Table 14-7](#) for baud rate calculations for all bit conditions, based on a 25-MHz bus clock. The two sets of selects allows the clock to be divided by a non-power of two to achieve other baud rates such as divide by 6, divide by 10, etc.

16.1.3 Block Diagram

Figure 16-1 shows the function principle of VREG3V3V2 by means of a block diagram. The regulator core REG consists of two parallel sub-blocks, REG1 and REG2, providing two independent output voltages.

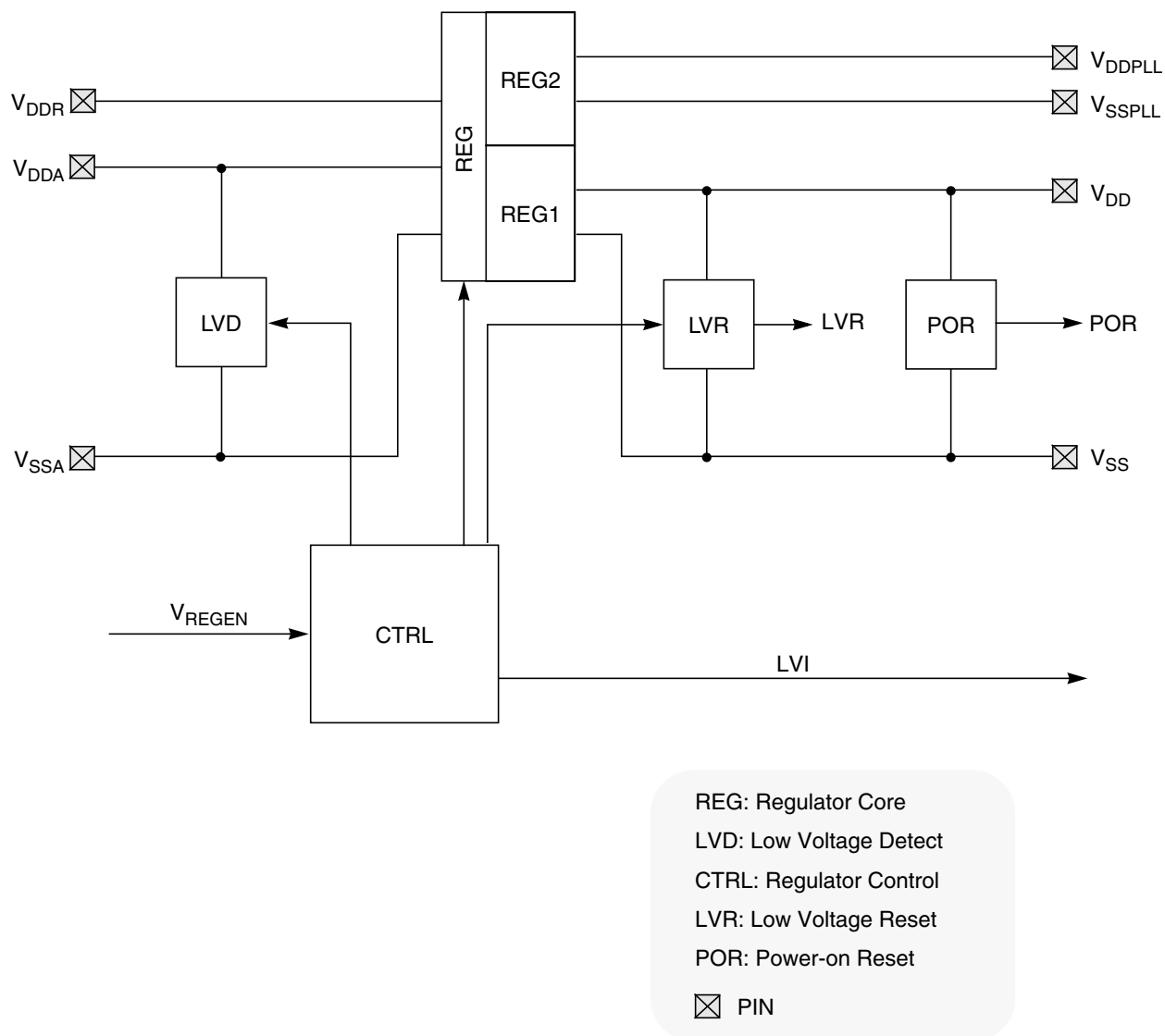


Figure 16-1. VREG3V3 Block Diagram

17.3 Memory Map and Registers

This section describes the [FTS16K](#) memory map and registers.

17.3.1 Module Memory Map

The [FTS16K](#) memory map is shown in [Figure 17-2](#). The HCS12 architecture places the Flash array addresses between [0xC000](#) and [0xFFFF](#). The content of the HCS12 Core PPAGE register is used to map the logical page ranging from address [0x8000](#) to [0xBFFF](#) to a physical 16K byte page in the Flash array memory.¹ The FPROT register (see [Section 17.3.2.5](#)) can be set to globally protect the entire Flash array or one growing downward from the Flash array end address. The higher address area is mainly targeted to hold the boot loader code since it covers the vector space. Default protection settings as well as security information that allows the MCU to restrict access to the Flash module are stored in the Flash configuration field described in [Table 17-1](#).

Table 17-1. Flash Configuration Field

Flash Address	Size (bytes)	Description
0xFF00–0xFF07	8	Backdoor Key to unlock security
0xFF08–0xFF0C	5	Reserved
0xFF0D	1	Flash Protection byte Refer to Section 17.3.2.5, “Flash Protection Register (FPROT)”
0xFF0E	1	Reserved
0xFF0F	1	Flash Security/Options byte Refer to Section 17.3.2.2, “Flash Security Register (FSEC)”

1. By placing 0x3F in the HCS12 Core PPAGE register, the 16 Kbyte page can be seen twice in the MCU memory map.

Table 18-12. 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.

18.3.2.6 Flash Status Register (FSTAT)

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

Module Base + 0x0005

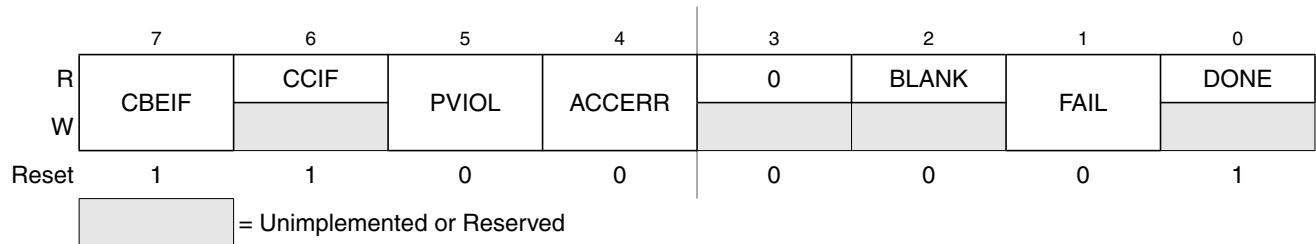


Figure 18-10. Flash Status Register (FSTAT)

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 18-13. 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 18-26). 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 18-26). 0 Command in progress 1 All commands are completed

18.4.1.2 Command Write Sequence

The Flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the ACCERR and PVIOL flags in the FSTAT register must be clear and the CBEIF flag should be tested to determine the state of the address, data, and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be started. If the CBEIF flag is clear, indicating the buffers are not available, a new command write sequence will overwrite the contents of the address, data, and command buffers.

A command write sequence consists of three steps which must be strictly adhered to with writes to the Flash module not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the Flash array memory.
2. Write a valid command to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the command.

The address written in step 1 will be stored in the FADDR registers and the data will be stored in the FDATA registers. When the CBEIF flag is cleared in step 3, the CCIF flag is cleared by the Flash command controller indicating that the command was successfully launched. For all command write sequences, the CBEIF flag will set after the CCIF flag is cleared indicating that the address, data, and command buffers are ready for a new command write sequence to begin. A buffered command will wait for the active operation to be completed before being launched. Once a command is launched, the completion of the command operation is indicated by the setting of the CCIF flag in the FSTAT register. The CCIF flag will set upon completion of all active and buffered commands.

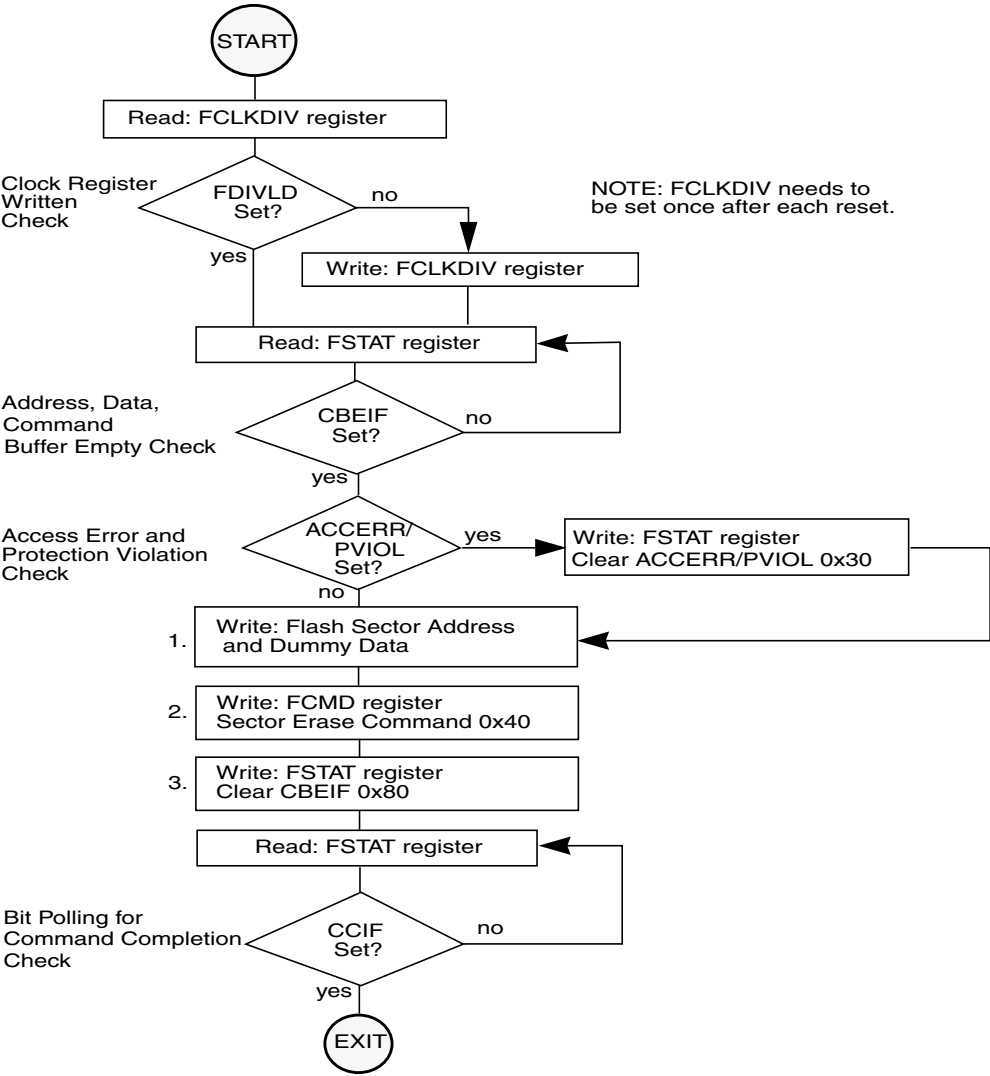


Figure 18-24. Example Sector Erase Command Flow

20.4.1.3.1 Erase Verify Command

The erase verify operation will verify that a Flash array is erased.

An example flow to execute the erase verify operation is shown in [Figure 20-24](#). The erase verify command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the erase verify command. The address and data written will be ignored.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the erase verify command.

After launching the erase verify command, the CCIF flag in the FSTAT register will set after the operation has completed unless a new command write sequence has been buffered. Upon completion of the erase verify operation, the BLANK flag in the FSTAT register will be set if all addresses in the Flash array are verified to be erased. If any address in the Flash array is not erased, the erase verify operation will terminate and the BLANK flag in the FSTAT register will remain clear.

20.4.1.4 Illegal Flash Operations

20.4.1.4.1 Access Error

The ACCERR flag in the FSTAT register will be set during the command write sequence if any of the following illegal Flash operations are performed causing the command write sequence to immediately abort:

1. Writing to the Flash address space before initializing the FCLKDIV register
2. Writing a misaligned word or a byte to the valid Flash address space
3. Writing to the Flash address space while CBEIF is not set
4. Writing a second word to the Flash address space before executing a program or erase command on the previously written word
5. Writing to any Flash register other than FCMD after writing a word to the Flash address space
6. Writing a second command to the FCMD register before executing the previously written command
7. Writing an invalid command to the FCMD register
8. Writing to any Flash register other than FSTAT (to clear CBEIF) after writing to the FCMD register
9. The part enters stop mode and a program or erase command is in progress. The command is aborted and any pending command is killed
10. When security is enabled, a command other than mass erase originating from a non-secure memory or from the background debug mode is written to the FCMD register
11. A 0 is written to the CBEIF bit in the FSTAT register to abort a command write sequence.

The ACCERR flag will not be set if any Flash register is read during the command write sequence. If the Flash array is read during execution of an algorithm (CCIF=0), the Flash module will return invalid data and the ACCERR flag will not be set. If an ACCERR flag is set in the FSTAT register, the Flash command controller is locked. It is not possible to launch another command until the ACCERR flag is cleared.

20.4.1.4.2 Protection Violation

The PVIOL flag in the FSTAT register will be set during the command write sequence after the word write to the Flash address space if any of the following illegal Flash operations are performed, causing the command write sequence to immediately abort:

1. Writing a Flash address to program in a protected area of the Flash array (see [Section 20.3.2.5](#)).
2. Writing a Flash address to erase in a protected area of the Flash array.
3. Writing the mass erase command to the FCMD register while any protection is enabled.

If the PVIOL flag is set, the Flash command controller is locked. It is not possible to launch another command until the PVIOL flag is cleared.

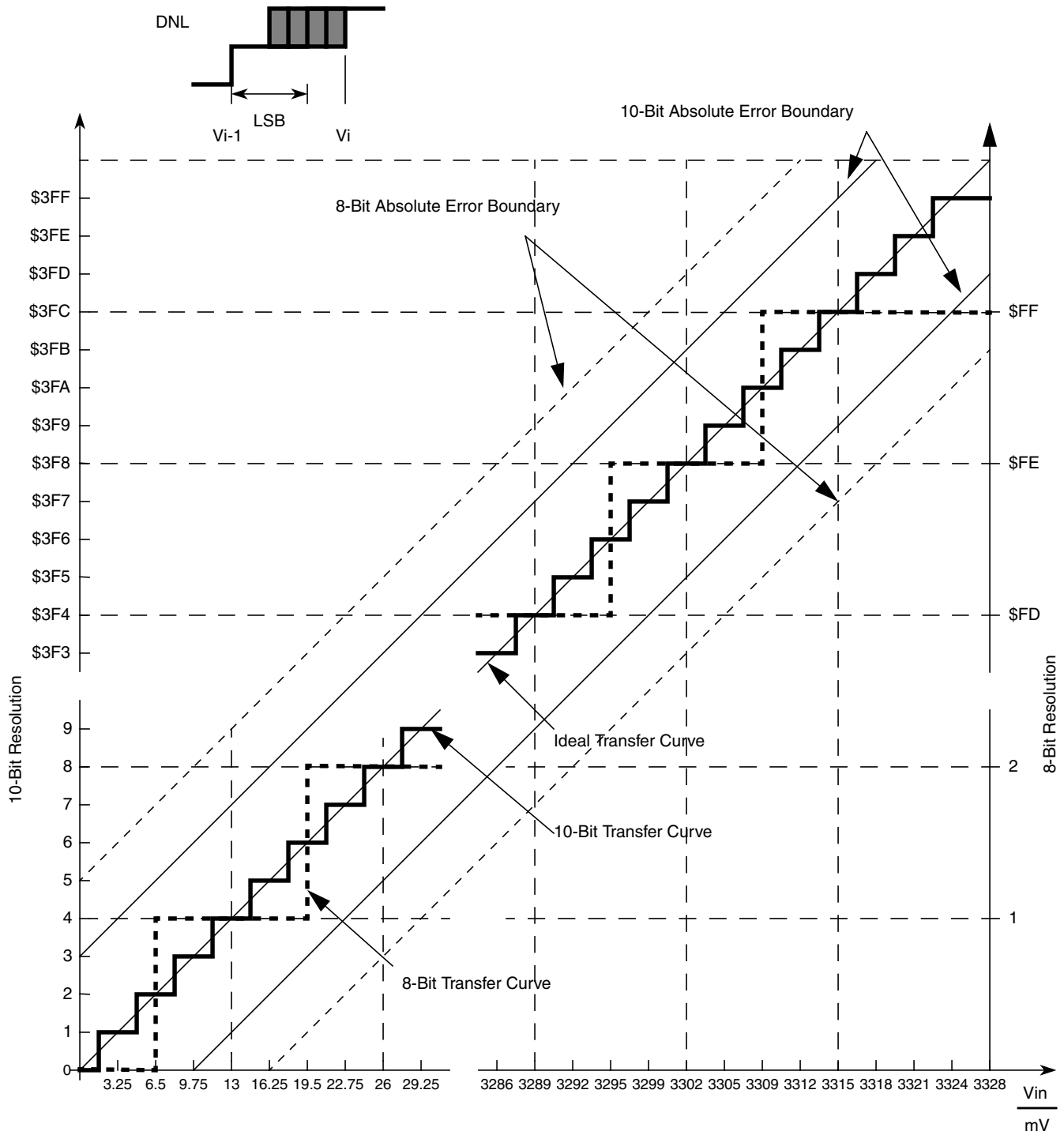


Figure A-1. ATD Accuracy Definitions

NOTE

Figure A-1 shows only definitions, for specification values refer to Table A-12.