

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	96KB (96K 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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12gc96cpbe

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

16.6	Interrupts	469
16.6.1	LVI — Low-Voltage Interrupt	469

Chapter 17

16 Kbyte Flash Module (S12FTS16KV1)

17.1	Introduction	471
17.1.1	Glossary	471
17.1.2	Features	471
17.1.3	Modes of Operation	472
17.1.4	Block Diagram	472
17.2	External Signal Description	472
17.3	Memory Map and Registers	473
17.3.1	Module Memory Map	473
17.3.2	Register Descriptions	475
17.4	Functional Description	486
17.4.1	Flash Command Operations	486
17.4.2	Operating Modes	500
17.4.3	Flash Module Security	500
17.4.4	Flash Reset Sequence	502
17.4.5	Interrupts	502

Chapter 18

32 Kbyte Flash Module (S12FTS32KV1)

18.1	Introduction	503
18.1.1	Glossary	503
18.1.2	Features	503
18.1.3	Modes of Operation	504
18.1.4	Block Diagram	504
18.2	External Signal Description	504
18.3	Memory Map and Registers	505
18.3.1	Module Memory Map	505
18.3.2	Register Descriptions	508
18.4	Functional Description	520
18.4.1	Flash Command Operations	520
18.4.2	Operating Modes	534
18.4.3	Flash Module Security	534
18.4.4	Flash Reset Sequence	536
18.4.5	Interrupts	536

Chapter 19

64 Kbyte Flash Module (S12FTS64KV4)

19.1	Introduction	537
19.1.1	Glossary	537
19.1.2	Features	537

4.3.2.3 Data Direction Register A (DDRA)

Module Base + 0x0002
Starting address location affected by INITRG register setting.

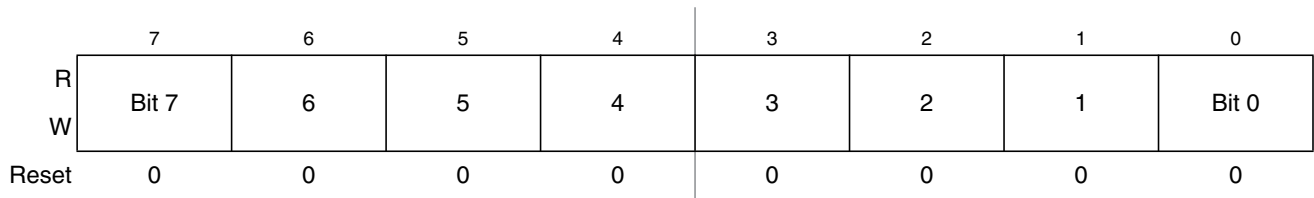


Figure 4-4. Data Direction Register A (DDRA)

Read: Anytime when register is in the map

Write: Anytime when register is in the map

This register controls the data direction for port A. When port A is operating as a general-purpose I/O port, DDRA determines the primary direction for each port A pin. A 1 causes the associated port pin to be an output and a 0 causes the associated pin to be a high-impedance input. The value in a DDR bit also affects the source of data for reads of the corresponding PORTA register. If the DDR bit is 0 (input) the buffered pin input state is read. If the DDR bit is 1 (output) the associated port data register bit state is read.

This register is not in the on-chip memory map in expanded and special peripheral modes. Therefore, these accesses will be echoed externally. It is reset to 0x00 so the DDR does not override the three-state control signals.

Table 4-3. DDRA Field Descriptions

Field	Description
7:0 DDRA	Data Direction Port A 0 Configure the corresponding I/O pin as an input 1 Configure the corresponding I/O pin as an output

7.3.2.2 Debug Status and Control Register (DBGSC)

Module Base + 0x0021

Starting address location affected by INITRG register setting.

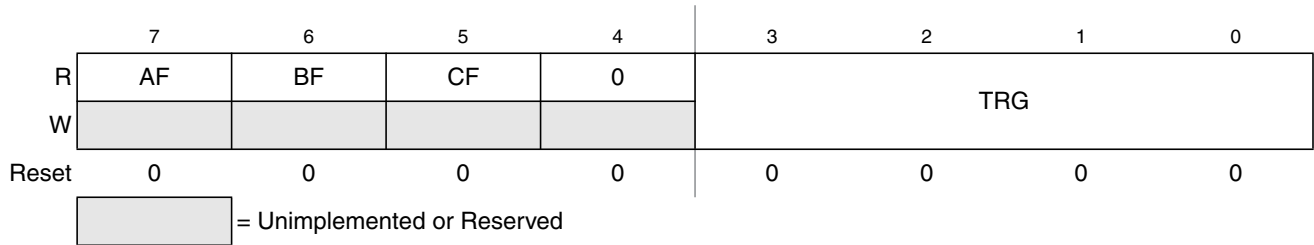


Figure 7-5. Debug Status and Control Register (DBGSC)

Table 7-5. DBGSC Field Descriptions

Field	Description
7 AF	Trigger A Match Flag — The AF bit indicates if trigger A match condition was met since arming. This bit is cleared when ARM in DBGSC1 is written to a 1 or on any write to this register. 0 Trigger A did not match 1 Trigger A match
6 BF	Trigger B Match Flag — The BF bit indicates if trigger B match condition was met since arming. This bit is cleared when ARM in DBGSC1 is written to a 1 or on any write to this register. 0 Trigger B did not match 1 Trigger B match
5 CF	Comparator C Match Flag — The CF bit indicates if comparator C match condition was met since arming. This bit is cleared when ARM in DBGSC1 is written to a 1 or on any write to this register. 0 Comparator C did not match 1 Comparator C match
3:0 TRG	Trigger Mode Bits — The TRG bits select the trigger mode of the DBG module as shown Table 7-6 . See Section 7.4.2.5, “Trigger Modes,” for more detail.

Table 7-6. Trigger Mode Encoding

TRG Value	Meaning
0000	A only
0001	A or B
0010	A then B
0011	Event only B
0100	A then event only B
0101	A and B (full mode)
0110	A and Not B (full mode)
0111	Inside range
1000	Outside range
1001 ↓ 1111	Reserved (Defaults to A only)

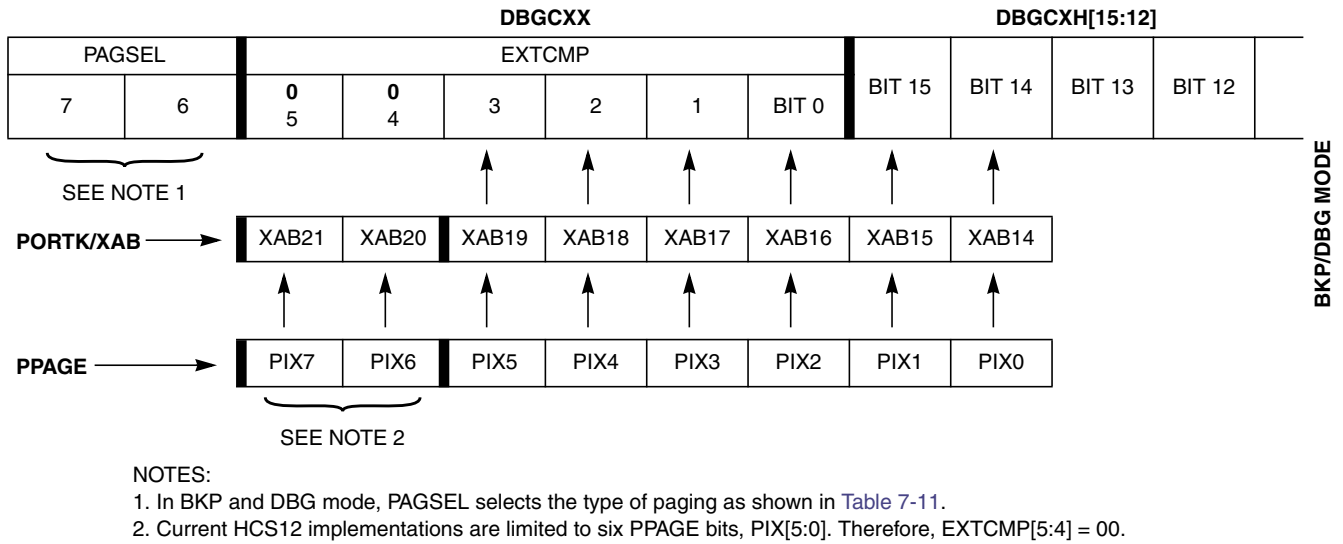


Figure 7-10. Comparator C Extended Comparison in BKP/DBG Mode

7.3.2.6 Debug Comparator C Register (DBGCC)

Module Base + 0x0026

Starting address location affected by INITRG register setting.

	15	14	13	12	11	10	9	8
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 7-11. Debug Comparator C Register High (DBGCCCH)

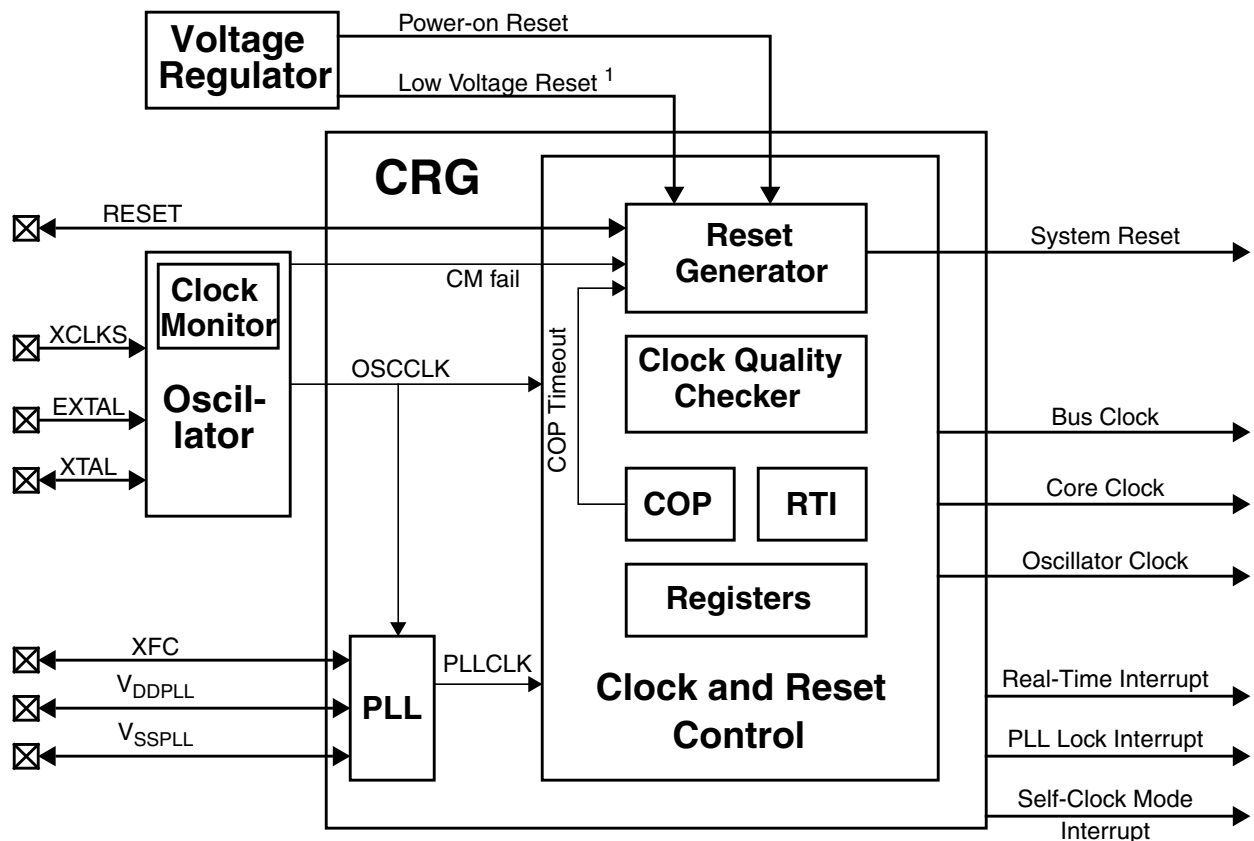
Module Base + 0x0027

Starting address location affected by INITRG register setting.

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

= Unimplemented or Reserved

Figure 7-12. Debug Comparator C Register Low (DBGCCCL)



¹ Refer to the device overview section for availability of the low-voltage reset feature.

Figure 9-1. CRG Block Diagram

9.2 External Signal Description

This section lists and describes the signals that connect off chip.

9.2.1 V_{DDPLL} , V_{SSPLL} — PLL Operating Voltage, PLL Ground

These pins provides operating voltage (V_{DDPLL}) and ground (V_{SSPLL}) for the PLL circuitry. This allows the supply voltage to the PLL to be independently bypassed. Even if PLL usage is not required V_{DDPLL} and V_{SSPLL} must be connected properly.

9.2.2 XFC — PLL Loop Filter Pin

A passive external loop filter must be placed on the XFC pin. The filter is a second-order, low-pass filter to eliminate the VCO input ripple. The value of the external filter network and the reference frequency determines the speed of the corrections and the stability of the PLL. Refer to the device overview chapter for calculation of PLL loop filter (XFC) components. If PLL usage is not required the XFC pin must be tied to V_{DDPLL} .

12.4.2.4 PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (reference [Figure 12-34](#) for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in [Figure 12-35](#). When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in [Figure 12-35](#) and described in [Section 12.4.2.5, “Left Aligned Outputs,”](#) and [Section 12.4.2.6, “Center Aligned Outputs.”](#)

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

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. When the channel is disabled ($PWMEx = 0$), the counter stops. When a channel becomes enabled ($PWMEx = 1$), the associated PWM counter continues from the count in the $PWMCNTx$ register. This allows the waveform to resume when the channel is re-enabled. When the channel is disabled, writing 0 to the period register will cause the counter to reset on the next selected clock.

NOTE

If the user wants to start a new “clean” PWM waveform without any “history” from the old waveform, the user must write to channel counter ($PWMCNTx$) prior to enabling the PWM channel ($PWMEx = 1$).

Generally, writes to the counter are done prior to enabling a channel to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled except that the new period is started immediately with the output set according to the polarity bit.

NOTE

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

The counter is 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).

Table 12-11. PWM Timer Counter Conditions

Counter Clears (0x0000)	Counter Counts	Counter Stops
When $PWMCNTx$ register written to any value	When PWM channel is enabled ($PWMEx = 1$). Counts from last value in $PWMCNTx$.	When PWM channel is disabled ($PWMEx = 0$)
Effective period ends		



In Figure 13-16, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

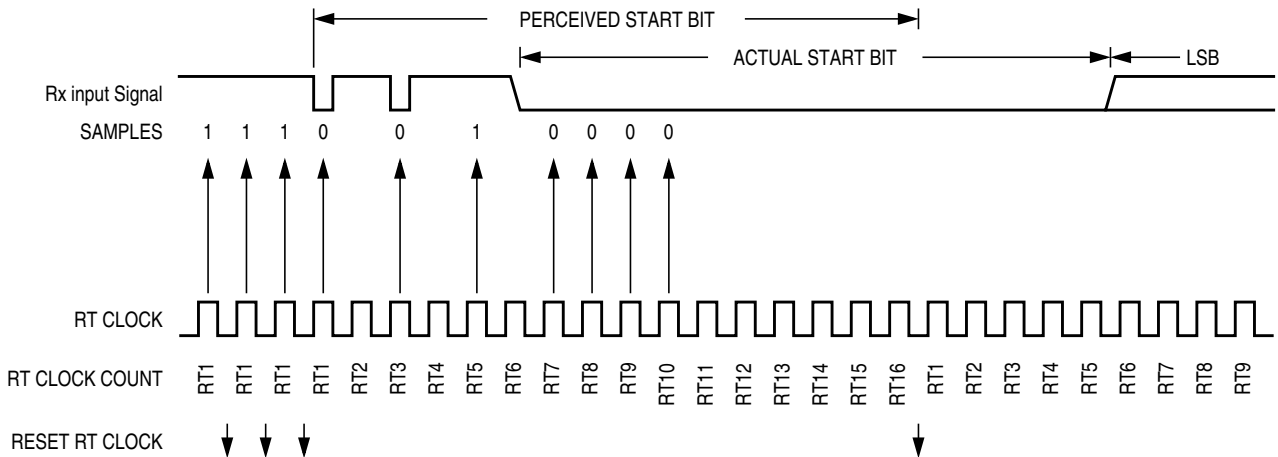


Figure 13-16. Start Bit Search Example 3

Figure 13-17 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

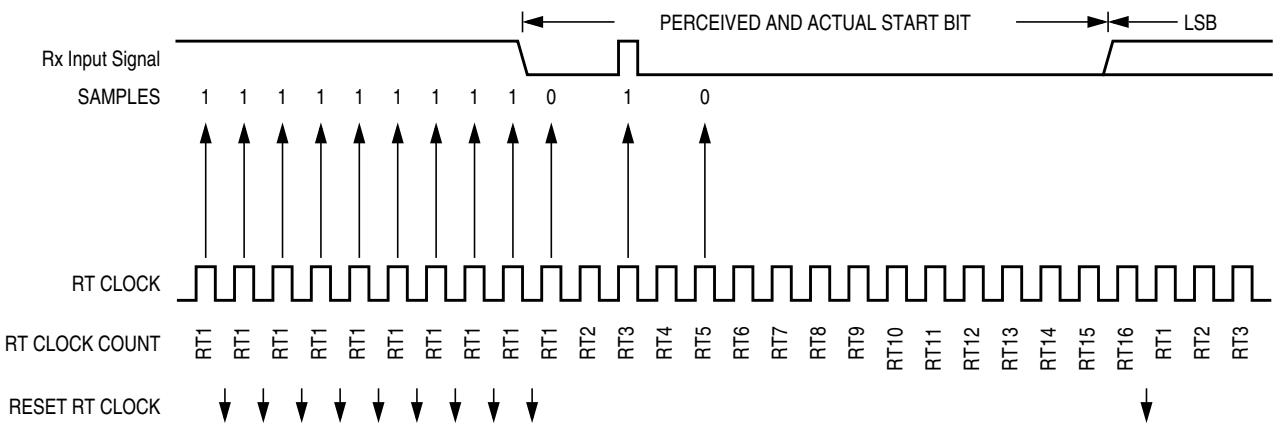


Figure 13-17. Start Bit Search Example 4

With the misaligned character shown in Figure 13-20, the receiver counts 167 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((167 - 160) / 167) \times 100 = 4.19\%$$

13.4.4.5.2 Fast Data Tolerance

Figure 13-21 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

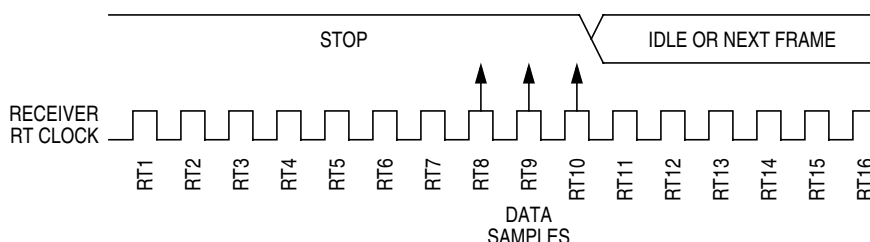


Figure 13-21. Fast Data

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 10 RTr cycles = 154 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 13-21, the receiver counts 154 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((160 - 154) / 160) \times 100 = 3.75\%$$

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 10 RTr cycles = 170 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 13-21, the receiver counts 170 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((176 - 170) / 176) \times 100 = 3.40\%$$

13.4.4.6 Receiver Wakeup

To enable the SCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled. The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

Chapter 15

Timer Module (TIM16B8CV1) Block Description

Table 15-1. Revision History

Version Number	Revision Dates	Effective Date	Author	Description of Changes
01.03	06 Feb 2006	06 Feb 2006	S. Chinnam	Corrected the type at 0x006 and later in the document from TSCR2 and TSCR1
01.04	08 July 2008	08 July 2008	S. Chinnam	Revised flag clearing procedure, whereby TEN bit must be set when clearing flags.
01.05	05 May 2010	05 May 2010	Ame Wang	-in 15.3.2.8/15-446,add Table 15-11 -in 15.3.2.11/15-450,TCRE bit description part,add Note -in 15.4.3/15-459,add Figure 15-29

15.1 Introduction

The basic timer consists of a 16-bit, software-programmable counter driven by a seven-stage programmable prescaler.

This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

This timer contains 8 complete input capture/output compare channels and one pulse accumulator. The input capture function is used to detect a selected transition edge and record the time. The output compare function is used for generating output signals or for timer software delays. The 16-bit pulse accumulator is used to operate as a simple event counter or a gated time accumulator. The pulse accumulator shares timer channel 7 when in event mode.

A full access for the counter registers or the input capture/output compare registers should take place in one clock cycle. Accessing high byte and low byte separately for all of these registers may not yield the same result as accessing them in one word.

15.1.1 Features

The TIM16B8CV1 includes these distinctive features:

- Eight input capture/output compare channels.
- Clock prescaling.
- 16-bit counter.
- 16-bit pulse accumulator.

Module Base + 0x0009

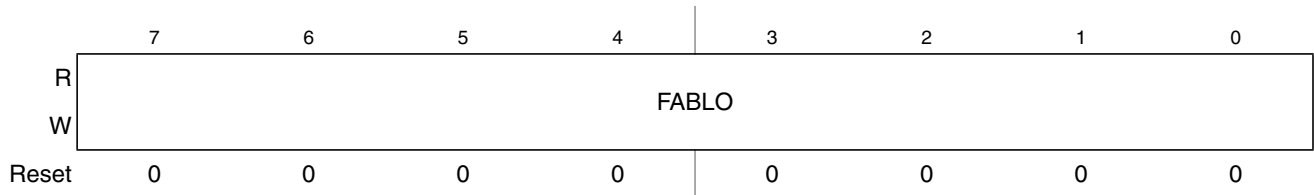


Figure 17-14. Flash Address Low Register (FADDRLO)

In normal modes, all FABHI and FABLO bits read 0 and are not writable. In special modes, the FABHI and FABLO bits are readable and writable. For sector erase, the MCU address bits [8:0] are ignored. For mass erase, any address within the Flash array is valid to start the command.

17.3.2.10 Flash Data Register (FDATA)

FDATAHI and FDATALO are the Flash data registers.

Module Base + 0x000A

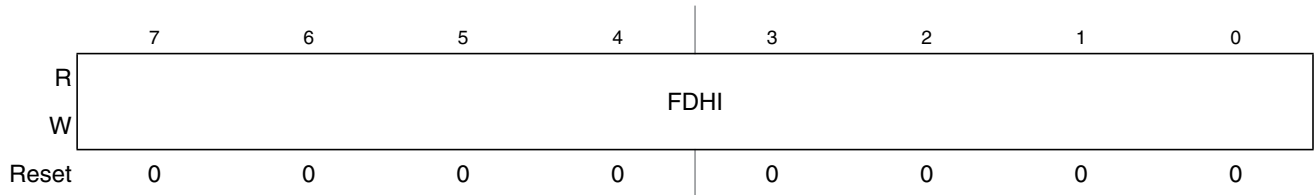


Figure 17-15. Flash Data High Register (FDATAHI)

Module Base + 0x000B

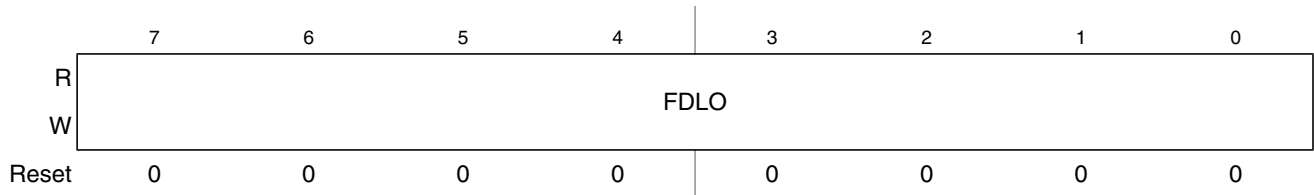


Figure 17-16. Flash Data Low Register (FDATALO)

In normal modes, all FDATAHI and FDATALO bits read 0 and are not writable. In special modes, all FDATAHI and FDATALO bits are readable and writable when writing to an address within the Flash address range.

17.3.2.11 RESERVED3

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

17.4.1.4 Illegal Flash Operations

17.4.1.4.1 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.

17.4.1.4.2 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 17.3.2.5](#)).
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.

Module Base + 0x000C

	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

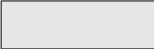
 = Unimplemented or Reserved

Figure 18-17. RESERVED3

All bits read 0 and are not writable.

18.3.2.12 RESERVED4

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

Module Base + 0x000D

	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 18-18. RESERVED4

All bits read 0 and are not writable.

18.3.2.13 RESERVED5

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

Module Base + 0x000E

	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 18-19. RESERVED5

All bits read 0 and are not writable.

18.3.2.14 RESERVED6

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

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.

addresses sequentially starting with 0xFF00–0xFF01 and ending with 0xFF06–0xFF07. The values 0x0000 and 0xFFFF are not permitted as keys. When the KEYACC bit is set, reads of the Flash array will return invalid data.

The user code stored in the Flash array must have a method of receiving the backdoor key from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If KEYEN[1:0] = 1:0 in the FSEC register, the MCU can be unsecured by the backdoor key access sequence described below:

1. Set the KEYACC bit in the FCNFG register
2. Write the correct four 16-bit words to Flash addresses 0xFF00–0xFF07 sequentially starting with 0xFF00
3. Clear the KEYACC bit in the FCNFG register
4. If all four 16-bit words match the backdoor key stored in Flash addresses 0xFF00–0xFF07, the MCU is unsecured and bits SEC[1:0] in the FSEC register are forced to the unsecure state of 1:0

The backdoor key access sequence is monitored by the internal security state machine. An illegal operation during the backdoor key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new backdoor key access sequence to be attempted. The following illegal operations will lock the security state machine:

1. If any of the four 16-bit words does not match the backdoor key programmed in the Flash array
2. If the four 16-bit words are written in the wrong sequence
3. If more than four 16-bit words are written
4. If any of the four 16-bit words written are 0x0000 or 0xFFFF
5. If the KEYACC bit does not remain set while the four 16-bit words are written

After the backdoor key access sequence has been correctly matched, the MCU will be unsecured. The Flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the four word backdoor key by programming bytes 0xFF00–0xFF07 of the Flash configuration field.

The security as defined in the Flash security/options byte at address 0xFF0F is not changed by using the backdoor key access sequence to unsecure. The backdoor key stored in addresses 0xFF00–0xFF07 is unaffected by the backdoor key access sequence. After the next reset sequence, the security state of the Flash module is determined by the Flash security/options byte at address 0xFF0F. The backdoor key access sequence has no effect on the program and erase protection defined in the FPROT register.

It is not possible to unsecure the MCU in special single chip mode by executing the backdoor key access sequence in background debug mode.

then FCLKDIV register bits PRDIV8 and FDIV[5:0] are to be set as described in [Figure 20-23](#).

For example, if the oscillator clock frequency is 950 kHz and the bus clock is 10 MHz, FCLKDIV bits FDIV[5:0] should be set to 4 (000100) and bit PRDIV8 set to 0. The resulting FCLK is then 190 kHz. As a result, the Flash algorithm timings are increased over optimum target by:

$$(200 - 190) / 200 \times 100 = 5\%$$

Command execution time will increase proportionally with the period of FCLK.

CAUTION

Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the Flash array cannot be performed if the bus clock runs at less than 1 MHz. Programming or erasing the Flash array with an input clock < 150 kHz should be avoided. Setting FCLKDIV to a value such that FCLK < 150 kHz can destroy the Flash array due to overstress. Setting FCLKDIV to a value such that $(1/\text{FCLK} + T_{\text{bus}}) < 5\mu\text{s}$ can result in incomplete programming or erasure of the Flash array cells.

If the FCLKDIV register is written, the bit FDIVLD is set automatically. If the FDIVLD bit is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written to, the Flash command loaded during a command write sequence will not execute and the ACCERR flag in the FSTAT register will set.

21.3.2.4 Flash Configuration Register (FCNFG)

The FCNFG register enables the Flash interrupts and gates the security backdoor key writes.

Module Base + 0x0003

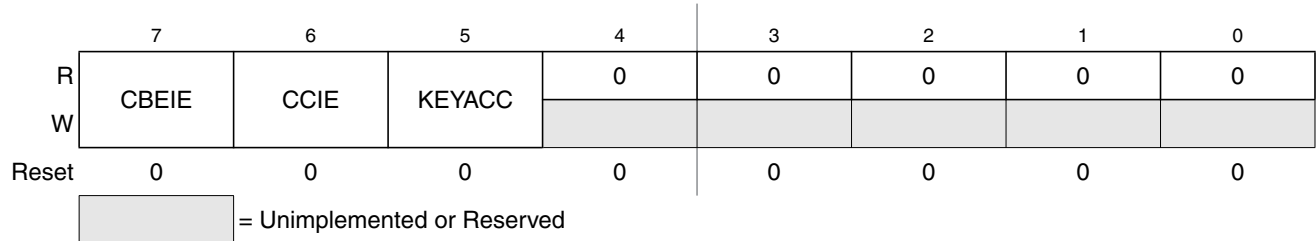


Figure 21-7. Flash Configuration Register (FCNFG)

CBEIE, CCIE, and KEYACC are readable and writable while remaining bits read 0 and are not writable. KEYACC is only writable if the KEYEN bit in the FSEC register is set to the enabled state (see [Section 21.3.2.2](#)).

Table 21-7. FCNFG Field Descriptions

Field	Description
7 CBEIE	Command Buffer Empty Interrupt Enable — The CBEIE bit enables the interrupts in case of an empty command buffer in the Flash module. 0 Command Buffer Empty interrupts disabled 1 An interrupt will be requested whenever the CBEIF flag is set (see Section 21.3.2.6)
6 CCIE	Command Complete Interrupt Enable — The CCIE bit enables the interrupts in case of all commands being completed in the Flash module. 0 Command Complete interrupts disabled 1 An interrupt will be requested whenever the CCIF flag is set (see Section 21.3.2.6)
5 KEYACC	Enable Security Key Writing. 0 Flash writes are interpreted as the start of a command write sequence 1 Writes to the Flash array are interpreted as a backdoor key while reads of the Flash array return invalid data

21.3.2.5 Flash Protection Register (FPROT)

The FPROT register defines which Flash sectors are protected against program or erase.

Module Base + 0x0004

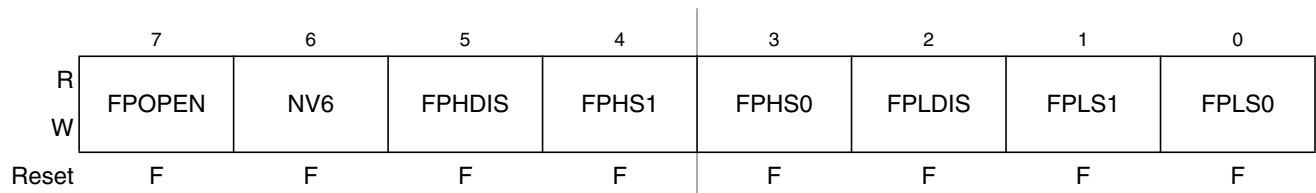


Figure 21-8. Flash Protection Register (FPROT)

The FPROT register is readable in normal and special modes. FPOPEN can only be written from a 1 to a 0. [FPLS\[1:0\]](#) can be written anytime until [FPLDIS](#) is cleared. [FPHS\[1:0\]](#) can be written anytime until

21.4.1.3.3 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 21-24. 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.

21.4.1.3.4 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 21-25](#). 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.