

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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12gc128cpbe">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12gc128cpbe</a>

the Flash EEPROM memory in the memory map (ROMCTL). At the rising edge of  $\overline{\text{RESET}}$ , the state of this pin is latched to the ROMON bit.

- PP6 = 1 in emulation modes equates to ROMON = 0 (ROM space externally mapped)
- PP6 = 0 in expanded modes equates to ROMON = 0 (ROM space externally mapped)

#### 1.3.4.19 PP[5:0] / KWP[5:0] / PW[5:0] — Port P I/O Pins [5:0]

PP[5:0] are general purpose input or output pins, shared with the keypad interrupt function. When configured as inputs, they can generate interrupts causing the MCU to exit stop or wait mode.

PP[5:0] are also shared with the PWM output signals, PW[5:0]. Pins PP[2:0] are only available in the 80-pin package version. Pins PP[4:3] are not available in the 48-pin package version.

#### 1.3.4.20 PJ[7:6] / KWJ[7:6] — Port J I/O Pins [7:6]

PJ[7:6] are general purpose input or output pins, shared with the keypad interrupt function. When configured as inputs, they can generate interrupts causing the MCU to exit stop or wait mode. These pins are not available in the 48-pin package version nor in the 52-pin package version.

#### 1.3.4.21 PM5 / SCK — Port M I/O Pin 5

PM5 is a general purpose input or output pin and also the serial clock pin SCK for the serial peripheral interface (SPI).

#### 1.3.4.22 PM4 / MOSI — Port M I/O Pin 4

PM4 is a general purpose input or output pin and also the master output (during master mode) or slave input (during slave mode) pin for the serial peripheral interface (SPI).

#### 1.3.4.23 PM3 / $\overline{\text{SS}}$ — Port M I/O Pin 3

PM3 is a general purpose input or output pin and also the slave select pin  $\overline{\text{SS}}$  for the serial peripheral interface (SPI).

#### 1.3.4.24 PM2 / MISO — Port M I/O Pin 2

PM2 is a general purpose input or output pin and also the master input (during master mode) or slave output (during slave mode) pin for the serial peripheral interface (SPI).

#### 1.3.4.25 PM1 / TXCAN — Port M I/O Pin 1

PM1 is a general purpose input or output pin and the transmit pin, TXCAN, of the CAN module if available.

#### 1.3.4.26 PM0 / RXCAN — Port M I/O Pin 0

PM0 is a general purpose input or output pin and the receive pin, RXCAN, of the CAN module if available.

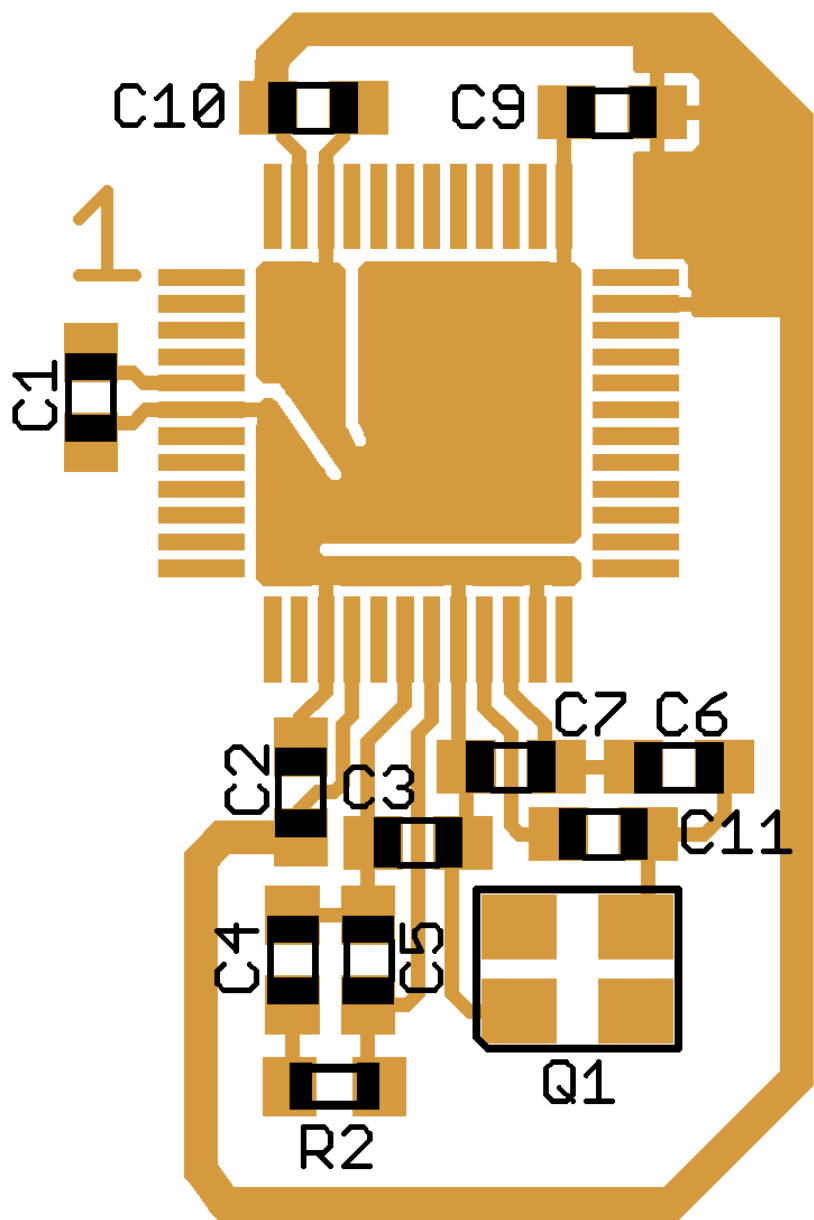


Figure 1-15. Recommended PCB Layout (48 LQFP) Colpitts Oscillator

Table 4-6. PEAR Field Descriptions

Field	Description
7 NOACCE	<b>CPU No Access Output Enable</b> Normal: write once Emulation: write never Special: write anytime 1 The associated pin (port E, bit 7) is general-purpose I/O. 0 The associated pin (port E, bit 7) is output and indicates whether the cycle is a CPU free cycle. This bit has no effect in single-chip or special peripheral modes.
5 PIPOE	<b>Pipe Status Signal Output Enable</b> Normal: write once Emulation: write never Special: write anytime. 0 The associated pins (port E, bits 6:5) are general-purpose I/O. 1 The associated pins (port E, bits 6:5) are outputs and indicate the state of the instruction queue This bit has no effect in single-chip or special peripheral modes.
4 NECLK	<b>No External E Clock</b> Normal and special: write anytime Emulation: write never 0 The associated pin (port E, bit 4) is the external E clock pin. External E clock is free-running if ESTR = 0 1 The associated pin (port E, bit 4) is a general-purpose I/O pin. External E clock is available as an output in all modes.
3 LSTRE	<b>Low Strobe (LSTRB) Enable</b> Normal: write once Emulation: write never Special: write anytime. 0 The associated pin (port E, bit 3) is a general-purpose I/O pin. 1 The associated pin (port E, bit 3) is configured as the $\overline{\text{LSTRB}}$ bus control output. If BDM tagging is enabled, $\overline{\text{TAGLO}}$ is multiplexed in on the rising edge of ECLK and $\overline{\text{LSTRB}}$ is driven out on the falling edge of ECLK. This bit has no effect in single-chip, peripheral, or normal expanded narrow modes. <b>Note:</b> $\overline{\text{LSTRB}}$ is used during external writes. After reset in normal expanded mode, $\overline{\text{LSTRB}}$ is disabled to provide an extra I/O pin. If $\overline{\text{LSTRB}}$ is needed, it should be enabled before any external writes. External reads do not normally need $\overline{\text{LSTRB}}$ because all 16 data bits can be driven even if the system only needs 8 bits of data.
2 RDWE	<b>Read/Write Enable</b> Normal: write once Emulation: write never Special: write anytime 0 The associated pin (port E, bit 2) is a general-purpose I/O pin. 1 The associated pin (port E, bit 2) is configured as the R/W pin This bit has no effect in single-chip or special peripheral modes. <b>Note:</b> R/W is used for external writes. After reset in normal expanded mode, R/W is disabled to provide an extra I/O pin. If R/W is needed it should be enabled before any external writes.

#### 4.4.3.1.2 Normal Expanded Wide Mode

In expanded wide modes, Ports A and B are configured as a 16-bit multiplexed address and data bus and Port E bit 4 is configured as the E clock output signal. These signals allow external memory and peripheral devices to be interfaced to the MCU.

Port E pins other than PE4/ECLK are configured as general purpose I/O pins (initially high-impedance inputs with internal pull resistors enabled). Control bits PIPOE, NECLK, LSTRE, and RDWE in the PEAR register can be used to configure Port E pins to act as bus control outputs instead of general purpose I/O pins.

It is possible to enable the pipe status signals on Port E bits 6 and 5 by setting the PIPOE bit in PEAR, but it would be unusual to do so in this mode. Development systems where pipe status signals are monitored would typically use the special variation of this mode.

The Port E bit 2 pin can be reconfigured as the  $R/\overline{W}$  bus control signal by writing “1” to the RDWE bit in PEAR. If the expanded system includes external devices that can be written, such as RAM, the RDWE bit would need to be set before any attempt to write to an external location. If there are no writable resources in the external system, PE2 can be left as a general purpose I/O pin.

The Port E bit 3 pin can be reconfigured as the  $\overline{LSTRB}$  bus control signal by writing “1” to the LSTRE bit in PEAR. The default condition of this pin is a general purpose input because the  $\overline{LSTRB}$  function is not needed in all expanded wide applications.

The Port E bit 4 pin is initially configured as ECLK output with stretch. The E clock output function depends upon the settings of the NECLK bit in the PEAR register, the IVIS bit in the MODE register and the ESTR bit in the EBICTL register. The E clock is available for use in external select decode logic or as a constant speed clock for use in the external application system.

#### 4.4.3.1.3 Normal Expanded Narrow Mode

This mode is used for lower cost production systems that use 8-bit wide external EPROMs or RAMs. Such systems take extra bus cycles to access 16-bit locations but this may be preferred over the extra cost of additional external memory devices.

Ports A and B are configured as a 16-bit address bus and Port A is multiplexed with data. Internal visibility is not available in this mode because the internal cycles would need to be split into two 8-bit cycles.

Since the PEAR register can only be written one time in this mode, use care to set all bits to the desired states during the single allowed write.

The PE3/ $\overline{LSTRB}$  pin is always a general purpose I/O pin in normal expanded narrow mode. Although it is possible to write the LSTRE bit in PEAR to “1” in this mode, the state of LSTRE is overridden and Port E bit 3 cannot be reconfigured as the  $\overline{LSTRB}$  output.

It is possible to enable the pipe status signals on Port E bits 6 and 5 by setting the PIPOE bit in PEAR, but it would be unusual to do so in this mode. LSTRB would also be needed to fully understand system activity. Development systems where pipe status signals are monitored would typically use special expanded wide mode or occasionally special expanded narrow mode.

The commands are described as follows:

- **ACK\_ENABLE** — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK\_ENABLE command itself also has the ACK pulse as a response.
- **ACK\_DISABLE** — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See [Section 6.4.3, “BDM Hardware Commands,”](#) and [Section 6.4.4, “Standard BDM Firmware Commands,”](#) for more information on the BDM commands.

The ACK\_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK\_ENABLE command is ignored by the target because it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO\_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TAGGO command will not issue an ACK pulse because this would interfere with the tagging function shared on the same pin.

## Chapter 7

# Debug Module (DBGV1) Block Description

### 7.1 Introduction

This section describes the functionality of the debug (DBG) sub-block of the HCS12 core platform.

The DBG module is designed to be fully compatible with the existing BKP\_HCS12\_A module (BKP mode) and furthermore provides an on-chip trace buffer with flexible triggering capability (DBG mode). The DBG module provides for non-intrusive debug of application software. The DBG module is optimized for the HCS12 16-bit architecture.

#### 7.1.1 Features

The DBG module in BKP mode includes these distinctive features:

- Full or dual breakpoint mode
  - Compare on address and data (full)
  - Compare on either of two addresses (dual)
- BDM or SWI breakpoint
  - Enter BDM on breakpoint (BDM)
  - Execute SWI on breakpoint (SWI)
- Tagged or forced breakpoint
  - Break just before a specific instruction will begin execution (TAG)
  - Break on the first instruction boundary after a match occurs (Force)
- Single, range, or page address compares
  - Compare on address (single)
  - Compare on address 256 byte (range)
  - Compare on any 16K page (page)
- At forced breakpoints compare address on read or write
- High and/or low byte data compares
- Comparator C can provide an additional tag or force breakpoint (enhancement for BKP mode)

Table 7-15. DBG3 Field Descriptions

Field	Description
7:6 BKAMB[H:L]	<p><b>Breakpoint Mask High Byte for First Address</b> — In dual or full mode, these bits may be used to mask (disable) the comparison of the high and/or low bytes of the first address breakpoint. The functionality is as given in <a href="#">Table 7-16</a>.</p> <p>The x:0 case is for a full address compare. When a program page is selected, the full address compare will be based on bits for a 20-bit compare. The registers used for the compare are {DBGCAH[5:0], DBGCAH[5:0], DBGCAH[7:0]}, where DBGCAH[5:0] corresponds to PPAGE[5:0] or extended address bits [19:14] and CPU address [13:0]. When a program page is not selected, the full address compare will be based on bits for a 16-bit compare. The registers used for the compare are {DBGCAH[7:0], DBGCAH[7:0]} which corresponds to CPU address [15:0].</p> <p><b>Note:</b> This extended address compare scheme causes an aliasing problem in BKP mode in which several physical addresses may match with a single logical address. This problem may be avoided by using DBG mode to generate breakpoints.</p> <p>The 1:0 case is not sensible because it would ignore the high order address and compare the low order and expansion addresses. Logic forces this case to compare all address lines (effectively ignoring the BKAMBH control bit).</p> <p>The 1:1 case is useful for triggering a breakpoint on any access to a particular expansion page. This only makes sense if a program page is being accessed so that the breakpoint trigger will occur only if DBGCAH compares.</p>
5:4 BKMB[H:L]	<p><b>Breakpoint Mask High Byte and Low Byte of Data (Second Address)</b> — In dual mode, these bits may be used to mask (disable) the comparison of the high and/or low bytes of the second address breakpoint. The functionality is as given in <a href="#">Table 7-17</a>.</p> <p>The x:0 case is for a full address compare. When a program page is selected, the full address compare will be based on bits for a 20-bit compare. The registers used for the compare are {DBGCBH[5:0], DBGCBH[5:0], DBGCBH[7:0]} where DBGCBH[5:0] corresponds to PPAGE[5:0] or extended address bits [19:14] and CPU address [13:0]. When a program page is not selected, the full address compare will be based on bits for a 16-bit compare. The registers used for the compare are {DBGCBH[7:0], DBGCBH[7:0]} which corresponds to CPU address [15:0].</p> <p><b>Note:</b> This extended address compare scheme causes an aliasing problem in BKP mode in which several physical addresses may match with a single logical address. This problem may be avoided by using DBG mode to generate breakpoints.</p> <p>The 1:0 case is not sensible because it would ignore the high order address and compare the low order and expansion addresses. Logic forces this case to compare all address lines (effectively ignoring the BKMBH control bit).</p> <p>The 1:1 case is useful for triggering a breakpoint on any access to a particular expansion page. This only makes sense if a program page is being accessed so that the breakpoint trigger will occur only if DBGCBH compares.</p> <p>In full mode, these bits may be used to mask (disable) the comparison of the high and/or low bytes of the data breakpoint. The functionality is as given in <a href="#">Table 7-18</a>.</p>
3 RWAEN	<p><b>Read/Write Comparator A Enable Bit</b> — The RWAEN bit controls whether read or write comparison is enabled for comparator A. See <a href="#">Section 7.4.2.1.1, “Read or Write Comparison,”</a> for more information. This bit is not useful for tagged operations.</p> <p>0 Read/Write is not used in comparison 1 Read/Write is used in comparison</p>
2 RWA	<p><b>Read/Write Comparator A Value Bit</b> — The RWA bit controls whether read or write is used in compare for comparator A. The RWA bit is not used if RWAEN = 0.</p> <p>0 Write cycle will be matched 1 Read cycle will be matched</p>



## Chapter 9

# Clocks and Reset Generator (CRGV4) Block Description

## 9.1 Introduction

This specification describes the function of the clocks and reset generator (CRGV4).

### 9.1.1 Features

The main features of this block are:

- Phase-locked loop (PLL) frequency multiplier
  - Reference divider
  - Automatic bandwidth control mode for low-jitter operation
  - Automatic frequency lock detector
  - CPU interrupt on entry or exit from locked condition
  - Self-clock mode in absence of reference clock
- System clock generator
  - Clock quality check
  - Clock switch for either oscillator- or PLL-based system clocks
  - User selectable disabling of clocks during wait mode for reduced power consumption
- Computer operating properly (COP) watchdog timer with time-out clear window
- System reset generation from the following possible sources:
  - Power-on reset
  - Low voltage reset
    - Refer to the device overview section for availability of this feature.
  - COP reset
  - Loss of clock reset
  - External pin reset
- Real-time interrupt (RTI)

The PLL is a frequency generator that operates in either acquisition mode or tracking mode, depending on the difference between the output frequency and the target frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

The VCO has a minimum operating frequency, which corresponds to the self-clock mode frequency  $f_{SCM}$ .

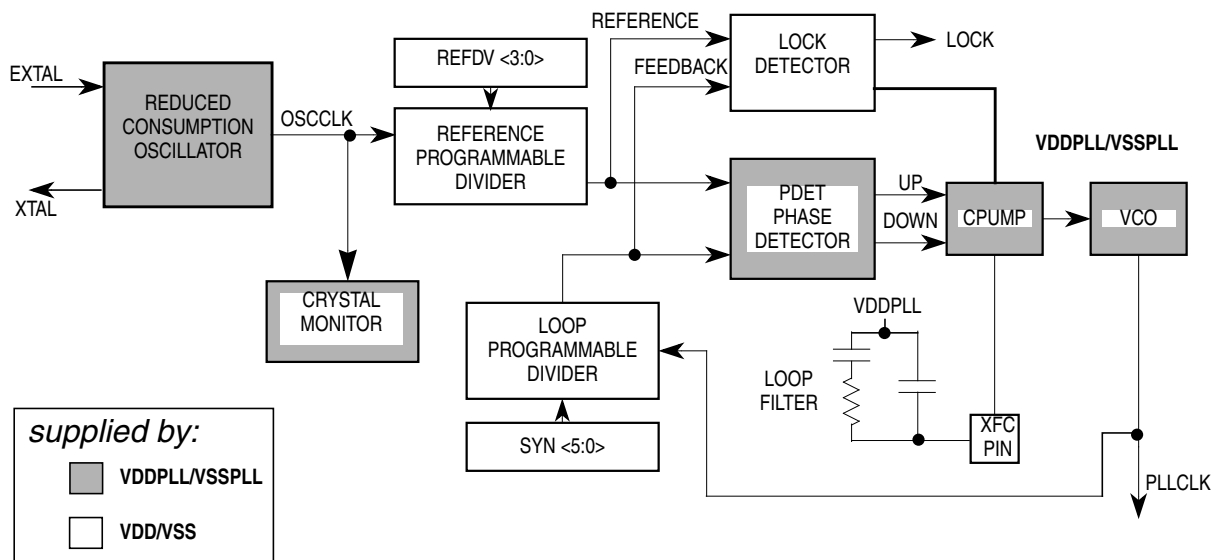


Figure 9-16. PLL Functional Diagram

### 9.4.1.1 PLL Operation

The oscillator output clock signal (OSCCLK) is fed through the reference programmable divider and is divided in a range of 1 to 16 ( $REFDV+1$ ) to output the reference clock. The VCO output clock, (PLLCLK) is fed back through the programmable loop divider and is divided in a range of 2 to 128 in increments of  $[2 \times (SYNR + 1)]$  to output the feedback clock. See Figure 9-16.

The phase detector then compares the feedback clock, with the reference clock. Correction pulses are generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external filter capacitor connected to XFC pin, based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, as described in the next subsection. The values of the external filter network and the reference frequency determine the speed of the corrections and the stability of the PLL.

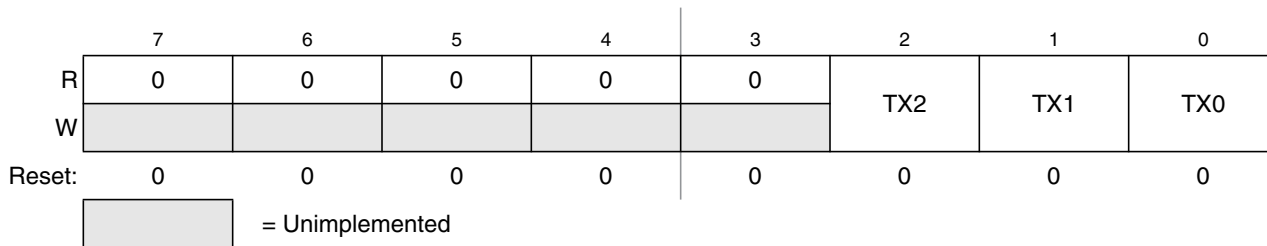
### 9.4.1.2 Acquisition and Tracking Modes

The lock detector compares the frequencies of the feedback clock, and the reference clock. Therefore, the speed of the lock detector is directly proportional to the final reference frequency. The circuit determines the mode of the PLL and the lock condition based on this comparison.

### 10.3.2.11 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CANTXFG register space.

Module Base + 0x000A



**Figure 10-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)**

#### NOTE

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK=1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Find the lowest ordered bit set to 1, all other bits will be read as 0

Write: Anytime when not in initialization mode

**Table 10-15. CANTBSEL Register Field Descriptions**

Field	Description
2:0 TX[2:0]	<b>Transmit Buffer Select</b> — The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see <a href="#">Section 10.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”</a> ). 0 The associated message buffer is deselected 1 The associated message buffer is selected, if lowest numbered bit

The following gives a short programming example of the usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000\_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000\_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software the selection of the next available Tx buffer.

- LDD CANTFLG; value read is 0b0000\_0110
- STD CANTBSEL; value written is 0b0000\_0110
- LDD CANTBSEL; value read is 0b0000\_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

The IDHITx indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

### 10.3.2.13 MSCAN Reserved Registers

These registers are reserved for factory testing of the MSCAN module and is not available in normal system operation modes.

Module Base + 0x000C, 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
	<div style="display: flex; align-items: center;"> <div style="width: 20px; height: 15px; background-color: #cccccc; border: 1px solid black; margin-right: 5px;"></div>             = Unimplemented           </div>							

**Figure 10-16. MSCAN Reserved Registers**

Read: Always read 0x0000 in normal system operation modes

Write: Unimplemented in normal system operation modes

#### NOTE

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

### 10.3.2.14 MSCAN Receive Error Counter (CANRXERR)

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

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
W								
Reset:	0	0	0	0	0	0	0	0
	<div style="display: flex; align-items: center;"> <div style="width: 20px; height: 15px; background-color: #cccccc; border: 1px solid black; margin-right: 5px;"></div>             = Unimplemented           </div>							

**Figure 10-17. MSCAN Receive Error Counter (CANRXERR)**

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

Write: Unimplemented

Module Base + 0x001C (CANIDMR4)  
 0x001D (CANIDMR5)  
 0x001E (CANIDMR6)  
 0x001F (CANIDMR7)

	7	6	5	4	3	2	1	0
R								
W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

**Figure 10-22. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 10-22. CANIDMR4–CANIDMR7 Register Field Descriptions**

Field	Description
7:0 AM[7:0]	<p><b>Acceptance Mask Bits</b> — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.</p> <p>0 Match corresponding acceptance code register and identifier bits</p> <p>1 Ignore corresponding acceptance code register bit</p>

### 10.4.5.5 MSCAN Initialization Mode

In initialization mode, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives the TXCAN pin into a recessive state.

#### NOTE

The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CANCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAACK, and CANTBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CANBTR0, CANBTR1 bit timing registers; CANIDAC; and the CANIDAR, CANIDMR message filters. See [Section 10.3.2.1, “MSCAN Control Register 0 \(CANCTL0\),”](#) for a detailed description of the initialization mode.

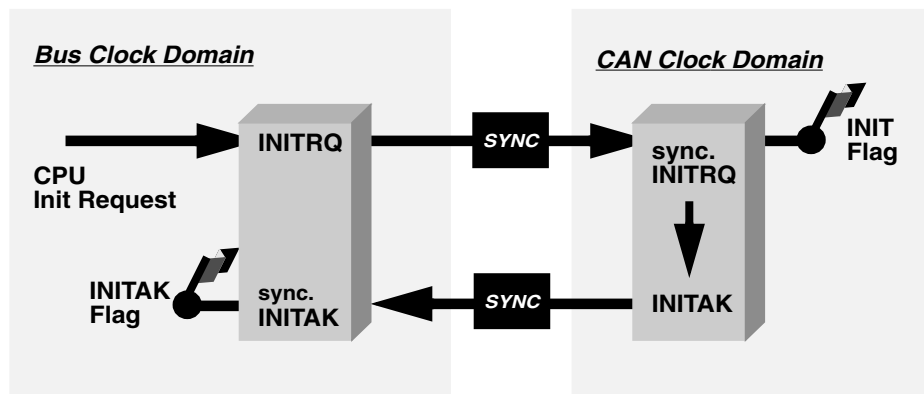


Figure 10-46. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see [Section Figure 10-46., “Initialization Request/Acknowledge Cycle”](#)).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

#### NOTE

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

Table 15-20. Pin Action

PAMOD	PEDGE	Pin Action
0	0	Falling edge
0	1	Rising edge
1	0	Div. by 64 clock enabled with pin high level
1	1	Div. by 64 clock enabled with pin low level

### NOTE

If the timer is not active ( $TEN = 0$  in TSCR), there is no divide-by-64 because the  $\div 64$  clock is generated by the timer prescaler.

Table 15-21. Timer Clock Selection

CLK1	CLK0	Timer Clock
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65536 as timer counter clock frequency

For the description of PACLK please refer [Figure 15-24](#).

If the pulse accumulator is disabled ( $PAEN = 0$ ), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

### 15.3.2.16 Pulse Accumulator Flag Register (PAFLG)

Module Base + 0x0021

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PAOVF	PAIF
W								
Reset	0	0	0	0	0	0	0	0
	Unimplemented or Reserved							

Figure 15-25. Pulse Accumulator Flag Register (PAFLG)

Read: Anytime

Write: Anytime

When the TFFCA bit in the TSCR register is set, any access to the PACNT register will clear all the flags in the PAFLG register. Timer module must stay enabled ( $TEN = 1$ ) while clearing these bits.

### 17.1.3 Modes of Operation

See [Section 17.4.2, “Operating Modes”](#) for a description of the Flash module operating modes. For program and erase operations, refer to [Section 17.4.1, “Flash Command Operations”](#).

### 17.1.4 Block Diagram

Figure 17-1 shows a block diagram of the FTS16K module.

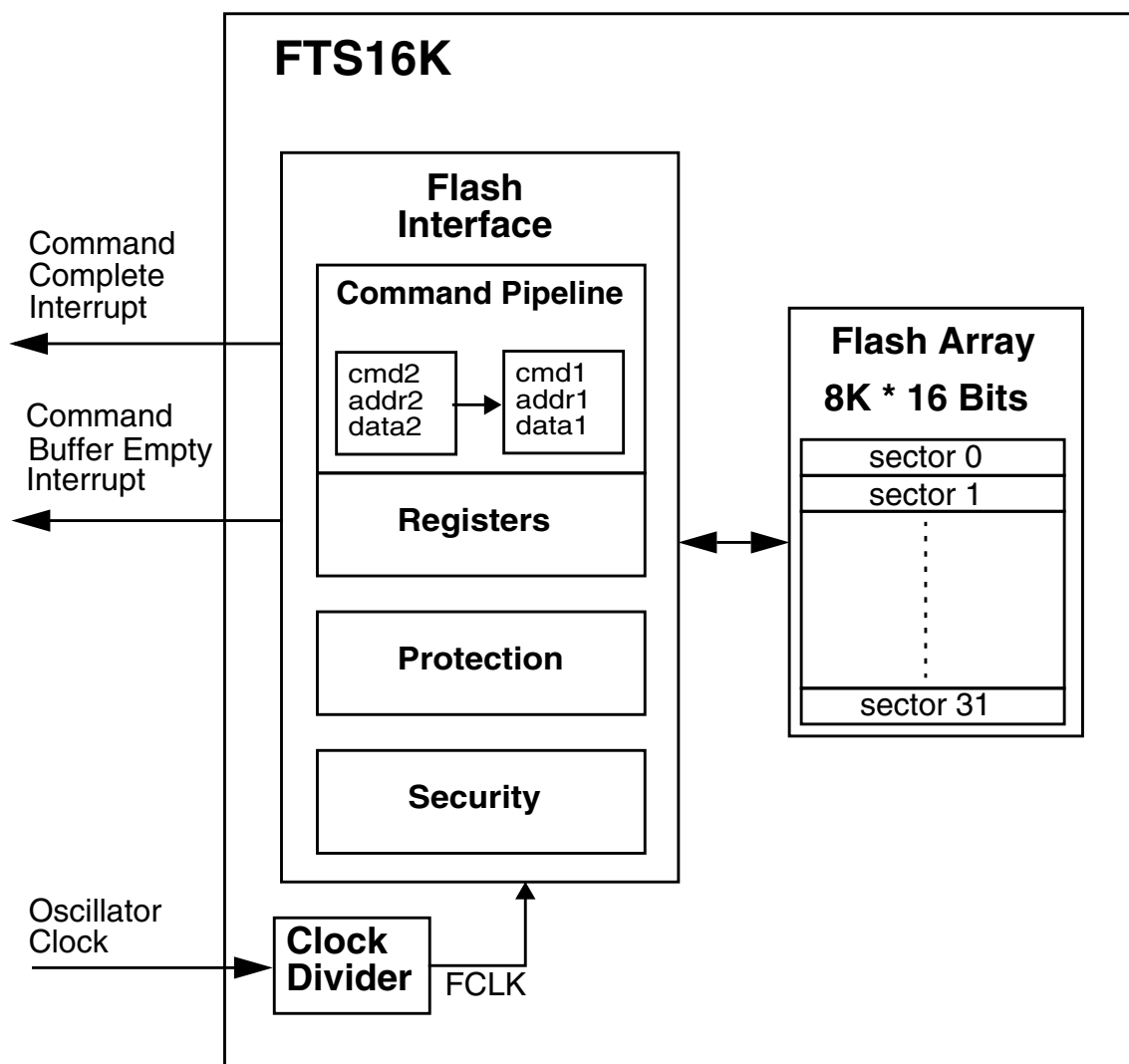


Figure 17-1. FTS16K Block Diagram

## 17.2 External Signal Description

The FTS16K module contains no signals that connect off-chip.



**Table 18-2. Flash Array Memory Map Summary**

MCU Address Range	PPAGE	Protectable Low Range	Protectable High Range	Array Relative Address <sup>(1)</sup>
0x4000–0x7FFF	Unpaged (0x3E)	0x4000–0x43FF 0x4000–0x47FF 0x4000–0x4FFF 0x4000–0x5FFF	N.A.	0x18000–0x1BFFF
0x8000–0xBFFF	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

<sup>1</sup>. Inside Flash block.

Table 18-9. Flash Protection Function

FPOPEN	FPHDIS	FPHS[1]	FPHS[0]	FPLDIS	FPLS[1]	FPLS[0]	Function <sup>(1)</sup>
1	1	x	x	1	x	x	No protection
1	1	x	x	0	x	x	Protect low range
1	0	x	x	1	x	x	Protect high range
1	0	x	x	0	x	x	Protect high and low ranges
0	1	x	x	1	x	x	Full Flash array protected
0	0	x	x	1	x	x	Unprotected high range
0	1	x	x	0	x	x	Unprotected low range
0	0	x	x	0	x	x	Unprotected high and low ranges

1. For range sizes refer to [Table 18-10](#) and [Table 18-11](#).

Table 18-10. 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

Table 18-11. Flash Protection Lower Address Range

FPLS[1:0]	Address Range	Range Size
00	0x4000–0x41FF	512 bytes
01	0x4000–0x43FF	1 Kbyte
10	0x4000–0x47FF	2 Kbytes
11	0x4000–0x4FFF	4 Kbytes

Figure 18-9 illustrates all possible protection scenarios. Although the protection scheme is loaded from the Flash array after reset, it is allowed to change in normal modes. This protection scheme can be used by applications requiring re-programming in single chip mode while providing as much protection as possible if no re-programming is required.

### 18.4.1.3.1 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 18-22](#). 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.

### 19.4.1.3.2 Program Command

The program operation will program a previously erased word in the Flash array using an embedded algorithm.

An example flow to execute the program operation is shown in [Figure 19-26](#). The program command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the program command. The data written will be programmed to the Flash array address written.
2. Write the program command, 0x20, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the program command.

If a word to be programmed is in a protected area of the Flash array, the PVIOL flag in the FSTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the FSTAT register will set after the program operation has completed unless a new command write sequence has been buffered. By executing a new program command write sequence on sequential words after the CBEIF flag in the FSTAT register has been set, up to 55% faster programming time per word can be effectively achieved than by waiting for the CCIF flag to set after each program operation.

Table 20-5. FSEC Field Descriptions

Field	Description
7–6 KEYEN[1:0]	<b>Backdoor Key Security Enable Bits</b> — The KEYEN[1:0] bits define the enabling of the backdoor key access to the Flash module as shown in Table 20-6.
5–2 NV[5:2]	<b>Nonvolatile Flag Bits</b> — The NV[5:2] bits are available to the user as nonvolatile flags.
1–0 SEC[1:0]	<b>Flash Security Bits</b> — The SEC[1:0] bits define the security state of the MCU as shown in Table 20-7. If the Flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to 1:0.

Table 20-6. Flash KEYEN States

KEYEN[1:0]	Status of Backdoor Key Access
00	DISABLED
01 <sup>(1)</sup>	DISABLED
10	ENABLED
11	DISABLED

1. Preferred KEYEN state to disable Backdoor Key Access.

Table 20-7. Flash Security States

SEC[1:0]	Status of Security
00	Secured
01 <sup>(1)</sup>	Secured
10	Unsecured
11	Secured

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 20.4.3, “Flash Module Security”.

### 20.3.2.3 RESERVED1

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

Module Base + 0x0002

	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 20-8. RESERVED1

All bits read 0 and are not writable.