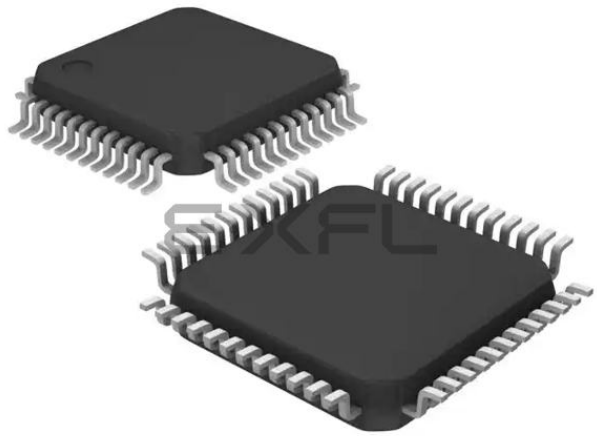

**Welcome to E-XFL.COM**

## 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 | Active |
| Core Processor | 12V1 |
| Core Size | 16-Bit |
| Speed | 25MHz |
| Connectivity | CANbus, IrDA, LINbus, SCI, SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 40 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 4K x 8 |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.13V ~ 5.5V |
| Data Converters | A/D 12x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-LQFP |
| Supplier Device Package | 48-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s12g128f0vlf |

**Table 1-17. 64-Pin LQFP Pinout for S12G48 and S12G64**

| | Function <----lowest-----PRIORITY-----highest----> | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|
| **Package Pin** | **Pin** | **2nd Func.** | **3rd Func.** | **4th Func** | **5th Func** | | **CTRL** | **Reset State** |
| 57 | PS5 | MOSI0 | — | — | — | $V_{DDX}$ | PERS/PPSS | Up |
| 58 | PS6 | SCK0 | — | — | — | $V_{DDX}$ | PERS/PPSS | Up |
| 59 | PS7 | API_EXTCLK | ECLK | $\overline{SS0}$ | — | $V_{DDX}$ | PERS/PPSS | Up |
| 60 | PM0 | RXCAN | — | — | — | $V_{DDX}$ | PERM/PPSM | Disabled |
| 61 | PM1 | TXCAN | — | — | — | $V_{DDX}$ | PERM/PPSM | Disabled |
| 62 | PM2 | — | — | — | — | $V_{DDX}$ | PERM/PPSM | Disabled |
| 63 | PM3 | — | — | — | — | $V_{DDX}$ | PERM/PPSM | Disabled |
| 64 | PJ7 | KWJ7 | — | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |

[1] The regular I/O characteristics (see Section A.2, “I/O Characteristics”) apply if the EXTAL/XTAL function is disabled

**Table 1-19. 64-Pin LQFP Pinout for S12GA48 and S12GA64**

| | Function <----lowest-----PRIORITY-----highest----> | | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|---|
| Package Pin | Pin | 2nd Func. | 3rd Func. | 4th Func | 5th Func | | CTRL | Reset State |
| 1 | PJ6 | KWJ6 | — | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 2 | PJ5 | KWJ5 | — | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 3 | PJ4 | KWJ4 | — | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 4 | RESET | — | — | — | — | $V_{DDX}$ | PULLUP | |
| 5 | VDDX | — | — | — | — | — | — | — |
| 6 | VDDR | — | — | — | — | — | — | — |
| 7 | VSSX | — | — | — | — | — | — | — |
| 8 | PE0[1] | EXTAL | — | — | — | $V_{DDX}$ | PUCR/PDPEE | Down |
| 9 | VSS | — | — | — | — | — | — | — |
| 10 | PE1[1] | XTAL | — | — | — | $V_{DDX}$ | PUCR/PDPEE | Down |
| 11 | TEST | — | — | — | — | N.A. | $\overline{\text{RESET}}$ pin | Down |
| 12 | PJ0 | KWJ0 | MISO1 | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 13 | PJ1 | KWJ1 | MOSI1 | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 14 | PJ2 | KWJ2 | SCK1 | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 15 | PJ3 | KWJ3 | $\overline{\text{SS1}}$ | — | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 16 | BKGD | MODC | — | — | — | $V_{DDX}$ | PUCR/BKPUE | Up |
| 17 | PP0 | KWP0 | ETRIG0 | API_EXTCLK | PWM0 | $V_{DDX}$ | PERP/PPSP | Disabled |
| 18 | PP1 | KWP1 | ETRIG1 | ECLKX2 | PWM1 | $V_{DDX}$ | PERP/PPSP | Disabled |
| 19 | PP2 | KWP2 | ETRIG2 | PWM2 | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 20 | PP3 | KWP3 | ETRIG3 | PWM3 | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 21 | PP4 | KWP4 | PWM4 | — | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 22 | PP5 | KWP5 | PWM5 | — | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 23 | PP6 | KWP6 | — | — | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 24 | PP7 | KWP7 | — | — | — | $V_{DDX}$ | PERP/PPSP | Disabled |
| 25 | PT7 | — | — | — | — | $V_{DDX}$ | PERT/PPST | Disabled |
| 26 | PT6 | — | — | — | — | $V_{DDX}$ | PERT/PPST | Disabled |
| 27 | PT5 | IOC5 | — | — | — | $V_{DDX}$ | PERT/PPST | Disabled |

### 1.8.9.4 Known Good Die Option (KGD)

**Table 1-32. KGD Option for S12GA192 and S12GA240**

| | Function <----lowest-----PRIORITY-----highest----> | | | | Power Supply | Internal Pull Resistor | |
|---|---|---|---|---|---|---|---|
| Wire Bond Die Pad | Pin | 2nd Func. | 3rd Func. | 4th Func. | | CTRL | Reset State |
| 1 | PJ6 | KWJ6 | SCK2 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 2 | PJ5 | KWJ5 | MOSI2 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 3 | PJ4 | KWJ4 | MISO2 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 4 | PA0 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 5 | PA1 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 6 | PA2 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 7 | PA3 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 8 | RESET | — | — | — | $V_{DDX}$ | PULLUP | |
| 9 | VDDX1 | — | — | — | — | — | — |
| 10 | VDDR | — | — | — | — | — | — |
| 11 | VSSX1 | — | — | — | — | — | — |
| 12 | PE0[1] | EXTAL | — | — | $V_{DDX}$ | PUCR/PDPEE | Down |
| 13 | VSS | — | — | — | — | — | — |
| 14 | PE1[1] | XTAL | — | — | $V_{DDX}$ | PUCR/PDPEE | Down |
| 15 | TEST | — | — | — | N.A. | $\overline{\text{RESET}}$ pin | Down |
| 16 | PA4 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 17 | PA5 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 18 | PA6 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 19 | PA7 | — | — | — | $V_{DDX}$ | PUCR/PUPAE | Disabled |
| 20 | PJ0 | KWJ0 | MISO1 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 21 | PJ1 | KWJ1 | MOSI1 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 22 | PJ2 | KWJ2 | SCK1 | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 23 | PJ3 | KWJ3 | $\overline{\text{SS1}}$ | — | $V_{DDX}$ | PERJ/PPSJ | Up |
| 24 | BKGD | MODC | — | — | $V_{DDX}$ | PUCR/BKPUE | Up |
| 25 | PB0 | ECLK | — | — | $V_{DDX}$ | PUCR/PUPBE | Disabled |
| 26 | PB1 | API_EXTCLK | — | — | $V_{DDX}$ | PUCR/PUPBE | Disabled |

**Table 11-15. Analog Input Channel Select Coding**

| SC | CD | CC | CB | CA | Analog Input Channel |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 |
| | 0 | 0 | 0 | 1 | AN1 |
| | 0 | 0 | 1 | 0 | AN2 |
| | 0 | 0 | 1 | 1 | AN3 |
| | 0 | 1 | 0 | 0 | AN4 |
| | 0 | 1 | 0 | 1 | AN5 |
| | 0 | 1 | 1 | 0 | AN6 |
| | 0 | 1 | 1 | 1 | AN7 |
| | 1 | 0 | 0 | 0 | AN7 |
| | 1 | 0 | 0 | 1 | AN7 |
| | 1 | 0 | 1 | 0 | AN7 |
| | 1 | 0 | 1 | 1 | AN7 |
| | 1 | 1 | 0 | 0 | AN7 |
| | 1 | 1 | 0 | 1 | AN7 |
| | 1 | 1 | 1 | 0 | AN7 |
| | 1 | 1 | 1 | 1 | AN7 |
| 1 | 0 | 0 | 0 | 0 | Internal_6, |
| | 0 | 0 | 0 | 1 | Internal_7 |
| | 0 | 0 | 1 | 0 | Internal_0 |
| | 0 | 0 | 1 | 1 | Internal_1 |
| | 0 | 1 | 0 | 0 | VRH |
| | 0 | 1 | 0 | 1 | VRL |
| | 0 | 1 | 1 | 0 | (VRH+VRL) / 2 |
| | 0 | 1 | 1 | 1 | Reserved |
| | 1 | 0 | 0 | 0 | Internal_2 |
| | 1 | 0 | 0 | 1 | Internal_3 |
| | 1 | 0 | 1 | 0 | Internal_4 |
| | 1 | 0 | 1 | 1 | Internal_5 |
| | 1 | 1 | X | X | Reserved |

### 13.3.2.6 ATD Control Register 5 (ATDCTL5)

Writes to this register will abort current conversion sequence and start a new conversion sequence. If the external trigger function is enabled (ETRIGE=1) an initial write to ATDCTL5 is required to allow starting of a conversion sequence which will then occur on each trigger event. Start of conversion means the beginning of the sampling phase.

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SC | SCAN | MULT | CD | CC | CB | CA |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-8. ATD Control Register 5 (ATDCTL5)**

Read: Anytime

Write: Anytime

**Table 13-14. ATDCTL5 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>SC | **Special Channel Conversion Bit** — If this bit is set, then special channel conversion can be selected using CD, CC, CB and CA of ATDCTL5. Table 13-15 lists the coding.<br>0 Special channel conversions disabled<br>1 Special channel conversions enabled |
| 5<br>SCAN | **Continuous Conversion Sequence Mode** — This bit selects whether conversion sequences are performed continuously or only once. If the external trigger function is enabled (ETRIGE=1) setting this bit has no effect, thus the external trigger always starts a single conversion sequence.<br>0 Single conversion sequence<br>1 Continuous conversion sequences (scan mode) |
| 4<br>MULT | **Multi-Channel Sample Mode** — When MULT is 0, the ATD sequence controller samples only from the specified analog input channel for an entire conversion sequence. The analog channel is selected by channel selection code (control bits CD/CC/CB/CA located in ATDCTL5). When MULT is 1, the ATD sequence controller samples across channels. The number of channels sampled is determined by the sequence length value (S8C, S4C, S2C, S1C). The first analog channel examined is determined by channel selection code (CD, CC, CB, CA control bits); subsequent channels sampled in the sequence are determined by incrementing the channel selection code or wrapping around to AN0 (channel 0).<br>0 Sample only one channel<br>1 Sample across several channels |
| 3–0<br>CD, CC,<br>CB, CA | **Analog Input Channel Select Code** — These bits select the analog input channel(s). Table 13-15 lists the coding used to select the various analog input channels.<br>In the case of single channel conversions (MULT=0), this selection code specifies the channel to be examined.<br>In the case of multiple channel conversions (MULT=1), this selection code specifies the first channel to be examined in the conversion sequence. Subsequent channels are determined by incrementing the channel selection code or wrapping around to AN0 (after converting the channel defined by the Wrap Around Channel Select Bits WRAP3-0 in ATDCTL0). When starting with a channel number higher than the one defined by WRAP3-0 the first wrap around will be AN11 to AN0. |

### 14.3.2.12 ATD Conversion Result Registers (ATDDR*n*)

The A/D conversion results are stored in 12 result registers. Results are always in unsigned data representation. Left and right justification is selected using the DJM control bit in ATDCTL3.

If automatic compare of conversions results is enabled (CMPE[*n*]=1 in ATDCMPE), these registers must be written with the compare values in left or right justified format depending on the actual value of the DJM bit. In this case, as the ATDDR*n* register is used to hold the compare value, the result will not be stored there at the end of the conversion but is lost.

***Attention, n is the conversion number, NOT the channel number!***

Read: Anytime

Write: Anytime

**NOTE**

For conversions not using automatic compare, results are stored in the result registers after each conversion. In this case avoid writing to ATDDRn except for initial values, because an A/D result might be overwritten.

#### 14.3.2.12.1 Left Justified Result Data (DJM=0)

Module Base +
0x0010 = ATDDR0, 0x0012 = ATDDR1, 0x0014 = ATDDR2, 0x0016 = ATDDR3
0x0018 = ATDDR4, 0x001A = ATDDR5, 0x001C = ATDDR6, 0x001E = ATDDR7
0x0020 = ATDDR8, 0x0022 = ATDDR9, 0x0024 = ATDDR10, 0x0026 = ATDDR11

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Result-Bit[11:0] | | | | | | | | | | | | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-14. Left justified ATD conversion result register (ATDDR*n*)**

Table 14-21 shows how depending on the A/D resolution the conversion result is transferred to the ATD result registers for left justified data. Compare is always done using all 12 bits of both the conversion result and the compare value in ATDDRn.

**Table 14-21. Conversion result mapping to ATDDRn**

| A/D resolution | DJM | conversion result mapping to ATDDR*n* |
|---|---|---|
| 8-bit data | 0 | Result-Bit[11:4] = conversion result,<br>Result-Bit[3:0]=0000 |
| 10-bit data | 0 | Result-Bit[11:2] = conversion result,<br>Result-Bit[1:0]=00 |
| 12-bit data | 0 | Result-Bit[11:0] = result |

## 15.1.2 Modes of Operation

### 15.1.2.1 Conversion Modes

There is software programmable selection between performing **single** or **continuous conversion** on a **single channel** or **multiple channels**.

### 15.1.2.2 MCU Operating Modes

- **Stop Mode**
  Entering Stop Mode aborts any conversion sequence in progress and if a sequence was aborted restarts it after exiting stop mode. This has the same effect/consequences as starting a conversion sequence with write to ATDCTL5. So after exiting from stop mode with a previously aborted sequence all flags are cleared etc.
- **Wait Mode**
  ADC10B16C behaves same in Run and Wait Mode. For reduced power consumption continuous conversions should be aborted before entering Wait mode.
- **Freeze Mode**
  In Freeze Mode the ADC10B16C will either continue or finish or stop converting according to the FRZ1 and FRZ0 bits. This is useful for debugging and emulation.

**Table 16-11. ATD Behavior in Freeze Mode (Breakpoint)**

| FRZ1 | FRZ0 | Behavior in Freeze Mode |
|---|---|---|
| 0 | 0 | Continue conversion |
| 0 | 1 | Reserved |
| 1 | 0 | Finish current conversion, then freeze |
| 1 | 1 | Freeze Immediately |

### 16.3.2.5 ATD Control Register 4 (ATDCTL4)

Writes to this register will abort current conversion sequence.

Module Base + 0x0004


**Figure 16-7. ATD Control Register 4 (ATDCTL4)**

Read: Anytime

Write: Anytime

**Table 16-12. ATDCTL4 Field Descriptions**

| Field | Description |
|---|---|
| 7–5<br>SMP[2:0] | **Sample Time Select** — These three bits select the length of the sample time in units of ATD conversion clock cycles. Note that the ATD conversion clock period is itself a function of the prescaler value (bits PRS4-0). Table 16-13 lists the available sample time lengths. |
| 4–0<br>PRS[4:0] | **ATD Clock Prescaler** — These 5 bits are the binary prescaler value PRS. The ATD conversion clock frequency is calculated as follows:<br>$f_{ATDCLK} = \frac{f_{BUS}}{2 \times (PRS + 1)}$<br>Refer to Device Specification for allowed frequency range of $f_{ATDCLK}$. |

**Table 16-13. Sample Time Select**

| SMP2 | SMP1 | SMP0 | Sample Time in Number of ATD Clock Cycles |
|---|---|---|---|
| 0 | 0 | 0 | 4 |
| 0 | 0 | 1 | 6 |
| 0 | 1 | 0 | 8 |
| 0 | 1 | 1 | 10 |
| 1 | 0 | 0 | 12 |
| 1 | 0 | 1 | 16 |
| 1 | 1 | 0 | 20 |

### 16.3.2.9 ATD Status Register 2 (ATDSTAT2)

This read-only register contains the Conversion Complete Flags CCF[15:0].

Module Base + 0x000A

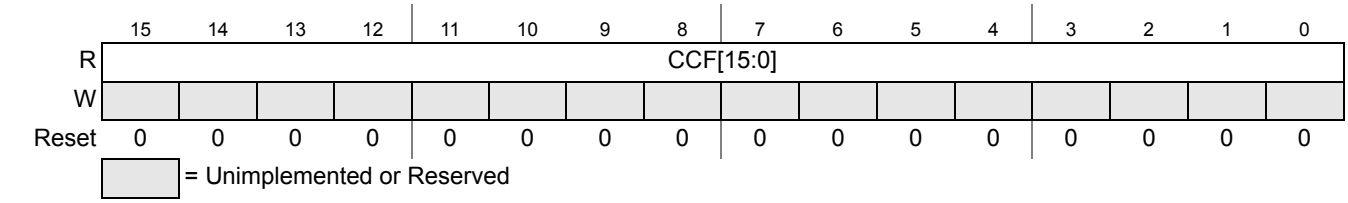


**Figure 16-11. ATD Status Register 2 (ATDSTAT2)**

Read: Anytime

Write: Anytime (for details see Table 16-18 below)

**Table 16-18. ATDSTAT2 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>CCF[15:0] | **Conversion Complete Flag *n* (*n*= 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0) (*n conversion number, NOT channel number!)*—** A conversion complete flag is set at the end of each conversion in a sequence. The flags are associated with the conversion position in a sequence (and also the result register number). Therefore in non-fifo mode, CCF[4] is set when the fifth conversion in a sequence is complete and the result is available in result register ATDDR4; CCF[5] is set when the sixth conversion in a sequence is complete and the result is available in ATDDR5, and so forth.<br>If automatic compare of conversion results is enabled (CMPE[*n*]=1 in ATDCMPE), the conversion complete flag is only set if comparison with ATDDR*n* is true. If ACMPIE=1 a compare interrupt will be requested. In this case, as the ATDDR*n* result register is used to hold the compare value, the result will not be stored there at the end of the conversion but is lost.<br>A flag CCF[*n*] is cleared when one of the following occurs:<br>A) Write to ATDCTL5 (a new conversion sequence is started)<br>B) If AFFC=0, write “1” to CCF[*n*]<br>C) If AFFC=1 and CMPE[*n*]=0, read of result register ATDDR*n*<br>D) If AFFC=1 and CMPE[*n*]=1, write to result register ATDDR*n*<br>In case of a concurrent set and clear on CCF[*n*]: The clearing by method A) will overwrite the set. The clearing by methods B) or C) or D) will be overwritten by the set.<br>0 Conversion number *n* not completed or successfully compared<br>1 If (CMPE[*n*]=0): Conversion number *n* has completed. Result is ready in ATDDR*n*.<br>If (CMPE[*n*]=1): Compare for conversion result number *n* with compare value in ATDDR*n*, using compare operator CMPGT[*n*] is true. (No result available in ATDDR*n*) |

**Table 20-11. SCISR1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>FE | **Framing Error Flag** — FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading SCI status register 1 (SCISR1) with FE set and then reading the SCI data register low (SCIDRL).<br>0 No framing error<br>1 Framing error |
| 0<br>PF | **Parity Error Flag** — PF is set when the parity enable bit (PE) is set and the parity of the received data does not match the parity type bit (PT). PF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear PF by reading SCI status register 1 (SCISR1), and then reading SCI data register low (SCIDRL).<br>0 No parity error<br>1 Parity error |

## 20.3.2.8 SCI Status Register 2 (SCISR2)

Module Base + 0x0005

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AMAP | 0 | 0 | TXPOL | RXPOL | BRK13 | TXDIR | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-11. SCI Status Register 2 (SCISR2)**

Read: Anytime

Write: Anytime

**Table 20-12. SCISR2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AMAP | **Alternative Map** — This bit controls which registers sharing the same address space are accessible. In the reset condition the SCI behaves as previous versions. Setting AMAP=1 allows the access to another set of control and status registers and hides the baud rate and SCI control Register 1.<br>0 The registers labelled SCIBDH (0x0000),SCIBDL (0x0001), SCICR1 (0x0002) are accessible<br>1 The registers labelled SCIASR1 (0x0000),SCIACR1 (0x0001), SCIACR2 (0x00002) are accessible |
| 4<br>TXPOL | **Transmit Polarity** — This bit control the polarity of the transmitted data. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.<br>0 Normal polarity<br>1 Inverted polarity |

When the transmit shift register is not transmitting a frame, the TXD pin goes to the idle condition, logic 1. If at any time software clears the TE bit in SCI control register 2 (SCICR2), the transmitter enable signal goes low and the transmit signal goes idle.

If software clears TE while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing TE.

To separate messages with preambles with minimum idle line time, use this sequence between messages:

1. Write the last byte of the first message to SCIDRH/L.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting the TE bit.
4. Write the first byte of the second message to SCIDRH/L.

### 20.4.5.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCI control register 2 (SCICR2) loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCI control register 1 (SCICR1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

The SCI recognizes a break character when there are 10 or 11(M = 0 or M = 1) consecutive zero received. Depending if the break detect feature is enabled or not receiving a break character has these effects on SCI registers.

If the break detect feature is disabled (BKDFE = 0):

- Sets the framing error flag, FE
- Sets the receive data register full flag, RDRF
- Clears the SCI data registers (SCIDRH/L)
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the receiver active flag, RAF (see 3.4.4 and 3.4.5 SCI Status Register 1 and 2)

If the break detect feature is enabled (BKDFE = 1) there are two scenarios[1]

The break is detected right from a start bit or is detected during a byte reception.

- Sets the break detect interrupt flag, BKDIF
- Does not change the data register full flag, RDRF or overrun flag OR
- Does not change the framing error flag FE, parity error flag PE.
- Does not clear the SCI data registers (SCIDRH/L)
- May set noise flag NF, or receiver active flag RAF.

1. A Break character in this context are either 10 or 11 consecutive zero received bits

Only bits related to implemented channels are valid.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 TIOS | R | RESERVED | RESERVED | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| | W | | | | | | | | |
| 0x0001 CFORC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | RESERVED | RESERVED | FOC5 | FOC4 | FOC3 | FOC2 | FOC1 | FOC0 |
| 0x0004 TCNTH | R | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| | W | | | | | | | | |
| 0x0005 TCNTL | R | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |
| | W | | | | | | | | |
| 0x0006 TSCR1 | R | TEN | TSWAI | TSFRZ | TFFCA | PRNT | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 TTOV | R | RESERVED | RESERVED | TOV5 | TOV4 | TOV3 | TOV2 | TOV1 | TOV0 |
| | W | | | | | | | | |
| 0x0008 TCTL1 | R | RESERVED | RESERVED | RESERVED | RESERVED | OM5 | OL5 | OM4 | OL4 |
| | W | | | | | | | | |
| 0x0009 TCTL2 | R | OM3 | OL3 | OM2 | OL2 | OM1 | OL1 | OM0 | OL0 |
| | W | | | | | | | | |
| 0x000A TCTL3 | R | RESERVED | RESERVED | RESERVED | RESERVED | EDG5B | EDG5A | EDG4B | EDG4A |
| | W | | | | | | | | |
| 0x000B TCTL4 | R | EDG3B | EDG3A | EDG2B | EDG2A | EDG1B | EDG1A | EDG0B | EDG0A |
| | W | | | | | | | | |
| 0x000C TIE | R | RESERVED | RESERVED | C5I | C4I | C3I | C2I | C1I | C0I |
| | W | | | | | | | | |
| 0x000D TSCR2 | R | TOI | 0 | 0 | 0 | RESERVED | PR2 | PR1 | PR0 |
| | W | | | | | | | | |
| 0x000E TFLG1 | R | RESERVED | RESERVED | C5F | C4F | C3F | C2F | C1F | C0F |
| | W | | | | | | | | |
| 0x000F TFLG2 | R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010–0x001F TCxH–TCxL[1] | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | W | | | | | | | | |
| | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | W | | | | | | | | |
| 0x0024–0x002B Reserved | R | | | | | | | | |
| | W | | | | | | | | |
| 0x002C OCPD | R | RESERVED | RESERVED | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| | W | | | | | | | | |
| 0x002D Reserved | R | | | | | | | | |
| 0x002E PTPSR | R | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| | W | | | | | | | | |
| 0x002F Reserved | R | | | | | | | | |
| | W | | | | | | | | |

**Figure 22-3. TIM16B6CV3 Register Summary**

Setting OCPDx to zero allows the internal register to drive the programmed state to OCx. This allows a glitch free switch over of port from general purpose I/O to timer output once the OCPDx bit is set to zero.

## 22.5 Resets

The reset state of each individual bit is listed within Section 22.3, “Memory Map and Register Definition” which details the registers and their bit fields

## 22.6 Interrupts

This section describes interrupts originated by the TIM16B6CV3 block. Table 22-18 lists the interrupts generated by the TIM16B6CV3 to communicate with the MCU.

**Table 22-18. TIM16B6CV3 Interrupts**

| Interrupt | Offset | Vector | Priority | Source | Description |
|---|---|---|---|---|---|
| C[5:0]F | — | — | — | Timer Channel 5–0 | Active high timer channel interrupts 5–0 |
| TOF | — | — | — | Timer Overflow | Timer Overflow interrupt |

The TIM16B6CV3 could use up to 7 interrupt vectors. The interrupt vector offsets and interrupt numbers are chip dependent.

### 22.6.1 Channel [5:0] Interrupt (C[5:0]F)

This active high outputs will be asserted by the module to request a timer channel 7 – 0 interrupt. The TIM block only generates the interrupt and does not service it. Only bits related to implemented channels are valid.

### 22.6.2 Timer Overflow Interrupt (TOF)

This active high output will be asserted by the module to request a timer overflow interrupt. The TIM block only generates the interrupt and does not service it.

## 25.4.6 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

If a Flash block is read during execution of an algorithm (CCIF = 0) on that same block, the read operation will return invalid data if both flags SFDIF and DFDIF are set. If the SFDIF or DFDIF flags were not previously set when the invalid read operation occurred, both the SFDIF and DFDIF flags will be set.

If the ACCERR or FPVIOL bits are set in the FSTAT register, the user must clear these bits before starting any command write sequence (see Section 25.3.2.7).

**CAUTION**

A Flash word or phrase must be in the erased state before being programmed. Cumulative programming of bits within a Flash word or phrase is not allowed.

### 25.4.6.1 Erase Verify All Blocks Command

The Erase Verify All Blocks command will verify that all P-Flash and EEPROM blocks have been erased.

**Table 25-31. Erase Verify All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x01 | Not required |

Upon clearing CCIF to launch the Erase Verify All Blocks command, the Memory Controller will verify that the entire Flash memory space is erased. The CCIF flag will set after the Erase Verify All Blocks operation has completed. If all blocks are not erased, it means blank check failed, both MGSTAT bits will be set.

**Table 25-32. Erase Verify All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | FPVIOL | None |
| | MGSTAT1 | Set if any errors have been encountered during the read[1] or if blank check failed . |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the read[1] or if blank check failed. |

[1] As found in the memory map for FTMRG32K1.

- Fast sector erase and phrase program operation
- Ability to read the P-Flash memory while programming a word in the EEPROM memory
- Flexible protection scheme to prevent accidental program or erase of P-Flash memory

#### 27.1.2.2 EEPROM Features

- 2 Kbytes of EEPROM memory composed of one 2 Kbyte Flash block divided into 512 sectors of 4 bytes
- Single bit fault correction and double bit fault detection within a word during read operations
- Automated program and erase algorithm with verify and generation of ECC parity bits
- Fast sector erase and word program operation
- Protection scheme to prevent accidental program or erase of EEPROM memory
- Ability to program up to four words in a burst sequence

#### 27.1.2.3 Other Flash Module Features

- No external high-voltage power supply required for Flash memory program and erase operations
- Interrupt generation on Flash command completion and Flash error detection
- Security mechanism to prevent unauthorized access to the Flash memory

### 27.1.3 Block Diagram

The block diagram of the Flash module is shown in Figure 27-1.

#### 27.4.4.2.1 Define FCCOB Contents

The FCCOB parameter fields must be loaded with all required parameters for the Flash command being executed. Access to the FCCOB parameter fields is controlled via the CCOBIX bits in the FCCOBIX register (see Section 27.3.2.3).

The contents of the FCCOB parameter fields are transferred to the Memory Controller when the user clears the CCIF command completion flag in the FSTAT register (writing 1 clears the CCIF to 0). The CCIF flag will remain clear until the Flash command has completed. Upon completion, the Memory Controller will return CCIF to 1 and the FCCOB register will be used to communicate any results. The flow for a generic command write sequence is shown in Figure 27-26.

**Table 27-49. Erase P-Flash Sector Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 27-27) |
| | | Set if an invalid global address [17:16] is supplied see Table 27-3) |
| | | Set if a misaligned phrase address is supplied (global address [2:0] != 000) |
| | FPVIOL | Set if the selected P-Flash sector is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 27.4.6.10 Unsecure Flash Command

The Unsecure Flash command will erase the entire P-Flash and EEPROM memory space and, if the erase is successful, will release security.

**Table 27-50. Unsecure Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0B | Not required |

Upon clearing CCIF to launch the Unsecure Flash command, the Memory Controller will erase the entire P-Flash and EEPROM memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. If the erase verify is not successful, the Unsecure Flash operation sets MGSTAT1 and terminates without changing the security state. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag is set after the Unsecure Flash operation has completed.

**Table 27-51. Unsecure Flash Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if command not available in current mode (see Table 27-27) |
| | FPVIOL | Set if any area of the P-Flash or EEPROM memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 27.4.6.11 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command will only execute if it is enabled by the KEYEN bits in the FSEC register (see Table 27-10). The Verify Backdoor Access Key command releases security if user-supplied keys match those stored in the Flash security bytes of the Flash configuration field (see

**Table 30-46. Erase Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [17:16] to identify Flash block |
| 001 | Global address [15:0] in Flash block to be erased | |

Upon clearing CCIF to launch the Erase Flash Block command, the Memory Controller will erase the selected Flash block and verify that it is erased. The CCIF flag will set after the Erase Flash Block operation has completed.

**Table 30-47. Erase Flash Block Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 001 at command launch |
| | | Set if command not available in current mode (see Table 30-27) |
| | | Set if an invalid global address [17:16] is supplied |
| | | Set if the supplied P-Flash address is not phrase-aligned or if the EEPROM address is not word-aligned |
| | FPVIOL | Set if an area of the selected Flash block is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |

### 30.4.6.9 Erase P-Flash Sector Command

The Erase P-Flash Sector operation will erase all addresses in a P-Flash sector.

**Table 30-48. Erase P-Flash Sector Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x0A | Global address [17:16] to identify P-Flash block to be erased |
| 001 | Global address [15:0] anywhere within the sector to be erased. Refer to Section 30.1.2.1 for the P-Flash sector size. | |

Upon clearing CCIF to launch the Erase P-Flash Sector command, the Memory Controller will erase the selected Flash sector and then verify that it is erased. The CCIF flag will be set after the Erase P-Flash Sector operation has completed.

| Address & Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000A FCCOBHI | R | CCOB15 | CCOB14 | CCOB13 | CCOB12 | CCOB11 | CCOB10 | CCOB9 | CCOB8 |
| | W | | | | | | | | |
| 0x000B FCCOBLO | R | CCOB7 | CCOB6 | CCOB5 | CCOB4 | CCOB3 | CCOB2 | CCOB1 | CCOB0 |
| | W | | | | | | | | |
| 0x000C FRSV1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000D FRSV2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000E FRSV3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x000F FRSV4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0010 FOPT | R | NV7 | NV6 | NV5 | NV4 | NV3 | NV2 | NV1 | NV0 |
| | W | | | | | | | | |
| 0x0011 FRSV5 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0012 FRSV6 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0013 FRSV7 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 31-4. FTMRG240K2 Register Summary (continued)**

### 31.3.2.1 Flash Clock Divider Register (FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

### A.4.3.1 ADC Accuracy Definitions

For the following definitions see also Figure A-1.
Differential non-linearity (DNL) is defined as the difference between two adjacent switching steps.

$$DNL(i) = \frac{V_i - V_{i-1}}{1LSB} - 1$$

The integral non-linearity (INL) is defined as the sum of all DNLs:

$$INL(n) = \sum_{i=1}^{n} DNL(i) = \frac{V_n - V_0}{1LSB} - n$$