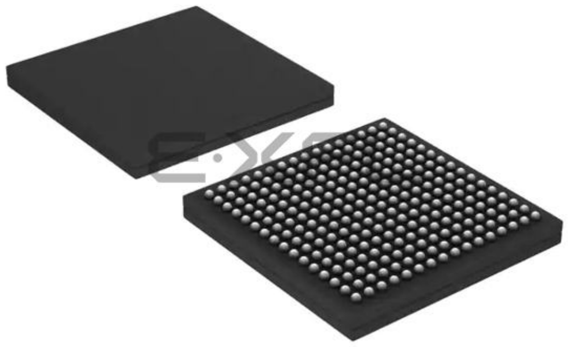
NXP USA Inc. - MCF5214CVF66J Datasheet



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	Coldfire V2
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	CANbus, EBI/EMI, I²C, SPI, UART/USART
Peripherals	DMA, LVD, POR, PWM, WDT
Number of I/O	142
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	256-LBGA
Supplier Device Package	256-MAPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mcf5214cvf66j

  - Ability to boot from internal Flash memory or external memories that are 8, 16, or 32 bits wide
- Reset
  - Separate reset in and reset out signals
  - Seven sources of reset:
    - Power-on reset (POR)
    - External
    - Software
    - Watchdog
    - Loss of clock
    - Loss of lock
    - Low-voltage detection (LVD)
  - Status flag indication of source of last reset
- Chip integration module (CIM)
  - System configuration during reset
  - Support for single chip, master, and test modes
  - Selects one of four clock modes
  - Sets boot device and its data port width
  - Configures output pad drive strength
  - Unique part identification number and part revision number
- General purpose I/O interface
  - Up to 142 bits of general purpose I/O for MCF5280/1/2
  - Up to 134 bits of general purpose I/O for MCF5214/6
  - Coherent 32-bit control
  - Bit manipulation supported via set/clear functions
  - Unused peripheral pins may be used as extra GPIO
- JTAG support for system-level board testing

1.1.8 SDRAM Controller

The SDRAM controller provides all required signals for glueless interfacing to a variety of JEDEC-compliant SDRAM devices. SRAS/SCAS address multiplexing is software configurable for different page sizes. To maintain refresh capability without conflicting with concurrent accesses on the address and data buses, $\overline{\text{SRAS}}$, $\overline{\text{SCAS}}$, $\overline{\text{DRAMW}}$, $\overline{\text{SDRAM\_CS}}$[1:0], and SCKE are dedicated SDRAM signals.

1.1.9 Test Access Port

The MCF5282 supports circuit board test strategies based on the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The test logic includes a test access port (TAP) consisting of a 16-state controller, an instruction register, and three test registers (a 1-bit bypass register, a 256-bit boundary-scan register, and a 32-bit ID register). The boundary scan register links the device’s pins into one shift register. Test logic, implemented using static logic design, is independent of the device system logic.

The MCF5282 implementation supports the following:

- Perform boundary-scan operations to test circuit board electrical continuity
- Sample MCF5282 system pins during operation and transparently shift out the result in the boundary scan register
- Bypass the MCF5282 for a given circuit board test by effectively reducing the boundary-scan register to a single bit
- Disable the output drive to pins during circuit-board testing
- Drive output pins to stable levels

1.1.10 UART Modules

The MCF5282 contains three full-duplex UARTs that function independently. The three UARTs can be clocked by the system clock, eliminating the need for an external crystal.

Each UART has the following features:

- Each can be clocked by the system clock, eliminating a need for an external UART clock
- Full-duplex asynchronous/synchronous receiver/transmitter channel
- Quadruple-buffered receiver
- Double-buffered transmitter
- Independently programmable receiver and transmitter clock sources
- Programmable data format:
  - — 5–8 data bits plus parity
  - — Odd, even, no parity, or force parity
  - — One, one-and-a-half, or two stop bits
- Each channel programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic wake-up mode for multidrop applications
- Four maskable interrupt conditions
- All three UARTs have DMA request capability
- Parity, framing, and overrun error detection
- False-start bit detection
- Line-break detection and generation

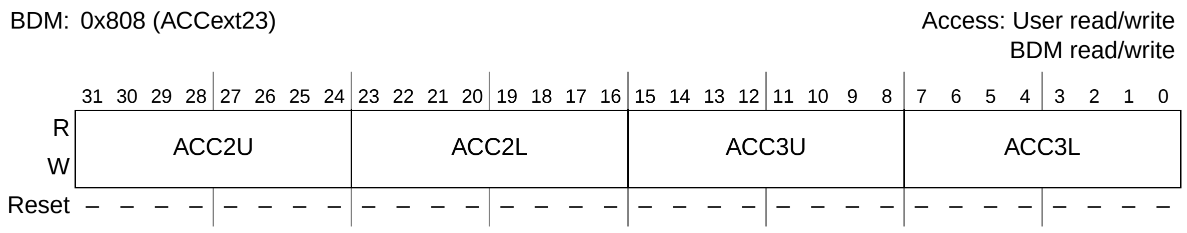


Figure 3-6. Accumulator Extension Register (ACCext23)

Table 3-7. ACCext23 Field Descriptions

Field	Description
31–24 ACC2U	Accumulator 2 upper extension byte
23–16 ACC2L	Accumulator 2 lower extension byte
15–8 ACC3U	Accumulator 3 upper extension byte
7–0 ACC3L	Accumulator 3 lower extension byte

3.3 Functional Description

The MAC speeds execution of ColdFire integer-multiply instructions (MULS and MULU) and provides additional functionality for multiply-accumulate operations. By executing MULS and MULU in the MAC, execution times are minimized and deterministic compared to the 2-bit/cycle algorithm with early termination that the OEP normally uses if no MAC hardware is present.

The added MAC instructions to the ColdFire ISA provide for the multiplication of two numbers, followed by the addition or subtraction of the product to or from the value in an accumulator. Optionally, the product may be shifted left or right by 1 bit before addition or subtraction. Hardware support for saturation arithmetic can be enabled to minimize software overhead when dealing with potential overflow conditions. Multiply-accumulate operations support 16- or 32-bit input operands in these formats:

- Signed integers
- Unsigned integers
- Signed, fixed-point, fractional numbers

The EMAC is optimized for single-cycle, pipelined 32 × 32 multiplications. For word- and longword-sized integer input operands, the low-order 40 bits of the product are formed and used with the destination accumulator. For fractional operands, the entire 64-bit product is calculated and truncated or rounded to the most-significant 40-bit result using the round-to-nearest (even) method before it is combined with the destination accumulator.

For all operations, the resulting 40-bit product is extended to a 48-bit value (using sign-extension for signed integer and fractional operands, zero-fill for unsigned integer operands) before being combined with the 48-bit destination accumulator.

```
                        then operandX[31:0] = {Rx[31:16], 0x0000}
                        else operandX[31:0] = {Rx[15:0],  0x0000}
               }
               else {operandY[31:0] = Ry[31:0]
                     operandX[31:0] = Rx[31:0]
               }
            /* perform the multiply */
            product[63:0] = (operandY[31:0] * operandX[31:0]) << 1
            /* check for product rounding */
            if (MACSR.R/T == 1)
               then { /* perform convergent rounding */
                     if (product[23:0] > 0x80_0000)
                        then product[63:24] = product[63:24] + 1
               else if ((product[23:0] == 0x80_0000) and and  (product[24] == 1))
                                then product[63:24] = product[63:24] + 1
               }
            /* sign-extend to 48 bits and combine with accumulator */
            /* check for the -1 * -1 overflow case */
    if ((operandY[31:0] == 0x8000_0000) and and  (operandX[31:0] == 0x8000_0000))
               then product[71:64] = 0x00                 /* zero-fill */
               else product[71:64] = {8{product[63]}}    /* sign-extend */
            if (inst == MSAC)
               then result[47:0] = ACCx[47:0] - product[71:24]
               else result[47:0] = ACCx[47:0] + product[71:24]
            /* check for accumulation overflow */
            if (accumulationOverflow == 1)
               then {MACSR.PAVn = 1
                     MACSR.V = 1
                     if (MACSR.OMC == 1)
                        then /* accumulation overflow,
                                saturationMode enabled */
                             if (result[47] == 1)
                                then result[47:0] = 0x007f_ffff_ff00
                                else result[47:0] = 0xff80_0000_0000
               }
            /* transfer the result to the accumulator */
            ACCx[47:0] = result[47:0]
               }
      MACSR.V = MACSR.PAVn
      MACSR.N = ACCx[47]
      if (ACCx[47:0] == 0x0000_0000_0000)
          then MACSR.Z = 1
          else MACSR.Z = 0
      if ((ACCx[47:39] == 0x00_0) || (ACCx[47:39] == 0xff_1))
          then MACSR.EV = 0
          else MACSR.EV = 1
break;
case 2:              /* unsigned integers */
   if (MACSR.OMC == 0 || MACSR.PAVn == 0)
      then {
            MACSR.PAVn = 0
            /* select the input operands */
            if (sz == word)
               then {if (U/Ly == 1)
                        then operandY[31:0] = {0x0000, Ry[31:16]}
                        else operandY[31:0] = {0x0000, Ry[15:0]}
                     if (U/Lx == 1)
```

Table 7-6. PLL/CLKOUT Stop Mode Operation

STPMD[1:0]	Operation During Stop Mode				
	System Clocks	CLKOUT	PLL	OSC	PMM
00	Disabled	Enabled	Enabled	Enabled	Enabled
01	Disabled	Disabled	Enabled	Enabled	Enabled
10	Disabled	Disabled	Disabled	Enabled	Enabled
11	Disabled	Disabled	Disabled	Disabled	Low-Power Option

NOTE

If LPCR[LPMD] is cleared, then the device stops executing code upon issue of a STOP instruction. However, no clocks are disabled.

7.3 Functional Description

The functions and characteristics of the low-power modes, and how each module is affected by, or affects, these modes are discussed in this section.

7.3.1 Low-Power Modes

The system enters a low-power mode by executing a STOP instruction. Which mode the device actually enters (either stop, wait, or doze) depends on what is programmed in LPCR[LPMD]. Entry into any of these modes idles the CPU with no cycles active, powers down the system and stops all internal clocks appropriately. During stop mode, the system clock is stopped low.

For entry into stop mode, the LPICR[ENBSTOP] bit must be set before a STOP instruction is issued.

A wakeup event is required to exit a low-power mode and return to run mode. Wakeup events consist of any of these conditions:

- Any type of reset
- Any valid, enabled interrupt request

The latter method of exiting from low-power mode, by a valid and enabled interrupt request, requires several things:

- An interrupt request whose priority is higher than the value programmed in the XLPM_IPL field of the LPICR
- An interrupt request whose priority higher than the value programmed in the interrupt priority mask (I) field of the core's status register
- An interrupt request from a source which is not masked in the interrupt controller's interrupt mask register
- An interrupt request which has been enabled at the module of the interrupt's origin

7.3.1.1 Run Mode

Run mode is the normal system operating mode. Current consumption in this mode is related directly to the system clock frequency.

[1] f_{ref} = input reference frequency
f_{sys} = CLKOUT frequency
MFD ranges from 0 to 7.
RFD ranges from 0 to 7.

CAUTION

XTAL must be tied low in external clock mode when reset is asserted. If it is not, clocks could be suspended indefinitely.

The external clock is divided by two internally to produce the system clocks.

9.7.2 Clock Operation During Reset

In external clock mode, the system is static and does not recognize reset until a clock is applied to EXTAL.

In PLL mode, the PLL operates in self-clocked mode (SCM) during reset until the input reference clock to the PLL begins operating within the limits given in the electrical specifications.

If a PLL failure causes a reset, the system enters reset using the reference clock. Then the system clock source changes to the PLL operating in SCM. If SCM is not functional, the system becomes static. Alternately, if the LOCEN bit in SYNCR is cleared when the PLL fails, the system becomes static. If external reset is asserted, the system cannot enter reset unless the PLL is capable of operating in SCM.

9.7.3 System Clock Generation

In normal PLL clock mode, the default system frequency is two times the reference frequency after reset. The RFD[2:0] and MFD[2:0] bits in the SYNCR select the frequency multiplier.

When programming the PLL, do not exceed the maximum system clock frequency listed in the electrical specifications. Use this procedure to accommodate the frequency overshoot that occurs when the MFD bits are changed:

1. Determine the appropriate value for the MFD and RFD fields in the SYNCR. The amount of jitter in the system clocks can be minimized by selecting the maximum MFD factor that can be paired with an RFD factor to provide the required frequency.
2. Write a value of RFD (from step 1) + 1 to the RFD field of the SYNCR.
3. Write the MFD value from step 1 to the SYNCR.
4. Monitor the LOCK flag in SYNSR. When the PLL achieves lock, write the RFD value from step 1 to the RFD field of the SYNCR. This changes the system clocks frequency to the required frequency.

NOTE

Keep the maximum system clock frequency below the limit given in the Electrical Characteristics.

9.7.4 PLL Operation

In PLL mode, the PLL synthesizes the system clocks. The PLL can multiply the reference clock frequency by 2x to 9x, provided that the system clock frequency remains within the range listed in electrical specifications. For example, if the reference frequency is 2 MHz, the PLL can synthesize frequencies of 4 MHz to 18 MHz. In addition, the RFD can reduce the system frequency by dividing the output of the PLL.

Table 15-8. Processor to SDRAM Interface (8-Bit Port, 9-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A18	A19	A20	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	18	19	20	21	22	23
Column	0	1	2	3	4	5	6	7	8						
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

Table 15-9. Processor to SDRAM Interface (8-Bit Port,10-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A20	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	19	20	21	22	23
Column	0	1	2	3	4	5	6	7	8	18				
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

Table 15-10. Processor to SDRAM Interface (8-Bit Port,11-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A22	A23
Row	17	16	15	14	13	12	11	10	9	19	21	22	23
Column	0	1	2	3	4	5	6	7	8	18	20		
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Table 15-11. Processor to SDRAM Interface (8-Bit Port,12-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23
Row	17	16	15	14	13	12	11	10	9	19	21	23
Column	0	1	2	3	4	5	6	7	8	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Table 15-12. Processor to SDRAM Interface (8-Bit Port,13-Column Address Lines)

Processor Pins	A17	A16	A15	A14	A13	A12	A11	A10	A9	A19	A21	A23
Row	17	16	15	14	13	12	11	10	9	19	21	23
Column	0	1	2	3	4	5	6	7	8	18	20	22
SDRAM Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

Power-Up Sequence:

```
move.w   #0x0026, d0//Initialize DCR
move.w   d0, DCR
move.l   #0xFF880300, d0 //Initialize DACR0
move.l   d0, DACR0
move.l   #0x00740075, d0//Initialize DMR0
move.l   d0, DMR0
```

Precharge Sequence:

```
move.l   #0xFF880308, d0//Set DACR0[IP]
move.l   d0, DACR0
move.l   #0xBEADDEED, d0//Write and value to memory location to init. precharge
move.l   d0, 0xFF880000
```

Refresh Sequence:

```
move.l   #0xFF888300, d0//Enable refresh bit in DACR0
move.l   d0, DACR0
```

Mode Register Initialization Sequence:

```
move.l   #0x00600075, d0//Mask bit 19 of address
move.l   d0, DMR0
move.l   #0xFF888340, d0//Enable DACR0[IMRS]; DACR0[RE] remains set
move.l   d0, DACR0
move.l   #0x00000000, d0//Access SDRAM address to initialize mode register
move.l   d0, 0xFF800800
```

20.4.3 SYNC*n*

The SYNC*n* pin is for synchronization of the timer counter. It can be used to synchronize the counter with externally-timed or clocked events. A high signal on this pin clears the counter.

20.5 Memory Map and Registers

See Table 20-3 for a memory map of the two GPT modules. GPTA has a base address of IPSBAR + 0x1A_0000. GPTB has a base address of IPSBAR + 0x1B_0000.

NOTE

Reading reserved or unimplemented locations returns zeroes. Writing to reserved or unimplemented locations has no effect.

Table 20-3. GPT Modules Memory Map

IPSBAR Offset		Bits 7–0	Access[1]
GPTA	GPTB		
0x1A_0000	0x1B_0000	GPT IC/OC Select Register (GPTIOS)	S
0x1A_0001	0x1B_0001	GPT Compare Force Register (GPTCFORC)	S
0x1A_0002	0x1B_0002	GPT Output Compare 3 Mask Register (GPTOC3M)	S
0x1A_0003	0x1B_0003	GPT Output Compare 3 Data Register (GPTOC3D)	S
0x1A_0004	0x1B_0004	GPT Counter Register (GPTCNT)	S
0x1A_0006	0x1B_0006	GPT System Control Register 1 (GPTSCR1)	S
0x1A_0007	0x1B_0007	Reserved[2]	—
0x1A_0008	0x1B_0008	GPT Toggle-on-Overflow Register (GPTTOV)	S
0x1A_0009	0x1B_0009	GPT Control Register 1 (GPTCTL1)	S
0x1A_000A	0x1B_000a	Reserved(2)	—
0x1A_000B	0x1B_000b	GPT Control Register 2 (GPTCTL2)	S
0x1A_000C	0x1B_000c	GPT Interrupt Enable Register (GPTIE)	S
0x1A_000D	0x1B_000d	GPT System Control Register 2 (GPTSCR2)	S
0x1A_000E	0x1B_000e	GPT Flag Register 1 (GPTFLG1)	S
0x1A_000F	0x1B_000f	GPT Flag Register 2 (GPTFLG2)	S
0x1A_0010	0x1B_0010	GPT Channel 0 Register High (GPTC0H)	S
0x1A_0011	0x1Bb_0011	GPT Channel 0 Register Low (GPTC0L)	S
0x1A_0012	0x1B_0012	GPT Channel 1 Register High (GPTC1H)	S
0x1A_0013	0x1B_0013	GPT Channel 1 Register Low (GPTC1L)	S
0x1A_0014	0x1B_0014	GPT Channel 2 Register High (GPTC2H)	S
0x1A_0015	0x1B_0015	GPT Channel 2 Register Low (GPTC2L)	S
0x1A_0016	0x1B_0016	GPT Channel 3 Register High (GPTC3H)	S

Chapter 22
Queued Serial Peripheral Interface (QSPI)

22.1 Introduction

This chapter describes the queued serial peripheral interface (QSPI) module.

22.1.1 Block Diagram

Figure 22-1 illustrates the QSPI module.

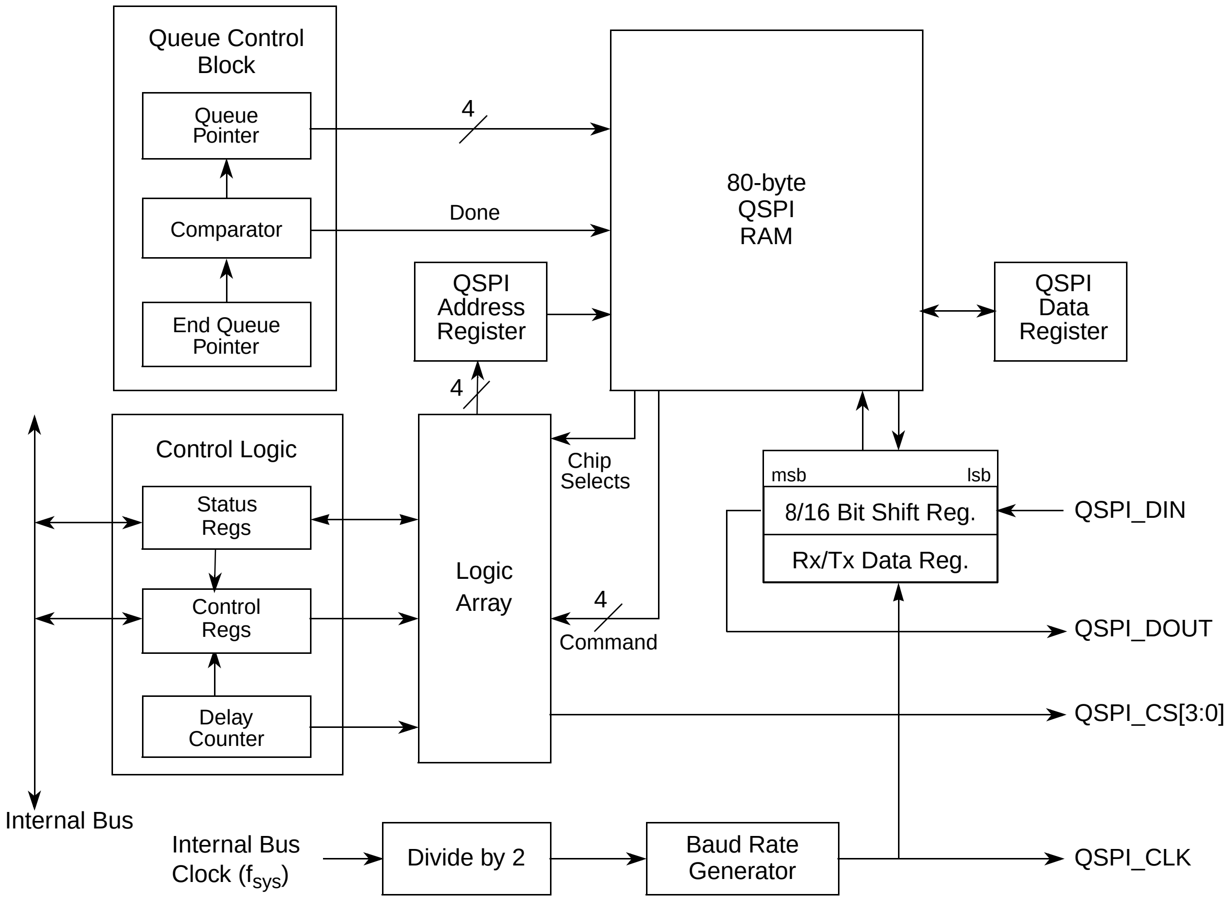


Figure 22-1. QSPI Block Diagram

24.3.4 Acknowledge

The transmitter releases the I2C_SDA line high during the acknowledge clock pulse as shown in Figure 24-9. The receiver pulls down the I2C_SDA line during the acknowledge clock pulse so that it remains stable low during the high period of the clock pulse.

If it does not acknowledge the master, the slave receiver must leave I2C_SDA high. The master can then generate a STOP signal to abort data transfer or generate a START signal (repeated start, shown in Figure 24-10 and discussed in Section 24.3.6, “Repeated START”) to start a new calling sequence.

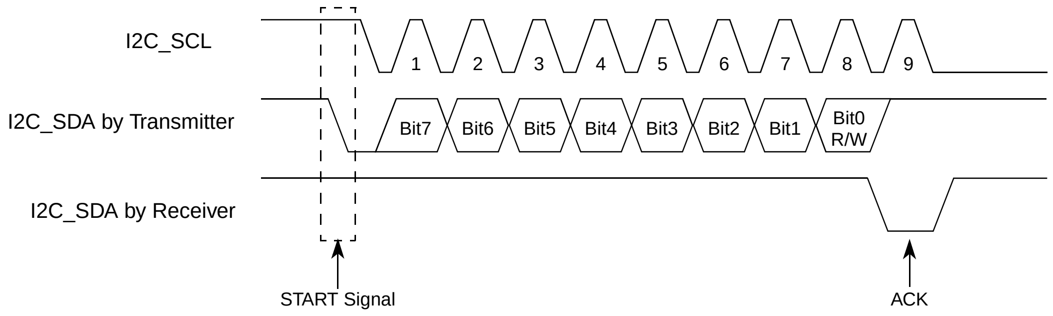


Figure 24-9. Acknowledgement by Receiver

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases I2C_SDA for the master to generate a STOP or START signal (Figure 24-9).

24.3.5 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of I2C_SDA while I2C_SCL is at logical high (see F in Figure 24-7). The master can generate a STOP even if the slave has generated an acknowledgment, at which point the slave must release the bus. The master may also generate a START signal following a calling address, without first generating a STOP signal. Refer to Section 24.3.6, “Repeated START.”

24.3.6 Repeated START

A repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication, as shown in Figure 24-10. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

Table 28-9. Queue 2 Operating Modes (continued)

MQ2[12:8]	Operating Modes
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	Externally triggered rising edge continuous-scan mode
10011	Externally triggered falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^{7}
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^{8}
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^{9}
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	Reserved mode

28.6.6 Status Registers

This subsection describes the QADC status registers.

28.6.6.1 QADC Status Register 0 (QASR0)

QASR0 contains information about the state of each queue and the current A/D conversion. The bits in this register are read anytime. For flag bits (CF1, PF1, CF2, PF2, TOR1, TOR2), writing a 1 has no effect; writing a 0 clears the bit. For QS[9:6] and CWP, writes have no effect. Stop mode resets this register.

The end of a queue is identified in the following cases:

- When execution is complete on the CCW in the location prior to the one pointed to by BQ2
- When the current CCW contains the end-of-queue code (channel 63) instead of a valid channel number

	7	6	5	4	3	2	1	0
Field	QS7	QS6	CWP5	CWP4	CWP3	CWP2	CWP1	CWP0
Reset	0000_0000							
R/W:	R							
Address	IPSBAR + 0x19_0010, 0x19_0011							

Figure 28-11. QADC Status Register 0 (QASR0)

Table 28-10. QASR0 Field Descriptions

Bit(s)	Name	Description
15, 13	CF*n*	Queue completion flag. Indicates that a queue scan has been completed. CF[1:2] is set by the QADC when the input channel sample requested by the last CCW in the queue is converted, and the result is stored in the result table. When CF*n* is set and queue completion interrupts are enabled (QACR*n*[CIE*n*] = 1), the QADC requests an interrupt. The interrupt request is cleared when a 0 is written to the CF1 bit after it has been read as a 1. Once set, CF1 can be cleared only by a reset or by writing a 0 to it. CF[1:2] is updated by the QADC regardless of whether the corresponding interrupt is enabled. This allows polled recognition of the queue scan completion.
14, 12	PF*n*	Queue pause flag. Indicates that a queue scan has reached a pause. PF[1:2] is set by the QADC when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. When PF*n* is set and interrupts are enabled (QACR*n*[PIE*n*] = 1), the QADC requests an interrupt. The interrupt request is cleared when a 0 is written to PF*n*, after it has been read as a 1. Once set, PF*n* can be cleared only by reset or by writing a 0 to it. PF1: 1 Queue 1 has reached a pause or gate closed before end-of-queue in gated mode. 0 Queue 1 has not reached a pause or gate has not closed before end-of-queue in gated mode. PF2: 1 Queue 2 has reached a pause. 0 Queue 2 has not reached a pause. See Table 28-11 for a summary of CCW pause bit response in all scan modes.
11–10	TOR*n*	Queue trigger overrun flag. Indicates that an unexpected trigger event has occurred for queue 1. TOR[1:2] can be set only while the queue is in the active state. Once set, TOR[1:2] is cleared only by a reset or by writing a 0 to it. 1 At least one unexpected queue 1 trigger event has occurred or queue 1 reaches an end-of-queue condition for the second time in externally gated continuous scan. 0 No unexpected queue 1 trigger events have occurred.

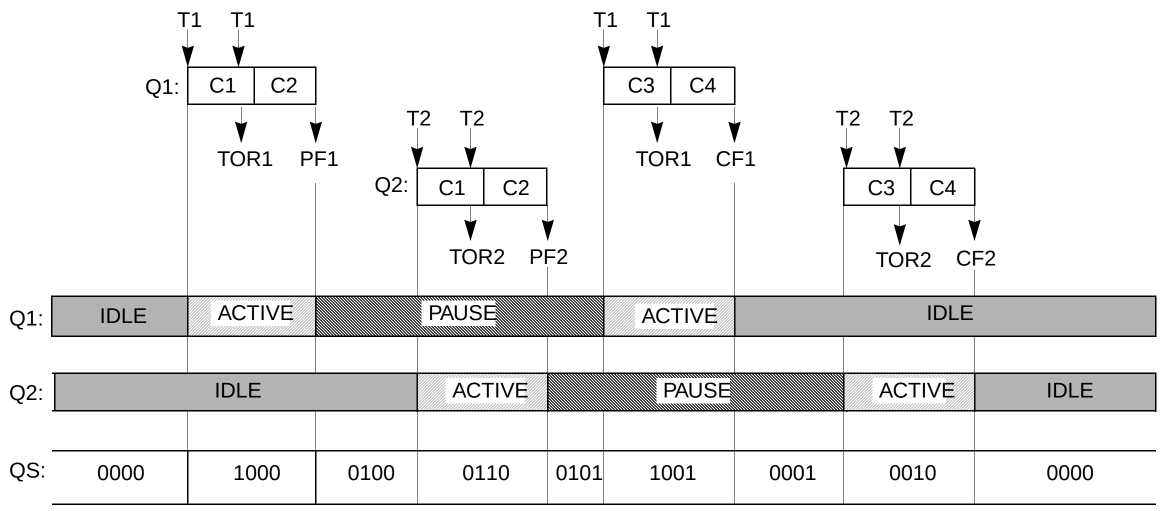


Figure 28-25. CCW Priority Situation 3

The next two situations consider trigger events that occur for the lower priority queue 2, while queue 1 is actively being serviced.

Situation S4 (Figure 28-26) shows that a queue 2 trigger event is recognized while queue 1 is active is saved, and as soon as queue 1 is finished, queue 2 servicing begins.

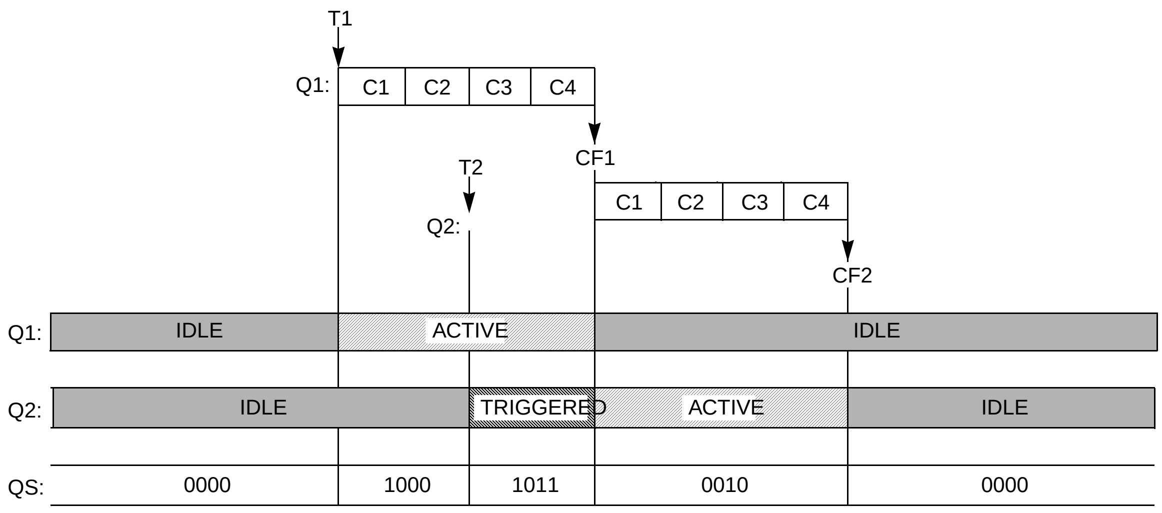


Figure 28-26. CCW Priority Situation 4

Situation S5 (Figure 28-27) shows that when multiple queue 2 trigger events are detected while queue 1 is busy, the trigger overrun error bit is set, but queue 1 execution is not disturbed. Situation S5 also shows that the effect of queue 2 trigger events during queue 1 execution is the same when the pause feature is used for either queue.

29.5.3.2 Reset Status Flags

For a POR reset, the POR and LVD bits in the RSR are set, and the SOFT, WDR, EXT, LOC, and LOL bits are cleared even if another type of reset condition is detected during the reset sequence for the POR.

If a loss-of-clock or loss-of-lock condition is detected while waiting for the current bus cycle to complete (5, 6) for an external reset request, the EXT, SOFT, and/or WDR bits along with the LOC and/or LOL bits are set.

If the RSR bits are latched (7) during the EXT, SOFT, and/or WDR reset sequence with no other reset conditions detected, only the EXT, SOFT, and/or WDR bits are set.

If the RSR bits are latched (4) during the internal reset sequence with the $\overline{\text{RSTI}}$ pin not asserted and no SOFT or WDR event, then the LOC and/or LOL bits are the only bits set.

For a LVD reset, the LVD bit in the RSR is set, and the SOFT, WDR, EXT, LOC, and LOL bits are cleared to 0 even if another type of reset condition is detected during the reset sequence for LVD.

Appendix B
Revision History

This appendix lists major changes between versions of the MCF5282UM document.

B.1 Changes Between Rev. 0 and Rev. 0.1

Table B-1. Rev. 0 to Rev. 0.1 Changes

Location	Description
Title page	Changed title from “MCF5282 ColdFire® Integrated Microprocessor User’s Manual” to “MCF5282 ColdFire® Microcontroller User’s Manual.”
33.1/33-1	Added “This product incorporates SuperFlash® technology licensed from SST.”
Table 9-4 on page 9-6	Changed equation in footnote to $f_{sys} = f_{ref} \times 2(MFD + 2)/2$ exp RFD; $f_{ref} \times 2(MFD + 2) \leq 80$ MHz, $f_{sys} \leq 66$ MHz.
Table 9-4 on page 9-6	Multiplied all PLL frequencies in table by 2.
Table 10-13 on page 10-13	Changed DTMRx to DTIMx.
Figure 10-1 on page 10-6	Changed bit numbers from 63–32 to 31–0.
Figure 10-3 on page 10-8	Changed bit numbers from 63–32 to 31–0.
Figure 10-5 on page 10-10	Changed bit numbers from 63–32 to 31–0.
14.2.4/14-22	Added Section 14.2.4, “Chip Configuration Signals.”
Table 14-3 on page 14-11	Added Table 14-3.
15.2/15-3	Added “Unlike the MCF5272, the MCF5282 does not have an independent SDRAM clock signal. For the MCF5282, the timing of the SDRAM controller is controlled by the CLKOUT signal.”
15.2.3.2/15-13	Added Section 15.2.3.2, “SDRAM Byte Strobe Connections.”
15.2.3.1/15-9	Added “Note: Because the MCF5282 has 24 external address lines, the maximum SDRAM address size is 128 Mbits.”
Figure 27-4 on page 27-6	Changed reset value to 0010_0000_0000_0000.
Chapter 30, “Debug Support”	Changed “PSTCLK” references to “CLKOUT.”
Figure 30-41 on page 30-45	Changed “$\overline{\text{TEA}}$” to “$\overline{\text{TA}}$.”