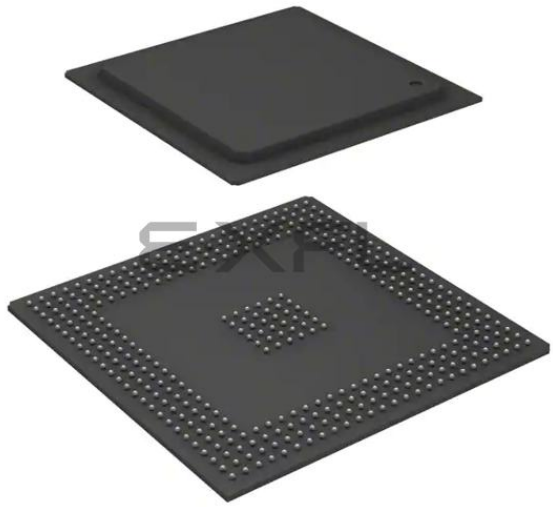

Welcome to E-XFL.COM

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	Not For New Designs
Core Processor	PowerPC
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	CANbus, EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	56
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 2.7V
Data Converters	A/D 32x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	388-BBGA
Supplier Device Package	388-PBGA (27x27)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mpc563mzp66r2

Tables

Table 2-14. MPC561/MPC563 Signal Reset State (continued)

Signal List[1]	Voltage	Slew Rate Controlled Option?	Drive Load (pF)[2]	Reset State	Hysteresis Enabled?	Function After HRESET, PORESET/TRST
IRQ4 /	2.6 V	No	NA	PD until reset negates [6, 8]	Yes	IRQ4
AT2 /	2.6 V	No	50 ; 25	PD until reset negates [6, 8]	No	
SGPIOC4 [3]	5 V	Yes	50 ; 50 [5]	PD until PRDS is set	Yes	
IRQ5[10] /	2.6 V	No	NA	PU2.6 until reset negates	Yes	MODCK1 until reset negates, then IRQ5
SGPIOC5 [7] /	2.6 V	No	50 ; 25	PU2.6 until PRDS is set	No	
MODCK1[11]	2.6 V	No	NA	PU2.6V until reset negates	No	
IRQ[6:7] [10] /	2.6 V	No	NA	PU2.6 until SPRDS is set	Yes	MODCK[2:3] until reset negates, then IRQ[6:7]
MODCK[2:3] [11]	2.6 V	No	NA	PU2.6 until reset negates	No	
PULL_SEL	5 V	No	NA	PU5, external pull device required	No	PULL_SEL
TSIZ[0:1] [7]	2.6 V	No	50 ; 25	PD when driver not enabled or until SPRDS is set	No	TSIZ[0:1]
RD/WR [7]	2.6 V	No	50 ; 25	PU2.6 when driver not enabled or until SPRDS is set	No	RD/WR
BURST [7]	2.6 V	No	50 ; 25	PU2.6 when driver not enabled or until SPRDS is set	No	BURST
BDIP [7]	2.6 V	No	50 ; 25	PU2.6 when driver not enabled or until SPRDS is set	No	BDIP
TS [7,12]	2.6 V	No	50 ; 25	PU2.6 when driver not enabled or until SPRDS is set	No	TS

When the operand falls in the range of double denormalized numbers it is considered a programming error. The hardware will handle this case as if the operand was ZERO.

The following check is done on the stored operand in order to determine whether it is a denormalized single-precision operand and invoke the floating-point assist interrupt handler:

$$(frS[1:11] \neq 0) \text{ AND } (frS[1:11] \leq 896) \qquad \textit{Eqn. 3-1}$$

Refer to the *RCPU Reference Manual* (Floating-Point Assist for Denormalized Operands) for complete description of handling denormalized floating-point numbers.

3.13.10.8 Optional Instructions

No optional instructions are supported.

3.14 Virtual Environment Architecture (VEA)

3.14.1 Atomic Update Primitives

Both the lwarx and stwcx instructions are implemented according to the PowerPC ISA architecture requirements. The MPC561/MPC563 does not provide support for snooping an external bus activity outside the chip. The provision is made to cancel the reservation inside the MPC561/MPC563 by using the $\overline{CR}$ and $\overline{KR}$ input signals. Internal buses are snooped for RCPU accesses, and the reservation mechanism can be used for multitask single master applications.

3.14.2 Effect of Operand Placement on Performance

The load/store unit hardware supports all of the PowerPC ISA load/store instructions. An optimal performance is obtained for naturally aligned operands. These accesses result in optimal performance (one bus cycle) for up to four bytes in size and good performance (two bus cycles) for double precision floating-point operands. Unaligned operands are supported in hardware and are broken into a series of aligned transfers. The effect of operand placement on performance is as stated in the VEA, except for the case of 8-byte operands. In that case, since the RCPU uses a 32-bit wide data bus, the performance is good rather than optimal.

3.14.3 Storage Control Instructions

The RCPU does not implement the following cache control instructions: icbi, dcbt, dcbi, dcbf, dcbz, dcbst, and dcbtst .

3.14.4 Instruction Synchronize (isync) Instruction

The isync instruction causes a reflect which waits for all prior instructions to complete and then executes the next sequential instruction. Any instruction after an isync will see all effects of prior instructions.

the input source to the SPLL (main system oscillator or EXTCLK). MODCK1, MODCK2, and MODCK3 together determine the multiplication factor at reset and the functionality of limp mode.

If the configuration of PITRTCLK and TMBCLK and the SPLL multiplication factor is to remain unchanged in power-down low-power mode, the MODCK signals should not be sampled at wake-up from this mode. In this case the $\overline{\text{PORESET}}$ pin should remain negated and $\overline{\text{HRESET}}$ should be asserted during the power supply wake-up stage.

When MODCK1 is cleared, the output of the main oscillator is selected as the input to the SPLL. When MODCK1 is asserted, the external clock input (EXTCLK pin) is selected as the input to the SPLL. In all cases, the system clock frequency ($freq_{gclk2}$) can be reduced by the DFNH[0:2] bits in the SCCR. Note that $freq_{gclk2(max)}$ occurs when the DFNH bits are cleared.

The TBS bit in the SCCR selects the time base clock to be either the SPLL input clock or GCLK2. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

The PITRTCLK frequency and source are specified by the RTDIV and RTSEL bits in the SCCR. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

When the $\overline{\text{PORESET}}$ pin is negated (driven to a high value), the MODCK1, MODCK2, and MODCK3 values are not affected. They remain the same as they were defined during the most recent power-on reset.

Table 8-1 shows the clock configuration modes during power-on reset ($\overline{\text{PORESET}}$ asserted).

NOTE

The MODCK[1:3] are shared functions with IRQ[5:7]. If $\overline{\text{IRQ}}$[5:7] are used as interrupts, the interrupt source should be removed during $\overline{\text{PORESET}}$ to insure the MODCK pins are in the correct state on the rising edge of $\overline{\text{PORESET}}$.

Table 8-1. Reset Clocks Source Configuration

MODCK[1:3][1]	Default Values after $\overline{\text{PORESET}}$						SPLL Options
	LME	RTSEL	RTDIV	MF + 1	PITCLK Division	TMBCLK Division	
000	0	0	0	1	4	4	Used for testing purposes.
001	0	0	1	1	256	16	Normal operation, PLL enabled. Main timing reference is crystal osc (20 MHz). Limp mode disabled.
010	1	0	1	5	256	4	Normal operation, PLL enabled. Main timing reference is crystal osc (4 MHz). Limp mode enabled.
011	1	0	1	1	256	16	Normal operation, PLL enabled. Main timing reference is crystal osc (20 MHz). Limp mode enabled.

The return to normal-high mode from normal-low, doze-high, low, and sleep mode is accomplished with the asynchronous interrupt. The sources of the asynchronous interrupt are:

- Asynchronous wake-up interrupt from the interrupt controller
- RTC, PIT, or time base interrupts (if enabled)
- Decrementer exception

The system responds quickly to asynchronous interrupts. The wake-up time from normal-low, doze-high, doze-low, and sleep mode caused by an asynchronous interrupt or a decrementer exception is only three to four clock cycles of maximum system frequency. In 40-MHz systems, this wake-up requires 75 to 100 ns. The asynchronous wake-up interrupt from the interrupt controller is level sensitive one. It will therefore be negated only after the reset of interrupt cause in the interrupt controller.

The timers' (RTC, PIT, time base, or decrementer) interrupts indications set status bits in the PLPRCR (TMIST). The clock module considers this interrupt to be pending asynchronous interrupt as long as the TMIST is set. The TMIST status bit should be cleared before entering any low-power mode.

Table 8-7 summarizes wake-up operation for each of the low-power modes.

Table 8-6. Power Mode Wake-Up Operation

Operation Mode	Wake-up Method	Return Time from Wake-up Event to Normal-High
Normal-low ("gear")	Software or Interrupt	Asynchronous interrupts: 3-4 maximum system cycles Synchronous interrupts: 3-4 actual system cycles
Doze-high	Interrupt	
Doze-low	Interrupt	
Sleep	Interrupt	3-4 maximum system clocks
Deep-sleep	Interrupt	< 500 Oscillator Cycles 125 µs – 4 MHz 25 µs – 20 MHz
Power-down	Interrupt	< 500 oscillator cycles + power supply wake-up
IRAMSTBY	External	Power-on sequence

8.7.3.1 Exiting from Normal-Low Mode

In normal mode (as well as doze mode), if the PLPRCR[CSRC] bit is set, the system toggles between low frequency (defined by PLPRCR[DFNL]) and high frequency (defined by PLPRCR[DFNH]. The system switches from normal-low mode to normal-high mode if either of the following conditions is met:

- An interrupt is pending from the interrupt controller; or
- The MSR[POW] bit is cleared (power management is disabled).

When neither of these conditions are met, the PLPRCR[CSRC] bit is set, and the asynchronous interrupt status bits are reset, the system returns to normal-low mode.

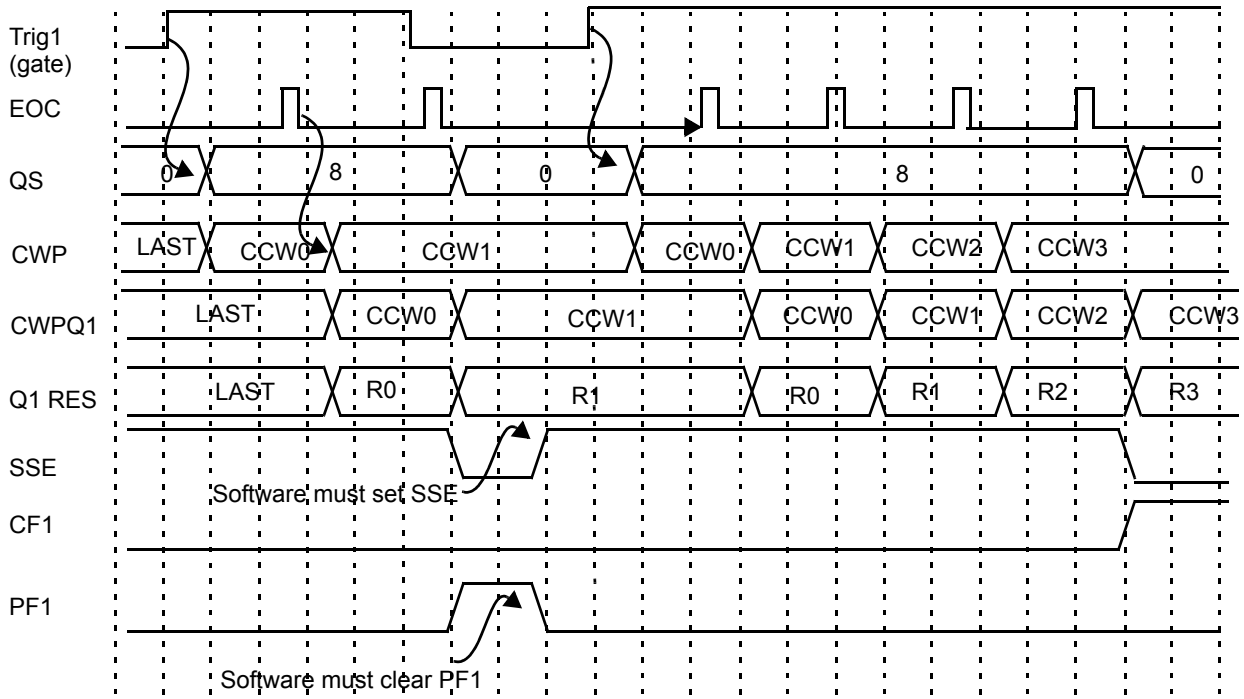

Figure 13-47. Gated Mode, Single-Scan Timing

Example 3 below shows the timing for conversions in gated continuous-scan mode with the same assumptions in the amended definition for the PF bit in this mode to reflect the condition that a gate closing occurred before the queue completed is a proposal under consideration at this time as example 2.

NOTE

At the end of Q1, the completion flag CF1 sets and the queue restarts. Also, note that if the queue starts a second time and completes, the trigger overrun flag TOR1 sets.

Table 14-14. Queue 2 Operating Modes (continued)

MQ2[3:7]	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	External trigger rising edge continuous-scan mode
10011	External trigger 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

NOTE

If BQ2 was assigned to the CCW that queue 1 is currently working on, then that conversion is completed before BQ2 takes effect.

Each time a CCW is read for queue 1, the CCW location is compared with the current value of the BQ2 pointer to detect a possible end-of-queue condition. For example, if BQ2 is changed to CCW3 while queue 1 is converting CCW2, queue 1 is terminated after the conversion is completed. However, if BQ2 is changed to CCW1 while queue 1 is converting CCW2, the QADC64E would not recognize a BQ2 end-of-queue condition until queue 1 execution reached CCW1 again, presumably on the next pass through the queue.

compared. When a match occurs, the SPIF flag is set and the QSPI stops and clears SPE, unless wraparound mode is enabled.

At reset, NEWQP is initialized to 0x0. When the QSPI is enabled, execution begins at queue address 0x0 unless another value has been written into NEWQP. ENDQP is initialized to 0x0 at reset but should be changed to the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When NEWQP changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to 0x0 transfers only the data in transmit RAM location 0x0.

15.6.4.1 Enabling, Disabling, and Halting the SPI

The SPE bit in the SPCR1 enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS0/$\overline{SS}$ pin to respond to the external initialization of a serial transfer.

When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access to all 160 bytes of the QSPI RAM. The QSPI can read-only the transmit data segment and the command control segment and can write-only the receive data segment of the QSPI RAM.

The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS0/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS0/$\overline{SS}$ is driven low externally.

Setting the HALT bit in SPCR3 stops the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is set, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip select pins with the value designated by the last command before the halt. If CONT was cleared, the QSPI drives the peripheral chip-select pins to the value in register PORTQS.

If HALT is set during the last command in the queue, the QSPI completes the last command, sets both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF or clear SPE. QSPI execution continues when the CPU clears HALT.

To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE can then be safely cleared, providing an orderly method of shutting down the QSPI quickly after the current serial transfer is completed. The CPU can disable the QSPI immediately by clearing SPE. However, loss of data from a current serial transfer may result and confuse an external SPI device.

15.6.4.2 QSPI Interrupts

The QSPI has three possible interrupt sources but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the interrupt cause must ascertained by reading the SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the interrupt source.

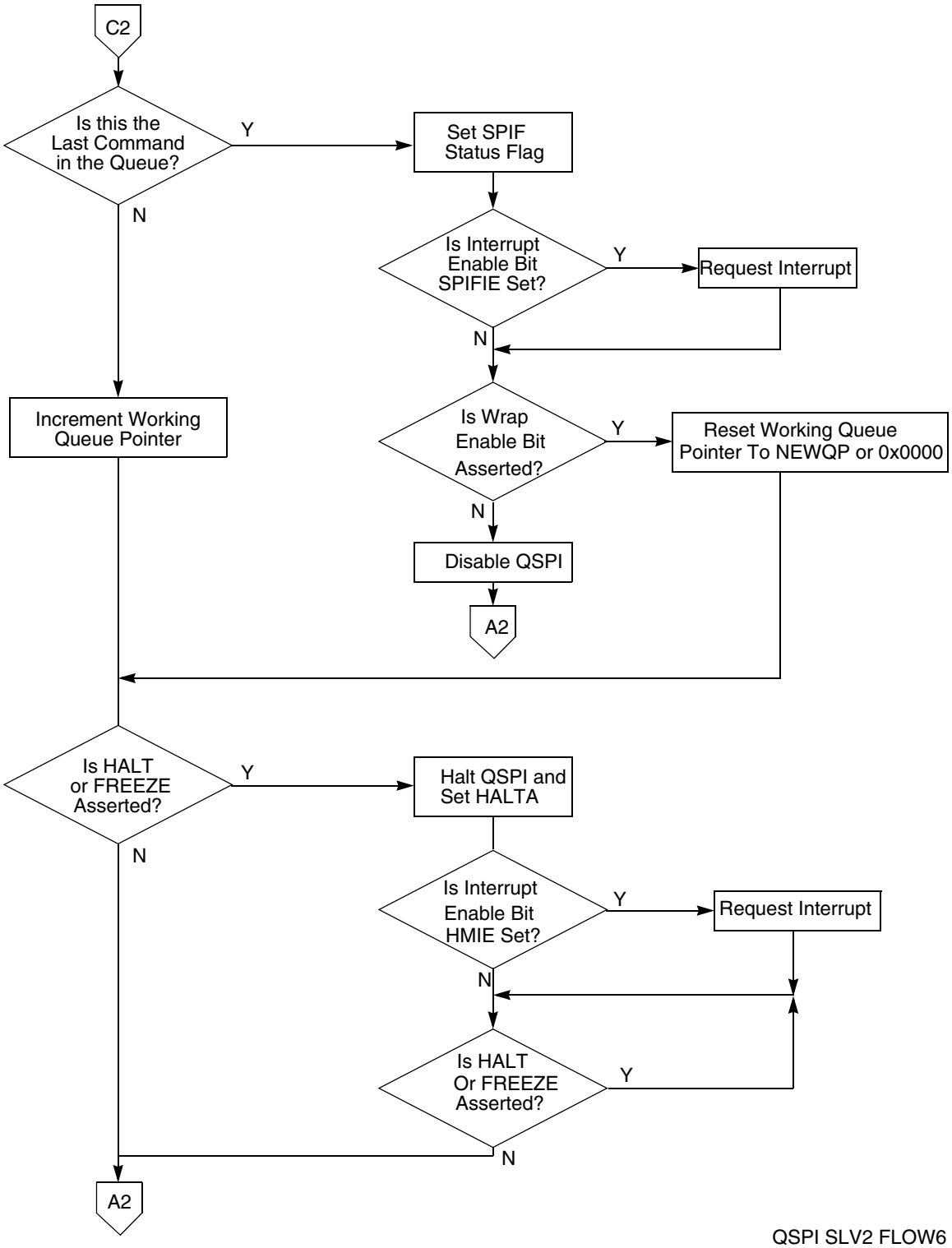

Figure 15-23. Flowchart of QSPI Slave Operation (Part 2)

— Possible use of signal as I/O port when MDASM function is not needed

- MIOS14 pulse width modulation submodule (MPWMSM):
  - — Output pulse width modulated (PWM) signal generation with no software involvement
  - — Built-in 8-bit programmable prescaler clocked by the MCPSM
  - — PWM period and pulse width values provided by software:
    - – Double-buffered for glitch-free period and pulse width changes
    - – Two-cycle minimum output period/pulse-width increment (50 ns @ 40 MHz)
    - – 50% duty-cycle output maximum frequency: 10 MHz
    - – Up to 16 bits output pulse width resolution
    - – Wide range of periods:
      - 16 bits of resolution: period range from 3.27 ms (with 50-ns steps) to 6.71 s (with 102.4 µs steps)
      - Eight bits of resolution: period range from 12.8 µs (with 50-ns steps) to 26.2 ms (with 102.4-µs steps)
    - – Wide range of frequencies:
      - Maximum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-2 prescaler selection: 305 Hz (3.27 ms)
      - Minimum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-4096 prescaler selection: 0.15 Hz (6.7 s)
      - Maximum output frequency at f_{SYS} = 40 MHz with eight bits of resolution and divide-by-2 prescaler selection: 78125 Hz (12.8 µs)
      - Minimum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-4096 prescaler selection: 38.14 Hz (8.2 ms)
  - — Programmable duty cycle from 0% to 100%
  - — Possible interrupt generation after every period
  - — Software selectable output pulse polarity
  - — Software readable output signal status
  - — Possible use of signal as I/O port when PWM function is not needed
- MIOS14 16-bit parallel port I/O submodule (MPIOSM):
  - — Up to 16 parallel I/O signals per MPIOSM
  - — Uses four 16-bit registers in the address space, one for data and one for direction and two reserved
  - — Simple data direction register (DDR) concept for selection of signal direction

17.2.1 Submodule Numbering, Naming, and Addressing

A block is a group of four 16-bit registers. Each of the blocks within the MIOS14 addressing range is assigned a block number. The first block is located at the base address of the MIOS14. The blocks are numbered sequentially starting from 0.

Table 17-18. MDASM Address Map (continued)

Address	Register
0x30 60E4	MDASM28 Status/Control Register Duplicated (MDASMSCRD)
0x30 60E6	MDASM28 Status/Control Register (MDASMSCR)
MDASM29	
0x30 60E8	MDASM29 Data A Register (MDASMAR)
0x30 60EA	MDASM29 Data B Register (MDASMBR)
0x30 60EC	MDASM29 Status/Control Register Duplicated (MDASMSCRD)
0x30 60EE	MDASM29 Status/Control Register (MDASMSCR)
MDASM30	
0x30 60F0	MDASM30 Data A Register (MDASMAR)
0x30 60F2	MDASM30 Data B Register (MDASMBR)
0x30 60F4	MDASM30 Status/Control Register Duplicated (MDASMSCRD)
0x30 60F6	MDASM30 Status/Control Register (MDASMSCR)
MDASM31	
0x30 60F8	MDASM31 Data A Register (MDASMAR)
0x30 60FA	MDASM31 Data B Register (MDASMBR)
0x30 60FC	MDASM31 Status/Control Register Duplicated (MDASMSCRD)
0x30 60FE	MDASM31 Status/Control Register (MDASMSCR)

17.9.6.2 MDASM Data A (MDASMAR) Register

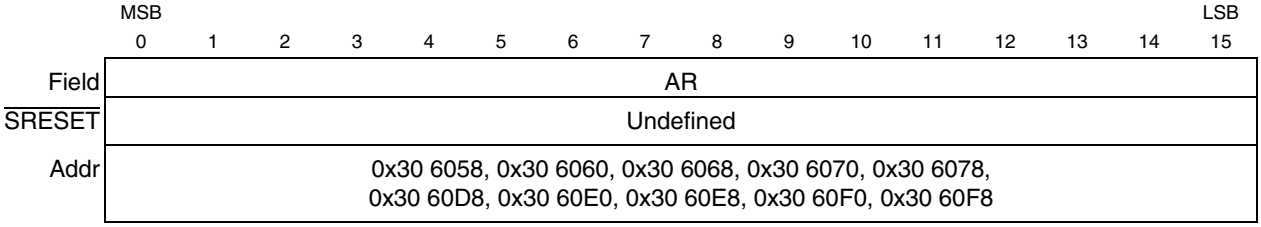

Figure 17-22. MDASM Data A Register (MDASMAR)

Table 17-25. MPWMSM Address Map (continued)

Address	Register
0x30 601C	MPWMSM3 Count Register (MPWMCNTR)
0x30 601E	MPWMSM3 Status/Control Register (MPWMSCR)
MPWMSM4	
0x30 6020	MPWMSM4 Period Register (MPWMPERR)
0x30 6022	MPWMSM4 Pulse Register (MPWMPULR)
0x30 6024	MPWMSM4 Count Register (MPWMCNTR)
0x30 6026	MPWMSM4 Status/Control Register (MPWMSCR)
MPWMSM5	
0x30 6028	MPWMSM5 Period Register (MPWMPERR)
0x30 602A	MPWMSM5 Pulse Register (MPWMPULR)
0x30 602C	MPWMSM5 Count Register (MPWMCNTR)
0x30 602E	MPWMSM5 Status/Control Register (MPWMSCR)
MPWMSM16	
0x30 6080	MPWMSM16 Period Register (MPWMPERR)
0x30 6082	MPWMSM16 Pulse Register (MPWMPULR)
0x30 6084	MPWMSM16 Count Register (MPWMCNTR)
0x30 6086	MPWMSM16 Status/Control Register (MPWMSCR)
MPWMSM17	
0x30 6088	MPWMSM17 Period Register (MPWMPERR)
0x30 608A	MPWMSM17 Pulse Register (MPWMPULR)
0x30 608C	MPWMSM17 Count Register (MPWMCNTR)
0x30 608E	MPWMSM17 Status/Control Register (MPWMSCR)
MPWMSM18	
0x30 6090	MPWMSM18 Period Register (MPWMPERR)
0x30 6092	MPWMSM18 Pulse Register (MPWMPULR)
0x30 6094	MPWMSM18 Count Register (MPWMCNTR)
0x30 6096	MPWMSM18 Status/Control Register (MPWMSCR)
MPWMSM19	
0x30 6098	MPWMSM19 Period Register (MPWMPERR)
0x30 609A	MPWMSM19 Pulse Register (MPWMPULR)
0x30 609C	MPWMSM19 Count Register (MPWMCNTR)

It is typical to use the pulse width modulation mode of the MDASM without interrupts, although an interrupt can be enabled to occur on the leading edge. When the output is an unchanging repetitive waveform, the MDASM continuously generates the signal without any software intervention. When the software needs to change the pulse width, a new trailing edge time is written to the MDASM. The output is changed on the next full pulse. When the software needs to change the output at a regular rate, such as an acceleration curve, the leading edge interrupt gives the software one period time to update the new trailing edge time.

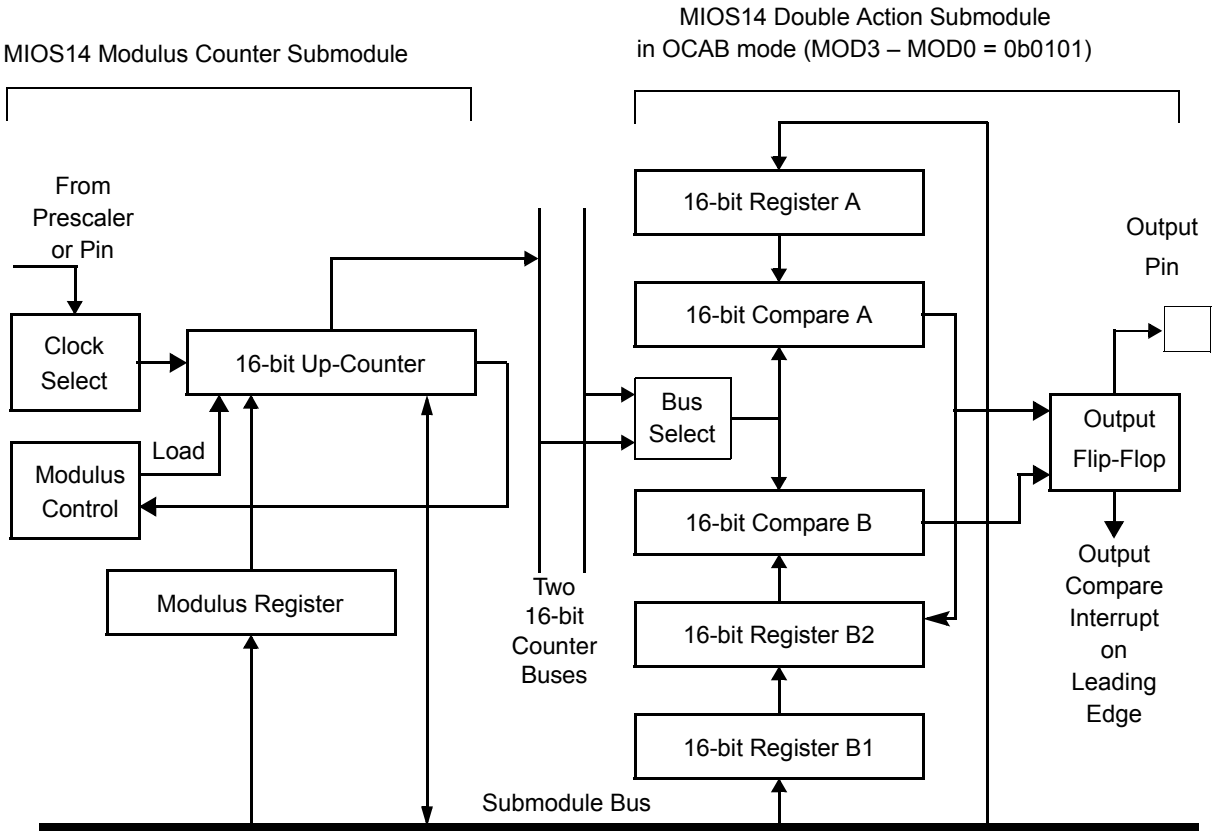

Figure 17-46. MIOS14 Example: Pulse Width Modulation Output

17.13.5 MIOS14 Input Pulse Accumulation

Counting the number of pulses on an input signal is another capability of the MIOS14. Pulse accumulation uses an MMCSM. Since the counters in the counter submodules are software accessible, pulse accumulation does not require the use of an action submodule. The pulse accumulation can operate continuously, interrupting only on binary overflow of the 16-bit counter. When an MMCSM is used, an interrupt can instead be created when the pulse accumulation reaches a preprogrammed value. To do that, the two's complement of the value is put in the modulus register and the interrupt occurs when the counter overflows.

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Field	RX_DATA															
$\overline{\text{SRESET}}$	0000_0000_0000_0000															
Addr	0x30 5C16															

Figure 18-18. Receive Data Register (RX_DATA)

18.4.6 Receive Shift Register (RX_SHIFTER)

RX_SHIFTER receives data serially from the PPM input signals PPM_RX[0:1] (depending on the value of PPMPCR[OP_16_8]). Data bits are shifted in on every PPM_TCLK cycle. Data in the RX_SHIFTER register is delivered directly to the MPC561/MPC563 internal modules with no wait time.

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Field	RX_SHIFTER															
$\overline{\text{SRESET}}$	0000_0000_0000_0000															
Addr	0x30 5C1A															

Figure 18-19. Receive Shifter Register (RX_SHIFTER)

18.4.7 Transmit Data Register (TX_DATA)

TX_DATA contains data from the internally multiplexed modules that is to be transmitted from the PPM module on the PPM_TX[1:0] signals (depending on the value in PPMPCR[OP_16_8]). Data bits are transmitted serially (shifted out) on each PPM_TCLK cycle. The data is shifted out least significant bit (LSB) first, therefore TX_DATA15 first, TX_DATA0 last.

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Field	TX_DATA															
$\overline{\text{SRESET}}$	0000_0000_0000_0000															
Addr	0x30 5C1E															

Figure 18-20. Transmit Data Register (TX_DATA)

18.4.8 General-Purpose Data Out (GPDO)

GPDO is an internal register whose data can be transmitted serially through the PPM. By default, the transmit configuration registers are set to transmit from this register. The value in GPDO[0:15] is written into TX_DATA[0:15].

Table 19-7. TPUMCR Bit Description (continued)

Bits	Name	Description
9	PSCK	Standard prescaler clock. Note that this bit has no effect if the extended prescaler is selected (EPSCKE = 1). 0 $f_{SYS} \div 32$ is input to TCR1 prescaler, if standard prescaler is selected 1 $f_{SYS} \div 4$ is input to TCR1 prescaler, if standard prescaler is selected
10	TPU3	TPU3 enable. The TPU3 enable bit provides compatibility with the TPU. If running TPU code on the TPU3, the microcode size should not be greater than 2 Kbytes and the TPU3 enable bit should be cleared to zero. The TPU3 enable bit is write-once after reset. The reset value is one, meaning that the TPU3 will operate in TPU3 mode. 0 TPU mode; zero is the TPU reset value 1 TPU3 mode; one is the TPU3 reset value NOTE: The programmer should not change this value unless necessary when developing custom TPU microcode.
11	T2CSL	TCR2 counter clock edge. This bit and the T2CG control bit determine the clock source for TCR2. Refer to Section 19.3.9, “Prescaler Control for TCR2” for details.
12:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TPU3 implementations that use hardware interrupt arbitration.

[1] If all TPUs connected to a DPTRAM are stopped, the DPTRAM is accessible.

19.4.2 Development Support Control Register (DSCR)

This register is accessible only when the TPU is in test mode; see Section 19.4.14, “Factory Test Registers.”

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Field	HOT4	—				BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
$\overline{\text{SRESET}}$	0000_0000_0000_0000															
Addr	0x30 4004 (TPU_A), 0x30_4404 (TPU_B)															

Figure 19-6. DSCR — Development Support Control Register

Table 19-8. DSCR Bit Descriptions

Bits	Name	Description
0	HOT4	Hang On T4 [1] 0 Exit wait on T4 state caused by assertion of HOT4 1 Enter wait on T4 state
1:4	—	Reserved
5	BLC	Branch Latch Control 0 Latch conditions into branch condition register before exiting halted state 1 Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period
6	CLKS	Stop clocks (to TCRs) 0 Do not stop TCRs 1 Stop TCRs during the halted state

23.2.2.1 Load/Store Support

There are two load/store address comparators E, and F. Each compares the 32 address bits and the cycle's attributes (read/write). The two least-significant bits are masked (ignored) whenever a word is accessed and the least-significant bit is masked whenever a half-word is accessed. (For more information refer to Section 23.2.1.2, "Byte and Half-Word Working Modes"). Each comparator generates two output signals: equal and less than. These signals are used to generate one of the following four events (one from each comparator): equal, not equal, greater than, less than.

There are two load/store data comparators (comparators G,H) each is 32 bits wide and can be programmed to treat numbers either as signed values or as unsigned values. Each data comparator operates as four independent byte comparators. Each byte comparator has a mask bit and generates two output signals: equal and less than, if the mask bit is not set. Therefore, each 32 bit comparator has eight output signals.

These signals are used to generate the "equal and less than" signals according to the compare size programmed (byte, half-word, word). When operating in byte mode all signals are significant, when operating in half-word mode only four signals from each 32 bit comparator are significant. When operating in word mode only two signals from each 32 bit comparator are significant.

From the new "equal and less than" signals and according to the compare type programmed one of the following four match events are generated: equal, not equal, greater than, less than. Therefore, from the two 32-bit comparators eight match indications are generated: Gmatch[0:3], Hmatch[0:3].

According to the lower bits of the address and the size of the cycle, only match indications that were detected on bytes that have valid information are validated, the rest are negated. Note that if the cycle executed has a smaller size than the compare size (e.g., a byte access when the compare size is word or half-word) no match indication will be asserted.

Using the match indication signals four load/store data events are generated in the following way.

Table 23-7. Load/Store Data Events

Event Name	Event Function[1]
G	(Gmatch0 | Gmatch1 | Gmatch2 | Gmatch3)
H	(Hmatch0 | Hmatch1 | Hmatch2 | Hmatch3)
(G&H)	((Gmatch0 & Hmatch0) | (Gmatch1 & Hmatch1) | (Gmatch2 & Hmatch2) | (Gmatch3 & Hmatch3))
(G | H)	((Gmatch0 | Hmatch0) | (Gmatch1 | Hmatch1) | (Gmatch2 | Hmatch2) | (Gmatch3 | Hmatch3))

1 '&' denotes a logical AND, 'I' denotes a logical OR

The four load/store data events together with the match events of the load/store address comparators and the instruction watchpoints are used to generate the load/store watchpoints and breakpoint according to the programming.

When the RCPU is in debug mode, program trace is not allowed. If program trace is enabled, a program trace synchronization message is generated when debug mode exits.

When the RCPU is in debug mode, data trace and R/$\overline{W}$ access are allowed.

The flow chart in Figure 24-83 shows RCPU development access configuration via READI. The modes of RCPU development access via READI are described below. Allowed modes are also summarized in Table 24-8 of Section 24.14.2.4, "RCPU Development Access Flow Diagram."

24.14.2.1 Enabling RCPU Development Access Via READI Signals

Reset sequencing is done by the tool to initialize the READI signals and registers by asserting $\overline{RSTI}$ (the device sends out the device ID message after the $\overline{RSTI}$ negation). System reset is held by the tool until the READI module is reset and initialized with desired RCPU development access setting.

NOTE

The READI module will ignore any incoming DSDI data messages when the module is not configured for RCPU development access.

24.14.2.2 Entering Background Debug Mode (BDM) Via READI Signals

There are three ways to enter debug mode (provided debug mode has been enabled):

1. Enter debug mode (halted state) out-of-system reset through READI module configuration. This is displayed in Figure 24-84.
2. Enter debug mode by downloading breakpoint instructions through RCPU development access when in non-debug (running) mode.
3. Enter debug mode if an exception or interrupt occurs.

When entering debug mode following an exception/breakpoint, the RCPU signals VFLS[0:1] are equal to 0b11. This causes READI to send a BDM status message to the tool indicating that the RCPU has entered debug mode and is now expecting instructions from the READI signals.

Debug mode enabling through READI and entering debug mode out of system reset is done by setting the following bits in the DC register (DME=0b1, DOR=0b1) during system reset. Debug mode entry causes RCPU to halt.

24.14.2.3 Non-Debug Mode Access of RCPU Development Access

The RCPU development access can be also be used while the RCPU is not halted (in debug mode). This feature is used to send in breakpoints or synchronization events to the RCPU. Please refer to Chapter 23, "Development Support" for further details.

Non-debug mode access of RCPU development can be achieved by configuring the READI module to take control of RCPU development access during module configuration of the DC register (DME=0b0, DOR=0bx).

Table F-10. Bus Operation Timing (continued)

Note: (V_{DD} = 2.6 V ± 0.1 V, V_{DDH} = 5.0 V ± 0.25 V, $T_A = T_L$ to T_H, 50 pF load unless noted otherwise)

	Characteristic	40 MHz		56 MHz[1]		Unit
		Min	Max	Min	Max	
26b	$\overline{CS}$ negated to D[0:31], High Z -GPCM- write access, ACS = '00', TRLX = '0' & CSNT = '0'	3		2.25		ns
26c	$\overline{CS}$ negated to D[0:31], High Z -GPCM- write access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0	8		5.71		ns
26d	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', EBDF = 0	28		20		ns
26e	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0	28		20		ns
26f	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] negated to D[0:31] HighZ -GPCM- write access, TRLX = '0', CSNT = '1', EBDF = 1	5		3.75		ns
26g	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1	5		3.75		ns
26h	$\overline{WE}$[0:3]/$\overline{BE}$[0:3] negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', EBDF = 1	24		17.25		ns
26i	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1	24		17.25		ns
27	$\overline{CS}$, $\overline{WE}$[0:3]/$\overline{BE}$[0:3] negated to ADDR[8:31] invalid -GPCM- write access[5]	0.75		1		ns

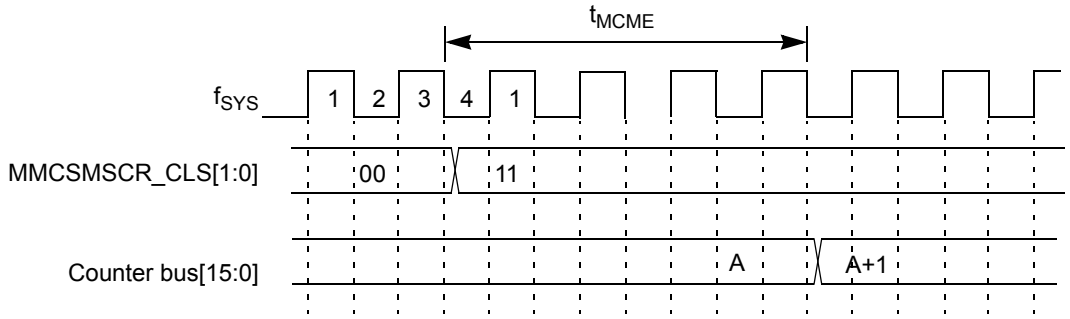

Figure F-56. MMCSM Prescaler Clock Select To Counter Bus Increment Timing Diagram

F.20.3 MDASM Timing Characteristics

Table F-26. MDASM Timing Characteristics

Note: All delays are in system clock periods.

Characteristics	Symbol	Min	Max
Input Modes: (IPWM, IPM, IC, DIS)			
MDASM input pin period	t_{PPER}	4	—
MDASM pin low time	t_{PLO}	2	—
MDASM pin high time	t_{PHI}	2	—
Input capture resolution	t_{CAPR}	—	2
Input pin to Counter Bus capture delay	t_{PCAP}	1	3[1]
Input pin to interrupt flag delay	t_{PFLG}	2	3
Input pin to PIN delay	t_{PIN}	1	2
Counter bus resolution	t_{CBR}	—	2[2]
Output Modes: (OC, OPWM)			
Output pulse width[3]	t_{PULW}	2	—
Compare resolution[3]	t_{COMR}	—	2[2]
Counter Bus to pin change	t_{CBP}	3	
Counter Bus to interrupt flag set.	t_{CBFLG}	3	

1 If the counter bus capture occurs when the counter bus is changing then the capture is delayed one cycle. In situations where the counter bus is stable when the input capture occurs the t_{PCAP} has a maximum delay of two cycles (the one-cycle uncertainty is due to the synchronizer).

2 Maximum resolution is obtained by setting CPSMPSL[3:0] =0x2 and MDASMSCR_CP[7:0] =0xFF.

3 Maximum output resolution and pulse width depends on counter (e.g., MMCSM) and MCPSM prescaler settings.

Table F-28. MPC561/MPC563 Signal Names and Pin Names (continued)

Signal Name	Pin Name	Ball Assignment
MPIO32B15/PPM_TX0	mpio32b15_ppm_tx0	L25
TPU_A/TPU_B		
A_TPUCH[0:15]	a_tpuch0	F3
	a_tpuch1	C5
	a_tpuch2	B5
	a_tpuch3	A5
	a_tpuch4	C6
	a_tpuch5	D6
	a_tpuch6	B6
	a_tpuch7	A6
	a_tpuch8	C7
	a_tpuch9	D7
	a_tpuch10	B7
	a_tpuch11	A7
	a_tpuch12	C8
	a_tpuch13	D8
	a_tpuch14	B8
	a_tpuch15	A8