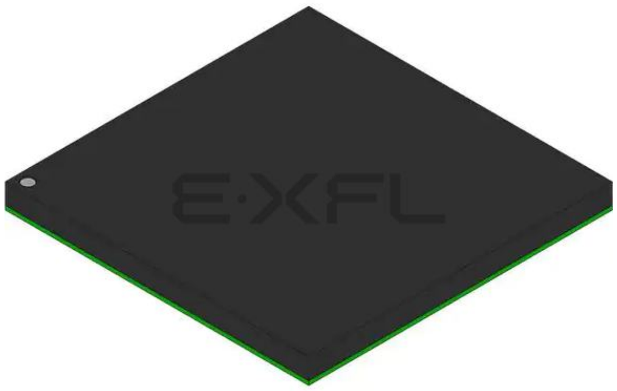

What is "Embedded - Microcontrollers"?

"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	PowerPC
Core Size	32-Bit Single-Core
Speed	40MHz
Connectivity	CANbus, EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	64
Program Memory Size	-
Program Memory Type	ROMless
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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	388-BBGA
Supplier Device Package	388-PBGA (27x27)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mpc561czp40

Table 2-6. PDMCR2 Field Description

Bits	Name	Description
0	PREDIS_EN	Enable for the Pre-discharge circuit to allow 5 volt external devices on the external data bus. 0 Bus pre-discharge disabled 1 Bus pre-discharge enabled **Note:** This bit is reserved in 66-MHz implementations.
1:3	—	Reserved
4:5	TCNC	Controls the function on the following pads: TXD2/QGPO2/C_CNTX0, RXD2/QGPI2/C_CNRX0, MPIO32B11/C_CNRX0, MPIO32B12/C_CNTX0. Refer to Table 2-7.
6	MPI7	Controls the pad MPIO32B7/MPWM5. 0 MPIO32B7 function will be selected. 1 MPWM5 function will be selected.
7	MPI8	Controls the pad MPIO32B8/MPWM20. 0 MPIO32B8 function will be selected. 1 MPWM20 function will be selected.
8	MPI9	Controls the pad MPIO32B9/MPWM21. 0 MPIO32B9 function will be selected. 1 MPWM21 function will be selected.
11:13	PPMPAD	Control the PPM module pads: MPIO32B14/PPM_RX0, MPIO32B15/PPM_TX0, MPWM2/PPM_TX1, MPWM3/PPM_RX1, MPIO32B10/PPM_TSYNC, MPIO32B13/PPM_TCLK. Refer to Table 2-8.
14:15	—	Reserved
16	PPMV	Selects the voltage of the PPM pads. 0 The voltage will be 2.6 V. 1 The voltage will be 5 V.
17:19	—	Reserved
20	MDO6	Selects the functionality of MDO6 0 The pad MPIO32B6/MPWM4/MDO6 will function as MDO6, and the pad MPWM18/MDO6 will function as MPWM18. 1 The pad MPWM18/MDO6 will function as MDO6, and the pad MPIO32B6/MPWM4/MDO6 will function according to MPI6 bit. This selection is enabled only if full port mode is implemented in the READI module, if full port mode is not selected then MPWM18/MDO6 will function as MPWM18, and MPIO32B6/MPWM4/MDO6 will function according to MPI6 bit. NOTE: It is recommended to use MPIO32B6/MPWM4/MDO6 for the Nexus port as MDO6 is enabled from reset.
21	MPI6	Controls the pad MPIO32B6/MPWM4/MDO6. 0 MPIO32B6 function will be selected. 1 MPWM4 function will be selected. This bit will be disabled if full port mode is enabled in the READI module, and MDO6 bit is logic ‘0’.
22:24	—	Reserved
25	PCSV	Selects the polarity of QSMCM module QSPI PCS signals in the PCS expanded mode. 0 Selects Active High. 1 Selects Active Low.

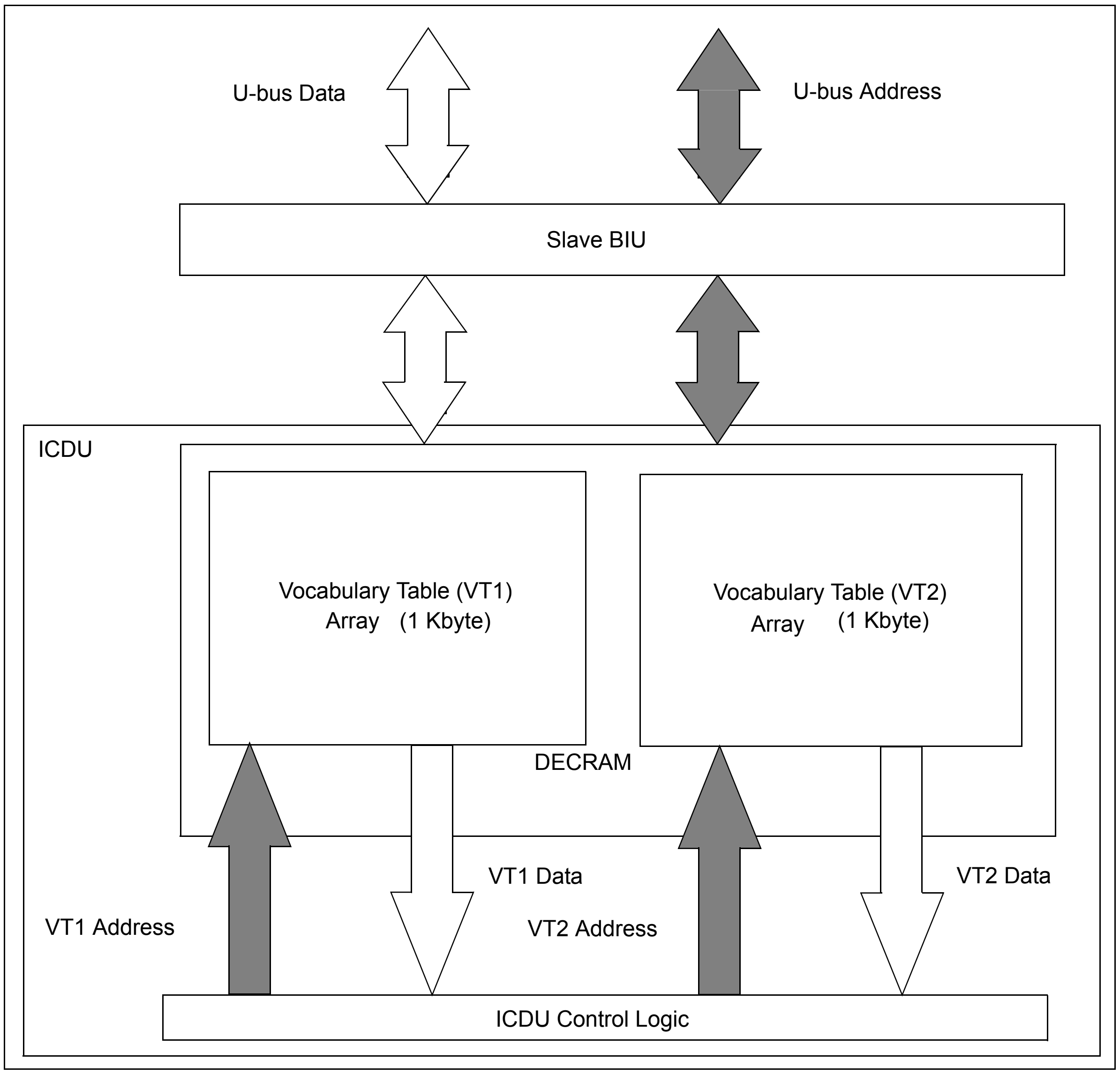

Figure 4-4. DECRAM Interfaces Block Diagram

4.4.1 General-Purpose Memory Operation

In the case of decompression off mode, the DECRAM can serve as a two-clock access general-purpose RAM for U-bus instruction fetches or four-clock access for read/write data operations. The base address of the DECRAM is 0x2F 8000. See Figure 4-6. The proper access rights to the DECRAM array may be defined by programming the R, D, and S bits of the BBCMCR register:

- Read/write or read only
- Instruction/data or data only
- Supervisor/user or supervisor only

Table 13-3. Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw (AN[0])	0, 2, 4, 6, 8, 10, 12, 14
ANx (AN[1])	1, 3, 5, 7, 9, 11, 13, 15
ANy (AN[2])	16, 18, 20, 22, 24, 26, 28, 30
ANz (AN[3])	17, 19, 21, 23, 25, 27, 29, 31

Table 13-4 shows the total number of analog input channels supported with zero to four external multiplexer chips using one QADC module.

Table 13-4. Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels				
No External MUX Chips	One External MUX Chip	Two External MUX Chips	Three External MUX Chips	Four External MUX Chips
16	20	27	34	41

NOTE: QADC64E External MUX Users

If either QADC64E_A or QADC64E_B is in external multiplexing (EMUX) mode then the multiplexer address signal channels, AN[52:54] should *not* be programmed into queues.

13.3 Programming the QADC64E Registers

The QADC64E has three global registers for configuring module operation:

- Module configuration register (Section 13.3.1, "QADC64E Module Configuration Register (QADMCR)")
- Interrupt register (Section 13.3.2, "QADC64E Interrupt Register (QADCINT)"
- Test register (QADCTEST) for factory tests.

The global registers are always defined to be in supervisor-only data space. Refer to Table 13-1 for the QADC64E_A address map and Table 13-2 for the QADC64E_B address map. See Section 13.3.1.4, "Supervisor/Unrestricted Address Space" for access modes of these registers.

The remaining five registers in the control register block control the operation of the queuing mechanism, and provide a means of monitoring the operation of the QADC64E.

- Control register 0 (QACR0) contains hardware configuration information (Section 13.3.5, "Control Register 0 (QACR0)")
- Control register 1 (QACR1) is associated with queue 1 (Section 13.3.6, "Control Register 1 (QACR1)")
- Control register 2 (QACR2) is associated with queue 2 (Section 13.3.7, "Control Register 2 (QACR2)")

Table 13-11. Queue 1 Operating Modes (continued)

MQ1[3:7]	Operating Modes
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
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	External gated single-scan mode (started with SSE1)
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^{1}
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	External gated continuous-scan mode

13.3.7 Control Register 2 (QACR2)

Control register 2 is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit, which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64E, and not changed afterwards.

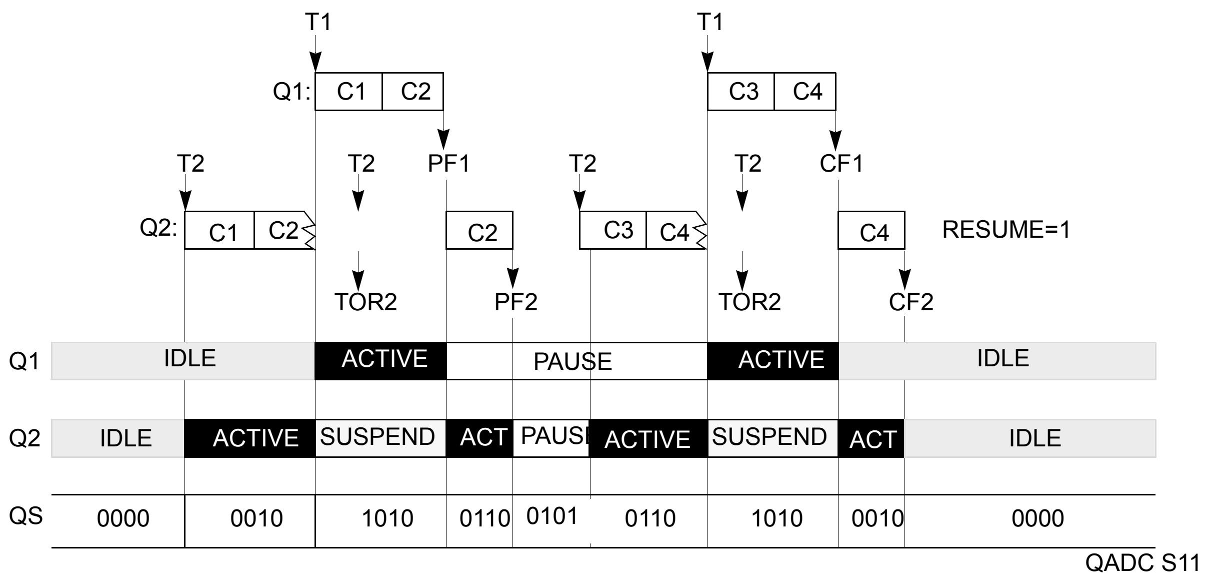

Figure 13-37. CCW Priority Situation 11

The above situations cover normal overlap conditions that arise with asynchronous trigger events on the two queues. An additional conflict to consider is that the freeze condition can arise while the QADC64E is actively executing CCWs. The conventional use for the freeze mode is for software/hardware debugging. When the CPU background debug mode is enabled and a breakpoint occurs, the freeze signal is issued, which can cause peripheral modules to stop operation. When freeze is detected, the QADC64E completes the conversion in progress, unlike queue 1 suspending queue 2. After the freeze condition is removed, the QADC64E continues queue execution with the next CCW in sequence.

Trigger events that occur during freeze are not captured. When a trigger event is pending for queue 2 before freeze begins, that trigger event is remembered when the freeze is passed. Similarly, when freeze occurs while queue 2 is suspended, after freeze, queue 2 resumes execution as soon as queue 1 is finished.

Situations 12 through 19 (Figure 13-38 to Figure 13-45) show examples of all of the freeze situations.

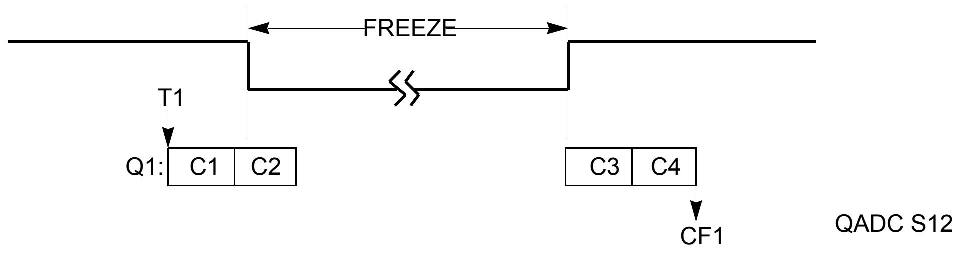

Figure 13-38. CCW Freeze Situation 12

Table 15-31. Effect of Parity Checking on Data Size

M	PE	Result
1	0	9 data bits
1	1	8 data bits, 1 parity bit

15.7.7.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDRx while other data is shifted out. The TE bit in SCCxR1 enables (TE = 1) and disables (TE = 0) the transmitter.

The shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCxR1 determines whether TXD is an open drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function, regardless of whether the pin is used by the SCI or as a general-purpose output pin.

Data to be transmitted is written to SCxDR, then transferred to the serial shifter. Before writing to TDRx, the transmit data register empty (TDRE) flag in SCxSR should be checked. When TDRE = 0, the TDRx contains data that has not been transferred to the shifter. Writing to SCxDR again overwrites the data. If TDRE = 1, then TDRx is empty, and new data may be written to TDRx, clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new data frame is in TDRx (TDRE = 0), then the new data is transferred from TDRx to the transmit serial shifter and TDRE is set automatically. An interrupt may optionally be generated at this point.

The transmission complete (TC) flag in SCxSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCxSR while TC is set, then writing new data to SCxDR, or writing to SCTQ[0:15] for transmit queue operation.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The SBK bit in SCCxR1 is used to insert break frames in a transmission. A non-zero integer number of break frames are transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To ensure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

16.4.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes
   a) Initialize the transmit and receive pin modes in CANCTRL0
   b) Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in CANCTRL1 and CANCTRL2
   c) Select the S-clock rate by programming the PRESDIV register
   d) Select the internal arbitration mode (LBUF bit in CANCTRL1)
2. Initialize message buffers
   a) The control/status word of all message buffers must be written either as an active or inactive message buffer.
   b) All other entries in each message buffer should be initialized as required
3. Initialize mask registers for acceptance mask as required
4. Initialize TouCAN interrupt handler
   a) Initialize the interrupt configuration register (CANICR) with a specific request level
   b) Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt
5. Negate the HALT bit in the module configuration register. At this point, the TouCAN attempts to synchronize with the CAN bus

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

16.4.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. This involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

to be four bytes, regardless of the value programmed in the RGN_SIZE[0:3] field. See Section 22.5.3, “CALRAM Overlay Configuration Register (CRAM_OVLCR)” for details.

The implemented bits of CRAM_RBAx bits are unaffected by reset (hard reset). The diagram below shows one such register, CRAM_RBA0, which provides the base address of overlay region 0.

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RGN_SIZE				—							RBA				
SRESET	Unaffected				0000_000							Unaffected				
Addr	0x38 0008–0x38 0024															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Field	RBA														—	
SRESET	Unaffected														00	

Figure 22-10. CALRAM Region Base Address Register (CRAM_RBAx)

Table 22-5. CRAM_RBAx Bit Descriptions

Bits	Name	Description
0:3	RGN_SIZE	These bits define the size of the overlay region. See Table 22-6 for sizes.
4:10	—	Reserved
11:29	RBA	The region base address defines the starting address of the memory to be overlayed.[1]
30:31	—	Reserved

[1] The overlay match address will include ISB in its comparison. The overlay can only be in the range of the ISB internal space.

Table 22-6. RGN_SIZE Encoding

RGN_SIZE	Number of Overlay Bytes
0000	Overlay block disabled
0001	Overlay block is 4 bytes
0010	Overlay block is 16 bytes
0011	Overlay block is 32 bytes
0100	Overlay block is 64 bytes
0101	Overlay block is 128 bytes
0110	Overlay block is 256 bytes
0111	Overlay block is 512 bytes
1xxx	Reserved

Note: The overlay size of 8 bytes cannot be selected

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, '|' 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.

- Gives an ability to control the execution of the processor and maintain control on it under all circumstances. The development port is able to force the CPU to enter to the debug mode even when external interrupts are disabled.
- It is possible to enter debug mode immediately out of reset thus allowing debugging of a ROM-less system.
- It is possible to selectively define, using an enable register, the events that will cause the machine to enter into the debug mode.
- When in debug mode detect the reason upon which the machine entered debug mode by reading a cause register.
- Entering into the debug mode in all regular cases is restartable in the sense that it is possible to continue to run the regular program from the location where it entered the debug mode.
- When in debug mode all instructions are fetched from the development port but load/store accesses are performed on the real system memory.
- Data Register of the development port is accessed using mtspr and mfspr instructions via special load/store cycles. (This feature together with the last one enables easy memory dump & load).
- Upon entering debug mode, the processor gets into the privileged state (MSR[PR] = 0). This allows execution of any instruction, and access to any storage location.
- An OR signal of all exception cause register (ECR) bits (ECR_OR) enables the development port to detect pending events while already in debug mode. An example is the ability of the development port to detect a debug mode access to a non existing memory space.

Figure 23-6 illustrates the debug mode logic implemented in the CPU.

23.4.5.1 SGPIO6/FRZ/$\overline{\text{PTR}}$ Signal

The SGPIOC6/FRZ/$\overline{\text{PTR}}$ signal powers up as the $\overline{\text{PTR}}$ function and its function is controlled by the GPC bits in the SIUMCR.

23.4.5.2 IWP[0:1]/VFLS[0:1] Signals

The power-up state of IWP[0:1]/VFLS[0:1] is controlled by setting the SIUMCR[DBGC]; see Table 6-8. They can also be set via the reset configuration word (See Section 7.5.2, “Hard Reset Configuration Word (RCW)”). The FRZ state is indicated by the value 0b11 on the VFLS[0:1] signals.

23.4.5.3 VFLS[0:1]/MPIO32B[3:4] Signals

The VFLS[0:1]/MPIO32B[3:4] signals power up as the MPIO32B[3:4] function and their function can be changed via the VFLS bit in the MIOS14TPCR register. The FRZ state is indicated by the value 0b11 on the VFLS[0:1] signals.

23.4.6 Development Port Registers

The development port consists logically of the three registers: development port instruction register (DPIR), development port data register (DPDR), and trap enable control register (TECR). These registers are physically implemented as two registers, development port shift register and trap enable control register. The development port shift register acts as both the DPIR and DPDR depending on the operation being performed. It is also used as a temporary holding register for data to be stored into the TECR. These registers are discussed below in more detail.

23.4.6.1 Development Port Shift Register

The development port shift register is a 35-bit shift register. Instructions and data are shifted into it serially from DSDI using DSCK (or CLKOUT depending on the debug port clock mode, refer to Section 23.4.6.4, “Development Port Serial Communications — Clock Mode Selection”) as the shift clock. These instructions or data are then transferred in parallel to the CPU, the trap enable control register (TECR). When the processor enters debug mode it fetches instructions from the DPIR which causes an access to the development port shift register. These instructions are serially loaded into the shift register from DSDI using DSCK (or CLKOUT) as the shift clock. In a similar way, data is transferred to the CPU by moving it into the shift register which the processor reads as the result of executing a “move from special purpose register DPDR” instruction. Data is also parallel-loaded into the development port shift register from the CPU by executing a “move to special purpose register DPDR” instruction. It is then shifted out serially to DSDO using DSCK (or CLKOUT) as the shift clock.

23.4.6.2 Trap Enable Control Register

The trap enable control register is a 9-bit register that is loaded from the development port shift register. The contents of the control register are used to drive the six trap enable signals, the two breakpoint signals, and the VSYNC signal to the CPU. The “transfer data to trap enable control register” commands will cause the appropriate bits to be transferred to the control register.

Previous Address (A1) = 0x0003 FC01, New Address (A2) = 0x0003 F365

Message Generation:

$$\begin{array}{rl} & \text{A1} = 0000\ 0000\ 0000\ 0011\ 1111\ 1100\ 0000\ 0001 \\ \oplus & \text{A2} = 0000\ 0000\ 0000\ 0011\ 1111\ 0011\ 0110\ 0101 \\ \hline & \text{M1} = 0000\ 0000\ 0000\ 0000\ 0000\ \underbrace{1111\ 0110\ 0100} \end{array}$$

Address Message (M1) = 1111 0110 0100

Address Re-Creation:

A1 $\oplus$ M1 = A2

$$\begin{array}{rl} & \text{A1} = 0000\ 0000\ 0000\ 0011\ 1111\ 1100\ 0000\ 0000 \\ \oplus & \text{M1} = 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0110\ 0100 \\ \hline & \text{A2} = 0000\ 0000\ 0000\ 0011\ 1111\ 0011\ 0110\ 0101 \end{array}$$

Figure 24-33. Relative Address Generation and Re-Creation

24.8.3 Queue Overflow Program Trace Error Message

A trace overrun error occurs when a trace message cannot be queued due to the queue being full, provided program trace is enabled.

The overrun error causes the message queue to be flushed, and an error message to be queued. The error code within the error message indicates that either a program/data/ownership trace overrun error has occurred or that only a program trace overrun has occurred. The next BTM will be a synchronization message. Refer to Table 24-20.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b0 0000, 0b0 0001, 0b0 0010, 0b0 0111)

Length = 11 bits

Figure 24-34. Error Message (Queue Overflow) Format

24.8.4 Branch Trace Message Operation

24.8.4.1 BTM Capture and Encoding Algorithm

BTM is accomplished by capturing instruction fetch information from the U-bus and instruction execution information from the RCPU (VF and VFLS signals), and combining them to generate program trace messages.

CONTROL BITS

Figure D-8. UART Receiver Parameters

D.7 New Input Capture/Transition Counter (NITC)

NITC allows, for a specified number of transitions, a TPU3 channel to: capture the value of a TCR (test configuration register) or any specified location in parameter RAM and then generate an interrupt request to notify the bus master (times of the two most recent transitions remain in parameter RAM), capture input continually or detect a specific number of transitions and end channel activity until reinitialization, or generate a link to other channels after the transitions have taken place. See Freescale TPU Progamming Note *New Input Capture/Input Transition Counter TPU Function (NITC), (TPUPN08/D).*

Figure D-9 shows all of the host interface areas for the NITC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0 1 2 3	Channel Function Select	xxxx – FQD Function Number. Assigned during microcode assembly. See Table D-1	0x30YY0C – 0x30YY12
0 1	Host Sequence	00 – Primary Channel (Normal Mode) 01 – Secondary Channel (Normal Mode) 10 – Primary Channel (Fast Mode) 11 – Secondary Channel (Fast Mode)	0x30YY14 – 0x30YY16
0 1	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Read TCR1 11 – Initialize	0x30YY18 – 0x30YY1A
0 1	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0 ■	Channel Interrupt Enable	x – Not Used	0x30YY0A
0 ■	Channel Interrupt Status	x – Not Used	0x30YY20

PARAMETER RAM

ADDRESS OFFSETS	BITS 0–15	
0x30XXW0	■	Param 0
0x30XXW2	■	Param 1
0x30XXW4	TCR1_VALUE	Param 2
0x30XXW6	CHAN_PINSTATE	Param 3
0x30XXW8	CORR_PINSTATE_ADDR	Param 4
0x30XXWA	EDGE_TIME_LSB_ADDR	Param 5
0x30XXWC	■	Param 6
0x30XXWE	■	Param 7

▣ = Written By RCPU	░ = Written by RCPU and TPU	W = Channel Number
□ = Written By TPU	■ = Unused Parameters	For address offsets: XX=41 for TPU_A, 45 for TPU_B YY=40 for TPU_A, 44 for TPU_B

Figure D-23. FQD Parameters — Secondary Channel

D.19 Read/Write Timers and Pin TPU3 Function (RWTPIN)

The RWTPIN Bank 1 TPU3 function enables the RCPU to read, via locations in PRAM, both the TCR1 and TCR2 timer counters, and than selectively load TCR1 or TCR2 with a RCPU-supplied value contained in PRAM. The function also allows control of the pin state and direction of the RWTPIN channel.

A pin-state parameter is maintained in PRAM and is updated upon every service request. It can contain a value of the current pin state whether the pin is programmed as an input or output.

The function also receives links. Upon receipt, it will read the two TCRs into PRAM, updating the pin-state parameter and generating a maskable interrupt request to the RCPU.

The RCPU can control the channel pin, the TCRs, or both. To control the channel pin only, the 'read TCR' option is used and the values returned ignored. Because this function controls the TCRs without affecting the channel pin, it can run on a TPU3 channel whose pin is controlled by a function running on another channel (for example, a slave stepper-motor channel). See Freescale TPU Progamming Note *Using The TPU Function Library And TPU Emulation Mode, (TPUPN00/D).*

Figure D-30 shows all of the host interface areas for the PTA function.

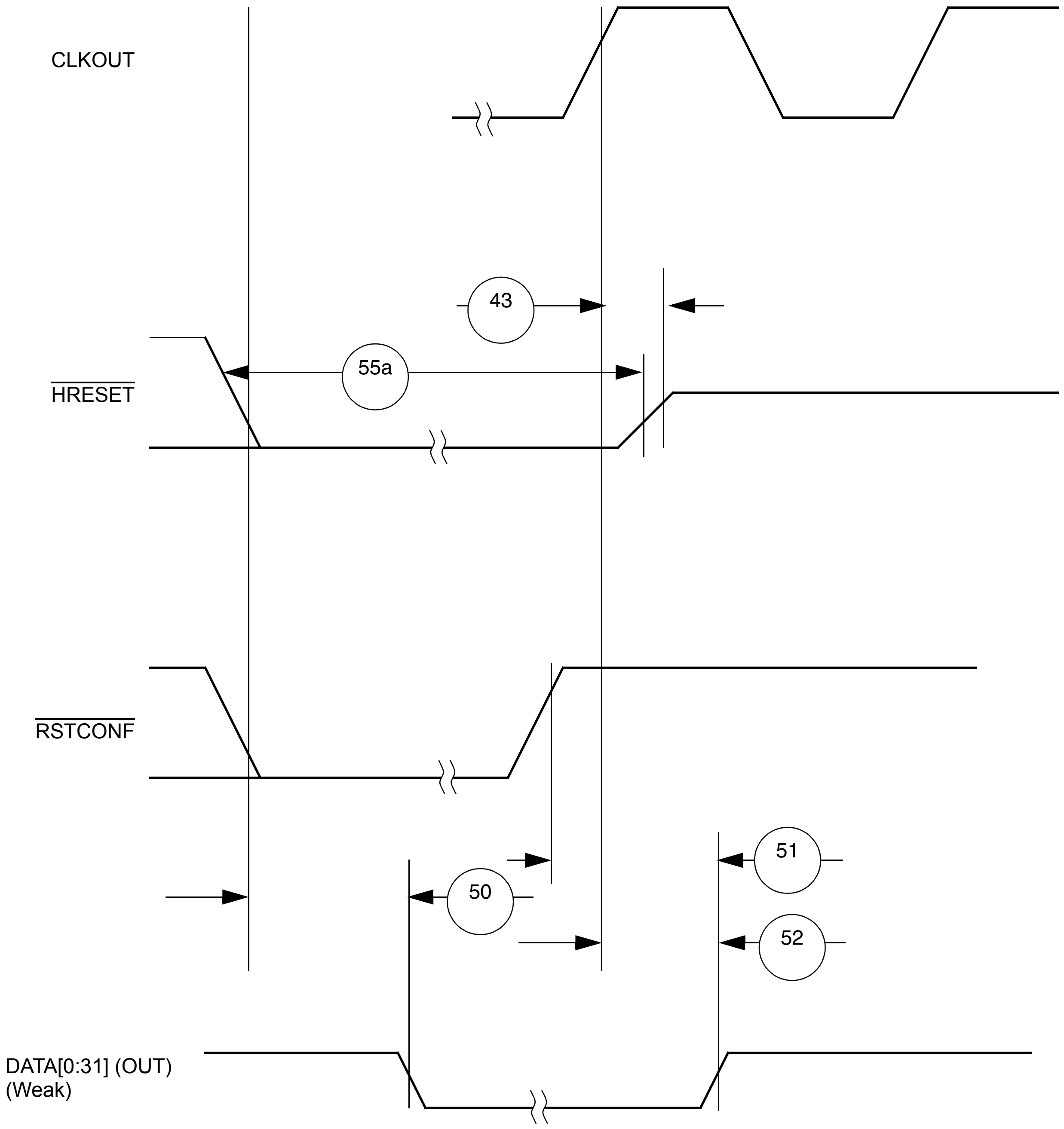

Figure F-35. Reset Timing – Data Bus Weak Drive During Configuration

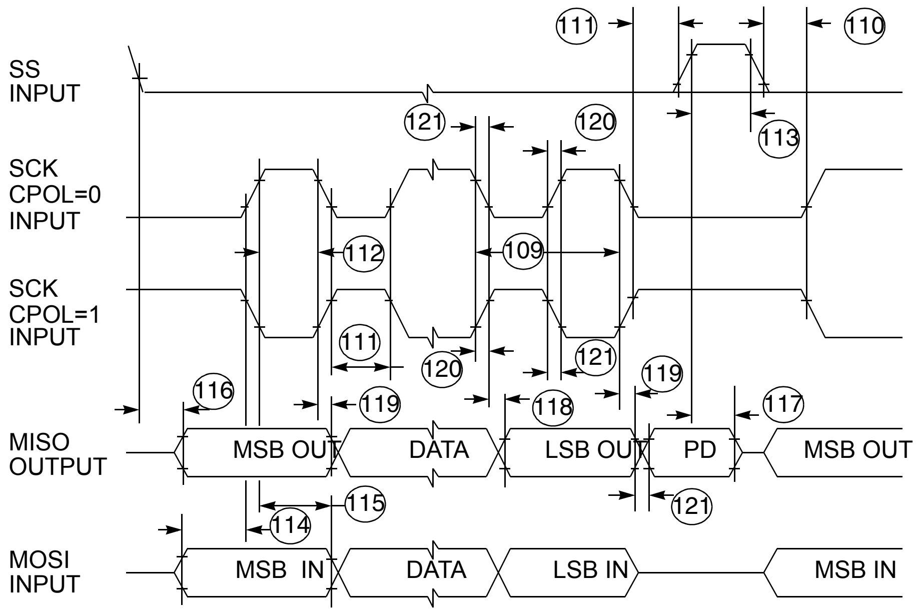

Figure F-42. QSPI Timing – Slave, CPHA = 0

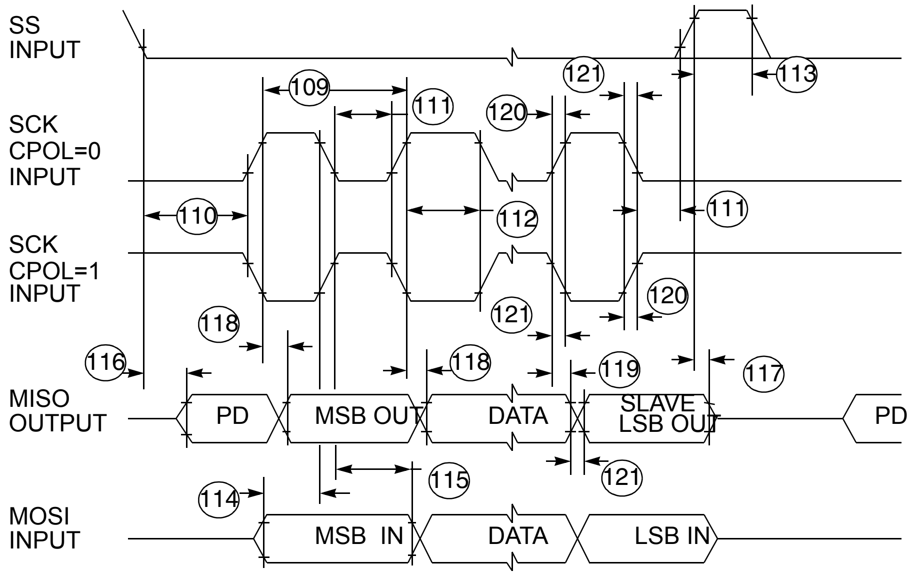

Figure F-43. QSPI Timing – Slave, CPHA = 1

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

Signal Name	Pin Name	Ball Assignment
MDA[11:15]	mda11	C20
	mda12	D20
	mda13	A21
	mda14	B21
	mda15	C21
MDA[27:31]	mda27	D21
	mda28	A22
	mda29	B22
	mda30	F24
	mda31	F25
MPWM[0:1]/MDI[1:2]	mpwm0_mdi1	F26
	mpwm1_mdo2	G23
MPWM2/PPM_TX1	mpwm2_ppm_tx1	G26
MPWM3/PPM_RX1	mpwm3_ppm_rx1	G25
MPWM16	mpwm16	G24
MPWM17/MDO3	mpwm17_mdo3	H23
MPWM[18:19]/MDO[6:7]	mpwm18_mdo6	H24
	mpwm19_mdo7	H25
VF0/MPIO32B0/MDO1	vf0_mpio32b0_mdo1	L23
VF1/MPIO32B1/MCKO	vf1_mpio32b1_mcko	L24
VF2/MPIO32B2/$\overline{\text{MSEI}}$	vf2_mpio32b2_msei_b	M24
VFLS0/MPIO32B3/$\overline{\text{MSEO}}$	vfls0_mpio32b3_mseo_b	M25
VFLS1/MPIO32B4	vfls1_mpio32b4	M26
MPIO32B5/MDO5	mpio32b5_mdo5	H26
MPIO32B6/MPWM4/MDO6	mpio32b6_mpwm4_mdo6	J23
MPIO32B7/MPWM5	mpio32b7_mpwm5	J24
MPIO32B[8:9]/MPWM[20:21]	mpio32b8_mpwm20	J25
	mpio32b9_mpwm21	J26
MPIO32B10/PPM_TSYNC	mpio32b10_ppm_tsync	K25
MPIO32B11/C_CNRX0	mpio32b11_c_cnrx0	K24
MPIO32B12/C_CNTX0	mpio32b12_c_cntx0	K23
MPIO32B13/PPM_TCLK	mpio32b13_ppm_tclk	K26
MPIO32B14/PPM_RX0	mpio32b14_ppm_rx0	L26

VSS	VSS	VSS	VSS	VSS	VSS
VSS	VSS	VSS	VSS	VSS	VSS
VSS	VSS	VSS	VSS	VSS	VSS
VSS	VSS	VSS	VSS	VSS	VSS
VSS	VSS	VSS	VSS	VSS	VSS
VSS	VSS	VSS	VSS	VSS	VSS

	1	2	3	4	5	6	7	8	9	10	11	12	13
P	IRQ4_B_AT2_SGPIOC4	IRQ2_B_CR_B_SGPIOC2_MDO5_MTS	IRQ0_B_SGPIOC0_MDO4	IRQ1_B_RSV_B_SGPIOC1									
R	SGPIOC7_IRQOUT_B_LWP0	BB_B_VF2_IWP3	BG_B_VF0_LWP1	BR_B_VF1_IWP2									
T	WE_B_AT0	WE_B_AT1	WE_B_AT2	WE_B_AT3									
U	CS0_B	CS1_B	CS2_B	CS3_B									
V	RD_WR_B	OE_B	TEA_B	TSIZ0									
W	TSIZ1	TS_B	TA_B	BDIP_B									
Y	BURST_B	BI_B_STS_B	ADDR_SGPIOA12	ADDR_SGPIOA11									
AA	VSS	VSS	VSS	QVDDL									
AB	VSS	VSS	QVDDL	VSS									
AC	VSS	QVDDL	VSS	NVDDL	VSS	ADDR_SGPIOA10	ADDR_SGPIOA18	ADDR_SGPIOA20	ADDR_SGPIOA23	NVDDL	ADDR_SGPIOA26	DATA_SGPIOD1	DATA_SGPIOD5
AD	QVDDL	VSS	NVDDL	VSS	VSS	QVDDL	ADDR_SGPIOA13	ADDR_SGPIOA16	ADDR_SGPIOA19	ADDR_SGPIOA21	ADDR_SGPIOA24	ADDR_SGPIOA25	DATA_SGPIOD0
AE	VSS	NVDDL	VSS	VSS	VSS	QVDDL	ADDR_SGPIOA14	ADDR_SGPIOA17	ADDR_SGPIOA31	ADDR_SGPIOA30	ADDR_SGPIOA28	ADDR_SGPIOA29	DATA_SGPIOD30
AF	NVDDL	VSS	VSS	VSS	VDDH	VSS	ADDR_SGPIOA15	ADDR_SGPIOA9	ADDR_SGPIOA8	ADDR_SGPIOA22	ADDR_SGPIOA27	DATA_SGPIOD31	DATA_SGPIOD3

Figure G-67. MPC561/MPC563 Ball Map (Black and White, page 2)