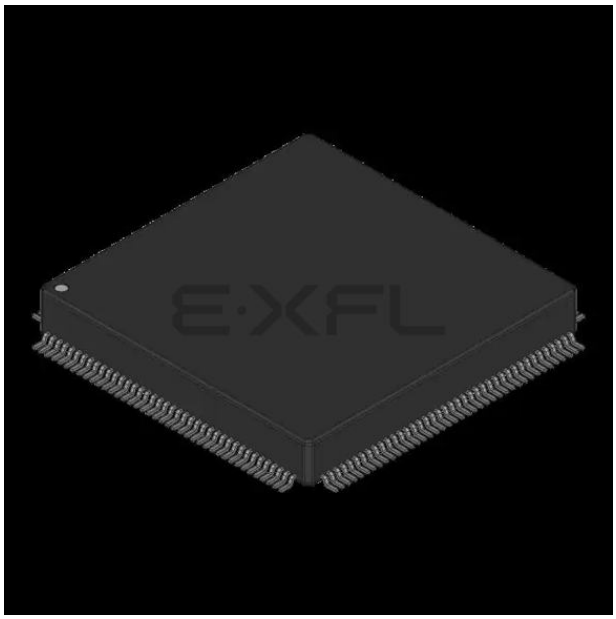


Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Active
Core Processor	Coldfire V2
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	33MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	-
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	160-BQFP
Supplier Device Package	160-QFP (28x28)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mcf5206ft33a

1.3.11 System Protection

The MCF5206 processor contains a 16-bit software watchdog timer with an 8-bit prescaler. The programmable software watchdog timer provides either a level 7 interrupt or a hardware reset on timeout. The MCF5206 processor also contains a reset status register that indicates the cause of the last reset.

1.3.12 JTAG

To help with system diagnostics and manufacturing testing, the MCF5206 processor includes dedicated user-accessible test logic that complies with the IEEE 1149.1 standard for boundary scan testability, often referred to as Joint Test Action Group, or JTAG. For more information, refer to the IEEE 1149.1 standard.

1.3.13 System Debug Interface

The ColdFire processor core debug interface supports real-time trace and Background-Debug Mode. A four-pin Background Debug Mode (BDM) interface provides system debug. The BDM is a proper subset of the BDM interface provided on Motorola's 683XX Family of parts.

In real-time trace, four status lines provide information on processor activity in real time (PST pins). A 4-bit wide debug data bus (DDATA) displays operand data, which helps track the machine's dynamic execution path as the change-of-flow instructions execute. These signals are multiplexed with an 8-bit parallel port for application development, which does not use real-time trace.

1.3.14 Pinout and Package

The MCF5206 device is supplied in a 160-pin plastic quad flat pack package.

Table 2-3. Byte Write-Enable Signals

TRANSFER SIZE	PORT SIZE	BURST	SIZ1	SIZ0	A1	A0	$\overline{WE0}$ D31-D24	$\overline{WE1}$ D23-D16	$\overline{WE2}$ D15-D8	$\overline{WE3}$ D7-D0
BYTE	8- bit	0	0	1	0	0	0	1	1	1
					0	1	0	1	1	1
					1	0	0	1	1	1
					1	1	0	1	1	1
		1	0	1	0	0	0	1	1	1
					0	1	0	1	1	1
					1	0	0	1	1	1
					1	1	0	1	1	1
	16- bit	0	0	1	0	0	0	1	1	1
					0	1	1	0	1	1
					1	0	0	1	1	1
					1	1	1	0	1	1
		1	0	1	0	0	0	1	1	1
					0	1	1	0	1	1
					1	0	0	1	1	1
					1	1	1	0	1	1
	32- bit	0	0	1	0	0	0	1	1	1
					0	1	1	0	1	1
					1	0	1	1	0	1
					1	1	1	1	1	0
		1	0	1	0	0	0	1	1	1
					0	1	1	0	1	1
					1	0	1	1	0	1
					1	1	1	1	1	0
WORD	8- bit	0	0	1	0	0	0	1	1	1
					0	1	0	1	1	1
					1	0	0	1	1	1
					1	1	0	1	1	1
		1	1	0	0	0	0	1	1	1
					0	1	0	1	1	1
					1	0	0	1	1	1
					1	1	0	1	1	1
	16- bit	0	1	0	0	0	0	0	1	1
					1	0	0	0	1	1
		1	1	0	0	0	0	0	1	1
					1	0	0	0	1	1
	32 bit	0	1	0	0	0	0	0	1	1
					1	0	1	1	0	0
		1	1	0	0	0	0	0	1	1
					1	0	1	1	0	0

2.5.3 Transfer Type (TT[1:0])

These three-state output signals indicate the type of access for the current bus cycle. TT[1:0] are not sampled by the MCF5206 during alternate master transfers. Table 2-7 lists the definitions of the TT[1:0] encodings.

Table 2-7. Bus Cycle Transfer Type Encoding

TT[1:0]	TRANSFER TYPE
0 0	Normal Access
0 1	Reserved
1 0	Emulator Access
1 1	CPU Space or Interrupt Acknowledge

2.5.4 Access Type and Mode (ATM)

This three-state output signal provides supplemental information for each transfer cycle type. ATM is not sampled by the MCF5206 during alternate master transfers. Table 2-8 lists the encoding for normal, debug and CPU space/interrupt-acknowledge transfer types.

Table 2-8. ATM Encoding

TRANSFER TYPE	INTERNAL TRANSFER MODIFIER	ATM (TS=0)	ATM (TS=1)
00 (Normal Access)	Supervisor Code	1	1
	Supervisor Data	0	1
	User Code	1	0
	User Data	0	0
10 (Debug Access)	Supervisor Code	1	1
	Supervisor Data	0	1
11 (CPU Space/ Acknowledge Access)	CPU Space - MOVEC Instruction	0	0
	Interrupt Acknowledge - level 7	1	0
	Interrupt Acknowledge - level 6	1	0
	Interrupt Acknowledge - level 5	1	0
	Interrupt Acknowledge - level 4	1	0
	Interrupt Acknowledge - level 3	1	0
	Interrupt Acknowledge - level 2	1	0
	Interrupt Acknowledge - level 1	1	0

2.5.5 Transfer Start ($\overline{TS}$)

The MCF5206 asserts this three-state bidirectional active-low signal for one clock period to indicate the start of each bus cycle. During alternate master accesses, the MCF5206 monitors transfer start ($\overline{TS}$) to detect the start of each alternate master bus cycle to determine if chip select or DRAM control signals need to be asserted.

2.5.6 Transfer Acknowledge ($\overline{TA}$)

This three-state bidirectional active-low synchronous signal indicates the completion of a requested data transfer operation. During transfers initiated by the MCF5206, transfer acknowledge ($\overline{TA}$) is an input signal from the referenced slave device indicating completion of the transfer.

$\overline{TA}$ is not used for termination during DRAM accesses initiated by the MCF5206.

When an alternate master is controlling the bus, $\overline{TA}$ may be driven as an output by the MCF5206 or may be driven by the referenced slave device to indicate the completion of the requested data transfer. If the alternate master requested transfer is to a chip select or default memory, the assertion of $\overline{TA}$ is controlled by the number of wait states and the setting of the Alternate Master Automatic Acknowledge (EMAA) bit in the Chip Select Control Registers (CSCRs) or the Default Memory Control Register (DMCR). If the alternate master requested transfer is a DRAM access, $\overline{TA}$ is driven by the MCF5206 as an output and asserted at the completion of the transfer.

2.5.7 Asynchronous Transfer Acknowledge ($\overline{ATA}$)

This active-low asynchronous input signal indicates the completion of a requested data transfer operation. Asynchronous transfer acknowledge ($\overline{ATA}$) is an input signal from the referenced slave device indicating completion of the transfer. $\overline{ATA}$ is synchronized internal to the MCF5206.

NOTE

The internal synchronized version of asynchronous transfer acknowledge ($\overline{ATA}$) is referred to as “internal asynchronous transfer acknowledge ($\overline{ATA}$).” Because of the time required to internally synchronize $\overline{ATA}$ during a read cycle, data is latched on the rising edge of CLK when the internal $\overline{ATA}$ is asserted. Consequently, data must remain valid for at least one clock cycle after the assertion of $\overline{ATA}$. Similarly, during a write cycle, data is driven until the rising edge of CLK when the internal $\overline{ATA}$ is asserted.

$\overline{ATA}$ must be driven for one full clock to ensure that the MCF5206 properly synchronizes the signal. $\overline{ATA}$ is not used for termination during DRAM accesses.

2.5.8 Transfer Error Acknowledge ($\overline{TEA}$)

This active-low input signal is asserted by the external slave to indicate an error condition for the current transfer. The assertion of transfer error acknowledge ($\overline{TEA}$) causes the MCF5206 to immediately abort the bus cycle. The assertion of $\overline{TEA}$ has precedence over the assertion of $\overline{ATA}$ and $\overline{TA}$.

Table 3-2. Format Field Encodings

ORIGINAL A7 @ TIME OF EXCEPTION, BITS 1:0	A7 @ 1ST INSTRUCTION OF HANDLER	FORMAT FIELD
00	Original A7 - 8	4
01	Original A7 - 9	5
10	Original A7 - 10	6
11	Original A7 - 11	7

- A 4-bit fault status field, FS[3:0], at the top of the system stack. This field is defined for access and address errors only and written as zeros for all other types of exceptions. See Table 3-3.

Table 3-3. Fault Status Encodings

FS[3:0]	DEFINITION
00xx	Reserved
0100	Error on instruction fetch
0101	Reserved
011x	Reserved
1000	Error on operand write
1001	Attempted write to write-protected space
101x	Reserved
1100	Error on operand read
1101	Reserved
111x	Reserved

- The 8-bit vector number, vector[7:0], defines the exception type and is calculated by the processor for all internal faults and represents the value supplied by the peripheral in the case of an interrupt. Refer to Table 3-1.

3.5 PROCESSOR EXCEPTIONS

3.5.1 Access Error Exception

An Access Error Exception vector number $2 occurs when a bus cycle terminates with an error condition. The exact processor response to an access error depends on the type of memory reference being performed.

For an instruction fetch, the processor postpones the error reporting until the faulted reference is needed by an instruction for execution. Therefore, faults that occur during instruction prefetches that are then followed by a change of instruction flow does not generate an exception. When the processor attempts to execute an instruction with a faulted opword and/or extension words, the access error is signaled and the instruction aborted. For this type of exception, the programming model has not been altered by the instruction generating the access error.

Figure 6-6 is a flowchart for write transfers to 8-, 16-, or 32-bit ports. Bus operations are similar for each case and vary only with the size indicated, the portion of the data bus used for the transfer and the specific number of cycles needed for each transfer.

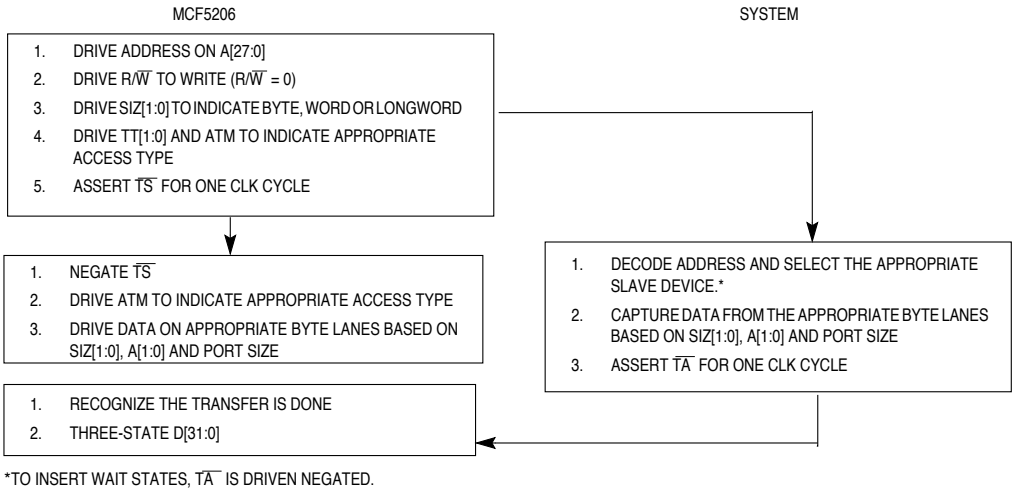


Figure 6-6. Byte-, Word-, and Longword-Write Transfer Flowchart

Figure 6-17 shows a user code byte read from an 8-bit port.

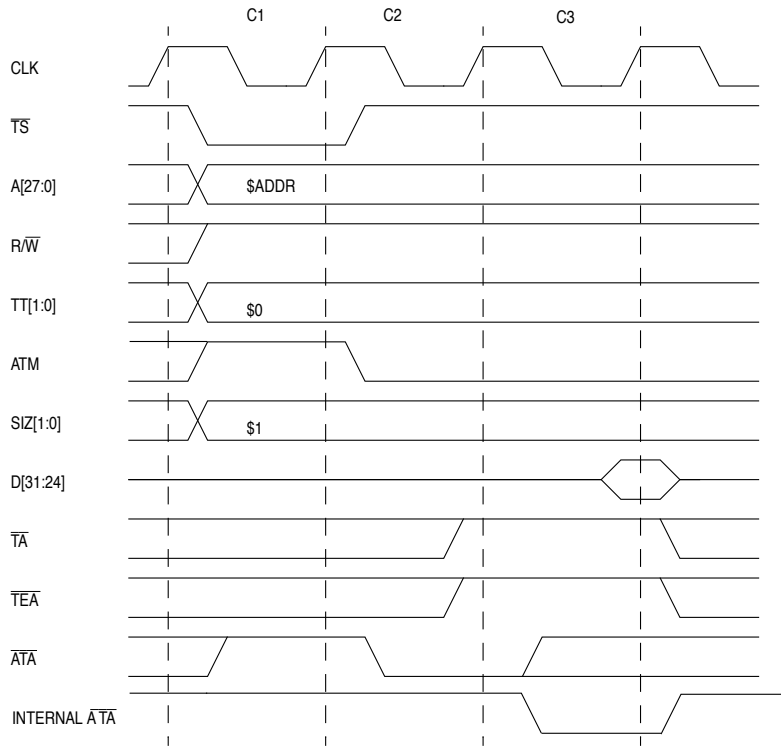


Figure 6-17. Byte-Read Transfer from an 8-Bit Port Using Asynchronous Termination (One Wait State)

Clock 1 (C1)

The read cycle starts in C1. During C1, the MCF5206 places valid values on the address bus (A[27:0]) and transfer control signals. The transfer type (TT[1:0]) signals identify the specific access type and ATM identifies the transfer as code. The read/write (R/$\overline{W}$) signal is driven high for a read cycle, and the size signals (SIZ[1:0]) are driven to $1 to indicate a byte transfer. The MCF5206 asserts $\overline{TS}$ to indicate the beginning of a bus cycle.

Clock 2 (C2)

During C2, the MCF5206 negates $\overline{TS}$ and drives ATM low to identify the transfer as user. The selected device(s) asserts $\overline{ATA}$.

Clock 3 (C3)

At the end of C3, the MCF5206 samples the level of internal asynchronous transfer acknowledge and if it is asserted, latches the current value of D[31:24]. If internal

asynchronous transfer acknowledge is asserted, the byte transfer is complete and the transfer terminates. If internal asynchronous transfer acknowledge is negated, the MCF5206 continues to sample internal asynchronous transfer acknowledge and inserts wait states instead of terminating the transfer. The MCF5206 continues to sample internal asynchronous transfer acknowledge until it is asserted. As long as $\overline{ATA}$ is asserted by the falling edge of C2, internal asynchronous transfer acknowledgethe is asserted by the rising edge of C3.

6.5.7 Asynchronous Acknowledge Write Transfer

Figure 6-18 is a flowchart for write transfers to 8-, 16-, or 32-bit ports with asynchronous termination. Bus operations are similar for each case and vary only with the size indicated, the portion of the data bus used for the transfer, and the specific number of cycles needed for each transfer.

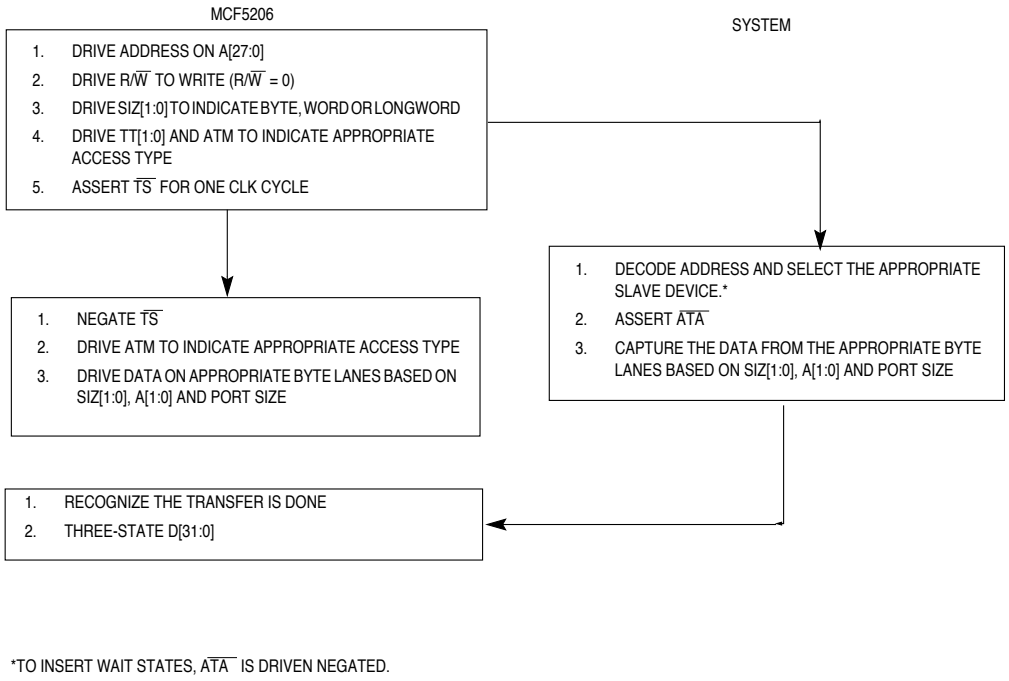


Figure 6-18. Byte-, Word-, and Longword-Write Transfer with Asynchronous Termination Flowchart

indicate a level 1 interrupt, EINT6 is cleared and EINT1 is set, indicating that only an external interrupt level 1 is currently pending. Running an interrupt-acknowledge cycle for the level 6 interrupt returns the spurious interrupt vector as the level 6 interrupt is no longer pending. EINT1 remains pending until the interrupt priority level signals change. For timing diagrams, refer to the electrical section.

You can assign as many as four interrupts to the same interrupt level, but you must assign unique interrupt priorities. The interrupt controller uses the interrupt priorities during an interrupt acknowledge cycle to determine which interrupt is being acknowledged. The interrupt priority bits determine the appropriate interrupt being acknowledged when multiple interrupts are assigned to the same level and are pending when the interrupt-acknowledge cycle is generated.

NOTE

You should not program interrupts to have the same level and priority. Interrupts can have the same level but different priorities. All level and priority combinations must be unique.

If an external interrupt request is being acknowledged and the AVEC bit in the corresponding ICR is not set, an external interrupt acknowledge cycle occurs. During an external interrupt acknowledge cycle, TT[1:0] and A[27:5] are driven high; A[4:2] are set to the interrupt level being acknowledged; and A[1:0] are driven low. Additionally, ATM is asserted when $\overline{TS}$ is asserted and ATM is negated when $\overline{TS}$ is negated. For nonautovector responses, the external device places the vector number on D[31:24]. For autovector responses, the autovector is generated internally and no external interrupt acknowledge cycle is run.

An interrupt request from the SWT does not require an external interrupt acknowledge cycle because SWIVR stores its interrupt vector number.

If an internal peripheral interrupt source is being acknowledged and the AVEC bit in the corresponding ICR is cleared, an internal interrupt-acknowledge cycle occurs with the internal peripheral supplying the interrupt vector number. If the corresponding AVEC bit is set, an internal interrupt-acknowledge cycle is run but the SIM internally generates an autovector. No external interrupt acknowledge cycle is run.

7.3 PROGRAMMING MODEL

7.3.1 SIM Registers Memory Map

Table 7-1 shows the memory map of all the SIM registers. The internal registers in the SIM are memory-mapped registers offset from the MBAR address pointer. The following list addresses several key notes regarding the programming model table:

- The Module Base Address Register can only be accessed in supervisor mode using the MOVEC instruction with an Rc value of $C0F.
- Underlined registers are status or event registers.

The SWIVR is an 8-bit write-only register, which is set to the uninitialized vector $0F at system reset.

Software Watchdog Interrupt Vector Register(SWIVR) Address MBAR + $42

7	6	5	4	3	2	1	0
SWIV7	SWIV6	SWIV5	SWIV4	SWIV3	SWIV2	SWIV1	SWIV0
RESET: 0	0	0	0	1	1	1	1

7.3.2.9 SOFTWARE WATCHDOG SERVICE REGISTER (SWSR). The SWSR is the location to which the SWT servicing sequence is written. To prevent an SWT timeout, you should write a $55 followed by a $AA to this register. Although both writes must occur in the order listed prior to the SWT timeout, any number of instructions or accesses to the SWSR can be executed between the two writes. This allows interrupts and exceptions to occur, if necessary, between the two writes.

The SWSR is an 8-bit write-only register. At system reset, the contents of SWSR are uninitialized.

Software Watchdog Service Register(SWSR) Address MBAR + $43

7	6	5	4	3	2	1	0
SWSR7	SWSR6	SWSR5	SWSR4	SWSR3	SWSR2	SWSR1	SWSR0
RESET: -	-	-	-	-	-	-	-

Write Only

7.3.2.10 PIN ASSIGNMENT REGISTER (PAR). The PAR lets you select certain signal pin assignments. You can select between address, chip select and write enables, data and parallel port signals, processor status (PST) and parallel port signals, and UART request-to-send and reset out.

The PAR is an 8-bit read-write register. At system reset, all bits are cleared.

Pin Assignment Register(PAR) Address MBAR + $CB

7	6	5	4	3	2	1	0
PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0
RESET: 0	0	0	0	0	0	0	0

R/W

PAR7 - Pin Assignment Bit 7

This bit lets you select the signal output on the $\overline{\text{RTS2}}/\overline{\text{RSTO}}$ pin as follows:

0 = Output reset out ($\overline{\text{RSTO}}$) signal on $\overline{\text{RTS2}}/\overline{\text{RSTO}}$ pin
1 = Output UART 2 request to send signal on $\overline{\text{RTS2}}/\overline{\text{RSTO}}$ pin

(CSMRs) and DRAM Controller Mask Register (DCMRs), looking for a match.

The priority is listed in Table 10-4 (from highest priority to lowest priority): Address Registers (CSARs), DRAM

Table 10-4. Chip Select, DRAM and Default Memory Address Decoding Priority

Chip select 0	Highest
Chip select 1	
Chip select 2	
Chip select 3	
Chip select 4	
Chip select 5	
Chip select 6	
Chip select 7	
Dram Bank 0	
Dram Bank 1	
Default Memory	Lowest

The MCF5206 compares the address and mask in Chip select 0 - 7 (Chip select 0 is compared first), then the address and mask in DRAM 0 - 1. If the address does not match in either or these, the MCF5206 uses the control bits in the Default Memory Control Register (DMCR) to control the bus transfer. If the Default Memory Control Register (DMCR) control bits are used, no chip select or DRAM control signals are asserted during the transfer.

10.3.2.2 ACCESS PERMISSION. DRAM bank accesses can be restricted based on transfer direction and attributes. Each DRAM bank can be enabled for read and/or write transfers using the WR and RD bits in the DCCRs. Each DRAM bank can have supervisor data, supervisor code, user data, and user code transfers masked from their address space using the SD, SC, UD, and UC bits in the DCMRs. The transfer address must match, the transfer direction must be enabled, and transfer attributes must be unmasked for a transfer to a DRAM bank to occur.

For example, if the DCARs, DCMRs, and DCCRs are programmed as shown in Table 10-5, DRAM bank 0 would start at address $04000000, and be 16 MByte, read/write, and available for supervisor transfers only. DRAM bank 1 would start at address $05000000 and be 1Mbyte, read-only, and available to all address spaces.

If a user data -read transfer was attempted to address $04000000, the transfer would not access DRAM bank 0, since user space transfers are masked. The transfer would not access DRAM bank 1 since the addresses do not match. Therefore, a Default Memory transfer would occur.

If a user data write transfer was attempted to address $05000000, the transfer would not access DRAM bank 0 since the addresses do not match. The transfer would not access

possible nonburst transfer in burst page mode requires 5 clocks. You can program the DCTR to generate slower burst-page-mode transfers.

Figure 10-16 illustrates the timing of a word read transfer to an 8-bit DRAM in burst page mode. In burst page mode after the first byte transfer of the burst is complete, $\overline{RAS}$ remains asserted while $\overline{CAS}$[0] and $\overline{TA}$ are negated and the column address of the second byte transfer of the burst is driven. After the $\overline{CAS}$ precharge time is met, $\overline{CAS}$[0] asserts for the second byte read transfer. When the second byte read transfer is completed, $\overline{RAS}$, $\overline{CAS}$[0], and $\overline{TA}$ are negated, ending the burst transfer.

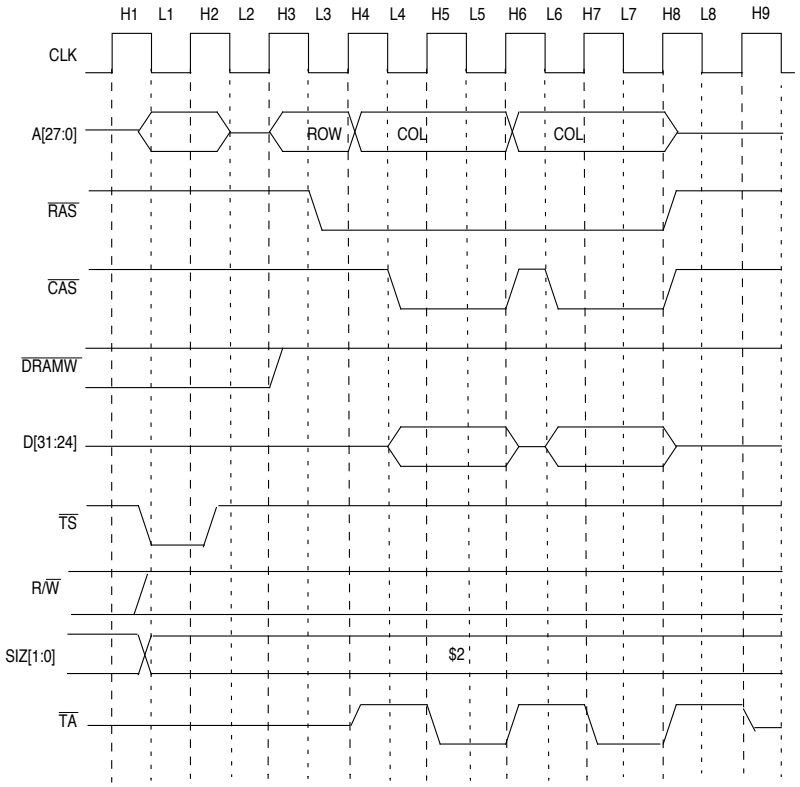


Figure 10-16. External Master Word Read Transfer in Burst Page Mode with 8-Bit DRAM

Clock H1/L1

An external master is the current bus master. The external master starts a DRAM burst word-write transfer by driving A[27:0], driving R/$\overline{W}$ high indicating a read transfer, driving

SIZ[1:0] to $2 indicating a word transfer, and asserting TS. These inputs to the MCF5206 must be set up with respect to the rising edge of CLK H2.

Clock H2

On the rising edge of CLK when TS is asserted, the MCF5206 registers the address and attribute signals. It internally decodes these signals and determine if the external master transfer is a DRAM access. The external master negates TS and must three-state A[27:0] after the rising edge of CLK H2, if the internal address multiplexing is to be used.

Clock H3

The MCF5206 has determined that the external master transfer is a DRAM access, so the MCF5206 drives A[27:0] with the same value as was registered on the rising edge of H2. A[27:9] is the DRAM row address. The MCF5206 also drives DRAMW high indicating a DRAM read cycle.

Clock L3

The MCF5206 asserts RAS to indicate the row address is valid on A[27:9].

Clock H4

The MCF5206 internally multiplexes the address and drives out the column address on A[27:9]. The MCF5206 also actively drives TA negated.

Clock L4

The MCF5206 asserts CAS[0] to indicate the column address is valid on A[27:9]. At this point the DRAM turns on it's output drivers and begin driving the data on D[31:24].

Clock H5

The MCF5206 asserts the TA signal to indicate that the first byte read transfer of the burst is completed and the read data is valid on D[31:24] on the next rising edge of CLK.

Clock H6

The MCF5206 negates CAS[0], and TA, ending the first byte read transfer of the burst. Because the bank is in burst page mode, the MCF5206 continues to assert RAS. The negation of CAS[0] starts the CAS precharge. The MCF5206 drives the column address on A[27:9] for second byte read transfer of the burst.

Clock L6

After the CAS precharge time is met, the MCF5206 asserts CAS[0] to indicate the column address is valid on A[27:9]. At this point the DRAM turns on its output drivers and begins driving the data bus.

control the $\overline{CAS}$ assertion and negation time during fast page mode and burst page mode transfers. Refer to Figure 10-21 for a timing diagram of EDO DRAM page mode transfers.

0 = DRAM banks are populated with standard DRAM, do not use EDO $\overline{CAS}$ timing
1 = DRAM banks are populated with EDO DRAM, use EDO $\overline{CAS}$ timing

NOTE

If neither fast page mode or burst page mode are enabled in the DRAM Control Register (DCCR), the EDO Enable bit has no affect on the DRAM waveform timing.

RCD - $\overline{RAS}$-to-$\overline{CAS}$ Delay Time

This field controls the number of system clocks between the assertion of $\overline{RAS}$ and the assertion of $\overline{CAS}$ for transfers in normal mode and for the initial transfer to a page in fast page mode and burst page mode. Because the column address is always driven 0.5 system clocks prior to the assertion of $\overline{CAS}$, RCD affects the driving of the column address. RCD does not affect refresh cycles. Refer to Figure 10-17 for normal mode timing. Refer to Figure 10-18 and Figure 10-19 for fast page mode and burst page mode timing.

0 = $\overline{RAS}$ asserts 1.0 system clock before the assertion of $\overline{CAS}$
1 = $\overline{RAS}$ asserts 2.0 system clocks before the assertion of $\overline{CAS}$

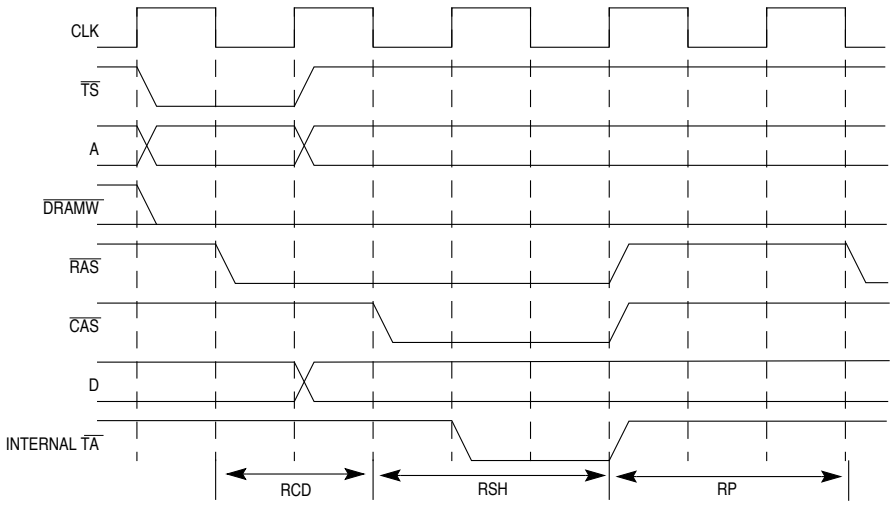
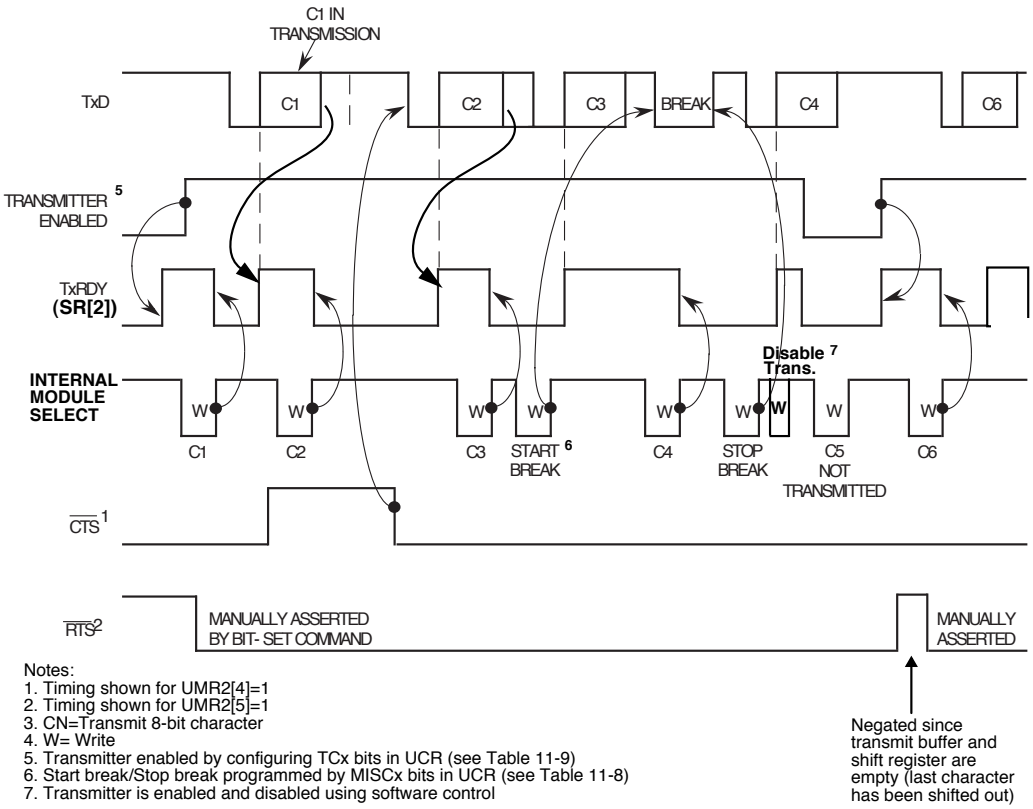


Figure 10-17. Normal Mode DRAM Transfer Timing



Notes:
1. Timing shown for UMR2[4]=1
2. Timing shown for UMR2[5]=1
3. CN=Transmit 8-bit character
4. W= Write
5. Transmitter enabled by configuring TCx bits in UCR (see Table 11-9)
6. Start break/Stop break programmed by MISCx bits in UCR (see Table 11-8)
7. Transmitter is enabled and disabled using software control

Figure 11-5. Transmitter Timing Diagram

“Force parity low” means forcing a 0 parity bit. “Force parity high” forces a 1 parity bit.

B/C1–B/C0 — Bits per Character

These bits select the number of data bits per character to be transmitted. The character length listed in Table 11-4 does not include start, parity, or stop bits.

Table 11-4. B/Cx Control Bits

B/C1	B/C0	BITS/CHARACTER
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

11.4.1.2 MODE REGISTER 2 (UMR2). UMR2 controls some of the UART module configuration. It is accessed when the mode register pointer points to UMR2, which occurs after any access to UMR1. Accesses to UMR2 do not change the pointer.

UMR2 MBAR + $180

7	6	5	4	3	2	1	0
CM1	CM0	TXRTS	TXCTS	SB3	SB2	SB1	SB0
RESET:							
0	0	0	0	0	0	0	0
READ/WRITE				SUPERVISOR OR USER			

CM1–CM0 — Channel Mode

These bits select a channel mode as listed in Table 11-5. See **Section 11.3.4 Looping Modes** for more information on the individual modes.

Table 11-5. CMx Control Bits

CM1	CM0	MODE
0	0	Normal
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

TxRTS — Transmitter Ready-to-Send

This bit controls the negation of the $\overline{\text{RTS}}$ signal.

In applications where the transmitter is disabled after transmission is complete, setting this bit causes the OP bit to be cleared automatically one bit time after the characters (if any) in the channel transmit shift register and the transmitter holding register are completely transmitted, including the programmed number of stop bits. This feature automatically terminates message transmission. You can perform this process by following these steps:

DATE: 9-2-98
REVISION NO.: 1.1
PAGES AFFECTED: SEE CHANGE BARS

SECTION 15
IEEE 1149.1 TEST ACCESS PORT (JTAG)

The MCF5206 includes dedicated user-accessible test logic that is fully compliant with the IEEE standard 1149.1 Standard Test Access Port and Boundary Scan Architecture. Use the following description in conjunction with the supporting IEEE document listed above. This section includes the description of those chip-specific items that the IEEE standard requires as well as those items specific to the MCF5206 implementation.

The MCF5206 JTAG test architecture implementation currently supports circuit board test strategies that are based on the IEEE standard. This architecture provides access to all of the data and chip control pins from the board edge connector through the standard four-pin test access port (TAP) and the active-low JTAG reset pin, $\overline{\text{TRST}}$. The test logic itself uses a static design and is wholly independent of the system logic, except where the JTAG is subordinate to other complimentary test modes (see the **Debug Support** section for more information). When in subordinate mode, the JTAG test logic is placed in reset and the TAP pins can be used for other purposes in accordance with the rules and restrictions set forth using a JTAG compliance-enable pin.

The MCF5206 JTAG implementation can:

- Perform boundary-scan operations to test circuit board electrical continuity
- Bypass the MCF5206 device by reducing the shift register path to a single cell
- Sample the MCF5206 system pins during operation and transparently shift out the result
- Set the MCF5206 output drive pins to fixed logic values while reducing the shift register path to a single cell
- Protect the MCF5206 system output and input pins from backdriving and random toggling (such as during in-circuit testing) by placing all system signal pins to high-impedance state

NOTE

The IEEE Standard 1149.1 test logic cannot be considered completely benign to those planning not to use JTAG capability. You must observe certain precautions to ensure that this logic does not interfere with system or debug operation. Refer to Section 15.6 Disabling the IEEE 1149.1 Standard Operation.

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MBAR + $0CB	PAR	8	Pin Assignment Register	$00	R/W
MBAR+$100	TMR1	16	Timer1 Mode Register	$0000	R/W
MBAR+$104	TRR1	16	Timer1 Reference Register	$FFFF	R/W
MBAR+$108	TCR1	16	Timer1 Capture Register	$0000	R
MBAR+$10C	TCN1	16	Timer1 Counter	$0000	R/W
MBAR+$111	TER1	8	Timer1 Event Register	$00	R/W
MBAR+$120	TMR2	16	Timer2 Mode Register	$0000	R/W
MBAR+$124	TRR2	16	Timer2 Reference Register	$FFFF	R/W
MBAR+$128	TCR2	16	Timer2 Capture Register	$0000	R
MBAR+$12C	TCN2	16	Timer2 Counter	$0000	R/W
MBAR+$131	TER2	8	Timer2 Event Register	$00	R/W
MBAR + $140	UMR1,2	8	UART1 Mode Registers	$00	R/W
MBAR+$144	USR/ UCSR	8	UART1 Status Register (R/W=1)/ UART1 Clock-Select Register (R/W=0)	USR=$00; UCSR=$DD	USR=R; UC-SR=W
MBAR+$148	UCR	8	UART1 Command Register	$00	W
MBAR+$14C	URB/UTB	8	UART1 Receive Buffer (R/W=1)/ UART1 Transmit Buffer (R/W=0)	URB=$FF; UTB=$00	URB=R; UTB=W
MBAR+$150	UIPCR/ UACR	8	UART Input Port Change Register (R/w=1);/ UART1 Auxilary Control Register (R/W=0)	UIPCR=$0F; UACR=$00	UIPCR=R; UACR=W;
MBAR+$154	UISR/ UIMR	8	UART1 Interrupt Status Register (R/W=1); UART1 Interrupt Mask Register (R/W=0)	UISR=$00; UIMR=$00	UISR=R; UIMR=W
MBAR+$158	UBG1	8	UART1 Baud Rate Generator Prescale MSB	uninitialized	W
MBAR+$15C	UBG2	8	UART1 Baud Rate Generator Prescale LSB	uninitialized	W
MBAR+$170	UIVR	8	UART1 Interrupt Vector Register	$0F	R/W
MBAR+$174	UIP	8	UART1 Input Port Register	$FF	R
MBAR+$178	UOP1	8	UART1 Output Port Bit Set CMD	UOP1[7:1]= undefined; UOP1=0	W
MBAR+$17C	UOP0	8	UART1 Output Port Bit Reset CMD	uninitialized	W
MBAR+$180	UMR1,2	8	UART2 Mode Registers	$00	R/W
MBAR+$184	USR/ UCSR	8	UART2 Status Register (R/W=1)/ UART1 Clock-Select Register (R/W=0)	USR=$00; UCSR=$DD	USR=R; UC-SR=W
MBAR+$188	UCR	8	UART2 Command Register	$00	W
MBAR+$18C	URB/UTB	8	UART2 Receive Buffer (R/W=1)/ UART1 Transmit Buffer (R/W=0)	URB=$FF; UTB=$00	URB=R; UTB=W
MBAR+$190	UIPCR/ UACR	8	UART2 Input Port Change Register (R/w=1);/ UART1 Auxilary Control Register (R/W=0)	UIPCR=$0F; UACR=$00	UIPCR=R; UACR=W;
MBAR+$194	UISR/ UIMR	8	UART2 Interrupt Status Register (R/W=1); UART1 Interrupt Mask Register (R/W=0)	UISR=$00; UIMR=$00	UISR=R; UIMR=W
MBAR+$198	UBG1	8	UART2 Baud Rate Generator Prescale MSB	uninitialized	R/W
MBAR+$19C	UBG2	8	UART2 Barud Rate Generator Prescale LSB	uninitialized	R/W
MBAR+$1B0	UIVR	8	UART2 Interrupt Vector Register	$0F	R/W
MBAR+$1B4	UIP	8	UART2 Input Port Register	$FF	R
MBAR+$1B8	UOP1	8	UART2 Output Port Bit Set CMD	UOP1[7:1]= undefined; UOP1=0	W
MBAR+$1BC	UOP0	8	UART2 Output Port Bit Reset CMD	uninitialized	W
MBAR + $1C5	PPDDR	8	Port A Data Direction Register	$00	R/W
MBAR + $1C9	PPDAT	8	Port A Data Register	$00	R/W
MBAR+$1E0	MADR	8	M-Bus Address Register	$00	R/W

To generate a ColdFire executable of the target debugger, you should use a ColdFire-compliant port of the same C compiler originally used to create the M68K debugger target. This procedure prevents differences in calling convention and parameter passing from C to handwritten assembly. Another advantage to this approach is that special C flags are retained. Many C compilers have special extensions as well.

Whichever approach is used, the assembly language lines that are outside the ColdFire instruction set must be identified. Any ColdFire assembler that properly flags nonColdFire instructions can be used. During the process of conversion, you can ignore architectural issues temporarily because the target is still an M68K.

Once the instruction set differences have been resolved, the architectural differences between the ColdFire and M68K need to be addressed.

B.3 INITIALIZATION CODE

The target software and firmware often execute code that identifies the type of processor. Such a process provides one port that works with various M68K Family members and implementations. The easiest way to identify the ColdFire architecture from other M68K processors is to execute an ILLEGAL opcode ($4AFC). This execution generates an exception stack frame while ensuring that the tracing is disabled. The first two bits of the exception stack frame would immediately determine whether the processor is a ColdFire processor.

Motorola suggests that ColdFire architecture testing be performed immediately to avoid executing potentially undefined opcodes in the ColdFire architecture. Unused opcodes in the ColdFire architecture are not guaranteed to result in an illegal instruction exception.

Another item to consider is that the ColdFire architecture will have integrated versions with modules yet to be defined. It may be a good idea to ensure that there are enough hooks to allow for initialization of routines.

B.4 EXCEPTION HANDLERS

When dealing with exceptions in debug-oriented software, it is often necessary to extract exception stack information to obtain the SR and PC. The format word (MC68010 and higher) is typically used by generalized exception routines. Their sole purpose is to catch unexpected exceptions and to easily use vector information to identify the cause of the exception. The MC68000 exception frame is different from that of other members of the M68K processors in that there is no notion of a format word. This difference would have forced target software to deal with exception stack frame differences already. The approach now in use provides guidance on handling ColdFire exception stack frame differences. In many low-level exception handlers, the extraction of the stacked SR, PC, or format word is performed in a common source file or the offsets are handled in some type of header file.

Interrupt handlers probably require no modification because in most cases, an interrupt occurs asynchronously with respect to normal program flow. Therefore, interrupt handlers cannot rely on items on the stack as it is often unnecessary to know exactly what was happening at the time of the interrupt.

dual stacks by placing some code at the beginning and end portions of exception handlers to change the stack pointer contents, if necessary, during exceptions. This approach has the disadvantage that interrupt latency would be degraded because interrupts would have to be disabled during the stack-swapping process, but enable full flexibility of the 68K stack model.