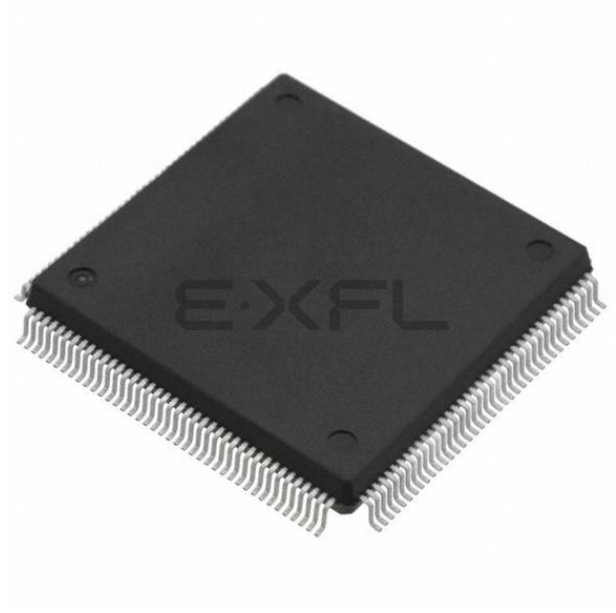


Welcome to E-XFL.COM

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	Obsolete
Core Processor	Coldfire V2
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
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/product-detail/nxp-semiconductors/mcf5206ab25a

LIST OF ILLUSTRATIONS (Continued)

Table 2-11. MCF5206 Signal Summary (Continued)

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	RESET STATE
Bus Driven	$\overline{BD}$	Out	Low	Negated
Clock Input	CLK	In	-	-
Reset	$\overline{RSTI}$	In	Low	-
Row Address Strobe	$\overline{RAS}$[1:0]	Out	Low	Master Reset - Negated Normal Reset - Unaffected
Column Address Strobe	$\overline{CAS}$[3:0]	Out	Low	Master Reset - Negated Normal Reset - Unaffected
DRAM Write	$\overline{DRAMW}$	Out	Low	Negated
Receive Data	RxD[1], RxD[2]	In	-	-
Transmit Data	TxD[1], TxD[2]	Out	-	Asserted
Request-To-Send	$\overline{RTS}$[1]	Out	Low	Negated
Request-To-Send	$\overline{RTS}$[2]/ $\overline{RSTO}$	Out/ Out	Low/ Low	Asserted
Clear-To-Send	$\overline{CTS}$[1], $\overline{CTS}$[2]	In	Low	-
Timer Input	TIN[1], TIN[2]	In	-	-
Timer Output	TOUT[1], TIN[2]	Out	-	Asserted
Serial Clock Line	SCL	In,Out	Low	Negated
Serial Data Line	SDA	In,Out	Low	Negated
General Purpose I/O/ Processor Status	PP[7:4]/ PST[3:0]	In.Out/ Out	-/ -	Three-stated
General Purpose I/O/ Debug Data	PP[3:0]/ DDATA[3:0]	In,Out/ Out	-/ -	Three-stated
Test Clock	TCK	In	-	-
Test Data Output/Development Serial Output	TDO/ DSO	Out/ Out	-/ -	Three-Stated/ Negated
Test Mode Select/ Break Point	TMS/ $\overline{BKPT}$	In/ In	-/ Low	-/ -
Test Data Input / Development Serial Input	TDI/ DSI	In/ In	-/ -	-/ -
Test Reset/Development Serial Clock	$\overline{TRST}$/ DSCLK	In/ In	Low/ -	-/ -
Motorola Test Mode	MTMOD	In	-	-
High Impedance	$\overline{HIZ}$	In	Low	-

3.6.1 Timing Assumptions

For the timing data presented in this section, the following assumptions apply:

1. The operand execution pipeline (OEP) is loaded with the opword and all required extension words at the beginning of each instruction execution. This implies that the OEP does not wait for the instruction fetch pipeline (IFP) to supply opwords and/or extension words.
2. The OEP does not experience any sequence-related pipeline stalls. For ColdFire 5200 processors, the most common example of this type of stall involves consecutive store operations, excluding the MOVEM instruction. For all STORE operations (except MOVEM), certain hardware resources within the processor are marked as “busy” for two clock cycles after the final DSOC cycle of the store instruction. If a subsequent STORE instruction is encountered within this 2-cycle window, it is stalled until the resource again becomes available. Thus, the maximum pipeline stall involving consecutive STORE operations is 2 cycles. The MOVEM instruction uses a different set of resources and this stall does not apply.
3. The OEP completes all memory accesses without any stall conditions caused by the memory itself. Thus, the timing details provided in this section assume that an infinite zero-wait state memory is attached to the processor core.
4. All operand data accesses are aligned on the same byte boundary as the operand size, i.e., 16-bit operands aligned on 0-modulo-2 addresses, 32-bit operands aligned on 0-modulo-4 addresses.

If the operand alignment fails these guidelines, it is misaligned. The processor core decomposes the misaligned operand reference into a series of aligned accesses as shown in Table 3-4.

Table 3-4. Misaligned Operand References

ADDRESS[1:0]	SIZE	KBUS OPERATIONS	ADDITIONAL C(R/W)
X1	Word	Byte, Byte	2(1/0) if read 1(0/1) if write
X1	Long	Byte, Word, Byte	3(2/0) if read 2(0/2) if write
10	Long	Word, Word	2(1/0) if read 1(0/1) if write

3.6.2 MOVE Instruction Execution Times

The execution times for the MOVE.{B,W} instructions are shown in Table 3-5, while Table 3-6 provides the timing for MOVE.L.

For all tables in this section, the execution time of any instruction using the PC-relative effective addressing modes is the same for the comparable An-relative mode.

The nomenclature “xxx.wl” refers to both forms of absolute addressing, xxx.w and xxx.l.

3.7 STANDARD ONE OPERAND INSTRUCTION EXECUTION TIMES

Table 3-7. One Operand Instruction Execution Times

OPCODE	□	EFFECTIVE ADDRESS							
		RN	(AN)	(AN)+	-(AN)	(D16,AN)	(D8,AN,XN*SF)	XXX.WL	#XXX
CLR.B	□	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.W	□	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
CLR.L	□	1(0/0)	1(0/1)	1(0/1)	1(0/1)	1(0/1)	2(0/1)	1(0/1)	—
EXT.W	Dx	1(0/0)	—	—	—	—	—	—	—
EXT.L	Dx	1(0/0)	—	—	—	—	—	—	—
EXTB.L	Dx	1(0/0)	—	—	—	—	—	—	—
NEG.L	Dx	1(0/0)	—	—	—	—	—	—	—
NEGX.L	Dx	1(0/0)	—	—	—	—	—	—	—
NOT.L	Dx	1(0/0)	—	—	—	—	—	—	—
Scc	Dx	1(0/0)	—	—	—	—	—	—	—
SWAP	Dx	1(0/0)	—	—	—	—	—	—	—
TST.B	□	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
TST.W	□	1(0/0)	3(1/0)	3(1/0)	3(1/0)	3(1/0)	4(1/0)	3(1/0)	1(0/0)
TST.L	□	1(0/0)	2(1/0)	2(1/0)	2(1/0)	2(1/0)	3(1/0)	2(1/0)	1(0/0)

3.10 BRANCH INSTRUCTION EXECUTION TIMES

Table 3-10. General Branch Instruction Execution Times

OPCODE	□	EFFECTIVE ADDRESS							
		RN	(AN)	(AN)+	-(AN)	(D16,AN) (D16,PC)	(D8,AN,Xi*SF) (D8,PC,Xi*SF)	XXX.WL	#XXX
BSR		–	–	–	–	3(0/1)	–	–	–
JMP	□	–	3(0/0)	–	–	3(0/0)	4(0/0)	3(0/0)	–
JSR	□	–	3(0/1)	–	–	3(0/1)	4(0/1)	3(0/1)	–
RTE		–	–	10(2/0)	–	–	–	–	–
RTS		–	–	5(1/0)	–	–	–	–	–

Table 3-11. BRA, Bcc Instruction Execution Times

OPCODE	FORWARD TAKEN	FORWARD NOT TAKEN	BACKWARD TAKEN	BACKWARD NOT TAKEN
BRA	2(0/0)	–	2(0/0)	–
Bcc	3(0/0)	1(0/0)	2(0/0)	3(0/0)

Figure 6-3 shows the required organization of data ports on the MCF5206 for 8-, 16-, and 32 bit devices. The four bytes shown are connected through the internal data bus and data multiplexer to the external data bus. This path is how the MCF5206 supports programmable port sizing and operand misalignment. The data multiplexer establishes the necessary connections for different combinations of address and data sizes.

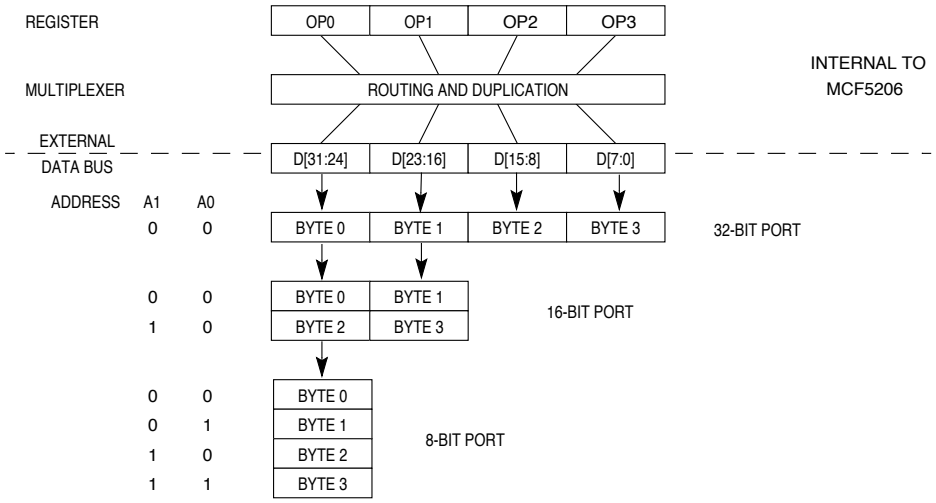


Figure 6-3. MCF5206 Interface to Various Port Sizes

The multiplexer takes the four bytes of the 32-bit bus and routes them to their required positions. For example, OP3 can be routed to D[7:0], as would be the normal case when interfacing to a 32-bit port. OP3 can be routed to D[23:16] for interfacing to a 16-bit port, or it can be routed to D[31:24] for interfacing to an 8-bit port. The operand size, address, and port size of the memory being accessed determines the positioning of bytes.

The MCF5206 can burst anytime the port size of the external slave being accessed is smaller than the operand size. If bursting is enabled, the MCF5206 burst transfers depending on the port size and operand alignment. For any transfer, the number of bytes transferred during a bus cycle is equal to or less than the size indicated by the SIZx outputs. For example, during the first bus cycle of a longword transfer to a 16-bit port where bursting is enabled, the SIZx outputs remain constant throughout the transfer and indicate that four bytes are to be transferred, although only two bytes are moved at a time.Table 6-5 lists the encodings for the SIZx bits for each port size for transfers where bursting is both enabled and disabled.

A[0] and A[1] also affect operation of the data multiplexer. During an operand transfer, A[31:2] indicate the longword base address of that portion of the operand to be accessed; A[1] and A[0] indicate the byte offset from the base. Table 6-6 lists the encoding of A[1] and A[0] and the corresponding byte offset from the longword base.

Figure 6-23 shows a bursting user data line-write transfer to a 32-bit port using asynchronous termination.

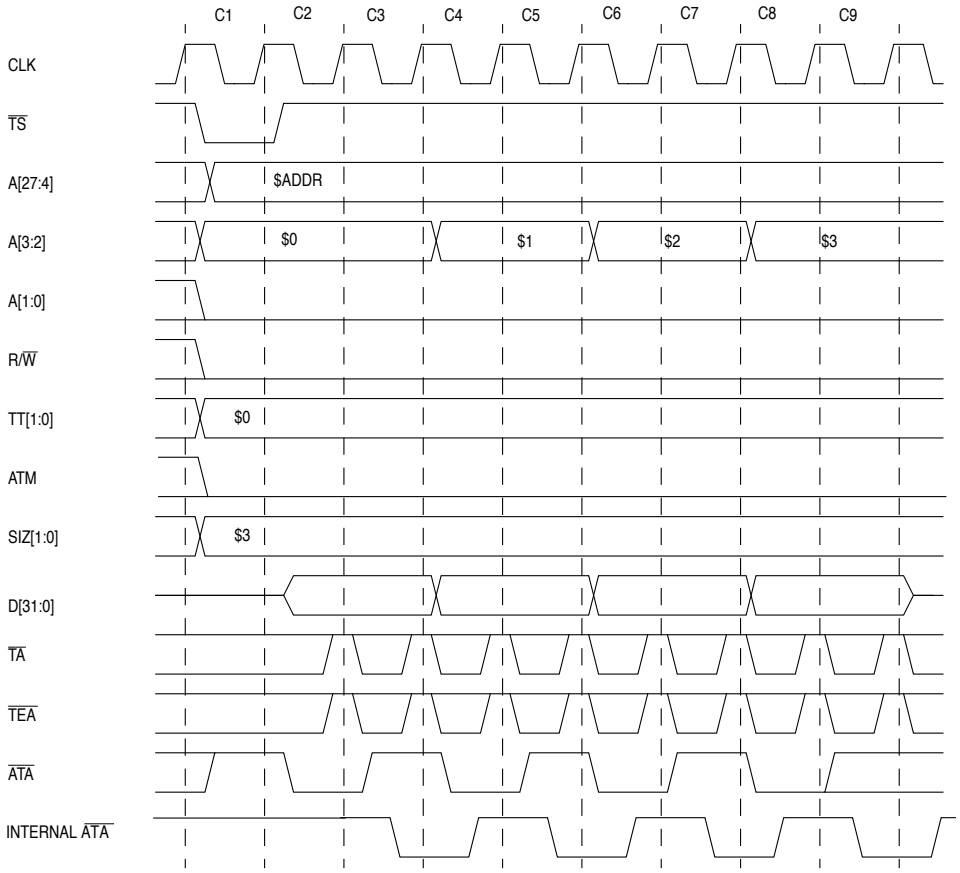


Figure 6-23. Bursting Line-Write from 32-Bit Port Using Asynchronous Termination (One Wait State)

Clock 1 (C1)

The write 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 data. The read/write (R/$\overline{W}$) signal is driven low for a write cycle, and the size signals (SIZ[1:0]) are driven to $3 to indicate a line transfer. The MCF5206 asserts $\overline{TS}$ to indicate the beginning of a bus cycle.

Table 6-11. MCF5206 Three-Wire Bus Arbitration Protocol Transition Conditions

EXPLICIT OWN	C1	N	N	A	-	-	-	-	Explicit Own
	C2	N	N	N	Y	-	-	-	Explicit Own
	C3	N	N	N	N	-	N	-	EM Own
	C4	N	N	N	-	-	Y	N	Explicit Own
	C5	N	N	N	N	-	Y	Y	EM Own
EM OWN	D1	N	N	N	-	-	-	-	EM Own
	D2	N	N	A	A	-	-	-	Explicit Own
	D3	N	N	A	N	N	-	-	Implicit Own
	D4	N	N	A	N	A	-	-	Explicit Own

NOTES:

1)"N" means negated; "A" means asserted; "EM" means external master.

2)End of Cycle: Whatever terminates a bus transaction whether it is normal or bus error. Note that bus cycles that result from a burst inhibited transfer are considered part of that original transfer.

3)$\overline{IBR}$ refers to an internal bus request.The output signals $\overline{BR}$ is a registered version of $\overline{IBR}$ when $\overline{BG}$ is negated and $\overline{BD}$ is negated. There is an internal bus request when the Coldfire core requires the external bus for an operand transfer.

Table 6-12. MCF5206 Three-Wire Arbitration Protocol State Diagram

STATE	OWN	BUS STATUS	BD
Reset	No	Not Driven	Negated
Implicit Own	Yes	Not Driven	Negated
Explicit Own	Yes	Driven	Asserted
EM Own	No	Not Driven	Negated

The MCF5206 can be in any one of four arbitration states during bus operation: reset, external master own, implicit ownership, and explicit ownership.

The reset state is entered whenever $\overline{RSTI}$ or software watchdog reset is asserted in any bus arbitration state. When $\overline{RSTI}$ and the software watchdog reset are negated, the MCF5206 proceeds to the implicit ownership state or external master ownership state, depending on bus grant ($\overline{BG}$).

The external master ownership state denotes the MCF5206 does not have ownership (bus grant ($\overline{BG}$) negated) of the bus and the MCF5206 does not drive the bus. The MCF5206 can assert memory control signals (i.e., $\overline{CS}$[7:0], $\overline{WE}$[3:0], $\overline{RAS}$[1:0] or $\overline{CAS}$[3:0]) $\overline{TA}$ and $\overline{BR}$ during this state.

The implicit ownership state indicates that the MCF5206 owns the bus because bus grant ($\overline{BG}$) is asserted to it. The MCF5206, however, is not ready to begin a bus cycle and the bus lock bit in the SIMR is cleared, and it keeps the bus three-stated until an internal bus request occurs or the bus lock bit in the SIMR is set to 1.

The MCF5206 explicitly owns the bus when the bus is granted to it (bus grant ($\overline{BG}$) asserted) and at least one bus cycle has initiated or the bus lock bit in the SIMR is set to 1. The MCF5206 asserts $\overline{BD}$ in this state to indicate the MCF5206 has explicit ownership of the bus. Until bus grant ($\overline{BG}$) is negated, the MCF5206 regains explicit ownership of the

- Addresses not assigned to a register and undefined register bits are reserved for future expansion. Write accesses to these reserved address spaces and reserved register bits have no effect; read accesses returns zeros.
- The reset value column indicates the register initial value at reset. Certain registers may be uninitialized at reset.
- The access column indicates if the corresponding register allows both read/write functionality (R/W), read-only functionality (R), or write-only functionality (W). An attempted read access to a write-only register returns zeros. An attempted write access to a read-only register is ignored and no write occurs.

Table 7-2. Memory Map of SIM Registers

ADDRESS	NAME	WIDTH	DESCRIPTION	RESET VALUE	ACCESS
MOVEC with $C0F	MBAR	32	Module Base Address Register	uninitialized (except V=0)	W
MBAR + $003	SIMR	8	SIM Configuration Register	$C0	R/W
MBAR + $014	ICR1	8	Interrupt Control Register 1 - External IRQ1/IPL1	$04	R/W
MBAR + $015	ICR2	8	Interrupt Control Register 2 - External IPL2	$08	R/W
MBAR + $016	ICR3	8	Interrupt Control Register 3 - External IPL3	$0C	R/W
MBAR + $017	ICR4	8	Interrupt Control Register 4 - External IRQ4/IPL4	$10	R/W
MBAR + $018	ICR5	8	Interrupt Control Register 5 - External IPL5	$14	R/W
MBAR + $019	ICR6	8	Interrupt Control Register 6 - External IPL6	$18	R/W
MBAR + $01A	ICR7	8	Interrupt Control Register 7 - External IRQ7/IPL7	$1C	R/W
MBAR + $01B	ICR8	8	Interrupt Control Register 8 - SWT	$1C	R/W
MBAR + $01C	ICR9	8	Interrupt Control Register 9 - Timer 1 Interrupt	$80	R/W
MBAR + $01D	ICR10	8	Interrupt Control Register 10 - Timer 2 Interrupt	$80	R/W
MBAR + $01E	ICR11	8	Interrupt Control Register 11 - MBUS Interrupt	$80	R/W
MBAR + $01F	ICR12	8	Interrupt Control Register 12 - UART 1 Interrupt	$00	R/W
MBAR + $020	ICR13	8	Interrupt Control Register 13 - UART2 Interrupt	$00	R/W
MBAR + $036	IMR	16	Interrupt Mask Register	$3FFE	R/W
MBAR + $03A	IPR	16	Interrupt Pending Register	$0000	R
MBAR + $040	RSR	8	Reset Status Register	$80 or $20	R/W
MBAR + $041	SYPCR	8	System Protection Control Register	$00	R/W
MBAR + $042	SWIVR	8	Software Watchdog Interrupt Vector Register	$0F	R/W
MBAR + $043	SWSR	8	Software Watchdog Service Register	uninitialized	W
MBAR + $0CB	PAR	8	Pin Assignment Register	$00	R/W

7.3.2 SIM Registers

7.3.2.1 MODULE BASE ADDRESS REGISTER (MBAR). The MBAR determines the base address location of all internal module resources such as registers as well as the definition of the types of accesses that are allowed for these resources.

The MBAR is a 32-bit write-only supervisor control register that physically resides in the SIM. It is accessed in the CPU address space via the MOVEC instruction with an Rc encoding of $C0F. The MBAR can be read and written to when in Background Debug mode (BDM). At system reset, the V-bit is cleared and the remainder of the MBAR bits are uninitialized. To

Table 7-3. Interrupt Control Register Assignments

INTERRUPT SOURCE	INTERRUPT CONTROL REGISTER (ICR)	ICR RESET VALUE	ICR IINTERRUPT LEVEL (IL2 - IL0) VALUE
External Interrupt Request 1 External Interrupt Priority Level 1	ICR1	$04	$1
External Interrupt Priority Level 2	ICR2	$08	$2
External Interrupt Priority Level 3	ICR3	$0C	$3
External Interrupt Request 4 External Interrupt Priority Level 4	ICR4	$10	$4
External Interrupt Priority Level 5	ICR5	$14	$5
External Interrupt Priority Level 6	ICR6	$18	$6
External Interrupt Request 7 External Interrupt Priority Level 7	ICR7	$1C	$7
Software Watchdog Timer	ICR8	$1C	$7
Timer 1	ICR9	$80	User Programmable
Timer 2	ICR10	$80	User Programmable
MBUS (I^2C)	ICR11	$80	User Programmable
UART 1	ICR12	$00	User Programmable
UART 2	ICR13	$00	User Programmable

The ICRs are 8-bit read-write registers. See Table 7-3 for the reset values of each ICR.

Interrupt Control Register(ICR)

7	6	5	4	3	2	1	0
AVEC	-	-	IL2	IL1	IL0	IP1	IP0

R/W

AVEC - Autovector Enable

This bit determines if the interrupt acknowledge cycle for the interrupt level indicated in IL2-IL0 for each interrupt input requires an autovector response. If this bit is set, a vector number is internally generated, which is the sum of the interrupt level, IL2-IL0, plus 24. Seven distinct autovectors can be used corresponding to the 7 levels of interrupt. If this bit is clear, the external device or internal module must return the vector number during an interrupt acknowledge cycle.

NOTE

For the SWT, the corresponding AVEC is a reserved bit and set to zero, disabling autovector generation in response to SWT generated interrupts. The SWT returns the interrupt vector in SWIVR. The AVEC bits in the interrupt control registers for the timers and MBUS peripherals are reserved and are always set to 1. The autovector value is generated for each of these interrupts.

0 = Interrupting source returns vector number during the interrupt-acknowledge cycle.
1 = SIM internally generates vector number during the interrupt-acknowledge cycle.

During a write transfers, these outputs indicate which bytes within a long-word transfer are being selected and which bytes of the data bus are used for the transfer. $\overline{WE}[0]$ controls D[31:24], $\overline{WE}[1]$ controls D[23:16], $\overline{WE}[2]$ controls D[15:8] and $\overline{WE}[3]$ controls D[7:0]. These signals provide byte data-select signals that are decoded from the SIZx, A[1:0] signals in addition to the programmed port size and burst capability of the memory being accessed, as shown in Table 8-1.

Table 8-1. Data Bus Byte Write-Enable Signals

TRANSFER SIZE	PORT SIZE	BURST	SIZ1	SIZ0	A1	A0	$\overline{WE}[0]$ D[31:24]	$\overline{WE}[1]$ D[23:16]	$\overline{WE}[2]$ D[15:8]	$\overline{WE}[3]$ D[7:0]
BYTE	8-BIT	0/1	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	0/1	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	0/1	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	1	0	0	0	0	0	1	1
					1	0	0	0	1	1
	32-BIT	0/1	1	0	0	0	0	0	1	1
					1	0	1	1	0	0

operates in fast page mode for ColdFire core initiated transfers, but operates in burst page mode for external master initiated transfers.

Figure 10-11 shows the effect of bus arbitration on the DRAM signals when the external bus is idle and a page is open in fast page mode.

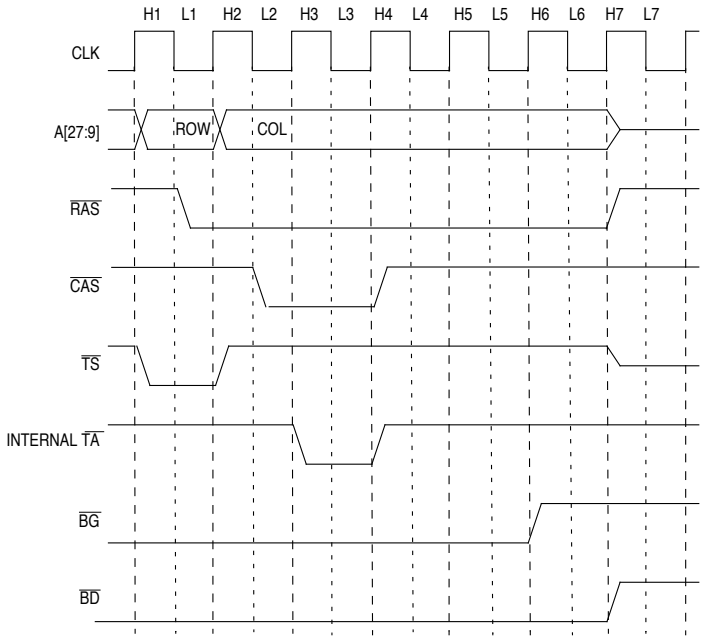


Figure 10-11. Bus Arbitration in Fast Page Mode

Clock H1

A Fast Page Mode transfer starts in H1. During H1, the MCF5206 drives the row address on A[27:9], and asserts $\overline{TS}$.

Clock L1

The MCF5206 asserts $\overline{RAS}$ to indicate the row address is valid on A[27:9].

Clock H2

The MCF5206 negates $\overline{TS}$, and drives the column address on A[27:9].

Clock L2

The MCF5206 asserts $\overline{CAS}$ to indicate the column address is valid on A[27:9].

mnemonic for the bit. The values shown below the register description are the values of those register bits after a hardware reset. A value of U indicates that the bit value is unaffected by reset. The read/write status is shown in the last line.

Table 11-2. UART Module Programming Model

UART1 2	UART1	REGISTER READ (R/W = 1)	REGISTER WRITE (R/W = 0)
MBAR+$180	MBAR+$140	Mode Register (UMR1, UMR2)	Mode Register (UMR1, UMR2)
MBAR+$184	MBAR+$144	Status Register (USR)	Clock-Select Register (UCSR)
MBAR+$188	MBAR+$148	DO NOT ACCESS[1]	Command Register (UCR)
MBAR+$18C	MBAR+$14C	Receiver Buffer (URB)	Transmitter Buffer (UTB)
MBAR+$190	MBAR+$150	Input Port Change Register (UIPCR)	Auxiliary Control Register (UACR)
MBAR+$194	MBAR+$154	Interrupt Status Register (UISR)	Interrupt Mask Register (UIMR)
MBAR+$198	MBAR+$158	Baud Rate Generator Prescale MSB (UBG1)	Baud Rate Generator Prescale MSB (UBG1)
MBAR+$19C	MBAR+$15C	Baud Rate Generator Prescale LSB (UBG2)	Baud Rate Generator Prescale LSB (UBG2)
		DO NOT ACCESS[1]	
MBAR+$1B0	MBAR+$170	Interrupt Vector Register (UIVR)	INterrupt Vector Register (UIVR)
MBAR+$1B4	MBAR+$174	Input Port Register (UIP)	DO NOT ACCESS[1]
MBAR+$1B8	MBAR+$178	DO NOT ACCESS[1]	Output Port Bit Set CMD (UOP1)[2]
MBAR+$1BC	MBAR+$17C	DO NOT ACCESS[1]	Output Port Bit Reset CMD (UOP0)[2]

NOTES: 1. This address is used for factory testing and should not be read. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents can also be changed.

2. Address-triggered commands.

11.4.1.1 MODE REGISTER 1 (UMR1). UMR1 controls some of the UART module configuration. This register can be read or written at any time and is accessed when the mode register pointer points to UMR1. The pointer is set to UMR1 by RESET or by a set pointer command using the control register. After reading or writing UMR1, the pointer points to UMR2.

UMR1 MBAR + $140

7	6	5	4	3	2	1	0
RXRTS	RXIRQ	ERR	PM1	PM0	PT	B/C1	B/C0
RESET							
0	0	0	0	0	0	0	0
READ/WRITE				SUPERVISOR OR USER			

RxRTS — Receiver Request-to-Send Control

1 = On receipt of a valid start bit, $\overline{RTS}$ is negated if the UART FIFO is full. $\overline{RTS}$ is reasserted when the FIFO has an empty position available.

0 = The receiver has no effect on $\overline{RTS}$. The RTS is asserted by writing a one to the Output Port Bit Set Register (UOP1)

MISC3–MISC0 — Miscellaneous Commands

These bits select a single command as listed in Table 11-9.

Table 11-9. MISCx Control Bits

MISC2	MISC1	MISC0	COMMAND
0	0	0	No Command
0	0	1	Reset Mode Register Pointer
0	1	0	Reset Receiver
0	1	1	Reset Transmitter
1	0	0	Reset Error Status
1	0	1	Reset Break-Change Interrupt
1	1	0	Start Break
1	1	1	Stop Break

The commands are described as follows:

Reset Mode Register Pointer

The reset mode register pointer command causes the mode register pointer to point to UMR1.

Reset Receiver

The reset receiver command resets the receiver. The receiver is immediately disabled, the FFULL and RxRDY bits in the USR are cleared, and the receiver FIFO pointer is reinitialized. All other registers are unaltered. Use this command instead of the receiver-disable command whenever the receiver configuration is changed (it places the receiver in a known state).

Reset Transmitter

The reset transmitter command resets the transmitter. The transmitter is immediately disabled and the TxEMP and TxRDY bits in the USR are cleared. All other registers are unaltered. Use this command instead of the transmitter-disable command whenever the transmitter configuration is changed (it places the transmitter in a known state).

Reset Error Status

The reset error status command clears the RB, FE, PE, and OE bits in the USR. This command is also used in the block mode to clear all error bits after a data block is received.

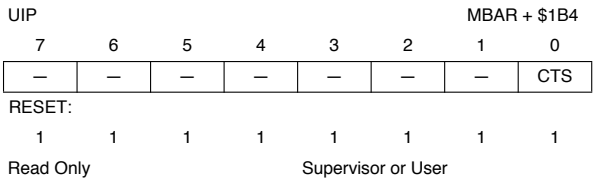
Reset Break-Change Interrupt

The reset break-change interrupt command clears the delta break (DBx) bit in the UISR.

IVR7–IVR0 — Interrupt Vector Bits

This 8-bit number indicates the offset from the base of the vector table where the address of the exception handler for the specified interrupt is located. The UIVR is reset to $0F, which indicates an uninitialized interrupt condition.

11.4.1.14.1 Input Port Register (UIP). The UIP register shows the current state of the $\overline{\text{CTS}}$ input.



$\overline{\text{CTS}}$ — Current State

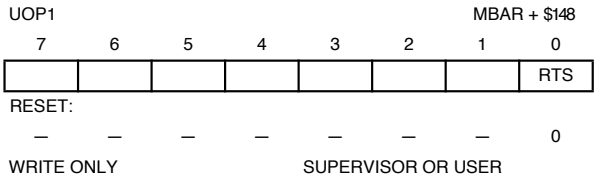
1 = The current state of the $\overline{\text{CTS}}$ input is logic 1
0 = The current state of the $\overline{\text{CTS}}$ input is logic 0

The information contained in this bit is latched and reflects the state of the input pin at the time that the UIP is read.

NOTE

This bit has the same function and value as the UIPCR bit 0.

11.4.1.14.2 Output Port Data Registers (UOP1, UOP0). The $\overline{\text{RTS}}$ output is set by a bit set command (writing to UOP1) and is cleared by a bit reset command (writing to UOP0).



$\overline{\text{RTS}}$ — Output Port Parallel Output

1 = A write cycle to the OPset address asserts the $\overline{\text{RTS}}$ signal.
0 = This bit is not affected by writing a zero to this address.

NOTE

The output port bits are inverted at the pins so the $\overline{\text{RTS}}$ set bit provides an asserted $\overline{\text{RTS}}$ pin.

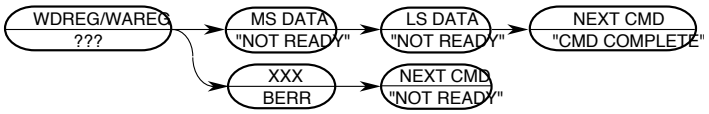
14.2.3.4.2 Write A/$\overline{D}$ Register (WAREG/WDREG). The operand (longword) data is written to the specified address or data register. All 32 register bits are altered by the write. A bus error response is returned if the CPU core is not halted.

Command Formats:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$2				$0				$8				A/D	REGISTER		
DATA [31:16]															
DATA [15:0]															

WAREG/WDREG Command

Command Sequence:



Operand Data:

Longword data is written into the specified address or data register. The data is supplied most significant word first.

Result Data:

Command complete status ($0FFFF) is returned when register write is complete.

14.3 REAL-TIME DEBUG SUPPORT

ColdFire processors provide support for the debug of real-time applications. For these types of embedded systems, the processor cannot be halted during debug but must continue to operate. The foundation of this area of debug support is that while the processor cannot be halted to allow debugging, the system can tolerate small intrusions into the real-time operation.

As discussed in the previous subsection, the debug module provides a number of hardware resources to support various hardware breakpoint functions. Specifically, three types of breakpoints are supported: PC with mask, operand address range, and data with mask. These three basic breakpoints can be configured into one- or two-level triggers with the exact trigger response also programmable.

14.3.1 Programming Model

In addition to the existing BDM commands that provide access to the processor's registers

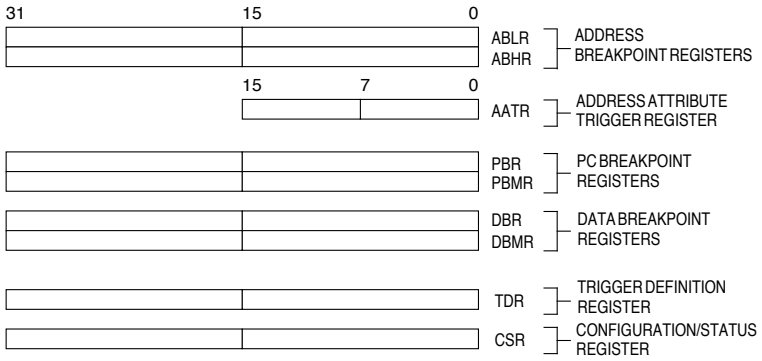


Figure 14-5. Debug Programming Model

and the memory subsystem, the Debug module contains a number of registers to support the required functionality. All of these registers are treated as 32-bit quantities, regardless of the actual number of bits in the implementation. The registers, known as the Debug Control Registers (DRc), are addressed using a 4-bit value as part of two new BDM commands (WDREG, RDREG).

These registers are also accessible from the processor's supervisor programming model through the execution of the WDEBUG instruction (Figure 14-5 illustrates the debug module programming model). Thus, the breakpoint hardware within the debug module can be accessed by the external development system using the serial interface, or by the operating system running on the processor core. It is the responsibility of the software to guarantee that all accesses to these resources are serialized and logically consistent. The hardware

15.5 RESTRICTIONS

The test logic is implemented using static logic design, and TCK can be stopped in either a high or low state without loss of data. The system logic, however, operates on a different system clock which is not synchronized to TCK internally. Any mixed operation requiring the use of 1149.1 test logic in conjunction with system functional logic that uses both clocks, must have coordination and synchronization of these clocks done externally to the MCF5206.

15.6 DISABLING THE IEEE 1149.1 STANDARD OPERATION

There are two methods by which the MCF5206 can be used without the IEEE 1149.1 test logic being active: (1) Nonuse of the JTAG test logic by either nontermination (disconnection) or intentional fixing of TAP logic values, and (2) Intentional disabling of the JTAG test logic by assertion of the $\overline{\text{JTAG}}$ signal (entering Debug mode).

There are several considerations that must be addressed if the IEEE 1149.1 logic is not going to be used once the MCF5206 is assembled onto a board. The prime consideration is to ensure that the IEEE 1149.1 test logic remains transparent and benign to the system logic during functional operation. This requires the minimum of either connecting the $\overline{\text{TRST}}$ pin to logic 0, or connecting the TCK clock pin to a clock source that supplies five rising edges and the falling edge after the fifth rising edge, to ensure that the part enters the test-logic-reset state. The recommended solution is to connect $\overline{\text{TRST}}$ to logic 0. Another consideration is that the TCK pin does not have an internal pullup as is required on the TMS, TDI, and $\overline{\text{TRST}}$ pins; therefore, it should not be left unterminated to preclude mid-level input values. Figure 15-3 shows pin values recommended for disabling JTAG with the MCF5206 in JTAG mode ($\overline{\text{JTAG}}$=0).

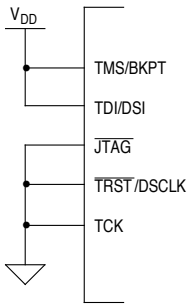
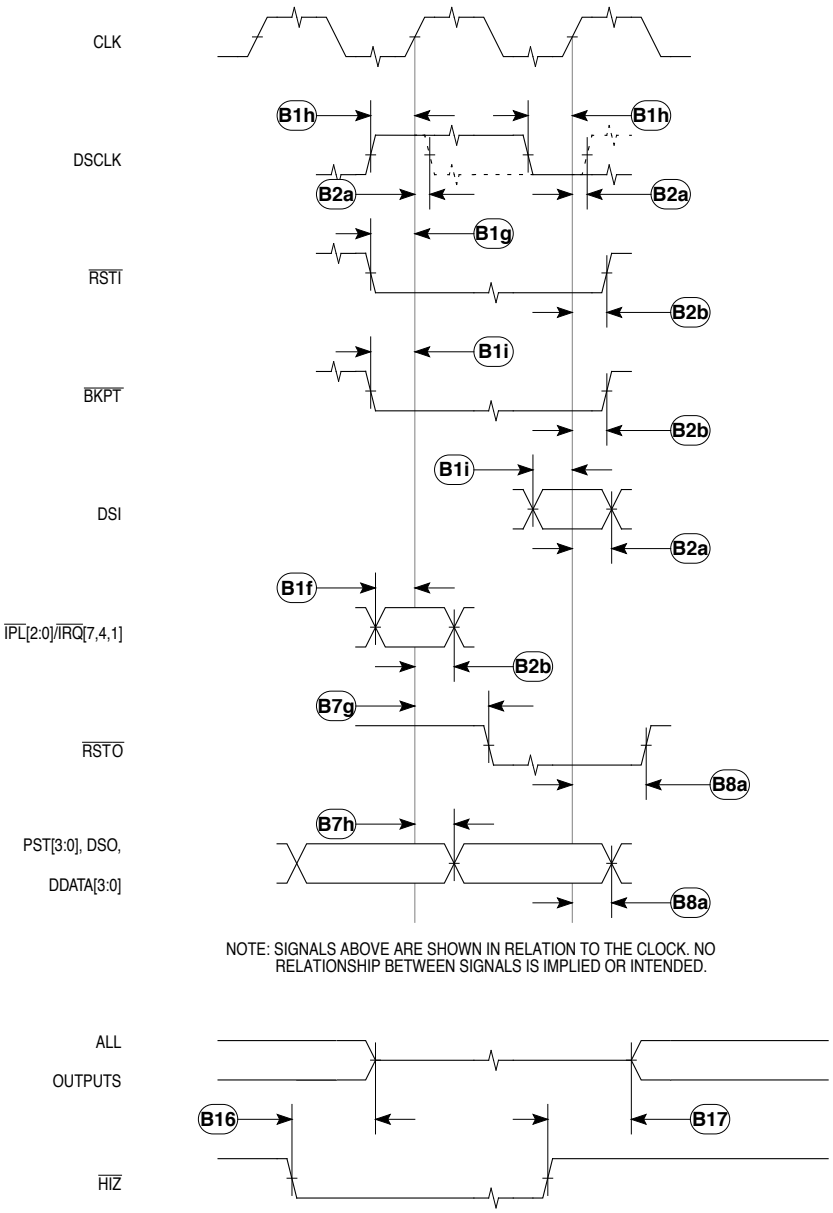


Figure 15-3. Disabling JTAG in JTAG Mode

A second method of using the MCF5206 without the IEEE 1149.1 logic being active is to select Debug mode by placing a logic 1 on the defined compliance enable pin, $\overline{\text{JTAG}}$. When $\overline{\text{JTAG}}$ is a logic 1, then the IEEE 1149.1 test controller is placed in the test-logic-reset state by the internal assertion of the $\overline{\text{TRST}}$ signal to the controller, and, the TAP pins function as debug mode pins. While in JTAG mode, input pins TDI/DSI, TMS/$\overline{\text{BKPT}}$, and



Miscellaneous Signal Timing

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