Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	Coldfire V3
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	EBI/EMI, I²C, UART/USART
Peripherals	DMA, POR, WDT
Number of I/O	16
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	208-BFQFP
Supplier Device Package	208-FQFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mcf5307cai66b

CONTENTS

The Version 2 ColdFire core implemented the original debug architecture, now called Revision A. Based on feedback from customers and third-party developers, enhancements have been added to succeeding generations of ColdFire cores. The Version 3 core implements Revision B of the debug architecture, providing more flexibility for configuring the hardware breakpoint trigger registers and removing the restrictions involving concurrent BDM processing while hardware breakpoint registers are active.

5.2 Signal Description

Table 5-1 describes debug module signals. All ColdFire debug signals are unidirectional and related to a rising edge of the processor core's clock signal. The standard 26-pin debug connector is shown in Section 5.7, "Motorola-Recommended BDM Pinout."

Table 5-1. Debug Module Signals

Signal	Description
Development Serial Clock (DSCLK)	Internally synchronized input. (The logic level on DSCLK is validated if it has the same value on two consecutive rising CLKIN edges.) Clocks the serial communication port to the debug module. Maximum frequency is 1/5 the processor CLK speed. At the synchronized rising edge of DSCLK, the data input on DSI is sampled and DSO changes state.
Development Serial Input (DSI)	Internally synchronized input that provides data input for the serial communication port to the debug module.
Development Serial Output (DSO)	Provides serial output communication for debug module responses. DSO is registered internally.
Breakpoint ($\overline{\text{BKPT}}$)	Input used to request a manual breakpoint. Assertion of $\overline{\text{BKPT}}$ puts the processor into a halted state after the current instruction completes. Halt status is reflected on processor status/debug data signals (PST[3:0]) as the value 0xF. If CSR[BKD] is set (disabling normal $\overline{\text{BKPT}}$ functionality), asserting $\overline{\text{BKPT}}$ generates a debug interrupt exception in the processor.
Processor Status Clock (PSTCLK)	Delayed version of the processor clock. Its rising edge appears in the center of valid PST and DDATA output. See Figure 5-2. PSTCLK indicates when the development system should sample PST and DDATA values. If real-time trace is not used, setting CSR[PCD] keeps PSTCLK, PST and DDATA outputs from toggling without disabling triggers. Non-quiescent operation can be reenabled by clearing CSR[PCD], although the emulator must resynchronize with the PST and DDATA outputs. PSTCLK starts clocking only when the first non-zero PST value (0xC, 0xD, or 0xF) occurs during system reset exception processing. Table 5-2 describes PST values. Chapter 7, "Phase-Locked Loop (PLL)," describes PSTCLK generation.
Debug Data (DDATA[3:0])	These output signals display the hardware register breakpoint status as a default, or optionally, captured address and operand values. The capturing of data values is controlled by the setting of the CSR. Additionally, execution of the WDDATA instruction by the processor captures operands which are displayed on DDATA. These signals are updated each processor cycle.
Processor Status (PST[3:0])	These output signals report the processor status. Table 5-2 shows the encoding of these signals. These outputs indicate the current status of the processor pipeline and, as a result, are not related to the current bus transfer. The PST value is updated each processor cycle.

Bytes are displayed in least-to-most-significant order. The processor captures only those target addresses associated with taken branches which use a variant addressing mode, that is, RTE and RTS instructions, JMP and JSR instructions using address register indirect or indexed addressing modes, and all exception vectors.

The simplest example of a branch instruction using a variant address is the compiled code for a C language case statement. Typically, the evaluation of this statement uses the variable of an expression as an index into a table of offsets, where each offset points to a unique case within the structure. For such change-of-flow operations, the MCF5307 uses the debug pins to output the following sequence of information on successive processor clock cycles:

1. Use PST (0x5) to identify that a taken branch was executed.
2. Using the PST pins, optionally signal the target address to be displayed sequentially on the DDATA pins. Encodings 0x9–0xB identify the number of bytes displayed.
3. The new target address is optionally available on subsequent cycles using the DDATA port. The number of bytes of the target address displayed on this port is configurable (2, 3, or 4 bytes).

Another example of a variant branch instruction would be a JMP (A0) instruction. Figure 5-3 shows when the PST and DDATA outputs that indicate when a JMP (A0) executed, assuming the CSR was programmed to display the lower 2 bytes of an address.

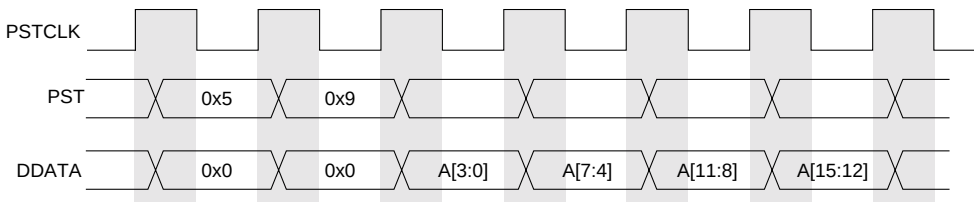

Figure 5-3. Example JMP Instruction Output on PST/DDATA

PST is driven with a 0x5 in the first cycle and 0x9 in the second. The 0x5 indicates a taken branch and the marker value 0x9 indicates a 2-byte address. Thus, the 4 subsequent DDATA nibbles display the lower 2 bytes of address register A0 in least-to-most-significant nibble order. The PST output after the JMP instruction completes depends on the target instruction. The PST can continue with the next instruction before the address has completely displayed on DDATA because of the DDATA FIFO. If the FIFO is full and the next instruction has captured values to display on DDATA, the pipeline stalls (PST = 0x0) until space is available in the FIFO.

5.4 Programming Model

In addition to the existing BDM commands that provide access to the processor's registers and the memory subsystem, the debug module contains nine registers to support the required functionality. These registers are also accessible from the processor's supervisor

Table 5-9 describes DBR fields.

Table 5-9. DBR Field Descriptions

Bits	Name	Description
31–0	Data	Data breakpoint value. Contains the value to be compared with the data value from the processor's local bus as a breakpoint trigger.

Table 5-10 describes DBMR fields.

Table 5-10. DBMR Field Descriptions

Bits	Name	Description
31–0	Mask	Data breakpoint mask. The 32-bit mask for the data breakpoint trigger. Clearing a DBR bit allows the corresponding DBR bit to be compared to the appropriate bit of the processor's local data bus. Setting a DBMR bit causes that bit to be ignored.

The DBR supports both aligned and misaligned references. Table 5-11 shows relationships between processor address, access size, and location within the 32-bit data bus.

Table 5-11. Access Size and Operand Data Location

A[1:0]	Access Size	Operand Location
00	Byte	D[31:24]
01	Byte	D[23:16]
10	Byte	D[15:8]
11	Byte	D[7:0]
0x	Word	D[31:16]
1x	Word	D[15:0]
xx	Longword	D[31:0]

5.4.6 Program Counter Breakpoint/Mask Registers (PBR, PBMR)

The PC breakpoint register (PBR) defines an instruction address for use as part of the trigger. This register's contents are compared with the processor's program counter register when TDR is configured appropriately. PBR bits are masked by clearing corresponding PBMR bits. Results are compared with the processor's program counter register, as defined in TDR. Figure 5-10 shows the PC breakpoint register.

5.6.1 Theory of Operation

Breakpoint hardware can be configured to respond to triggers in several ways. The response desired is programmed into TDR. As shown in Table 5-21, when a breakpoint is triggered, an indication (CSR[BSTAT]) is provided on the DDATA output port when it is not displaying captured processor status, operands, or branch addresses.

Table 5-21. DDATA[3:0]/CSR[BSTAT] Breakpoint Response

DDATA[3:0]/CSR[BSTAT] [1]	Breakpoint Status
0000/0000	No breakpoints enabled
0010/0001	Waiting for level-1 breakpoint
0100/0010	Level-1 breakpoint triggered
1010/0101	Waiting for level-2 breakpoint
1100/0110	Level-2 breakpoint triggered

[1] Encodings not shown are reserved for future use.

The breakpoint status is also posted in CSR. Note that CSR[BSTAT] is cleared by a CSR read when either a level-2 breakpoint is triggered or a level-1 breakpoint is triggered and a level-2 breakpoint is not enabled. Status is also cleared by writing to TDR.

BDM instructions use the appropriate registers to load and configure breakpoints. As the system operates, a breakpoint trigger generates the response defined in TDR.

PC breakpoints are treated in a precise manner—exception recognition and processing are initiated before the excepting instruction is executed. All other breakpoint events are recognized on the processor's local bus, but are made pending to the processor and sampled like other interrupt conditions. As a result, these interrupts are imprecise.

In systems that tolerate the processor being halted, a BDM-entry can be used. With TDR[TRC] = 01, a breakpoint trigger causes the core to halt (PST = 0xF).

If the processor core cannot be halted, the debug interrupt can be used. With this configuration, TDR[TRC] = 10, the breakpoint trigger becomes a debug interrupt to the processor, which is treated higher than the nonmaskable level-7 interrupt request. As with all interrupts, it is made pending until the processor reaches a sample point, which occurs once per instruction. Again, the hardware forces the PC breakpoint to occur before the targeted instruction executes. This is possible because the PC breakpoint is enabled when interrupt sampling occurs. For address and data breakpoints, reporting is considered imprecise because several instructions may execute after the triggering address or data is detected.

As soon as the debug interrupt is recognized, the processor aborts execution and initiates exception processing. This event is signaled externally by the assertion of a unique PST value (PST = 0xD) for multiple cycles. The core enters emulator mode when exception processing begins. After the standard 8-byte exception stack is created, the processor

Note that the internal DMA has higher priority than the core if the internal DMA has its bandwidth BWC bits set to 000 (maximum bandwidth).

- Park on master core priority (PARK = 01)—The core retains bus mastership as long as it needs it. After it negates its internal bus request, the core does not have to rearbitrate for the bus unless the DMA module has requested the bus when it is idle. The DMA module can be granted bus mastership only when the core is not asserting its bus request. See Figure 6-11.

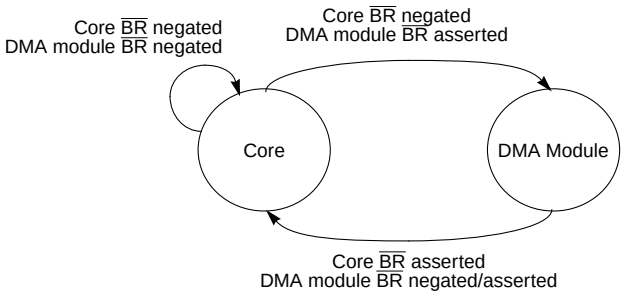

Figure 6-11. Park on Master Core Priority (PARK = 01)

- Park on master DMA priority (PARK = 10)—The DMA module retains bus mastership as long as it needs it. After it negates its internal bus request, the DMA module does not have to rearbitrate for the bus unless the core has requested the bus when it is idle. The core can be granted bus mastership only when the DMA module is not asserting its bus request. See Figure 6-12.

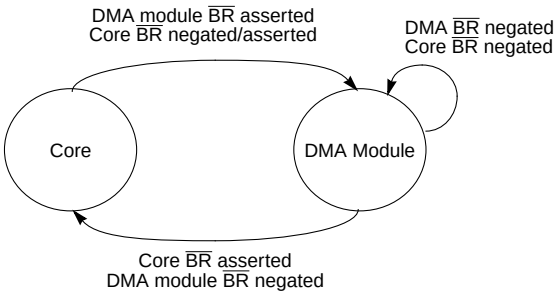

Figure 6-12. Park on DMA Module Priority (PARK = 10)

7.1.1 PLL:PCLK Ratios

The specifications for the clocks in the PLL module are summarized in Table 0-1.

Table 0-1. PLL Clock Specifications

Symbol	Description	Frequency		
—	PLL lock time	2.2 mS with CLKIN running at 45 MHz		
CLKIN	Input clock	16.67 MHz—45 MHz		
PCLK	Internal processor clock	33.34 MHz—90 MHz (CLKIN x 2)		
PSTCLK	Processor status clock	33.34 MHz—90 MHz (CLKIN x 2)		
BCLKO	Output clock	16.67 MHz—45 MHz	11.11 MHz—30 MHz	8.24 MHz—22.5 MHz
BCLKO/PCLK ratio		**1/2**	**1/3**	**1/4**

7.2 PLL Operation

The following sections provide detailed information about the three PLL modes.

7.2.1 Reset/Initialization

The PLL receives $\overline{\text{RSTI}}$ as an input directly from the pin. Additionally, signals are multiplexed with D[3:0]/FREQ[1:0]:DIVIDE[1:0] while $\overline{\text{RSTI}}$ is asserted. These signals are sampled during reset and registered by the PLL on the negation of $\overline{\text{RSTI}}$ to provide initialization information. FREQ[1:0] and DIVIDE[1:0] are used by the PLL to select the CLKIN frequency range and set the CLKIN/PCLK ratio, respectively.

7.2.2 Normal Mode

PCLK is divided to create the system bus clock, BCLKO. At reset, the logic level of DIVIDE[1:0]/D[1:0] determines the BCLKO divisor. The bus clock can be 1/2, 1/3, or 1/4 of the PCLK frequency.

7.2.3 Reduced-Power Mode

The PCLK can be turned off in a predictable manner to conserve system power. To allow fast restart of the MCF5307 processor core, the PLL continues to operate at the frequency configured at reset. PCLK is disabled using the CPU STOP instruction and resumes normal operation on interrupt, as described in Section 7.2.4, “PLL Control Register (PLLCR).”

8.5.4 I²C Status Register (I2SR)

This I2SR contains bits that indicate transaction direction and status.

	7	6	5	4	3	2	1	0
Field	ICF	IAAS	IBB	IAL	—	SRW	IIF	RXAK
Reset	1000_0001							
R/W	R			R/W	R		R/W	R
Address	MBAR + 0x28C							

Figure 8-8. I²CR Status Register (I2SR)

Table 8-5 describes I2SR fields.

Table 8-5. I2SR Field Descriptions

Bits	Name	Description
7	ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer in progress 1 Transfer complete. Set by the falling edge of the ninth clock of a byte transfer.
6	IAAS	I²C addressed as a slave bit. The CPU is interrupted if I2CR[IIEN] is set. Next, the CPU must check SRW and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed. 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5	IBB	I²C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4	IAL	Arbitration lost. Set by hardware in the following circumstances. (IAL must be cleared by software by writing zero to it.) • SDA sampled low when the master drives high during an address or data-transmit cycle. • SDA sampled low when the master drives high during the acknowledge bit of a data-receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it.
3	—	Reserved, should be cleared.
2	SRW	Slave read/write. When IAAS is set, SRW indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I²C module is a slave and has an address match. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.
1	IIF	I²C interrupt. Must be cleared by software by writing a zero to it in the interrupt routine. 0 No I²C interrupt pending 1 An interrupt is pending, which causes a processor interrupt request (if IIEN = 1). Set when one of the following occurs: • Complete one byte transfer (set at the falling edge of the ninth clock) • Reception of a calling address that matches its own specific address in slave-receive mode • Arbitration lost
0	RXAK	Received acknowledge. The value of SDA during the acknowledge bit of a bus cycle. 0 An acknowledge signal was received after the completion of 8-bit data transmission on the bus 1 No acknowledge signal was detected at the ninth clock.

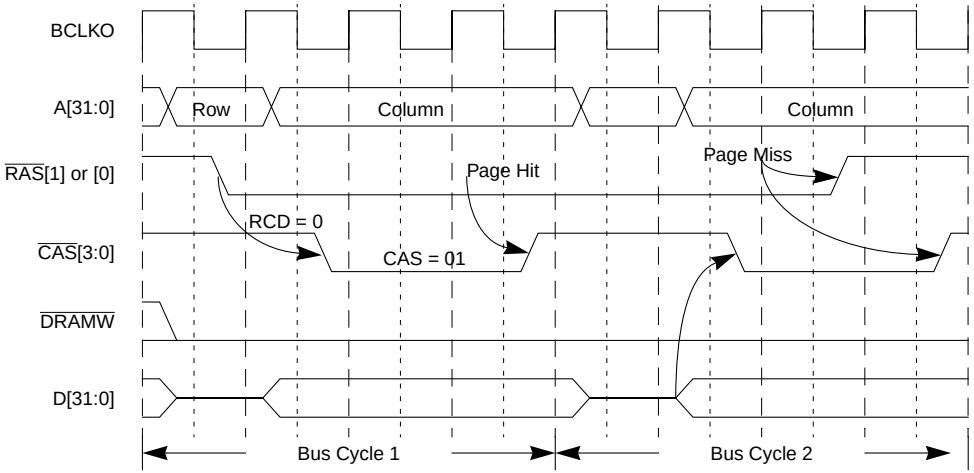

Figure 11-10. Write Hit in Continuous Page Mode

11.3.3.4 Extended Data Out (EDO) Operation

EDO is a variation of page mode that allows the DRAM to continue driving data out of the device while $\overline{CAS}$ is precharging. To support EDO DRAMs, the DRAM controller delays internal termination of the cycle by one clock so data can continue to be captured as $\overline{CAS}$ is being precharged. For data to be driven by the DRAMs, $\overline{RAS}$ is held after $\overline{CAS}$ is negated. EDO operation does not affect write operations. EDO DRAMs can be used in continuous page or burst page modes. Single accesses not followed by a hit in the page look like non-page-mode accesses.

Figure 11-11 shows four consecutive EDO accesses. Note that data is sampled after $\overline{CAS}$ is negated and that on the last page access, $\overline{CAS}$ is held until after data is sampled to assure that the data is driven. This allows $\overline{RAS}$ to be precharged before the end of the cycle.

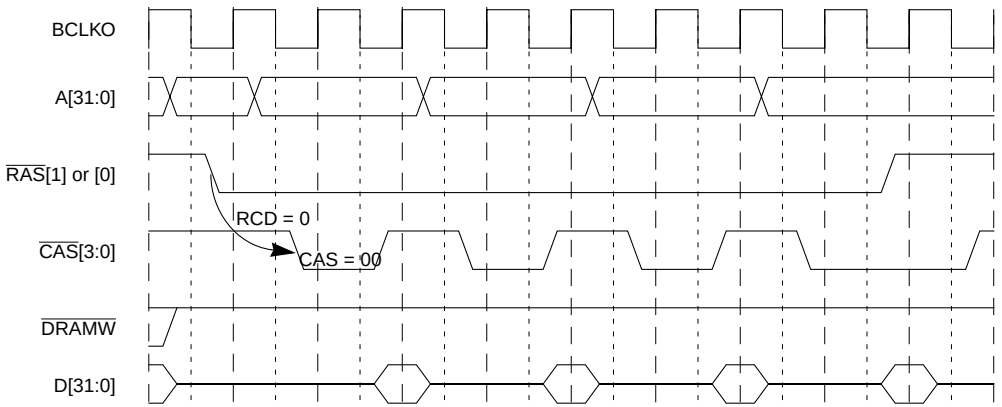

Figure 11-11. EDO Read Operation (3-2-2-2)

Table 11-12. DCR Field Descriptions (Synchronous Mode) (Continued)

Bits	Name	Description
12	COC	Command on SDRAM clock enable (SCKE). Implementations that use external multiplexing (NAM = 1) must support command information to be multiplexed onto the SDRAM address bus. 0 SCKE functions as a clock enable; self-refresh is initiated by the DRAM controller through DCR[IS]. 1 SCKE drives command information. Because SCKE is not a clock enable, self-refresh cannot be used (setting DCR[IS]). Thus, external logic must be used if this functionality is desired. External multiplexing is also responsible for putting the command information on the proper address bit.
11	IS	Initiate self-refresh command. 0 Take no action or issue a SELFX command to exit self refresh. 1 If DCR[COC] = 0, the DRAM controller sends a SELF command to both SDRAM blocks to put them in low-power, self-refresh state where they remain until IS is cleared, at which point the controller sends a SELFX command for the SDRAMs to exit self-refresh. The refresh counter is suspended while the SDRAMs are in self-refresh; the SDRAM controls the refresh period.
10–9	RTIM	Refresh timing. Determines the timing operation of auto-refresh in the DRAM controller. Specifically, it determines the number of clocks inserted between a REF command and the next possible ACTV command. This same timing is used for both memory blocks controlled by the DRAM controller. This corresponds to t_{RC} in the SDRAM specifications. 00 3 clocks 01 6 clocks 1x 9 clocks
8–0	RC	Refresh count. Controls refresh frequency. The number of bus clocks between refresh cycles is (RC + 1) * 16. Refresh can range from 16–8192 bus clocks to accommodate both standard and low-power DRAMs with bus clock operation from less than 2 MHz to greater than 50 MHz. The following example calculates RC for an auto-refresh period for 4096 rows to receive 64 mS of refresh every 15.625 µs for each row (625 bus clocks at 40 MHz). This operation is the same as in asynchronous mode. # of bus clocks = 625 = (RC field + 1) * 16 RC = (625 bus clocks/16) -1 = 38.06, which rounds to 38; therefore, RC = 0x26.

11.4.3.2 DRAM Address and Control Registers (DACR0/DACR1) in Synchronous Mode

The DRAM address and control registers (DACR0 and DACR1), shown in Figure 11-16, contain the base address compare value and the control bits for both memory blocks 0 and 1 of the DRAM controller. Address and timing are also controlled by bits in DACR*n*.

	31 … 18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	BA		—	RE	—	CASL		—	CBM			—	IMRS	PS		IP	PM	—	
Reset	Uninitialized			0	Uninitialized								0	Uninitialized					
R/W	R/W																		
Addr	MBAR+0x108 (DACR0); 0x110(DACR1)																		

Figure 11-16. DACR0 and DACR1 Registers (Synchronous Mode)

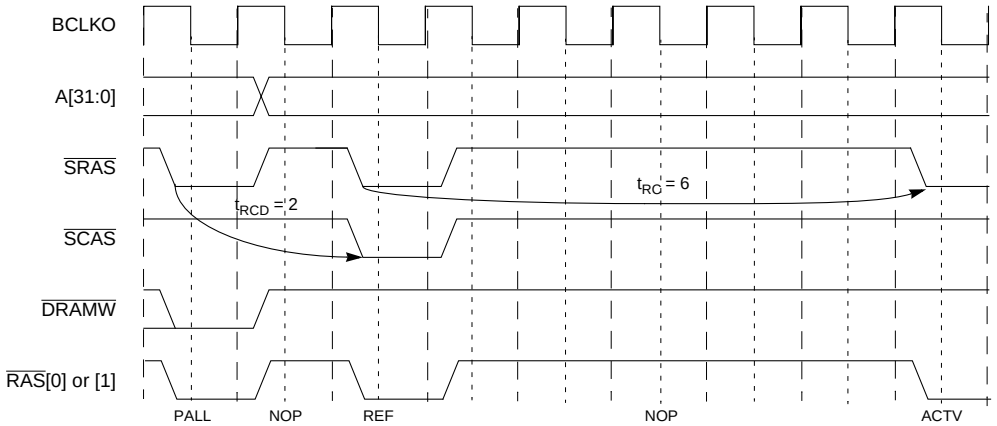

Figure 11-22. Auto-Refresh Operation

11.4.4.6 Self-Refresh Operation

Self-refresh is a method of allowing the SDRAM to enter into a low-power state, while at the same time to perform an internal refresh operation and to maintain the integrity of the data stored in the SDRAM. The DRAM controller supports self-refresh with DCR[IS]. When IS is set, the SELF command is sent to the SDRAM. When IS is cleared, the SELFX command is sent to the DRAM controller. Figure 11-23 shows the self-refresh operation.

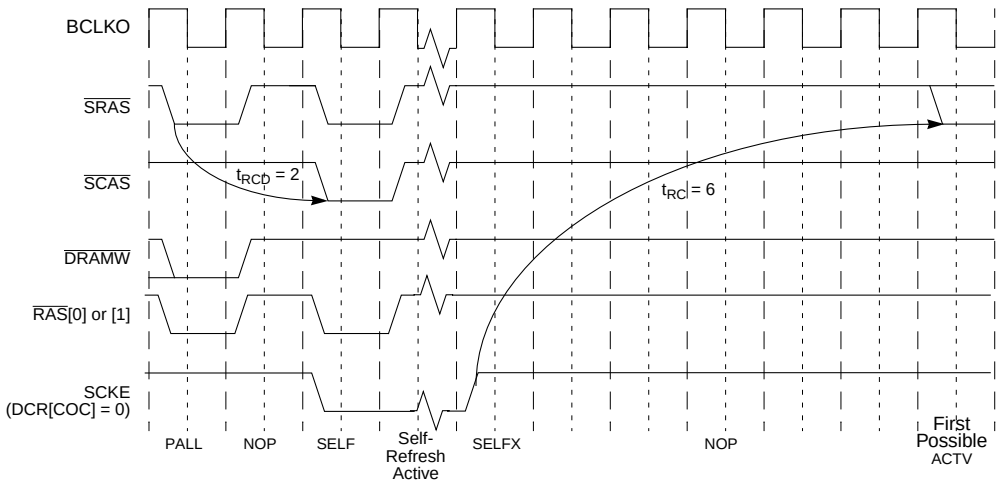

Figure 11-23. Self-Refresh Operation

Table 13-5. Calculated Time-out Values (90-MHz Processor Clock) (Continued)

TMR[PS]		TMR[CLK] = 10 (System Bus Clock/16)			TMR[CLK] = 01 (System Bus Clock/1)		
Decimal	Hex	45 MHz	30 MHz	22.5 MHz	45 MHz	30 MHz	22.5 MHz
166	A6	3.89138	5.83707	7.78276	0.24321	0.36482	0.48642
167	A7	3.91468	5.87203	7.82937	0.24467	0.367	0.48934
168	A8	3.93799	5.90698	7.87597	0.24612	0.36919	0.49225
169	A9	3.96129	5.94193	7.92257	0.24758	0.37137	0.49516
170	AA	3.98459	5.97688	7.96918	0.24904	0.37356	0.49807
171	AB	4.00789	6.01184	8.01578	0.25049	0.37574	0.50099
172	AC	4.03119	6.04679	8.06238	0.25195	0.37792	0.5039
173	AD	4.05449	6.08174	8.10899	0.25341	0.38011	0.50681
174	AE	4.0778	6.11669	8.15559	0.25486	0.38229	0.50972
175	AF	4.1011	6.15165	8.20219	0.25632	0.38448	0.51264
176	B0	4.1244	6.1866	8.2488	0.25777	0.38666	0.51555
177	B1	4.1477	6.22155	8.2954	0.25923	0.38885	0.51846
178	B2	4.171	6.2565	8.342	0.26069	0.39103	0.52138
179	B3	4.1943	6.29146	8.38861	0.26214	0.39322	0.52429
180	B4	4.21761	6.32641	8.43521	0.2636	0.3954	0.5272
181	B5	4.24091	6.36136	8.48181	0.26506	0.39759	0.53011
182	B6	4.26421	6.39631	8.52842	0.26651	0.39977	0.53303
183	B7	4.28751	6.43127	8.57502	0.26797	0.40195	0.53594
184	B8	4.31081	6.46622	8.62162	0.26943	0.40414	0.53885
185	B9	4.33411	6.50117	8.66823	0.27088	0.40632	0.54176
186	BA	4.35742	6.53612	8.71483	0.27234	0.40851	0.54468
187	BB	4.38072	6.57108	8.76144	0.27379	0.41069	0.54759
188	BC	4.40402	6.60603	8.80804	0.27525	0.41288	0.5505
189	BD	4.42732	6.64098	8.85464	0.27671	0.41506	0.55342
190	BE	4.45062	6.67593	8.90125	0.27816	0.41725	0.55633
191	BF	4.47392	6.71089	8.94785	0.27962	0.41943	0.55924
192	C0	4.49723	6.74584	8.99445	0.28108	0.42161	0.56215
193	C1	4.52053	6.78079	9.04106	0.28253	0.4238	0.56507
194	C2	4.54383	6.81574	9.08766	0.28399	0.42598	0.56798
195	C3	4.56713	6.8507	9.13426	0.28545	0.42817	0.57089
196	C4	4.59043	6.88565	9.18087	0.2869	0.43035	0.5738
197	C5	4.61373	6.9206	9.22747	0.28836	0.43254	0.57672
198	C6	4.63704	6.95555	9.27407	0.28981	0.43472	0.57963
199	C7	4.66034	6.99051	9.32068	0.29127	0.43691	0.58254
200	C8	4.68364	7.02546	9.36728	0.29273	0.43909	0.58545
201	C9	4.70694	7.06041	9.41388	0.29418	0.44128	0.58837
202	CA	4.73024	7.09536	9.46049	0.29564	0.44346	0.59128
203	CB	4.75354	7.13032	9.50709	0.2971	0.44564	0.59419
204	CC	4.77685	7.16527	9.55369	0.29855	0.44783	0.59711
205	CD	4.80015	7.20022	9.6003	0.30001	0.45001	0.60002

Chapter 14
UART Modules

This chapter describes the use of the universal asynchronous/synchronous receiver/transmitters (UARTs) implemented on the MCF5307 and includes programming examples. All references to UART refer to one of these modules.

14.1 Overview

The MCF5307 contains two independent UARTs. Each UART can be clocked by BCLKO, eliminating the need for an external crystal. As Figure 14-1 shows, each UART module interfaces directly to the CPU and consists of the following:

- Serial communication channel
- Programmable transmitter and receiver clock generation
- Internal channel control logic
- Interrupt control logic

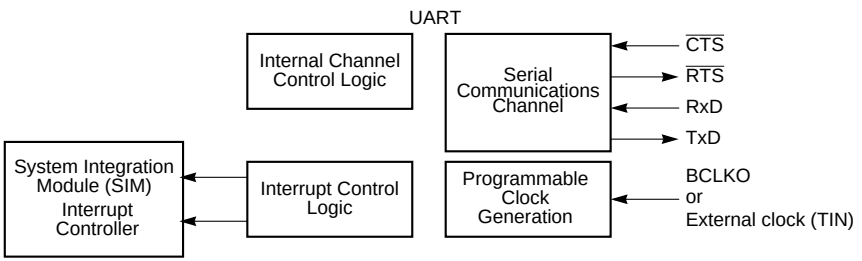

Figure 14-1. Simplified Block Diagram

The serial communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter deriving an operating frequency from BCLKO or an external clock using the timer pin. The transmitter converts parallel data from the CPU to a serial bit stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (TxD). See Section 14.5.2.1, “Transmitting.”

The receiver converts serial data from the channel receiver serial data input (RxD) to parallel format, checks for a start, stop, and parity bits, or break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be polled- or interrupt-driven. See Section 14.5.2.2, “Receiver.”

14.3.2 UART Mode Register 2 (UMR2*n*)

UART mode registers 2 (UMR2*n*) control UART module configuration. UMR2*n* can be read or written when the mode register pointer points to it, which occurs after any access to UMR1*n*. UMR2*n* accesses do not update the pointer.

	7	6	5	4	3	0
Field	CM		TxRTS	TxCTS	SB	
Reset	0000_0000					
R/W	R/W					
Address	MBAR + 0x1C0, 0x200. After UMR1n is read or written, the pointer points to UMR2n.					

Figure 14-3. UART Mode Register 2 (UMR2*n*)

Table 14-3 describes UMR2*n* fields.

Table 14-3. UMR2*n* Field Descriptions

Bits	Name	Description
7–6	CM	Channel mode. Selects a channel mode. Section 14.5.3, “Looping Modes,” describes individual modes. 00 Normal 01 Automatic echo 10 Local loop-back 11 Remote loop-back
5	TxRTS	Transmitter ready-to-send. Controls negation of $\overline{\text{RTS}}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{\text{RTS}}$ control is not permitted and disables $\overline{\text{RTS}}$ control for both. 0 The transmitter has no effect on $\overline{\text{RTS}}$. 1 In applications where the transmitter is disabled after transmission completes, setting this bit automatically clears UOP[RTS] one bit time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits.
4	TxCTS	Transmitter clear-to-send. If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. 0 $\overline{\text{CTS}}$ has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of $\overline{\text{CTS}}$ each time it is ready to send a character. If $\overline{\text{CTS}}$ is asserted, the character is sent; if it is negated, the channel TxD remains in the high state and transmission is delayed until $\overline{\text{CTS}}$ is asserted. Changes in $\overline{\text{CTS}}$ as a character is being sent do not affect its transmission.

Table 14-4. USR*n* Field Descriptions (Continued)

Bits	Name	Description
5	PE	Parity error. Valid only if RxRDY = 1. 0 No parity error occurred. 1 If UMR1n[PM] = 0x (with parity or force parity), the corresponding character in the FIFO was received with incorrect parity. If UMR1n[PM] = 11 (multidrop), PE stores the received A/D bit.
4	OE	Overrun error. Indicates whether an overrun occurs. 0 No overrun occurred. 1 One or more characters in the received data stream have been lost. OE is set upon receipt of a new character when the FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the receiver shift register and its break detect, framing error status, and parity error, if any, are lost. OE is cleared by the RESET ERROR STATUS command in UCRn.
3	TxEMP	Transmitter empty. 0 The transmitter buffer is not empty. Either a character is being shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming UCRn[TC]. 1 The transmitter has underrun (both the transmitter holding register and transmitter shift registers are empty). This bit is set after transmission of the last stop bit of a character if there are no characters in the transmitter holding register awaiting transmission.
2	TxRDY	Transmitter ready. 0 The CPU loaded the transmitter holding register or the transmitter is disabled. 1 The transmitter holding register is empty and ready for a character. TxRDY is set when a character is sent to the transmitter shift register and when the transmitter is first enabled. If the transmitter is disabled, characters loaded into the transmitter holding register are not sent.
1	FFULL	FIFO full. 0 The FIFO is not full but may hold up to two unread characters. 1 A character was received and is waiting in the receiver buffer FIFO.
0	RxRDY	Receiver ready 0 The CPU has read the receiver buffer and no characters remain in the FIFO after this read. 1 One or more characters were received and are waiting in the receiver buffer FIFO.

14.3.4 UART Clock-Select Registers (UCSR*n*)

The UART clock-select registers (UCSR*n*) select an external clock on the TIN input (divided by 1 or 16) or a prescaled BCLKO as the clocking source for the transmitter and receiver. See Section 14.5.1, “Transmitter/Receiver Clock Source.” The transmitter and receiver can use different clock sources. To use BCLKO for both, set UCSR*n* to 0xDD.

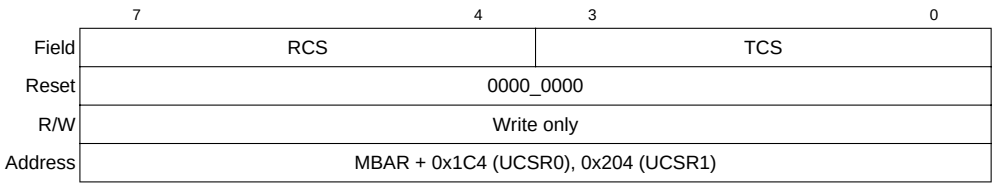

Figure 14-5. UART Clock-Select Register (UCSR*n*)

17.8.1 DMA Request ($\overline{\text{DREQ}}$[1:0]/PP[6:5])

The DMA request pins ($\overline{\text{DREQ}}$[1:0]/PP[6:5]) can serve as the DMA request inputs or as two bits of the parallel port, as determined by individually programmable bits in the PAR.

These inputs are asserted by a peripheral device to request an operand transfer between that peripheral and memory by either channel 0 or 1 of the on-chip DMA.

Note that DMA acknowledge indication is displayed on TM[2:0], during DMA transfers of channel 0 and 1.

17.9 Serial Module Signals

The signals in the following sections are used to transfer serial data between the two UART modules and external peripherals.

17.9.1 Transmitter Serial Data Output (TxD)

TxD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out least-significant bit (lsb) first on TxD on the falling edge of the clock source.

17.9.2 Receiver Serial Data Input (RxD)

Data received on RxD is sampled on the rising edge of the clock source, with the lsb received first.

17.9.3 Clear to Send ($\overline{\text{CTS}}$)

This input can generate an interrupt on a change of state.

17.9.4 Request to Send ($\overline{\text{RTS}}$)

This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter's $\overline{\text{CTS}}$, $\overline{\text{RTS}}$ can control serial data flow.

17.10 Timer Module Signals

The signals in the following sections are external interfaces to the two general-purpose MCF5307 timers. These 16-bit timers can capture timer values, trigger external events or internal interrupts, or count external events.

Chapter 20
Electrical Specifications

This chapter describes the AC and DC electrical specifications and thermal characteristics for the MCF5307. Note that this information was correct at the time this book was published. As process technologies improve, there is a likelihood that this information may change. To confirm that this is the latest information, see Motorola's ColdFire webpage, http://www.motorola.com/coldfire.

20.1 General Parameters

Table 20-1 lists maximum and minimum ratings for supply and operating voltages and storage temperature. Operating outside of these ranges may cause erratic behavior or damage to the processor.

Table 20-1. Absolute Maximum Ratings

Rating	Symbol	Value	Units
Supply voltage	V_{cc}	-0.3 to +4.0	V
Maximum operating voltage	V_{cc}	+3.6	V
Minimum operating voltage	V_{cc}	+3.0	V
Input voltage	V_{in}	-0.5 to +5.5	V
Storage temperature range	T_{stg}	-55 to +150	^{o}C

Table 20-2 lists junction and ambient operating temperatures.

Table 20-2. Operating Temperatures

Characteristic	Symbol	Value	Units
Maximum operating junction temperature	T_j	105	^{o}C
Maximum operating ambient temperature	T_{Amax}	70 [1]	^{o}C
Minimum operating ambient temperature	T_{Amin}	0	^{o}C

[1] This published maximum operating ambient temperature should be used only as a system design guideline. All device operating parameters are guaranteed only when the junction temperature lies within the specified range.

Table 20-3 lists DC electrical operating temperatures. This table is based on an operating

Table A-7. UART1 Module Programming Model

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x200	UART mode registers[1]—(UMR1n) [p. 14-4], (UMR2n) [p. 14-6]	—		
0x204	(Read) UART status registers—(USRn) [p. 14-7]	—		
	(Write) UART clock-select register[1]—(UCSRn) [p. 14-8]	—		
0x208	(Read) Do not access[2]	—		
	(Write) UART command registers—(UCRn) [p. 14-9]	—		
0x20C	(UART/Read) UART receiver buffers—(URBn) [p. 14-11]	—		
	(UART/Write) UART transmitter buffers—(UTBn) [p. 14-11]	—		
0x210	(Read) UART input port change registers—(UIPCRn) [p. 14-12]	—		
	(Write) UART auxiliary control registers[1]—(UACRn) [p. 14-12]	—		
0x214	(Read) UART interrupt status registers—(UISRn) [p. 14-13]	—		
	(Write) UART interrupt mask registers—(UIMRn) [p. 14-13]	—		
0x218	UART divider upper registers—(UDUn) [p. 14-14]	—		
0x21C	UART divider lower registers—(UDLn) [p. 14-14]	—		

Table A-7. UART1 Module Programming Model (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x220–0x22C	Do not access[2]	—		
0x230	UART interrupt vector register—(UIVRn) [p. 14-15]	—		
0x234	(Read) UART input port registers—(UIPn) [p. 14-15]	—		
	(Write) Do not access[2]	—		
0x238	(Read) Do not access[2]	—		
	(Write) UART output port bit set command registers—(UOP1n[3]) [p. 14-15]	—		
0x23C	(Read) Do not access[2]	—		
	(Write) UART output port bit reset command registers—(UOP0n[3]) [p. 14-15]	—		

1 UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

2 This address is for factory testing. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

3 Address-triggered commands

Table A-8. Parallel Port Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x244	Parallel port data direction register (PADDR) [p. 15-2]		Reserved	
0x248	Parallel port data register (PADAT) [p. 15-2]		Reserved	

Table A-9. I^2C Interface Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x280	I^2C address register (IADR) [p. 8-6]	Reserved		
0x284	I^2C frequency divider register (IFDR) [p. 8-7]	Reserved		