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 V4
Core Size	32-Bit Single-Core
Speed	162MHz
Connectivity	EBI/EMI, I²C, UART/USART
Peripherals	DMA, WDT
Number of I/O	16
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 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/pro/item?MUrl=&PartUrl=mcf5407cai162

ColdFire is a registered trademark and DigitalDNA is a trademark of Motorola, Inc.
I^2C is a registered trademark of Philips Semiconductors

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors

Document Comments: FAX (512) 895-2638, Attn: TECD Applications Engineering

1.2.1 Process

The MCF5407 is manufactured in a 0.22-μ CMOS process with quad-layer-metal routing technology. This process combines the high performance and low power needed for embedded system applications. Inputs are 3.3-V tolerant; outputs are CMOS or open-drain CMOS with outputs operating from VDD + 0.5 V to GND - 0.5 V, with guaranteed TTL-level specifications.

1.3 ColdFire Module Description

The following sections provide overviews of the various modules incorporated in the MCF5407.

1.3.1 ColdFire Core

The Version 4 ColdFire core consists of two independent and decoupled pipelines to maximize performance—the instruction fetch pipeline (IFP) and the operand execution pipeline (OEP).

1.3.1.1 Instruction Fetch Pipeline (IFP)

The four-stage instruction fetch pipeline (IFP) is designed to prefetch instructions for the operand execution pipeline (OEP). Because the fetch and execution pipelines are decoupled by a ten-instruction FIFO buffer, the fetch mechanism can prefetch instructions in advance of their use by the OEP, thereby minimizing the time stalled waiting for instructions. To maximize the performance of conditional branch instructions, the Version 4 IFP implements a sophisticated two-level acceleration mechanism.

The first level is an 8-entry, direct-mapped branch cache with a 2-bit prediction state (strongly/weakly, taken/not-taken) for each entry. The branch cache implements instruction folding techniques. These allow conditional branch instructions that are predicted correctly as taken to execute in zero cycles.

For those conditional branches with no information in the branch cache, a second-level, direct-mapped prediction table containing 128 entries is accessed. Again, each entry uses the same 2-bit prediction state definition as the branch cache. This branch prediction state is then used to predict the direction of prefetched conditional branch instructions.

Other change-of-flow instructions, including unconditional branches, jumps, and subroutine calls, use a similar mechanism where the IFP calculates the target address. The performance of subroutine return instructions is improved through the use of a four-entry, LIFO return stack.

In all cases, these mechanisms allow the IFP to redirect the fetch stream down the path predicted to be taken well in advance of the actual instruction execution. The result is significantly improved performance.

1.3.1.2 Operand Execution Pipeline (OEP)

The prefetched instruction stream is gated from the FIFO buffer into the five-stage OEP. The OEP consists of two, traditional two-stage RISC compute engines with a register file access feeding an arithmetic/logic unit (ALU). The compute engine located at the top of the OEP is typically used for operand memory address calculations (the address ALU), while the compute engine located at the bottom of the pipeline is used for instruction execution (the execution ALU). The resulting structure provides 3.9 Gbytes/S data operand bandwidth at 162 MHz to the two compute engines and supports single-cycle execution speeds for most instructions, including all load, store, and most embedded-load operations. In response to users and developers' comments, the V4 design supports execution of the ColdFire Revision B instruction set, which adds a small number of new instructions to improve performance and code density.

The OEP also implements two advanced performance features. It dynamically determines the appropriate location of instruction execution (either in the address ALU or the execution ALU) based on the pipeline state. The address compute engine, in conjunction with register renaming resources, can be used to execute a number of heavily-used opcodes and forward the results to subsequent instructions without any pipeline stalls. Additionally, the OEP implements instruction folding techniques involving MOVE instructions so that two instructions can be issued in a single machine cycle. The resulting microarchitecture approaches the performance of a full superscalar implementation, but at a much lower silicon cost.

1.3.1.3 MAC Module

The MAC unit provides signal processing capabilities for the MCF5407 in a variety of applications including digital audio and servo control. Integrated as an execution unit in the processor's OEP, the MAC unit implements a three-stage arithmetic pipeline optimized for 16 x 16 multiplies. Both 16- and 32-bit input operands are supported by this design in addition to a full set of extensions for signed and unsigned integers, plus signed, fixed-point fractional input operands.

1.3.1.4 Integer Divide Module

Integrated into the OEP, the divide module performs operations using signed and unsigned integers. The module supports word and longword divides producing quotients and/or remainders.

1.3.2 Harvard Architecture

A Harvard memory architecture implements separate instruction and data buses to the processor-local memories, removing conflicts between instruction fetches and operand accesses.

or 32-bit ports. The base address, access permissions, and internal bus transfer terminations are programmable with configuration registers for each chip select. $\overline{CS0}$ also provides global chip select functionality of boot ROM upon reset for initializing the MCF5407.

1.3.8.3 16-Bit Parallel Port Interface

A 16-bit general-purpose programmable parallel port serves as either an input or an output on a pin-by-pin basis.

1.3.8.4 Interrupt Controller

The interrupt controller provides user-programmable control of ten internal peripheral interrupts and implements four external fixed interrupt-request pins. Each internal interrupt can be programmed to any one of seven interrupt levels and four priority levels within each of these levels. Additionally, the external interrupt request pins can be mapped to levels 1, 3, 5, and 7 or levels 2, 4, 6, and 7. Autovector capability is available for both internal and external interrupts.

1.3.8.5 JTAG

To help with system diagnostics and manufacturing testing, the MCF5407 processor includes dedicated user-accessible test logic that complies with the IEEE 1149.1a standard for boundary-scan testability, often referred to as the Joint Test Action Group, or JTAG. For more information, refer to the IEEE 1149.1a standard.

1.3.9 System Debug Interface

The ColdFire processor core debug interface is provided to support system debugging in conjunction with low-cost debug and emulator development tools. Through a standard debug interface, users can access real-time trace and debug information. This allows the processor and system to be debugged at full speed without the need for costly in-circuit emulators. The debug unit in the MCF5407 is a compatible upgrade to the MCF52xx and MCF53xx debug modules with added breakpoint registers and support for I/O interrupt request servicing while in emulator mode.

The on-chip breakpoint resources include a total of 13 programmable registers—two sets of address registers (each with two 32-bit registers), two sets of data registers (each with a 32-bit data register plus a 32-bit data mask register), one 32-bit PC register plus a 32-bit PC mask register and three additional 32-bit PC registers. These registers can be accessed through the dedicated debug serial communication channel or from the processor's supervisor mode programming model. The breakpoint registers can be configured to generate triggers by combining the address, data, and PC conditions in a variety of single or dual-level definitions. The trigger event can be programmed to generate a processor halt or initiate a debug interrupt exception.

The MCF5407's new interrupt servicing options during emulator mode allow real-time critical interrupt service routines to be serviced while processing a debug interrupt event, thereby ensuring that the system continues to operate even during debugging.

Table I-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiple accumulate unit
MBAR	Memory base address register
MSB	Most-significant byte
msb	Most-significant bit
Mux	Multiplex
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PCLK	Processor clock
PLL	Phase-locked loop
PLRU	Pseudo least recently used
POR	Power-on reset
PQFP	Plastic quad flat pack
RISC	Reduced instruction set computing
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Table 2-6. Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	index register i (can be an address or data register: Ai, Di)
Register Names	
ACC	MAC accumulator register
CCR	Condition code register (lower byte of SR)
MACSR	MAC status register
MASK	MAC mask register
PC	Program counter
SR	Status register
Port Name	
PSTDDATA	Processor status/debug data port
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
<ea>	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Both instruction and data caches
dc	Data cache
ic	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
<>	identifies an indirect data address referencing memory

4.8.2 The Cache at Start-Up

As Figure 4-4 (A) shows, after power-up, cache contents are undefined; V and M may be set on some lines even though the cache may not contain the appropriate data for start up. Because reset and power-up do not invalidate cache lines automatically, the cache should be cleared explicitly by setting CACR[DCINVA,ICINVA] before the cache is enabled (B).

After the entire cache is flushed, cacheable entries are loaded first in way 0. If way 0 is occupied, the cacheable entry is loaded into the same set in way 1, as shown in Figure 4-4 (D). This process is described in detail in Section 4.9, “Cache Operation.”

4.10 Cache Registers

This section describes the MCF5407 implementation of the Version 4 cache registers.

4.10.1 Cache Control Register (CACR)

The CACR in Figure 4-8 contains bits for configuring the cache. It can be written by the MOVEC register instruction and can be read or written from the debug facility. A hardware reset clears CACR, which disables the cache; however, reset does not affect the tags, state information, or data in the cache.

	31	30	29	28	27	26	25	24	23	20	19	18	17	16
Field	DEC	DW	DESB	DDPI	DHLCK	DDCM		DCINVA	—		BEC	BCINVA	—	
Reset	0000_0000_0000_0000													
R/W	Write (R/W by debug module)													

	15	14	13	12	11	10	9	8	7	0
Field	IEC	—	DNFB	IDPI	IHLCK	IDCM	—	ICINVA	—	
Reset	0000_0000_0000_0000									
R/W	Write (R/W by debug module)									
Rc	0x002									

Figure 4-8. Cache Control Register (CACR)

Table 4-4 describes CACR fields.

Table 4-4. CACR Field Descriptions

Bits	Name	Description
31	DEC	Enable data cache. 0 Cache disabled. The data cache is not operational, but data and tags are preserved. 1 Cache enabled.
30	DW	Data default write-protect. For normal operations that do not hit in the RAMBARs or ACRs, this field defines write-protection. See Section 4.9.1, “Caching Modes.” 0 Not write protected. 1 Write protected. Write operations cause an access error exception.
29	DESB	Enable data store buffer. Affects the precision of transfers. CACR[DESB] has precedence over CACR[9–8] and ACRn[9–8]; therefore, the store buffer must be disabled to use imprecise mode. 0 Imprecise-mode, write-through or cache-inhibited writes bypass the store buffer and generate bus cycles directly. Section 4.9.5.2.1, “Push and Store Buffers,” describes the associated performance penalty. 1 The four-entry FIFO store buffer is enabled; to maximize performance, this buffer defers pending imprecise-mode, write-through or cache-inhibited writes. Precise-mode, cache-inhibited accesses always bypass the store buffer. Precise and imprecise modes are described in Section 4.9.2, “Cache-Inhibited Accesses.”
28	DDPI	Disable CPUSHL invalidation. 0 Normal operation. A CPUSHL instruction causes the selected line to be pushed if modified and then invalidated. 1 No clear operation. A CPUSHL instruction causes the selected line to be pushed if modified, then left valid.

```
        addq.l          #1,d0           ;increment to next way
        move.l          d0,a0           ;set = 0, way = d0
        cmpi.l          #4,d0           ;flushed all the ways?
        bne             setloop
        rts
```

The following CACR loads assume the instruction cache has been invalidated, the default instruction cache mode is cacheable, and the default data cache mode is copyback.

```
dataCacheLoadAndLock:

        move.l  #0xa3080800,d0; enable and invalidate data cache ...
        movec   d0,cacr ; ... in the CACR
```

The following code preloads half of the data cache (4 Kbytes). It assumes a contiguous block of data is to be mapped into the data cache, starting at a 0-modulo-4K address.

```
        move.l  #256,d0         ;256 16-byte lines in 4K space
        lea     data_,a0        ; load pointer defining data area
dataCacheLoop:
        tst.b   (a0)            ;touch location + load into data cache
        lea     16(a0),a0       ;increment address to next line
        subq.l  #1,d0           ;decrement loop counter
        bne.b   dataCacheLoop   ;if done, then exit, else continue

; A 4K region has been loaded into levels 0 and 1 of the 8K data cache. lock it!

        move.l  #0xaa088000,d0  ;set the data cache lock bit ...
        movec   d0,cacr         ; ... in the CACR
        rts

        align   16
```

The following CACR loads assume the data cache has been invalidated, the default instruction cache mode is cacheable and the default operand cache mode is copyback.

Note that this function must be mapped into a cache inhibited or SRAM space or these text lines will be prefetched into the instruction cache, which may displace some of the 8-Kbyte space being explicitly fetched.

```
instructionCacheLoadAndLock:

        move.l  #0xa2088100,d0  ;enable and invalidate the instruction
        movec   d0,cacr         ;cache in the CACR
```

The following code segments preload half of the instruction cache (8 Kbytes). It assumes a contiguous block of data is to be mapped, starting at a 0-modulo-8K address

```
        move.l  #512,d0         ;512 16-byte lines in 8K space
        lea     code_,a0        ;load pointer defining code area
instCacheLoop:
;       intouch (a0)            ;touch location + load into instruction cache


; Note in the assembler we use, there is no INTOUCH opcode. The following
; is used to produce the required binary representation

        cpushl  #nc,(a0)        ;touch location + load into
```

5.5.3.3.4 Write Memory Location (WRITE)

Write data to the memory location specified by the longword address. The address space is defined by BAAR[TT,TM]. Hardware forces low-order address bits to zeros for word and longword accesses to ensure that word addresses are word-aligned and longword addresses are longword-aligned.

Command Formats:

	15 12	11 8	7 4	3 1
Byte	0x1	0x8	0x0	0x0
	A[31:16]			
	A[15:0]			
	X X X X	X X X X	D[7:0]	
Word	0x1	0x8	0x4	0x0
	A[31:16]			
	A[15:0]			
	D[15:0]			
Longword	0x1	0x8	0x8	0x0
	A[31:16]			
	A[15:0]			
	D[31:16]			
	D[15:0]			

Figure 5-26. WRITE Command Format

5.5.3.3.5 Dump Memory Block (DUMP)

DUMP is used with the READ command to access large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. If an initial READ is not executed before the first DUMP, an illegal command response is returned. The DUMP command retrieves subsequent operands. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP commands use this address, perform the memory read, increment it by the current operand size, and store the updated address in the temporary register.

NOTE:

DUMP does not check for a valid address; it is a valid command only when preceded by NOP, READ, or another DUMP command. Otherwise, an illegal command response is returned. NOP can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a DUMP command is processed, allowing the operand size to be dynamically altered.

Command/Result Formats:

		15 12	11 8	7 4	3 0
Byte	Command	0x1	0xD	0x0	0x0
	Result	X X X X	X X X X	D[7:0]	
Word	Command	0x1	0xD	0x4	0x0
	Result	D[15:0]			
Longword	Command	0x1	0xD	0x8	0x0
	Result	D[31:16]			
		D[15:0]			

Figure 5-28. DUMP Command/Result Formats

Table II-i. Acronyms and Abbreviated Terms (Continued)

Term	Meaning
NOP	No operation
PCLK	Processor clock
PLL	Phase-locked loop
POR	Power-on reset
Rx	Receive
SIM	System integration module
SOF	Start of frame
TAP	Test access port
TTL	Transistor-to-transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Table 11-13 describes DACR*n* fields.

Table 11-13. DACR0/DACR1 Field Descriptions (Synchronous Mode)

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>31–18</td><td>BA</td><td>Base address register. With DCMR[BAM], determines the address range in which the associated DRAM block is located. Each BA bit is compared with the corresponding address of the current bus cycle. If all unmasked bits match, the address hits in the associated DRAM block. BA functions the same as in asynchronous operation.</td></tr>
<tr><td>17–16</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>15</td><td>RE</td><td>Refresh enable. Determines when the DRAM controller generates a refresh cycle to the DRAM block.
0 Do not refresh associated DRAM block
1 Refresh associated DRAM block</td></tr>
<tr><td>14</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>13–12</td><td>CASL</td><td>$\overline{\text{CAS}}$ latency. Affects the following SDRAM timing specifications. Timing nomenclature varies with manufacturers. Refer to the SDRAM specification for the appropriate timing nomenclature:
<table>
<tr><th rowspan="2">Parameter</th><th colspan="4">Number of Bus Clocks</th></tr>
<tr><th>CASL= 00</th><th>CASL = 01</th><th>CASL= 10</th><th>CASL= 11</th></tr>
<tr><td>t_{RCD}—$\overline{\text{SRAS}}$ assertion to $\overline{\text{SCAS}}$ assertion</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{CASL}—$\overline{\text{SCAS}}$ assertion to data out</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{RAS}—ACTV command to precharge command</td><td>2</td><td>4</td><td>6</td><td>6</td></tr>
<tr><td>t_{RP}—Precharge command to ACTV command</td><td>1</td><td>2</td><td>3</td><td>3</td></tr>
<tr><td>t_{RWL},t_{RDL}—Last data input to precharge command</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>t_{EP}—Last data out to precharge command)</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>11</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>10–8</td><td>CBM</td><td>Command and bank MUX [2:0]. Because different SDRAM configurations cause the command and bank select lines to correspond to different addresses, these resources are programmable. CBM determines the addresses onto which these functions are multiplexed.
CBM Command Bit Bank Select Bits
000 17 18 and up
001 18 19 and up
010 19 20 and up
011 20 21 and up
100 21 22 and up
101 22 23 and up
110 23 24 and up
111 24 25 and up
This encoding and the address multiplexing scheme handle common SDRAM organizations. Bank select bits include a base bit and all address bits above for SDRAMs with multiple bank select bits.</td></tr>
<tr><td>7</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

Chapter 12
DMA Controller Module

This chapter describes the MCF5407 DMA controller module. It provides an overview of the module and describes in detail its signals and registers. The latter sections of this chapter describe operations, features, and supported data transfer modes in detail.

12.1 Overview

The direct memory access (DMA) controller module provides an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in Figure 12-1, provides four channels that allow byte, word, or longword operand transfers. Each channel has a dedicated set of registers that define the source and destination addresses (SAR*n* and DAR*n*), byte count (BCR*n*), and control and status (DCR*n* and DSR*n*). Transfers can be dual or single address to off-chip devices or dual address to on-chip devices, such as UART, SDRAM controller, and parallel port.

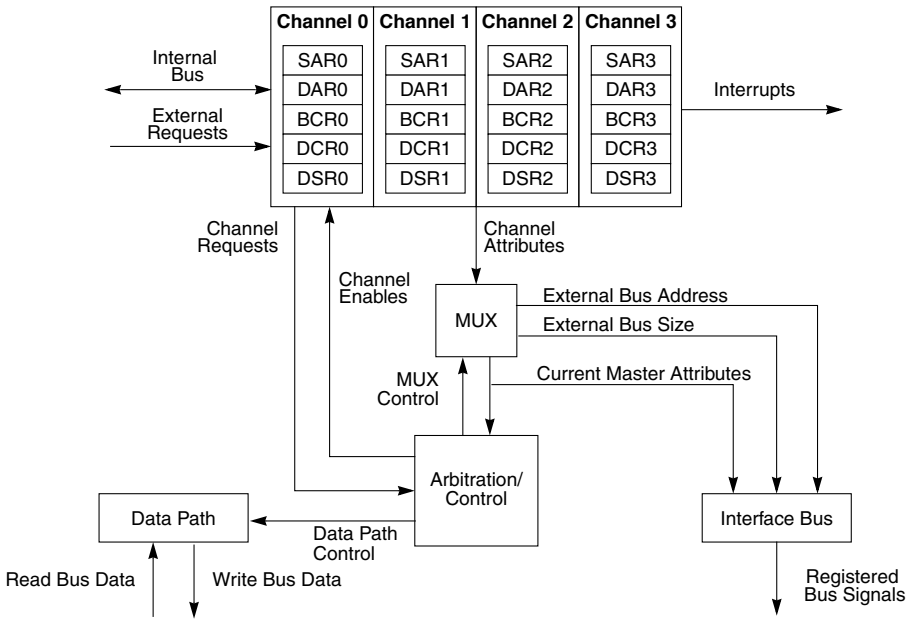

Figure 12-1. DMA Signal Diagram

12.1.1 DMA Module Features

The DMA controller module features are as follows:

- Four fully independent, programmable DMA controller channels/bus modules
- Auto-alignment feature for source or destination accesses
- Dual- and single-address transfers
- Two external request pins ($\overline{\text{DREQ}}$[1:0]) provided for channels 1 and 0
- Two external acknowledge pins (DACK[1:0]) provided for channels 1 and 0
- Channels 2 and 3 have request signals connected to the interrupt lines of UART0 and UART1, programmable through the channel select field MODCTL[DSL]. See Section 14.3.4, “Modem Control Register (MODCTL).”
- Channel arbitration on transfer boundaries
- Data transfers in 8-, 16-, 32-, or 128-bit blocks using a 16-byte buffer
- Continuous-mode and cycle-steal transfers
- Independent transfer widths for source and destination
- Independent source and destination address registers
- Data transfer can occur in as few as two clocks

12.2 DMA Signal Description

Table 12-1 briefly describes the DMA module signals that provide handshake control for either a source or destination external device.

Table 12-1. DMA Signals

Signal	I/O	Description
$\overline{\text{DREQ}}$[1:0]/ PP[6:5]	I	External DMA request. $\overline{\text{DREQ}}$[1:0] can serve as the DMA request inputs or as two parallel port bits. They are programmable individually through the PAR. A peripheral device asserts these inputs to request an operand transfer between it and memory. $\overline{\text{DREQ}}$ signals are asserted to initiate DMA accesses in the respective channels. The system should drive unused $\overline{\text{DREQ}}$ signals to logic high. Although each channel has an individual $\overline{\text{DREQ}}$ signal, in the MCF5407 only channels 0 and 1 connect to external $\overline{\text{DREQ}}$ pins. $\overline{\text{DREQ2}}$ and $\overline{\text{DREQ3}}$ are programmable for use with UART0 and UART1 through MODCTL[DSL]. See Section 14.3.4, “Modem Control Register (MODCTL).”
TT[1:0]/ PP[1:0]	O	Transfer type. A DMA access is indicated by the transfer type pins, TT[1:0] = 01. The transfer modifier, TM[2:0], and DMA acknowledgement, DACK[1:0], configurations shown below are meaningful only if TT[1:0] = 01, indicating an external master or DMA access.

```
    move.w TMR0,D0;save the contents of TMR0 while setting
    bset #0,D0 ;the 0 bit. This enables timer 0 and starts counting
    move.w D0, TMR0 ;load the value back into the register, setting TMR0[RST]

T0_LOOP

    move.b TER0,D1 ;load TER0 and see if
    btst #1,D1 ;TER0[REF] has been set
    beq T0_LOOP

    addi.l #1,D2;Increment D2
    cmp.l #5,D2;Did D2 reach 5? (i.e. timer ref has timed)
    beq T0_FINISH;If so, end timer0 example. Otherwise jump back.

    move.b #0x02,D0 ;writing one to TER0[REF] clears the event flag
    move.b D0,TER0
    jmp T0_LOOP

T0_FINISH
    HALT;End processing. Example is finished
```

13.5 Calculating Time-Out Values

The formula below determines time-out periods for various reference values:

Time-out period = (1/clock frequency) x (1 or 16) x (TMR*n*[PS] + 1) x (TRR*n*[REF])

When calculating time-out periods, add 1 to the prescaler to simplify calculating, because TMR*n*[PS] = 0x00 yields a prescaler of 1 and TMR*n*[PS] = 0xFF yields a prescaler of 256. For example, if a 54-MHz timer clock is divided by 16, TMR*n*[PS] = 0x7F, and the timer is referenced at 0xABCD (43,981 decimal), the time-out period is as follows:

Time-out period = (1/54,000,000) x (16) x (127 + 1) x (43,981) = 1.67 S

The time-out values in Table 13-4 represent the time it takes the counter value in TCN*n* value to go from 0x0000 to the default reference value, TRR*n*[REF] = 0xFFFF. Time-out values shown for CLKIN are divided by 1 and by 16 (TMR*n*[CLK] is 01 or 10, respectively).

Any clock source (CLKIN ÷ 1, CLKIN ÷ 16, or TIN) can be prescaled using TMR*n*[PS].

Table 13-4. Time-Out Values (in Seconds)—TRR[REF] = 0xFFFF (162-MHz Processor Clock)

TMR[PS] (Dec)	CLK = 10 (÷ 1)			CLK = 01 (÷ 16)		
	CLKIN (MHz)					
	54	40.5	32.4	54	40.5	32.4
0	0.019	0.026	0.032	0.001	0.002	0.002
1	0.039	0.052	0.065	0.002	0.003	0.004
2	0.058	0.078	0.097	0.004	0.005	0.006
3	0.078	0.104	0.129	0.005	0.006	0.008
4	0.097	0.129	0.162	0.006	0.008	0.010
5	0.117	0.155	0.194	0.007	0.010	0.012

TMR[PS] (Dec)	CLK = 10 (÷ 1)			CLK = 01 (÷ 16)		
	CLKIN (MHz)					
	54	40.5	32.4	54	40.5	32.4
128	2.505	3.340	4.175	0.157	0.209	0.261
129	2.524	3.366	4.207	0.158	0.210	0.263
130	2.544	3.392	4.240	0.159	0.212	0.265
131	2.563	3.418	4.272	0.160	0.214	0.267
132	2.583	3.443	4.304	0.161	0.215	0.269
133	2.602	3.469	4.337	0.163	0.217	0.271

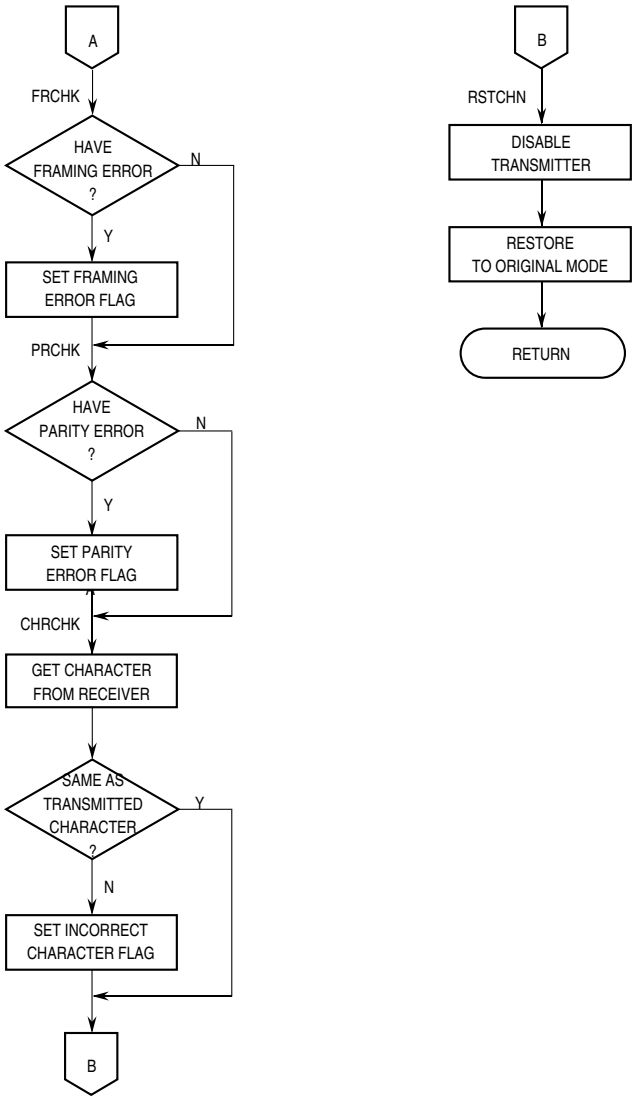

Figure 14-39. UART Mode Programming Flowchart (Sheet 3 of 5)

Chapter 15
Parallel Port (General-Purpose I/O)

This chapter describes the operation and programming model of the parallel port pin assignment, direction-control, and data registers. It includes a code example for setting up the parallel port.

15.1 Parallel Port Operation

The MCF5407 parallel port module has 16 signals, which are programmed as follows:

- The pin assignment register (PAR) selects the function of the 16 multiplexed pins.
- Port A data direction register (PADDR) determines whether pins configured as parallel port signals are inputs or outputs.
- The Port A data register (PADAT) shows the status of the parallel port signals.

The operations of the PAR, PADDR, and PADAT are described in the following sections.

15.1.1 Pin Assignment Register (PAR)

The pin assignment register (PAR), which is part of the system integration module (SIM), defines how each PAR bit determines each pin function, as shown in Figure 15-1.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0
PAR[n] = 0	PP15	PP14	PP13	PP12	PP11	PP10	PP9	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
PAR[n] = 1	A31	A30	A29	A28	A27	A26	A25	A24	$\overline{\text{TIP}}$	$\overline{\text{DREQ0}}$	$\overline{\text{DREQ1}}$	TM2	TM1/ DACK1	TM0/ DACK0	TT1	TT0
Reset	Determined by driving D4/ADDR_CONFIG with a 1 or 0 when $\overline{\text{RSTI}}$ negates. The system is configured as PP[15:0] if D4 is low; otherwise alternate pin functions selected by PAR[n] = 1 are used.															
R/W	R/W															
Address	Address MBAR + 0x004															

Figure 15-1. Parallel Port Pin Assignment Register (PAR)

If PP[9:8]/A[25:24] are unavailable because A[25:0] are needed for external addressing, PP[15:10]/A[31:26] can be configured as general-purpose I/O. Table 15-1 summarizes MCF5407 parallel port pins, described in detail in Chapter 17, “Signal Descriptions.”

17.5.5 Data/Configuration Pins (D[7:0])

This section describes data pins, D[7:0], that are read at reset for configuration. Table 17-11 shows pin assignments.

Table 17-11. Data Pin Configuration

Pin	Function
D7	Auto-acknowledge configuration (AA_CONFIG)
D[6:5]	Port size configuration (PS_CONFIG[1:0])
D4	Address configuration (ADDR_CONFIG/D4)
D3	Byte enable configuration (BE_CONFIG)
D[2:0]	Divide control (DIVIDE[2:0])

17.5.5.1 D[7:5,3]—Boot Chip-Select ($\overline{CS0}$) Configuration

D[7:5,3] determine defaults for the global chip select ($\overline{CS0}$), the only chip select valid at reset. These signals correspond to bits in chip-select configuration register 0 (CSCR0).

17.5.5.2 D7—Auto Acknowledge Configuration (AA_CONFIG)

At reset, the enabling and disabling of auto acknowledge for boot $\overline{CS0}$ is determined by the logic level driven on D7 at the rising edge of $\overline{RSTI}$. AA_CONFIG is multiplexed with D7 and sampled only at reset. The D7 logic level is reflected as the reset value of CSCR[AA]. Table 17-12 shows how the D7 logic level corresponds to the auto acknowledge timing for $\overline{CS0}$ at reset. Note that auto acknowledge can be disabled by driving a logic 0 on D7 at reset.

Table 17-12. D7 Selection of $\overline{CS0}$ Automatic Acknowledge

D7 (CSCR0[AA])	Boot $\overline{CS0}$ AA
0	Disabled
1	Enabled with 15 wait states

17.5.5.3 D[6:5]—Port Size Configuration (PS_CONFIG[1:0])

The default port size value of the boot $\overline{CS0}$ is determined by the logic levels driven on D[6:5] at the rising edge of $\overline{RSTI}$, which are reflected as the reset value of CSCR[PS]. Table 17-13 shows how the logic levels of D[6:5] correspond to the $\overline{CS0}$ port size at reset.

Table 17-13. D6 and D5 Selection of $\overline{CS0}$ Port Size

D[6:5] (CSCR0[PS])	Boot $\overline{CS0}$ Port Size
00	32-bit port
01	8-bit port
1x	16-bit port

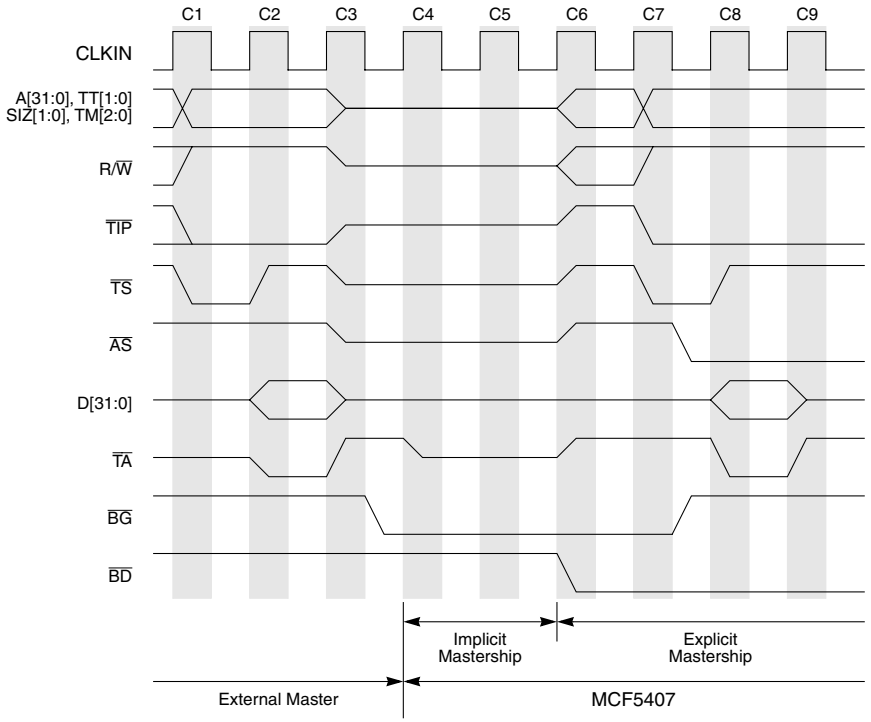

Figure 18-28. Two-Wire Implicit and Explicit Bus Mastership

In Figure 18-28, the external device is master during C1 and C2. It releases bus control in C3 by asserting $\overline{BG}$ to the MCF5407. During C4 and C5, the MCF5407 is implicit master because no internal access is pending. In C5, an internal bus request becomes pending, causing the MCF5407 to become explicit bus master in C6 by asserting $\overline{BD}$. In C7, the external device removes the bus grant to the MCF5407. The MCF5407 does not release the bus (the MCF5407 continues to assert $\overline{BD}$) until the transfer ends.

NOTE:

The MCF5407 can start a transfer in the clock cycle after $\overline{BG}$ is asserted. The external master must not assert $\overline{BG}$ to the MCF5407 while driving the bus or the part may be damaged.

Figure 18-29 is a MCF5407 bus arbitration state diagram. States are described in Table 18-6.