Advance Information

MCF5407PB/D
Rev. 3.3, 2/2003

MCF5407 Integrated ColdFire® Microprocessor Product Brief

MOTOROLA intelligence everywhere™ | digital dna™

This document is an overview of the MCF5407 ColdFire processor, focusing on feature enhancements over the MCF5307. It includes general descriptions of features and of the various modules incorporated in the MCF5407. It describes the V4 programming model as it is implemented in the MCF5407.

1.1 Features

The MCF5407 integrated microprocessor combines a Version 4 ColdFire processor core with the following components, as shown in Figure 1:

- Harvard architecture memory system with 16-Kbyte instruction cache and 8-Kbyte data cache
- Two, 2-Kbyte on-chip SRAMs
- Integer/fractional multiply-accumulate (MAC) unit
- Divide unit
- System debug interface
- DRAM controller for synchronous and asynchronous DRAM
- Four-channel DMA controller
- Two general-purpose timers
- Two UARTs, one that supports synchronous operations
- I^2C™ interface
- Parallel I/O interface
- System integration module (SIM)

Designed for embedded control applications, the MCF5407 delivers 316 Dhrystone MIPS at 220 MHz or 233 Dhrystone MIPS at 162 MHz.

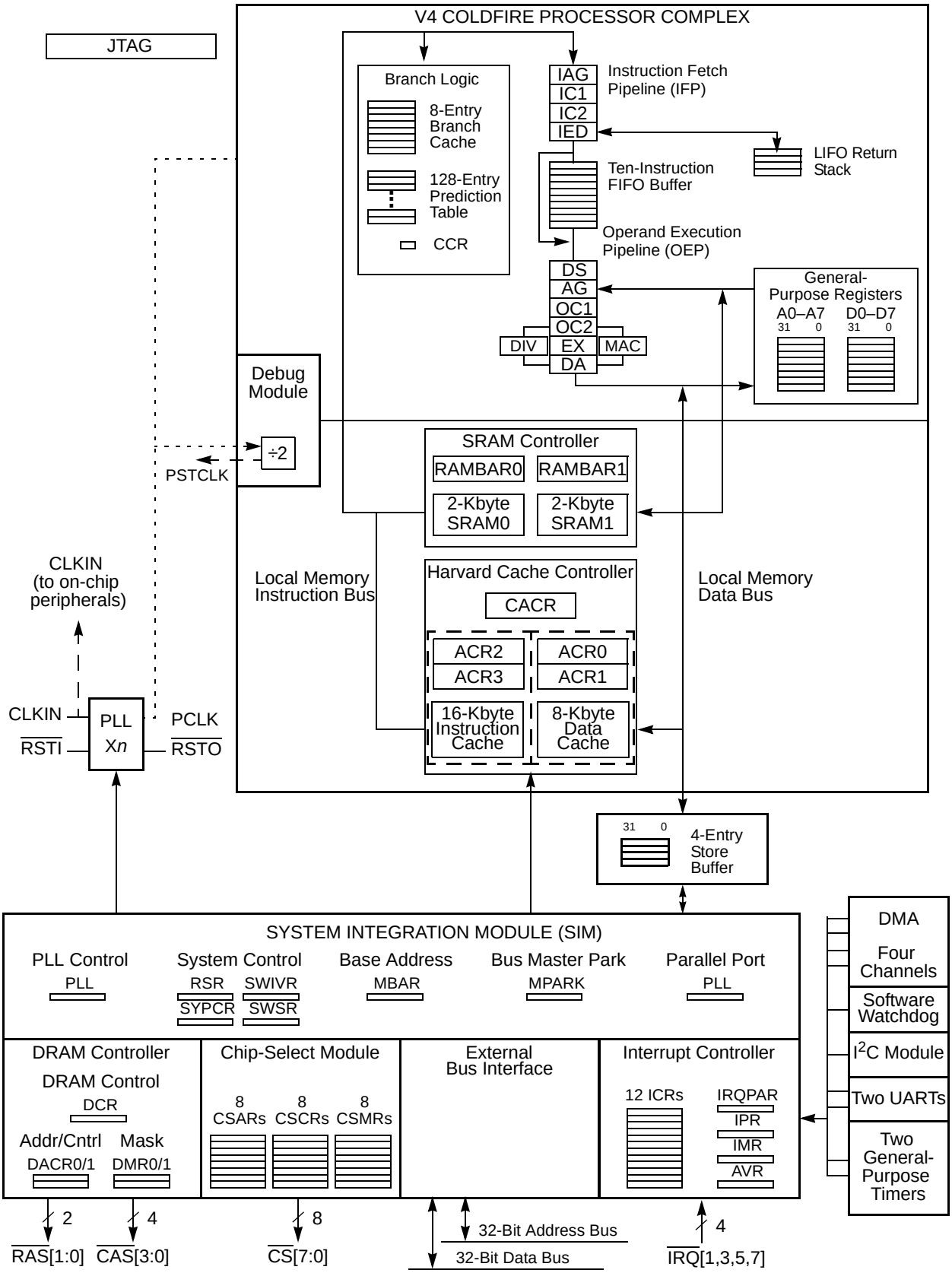

Figure 1. MCF5407 Block Diagram

Although the MCF5407 offers obvious performance upgrade advantages, its rich memory and peripheral integration at inexpensive prices should not be overlooked. Features common to many embedded applications, such as DMAs, various DRAM controller interfaces, and on-chip memories, are integrated in a cost-effective manner using aggressive process technologies.

The MCF5407 extends the legacy of Motorola's 68K family by providing a compatible path for 68K and ColdFire customers in which development tools and customer code are quickly leveraged. In fact, customers moving from 68K to ColdFire can use code translation and emulation tools that facilitate modifying 68K assembly code to the ColdFire architecture. The package, pinout, and integration mix of the MCF5407 create an especially simple upgrade for current MCF5307 designs with over triple the system performance.

The revolutionary ColdFire microprocessor architecture provides new levels of price and performance to cost-sensitive markets. Based on the concept of variable-length RISC technology, the ColdFire family combines the architectural simplicity of conventional 32-bit RISC with a memory-saving, variable-length instruction set. In defining the ColdFire architecture for embedded processing applications, a 68K-code compatible core was created that combines the performance advantages of a RISC architecture with the optimum code density of a streamlined, variable-length M68000 instruction set.

By using a variable-length instruction set architecture, embedded system designers using ColdFire RISC processors enjoy significant advantages over conventional fixed-length RISC architectures. The denser binary code for ColdFire processors consumes less memory than many fixed-length instruction set RISC processors available. This improved code density means more efficient system memory use for a given application, and allows use of slower, less costly memory to help achieve a target performance level.

The MCF5407 is the first standard product to implement the Version 4 ColdFire microprocessor core. The V4 microarchitecture implements a number of advanced techniques, including a Harvard memory architecture, branch cache acceleration logic, and limited superscalar support (dual-instruction issue), which contribute to the 316 Dhrystone MIPS performance level. Increasing the internal speed of the core also allows higher performance while providing the system designer with an easy-to-use lower speed system interface. The processor complex frequency is an integer multiple, 3 to 6 times, of the external bus frequency. The core clock can be stopped to support a low-power mode in the MCF5407.

Serial communication channels are provided by two programmable full-duplex UARTs, one of which provides synchronous communications for soft-modem applications, and an I^2C interface module. Four channels of DMA allow for fast data transfer using a programmable burst mode independent of processor execution. The two 16-bit general-purpose multimode timers provide separate input and output signals. For system protection, the processor includes a programmable 16-bit software watchdog timer. In addition, common system functions such as chip selects, interrupt control, bus arbitration, and an IEEE 1149.1 JTAG module are included.

A sophisticated debug interface supports background-debug mode plus real-time trace and debug with an expanded set of on-chip breakpoint registers. This interface is present in all ColdFire standard products and allows common emulator support across the entire family of microprocessors.

1.2 MCF5407 Features

The following list summarizes MCF5407 features:

- ColdFire processor core
  - — Variable-length RISC, clock-multiplied Version 4 microprocessor core
  - — Implements Revision B of the ColdFire instruction set architecture (ISA), which leverages the 68K programming model

  - Two independent decoupled pipelines: four-stage instruction fetch pipeline (IFP) and five-stage operand execution pipeline (OEP)
  - Ten-instruction FIFO buffer provides decoupling between the pipelines
  - Limited superscalar design achieves performance levels close to dual-issue performance
  - Programmable two-level branch acceleration mechanism with an 8-entry branch cache plus a 128-entry prediction table for increased performance
  - 32-bit internal address bus supporting 4 Gbytes of linear address space
  - 32-bit data bus
  - 16 user-accessible, 32-bit-wide, general-purpose registers
  - Supervisor/user modes for system protection
  - Vector base register to relocate exception-vector table
  - Optimized for high-level language constructs
- Multiply and accumulate unit (MAC)
  - Provides high-speed, complex arithmetic processing for DSP applications
  - Tightly coupled to the OEP
  - Three-stage execute pipeline with one clock issue rate for 16 x 16 operations
  - Supports 16 x 16 and 32 x 32 multiplies, all with 32-bit accumulate
  - Supports signed or unsigned integers, plus signed fractional operands
- Hardware integer divide unit
  - Supports unsigned and signed integer divides
  - Tightly coupled to the OEP
  - Supports 32/16, and 32/32 operations producing quotient and/or remainder results
- 16-Kbyte instruction cache, 8-Kbyte data cache
  - Four-way set-associative organization
  - Operates at higher processor core frequency
  - Provides pipelined, single-cycle access to critical code and data
  - Data cache supports write-through and copyback modes
  - Four-entry, 32-bit store buffer to improve performance of operand writes
- Two, 2-Kbyte SRAMs
  - Programmable location anywhere within 4-Gbyte linear address space
  - Operates at higher core frequency
  - Provides pipelined, single-cycle access to critical code and/or data
  - Each block can be mapped to either the instruction or data operand bus
- DMA controller
  - Four fully-programmable channels: two support external requests and external acknowledges
  - Supports dual-address and single-address transfers with 8-, 16-, and 32-bit data capability
  - Source/destination address pointers that can increment or remain constant
  - 24-bit transfer counter per channel

  - Operand packing and unpacking supported
  - Auto-alignment transfers supported for efficient block movement
  - Supports bursting and cycle steal
  - Provides two bus clock internal access
  - Automatic DMA transfers from on-chip UARTs using internal interrupts
- DRAM controller
  - Support for synchronous DRAM (SDRAM), extended-data-out (EDO) DRAM, and fast page mode
  - Supports up to 512 Mbytes of DRAM
  - Programmable timer provides CAS-before-RAS refresh for asynchronous DRAMs
  - Support for two separate memory blocks
- Two UARTs
  - One UART offers synchronous mode with expanded buffers for soft modem support
  - Full-duplex operation
  - Flexible baud-rate generator
  - Modem control signals available ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$)
  - Processor-interrupt capability
- Dual 16-bit general-purpose multiple-mode timers
  - 8-bit prescaler
  - Timer input and output pins
  - Processor-interrupt capability
  - Up to 18.5-nS resolution at 54 MHz
- I^2C module
  - Interchip bus interface for EEPROMs, LCD controllers, A/D converters, keypads
  - Fully compatible with industry-standard I^2C bus
  - Master or slave modes support multiple masters
  - Automatic interrupt generation with programmable level
- System interface module (SIM)
  - Chip selects provide direct interface to 8-, 16-, and 32-bit SRAM, ROM, FLASH, and memory-mapped I/O devices
  - Eight, fully-programmable chip selects, each with a base address register
  - Programmable wait states and port sizes per chip select
  - User-programmable processor clock/input clock frequency ratio
  - Programmable interrupt controller
  - Low interrupt latency
  - Four external interrupt request inputs
  - Programmable autovector generator
  - Software watchdog timer

- 16-bit general-purpose I/O interface
- IEEE 1149.1 test (JTAG) module
- System debug support
  - Real-time trace for determining dynamic execution path while in emulator mode
  - Background debug mode (BDM) for debug features while halted
  - Real-time debug support, including 13 user-visible hardware breakpoint registers
  - Supports servicing of critical, real-time interrupt requests while the BDM is in emulator mode
  - Supports comprehensive emulator functions through trace and breakpoint logic
- On-chip PLL
  - Accepts various clock input (CLKIN) frequencies between 25 and 54 MHz
  - Supports core frequencies between 100 and 162 MHz
  - Supports low-power mode
- Product offerings
  - 316 Dhrystone MIPS at 220 MHz
  - 233 Dhrystone MIPS at 162 MHz
  - Implemented in 0.22 µ, quad-layer-metal process technology with 1.8-V operation (3.3-V compliant I/O pads)
  - 208-pin plastic QFP package
  - 0 to 70° C operating temperature at 162 and 220 MHz
  - -40 to 85° C operating temperature at 162 MHz

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 that allow conditional branch instructions which 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 net effect 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, 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

Some embedded applications can benefit greatly from the integer divide unit. Integrated as another engine in the processor’s OEP, the divide module performs a variety of 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 is implemented to support the increased bandwidth requirements of the V4 processor pipelines. In this design featuring separate instruction and data buses to the processor-local memories, available bandwidth to the processor reaches 1.3 Gbytes/S at 162 MHz and conflicts between instruction fetches and operand accesses are removed.

1.3.2.1 16-Kbyte Instruction Cache/8-Kbyte Data Cache

Attached to the Harvard memory architecture are a 16-Kbyte instruction cache and an 8-Kbyte data cache. These four-way, set-associative designs improve system performance by providing pipelined, single-cycle access on instruction fetches and operand accesses that hit in these memories.

As with all ColdFire caches, these controllers implement a non-lockup, streaming design to maximize performance. The use of processor-local memories decouples performance from external memory speeds and increases available bandwidth for external devices or the on-chip 4-channel DMA.

Both caches implement line-fill buffers to optimize the performance of line-sized (16-byte) burst accesses. Additionally, the data cache supports operation of copyback, write-through or noncacheable modes. A 4-entry, 32-bit buffer is used for cache line push operations and can be configured for deferred write buffering while in write-through or non-cacheable modes.

The new INTOUCH instruction can be used to prefetch instructions to be locked in the instruction cache using the cache locking feature. This function may be desirable in certain systems where deterministic real-time performance is critical.

1.3.2.2 Internal 2-Kbyte SRAMs

The two 2-Kbyte on-chip SRAM modules are also connected to the Harvard memory architecture, and provide pipelined, single-cycle access to those memory regions mapped to these devices. Each memory can be independently mapped to any 0-modulo-2K location within the 4-Gbyte address space, and configured to respond either to instruction or to data accesses. Time-critical functions can be mapped onto the instruction memory bus, while the system stack and/or heavily-referenced data operands can be mapped onto the data memory bus.

1.3.3 DRAM Controller

The MCF5407 DRAM controller provides a direct interface for up to two blocks of DRAM. The controller supports 8-, 16-, or 32-bit memory widths, and can easily interface to PC-100 DIMMs. A unique addressing scheme allows for increases in system memory size without rerouting address lines and rewiring boards. The controller operates in normal mode or in page mode and supports SDRAMs and EDO DRAMs.

1.3.4 DMA Controller

The MCF5407 provides four fully-programmable DMA channels for quick data transfer. Dual- and single-address modes provide the ability to program bursting and cycle steal. Data transfers are 32 bits long with packing and unpacking supported along with an auto-alignment option for efficient block transfers. Automatic block transfers from on-chip serial UARTs are also supported through the DMA channels.

1.3.5 UART Modules

The MCF5407 contains two UARTs, which function independently. One UART has been enhanced to provide synchronous operation and a CODEC interface for soft modem support. Each UART can be clocked by the system bus clock, eliminating the need for an external crystal. Each UART module interfaces directly to the CPU, as shown in Figure 2.

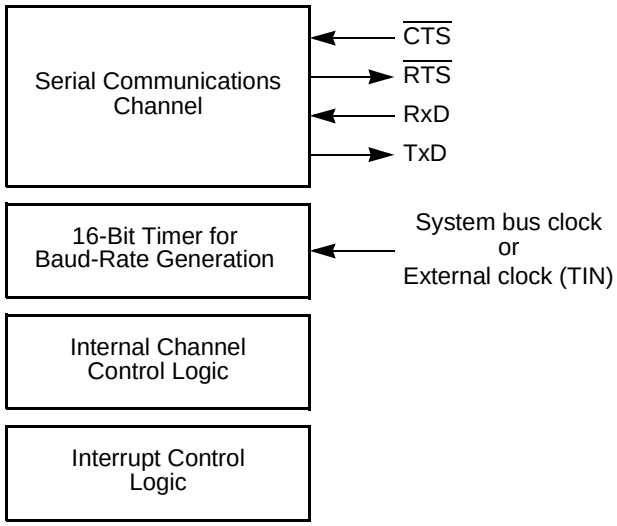

Figure 2. UART Module Block Diagram

Each UART module consists of the following major functional areas:

- Serial communication channel
- 16-bit timer for baud-rate generation
- Internal channel control logic
- Interrupt control logic

In addition, UART1 is enhanced to provide a CODEC interface for soft modem support. UART1 can be programmed to function like UART0 or in one of three following modem modes:

- An 8-bit CODEC interface
- A 16-bit CODEC interface
- An audio CODEC ‘97 (AC97) digital interface controller

Each UART contains an on-chip baud-rate generator, which provides both standard and nonstandard baud rates. Data formats can be 5, 6, 7, or 8 bits with even, odd, or no parity, and up to 2 stop bits in 1/16 increments. The UARTs include the following transmit and receive FIFO buffers:

- UART0 has a 4-byte FIFO receive buffer and a 2-byte FIFO transmit buffer.
- In UART1, the Tx and Rx FIFOs can hold the following:
  - — 32 1-byte samples when programmed as a UART or as an 8-bit CODEC interface
  - — 16 2-byte samples when programmed as a 16-bit CODEC interface
  - — 16 20-bit samples when programmed as a Digital AC ’97 Controller

The UART modules also provide several error-detection and maskable-interrupt capabilities. Modem support includes request-to-send ($\overline{\text{RTS}}$) and clear-to-send ($\overline{\text{CTS}}$) lines.

CLKIN provides the time base through a programmable prescaler. The UART time scale can also be sourced from a timer input. Full-duplex, auto-echo loopback, local loopback, and remote loopback modes allow testing of UART connections. The programmable UARTs can interrupt the CPU on various normal or error-condition events.

1.3.6 Timer Module

The timer module includes two general-purpose timers, each of which contains a free-running 16-bit timer for use in any of three modes. One mode captures the timer value with an external event. Another mode triggers an external signal or interrupts the CPU when the timer reaches a set value, while a third mode counts external events.

The timer unit has an 8-bit prescaler that allows programming of the clock input frequency, which is derived from the system bus cycle or an external clock input pin (TIN). The programmable timer-output pin generates either an active-low pulse or toggles the output.

1.3.7 I^2C Module

The I^2C interface is a two-wire, bidirectional serial bus used for quick data exchanges between devices. The I^2C minimizes the interconnection between devices in the end system and is best suited for applications that need occasional bursts of rapid communication over short distances among several devices. The I^2C can operate in master, slave or multiple-master modes.

1.3.8 System Interface

The MCF5407 processor provides a direct interface to 8-, 16-, and 32-bit FLASH, SRAM, ROM, and peripheral devices through the use of fully-programmable chip selects and write enables. Support for burst ROMs is also included. Through the on-chip PLL, users can input a slower clock (25 to 54 MHz) that is internally multiplied to create the faster processor clock (100 to 162 MHz).

1.3.8.1 External Bus Interface

The bus interface controller transfers information between the ColdFire core or DMA, and memory, peripherals, or other devices on the external bus. The external bus interface provides up to 32 bits of address bus space, a 32-bit data bus, and all associated control signals. This interface implements an extended synchronous protocol that supports bursting operations.

Simple two-wire request/acknowledge bus arbitration between the MCF5407 processor and another bus master, such as an external DMA device, is glueless with arbitration logic internal to the MCF5407 processor. Multiple-master arbitration is also available with some simple external arbitration logic.

1.3.8.2 Chip Selects

Eight fully-programmable chip select outputs support the use of external memory and peripheral circuits with user-defined wait-state insertion. These signals interface to 8-, 16-, 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 contained 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 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 and 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.

To support program trace, the Version 4 debug module has combined the processor status and debug data outputs into a single 8-bit bus (PSTDDATA[7:0]). This bus along with the PSTCLK output provide execution status, captured operand data and branch target addresses defining processor activity at one-half the CPU's clock rate.

1.3.10 PLL Module

The MCF5407 PLL module is shown in Figure 3.

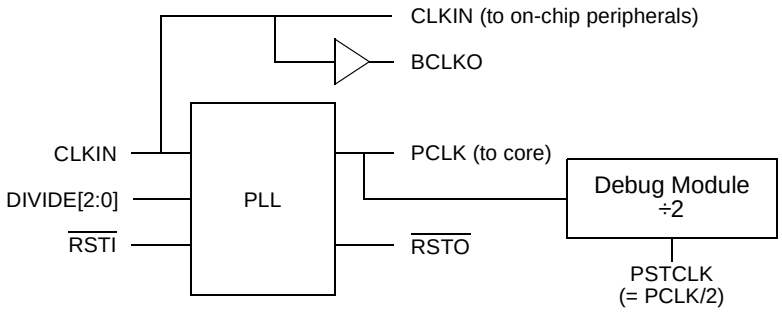

Figure 3. PLL Module

The PLL module's three modes of operation (reset, normal, and reduced power) are described as follows.

- Reset mode—When $\overline{\text{RSTI}}$ is asserted, the PLL enters reset mode. At reset, the PLL asserts $\overline{\text{RSTO}}$ from the MCF5407. The core:bus frequency ratio along with other MCF5407 configuration information are sampled during reset.
- Normal mode—In normal mode, the input frequency programmed at reset is clock-multiplied to provide the processor clock (PCLK).
- Reduced-power mode—In reduced-power mode, the PCLK is disabled by executing a sequence including programming a control bit in the system configuration register (SCR) and then executing the STOP instruction. Register contents are retained in reduced-power mode, so the system can be reenabled quickly when an unmasked interrupt or reset is detected.

1.4 Programming Model, Addressing Modes, and Instruction Set

The ColdFire programming model is separated into two privilege modes—supervisor and user, which is indicated by the S bit in the status register (SR). The processor identifies a logical address by accessing either the supervisor or user address space, which differentiates between supervisor and user modes.

- User mode—When the processor is in user mode (SR[S] = 0), only a subset of the registers can be accessed and privileged instructions cannot be executed. Typically, most application processing occurs in user mode. Entry into user mode is usually accomplished by executing a return from exception instruction (RTE) (assuming the value of SR[S] saved on the stack is 0) or a MOVE, SR instruction (assuming SR[S] is 0).
- Supervisor mode—Supervisor mode protects system resources from uncontrolled access by users. In supervisor mode, complete access to all registers and the entire ColdFire instruction set is provided. Typically, system programmers use the supervisor programming model to implement operating system functions and provide I/O control. The supervisor programming model provides access to the same registers as the user model, plus additional registers for configuring on-chip system resources, as described in Section 1.4.3, "Supervisor Registers."

  Exceptions (including interrupts) are handled in supervisor mode.

1.4.1 Programming Model

Figure 4 shows the MCF5407 programming model.

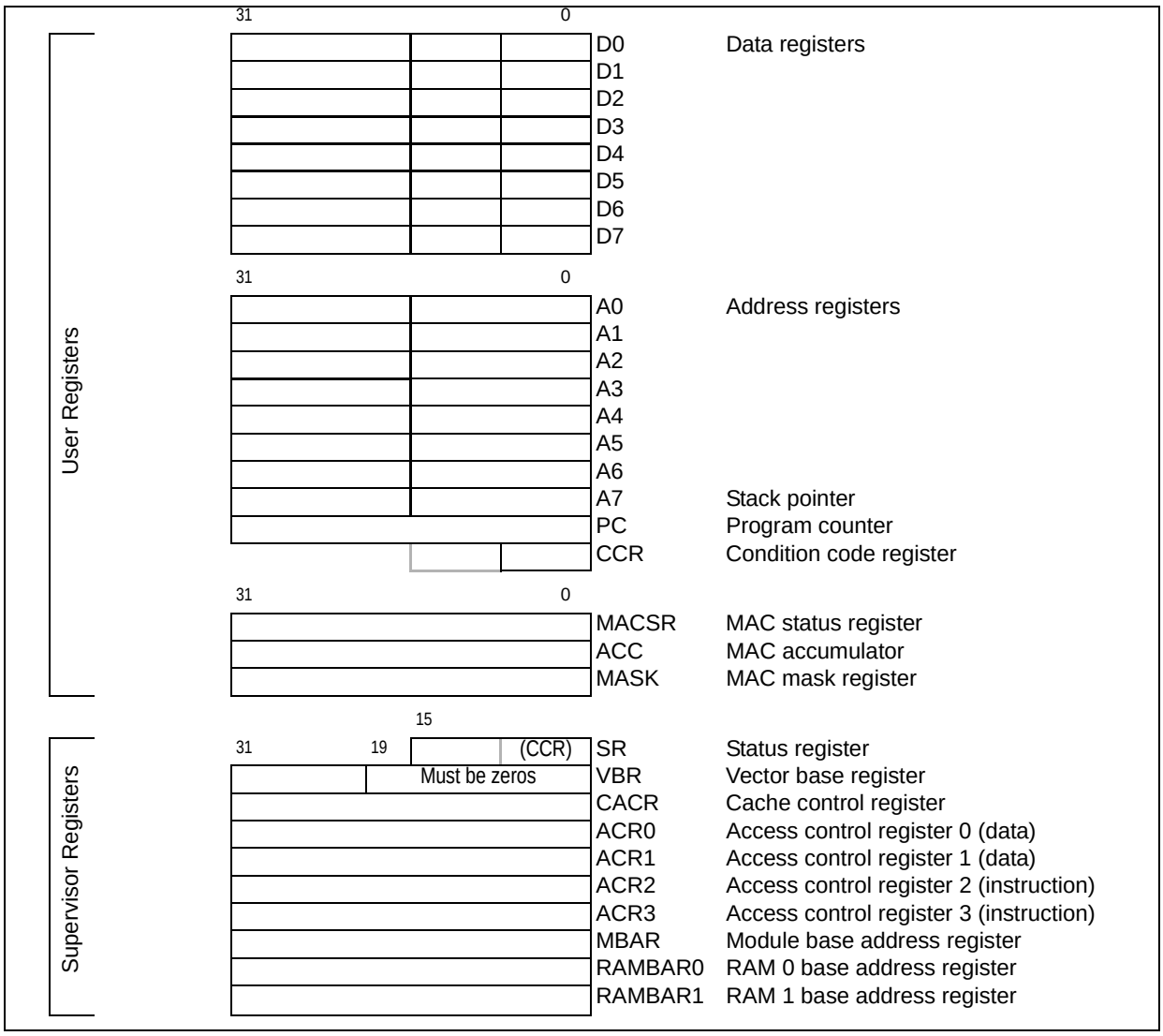

Figure 4. ColdFire MCF5407 Programming Model

1.4.2 User Registers

The user programming model, shown in Figure 4, is summarized in Table 1.

Table 1. User-Level Registers

Register	Description
Data registers (D0–D7)	These 32-bit registers are for bit, byte, word, and longword operands. They can also be used as index registers.
Address registers (A0–A7)	These 32-bit registers serve as software stack pointers, index registers, or base address registers. The base address registers can be used for word and longword operations. A7 functions as a hardware stack pointer during stacking for subroutine calls and exception handling.
Program counter (PC)	Contains the address of the instruction currently being executed by the MCF5407 processor.
Condition code register (CCR)	The CCR is the lower byte of the SR. It contains indicator flags that reflect the result of a previous operation and are used for conditional instruction execution.
MAC status register (MACSR)	Defines the operating configuration of the MAC unit and contains indicator flags from the results of MAC instructions.
Accumulator (RACC)	General-purpose register used to accumulate the results of MAC operations.
Mask register (RMASK)	General-purpose register provides an optional address mask for MAC instructions that fetch operands from memory. It is useful in the implementation of circular queues in operand memory.

1.4.3 Supervisor Registers

Table 2 summarizes the MCF5407 supervisor-level registers.

Table 2. Supervisor-Level Registers

Register	Description
Status register (SR)	The upper byte of the SR provides interrupt information in addition to a variety of mode indicators signaling the operating state of the ColdFire processor. The lower byte of the SR is the CCR, as shown in Figure 4.
Vector base register (VBR)	Defines the upper 12 bits of the base address of the exception vector table used during exception processing. The low-order 20 bits are forced to zero, locating the vector table on 0-modulo-1 Mbyte address.
Cache configuration register (CACR)	Defines the operating modes of the Version 4 cache memories. Control fields configuring the instruction, data and branch cache are provided by this register, along with the default attributes for the 4-Gbyte address space.
Access control registers (ACR0/1, ACR2/3)	Define address ranges and attributes associated with various memory regions within the 4-Gbyte address space. Each ACR defines the location of a given memory region and assigns attributes such as write-protection and cache mode (copyback, write-through, cacheability). ACR0 and ACR1 support data memory; ACR2 and ACR3 support instruction memory. Additionally, CACR fields assign default attributes to the instruction and data memory spaces.
RAM base address registers (RAMBAR0, RAMBAR1)	Provide the logical base address for the two 2-Kbyte SRAM modules and define attributes and access types allowed for the corresponding SRAM.
Module base address register (MBAR)	Defines the logical base address for the memory-mapped space containing the control registers for the on-chip peripherals.

1.4.4 Addressing Modes

Operands can be signed or unsigned and are contained in registers, memory, or the instructions themselves. The operand specifiers and size for each operation are either explicitly encoded in the instruction or implicitly defined by the instruction's definition. Table 3 shows MCF5407 data formats.

Table 3. MCF5407 Data Formats

Data Format	Size
Bit	1 bit
Byte	8 bits
Word	16 bits
Longword	32 bits

Table 4 shows notational conventions used throughout this document.

Table 4. 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

Table 4. Notational Conventions (continued)

Instruction	Operand Syntax
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
<xxx>	identifies an absolute address referencing memory
dn	Signal displacement value, *n* bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication
/	Arithmetic division
~	Invert; operand is logically complemented
&	Logical AND
|	Logical OR
^	Logical exclusive OR
<<	Shift left (example: D0 << 3 is shift D0 left 3 bits)
>>	Shift right (example: D0 >> 3 is shift D0 right 3 bits)
→	Source operand is moved to destination operand
←→	Two operands are exchanged
sign-extended	All bits of the upper portion are made equal to the high-order bit of the lower portion
If <condition> then <operations> else <operations>	Test the condition. If true, the operations after ‘then’ are performed. If the condition is false and the optional ‘else’ clause is present, the operations after ‘else’ are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.

Table 4. Notational Conventions (continued)

Instruction	Operand Syntax
Subfields and Qualifiers	
{}	Optional operation
()	Identifies an indirect address
d_n	Displacement value, n-bits wide (example: d_{16} is a 16-bit displacement)
Address	Calculated effective address (pointer)
Bit	Bit selection (example: Bit 3 of D0)
lsb	Least significant bit (example: lsb of D0)
LSB	Least significant byte
LSW	Least significant word
msb	Most significant bit
MSB	Most significant byte
MSW	Most significant word
Condition Code Register Bit Names	
C	Carry
N	Negative
V	Overflow
X	Extend
Z	Zero

1.4.4.1 Addressing Capability Summary

The MCF5407 processor supports seven addressing modes (refer to Table 5). Register indirect addressing modes support postincrement, predecrement, offset, and indexing, which are particularly useful for handling data structures common to sophisticated embedded applications and high-level languages. The program counter indirect mode also has indexing and offset capabilities. This addressing mode is typically required to support position-independent code. As part of the indexed addressing mode, ColdFire architecture supports an optional scale factor that can be applied to the index register to easily access byte, word, or longword entries within an array (x1, x2, x4).

An instruction's effective addressing mode can specify the operand in one of three ways:

- It can specify the data value directly as an immediate operand
- It can specify the register containing the operand
- It can specify the addressing calculation needed to reference the memory location containing the operand

Each addressing mode has a unique assembler syntax. In addition to the generalized format where the addressing mode is specified directly in the instruction, some opcodes implicitly define the effective address. Table 5 summarizes supported effective addressing modes.

Table 5. ColdFire Effective Addressing Modes

Addressing Modes	Syntax	Mode Field	Reg. Field	Category			
				Data	Memory	Control	Alterable
Register direct							
Data	Dn	000	reg. no.	X	—	—	X
Address	An	001	reg. no.	—	—	—	X
Register indirect							
Address	(An)	010	reg. no.	X	X	X	X
Address with Postincrement	(An)+	011	reg. no.	X	X	—	X
Address with Predecrement	–(An)	100	reg. no.	X	X	—	X
Address with Displacement	(d_{16}, An)	101	reg. no.	X	X	X	X
Address register indirect with scaled index 8-bit displacement	(d_8, An, Xi*SF)	110	reg. no.	X	X	X	X
Program counter indirect with displacement	(d_{16}, PC)	111	010	X	X	X	—
Program counter indirect with scaled index 8-bit displacement	(d_8, PC, Xi*SF)	111	011	X	X	X	—
Absolute data addressing							
Short	(xxx).W	111	000	X	X	X	—
Long	(xxx).L	111	001	X	X	X	—
Immediate	#<xxx>	111	100	X	X	—	—

Table 6 lists addressing modes for MOVE instructions.

Table 6. Specific Effective Addressing Modes for MOVE Instructions

Source <EA>	Destination <EA>
Dy	All
Ay	All
(Ay)	All
(Ay)+	All
-(Ay)	All

Table 6. Specific Effective Addressing Modes for MOVE Instructions (continued)

Source <EA>	Destination <EA>
(d_{16},Ay) (d_{16},PC)	Dx Ax (Ax) (Ax)+ -(Ax) (d_{16},Ax)
(d_8,Ay,Xi*SF) (d_8,PC,Xi*SF)	Dx Ax (Ax) (Ax)+ -(Ax)
(xxx).W (xxx).L	Dx Ax (Ax) (Ax)+ -(Ax)
#<data>	Dx Ax (Ax) (Ax)+ -(Ax) (d_{16},Ax) [1]

[1] Note that this applies only to move.b and move.w instructions

Table 7 lists additional addressing variants.

Table 7. Addressing Variants Used by Certain Instructions

Addressing Variant	Allowable Modes
<ea-1>	Dn (An) (An)+ -(An) (d_{16},An)
<ea-2>	(An) (d_{16},An)
<ea-3>	(An) (d_{16},An) (d_8,An,Xi*SF) (xxx).W (xxx).L (d_{16},PC) (d_8,PC,Xi*SF)

1.4.5 Instruction Set

The Version 4 ColdFire core implements Revision B of the instruction set, which adds opcodes to enhance support for byte- and word-sized operands and position-independent code. The ColdFire instruction set supports high-level languages and is optimized for those instructions most commonly generated by compilers in embedded applications. Table 9 and Table 10 provide an alphabetized listing of the ColdFire

instruction set opcodes, supported operation sizes, and assembler syntax. For two-operand instructions, the first operand in the syntax is generally the source operand, and the second operand is the destination.

Because the ColdFire architecture provides an upgrade path for 68K customers, its instruction set supports most of the common 68K opcodes. A majority of the instructions are binary compatible or optimized 68K opcodes. This feature, when coupled with the code conversion tools from third-party developers, generally minimizes software porting issues for customers with 68K applications.

The following list summarizes new and enhanced instructions of Revision B ISA:

- New instructions:
  - INTOUCH loads blocks of instructions to be locked in the instruction cache
  - MOV3Q.L moves 3-bit immediate data to destination location
  - MVS.{B,W} sign-extends the source operand and moves it to destination register
  - MVZ.{B,W} zero-fills the source operand and moves it to destination register
  - SATS.L updates bit 31 of destination register depending on CCR overflow bit
  - TAS.B tests and set byte operand being addressed.
- Enhancements to existing Revision A instructions:
  - Longword support for branch instructions (Bcc, BRA, BSR)
  - Byte and word support for compare instructions (CMP, CMPI)
  - Byte and longword support for MOVE.x where the source is of type #<data> and the destination is of type d16(Ax); that is, move.b #<data>, d16(Ax)

Table 8 lists the new and enhanced instructions.

Table 8. ColdFire ISA B Extension Summary

Instruction	Mnemonic
Branch Always	bra.l
Branch Conditionally	bcc.l
Branch to Subroutine	bsr.l
Compare	cmp.{b,w}
Compare Immediate	cmpi.{b,w}
Instruction Fetch Touch	intouch
Move 3-Bit Data Quick	mov3q.l
Move Data Source to Destination	move.{b,w}
Move with Sign Extend	mvs.{b,w}
Move with Zero-Fill	mvz.{b,w}
Signed Saturate	sats.l
Test and Set an Operand	tas.b

Table 9 describes supervisor-level instructions.

Table 9. Supervisor-Level Instruction Set Summary

Instruction	Operand Syntax	Operand Size	Operation
CPUSHL	(An)	Unsized	Invalidate instruction cache line Push and invalidate data cache line Push data cache line and invalidate (I,D)-cache lines
HALT [1]	none	Unsized	Enter halted state
INTOUCH	(Ax)	Unsized	Touch instruction space at address Ax
MOVE from SR	SR, Dx	.W	SR → Dx
MOVE to SR	Dy,SR #<data>,SR	.W	Source → SR
MOVEC	Ry,Rc	.L	Ry → Rc **Rc** **Register Definition** 0x002 Cache control register (CACR) 0x004 Access control register 0 (ACR0) 0x005 Access control register 1 (ACR1) 0x006 Access control register 2 (ACR2) 0x007 Access control register 3 (ACR3) 0x801 Vector base register (VBR) 0xC04 RAM base address register 0 (RAMBAR0) 0xC05 RAM base address register 1 (RAMBAR1)
RTE	None	Unsized	(SP+2) → SR; SP+4 → SP; (SP) → PC; SP + formatfield — SP
STOP	#<data>	.W	Immediate data → SR; enter stopped state
WDEBUG	<ea-2>y	.L	<ea-2>y → debug module

[1] The HALT instruction can be configured to allow user-mode execution by setting CSR[UHE].

Table 10 describes user-level instructions.

Table 10. User-Level Instruction Set Summary

Instruction	Operand Syntax	Operand Size	Operation
ADD	Dy,<ea>x <ea>y,Dx	.L .L	Source + destination → destination
ADDA	<ea>y,Ax	.L	Source + destination → destination
ADDI	#<data>,Dx	.L	Immediate data + destination → destination
ADDQ	#<data>,<ea>x	.L	Immediate data + destination → destination
ADDX	Dy,Dx	.L	Source + destination + X → destination
AND	Dy,<ea>x <ea>y,Dx	.L .L	Source & destination → destination
ANDI	#<data>,Dx	.L	Immediate data & destination → destination
ASL	Dy,Dx #<data>,Dx	.L .L	X/C ← (Dx << Dy) ← 0 X/C ← (Dx << #<data>) ← 0
ASR	Dy,Dx #<data>,Dx	.L .L	MSB → (Dx >> Dy) → X/C MSB → (Dx >> #<data>) → X/C
Bcc	<label>	.B,.W,.L	If condition true, then PC + 2 + d_n → PC
BCHG	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z, Bit of destination

Table 10. User-Level Instruction Set Summary (continued)

Instruction	Operand Syntax	Operand Size	Operation
BCLR	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z; 0 → bit of destination
BRA	<label>	.B,.W,.L	PC + 2 + d_n → PC
BSET	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z; 1→ bit of destination
BSR	<label>	.B,.W,.L	SP – 4 → SP; next sequential PC→ (SP); PC + 2 + d_n → PC
BTST	Dy,<ea>x #<data>,<ea-1>x	.B,.L .B,.L	~(<bit number> of destination) → Z
CLR	<ea>y,Dx	.B,.W,.L	0 → destination
CMP	<ea>y,Ax	.B,.W,.L	Destination – source
CMPA	<ea>y,Dx	.B,.W,.L	Destination – source
CMPI	<ea>y,Dx	.B,.W,.L	Destination – immediate data
DIVS	<ea-1>y,Dx <ea>y,Dx	.W .L	Dx /<ea>y → Dx {16-bit remainder; 16-bit quotient} Dx /<ea>y → Dx {32-bit quotient} Signed operation
DIVU	<ea-1>y,Dx Dy,<ea>x	.W .L	Dx /<ea>y → Dx {16-bit remainder; 16-bit quotient} Dx /<ea>y → Dx {32-bit quotient} Unsigned operation
EOR	Dy,<ea>x	.L	Source ^ destination → destination
EORI	#<data>,Dx	.L	Immediate data ^ destination → destination
EXT	#<data>,Dx	.B →.W .W →.L	Sign-extended destination → destination
EXTB	Dx	.B →.L	Sign-extended destination → destination
HALT [1]	None	Unsized	Enter halted state
JMP	<ea-3>y	Unsized	Address of <ea> → PC
JSR	<ea-3>y	Unsized	SP – 4 → SP; next sequential PC → (SP); <ea> → PC
LEA	<ea-3>y,Ax	.L	<ea> → Ax
LINK	Ax,#<d16>	.W	SP – 4 → SP; Ax → (SP); SP → Ax; SP + d16 → SP
LSL	Dy,Dx #<data>,Dx	.L .L	X/C ← (Dx << Dy) ← 0 X/C ← (Dx << #<data>) ← 0
LSR	Dy,Dx #<data>,Dx	.L .L	0 → (Dx >> Dy) → X/C 0 → (Dx >> #<data>) → X/C
MAC	Ry,RxSF	.L + (.W × .W) → .L .L + (.L × .L) → .L	ACC + (Ry × Rx){<< 1 | >> 1} → ACC ACC + (Ry × Rx){<< 1 | >> 1} → ACC; (<ea>y{&MASK}) → Rw
MACL	Ry,RxSF,<ea-1>y,Rw	.L + (.W × .W) → .L, .L .L + (.L × .L) → .L, .L	ACC + (Ry × Rx){<< 1 | >> 1} → ACC ACC + (Ry × Rx){<< 1 | >> 1} → ACC; (<ea-1>y{&MASK}) → Rw
MOV3Q	#<data>,<ea>x	.L	3-bit immediate→destination
MOVE	<ea>y,<ea>x	.B,.W,.L	<ea>y → <ea>x

Table 10. User-Level Instruction Set Summary (continued)

Instruction	Operand Syntax	Operand Size	Operation
MOVE from MAC	MASK,Rx ACC,Rx MACSR,Rx	.L	Rm → Rx
	MACSR,CCR	.L	MACSR → CCR
MOVE to MAC	Ry,ACC Ry,MACSR Ry,MASK	.L	Ry → Rm
	#<data>,ACC #<data>,MACSR #<data>,MASK	.L	#<data> → Rm
MOVE from CCR	CCR,Dx	.W	CCR → Dx
MOVE to CCR	Dy,CCR #<data>,CCR	.B	Dy → CCR #<data> → CCR
MOVEA	<ea>y,Ax	.W,.L → .L	Source → destination
MOVEM	#<list>,<ea-2>x <ea-2>y,#<list>	.L .L	Listed registers → destination Source → listed registers
MOVEQ	#<data>,Dx	.B → .L	Sign-extended immediate data → destination
MSAC	Ry,RxSF	.L - (.W × .W) → .L .L - (.L × .L) → .L	ACC – (Ry × Rx){<< 1 | >> 1} → ACC
MSACL	Ry,RxSF,<ea-1>y,Rw	.L - (.W × .W) → .L, .L .L - (.L × .L) → .L, .L	ACC – (Ry × Rx){<< 1 | >> 1} → ACC; (<ea-1>y{&MASK}) → Rw
MULS	<ea>y,Dx	.W X .W → .L .L X .L → .L	Source × destination → destination Signed operation
MULU	<ea>y,Dx	.W X .W → .L .L X .L → .L	Source × destination → destination Unsigned operation
MVS	<ea>y,Dx	.B,.W	Sign-extended source → destination
MVZ	<ea-1>y,Dx	.B,.W	Zero-filled source → destination
NEG	Dx	.L	0 – destination → destination
NEGX	Dx	.L	0 – destination – X → destination
NOP	none	Unsized	Synchronize pipelines; PC + 2 → PC
NOT	Dx	.L	~ Destination → destination
OR	<ea>y,Dx Dy,<ea>x	.L	Source | destination → destination
ORI	#<data>,Dx	.L	Immediate data | destination → destination
PEA	<ea-3>y	.L	SP – 4 → SP; Address of <ea> → (SP)
PULSE	none	Unsized	Set PST= 0x4
REMS	<ea-1>,Dx	.L	Dx/<ea>y → Dw {32-bit remainder} Signed operation
REMU	<ea-1>,Dx	.L	Dx/<ea>y → Dw {32-bit remainder} Unsigned operation
RTS	none	Unsized	(SP) → PC; SP + 4 → SP

Table 10. User-Level Instruction Set Summary (continued)

Instruction	Operand Syntax	Operand Size	Operation
SATS	Dx	.L	if CCR.V=1, then if Dx[31] =0 then 0x80000000 → Dx else 0x7FFFFFFF→ Dx else Dx is unchanged
Scc	Dx	.B	If condition true, then 1s — destination; Else 0s → destination
SUB	<ea>y,Dx Dy,<ea>x	.L .L	Destination – source → destination
SUBA	<ea>y,Ax	.L	Destination – source → destination
SUBI	#<data>,Dx	.L	Destination – immediate data → destination
SUBQ	#<data>,<ea>x	.L	Destination – immediate data → destination
SUBX	Dy,Dx	.L	Destination – source – X → destination
SWAP	Dx	.W	MSW of Dx ←→ LSW of Dx
TAS	<ea>x	.B	Set CCR; 1→ Bit 7 of <ea>x
TRAP	#<vector>	Unsized	SP – 4 → SP;PC → (SP); SP – 2 → SP;SR → (SP); SP – 2 → SP; format → (SP); Vector address → PC
TRAPF	None #<data>	Unsized .W .L	PC + 2 → PC PC + 4 → PC PC + 6 → PC
TST	<ea>y	.B,.W,.L	Set condition codes
UNLK	Ax	Unsized	Ax →SP; (SP) → Ax; SP + 4 → SP
WDDATA	<ea>y	.B,.W,.L	<ea>y →DDATA port

[1] By default the HALT instruction is a supervisor-level instruction; however, it can be configured to allow user-mode execution by setting CSR[UHE].

1.5 General Device Information

Table 11 shows the MCF5407 package, temperature, and frequency specifications,

Table 11. MCF5407 Package/Frequency

Package	Operating Temperature	Frequency
208 plastic QFP	0 to 70° C	54 MHz max CLKIN/ 220 MHz max PCLK
208 plastic QFP	-40 to 85° C	54 MHz max CLKIN/ 162 MHz max PCLK

Table 12 lists additional MCF5407 documentation.

Table 12. Documentation

Documentation Number	Documentation Title
MCF5407UM/AD	*MCF5407 User's Manual*
TBD	*Application Note: Migrating from the ColdFire MCF5307 to the MCF5407*
MCF5200PRM/AD Rev 1	ColdFire Family *Programmer's Reference Manual*

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) 933-2625,
Attn: TECD Applications Engineering

MCF5407PB/D