Freescale Semiconductor
Application Note

AN1983
Rev. 1, 07/2005

Software Compatibility Considerations for HCS12, HC16 and 56800/E Devices

David Barron

Contents

1. Introduction

This document provides general information that will help in porting code from the HCS12/HC16 microcontroller cores to the higher-performance 56800/E cores. The 56800/E controllers provide a migration path that will help in designing code for the 56800/E core in order to port from the End-Of-Life (EOL) HC16 devices or to migrate from HCS12 designs to the higher-performance controllers in the 56800/E families.

This document will focus on the similarities and differences between the two families with respect to Assembly language coding. Both families have C/C++ compilers available with fully integrated environments and porting C code is a relatively simple process, so it will be discussed only briefly here.

2. Motivation

As products evolve from generation to generation, system costs often decrease while functionality increases. This usually requires an increase in the performance capability of the MCU.

There are several ways in which this evolution can take place. A reduction in the product's parts count can offset the greater cost of the higher-performance processor, or additional functionality can be added to the product to help distinguish it in the market place.

The 56800 family of controllers provides a migration path to higher-performance cores with integrated Flash memory at extended operating temperatures.

2.1 Increased Performance

The 56800/E family of processors can significantly increase the performance of a conventional MCU-based system, providing the needed room for growth.

The HCS12 has a maximum bus cycle speed of 25MHz, while the HC16 has a maximum bus cycle speed of 16MHz. Both of these architectures require multiple bus cycles per instruction for most types of instructions. The Multiply and Accumulate (MAC) instruction, the heart of most Digital Signal Processing (DSP) algorithms, is a good example of a multiple bus cycle instruction in these cores. The MAC instruction requires up to 12 to 13 bus cycles per iteration on both of these architectures.

The 56800 has a maximum bus clock of 80MHz and requires two bus cycles for most instructions, including the MAC instruction. This is a significant increase in throughput over the HCS12/HC16 cores. The 56800E, the second-generation core, provides a maximum performance of 120 MIPS at 120MHz core frequency for RAM-based controllers and a maximum of 60MIPS at 60MHz core frequency for Flash-based controllers.

These comparisons are illustrated in **Table 2-1** for the MAC instruction.

Table 2-1. Performance Comparison

Core	Maximum Bus Frequency (MHz)	MMACS
M68HC12	8	(8/13) = 0.6
M68HC16	16	(16/12) = 1.3
M68HCS12	25	(25/13) = 1.9
56800	80	(80/2) = 40
56800E	120	(120/1) = 120
56800E with Flash	60	(60/1) = 60

2.2 Consolidating Functions

Designers are driven to reduce product costs in order to be competitive in the market, and one of the most common methods to achieve a cost reduction is to consolidate functionality. This can reduce the total part count of the product, bringing the cost down, and can also save space on the board. The 56800/E family of controllers have the additional core performance and peripherals necessary to combine the functionality of several standard MCUs.

2.3 Signal Processing

Increasing functionality in a product quite often involves the need to perform DSP functions such as digital filters, modems, echo cancellers, voice compression algorithms, motor control algorithms, etc. Obviously a conventional microcontroller will become overburdened very quickly when performing signal processing functions that require a significant throughput. However, the 56800/E cores have highly optimized, DSP-specific instructions and can perform highly complex signal processing algorithms at high sample rates, yet still have overhead remaining for the necessary MCU control functions.

3. Core Comparison

3.1 Programming Model

The HCS12 is the next generation HC12 and is the standard Freescale 16-bit microcontroller core, with many derivatives available. The HCS12/HC16 cores are both derived from the earlier HC11 8-bit core and therefore have very similar architectures. The instuction sets of these cores are a proper superset of the HC11 core, so the parts are highly compatible.

The 56800E is the next-generation 56800 core, so these two processors are also very similar in both functionality and instruction sets. The main differences between these two cores is that the addressing range was increased, byte addressing mode was added, and the core clock speed was enhanced for the 56800E core.

In general, the 56800/E cores offer a larger register set than do the HCS12/HC16 cores, and when combined with the more sophisticated address generation unit (two AGUs), make it easy to port to these cores (more available resources makes the task easier). The 56800/E processors offer all the necessary addressing modes (i.e., post-decrement, post-increment, indexed, immediate, etc.) for performing general-purpose MCU tasks.

The programming models (register sets) are illustrated for reference.

3.1.1 M68HCS12

The HCS12 programming model consists of seven registers; two 8-bit accumulators which can be concatenated into one 16-bit accumulator; two index registers; a stack pointer; a program counter; and a condition code register. As shown in **Figure 3-1**, the two 8-bit registers, A and B, can be accessed as one 16-bit register D.

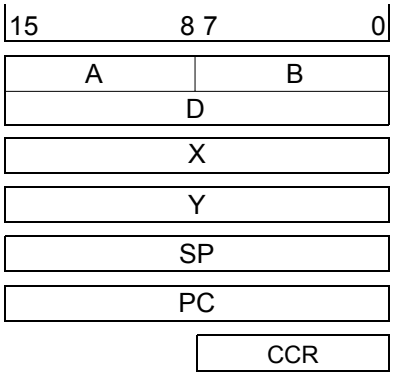

Figure 3-1. M68HCS12 Programming Model

3.1.2 M68HC16

The HC16 programming model, shown in **Figure 3-2**, consists of the same basic set of registers as the HCS12, plus an extra accumulator, an extra index register, and several special-purpose registers for performing DSP-specific instructions. The address registers have been extended from the HCS12's 16 bits to 20 bits by adding a four-bit extension field to allow for 1MB addressing.

A Multiply and Accumulate (MAC) unit was added to the core in the HC16 for performing low-frequency DSP applications. The MAC unit on the HC16 consists of two operand registers (HR and IR), a 36-bit accumulator, and two power-of-two modulo mask registers (XMSK and YMSK).

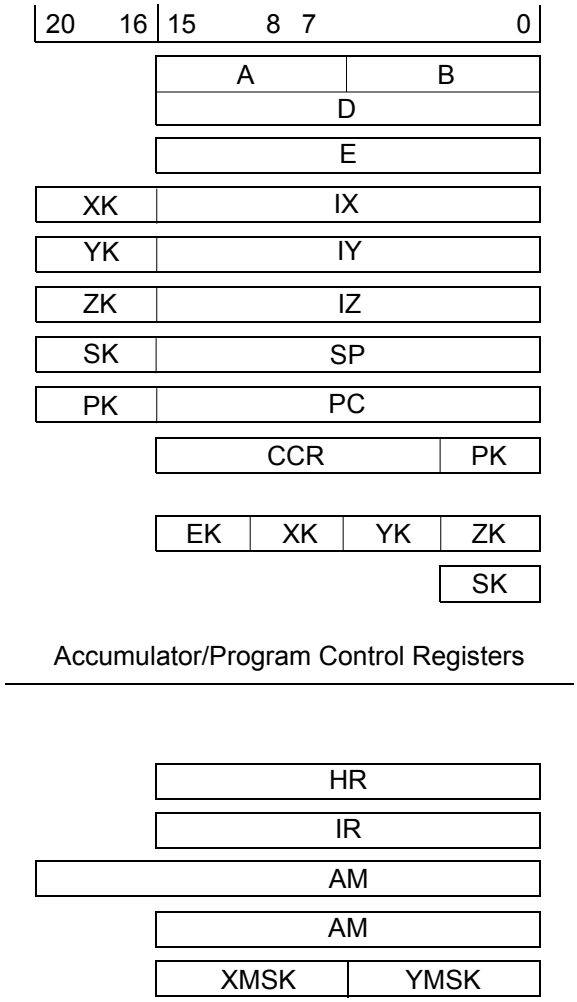

Figure 3-2. M68HC16 Programming Model

3.1.3 56800

The 56800 programming model consists of two 36-bit accumulators (the three registers that make up each of the accumulators can be individually accessed); three 16-bit general-purpose registers (two of which can be concatenated into a single 32-bit register); four address registers with an associated indexing register and modulo register; a loop counter and loop address register for hardware do loops; a program counter and status register (the status register contains the condition codes); and an operating mode register. The modulo register can be applied to R0 and/or R1 and can perform modulo addressing on a buffer of any length. The start address of the modulo buffer is still required to be on a power-of-two address boundary.

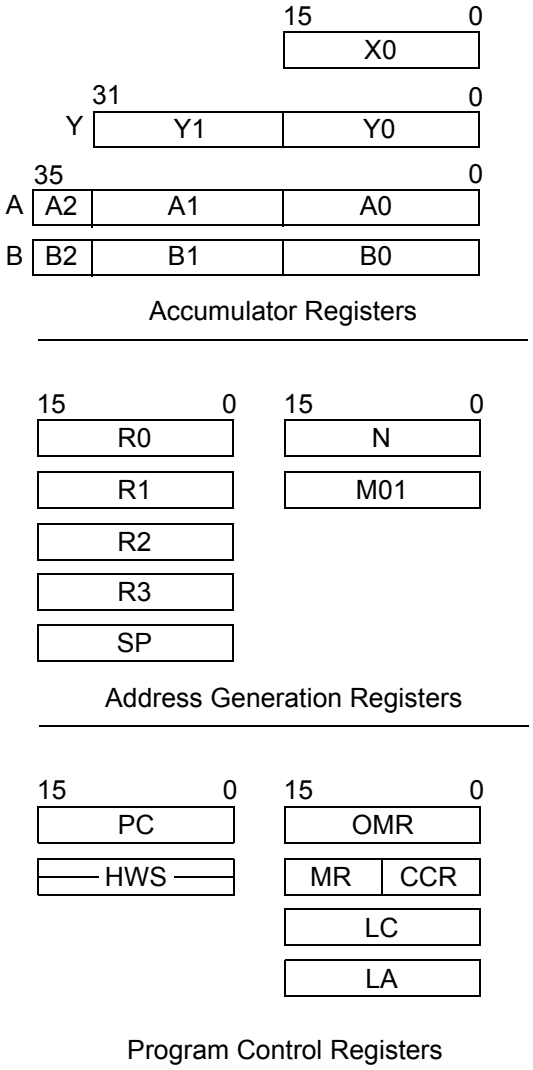

Figure 3-3. 56800 Programming Model

3.1.4 56800E

The 56800E programming model consists of four 36-bit accumulators (the three registers that make up each of the accumulators can be individually accessed); three 16-bit general purpose registers (two of which can be concatenated into a single 32-bit register); six address registers with two associated indexing registers and one modulo register; a loop counter and loop address register for hardware do loops; a program counter and status register; and an operating mode register. The modulo register can be applied to R0 and/or R1 and can perform modulo addressing on a buffer of any length. The start address of the modulo buffer is still required to be on a power-of-two address boundary.

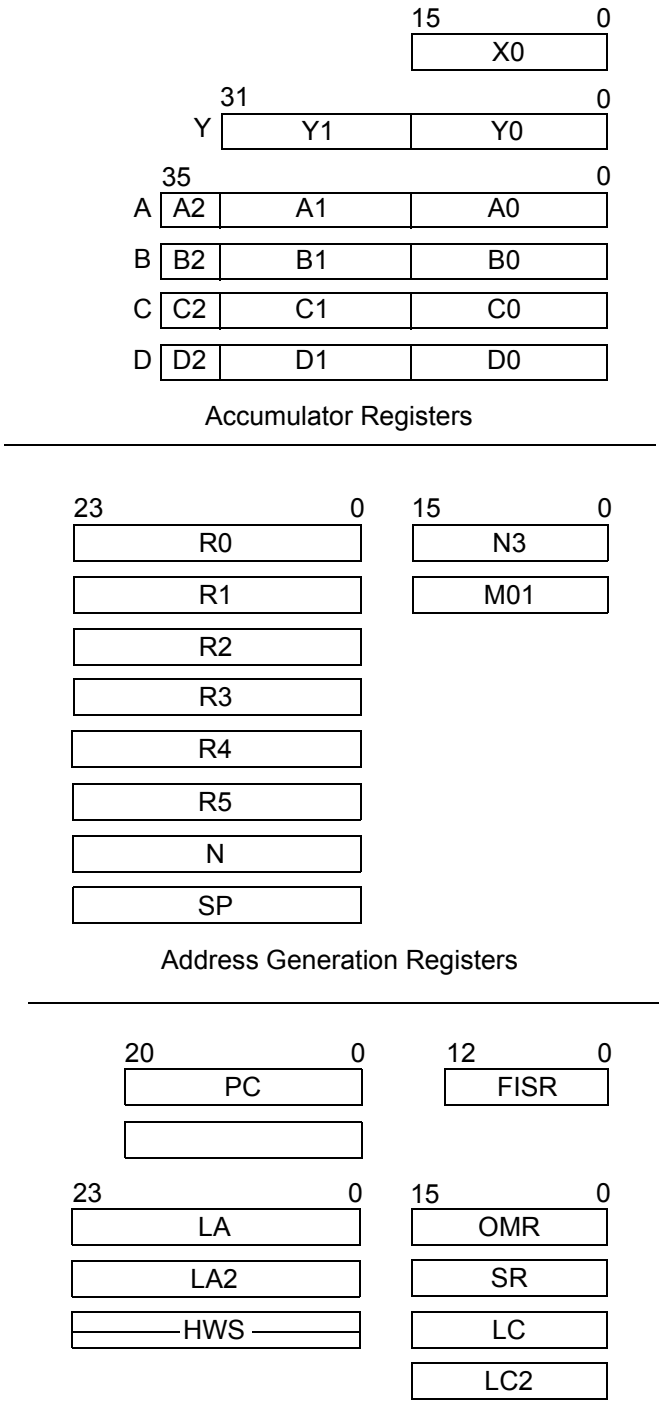

Figure 3-4. 56800E Programming Model

3.2 Instruction Set

The instruction sets of MCUs can be divided into functional groups such as those in the following list. Each of these groups will be studied in more detail, with emphasis given to the similarities as well as the differences between the HCS12/HC16 family and the 56800/E family of processors.

- Data movement instructions
- Mathematic instructions
- Bit test and manipulation instructions
- Shift and rotate instructions
- Program control instructions
- Indexing and address extension instructions
- Stack instructions
- Condition code instructions
- DSP instructions
- Stop and wait instructions
- Background mode
- Unique instructions, including fuzzy logic

3.2.1 Data Movement Instructions

The data movement instructions include instructions to load, move, store, transfer, and exchange data from a register, memory, or peripheral to another register, memory, or peripheral.

The HCS12/HC16 instruction sets specifies the load, store, and move instructions as independent mnemonics, whereas the 56800/E combines them into a single move mnemonic. However, they still perform essentially the same functions. The HCS12 provides an indirect indexed addressing mode that none of the other cores support. In indirect indexing mode, the index register contains the address of the memory location that contains the address of the memory location to be acted on. This is useful when dealing with a table of addresses.

One of the main differences between the two families is that the HCS12/HC16 MCUs do not have parallel data buses and therefore do not support parallel data moves. The 56800/E processors have three independent data and address buses, one for program fetches and two for data accesses. The 56800/E can execute an instruction, e.g., MAC, and read the next two 16-bit operands in a single instruction cycle.

The data sizes and memory organization supported by the four processors are listed in **Table 3-1**. The Memory Organization column shows the byte ordering of each architecture. The phrase “Little-endian” indicates that the Least Significant Byte (LSB) occupies the lowest address space, and the phrase “Big-endian” indicates that the Most Significant Byte (MSB) occupies the lowest address space.

Table 3-1. Supported Data Sizes

Core	Data Size Supported	Address Space	Memory Organization
M68HCS12	Byte, Word[1]	64K Bytes[2]	Big-endian (Unified)
M68HC16	Byte, Word, Long Word	1M Bytes[3]	Big-endian (Unified[4])
56800	Word[5]	64K Words Program 64K Words Data	Little-endian (Dual Harvard)
56800E	Byte[6], Word, Long Word	4M Bytes Program 32M Bytes Data	Little-endian (Dual Harvard)

1 The M68HCS12 supports 32-bit results for certain instructions, but no 32-bit memory accesses.

2 HCS12 devices can support a paged memory scheme to increase memory space and address more than 64K Bytes when using external memory. This applies to derivatives that have an external memory interface.

3 Configured as 16 banks of 64K bytes each.

4 The M68HC16 can be configured for independent access to program space and data space.

5 The 56800 supports 36-bit results, but all memory access is word-oriented.
word = 16-bit
long word = 32-bit

6 The 56800E uses byte and word pointers to access byte and word addresses, respectively.

Both families support all of the standard addressing modes needed. **Table 3-2** lists the addressing modes of each processor and some examples of each. The HCS12 is the only core which supports pre-decrement and pre-increment as well as indexed indirect addressing modes. The 56800/Efamily supports parallel data moves for most of these addressing modes.

Table 3-2. Addressing Modes and Examples

Addressing Mode	M68HCS12	M68HC16	56800	56800E
Register Direct				
Any Register	ABA	ABA	ADD B, A	ADD B, A
Register Indirect				
No Update	LDAA 0, X	LDAA 0, X LDAA E, X	MOVE X: (R0), A	MOVE.W X: (R0), A
Post-increment	LDAA 2, X+ MOVW 2, X+, $1000	MOVW 2, X, $1000	MOVE X: (R0)+, A	MOVE.W X: (R0)+N, A
Post-decrement	LDAA 2, X- MOVW 2 , X-, $1000	MOVW -2, X, $1000	MOVE X: (R0)-, A	MOVEW X: (R0)-, A

Table 3-2. Addressing Modes and Examples (Continued)

Addressing Mode	M68HCS12	M68HC16	56800	56800E
Pre-increment	LDAA 2, +X MOVW 2, +X, $1000			
Pre-decrement	LDAA 2, -X MOVW 2, -X, $1000			
Post-update by offset register			MOVE X: (R0)+N, A	MOVE.W X: (R0)+N, A

Indexed	LDAA 2, X
	LDAA B, X
	MOVW 2, X, $1000
		LDAA 2, X	MOVE X: (R0+N)
			MOVE X: (R0+2)
				MOVE.W X: (R0+N)
				MOVE.W X: (R0+2)

Indexed Indirect	LDAA [2, X] LDAA [D, X]			
Immediate				
Immediate	LDAA #2	LDAA #2	MOVE #2, A	MOVE.W #2, A
Absolute				
Absolute	LDAA $1000	LDAA $1000	MOVE X: $1000, A	MOVE.W X: $1000, A

3.2.2 Mathematic Instructions

The math instructions include addition and subtraction; binary-coded decimal; compare and test; multiplication and division; decrement and increment; clear; complement and negate; and boolean logic.

Both families fully support addition and subtraction and signed and unsigned multiplication and division. The HCS12/HC16 family supports 8-bit and 16-bit math, with 32-bit results for multiplication and 32-bit dividends for division. The 56800/E support 16-bit and 32-bit math, with 32-bit results for multiplication and 32-bit dividends for division, and has facilities for extended precision. Both families support integer and fractional multiply and divide instructions.

The main difference in the divide instruction between the two families is that the 56800/E provides a divide iteration instruction, which produces one bit of result per iteration. If a 16-bit result is to be produced, then the divide iteration must be executed 16 times. The divide instruction on the HCS12/HC16 family is a multi-cycle instruction that always produces a 16-bit result. The divide instruction on HCS12/HC16 can use either a 32-bit dividend or a 16-bit dividend. The divide iteration on the 56800/E always uses a 32-bit dividend.

The main difference in the multiplication instructions between the two families is the extended precision utilities and the rounding utilities that the 56800/E provide, e.g., MPYSU, MPYR, etc.

Both families support a Multiply and Accumulate (MAC) instruction for implementing DSP applications. The main difference here is the number of instruction cycles required to execute a MAC iteration, including fetching the data for the next iteration.

The HCS12 EMAC instruction fetches both operands, multiplies them together, then adds the result to an accumulator register. It does not update the operands' addresses and, therefore, requires additional instruction cycles for block data operations. The HCS12's MAC instruction assumes integer math. The HC16's MAC instruction multiplies the two operands together, adds the resulting 32-bit value to an accumulator register, then updates the operands' addresses and fetches the next two operands from memory. The HC16's MAC instruction assumes fractional math. The HC16 also has a repeat MAC instruction (RMAC) that reduces the overhead of the iteration, making the process more efficient.

The 56800/E family has a MAC instruction that multiplies two 16-bit operands together and adds the 32-bit result to an accumulator register while simultaneously fetching the next two operands with address update. This all happens within a single instruction cycle. Both of these cores support a repeat instruction (REP) that removes looping overhead. The final result is one instruction cycle per MAC iteration. The 56800's MAC instruction assumes fractional math, but the 56800E has both a fractional MAC and an integer IMAC instruction.

The HC16 supports data saturation, an important feature when processing large blocks of data with the MAC instruction, and convergent rounding as part of the DSP instruction set. Rounding occurs during a transfer from AM to E registers. The 56800/E supports data saturation as well as data limiting (data limiting occurs as a result of a move from an accumulator when the extension register is in use) and both convergent and two's compliment rounding. The 56800/E processors have several instructions with rounding functions built in for efficiency, such as the Multiply-Accumulate-Round instruction (MACR) and the Multiply-Round instruction (MPYR).

The HCS12/HC16 family supports extended precision addition and subtraction via the add and subtract through the carry bit, but does not have instructions which directly support extended precision multiply operations. The 56800/E family has instructions specifically designed to support extended precision operations to any length, using a mixed mode signed-unsigned multiply instruction, such as MPYSU or IMACSU.

The HCS12/HC16 family has instructions for handling binary coded decimal (BCD), which the 56800/E does not directly support.

Both families fully support compare and test operations on registers and memory, as well as decrement, increment, clear, complement and negate.

Some of the differences are that the 56800/E processors have an absolute value instruction for generating magnitudes, which the HCS12/HC16 family does not, but the HCS12/HC16 family has a minimum and maximum instruction, which the 56800/E family does not. The 56800/E supports a conditional transfer instruction for efficient block data searching.

Both families effectively support the necessary boolean logic functions.

3.2.3 Bit Test and Manipulation Instructions

Both families fully support bit clear, bit set, and bit test instructions; however, the 56800/E also supports bit change instructions for toggling bits.

3.2.4 Shift and Rotate Instructions

The main difference between the two families with respect to bit manipulation is that the 56800/E family supports single-cycle multi-bit shift instructions via an arithmetic multi-bit shifter.

The 56800/E processors also have a normalize iteration instruction (NORM) to aid in floating-point operations. The 56800E has increased the efficiency of normalizing even more by implementing a Count-Leading-Bits (CLB) instruction that returns the number of bit shifts needed to normalize and is used directly by the multi-bit-shift instruction. This reduces the normalize process to a two-cycle operation.

3.2.5 Program Control Instructions

The program control instructions are similar between the two families. They both have short and long branch, short and long conditional branch, short and long bit test and branch conditionally, and short and long branch to subroutines.

In addition to the generic set of branching instructions, the 56800/E family also supports zero overhead looping. The REP instruction will repeat a single instruction a specified number of times. This is effective for performing block MAC operations or bit shift operations. The DO instruction will repeat a block of instructions a specified number of times with no overhead and is useful for any time-critical loop in which the looping overhead is significant. The 56800E supports up to two nested hardware do-loops. The HC16 provides a repeat MAC instruction (RMAC) to reduce iteration overhead for the MAC.

The 56800/E cores achieve single cycle execution by using an instruction pipeline that allows parallel fetch-decode-execute operations. The instruction pipeline of the 56800E is deeper than the 56800, with about eight stages, allowing it to achieve one full instruction cycle in a single core clock cycle. The deep pipeline, however, does not come without some penalty. For example, when executing conditional branches, the pipeline's execution unit can be flushed, causing the core to stall while it refills. To help alleviate this problem, the 56800E has introduced delayed instructions (delayed branches and delayed returns from subroutines/interrupts) which allow the core to execute one or more instructions following the delayed instruction before execution is switched.

3.2.6 Indexing and Address Extension Instructions

The HCS12 has two index registers and the HC16 has three. These index registers can be added to a constant or to an accumulator to form the final address. The HC16 also provides a 4-bit extension register for each of the three index registers, resulting in a 20-bit address field.

The 56800/E family does not have index registers per se; instead, it makes use of a general-purpose set of address registers (the 56800 has four; the 56800E has six) and offset registers (the 56800 has one, the 56800/E has two) which are accessed and modified using the move instruction and two AGUs.

The 56800/E processors contain two Address Generation Units (AGUs): one for complex and modulo address calculations and the other for simple increment/decrement address update. Both units are used during dual parallel data moves.

Both of these families provide addressing modes that allow an immediate offset from an address or a variable offset (value contained in a register) from an address. However, only the HCS12 provides for an indirect address mode (the value that is read from memory is then used as the address of the operand). The HCS12 is also the only core with pre-increment and pre-decrement capability.

Address extension refers primarily to the HC16, because it uses 4-bit extension registers to extend the addressing range to 20 bits. Special instructions are required to manipulate these registers. The HCS12 and 56800 are 16-bit address machines and therefore do not use extension registers, while the 56800E's address registers are 24 bits in length and do not need extension registers.

3.2.7 Stack Instructions

The main difference between the families here is that the stack grows down for the HCS12/HC16 and grows up for the 56800/E. It is also interesting to note that the stack pointer points to the last location used for the 56800/E family as well as for the HCS12, but points to the next available location for the HC16.

3.2.8 Condition Code Instructions

The HCS12 has a very basic set of condition codes that it uses for decision making, i.e., conditional branching. The available flags on the HCS12 are the Half carry, Negative, Zero, oVerflow, and Carry, corresponding to H, N, Z, V, and C, respectively. In addition to these flags, the HC16 introduces the accumulator M oVerflow, Extension bit oVerflow, and the Saturate Mode, corresponding to MV, EV, and SM, respectively. With these additional flags, the condition codes for the HC16 can detect overflows in the M accumulator, M accumulator extension register in use, and can force saturation when the M accumulator overflows.

The 56800/E includes the same basic set of condition flags as the HCS12/HC16 and additional flags. The overflow flag on the 56800/E families is a general-purpose overflow flag which can apply to any general-purpose data register. The new flags that have been introduced in this family are the Loop Flag, SiZe, Limit, Extension in use, and Unnormalized, corresponding to LF, SZ, L, E, and U, respectively.

The loop flag indicates whether the hardware do-loop is active or not. This is needed to preserve the do-loop through interrupts and also for nested do-loops on the 56800E. The size flag is used to predict impending data overflow before it actually happens, by comparing bits 30 and 29 in an accumulator. The limit flag detects the occurrence of auto limiting when moving data from a register to memory. The extension flag indicates that the extension register associated with an accumulator is in use; in other words, the data is larger than 32 bits. And, finally, the unnormalized flag indicates whether the value in a register is between 1 and 0.5 for positive numbers, and between -1.0 and -0.5 for negative numbers. This is useful for floating-point implementations.

Both families update the condition codes for all math-related instructions and all other instructions that make sense. However, the HCS12/HC16 family will update the condition codes as a result of move instructions as well. The 56800/E does not modify the condition code register as a result of any move instruction.

3.2.9 DSP Instructions

This section reviews the similarities and differences between the DSP-specific instructions for both families.

At the heart of every DSP algorithm is the Multiply and Accumulate (MAC) instruction. It is used for implementing filters, modems, echo cancellers, voice compression algorithms, motor control algorithms, etc. All of these cores support the MAC instruction to some degree.

The HCS12 supports the Multiply and Accumulate instruction via the EMACS instruction. The EMACS instruction will multiply two 16-bit operands stored in memory and accumulate a 32-bit result in a third memory location. EMACS is a signed integer operation. The two 16-bit operands are fetched from memory using the two index registers, X and Y, and the 32-bit result is stored in a memory-based accumulator pointed to by the extended address supplied in the instruction. The EMACS requires 13 instruction cycles to execute.

The HC16 has a set of instructions to control the Multiply and Accumulate unit that is included in the HC16. The MAC instruction on the HC16 will multiply two 16-bit operands stored in registers and accumulate the 32-bit result into a 36-bit accumulator register. It will then update the index registers, IX and IY, to point to the next two operands and fetch them from memory into the HR and IR registers. The Multiply and Accumulate unit uses signed fractional math. This instruction requires 12 instruction cycles to complete, but includes the operand fetch and address update as part of the process. Power-of-two modulo buffers are also supported via the XMSK and YMSK registers. The HC16 also has a repeat MAC instruction (RMAC), further reducing the looping overhead associated with DSP algorithms.

The MAC instruction is an intrinsic property of the 56800/E family and is therefore very efficient. Both of these cores allow a single instruction cycle MAC, which will multiply two 16-bit operands together and accumulate a 36-bit result, then fetch the next two operands from memory and update the operand address pointers. This instruction executes in one instruction cycle. One of the pointers can use modulo math during update and is not restricted to power-of-two (the start address must be power-of-two, but the size can be any size within a normal addressing range). This family also has a repeat instruction (REP) that can be applied to the MAC, resulting in a single instruction cycle per MAC for the inner core of a filter, for example.

3.2.10 Stop and Wait Instructions

Both families provide a Wait and a Stop instruction that puts the chip into “low power mode” and “very low power mode”, respectively.

3.2.11 Background Mode

Both families have a background or debug mode to facilitate system development. They both have dedicated hardware on the chip for debug mode processing, including breakpoint handling, register read/write and memory read/write.

The 56800/E includes two modules: the On-Chip Emulation (OnCE) module, and the Test Access Port (TAP, commonly called the JTAG port), providing board and chip-level testing and software debugging capability.

3.2.12 Unique Instructions

There are a few instructions that are unique to the families. For example the, HCS12/HC16 family directly supports Binary-Coded Decimal (BCD) math via a half carry bit in the status register and the Decimal Adjust Accumulator (DAA) instruction. The 56800/E family has no BCD-specific instructions.

The HCS12 supports several special-purpose functions, such as fuzzy logic and table interpolation, which none of the other processors support. These functions would have to be implemented in software when porting to the 56800/E cores.

The HCS12 also has some unique and useful looping primitives that will decrement or increment and branch on condition that none of the other processors have. However, the 56800/E family supports zero-overhead looping, which can be used to replicate the performance improvement of the HCS12 looping primitives.The DO instruction of the 56800/E family provides a mechanism for repeating a block of instructions a specified number of times, with no test and branch condition needed. The REP instruction also supports zero-overhead looping, but will repeat only a single word instruction a specified number of times. Both of these instructions are useful for processing high-bandwidth data with minimal overhead.

3.3 Interrupts

In general, there are two types of interrupts: non-maskable and maskable. The non-maskable interrupts are typically associated with core events such as hardware reset, illegal instruction, Software Interrupt Instruction (SWI), etc. The maskable interrupts are associated with on-board peripherals and external Interrupt Requests (IRQs).

The number and type, as well as the programmability, of interrupts are device dependent, and the appropriate user manual should be consulted for specific information. Depending on the particular device, interrupts can be level-triggered or edge-triggered and programmed for various priority levels.

In addition to normal interrupt processing, the 56800E supports a fast interrupt that uses hardware shadow registers, providing a mechanism for handling high-speed data rates with minimal overhead.

3.4 Peripherals

Both of these families offer a wide range of peripherals to cover most applications. Obviously, when migrating from one device or family to another, it is important to find a device that offers the right combination of peripherals for the target application.

The 56800/E family offers a standard set of asynchronous and synchronous serial devices, including SPI, SCI, SSI, CAN. Additionally, it also offers a full set of timers, including watch dog timer, and A/D converters and pulse width modulators.

See the appropriate user manual for specific peripheral combinations.

3.5 Memory

The HCS12/HC16 family has a unified memory architecture that allows program and data to reside in the same memory space. However the 56800/E family has a dual Harvard architecture in which the program and data occupy unique memory spaces. The advantage of the dual Harvard architecture is that it allows simultaneous accesses to program and data, thereby increasing the throughput of the processor.

This split between program and data must be taken into consideration when determining which device best fits the needs of the target application. The application must be analyzed for program size and data size, determining both constant and variable storage requirements. There are several devices available that provide different combinations of program and data memory. Not only do the sizes vary, but the type of memory, e.g., RAM or Flash, also varies.

The type of memory required in the device, RAM or Flash, will be dependent on whether the device is targeted for a stand-alone application or part of a larger system. For the stand-alone case, the device will need some type of permanent memory, such as Flash, to store the program and constant data. On the other hand, if the device is part of a larger system, then it may be programmed via an external interface and only require temporary storage, such as RAM.

If external memory is needed, then the real-time requirements of the application must be considered. The HC16 and HCS12 have components with an external bus interface and, when using internal memory or external memory, the performance is similiar because of the nature of the Von Neumann bus architecture. The 56800/E family provides a single data port to the outside world via the EMI and, as a result, will serialize any program and/or data accesses to external memory. Thus, performance can degrade when transitioning from using internal memory to using external memory. This can be alleviated by linking the high overhead routines into internal memory, where the core will perform parallel accesses to the memory.

4. Software Development

Porting high-level C code from one processor family to another usually requires very little effort, as long as the destination processor has adequate tools and resources available. On the other hand, porting Assembly code can be a significant task, because Assembly language is processor-specific and most often requires a total rewrite of the code. Also, any device drivers that are used to interface to the on-chip peripherals are device-specific and usually must be rewritten.

Fortunately, to help reduce the porting task, the CodeWarrior family of software and hardware development tools supports both HCS12 and controller devices. The CodeWarrior Development Studio for Freescale 56800/E Controllers is a powerful and easy-to-use tools suite designed to increase software development productivity. CodeWarrior's Integrated Development Environment (IDE) provides features such as the Processor Expert™ (PE) application design tool, Stationery Wizard and project manager with templates that handle the device-specific details.

Processor Expert is an advanced, component-oriented, Rapid Application Development environment for embedded systems. PE is based on the "Embedded Beans" specification, which encapsulates the functionality of basic elements of embedded systems, including the CPU core, on-chip peripherals, stand-alone peripherals, virtual devices, programmable arrays, and pure software algorithms. Embedded beans facilitate porting, as many are available on all 16-bit platforms, enabling the application to interface to them in the same fashion, regardless of which physical processor the code is running on.

These tools allows the developer to focus on the application software, rather than wasting effort in learning device-specific details.

Another factor that greatly affects the effort involved in porting Assembly language from one processor to another is the destination device's available resources, e.g., CPU registers, RAM, ROM, MIPS, etc. The 56800/E family of controllers have a larger register set than the HCS12/HC16 MCUs, as well as significantly more MIPS, the number of real-time instruction available, which makes the task of porting assembly code much easier.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN1983
Rev. 0
07/2005