SH Series Overview

SuperH^{TM} RISC Engine Embedded Processors

HITACHI

Notice

When using this document, keep the following in mind:

1. This document may, wholly or partially, be subject to change without notice.
2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. MEDICAL APPLICATIONS: Hitachi's products are not authorized for use in MEDICAL APPLICATIONS without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant Hitachi sales offices when planning to use the products in MEDICAL APPLICATIONS.

Table of Contents

Section 1 Introduction Challenges of Next-Generation Applications 1
1.1 Demands on Microprocessors.. 3
1.2 Hitachi's SH Series Rises to the Challenge of Next-Generation Applications 5
1.2.1 Low Power Dissipation.. 6
1.2.2 High Integration... 6

Section 2 SH Series Family Architecture...9
2.1 Common Features 9
2.1.1 Register Set 11
2.1.2 Addressing Modes 14
2.1.3 Instruction Set.. 16
2.2 SH-1's Implementation of the Architecture... 19
2.3 SH-2 Implementation -- Additions to the Architecture 19
2.4 SH-3 Implementation--Additions to the Architecture 22
2.4.1 Memory Management in the SH-3 23
2.4.2 Additional Instructions in the SH-3.. 25

Section 3 Handling Exceptions Vector -- Table Organization..........................27

Section 4 Power-Down Modes -- CPU Operating States......................................31

Section 5 External Memory Control -- Buses..33

Section 6 Peripherals...35
6.1 Common Features.. 35
6.2 Peripherals and Memory on the SH-1 Devices.. 37
6.3 Peripherals on the SH-2 family... 39
6.4 Peripherals on the SH-3 family... 40

Section 7 Device Summary and Packaging..43

Section 8 Support Tools...47
8.1 Hitachi Tools.. 47
8.2 Third-party Vendors.. 51

Section 1 Introduction Challenges of Next-Generation Applications

Recent years have seen major advances in the capabilities of consumer electronics. These next-generation applications are typified by Personal Digital Assistants (PDAs), multimedia/entertainment systems and wireless communications devices. These emerging markets and applications represent an opportunity for unprecedented growth. At the same time, established markets and products, such as automotive and industrial applications or computer peripherals such as hard disk drives and telecommunications adapters, are briskly moving toward increased performance.

Consumers continue to expect more features, more capacity and higher communication rates. User interfaces with advanced features, such as handwriting and voice recognition, are becoming common expectations in advanced PDAs. Video games and multimedia applications demand higher resolution images, faster image-rendering, image-compression and image-decompression rates and higher sampling rates for synthetic audio. I order to provide higher data rates, reliability and security, communications devices must utilize complex coding and signal processing techniques. Increasing disk capacity demands higher areal density with faster an more-precise head-position-control loops.

These end-product needs place a common requirement upon the system design--greater processing power from the microprocessors and microcontrollers.

End-user expectations in these markets do not end with increased performance, however. At the same time, these products are expected to offer portability and low cost. Portability demands both compactness and battery operation. In turn, compactness demands high integration and high-density packaging; similarly, acceptable operating time on realistically-sized batteries requires low power utilization.

Historically, products were optimized for one parameter--performance, size or power. Often, optimization consisted of maximizing one parameter at the essentially unrestrained expense of the others. For example, engineering workstations are optimized for computational performance; within wide limits, physical size and power consumption of desktop machines are distant secondary considerations. For next-generation applications, however, all three parameters are squeezed simultaneously. While it is, of course, not possible optimize all parameters in any one design, an application-specific figure of merit, requiring suitably high marks in all three areas, must somehow be achieved.

Finally, the dynamics of these consumer markets dictate short product lifetimes. As a result, OEMs must get their products to the market expeditiously. It has been demonstrated that late entry into the market has a more significant negative impact on a product's bottom line than, for example, increased costs for development, manufacturing or marketing. Short product lifetimes and timely introduction require rapid product development.

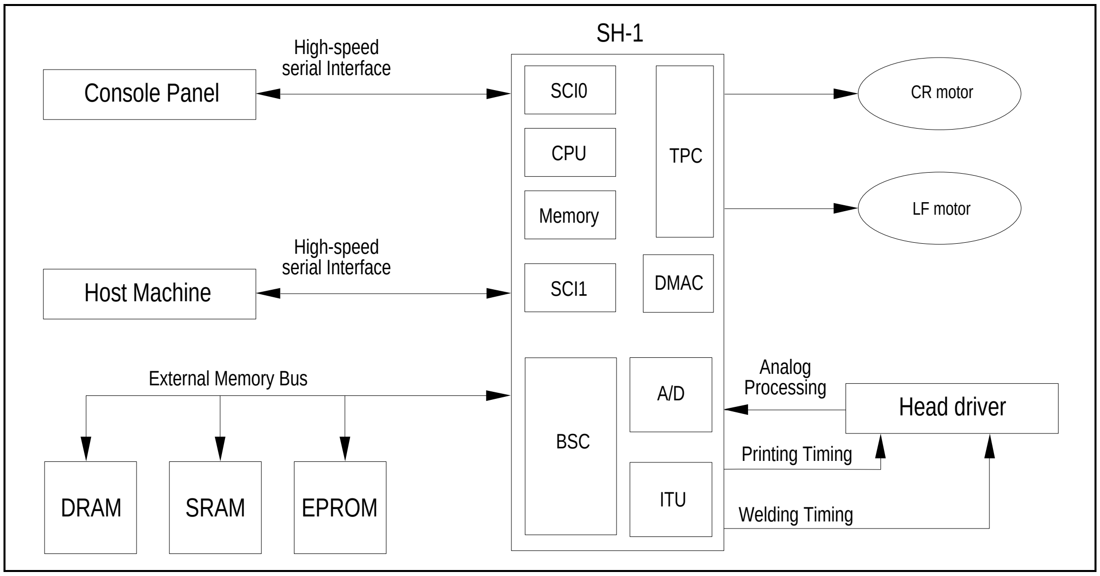


Figure 1.1 Advanced Printer Application

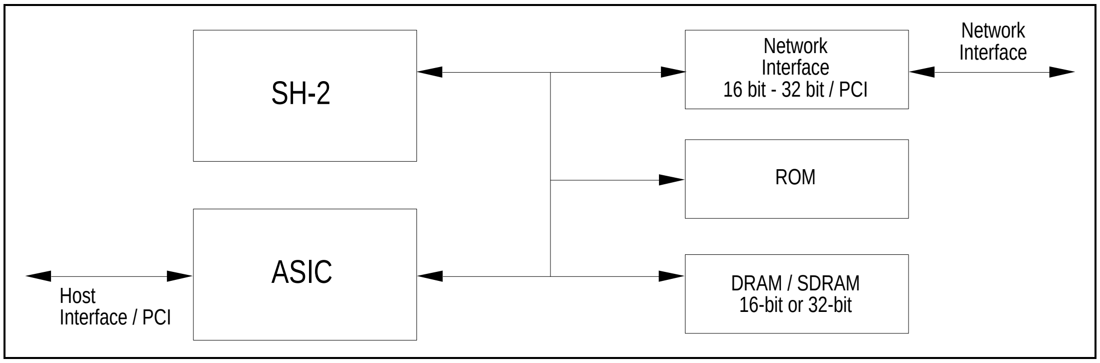


Figure 1.2 Network Interface Controller Application

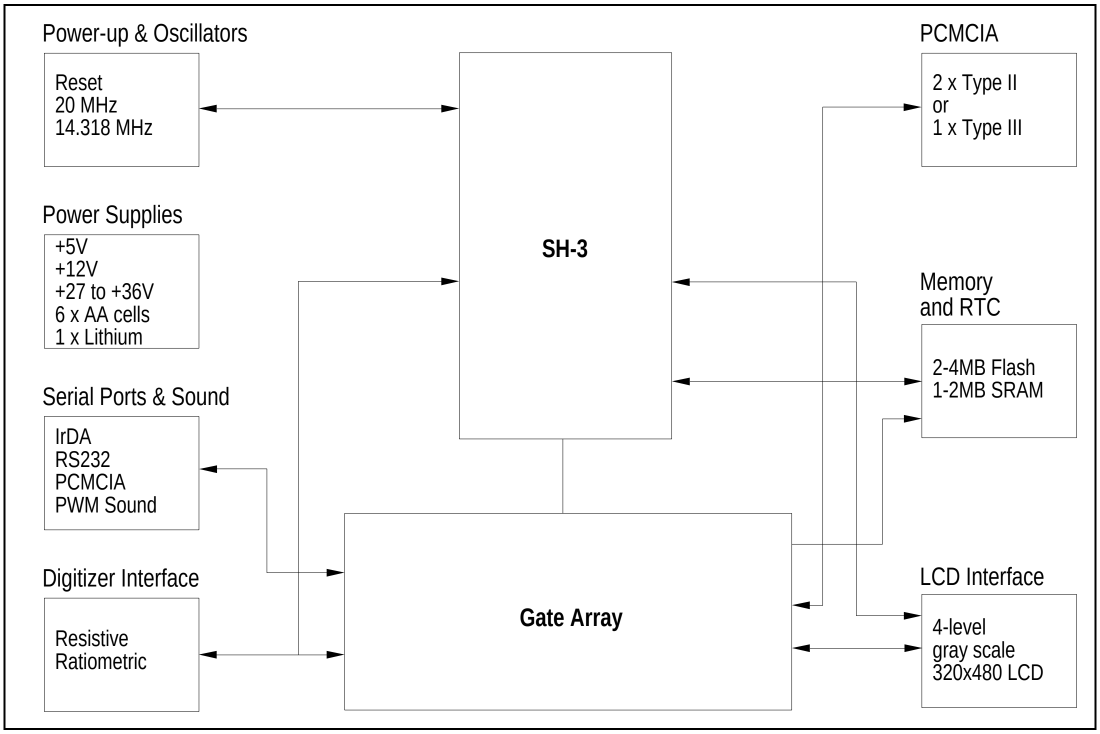


Figure 1.3 Personal Digital Assistant Application

1.1 Demands on Microprocessors

At the core of each of these advanced end-products is a microprocessor or microcontroller. Thus, the demands placed upon the end products apply to this processor as well. To be successful in this equipment, the processor must offer a certain level of performance while simultaneously satisfying the need for portability and low system cost. How do these needs translate into microprocessor requirements?

Next-generation applications demand high throughput from the processors chosen, dictating 32-bit architectures. To meet the performance requirements of these applications, high clock speed, high bus speeds, low clocks-per-instruction are demanded of the implementations.

Software complexity and time-to-market requirements demand efficient, straightforward programmability of the architectures. High-level languages (HLLs), long a staple of workstation design, are increasingly used in embedded applications; this places further requirements on the processor's architecture. A large, uniform address space must be provided, accessible via a wide selection of addressing modes. A rich choice of data types must be available to a powerful instruction set. Fast, flexible mechanisms must be provided for passing parameters and control between procedures/functions.

However, conventional instructions and addressing modes may not be sufficient in these 32-bit architectures. In addition, specialized instructions and subsystems may be needed in order to meet the throughput requirements. For example, digital signal processing (DSP) in the application may require specialized high-performance instructions, addressing modes and control structures; raster graphics might require specialized hardware capable of performing bit/pixel manipulation, BITBLT, line/curve drawing and clipping hardware.

Processing power must be achieved in the context of low power consumption; that is, absolute MIPs and MIPs/milliwatt are both important. Lower power dissipation requires a combination of a low-power fabrication process, clever circuit design and a variety of power-saving operating modes in the implementation.

Fast, wide, on-chip buses can be achieved more readily, and with less power, than off-chip buses; high performance with low power argues in favor of large on-chip memories. These may take the form of program and data caches, general-purpose RAM for data or EPROM/ROM for storage of programs and constants.

System-level compactness is also easier to achieve with large on-chip memories. In fact, it is desirable to implement as much as is usable on the same die as the processing core. Thus, commonly-used peripherals should be implemented on-chip; serial I/O, parallel I/O, DMA, timers, A/D converters and DMA are the most likely candidates for inclusion.

As software complexity continues to increase, uniformity and maintainability become increasingly important. Thus, support for, and availability of, operating systems become requirements. Operating-system robustness, particularly in a user-programmable product, depends upon hardware-based memory management, with its attendant inter-process isolation and protection.

Engineers faced with short product-design cycles demand ease of use from their processors. Naturally, integration and straightforward interfacing simplify the hardware design. However, software design represents an ever-increasing percentage of the intellectual content in these end products. Efficient software design requires the use of high-level languages with powerful debugging tools. Thus, the architecture should be supported by efficient compilers running on a variety of host platforms. Program debugging should be accomplished via high-level debuggers in cooperation with full-speed emulators.

Ideally, once the design team has committed to an architecture, the intellectual and monetary investment in learning and supporting that architecture should be applicable to a range of end products. Thus, the architecture should be uniform across a wide-ranging family; similarly, the tools should be usable across the entire family.

1.2 Hitachi's SH Series Rises to the Challenge of Next-Generation Applications

Hitachi designed the SH series of processors specifically to address these next-generation applications. The SH series currently consists of three families, SH-I, -2 and -3, each optimized for specific application segments. Future plans call for continued expansion and extension of the families, further increasing system-level performance while retaining upward compatibility.

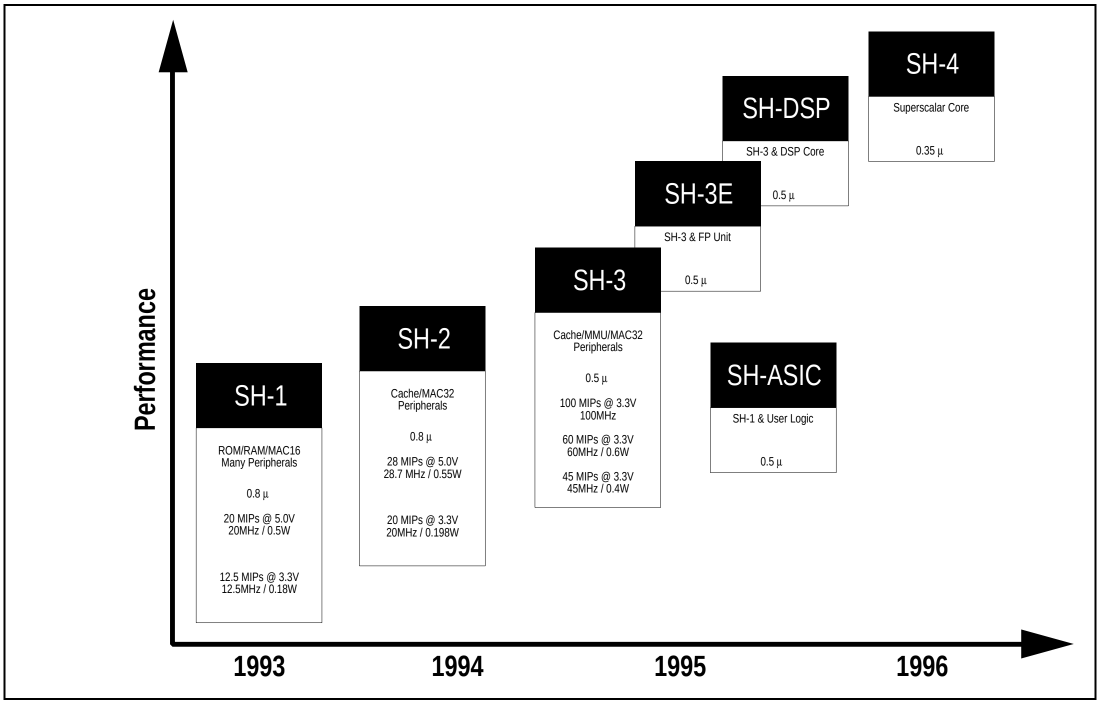


Figure 1.4 SH Growth Path by Engine

The SH series features an extensible 32-bit RISC architecture. Its basic form is realized in the SH-I family; the SH-2 and SH-3 families progressively extend the architecture. The common architecture includes 56 instruction types, composed of general-purpose instructions augmented by multiply-and-accumulate instructions for DSP applications. The SH-I devices (SH70xx) offer a high degree of memory and peripheral integration. The SH-2 family (SH76xx) adds cache, extended power-reduction functions and specific higher-performance instructions (a total of 62 instruction types); it is optimized for interfacing with external memories and peripherals, including tightly-coupled processors. The SH-3 (SH77xx) provides yet-higher performance in a bus-based context; its extensions include virtual-memory management, a more powerful cache and expanded instructions (a total of 66 instruction types). Future extensions to the series will add an FPU, superscalar implementations, graphics functionality and additional DSP features.

Being RISC processors, the SH-series devices process most instructions in a single clock cycle, through a five-stage pipeline. High clock speeds, up to 100 MHz for the SH-3, provide high peak instruction-execution rates. Fast on-chip memories, organized conventionally or as caches, ensure that high *average* instruction-execution rates are also maintained in cost-effective systems.

1.2.1 Low Power Dissipation

Recognizing that this blazing speed would be inaccessible by portable applications if power dissipation was excessive, Hitachi designed these families in low-power sub-micron CMOS processes with low-voltage capabilities. Low static operating current is stressed in all circuit designs; low dynamic (peak) currents are guaranteed by logic and circuit design. In the SH-3, for example, the cache incorporates a number of power-reducing features. First, the sense amplifiers utilize SRAM circuit-design techniques which reduce word-line voltage swings to 60 mV; this permits lower currents to be used without sacrificing speed. Second, a unique cache-way-enable circuit minimizes power demands by enabling the sense amplifiers only in that portion of the cache which has `hit' in a given access; this reduces the cache data array's operating current by 75%.

Further, all implementations include a variety of software-controlled power-reduction mechanisms; each family embodies a selection from a palette including standby and sleep modes, clock-speed control and selective module shutdown. For example, the SH-3 permits the clocks for the CPU, the on-chip peripherals and the external bus to be separately optimized. This flexibility permits the system designer to choose the optimum combination of low power and system responsiveness for each application; further, the final choice need not be made by the OEM, but can instead be fine-tuned by the *user*. Given the wealth of options, it is now practical to incorporate a variety of techniques into the product and to let the system's end user decide the optimal choice for the particular operating conditions.

The result? -- low operating and standby currents, leading to longer battery life. For example, the typical standby current in the SH-1 family's SH7034 is a mere trickle of 10 nanoamperes.

1.2.2 High Integration

Hitachi has met the dual needs of compactness and low system cost by incorporating a broad choice of on-chip memories and peripherals in the SH series. For example, various members of the SH-I family provide up to 64 Kbytes of ROM and up to 8 Kbytes of general-purpose RAM. The SH-2 provides 4 Kbytes of unified, 4-way set-associative cache. The SH-3 devices advance another major step, combining unified caches of up to 8 Kbytes with a 128-entry Translation Lookaside Buffer (TLB) for virtual-memory management.

Peripheral integration is aggressive in the SH series; this is particularly true for the SH-1 family. All members of this family provide a four-channel Direct Memory Access Controller (DMAC), two Serial Communications Interfaces (SCIs), five 16-bit timers and a separate watchdog timer. In addition, as many as 40 parallel I/O lines are provided on some devices. Consistent with its role as a host for feature-rich operating systems, the SH-3 incorporates a 100-year Real-Time Clock (RTC) with periodic- and alarm-interrupt capabilities.

In addition, all SH-series processors feature direct interfaces to external memories. The on-chip Bus State Controller, via the external bus, provides the specialized inputs, outputs and functions for different types, widths, and speeds of external memory. All devices in the series support static memory and dynamic RAM; where appropriate, refresh and high-speed-access modes are supported. In addition, the SH-2 and -3 can handle synchronous DRAMs, burst-mode ROMs and pseudo-static RAMs.

The future evolution of the SH series includes the SH-3E, the SH-DSP and the SH-4; these are marked by architectural extension, additional functional integration and process improvements. The SH-3E extends the SH-3 family primarily by adding an FPU. Enhanced digital-signal-processing capability is implemented in the SH-DSP. The escalating throughput requirements of future systems will be met by the superscalar core of the SH-4; its 0.35 µm implementation will provide more than 300 MIPs.

For markets that require a more tailored solution, application-specific microprocessors can be created around the SH cores. Hitachi's HG71C series combines the SH-I microprocessor with a cell-based implementation of the user's system logic; these MicroCore CBIC devices range in complexity from 100K to 350K gates. Standard cells include analog and digital functions; a memory compiler generates optimized RAM and ROM blocks. Simulation and emulation tools are available to support all phases of the design process.

Section 2 SH Series Family Architecture

All devices in all families in the SH series employ a common 32-bit RISC architecture, designed specifically by Hitachi to meet the needs of next-generation applications. This architecture is implemented in the SH-1 family; the SH-2 and SH-3 families feature upward-compatible extensions to this baseline.

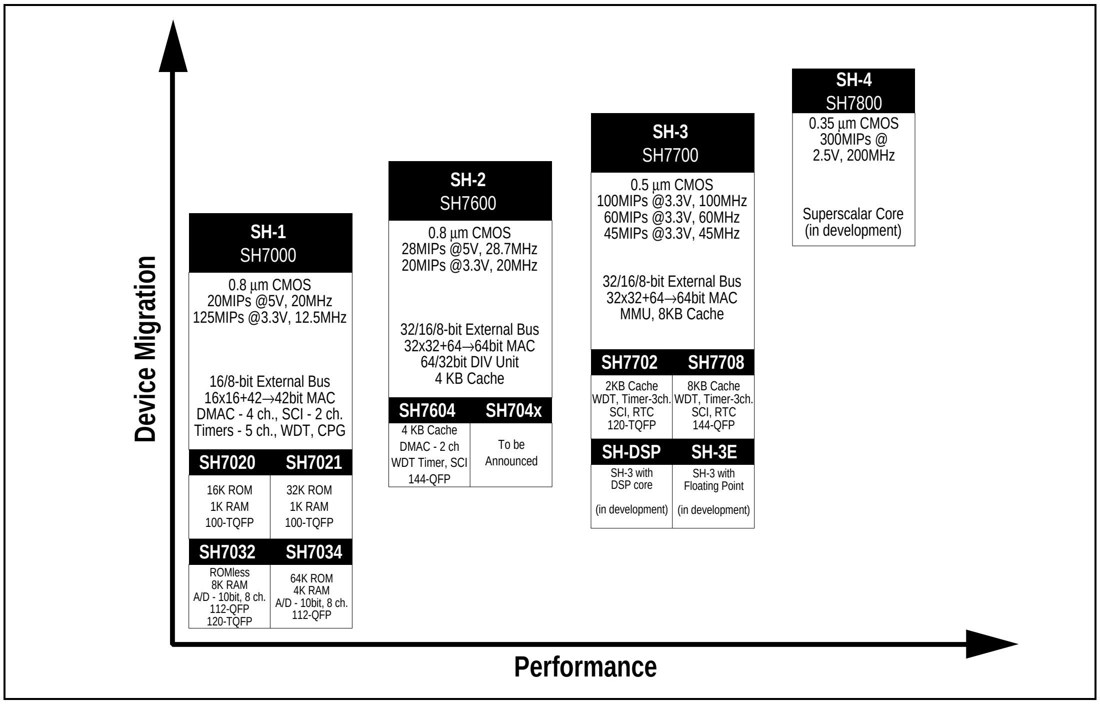


Figure 2.1 SH Growth Path by Device

2.1 Common Features

Significant features of the common RISC architecture include a large, uniform logical address space, a load-store methodology and a high-level-language orientation. The instruction set is encoded in 16 bits for high code density; memory-access instructions utilize a large selection of addressing modes, while general-purpose instructions (logical and arithmetic, for example) operate on 32-bit integers stored in a large set of registers.

A five-stage pipeline, featuring a hardware multiplier, is used in all implementations. Since compiler-generated code exhibits a high frequency of non-sequential fetches, such as branches and calls, a delay-slot mechanism has been provided, reducing pipeline-breakage penalties. Internal data paths are 32 bits wide in all implementations.

The SH series' memory organization provides a 4-gigabyte, byte addressable, uniform (unsegmented) logical address space. Operands may be stored as bytes (8 bits), words (16 bits) or longwords (32 bits). Universally, memory-resident operands can be stored in big-endian format -- that is, the most-significant byte of an operand is stored in the byte at the base address of the operand; successively less significant bytes, if any, of the operand are stored in consecutive bytes at successively higher memory addresses. In addition, the SH-2 family can support little-endian format in a portion of its memory space; the SH-3 family can support little-endian format in its entire memory space.

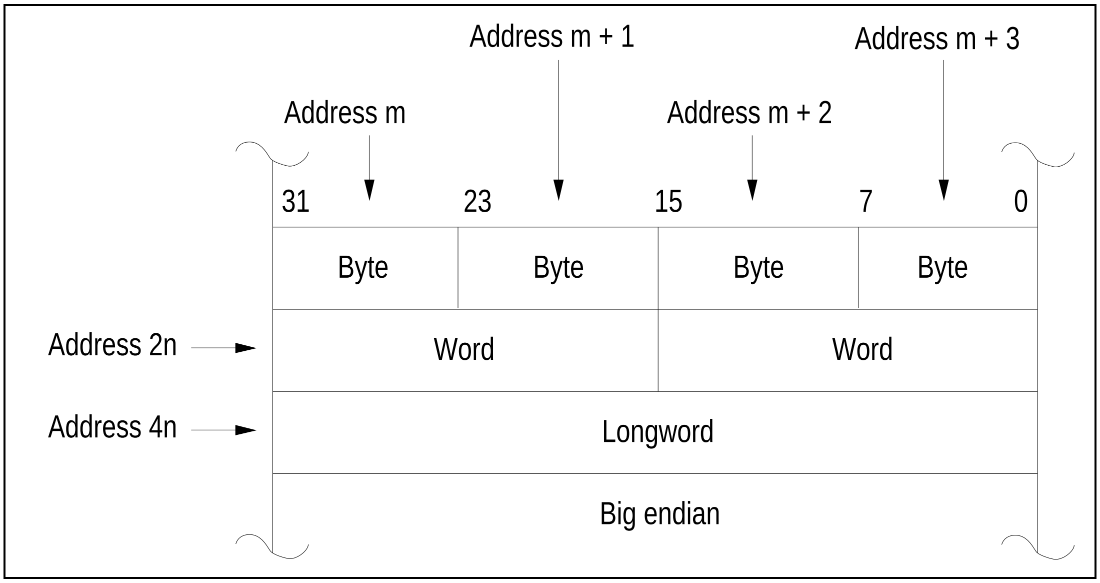


Figure 2.2 Big-Endian Storage of Operands in Memory

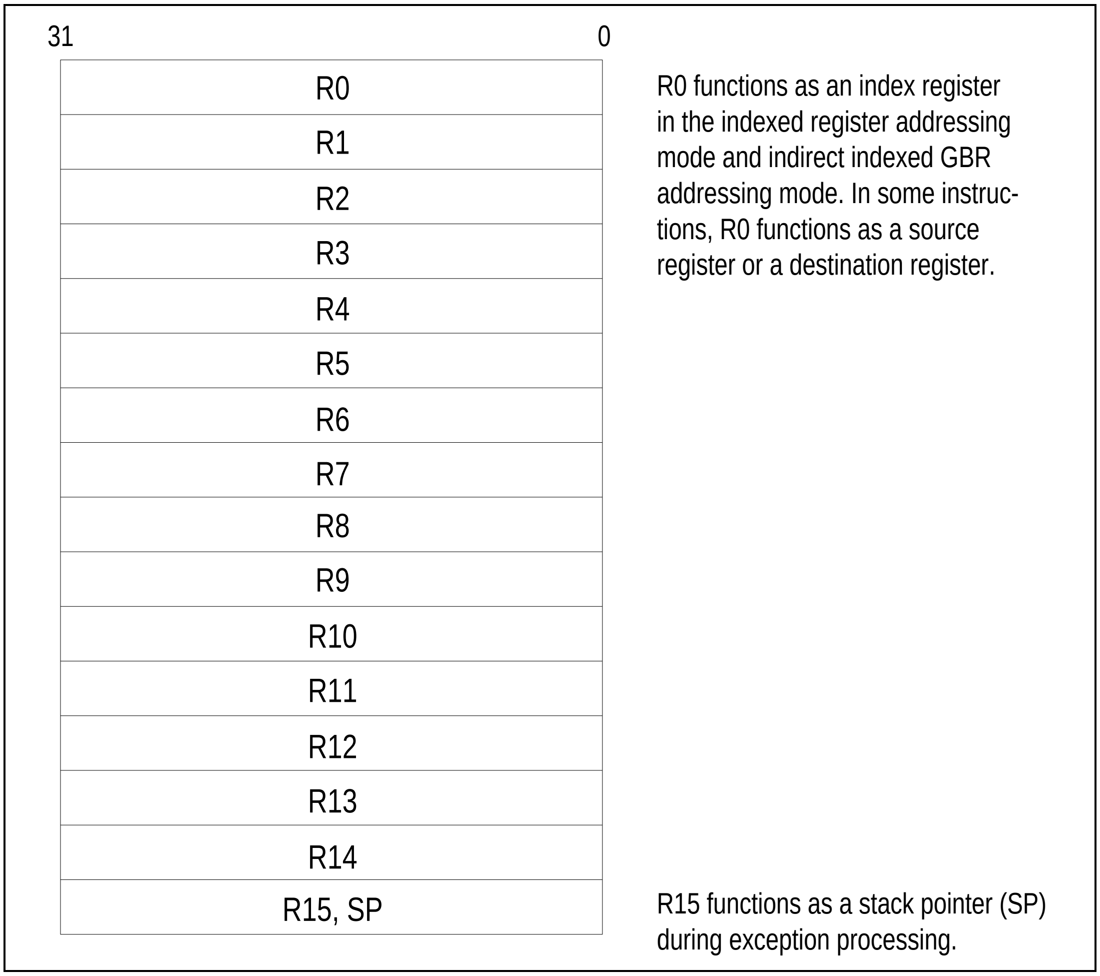
Memory-resident operands must be aligned at their natural boundaries, according to their length; accesses which violate this rule are not guaranteed to produce correct results. This means that a byte-length operand may be stored at any location. Words must be stored at locations with even base addresses; longwords may be stored only at base addresses which are multiples of 4.

2.1.1 Register Set

General-Purpose Registers

The 16-bit-encoded RISC instruction set operates between any two registers chosen from a bank of 16 general-purpose 32-bit registers. Orthogonality is maintained across the register bank. However, two of the registers have additional implicit functions. Specifically, R15 is also used as the Stack Pointer (SP), for those addressing operations which use the stack. In addition, R0 is used as an index register for two of the addressing modes; it is also used as a source or destination register for certain instructions.

Register-resident operands are always treated as longwords. Memory can be read or written on a byte, word or longword basis; when a byte- or word-length load is executed from memory, the operand is sign-extended into the 32-bit register.



R0 functions as an index register in the indexed register addressing mode and indirect indexed GBR addressing mode. In some instructions, R0 functions as a source register or a destination register.

R15 functions as a stack pointer (SP) during exception processing.

Figure 2.3 General-Purpose Registers

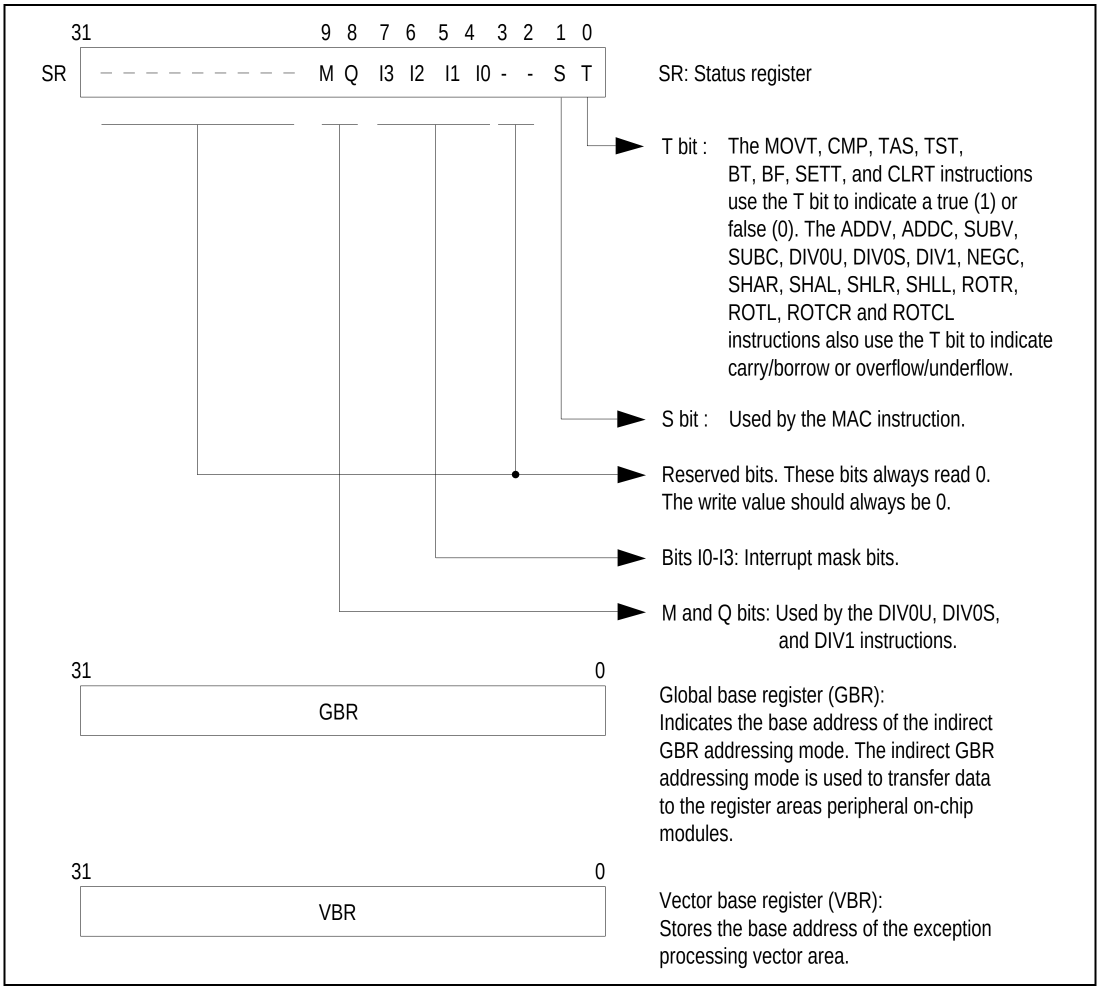
Control Registers

The Control Registers store the information that defines the processing state of the CPU; they also provide base addresses for certain operations or addressing modes. They consist of the Status Register (SR), the Global Base Register (GBR) and the Vector Base Register (VBR).

The SR contains operating status, which is updated by instruction execution and by external events and which determines, in part, the execution of succeeding instructions and the CPU's response to future events. The SR contains, for example, the T-bit, a result-flag from logical tests and a test-bit for conditional branches; it also includes the interrupt mask bits.

The GBR contains the base address used for the GBR-based addressing modes. These addressing modes provide a compact address format for accesses within an `address window' which can be placed anywhere in the logical address space; access within the window can be through a displacement or a general-purpose register.

The VBR points to the base of the Vector Table, which dispatches the service routines for exceptions and interrupts.



SR: Status register

T bit : The MOVT, CMP, TAS, TST, BT, BF, SETT, and CLRT instructions use the T bit to indicate a true (1) or false (0). The ADDV, ADDC, SUBV, SUBC, DIV0U, DIV0S, DIV1, NEGC, SHAR, SHAL, SHLR, SHLL, ROTR, ROTL, ROTCR and ROTCL instructions also use the T bit to indicate carry/borrow or overflow/underflow.

S bit : Used by the MAC instruction.

Reserved bits. These bits always read 0. The write value should always be 0.

Bits I0-I3: Interrupt mask bits.

M and Q bits: Used by the DIV0U, DIV0S, and DIV1 instructions.

Global base register (GBR): Indicates the base address of the indirect GBR addressing mode. The indirect GBR addressing mode is used to transfer data to the register areas peripheral on-chip modules.

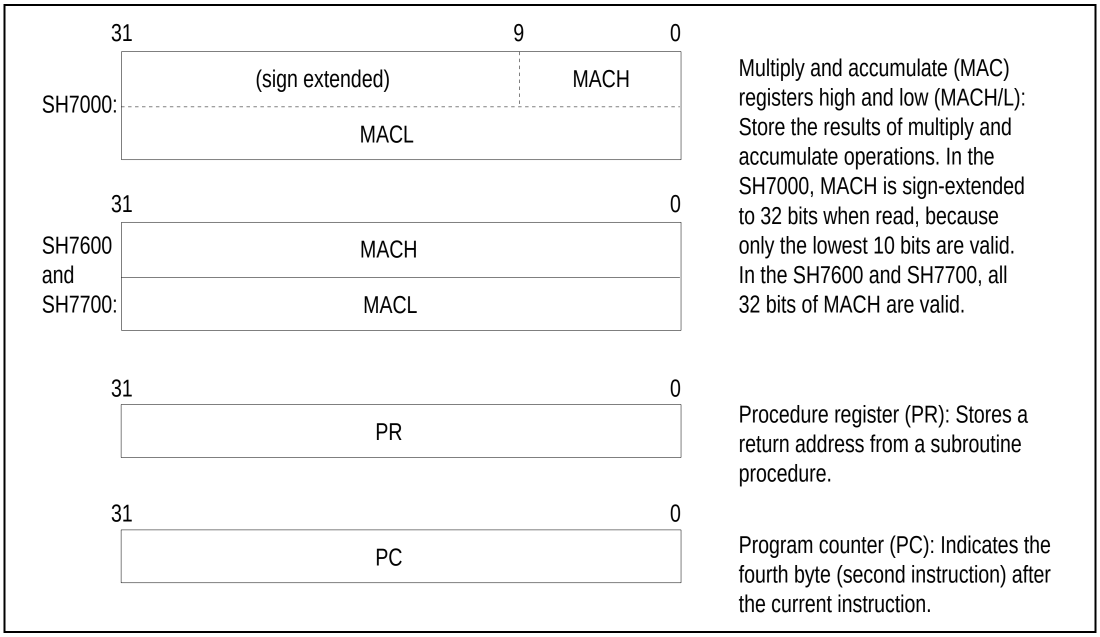
Vector base register (VBR): Stores the base address of the exception processing vector area.

Figure 2.4 Control Registers

System Registers

The System Registers include, among others, the registers that maintain the correct instruction-execution sequence. These registers include the Program Counter (PC), which points to the current instruction, and the Procedure Register (PR), which saves the return address during subroutine calls. Two other registers are included in this set, the Multiply-And-Accumulate Registers (MACH and MACL).

The MACH and MACL registers serve as one of the sources and as the destination for all multiply-and-accumulate (MAC) instructions; each of these registers is 32 bits long. Concatenated, they are architecturally capable of containing a 64-bit value; however, the valid range of the operands they may contain is family-dependent.



Multiply and accumulate (MAC) registers high and low (MACH/L): Store the results of multiply and accumulate operations. In the SH7000, MACH is sign-extended to 32 bits when read, because only the lowest 10 bits are valid. In the SH7600 and SH7700, all 32 bits of MACH are valid.

Procedure register (PR): Stores a return address from a subroutine procedure.

Program counter (PC): Indicates the fourth byte (second instruction) after the current instruction.

Figure 2.5 System Registers

2.1.2 Addressing Modes

Effective support of high-level languages requires a flexible set of memory-addressing modes; however, compact instruction coding constrains the total number of addressing modes. The SH architecture includes 12 judiciously-chosen addressing modes, illustrated below. The addressing modes have been carefully selected to permit efficient code generation by modern compilers.

Operands in registers can be addressed directly (*direct register*). Registers may also be used as pointers to memory (*indirect register*); for this mode, an optional pre-decrement or post-increment supports stack operations (*pre-decrement indirect register or post-increment indirect register*).

Registers may also be used as pointers to a base address, with an offset computed from a 4-bit displacement encoded in the instruction (*indirect register w/displacement*); the always-positive displacement is scaled by 1, 2 or 4 (left-shifted by 0, 1 or 2 bits), corresponding to the size of the operand being accessed -- byte, word or longword. This mechanism is well-suited for addressing elements of relatively small tables or arrays whose subscripts are known at compile-time. Alternatively, a base register may be added to R0 to provide the address (*indirect indexed register*); this permits accesses within an arbitrarily large structure for which the offset must be computed at run-time.

The indirect register modes, discussed in the previous paragraph, can also use the contents of the Global Base Register as their base address (*indirect GBR w/displacement and indirect indexed GBR*). These modes address the elements of statically-addressed data structures which are common to all procedures and functions; for example, they are suitable for accessing the VO control registers of the on-chip peripherals.

The Program Counter can also be used as a base-address pointer. In one mode (*indirect PC w/displacement*), the positive 8-bit displacement is scaled according to the size of the operand (byte, word or longword) and added to the PC, thus forming the memory address. Another mode (PC relative) scales a signed 8- or 12-bit displacement by 2 and adds it to the PC. All of these modes are designed to access constants stored in the code space of the procedure.

Finally, *immediate* addressing permits commonly-used small constants, 8-bit signed or unsigned as appropriate for the instruction, to be encoded compactly within the instruction.

Table 2.1 Addressing Modes and Effective Addresses

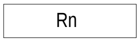
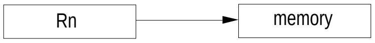
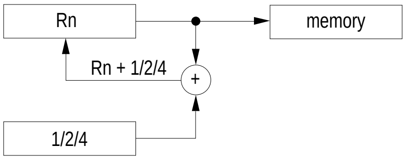
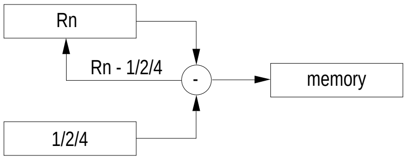
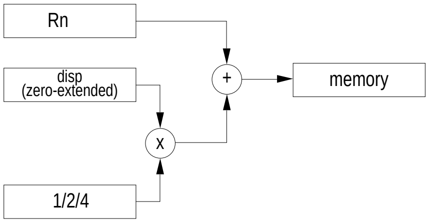
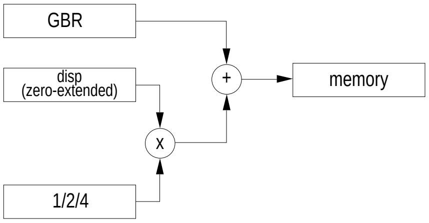
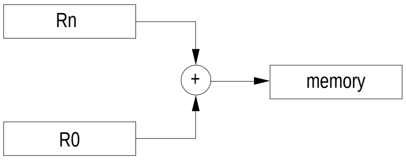
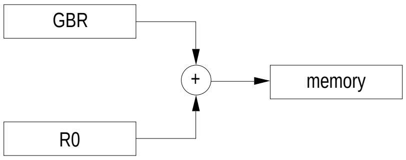
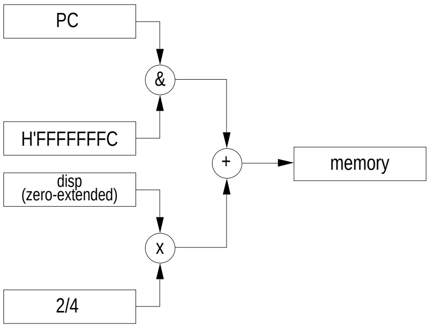
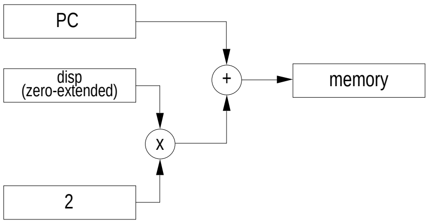
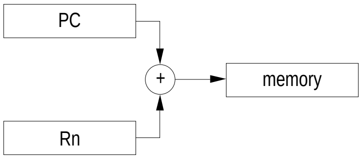
Addressing Mode	Addressing Mode
Direct register addressing: Rn  	Indirect register addressing: @Rn 
Post-increment indirect register addressing: @Rn +  	Post-decrement indirect register addressing: @ - Rn 
Indirect register addressing with displacement: @(disp:4, Rn)  	Indirect GBR addressing with displacement: @(disp:8, GBR) 
Indirect indexed register addressing: @(R0, Rn)  	Indirect indexed GBR addressing: @(R0, GBR) 

Table 2.1 Addressing Modes and Effective Addresses (cont)

Addressing Mode	Addressing Mode
PC relative addressing with displacement: @(disp:8, PC)  PC & H'FFFFFFFC + memory disp (zero-extended) x 2/4	PC relative addressing: disp:8, disp: 12 
Immediate addressing: #imm: 8 The 8-bit immediate data (imm) for the TST, AND, OR, and XOR instructions are zero-extended. The 8-bit immediate data (imm) for the MOV, ADD, and CMP/EQ instructions are sign-extended. Immediate data (imm) for the TRAPA instruction is zero-extended and is quadrupled.	PC relative addressing: Rn 

2.1.3 Instruction Set

The SH series' instruction set has been designed for effective support of high-level languages (HLLs). The individual instructions have been chosen to provide complete, efficient support; at the same time, great care has been exercised to exclude instructions which are difficult for compilers to use. The resulting instruction set features 16-bit encoding and permits compact code generation by today's compilers. The instruction set, composed of six *classes*, is summarized by type in Table 2.2.

Since the SH series features a RISC architecture, the arithmetic and logical instructions utilize operands stored in the general-purpose registers. Operations between memory and registers are restricted to load/store operations (data transfer), which transfer bytes, words or longwords.

Table 2.2 SH Instruction Set

			Applicable Instructions			
Class	**Operation Code**	**Function**	**SH-1**	**SH-2**	**SH-3**	**#Instructions / #Types**
Data transfer	MOV, MOVA, MOVT, SWAP, XTRCT	Data transfer	√	√	√	SH-1: 39/5 SH-2: 39/5 SH-3: 40/6
Arithmetic operations	ADD, ADDC, ADDV, CMP/cond, DIV1, DIVOS, DIVOU, EXTS, EXTU, MAC, MULU, MULS, NEG, NEGC, SUB, SUBC, SUBV	Add, Subtract, Multiply, Divide (initialize and step), Multiply-And-Accumulate, Negate, Extract	√	√	√	SH-1: 26/17 SH-2: 33/21 SH-3: 34/21
	MUL, DMULS, DMULU, DT	Double-Length Multiply, Decrement-and-Test		√	√	
Logic operations	AND, NOT, OR, TAS, TST, XOR	Bitwise logic operations, Memory Test-And-Set, Bit Test	√	√	√	14/6
Shift	ROTL, ROTR, ROTCL, ROTCR, SHAL, SHAR, SHLL, SHLLn, SHLR, SHLRn	Rotate, Shift-one-bit (arithmetic/logical), Shift-n-bits (logical, n=1,2,8, or 16)	√	√	√	SH-1: 14/10 SH-2: 14/10 SH-3: 16/12
	SHAD, SHLD	Shift-n-bits dynamic (arithmetic/logical, -325≤n≤31)			√	
Branch	BF, BT, BRA, BSR, JMP, JSR, RTS	Conditional branch; unconditional branch/call/jump/return with delayslot	√	√	√	SH-1: 7/7 SH-2: 11/9 SH-3: 11/9
	BRAF, BSRF, BF/S, BT/S	Far branch/call, conditional branch with delayslot		√	√	
System control	CLRT, CLRMAC, LDC, LDS, NOP, RTE, SETT, SLEEP, STC, STS, TRAPA	Clear T-bit	√	√	√	SH-1:31/11 SH-2: 31/11 SH-3: 74/14
	LDTLB, PREF, SETS, CLRS				√	
Total						SH-1: 133/56 SH-2: 142/62 SH-3: 189/66

The *data transfer* class include the load/store instructions for getting operands from memory and returning results to memory; these instructions can utilize any of the addressing modes of the SH. In addition, this class includes instructions for generating and storing addresses, saving the T-bit as a register-resident logical variable, and swapping/extracting fields of the 32-bit general-purpose registers.

The *arithmetic* class of instructions performs addition and subtraction for signed and unsigned operands of single and extended precision; sign- and zero-extension mechanisms convert operand types and sizes; a negation function manipulates the arithmetic sign. Single-length multiplication of signed and unsigned 16-bit operands (MULS .W and MULU .W) is also included, returning 32-bit results. Three instructions are provided as building blocks for division routines; two of these instructions initialize the signed-and unsigned-division sequences and are used with the repetitively-executed third instruction. These instructions produce their results in the SR's M-, Q- and T-bits; a user-written function accumulates the quotient, subtracts the divisor from the dividend on each pass and, optionally, calculates the final remainder. Comparison instructions, essentially subtractions which return only flags as results, are included in this class.

The *logic* class of instructions operates bitwise on 32-bit operands, performing the classical AND, OR, EXCLUSIVE-OR and INVERT operations. Various masked bit-tests are also performed on registers and memory, returning a flag-result without modifying the operands. An atomic test-and-set is also provided, which tests the value of the memory-resident byte operand, recording the result in the SR's T-bit; the bus is not released until the memory-resident byte's sign bit is set, as the instruction completes. This instruction is vital for fast synchronization semaphores in multi-tasking and multi-processing systems.

The *shift* class of instructions is composed of shift (logical and arithmetic) and rotate (through or past the T-bit) operations in either the left or right direction. The majority of these instructions shift or rotate by one bit position; however, the logical shifts can shift by 1, 2, 8 or 16 bits.

The *branch* class of instructions includes branches (conditional and unconditional), branches-to-subroutines (conditional and unconditional), jump, jump-to-and return-from-subroutine operations. Branches are PC-relative, while jumps jump through a pointer in a general-purpose register. Conditional operations depend upon the SR's T-bit; unconditional branches/jumps feature a delay slot--that is, the instruction immediately following them is executed before the branch, utilizing otherwise-idle time for the execution unit. The instructions which call or return from subroutines utilize the PR to store or retrieve the PC, respectively. Thus, the PR is effectively a fast, one-element-deep stack.

The *system control* class consists of instructions to load/modify/store the system registers and the control registers, to service exceptions and to manage transitions into the power-down modes. Two of these instructions are used to `connect' the PR with a memory-resident stack. Each subroutine call is preceded by a push (STS .L PR, with pre-decrement indirect register addressing) to the stack area; similarly, the return from a subroutine is followed by a pop (LDS .L PR, with post-increment indirect register addressing) from the stack area. This class also includes the trap instructions and the no-operation instruction.

2.2 SH-1's Implementation of the Architecture

The SH-1 family stresses high integration of memories and peripherals. It represents the simplest form of the SH series' architecture and implementation, while retaining high performance features. For example, all SH-1 implementations incorporate a 16-bit hardware multiplier, which produces a 32-bit result.

The SH-1 implements a *16bit x 16bit + 42bit → 42bit* multiply-and-accumulate (MAC .W) instruction; the accumulated results are stored in the concatenated MACL/MACH register pair (a combined register length of 64 bits) via sign extension. During execution of the SH-1's MAC .W, two signed 16-bit operands (from memory, each accessed via post-incremented indirect register addressing modes) are multiplied together, producing a signed 32-bit product. The current value in MACH/MACL, interpreted as a 42-bit two's-complement signed integer is added to the product; the 42-bit signed-integer sum is sign-extended to 64 bits and stored in MACH/MACL. Optionally, a *32-bit saturation* mechanism can be set (SR's S-bit = 1); it causes the accumulated result to be treated as a 32-bit signed integer, stored only in MACL. When it is enabled and a positive or negative 32-bit arithmetic overflow occurs during a MAC .W instruction, this mechanism sets the accumulated value in MACL to the maximum or minimum value for a 32-bit integer, respectively; in this mode, the MACH's least-significant bit is used as an overflow flag.

2.3 SH-2 Implementation -- Additions to the Architecture

The SH-2 family adds six instruction types, extending the SH-1 architecture while maintaining upward compatibility. SH-2 processors are intended for systems utilizing external buses; therefore, their designs stress flexibility and incorporate higher-performance computational elements while de-emphasizing on-chip peripherals. All SH-2 implementations incorporate a 32-bit hardware multiplier and a 64/32-bit divide unit.

Taking advantage of its additional hardware units, SH-2 offers extended multiply and divide operations. The SH-2's single-length multiply instruction, MUL, can accept longword operands (MUL .L), producing a 32-bit result. Further, the SH-2 includes double-length signed/unsigned multiply instructions (DMULS .L/DMULU.L), returning a 64-bit result. While there are no changes to the divide instructions, the divide unit can be accessed as a peripheral. The divide unit executes a complete division function in 39 clock periods, producing a 32-bit quotient and a 32-bit remainder. The divide unit utilizes 64-bit or 32-bit dividends; the divisor is always a 32-bit integer. Thus, the functions implemented in the divide unit can be described as *64b/32b→32bq...32br* and *32b/32b→32bq...32br.*

Several extensions have been implemented in the SH-2's multiply-and-accumulate instructions. For example, MAC .W has been extended from the SH-1's implementation; it now produces a true 64-bit accumulated result. Thus, all 64-bit-integer values of MACH/MACL are valid for the SH-2, removing the 42-bit-constraint ($16b\ x\ 16b + 64b \rightarrow 64b$). When the SH-2's saturation mechanism is enabled, the SH-2's MAC .W behaves identically to the SH-1's. Additionally, a longword multiply-and-accumulate (MAC .L) has been included; this instruction performs a longword multiply and accumulates a 64-bit sum ($32b\ x\ 32b + 64b \rightarrow 64b$). When saturation is enabled, the accumulated result of MAC.Lis constrained to the two's complement values representable in 48 bits (-2^{47} to 2^{47}-1); this result is sign-extended into the 64-bit MACH/MACL.

Control of program flow has been enhanced by additional instructions -- delayed-conditional-branch (BF/S and BT/S), Branch-Far (BF), Branch-to-SubRoutine-Far (BSRF) and Decrement-and-Test (DT).BF/S and BT/S provide additional time optimization for program loops. DT is a compact single-instruction subset of the more general two-instruction end-of-loop test; it provides a highly-effective optimization choice for compilers, usable in a large percentage of cases.

BF and BSRF provide the ability to branch or call through a register-resident offset, rather than through a displacement encoded within the instruction, extending their range. An addressing mode has been added in the SH-2 for use by these instructions. The new addressing mode is a variant on PC relative addressing; in the SH-2, a general-purpose register can also be added to the PC to form the target's memory address.

The SH-2 provides a unified instruction-and-data cache with a 4-way set-associative organization. The cache features a 16-byte line size, an LRU replacement algorithm, and a write-through policy. If desired, half of the cache can instead be used as a general-purpose SRAM; the remaining cache is then organized as 2-way set-associative. The cache can be dynamically bypassed during any given access by using an alternate address for the desired memory location; the read-modify-write cycle of the test-and-set (TAS .B) instruction always bypasses the cache. The cache can also be completely disabled. The cache's address (tag) and data arrays can be accessed directly; software can purge the cache explicitly, associatively or completely.

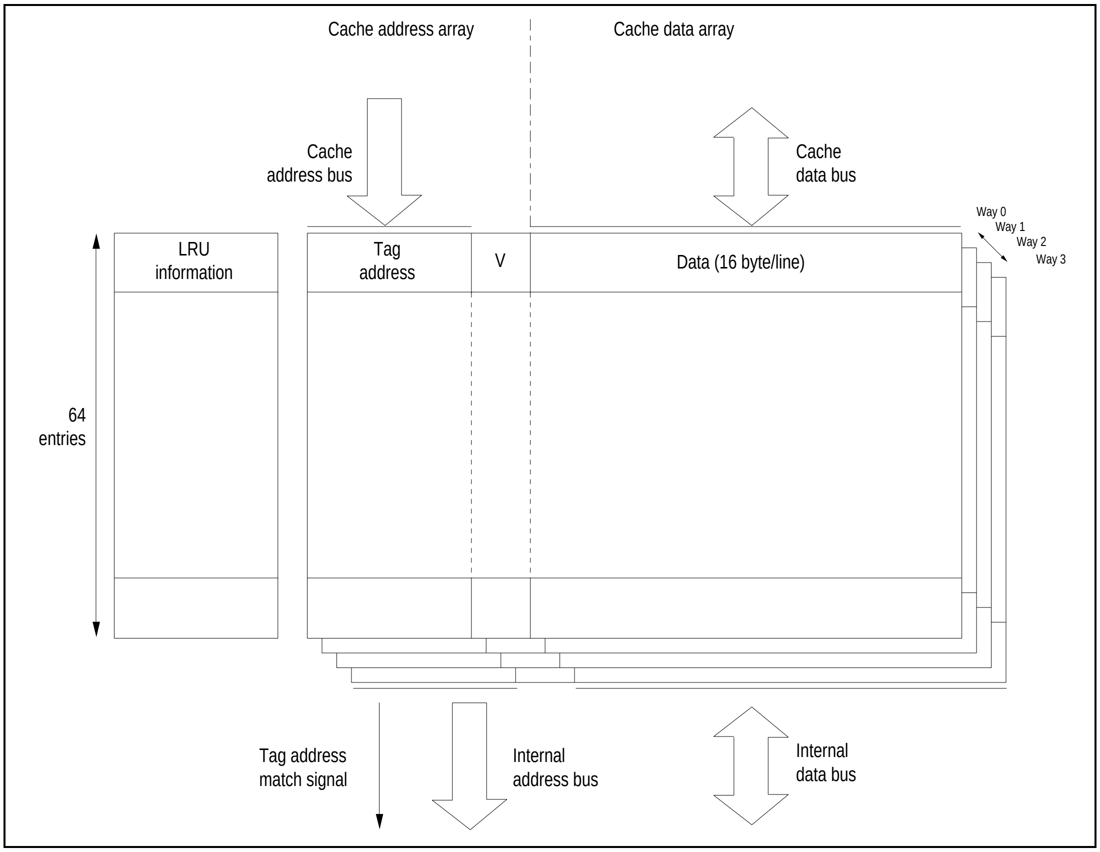


Figure 2.6 SH-2 Unified Cache

To simplify memory sharing in multiprocessor systems, the SH-2 supports both little- and big-endian memory- storage formats in one of its external memory areas; this area can be shared with another microprocessor using either `endian' convention. The user's programming of the Bus State Controller selects the storage format.

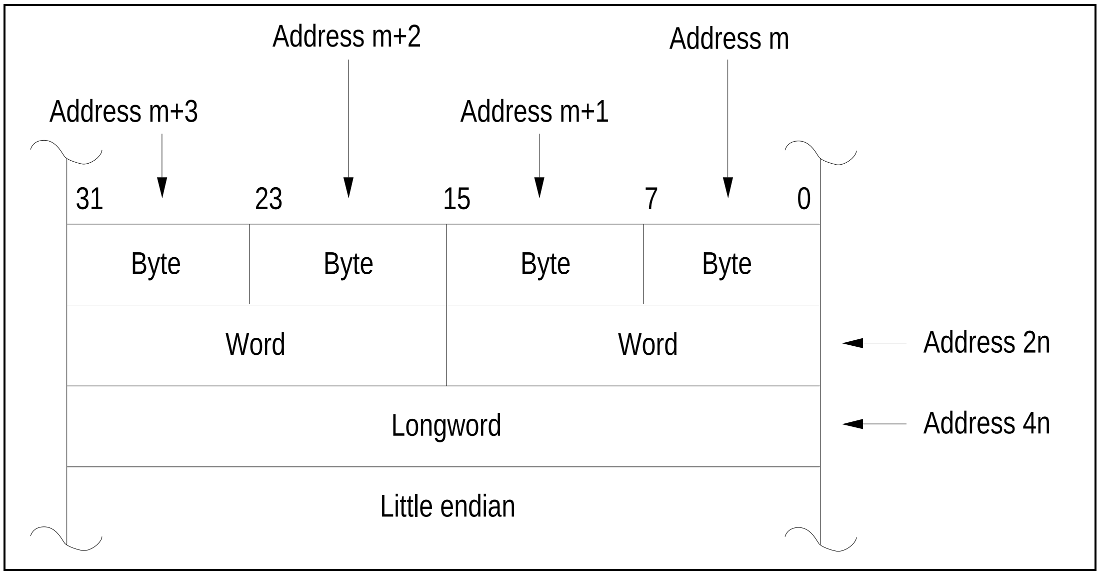


Figure 2.7 Little-Endian Storage of Operands in Memory

2.4 SH-3 Implementation--Additions to the Architecture

The SH-3 family extends the SH-2 architecture while maintaining upward compatibility. SH-3 processors serve bus-intensive systems with higher performance requirements and greater software complexity. Therefore, a major feature of the SH-3 architecture is the inclusion of a Memory Management Unit (MMU), which provides address translation and memory protection; as a result, the SH-3 is especially well-suited as a platform for operating systems, including those which support virtual memory.

The SH-3 provides a unified instruction-and-data cache with a 4-way set-associative organization. It features a 16-byte line size, an LRU replacement algorithm, a write-back (copy-back) policy and a write-allocate (fetch-on-write) mechanism. The cache is physically addressed, which prevents aliasing in virtual-memory systems. High performance is assured by accessing the cache and the MMU in parallel.

All SH-3 implementations incorporate a 32-bit hardware multiplier and a 32-bit barrel shifter. Big- or little-endian memory-storage convention can be chosen, via a jumper, at reset. The SH-3 retains all instructions and addressing modes from the SH-2; four additional instructions are provided to utilize the MMU, cache and barrel shifter.

2.4.1 Memory Management in the SH-3

The processors of the SH-3 family can operate in either of two modes -- user and privileged; privileged mode is normally reserved for the trusted procedures of the operating system, while application code runs in the user mode.

Memory management depends on the address translation mechanism, which is implemented in the MMU. When address translation is enabled, each access to memory is subject to translation; the process of translation, from the program's logical (also called virtual) address to the bus's physical address, is performed according to tables stored in memory. Simultaneously, the intended access is checked for memory-protection violations. If the access can be performed without any violations, the access is allowed to proceed; if not, the instruction is aborted and control is passed to the operating system.

Memory management is performed on a page basis; pages can be either 1 Kbyte or 4 Kbytes long. Separate mappings are maintained for the user and privileged modes; further, each process can have its own mapping; thus, each active process can access its own or shared code and data, but can be prohibited from accessing other processes' (or the operating system's) code and data. Of course, in this table-driven scheme, the mapping tables are a resource of the operating system, and are protected from all user-mode processes.

Since translation is performed on each memory access, high performance demands high-speed translation hardware; this hardware is implemented in the MMU's Translation Lookaside Buffer (TLB). The TLB is a specialized 4-way set-associative cache that contains 128 translations, called Page Table Entries (PTEs). Whenever a virtual address is presented for translation, the MMU examines the TLB; if present, the corresponding physical address is immediately available. If the required translation is not present, it must be acquired from the memory-resident tables and added to the TLB; if the necessary set of the TLB is full, this will require replacing a TLB entry. In order to maximize `hits' in the TLB, the MMU tries to maintain the most-needed translations in it. The MMU provides a mechanism for nominating a candidate in the set, and can support any software-driven PTE-replacement strategy.

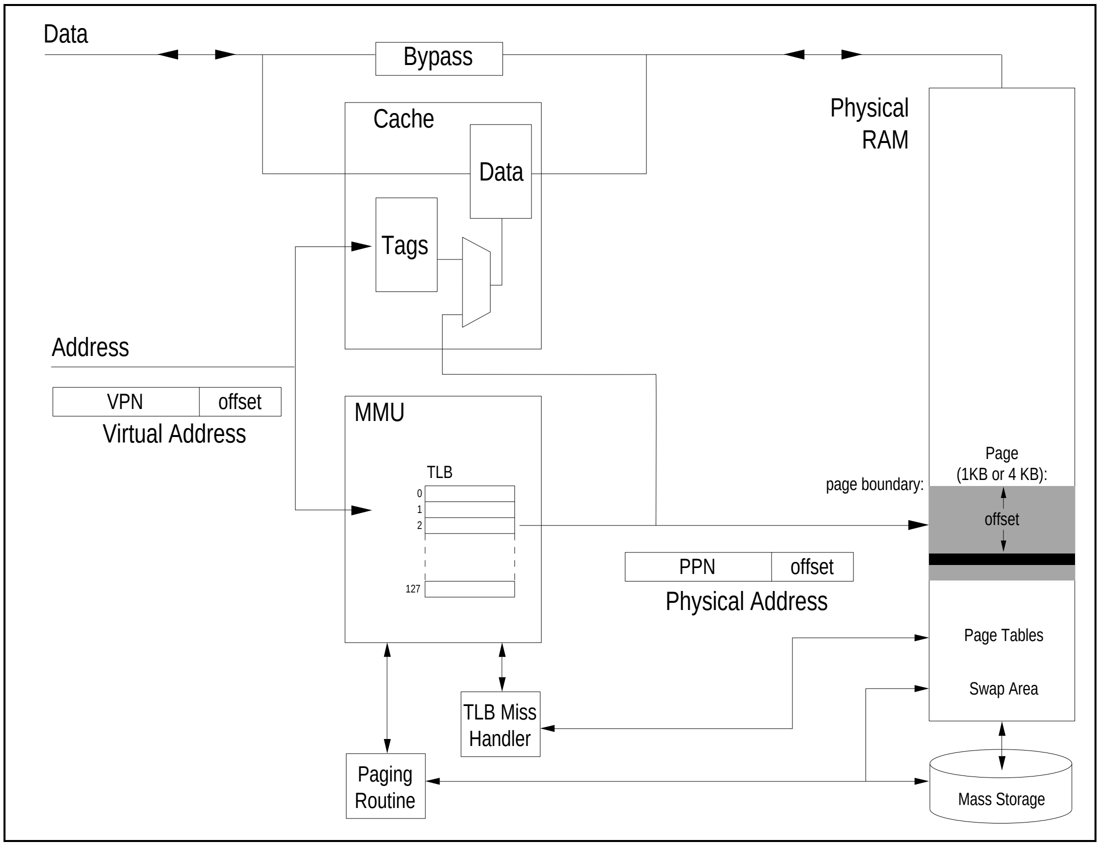


Figure 2.8 Address Translation with the MMU

Naturally, this mechanism must be protected from inadvertent disruption by an errant process; thus, the tables and registers of the MMU (and certain other resources) are not accessible by user-mode processes. As a result, the programming model of the SH-3 is mode-dependent, as shown; certain instructions cannot be executed while operating in the user mode. From the perspective of a user-mode process, the register model is the same as for any SH-series processor; however, the privileged mode has access to a number of additional registers. These additional registers accelerate transitions between operating modes, such as when entering or exiting an exception handler; such mechanisms minimize task-switch and interrupt latencies, reducing operating-system overhead.

In memory-managed systems, `hardware' is considered to be an operating-system resource. Thus, in the SH-3, caches and I/O are accessible only while operating in the privileged mode. Since hardware is often serviced by interrupts, the interrupt mechanism is privileged; thus, the VBR becomes a privileged-mode resource, inaccessible from the user mode. Essentially, the system environment is `hidden' from programs running in the user mode; application code sees a `pure' code-execution environment. The operating system performs all system-related tasks, at its discretion, on behalf of the applications.

The PTEs contain a *valid* bit, indicating that the required page is present in physical memory; this mechanism permits the implementation of virtual-memory (VM) systems. In VM systems, the memory hierarchy is extended by an additional level. A relatively small amount of physical memory (typically semiconductor RAM) is backed by a larger, lower-cost-per-bit mass-storage system (such as a magnetic disk); this two-level hierarchy serves instead of a large physical memory. Only the most-needed (frequently accessed) portions of applications' code and data are actually present in the physical memory; pages are brought into physical memory, by the operating system, as needed during the execution of the application code. If an intelligent *paging* algorithm is utilized, application-code performance does not degrade significantly compared to fully resident code in a large physical memory. Thus, high performance can be achieved in a cost-effective manner.

2.4.2 Additional Instructions in the SH-3

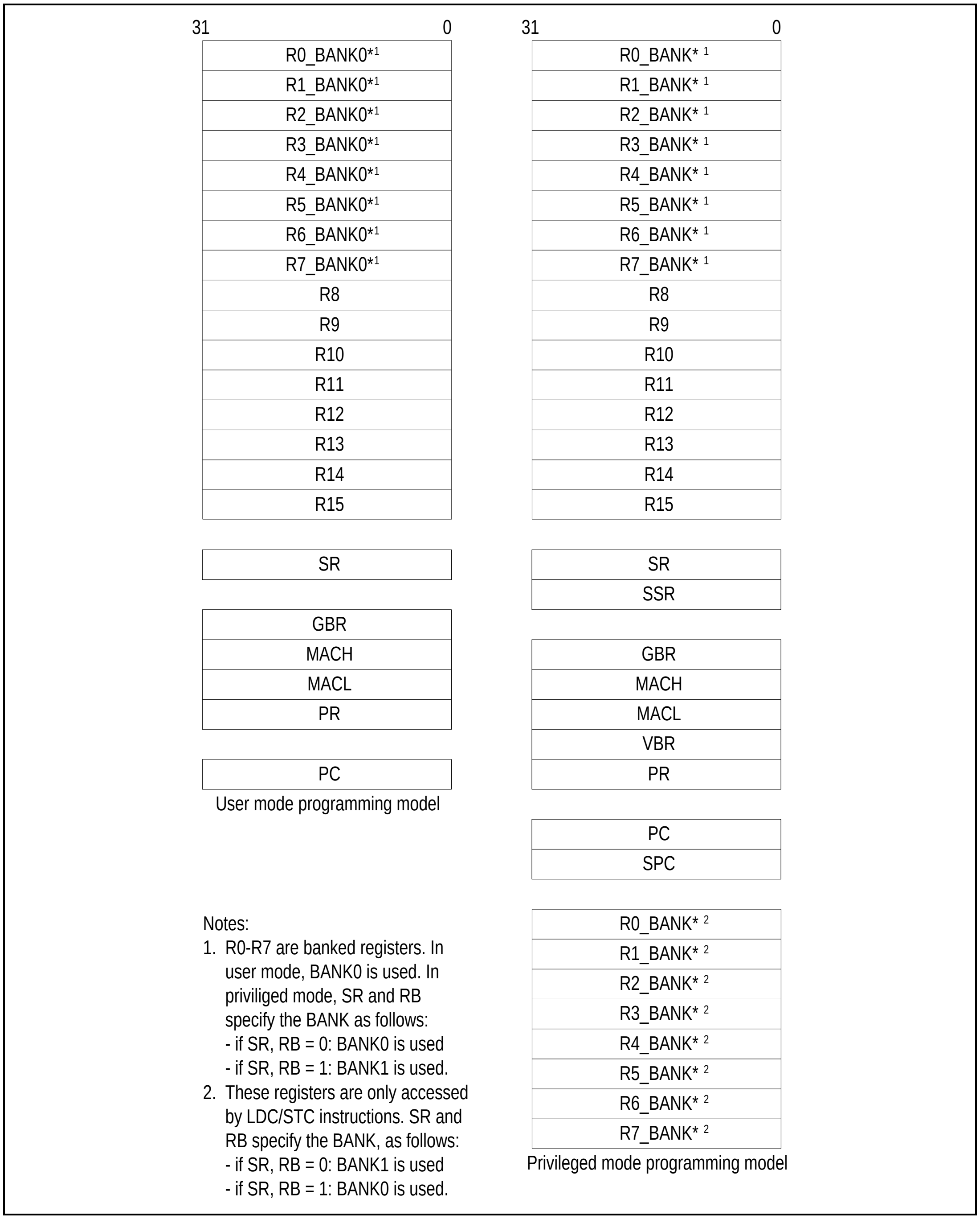
The SH-3 implements four additional instruction types, beyond the SH-2; they are related to the MMU, the cache and the barrel shifter.

The MMU is serviced by the load-page-table-entry-to-TLB (LDTLB) instruction; this instruction causes the TLB to be written. It is typically used by an O.S. routine to transfer a mapping from the memory-resident page tables to the TLB, possibly replacing a prior entry.

The unified cache is serviced by the cache-pre-fetch instruction (PREF); this causes a line of the cache to be filled from memory without otherwise using the pre-fetched data.

The barrel shifter permits the implementation of fast n-bit (dynamic) logical and arithmetic shift instructions (SHLD and SHAD). The shift count, a 6-bit signed integer, is taken from a general-purpose register; positive values indicate a left shift.

The system control instructions are extended to permit the SR's S-bit to set and cleared directly (SETS and CLRS); this streamlines control of the multiply-and-accumulate instructions' saturation mode.



Notes:

1. R0-R7 are banked registers. In user mode, BANK0 is used. In priviliged mode, SR and RB specify the BANK as follows:
   - if SR, RB = 0: BANK0 is used
   - if SR, RB = 1: BANK1 is used.
2. These registers are only accessed by LDC/STC instructions. SR and RB specify the BANK, as follows:
   - if SR, RB = 0: BANK1 is used
   - if SR, RB = 1: BANK0 is used.

Figure 2.9 User/Privileged Programming Models of the SH-3

Section 3 Handling Exceptions Vector -- Table Organization

A complex software system requires an underlying system for handling exceptional conditions; efficient service of hardware demands a highly evolved mechanism for servicing interrupts. Secure operating systems require a bulletproof system for handling faults caused by hardware and software events. In the SH, all of these needs are met by a comprehensive system of exceptions, interrupts and traps.

The families of the SH Series provide a single, uniform mechanism for handling all exceptions, whether caused by hardware or software conditions. Reset (power-on and manual), address errors (by the CPU or DMAC), interrupts (external or internal) and instructions (traps, undefined instructions and illegal slot instruction) generate exceptions.

Exception handling in the SH series revolves around a single table, the Exception Vector Table, which is utilized for dispatching all exception handlers; the Vector Base Register, VBR, points to the beginning of this table. Exception conditions, several of which may occur simultaneously, are prioritized by dedicated hardware; the resulting unique vector number is used as an index into the table to select the appropriate handler's entry point, a longword containing the address of the first byte of code in the handler. The vector table organization used in the SH-1 is shown below.

In the case of reset, both power-on and manual, the vectors are of a different form and are two longwords in length. The first longword contains the initial value for the Program Counter (the first logical instruction); the second longword contains the initial value for the Stack Pointer (the base of the stack).

Exception handlers can be thought of as hardware-dispatched subroutine calls; like a subroutine call, an exception causes the return address (the current PC) to be saved. However, exceptions, unlike subroutine calls, save the PC directly on the stack; in addition, the return status (the current SR) is saved on the stack as well. When the exception handler has completed its function, a return-from-exception instruction restores both registers and thereby causes control to be returned to the application code which was being executed prior to the exception. If the exception handler determines that the application code has caused or encountered a fatal error, the operating system can easily terminate the failed procedure and continue other system operations.

The Exception Vector Tables for the SH-1 and SH-2 are nearly identical, with some differences in the interrupt section, due to differences in their interrupt controllers.

For the SH-3, the Exception Vector Table is modified to include vectors for handling address-translation issues, such as TLB misses and memory-protection violations. The number of discrete exception vectors is minimized; the specific event causing an exception is determined by software. The handler examines a 12-bit exception code, unique for each event, which appears in either the Exception Event Register (EXPEVT) or the Interrupt Event Register (INTEVT); if the code indicates that a trap has occurred, the specific trap can be identified by examining a 4-bit field in the Trap Register (TRA).

Table 3.1 Exception Vector Table (SH-1)

Exception Source		Vector Number	Vector Table Address Offset
Power-on reset	PC	0	H'00000000 - H'00000003
	SP	1	H'00000004 - H'00000007
Manual reset	PC	2	H'00000008 - H'0000000B
	SP	3	H'0000000C - H'0000000F
General illegal instruction		4	H'00000010 - H'00000013
(Reserved for system use)		5	H'00000014 - H'00000017
Illegal slot instruction		6	H'00000018 - H'0000001B
(Reserved for system use)		7	H'0000001C - H'0000001F
		8	H'00000020 - H'00000023
CPU address error		9	H'00000024 - H'00000027
DMA address error		10	H'00000028 - H'0000002B
Interrupts	NMI	11	H'0000002C - H'0000002F
	User break	12	H'00000030 - H'00000033
(Reserved for system use)		13-31	H'00000034 - H'00000037 to
			H'0000007C - H'0000007F
Trap instruction (user vectors)		32-63	H'00000080 - H'00000083 to
			H'000000FC - H'000000FF
Interrupts	IRQO	64	H'00000100 - H'00000103
	IRQ1	65	H'00000104 - H'00000107
	IRQ2	66	H'00000108 - H'0000010B
	IRQ3	67	H'0000010C - H'0000010F
	IRQ4	68	H'00000110 - H'00000113
	IRQ5	69	H'00000114 - H '00000117
	IRQ6	70	H'00000118 - H'0000011B
	IRQ7	71	H'0000011 C - H'0000011F
	On-chip modules	72-255	H'00000120 - H'00000123 to
			H'000003FC - H'000003FF

The SH-3 also provides mechanisms to expedite exception handling; they minimize the time spent in transitioning between operating modes. For example, the user-mode PC and SR are automatically captured in the SPC and SSR when an exception occurs, and are restored when a return-from-exception (RTE) instruction is executed. Similarly, half of the general-purpose registers are banked; the user mode can access only one bank, but the privileged mode has access to both. Bank switching occurs automatically during mode transitions, providing a fast context switch.

Naturally, all resources of the exception mechanism -- registers, tables, and handlers--are privileged in the SH-3 and, therefore, are inaccessible by user-mode processes; this permits the implementation of secure, robust systems.

Section 4 Power-Down Modes -- CPU Operating States

Conventionally, high computational performance comes at the price of high power dissipation in the CPU. In the SH Series, great care has been taken in the fabrication process and in the circuit design to ensure that power dissipation is kept to a minimum; the SH-3's cache design, previously discussed, typifies this approach. However, the statistics of system usage provide yet another opportunity for power savings through use of different operating modes, each with a different power consumption.

Many real-world systems do not need to operate at their maximum computational performance at all times; instead, they may be able to operate at reduced capacity for a large percentage of the time, providing their peak performance for only short periods of time. This is typical, for example, of personal digital assistants (PDAs) -- peak performance may be needed when performing handwriting recognition, but relatively low performance is adequate for outputting the results to the display (even less power is needed once the display has been updated and is being presented statically for the user to read). This high ratio of peak-to-average performance permits battery power to be saved by permitting a weighted average, over time, of different power-dissipation levels in different operating modes.

The SH Series incorporates a wealth of mechanisms for power savings of this type, providing great flexibility in the design of efficient systems. Effectively, the CPU and its peripherals can be `reined in' to lower performance (and lower power-dissipation) levels, while retaining the ability to be quickly `turned loose' when performance demands become greater. It is practical, then, to design systems which tailor their performance, second by second, to the user's demands, thereby maximizing battery life.

The various mechanisms include sleep mode, standby mode, module standby mode, clock-speed reduction and bus-speed reduction. All members of the SH series support sleep mode and standby mode; these are considered `power down' modes, because CPU processing halts. The additional mechanisms are implemented in various families.

When no power-saving mechanisms are engaged, the processor is said to be running in the *normal mode;* maximum performance is available.

In *sleep mode*, the CPU is halted, but the clock generator and all on-chip peripherals continue operation; on- chip memories and registers are held while external memories are refreshed. This mode is entered under software control by executing the SLEEP instruction with the SBYCR's (STBCR's in the SH-3) SBY (STBY in the SH-3) bit cleared. It is exited by either an interrupt or a reset; recovery is rapid, since the clock is stable and states have been maintained.

In *standby mode*, the CPU, the clock generator and the on-chip peripherals are all halted; on-chip memories and registers are held and, where supported, external memories can be self-refreshed. This mode is entered under software control by executing the SLEEP instruction with the SBYCR's (STBCR's in the SH-3) SBY (STBY in the SH-3) bit set. It is exited by either a non-maskable interrupt or a reset; sufficient time must be allocated, during the exit, to permit the clock generator to stabilize.

Table 4.1 Typical Current Consumption

		Mode			
Family	**Device**	**Normal**	**Sleep**	**Standby**	**Condition**
SH-1	SH7034	100 mA	60 mA	0.010 µA	Vcc=5.OV, f=20 MHz
		60 mA	40 mA	0.010 µA	Vcc=3.3V, f=12.5 MHz
SH-2	SH7604	110 mA	80 mA	1.0 µA	Vcc=5.OV, f=28.7 MHz
		60 ma	40 mA	1.0 µA	Vcc=3.3V, f=12.5 MHz
SH-3	SH7708	180 mA	20 mA	1.0 µA	Vcc=3.3V, f=60 MHz

The SH-2 and SH-3 support other clock-related mechanisms for control of system performance and power dissipation. For example, both families support the *module standby mode*. In this mode, certain peripheral modules can be halted, on a module-by-module basis, while CPU processing continues. The clock generator continues to run, but the clock is not passed to those peripheral modules which are `standing by'; those modules' operating states are held or initialized. Module standby mode can be combined with the CPIJ's other operating modes, such as normal and sleep.

The SH-2's clock generator incorporates a programmable phase-locked-loop (PLL) to generate the system clocks; the PLL can be used to optionally double or quadruple the frequency of a reference clock, which may be supplied from an external source or by the on-chip crystal oscillator. The clock-multiplication ratio (1, 2 or 4) can be changed at any time under software control, allowing the CPU to `shift gears' as needed by the user. The external bus has a basic cycle time of two processor clocks at all times.

The SH-3's clock generator uses a mechanism similar to the SH-2's but with even greater flexibility; three clocks are separately generated -- one each for the CPU, the on-chip peripherals and the external bus. In general, this means that the internal CPU clock frequency can be 1, 2 or 4 times the external bus's frequency; also, typically, the clock for the on-chip peripherals is one-half or one-quarter of the internal CPU clock's frequency.

Section 5 External Memory Control -- Buses

All devices in the SH series can utilize memory and peripheral devices on an external bus; this bus is controlled by an on-chip Bus State Controller (BSC). The BSC is responsible for dividing the physical memory space into separate areas and determining the bus characteristics of each of those areas; for example, the BSC defines the data-bus width for each area and defines the type of external memories (e.g., SRAM, DRAM, SDRAM) supported in each area. During each access, the BSC decodes the physical addresses into external area-select signals, produces strobes, responds to bus-status signals and inserts wait states as needed; it produces the type-specific signals to read, write and refresh (if necessary) the memory.

Architecturally, the SH series' physical address space is 4 gigabytes, since it is addressed via 32 bits. However, each family implements a subset of this space, consistent with its target applications. Thus, the SH-1, -2 and -3 families feature 32M, 128M and 512M bytes of usable physical address space, respectively.

In the SH-1, the external bus has a maximum width of 16 bits. The physical address space is decoded into eight areas of 4 megabytes each; address bits A28 through A31 are not decoded. Address bit 27 determines the width of the data bus (8 or 16 bits) for each area. Static memory and DRAM are supported variously in the external areas.

In the SH-2, the external bus has a maximum width of 32 bits. The physical address space is decoded into four areas of 32 megabytes each; higher-order address bits are decoded to utilize the on-chip cache in different modes (e.g., bypass, purge) or for other purposes (e.g., selecting the on-chip peripherals). Within each area, the width of the data-bus (8, 16 or 32 bits) is determined by a register field. Static memory, burst ROM, DRAM, SDRAM and PSRAM are supported variously in the external areas.

The SH-3's external bus has a maximum width of 32 bits. The physical address space is decoded into eight areas (of these, seven are external, the eighth being the on-chip peripherals) of 64 megabytes each; address bits 29 through 31 are not decoded. Within each area, the width of the data-bus (8, 16 or 32 bits) is determined by either a register field or external pins. Static memory, burst ROM, DRAM, SDRAM, PSRAM and PCMCIA are supported variously in the external areas.

Both the SH-2 and the SH-3 feature on-chip clock generators which incorporate phase-locked loops (PLLs). The PLLs minimize the difference (skew) between internal and external clocks, thus simplifying the design of high-performance systems. The PLLs can also optionally multiply the external (reference) clock to generate the internal CPU clock. The SH-3's clock generator offers two additional features. First, the SH-3 CPU can operate its internal circuitry at a multiple (1, 2 or 4) of its external-bus speed; second, its internal peripheral clock can be a subharmonic of the CPU frequency.

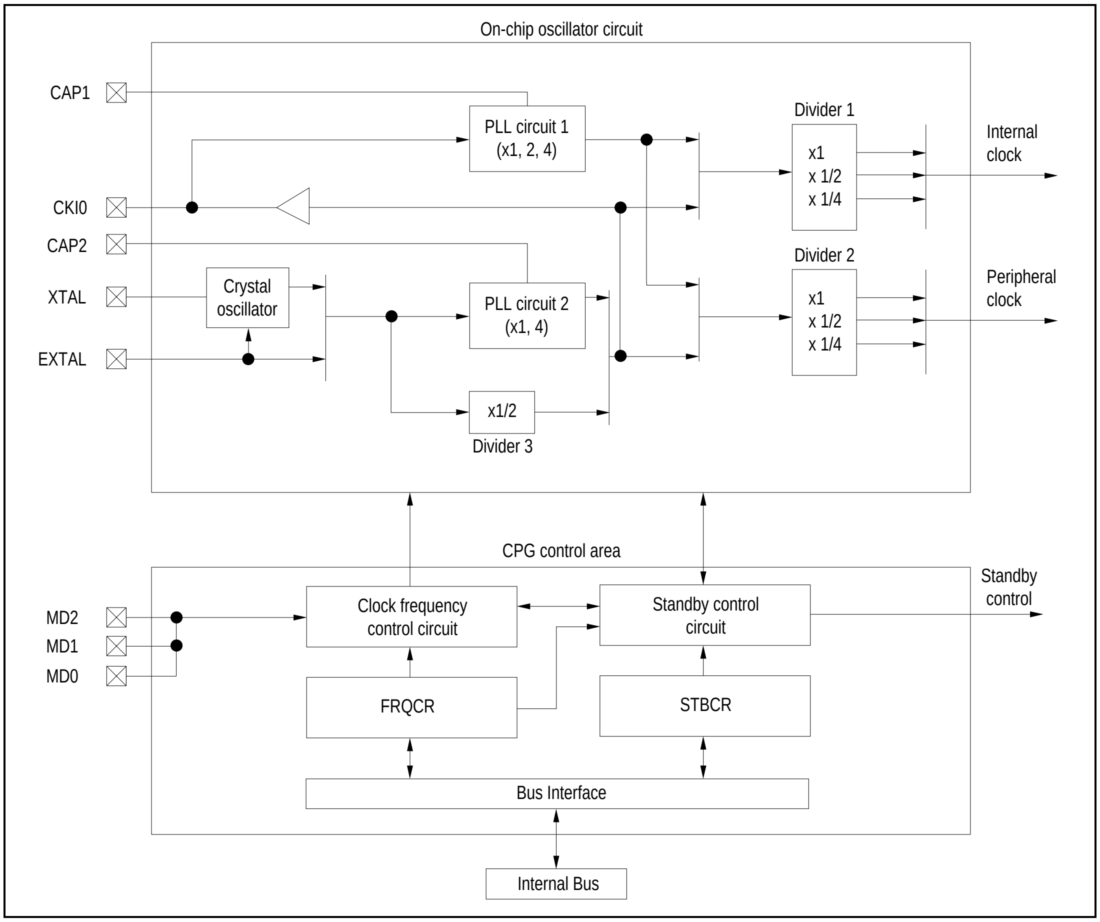


Figure 5.1 Clock Pulse Generator of the SH-3

Section 6 Peripherals

6.1 Common Features

Using Hitachi's high-density sub-micron CMOS processes, all processors in the SH series offer on-chip peripheral functions, simplifying system design. The following peripherals are included in all SH series micros.

A *serial communications interface* (SCI), with integral bit-rate generator (BRG), provides full-duplex synchronous or asynchronous serial communications. The SCI performs double-buffered serial/parallel conversion, insertion/stripping of start and stop bits, parity generation/checking and detection of framing and overrun errors. In asynchronous mode, data length, number of stop bits and parity type are programmable. Some SCIs also include a smart-card interface.

The *watchdog timer* (WDT) is an eight-bit up-counter driven by one of eight input clocks, which are derived from the peripheral clock. Its primary purpose it to reset the processor in the event of a stalled system; under normal operation, the system software should periodically clear it, to prevent overflow. If the timer overflows (indicating that the system has stalled), it generates a reset (watchdog-timer mode) or an interrupt (interval-timer mode). It can also be used, when exiting the standby mode via the non-maskable interrupt (NMI), to provide the necessary oscillator-settling time.

A *user break controller* (UBC) is provided in all SH series micros. The UBC is a debugging device that can be utilized at run time by system code; it is used to generate breakpoints on memory accesses or on code execution, including ROM-resident code. Implementation of the UBC varies across the families of the SH series; a description of a conventional channel follows. A single channel of the UBC examines, during each bus cycle of the CPU and/or DMAC, the bus cycle's address, type (instruction-fetch or data-access), direction (read and/or write) and size (byte, word or longword), comparing them against stored parameters (i.e., against the breakpoint's definition); address comparisons are masked on a bitwise basis. When a match is encountered, a user-break exception is generated.

Each member of the SH series includes an on-chip *interrupt controller* (INTC). The INTC is responsible for receiving all interrupt requests from on-chip and external sources, prioritizing them, comparing the highest requested priority against the currently-active interrupt level and asserting a single, appropriate request to the CPU. The INTC also assigns, according to user programming, the interrupt priority to be associated with each peripheral module.

In addition, various devices in the series provide parallel I/O, several types of timers, direct-memory-access controllers (DMACs), real-time clocks (RTCs) and analog-to-digital converters (A/Ds).

Table 6.1 SH Series Memory and 1/O

	SH-1				SH-2	SH-3	
Feature	**SH7020**	**SH7021**	**SH7032**	**SH7034**	**SH7604**	**SH7702**	**SH7708**
ROM on-chip	16kB	32kB	--	64kB	--	--	
RAM on-chip,	1kB	1kB	8kB	4kB	(notes 1, 2)	(note 2)	
Cache	--	--	--	--	4k	2k	8k
MMU w/TLB	--	--	--		--	128 entries	
I/O Lines	32	32	32	32	--	8	8
Input Lines	0	0	8	8	--	--	
DMA	4 channels	4 channels	4 channels	4 channels	2 channels	--	
Timers	5 channels x 16 bits	5 channels x 16 bits	5 channels x 16 bits	5 channels x 16 bits	1 ch x 16 bits (free-running)	3 channels x 32 bits	
Timing Pattern Controller	4 channels x 4 bits	4 channels x 4 bits	4 channels x 4 bits	4 channels x 4 bits	--	--	
Watch-Dog Timer	1	1	1	1	1	--	
Serial Comm. Interface	2 channels	2 channels	2 channels	2 channels	1 channel	1 channel	
User Break Controller	1 channel	1 channel	1 channel	1 channel	2 channels	2 channels	
R/T Clock	--	--	--	--	--	1	1
A/D	--	--	8 channels x 10 bits	8 channels x 10 bits	--	--	
External Interrupts	9	9	9	9	5	(note 3)	(note 3)
Internal nterrupts	30	30	31	31	11	5	5
External Data Path Width	16-,8-bit	16-,8-bit	16-,8-bit	16-,8-bit	32-,16-,8-bit.	16-,8-bit.	32-,16-,8-bit.
Bus Controller (Memory Interfaces)	SRAM, DRAM	SRAM, DRAM	SRAM, DRAM	SRAM, DRAM	SRAM, DRAM, SDRAM, Burst ROM, PSRAM,	SRAM, DRAM, Burst ROM, PSRAM, 1 x PCMCIA	SRAM, DRAM, SDRAM, Burst ROM, PSRAM, 2 x PCMCIA

Notes: 1. In two-way cache mode, 2KB of cache can be defined as general-purpose RAM.

2. Disabled cache can be directly addressed; thus, can serve as general-purpose RAM.

3. NMI plus IRL0-IRL3 encoded.

6.2 Peripherals and Memory on the SH-1 Devices

The SH7032 and SH7034 include 40 lines of parallel I/O, partitioned into 3 ports. Ports A and B are each 16 bits wide; their pins are individually programmable as inputs or outputs. Port C is an 8-bit-wide input port. The SH7020 and SH7021 feature 32 lines, implemented in Ports A and B.

The SH-1 micros each incorporate two SCIs, each with an integral bit-rate generator (BRG); the SCIs are functionally identical.

A four-channel DMAC included on the SH-1s transfers, in byte or word units, up to 65536 units between selected source/destination pairs. Two of the channels support single-address transfers; this mode transfers (in a single bus cycle) data between an external DREQ/DACK-accessible peripheral device and an external-bus-resident memory/device. All channels support dual-address transfers; this mode transfers data, in two bus cycles, between two bus-resident memories/devices through a temporary location--these transfers can utilize any combination of internal or external memory or peripheral as the source and destination, except the DMAC itself.

The SH-1s include a 16-bit *integrated timer-pulse unit* (ITU), composed of 5 channels. Channels O and 1 are up-counters. Up/down-counting is performed by channel 2 in the phase-counting mode and by channels 3 and 4 in the complementary pulse-width-modulator (PWM) mode. Phase-counting mode utilizes a pair of input signals with a guadrature (90°) phase relationship, as supplied by a bi-directional shaft encoder or rotary `switch'; PWM mode permits the generation of variable-frequency and variable-duty-cycle output waveforms.

A programmable *timing pattern controller* (TPC) is connected to the SH-1's ITU; it provides the ability to generate arbitrary digital output signals whose edges coincide with selected compare-match signals from the ITU's timers. Sixteen latched output bits are provided, partitioned into four groups of four; each group-of-four can be updated synchronously with a group-selected compare-match signal from the ITU. The output latches are updated from a 16-bit next-data register; the updating of each output bit can be enabled by a mask bit. In this mode, the period between successive compare-matches is the minimum separation between successive edges of an output waveform; this mode can be used, for example, to generate a phased set of waveforms. A non-overlapped mode is also available, for generating waveforms with no overlap; in this mode, both compare-match signals (from a single channel of the ITU) are used to time the updating of a group-of-four. One compare-match determines the minimum separation between successive falling (for example) edges of the output, while the other compare-match determines the minimum separation between the falling (for example) edge of one waveform and the rising (for example) edge of any other waveform. The TPC can also be used to initiate DMA transfers.

The SH-1's User Break Controller (UBC) allows a single breakpoint definition; it is conventional in structure and operation.

Interfacing with the analog world is simplified by the inclusion, on the SH7032 and SH7034, of a 10-bit *analog-to-digital converter* (A/D) preceded by an eight-channel analog multiplexer and a sample-and-hold circuit. The analog conversion range can be set anywhere within the CPU's operating voltage range via an analog reference input; a single channel is converted in 6.7 µs at 20 MHz. Conversion can be constrained to a single channel; alternatively, a subset of the channels can be successively sampled and converted. Conversion can take place continuously or on a one-shot basis; it can be gated by software or initiated in response to an external event.

The wealth of on-chip peripherals in the SH-1 family's devices requires each of the I/O-port pins to have two or more possible functions. Therefore, a *pin function controller* (PFC) peripheral is provided to define the actual function of each pin.

The SH7032 includes 8 Kbytes of RAM; the SH7034 offers 64 Kbytes of ROM (or EPROM) and 4 Kbytes of RAM. The SH7020 features 16 Kbytes of ROM and 1 Kbyte of SRAM; the SH7021 includes 32 Kbytes of ROM and 1 Kbyte of SRAM.

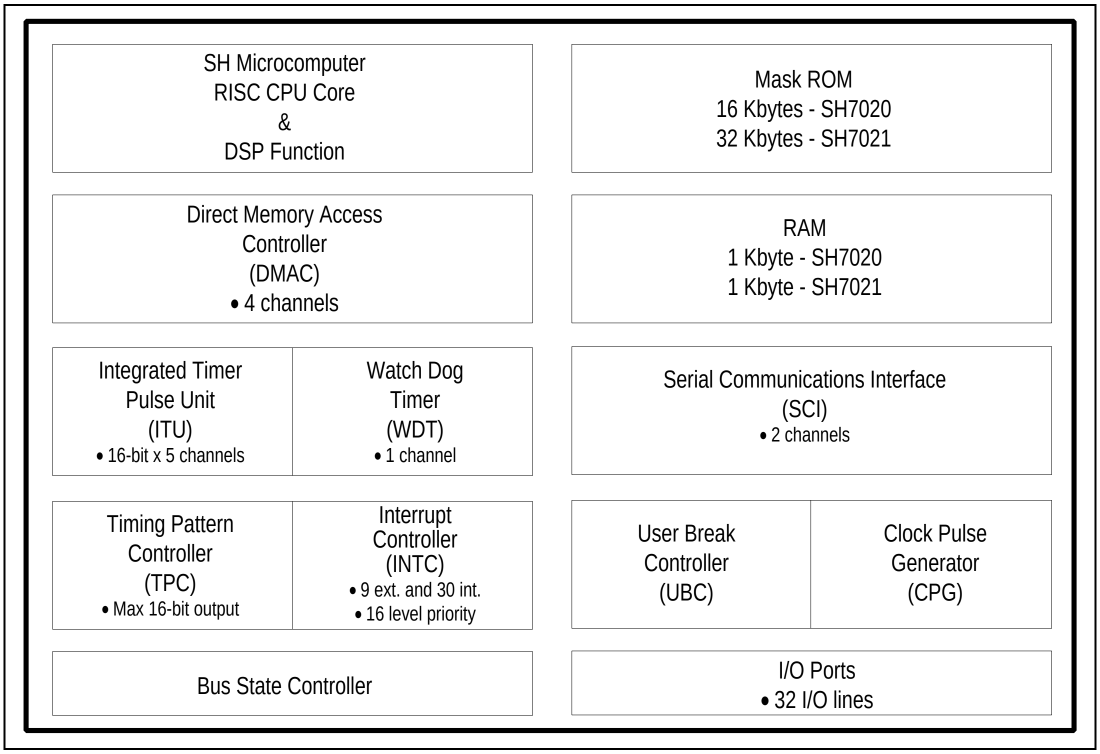


Figure 6.1 SH-1 Device Block Diagrams--SH7020/SH7021

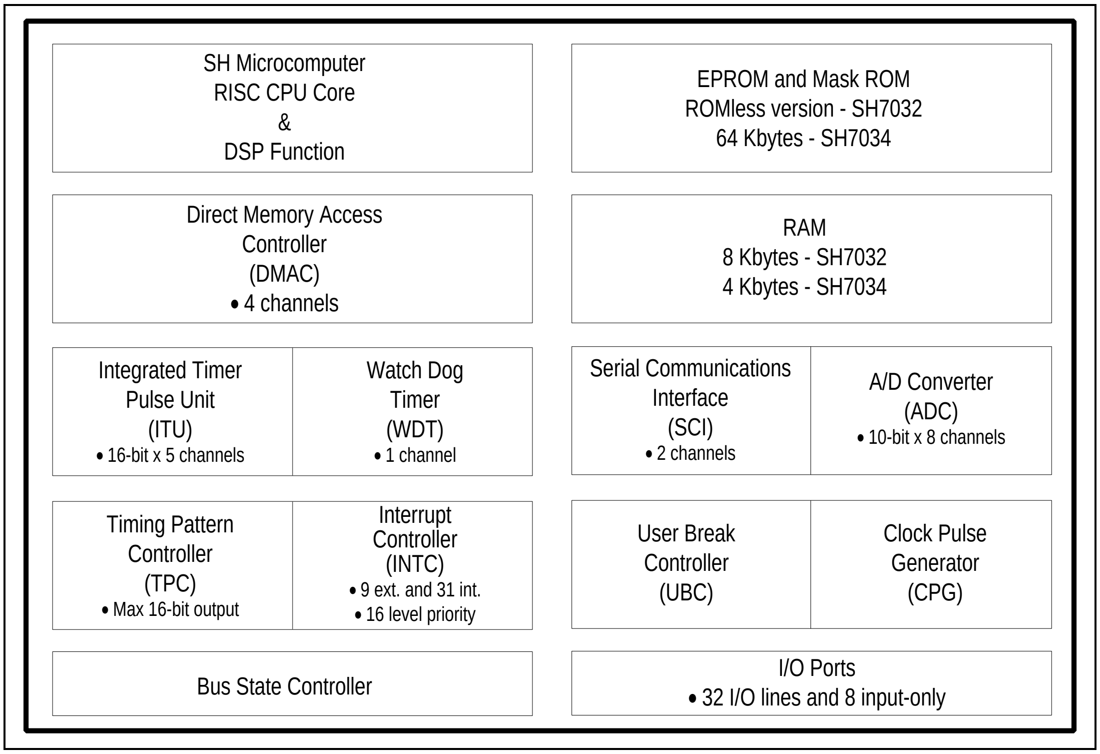


Figure 6.2 SH-1 Device Block Diagrams--SH7032/SH7034

6.3 Peripherals on the SH-2 family

The SH7604 features a single SCI with integral bit-rate generator.

A two-channel DMAC included on the SH7604 transfers, in byte, word, longword or 16-byte units, up to 16,777,216 units between selected source/destination pairs. All channels support both single-address and dual-address transfers. The single-address mode transfers (in a single bus cycle) data between an external DREQ/DACK-accessible peripheral device and an external-bus-resident memory/device. Dual-address mode transfers data, in two bus cycles, between two bus-resident memories/devices through a temporary location--these transfers can utilize any choice of external memory, internal or external memory-mapped peripheral as the source and destination, except the DMAC, BSC and UBC.

Finally, the SH76(14 includes a 16-bit *free-running timer* (FRT) with compare-match and input-capture features. The FRT utilizes either an external clock or a clock scaled from the CPU clock. The compare-match feature allows values to be preset into a pair of match registers; when the FRT's count matches one of these registers, an edge can occur in the corresponding output waveform--two such waveforms can be generated independently. The input-capture feature causes the FRT's current count to be copied into a latch register in response to an external event (a rising or falling edge); this provides a mechanism for measuring external pulse widths or time intervals.

The SH-2's UBC is a superset of the SH-1's; it can be operated in a fully-SH-1-compatible mode. The UBC offers two break channels with independent or sequential operation (first A, then B); channel A is conventional. The channel B breakpoint definition also includes the data bus; the data-bus comparison can be masked on a bitwise basis.

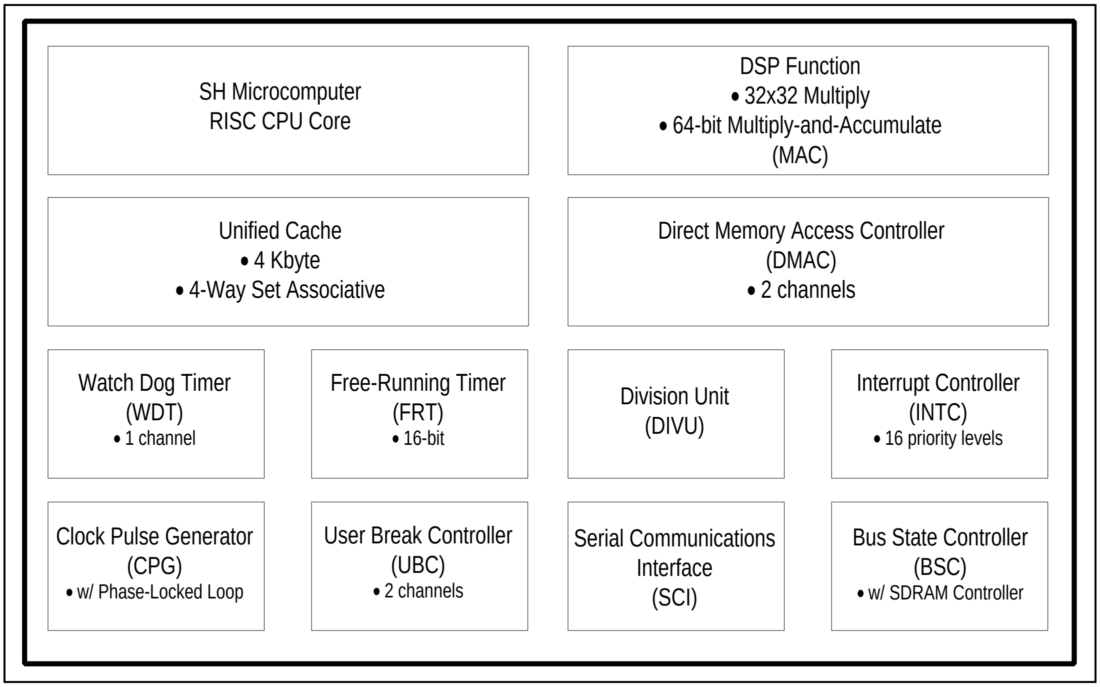


Figure 6.3 SH-2 Device Block Diagram--SH7604

6.4 Peripherals on the SH-3 family

The SH7702 and SH7708 feature an 8-bit parallel I/O port; each bit can be separately defined as an input or output.

A single SCI with integral BRG is provided; it includes a smart-card interface.

A *real-time clock* (RTC), with a dedicated on-chip 32.768-KHz oscillator, keeps track of the calendar time in binary-coded-decimal second/minute/hour/day/date/month/year format. Resolution of one second is maintained, with automatic correction in leap years. An alarm interrupt is included; the alarm comparator compares the seconds through month counters against a corresponding set of user-programmable registers. Interrupts can also be generated at selected intervals between 1/256th to 2 seconds.

The SH-3s incorporate a three-channel *timer unit* (TMU); each channel consists of a 32-bit down-counter and various registers. At underflow, each channel is re-loaded from its corresponding constant register; an interrupt can be generated when each counter underflows. Each of the three channels is driven by a separate selection of the peripheral clock or various subharmonics (1/4th, 1/16th, 1/64th or 1/256th). One channel provides an input-capture function.

The SH-3's UBC provides two breakpoint definitions; channel A is conventional. Channel B also includes the data bus in the definition; the data-bus comparison is bitwise-masked. The two channels can independently active or can be sequentially active (first A, then B).

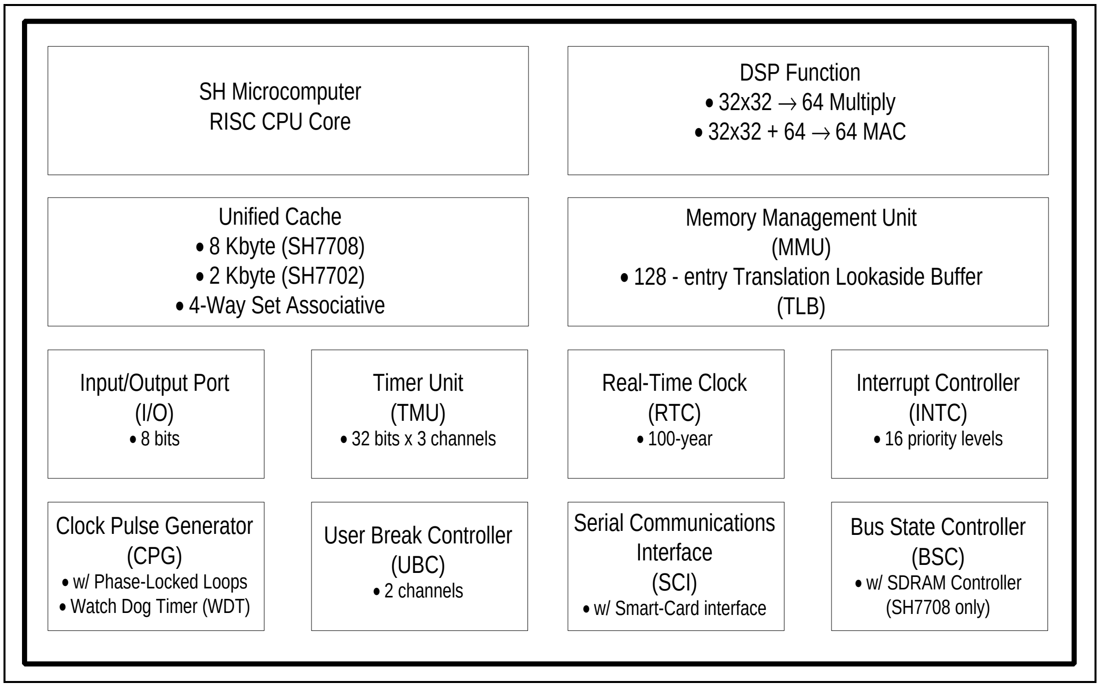


Figure 6.4 SH-3 Device Block Diagrams -- SH7702/SH7708

Section 7 Device Summary and Packaging

Devices of the SH series are packaged in plastic Quad Flat Packs (QFPs) or Thin Quad Flat Packs (TQFPs). Hitachi currently offers the SH devices summarized in the table. Since the SH series is constantly being expanded, please contact your local Hitachi representative for current information.

Table 7.1 SH Device Summary

				Part Number			Package	
Family	Device	VCC (VDC)	Freq (MHz)	OTP/ZTAT (HD647....)	ROM (HD643....)	ROMLESS (HD641....)	QPF-	TQFP-
SH-1	SH7020	5.0	20	7021X20[1]	7020X20			100
		3.3	12.5		7020VX12			100
	SH7021	5.0	20	7021VX20[1]	7021X20			100
		3.3	12.5		7021VX12			100
	SH7032	5.0	20			7032F20	112	
			16.6			7032X16		120
		3.3	12.5			7032VF12	112	
	SH7034	5.0	20	7034F20	7034SF20		112	
		3.3	12.5	7034VF12	7034SVF12		112	
SH-2	SH7604	5.0	28.7			7604SF28	144	
		3.3	20			7604SVF20	144	
SH-3	SH7702	3.3	45			7702X45		120
	SH7708	3.3	60			7708F60	144	
			100			7708F100[2]	144	

Notes: 1. Under development for 2Q96.

2. Under development for 4Q96.

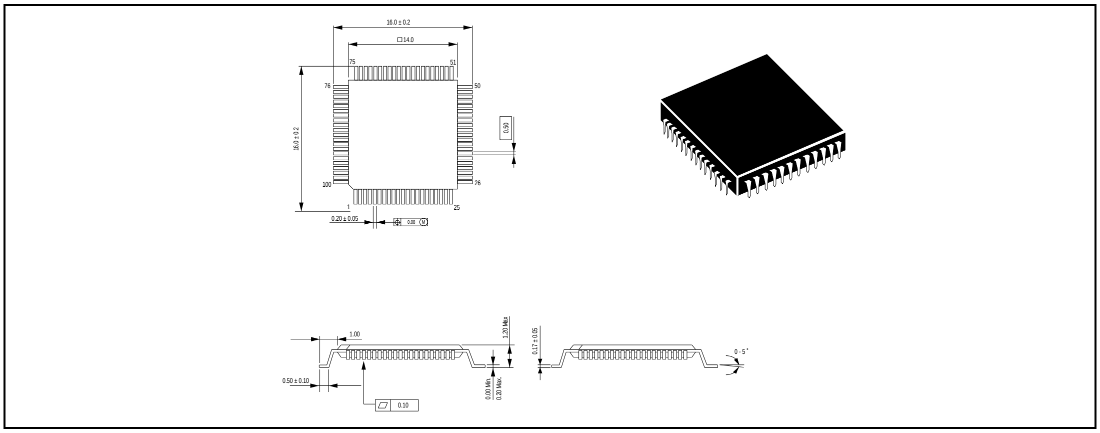


Figure 7.1 100-pin Thin Quad Flat Pack (TFP-100B)

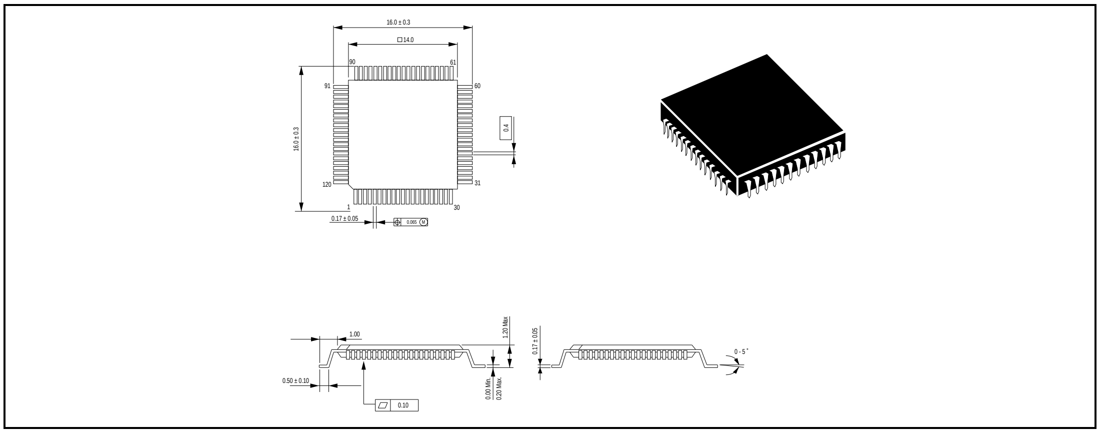


Figure 7.2 120-pin Thin Quad Flat Pack (TFP-120)

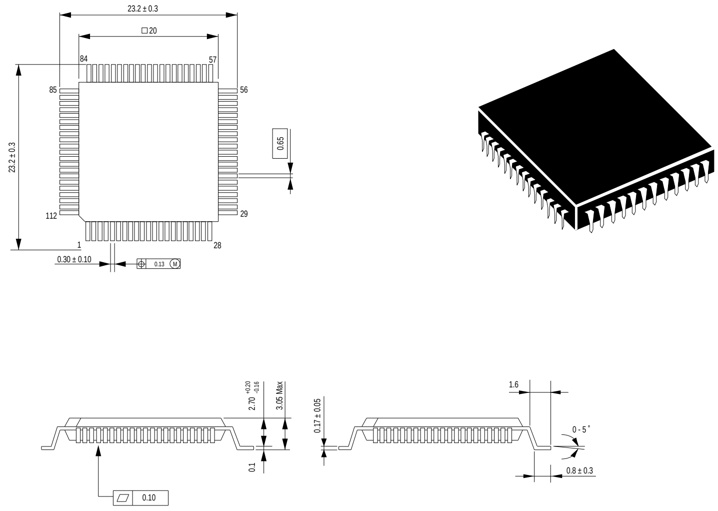


Figure 7.3 112-pin Quad Flat Pack (FP-112)

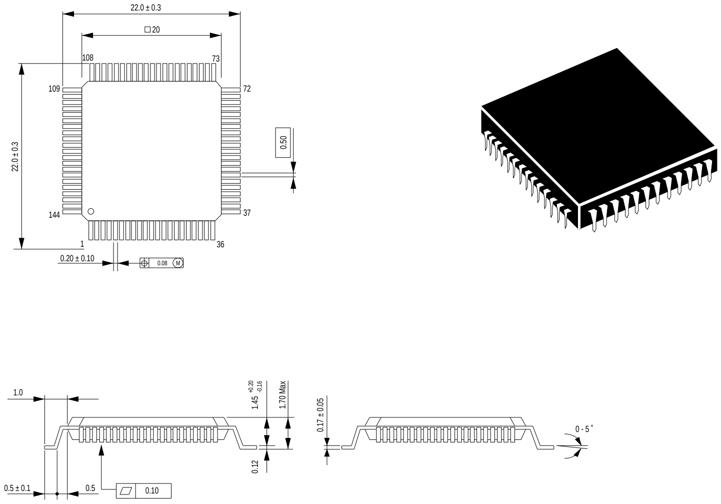


Figure 7.4 144-pin Quad Flat Pack (FP-144A)

Section 8 Support Tools

The power of any microprocessor's architecture and implementation can be fully utilized only when a complete set of development tools, both hardware and software, is available. The software-intensive nature of next-generation applications requires the availability of optimizing high-level language compilers. Ever-increasing pressure for fast product-design cycles and short time-to-market demands the use of efficient debugging tools during the design and integration phases. Ideally, the user should be able to consider a choice of tools, from several vendors, running on a variety of platforms.

For the SH series, Hitachi has directly met these needs by developing its complete suite of hardware and software tools. In addition, numerous third-party tool vendors have provided support tools for the SH. Therefore, the user has access to a wide variety of evaluation tools, development tools and operating systems.

8.1 Hitachi Tools

Hitachi's proprietary SH support tools include a C compiler suite, evaluation boards, emulators and a real-time operating system. The user's choice of development platforms includes workstations (Sun Microsystems and Hewlett Packard) and personal computers (x86-compatible).

The C Compiler suite includes a highly optimized compiler capable of generating fast, compact code for the SH-1, -2 and -3 architectures. Output files from the compiler can include object code, assembler source code and listings. The complete package includes a macro-assembler, linker, librarian, debugger and other utilities; the symbolic debugger incorporates a simulator with an accurate model of the CPU's pipeline. As a result, code debugging can begin in a software-only environment, in parallel with the OEM's system-hardware development; in fact, the debugger provides accurate measurement of the code's execution time, in terms of CPU clocks.

If desired, code can be downloaded to one of Hitachi's Evaluation Boards (EVBs) for execution. The EVBs provide a minimal operating environment for the supported processor, supplementing the on-chip hardware as necessary. Typically, a serial host-interface is provided, as is a firmware-based monitor and sufficient RAM to support it and to permit downloading of moderate-size user programs.

Hitachi's E7000 Emulator provides powerful debugging capabilities to developers of SH-based systems. Supported on Sun, HP workstations and x86-compatible PCs, it minimizes debugging time in all phases of system development. It can be used for software testing, hardware debugging, hardware-software integration and during system troubleshooting/repair.

With the E7000, user code executes, in real time, on user hardware. System resources can be mapped from the emulator into the system hardware; memory-protection mechanisms can be applied to individual blocks of memory. Code is downloaded from the host development system and executed under control of a high-level-language-oriented source-code debugger, with a graphical user interface (GUI). Debugging commands can be expressed in terms of HLL statements and data structures; results are expressed in the HLL context. Debugging can also be carried out at the low level of bytes, registers and opcodes; therefore, at any time, the approach can be chosen that is best-suited to the problem at hand. The E7000 features software breakpoints, hardware breakpoints and program tracing.

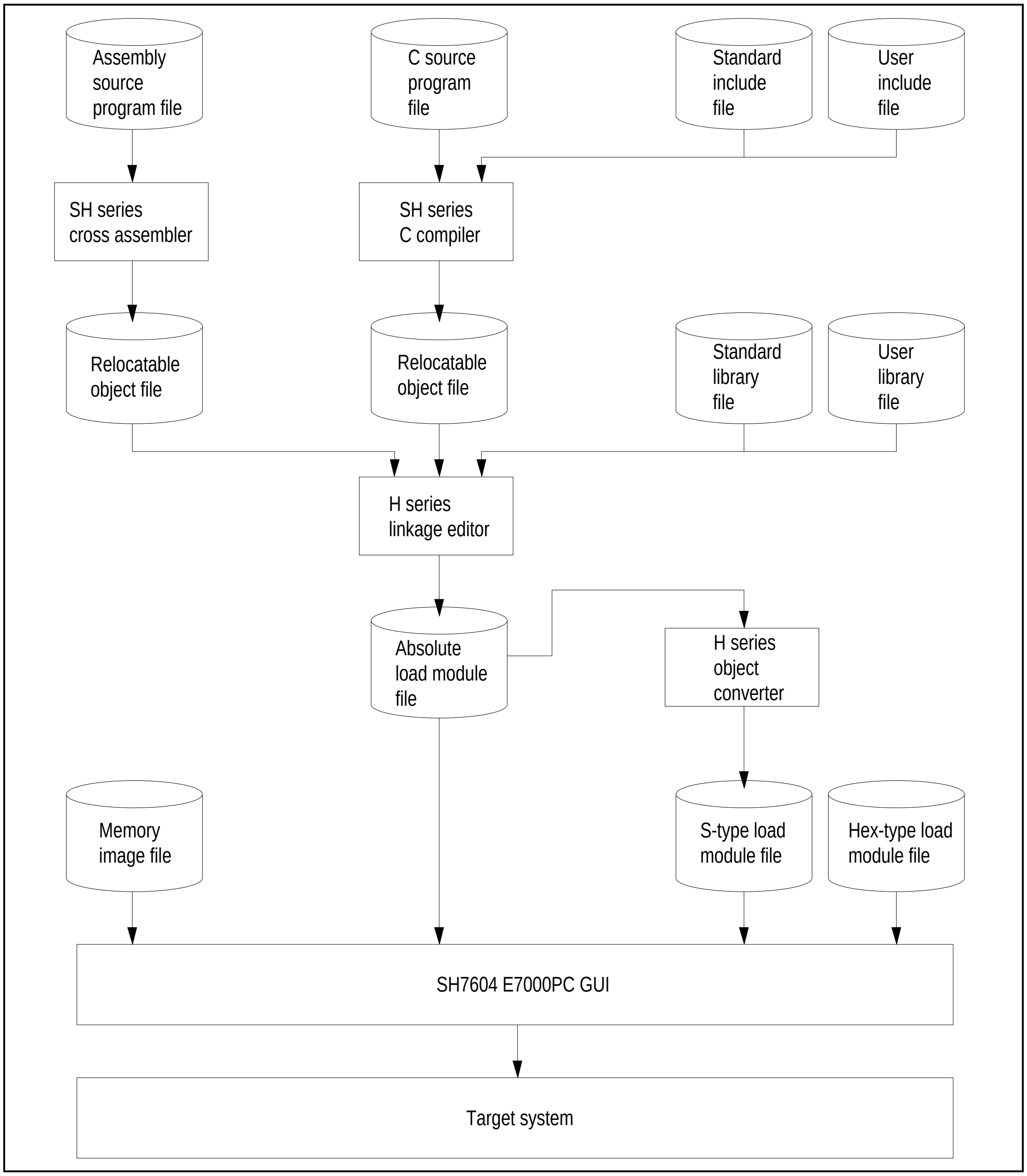


Figure 8.1 Information Flow in SH Tool Chain

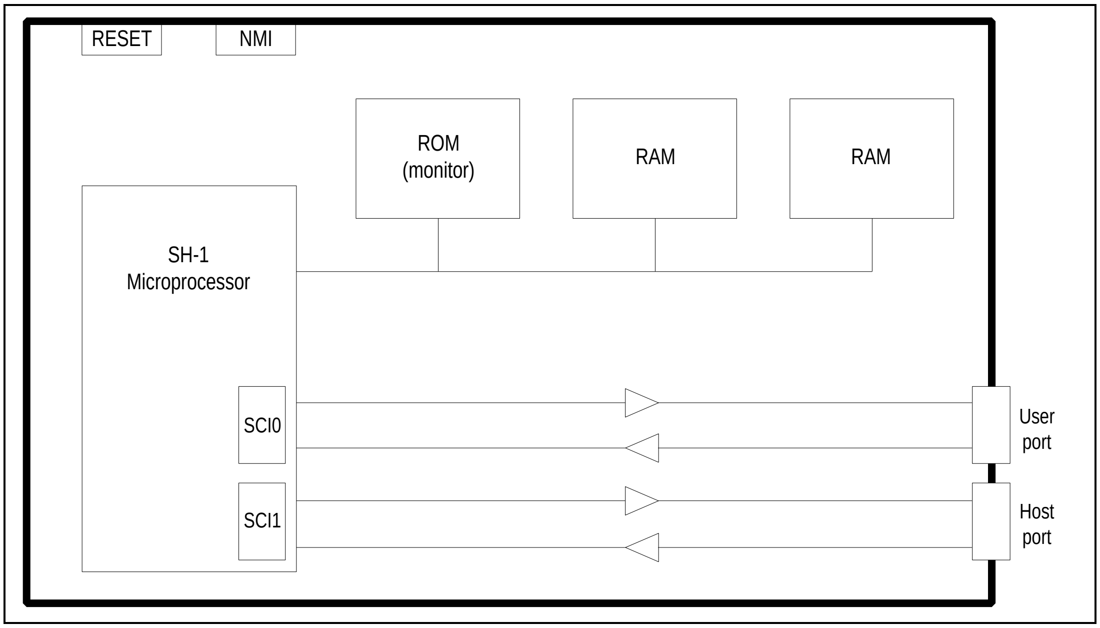


Figure 8.2 Block Diagram of Typical SH Evaluation Board (SH-1 Low-Cost EVB)

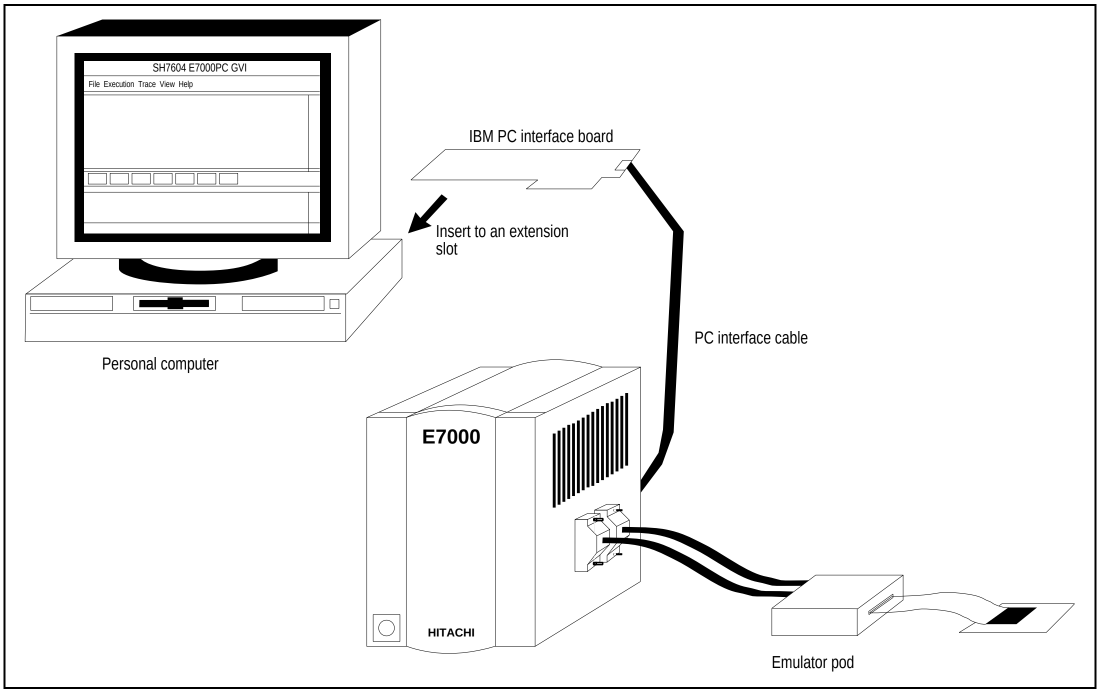


Figure 8.3 Hitachi's E7000PC Emulator

Table 8.1 E7000 Features

	Implementation		
Feature	E7000	E7000PC	
Emulation (mapping) Memory	Standard: 512 Kbytes Optional expansion boards: 1 MByte and 4-Mbyte	512 Kbytes	
Real-Time Emulation		Yes	
Trace Memory		32K cycles	
Breakpoints		Software (execution): 255 Hardware (comparison): 4	
External Probe		Inputs: 8 (SH-1) 1 (SH-2) can be traced and/or breakpointed Output: 1 trigger for scope or logic analyzer	
Code Profiling Tools		Subroutine-execution time Subroutine-call frequency	
Local Mass Storage	3-1/2" floppy disk, for: • loading/saving/verifying user programs • saving emulation results	N/A	
SH Processors Supported (package)	SH-1:	SH7020/21	(TQFP-100)
		SH7034	(QFP-112)
		SH7032	(QFP- 112)
			(TQFP- 120)
	SH-2:	SH7604	(QFP-144)
	SH-3:	SH7702	(TQFP-120)
		SH7708	(QFP-144)
Host System (architecture)	Sun Microsystems (SPARC)	Industry-standard PC (≥ '386)	
Host Operating System	Unix: SunOS (≥v4.1.1) Open Windows (≥ v2.0) X-Windows (≥ X11R4)	Windows 3.0 or later	
Host Interface (spec)	Standard: Serial (RS-232) Optional: Ethernet(10BASE5)	Parallel (proprietary) Serial (RS-232) (1 OBASE2)	
Printer Interface		Centronics (integral)	
Graphic User Interface		Yes: Multi-window environment Source-level debugger Graphical display of trace data	

8.2 Third-party Vendors

Hardware and software development tools are available from a wide variety of independent vendors; a brief summary is included here. Hitachi periodically publishes a third-party products catalog; due to the rapid growth of third-party support for the SH, please contact your Hitachi representative for a current summary. Hitachi also provides a toll-free product-information service; when calling 1-800-285-1601, please request extension 27.

Third-party offerings include computer-aided software-engineering tools, compiler suites, real-time operating systems (RTOSs) and emulators. For detailed information on specific products, please contact the vendor directly.

The Free Software Foundation's GNU tools, as targeted to the SH by Cygnus, provide a low-cost, yet extremely capable C/C++ programming environment. The source code and object code are available for the cost of the distribution media; they can, in fact, be downloaded from the Internet at no cost. As modified by Cygnus, these tools operate in either the Unix or DOS environment.

CARDtools Systems offers an innovative software front-end design tool for RTOS-based SH systems. Before hardware and software are written, CARDtools can provide a rigorous analysis of software-design integrity and system performance, based upon an architectural description of the application's design and performance parameters of the target operating system. Simulation permits alternate design approaches to be evaluated early in the design cycle, minimizing risk and reducing time-to-market.

Table 8.2 Third-Party Support Products

Product Type	Vendor	Contact Info	Product
Compiler Suites	Cygnus Support, Inc. 1937 Landings Drive Mountain View, CA 94043	Phone: (415) 903-1400 FAX: (415) 903-0122 Contact: Bill Smarzo	*CDK C* *Developer's Kit* gnu C/C++ with X-Windows GUI
	Green Hills Software, Inc. One Cranberry Hill Lexington, MA 02173	Phone: (617) 862-2002 FAX: (617) 863-2633 Contact: Tamara Calzi	C/C++ (w/*MULTI* source-level debug GUI)
Operating Systems	Accelerated Technology, Inc. P.O. Box 850245 Mobile, AL 36685	Phone: (334) 661-5770 FAX: (334) 661-5788 Contact: James E. Little	*Nucleus*+ (compatible with Hitachi, Cygnus or Green Hills compiler suite)
	Integrated Systems, Inc. 3260 Jay Street Santa Clara, CA 95054-3309	Phone: (408) 980-1500 FAX: (408) 980-0400 Contact: Lisa Christie	*pSOS* (compatible with Green Hills compiler suite)
	JMI Software Consultants, Inc. P.O. Box 481 904 Sheble Lane Spring House, PA 19477	Phone: (215) 628-0840 FAX: (215) 628-0353 Contact: Ed Rathje	C *Executive* (compatible with Cygnus compiler suite)
	Wind River Systems 1010 Atlantic Avenue Alameda, CA 94501	Phone: (800) 545-WIND Phone: (510) 748-4100 FAX: (510) 814-2010 Contact: Victoria Barnes	*VxWorks* (compatible with Cygnus compiler suite)
Application Support	CARDtools Systems, Inc. 333 Cobalt Way, Suite 107 Sunnyvale, CA 94086	Phone: (408) 559-4240 FAX: (408) 559-4246 Contact: Joseph Rothman	*CARDtools* Real-time Design Software
	Datalight 307 N. Olympic Avenue Arlington, WA 98223	Phone:(360) 435-8086 FAX: (360) 435-0253 Contact: Tim Gillman	*Card Trick* Flash File System
	The Duck Corporation 375 Greenwich Street New York, NY 10013	Phone: (212) 941-2400 FAX: (212) 941-3853 Contact: Stan Marder	Vdeo Imaging Software
	Voxware, Inc. 313 Poinsettia Avenue Manhattan Beach, CA 90266	Phone: (310)318-0722 FAX: (310) 318-1317 Contact: Kevin Bobek	Voice Recognition Software
Hardware Tools	Hewlett Packard 1900 Garden of the Gods Road P.O. Box 2197 Colorado Springs, CO 80901-2197	Phone: (719) 590-5311 Contact: Walter Uglow	HP6400 Emulator
	Nissho Electronics (USA) Corp. 18201 Karman Avenue, Suite 350 Irvine, CA 92715	Phone: (714) 261-8811x129 FAX: (714) 261-0934 Contact: Lisa Murimoto	Densan (Japan) VME Boards
	Orion Instruments 1376 Borregas Avenue Sunnyvale, CA 94089-1004	Phone: (800) 729-7700 Phone: (408) 747-0440 FAX: (408) 747-0688 Contact: Jan Liband	Advice ICE Emulator
	Sophia Systems 711 B Charcot Avenue San Jose, CA 95131	Phone: (408) 943-9300 FAX: (408)943-9303 Contact: David Freemire	Emulator