Advanced Embedded Microcontroller Seminar--Day 2

Andrew Huang
bunnie@mit.edu

Agenda

• Day 2: 1/21
  – Designing the SH-1 into your system
  – Details of the SH1 integrated peripherals
  – Overview of the SH1WH, on-board peripheral specs
  – development environment
SH-1 Block Diagram

Location of Hardware Manual

  - difficult to read sometimes, but the comprehensive source of information on the SH-1
  - today’s lecture distills and presents some of the most important concepts to get you started
Getting in to the SH-1: Features

- 32-bit RISC CPU with MAC instruction
- interrupt controller
  - 9 external interrupt pins
  - 31 internal interrupt sources
  - 16 priority levels
- user break controller
  - for trapping on special conditions, debugging
- clock generator
  - drives an external crystal to make internal timebase
- bus state controller
  - 8/16 bit accesses
  - muxed or separate address and data
  - controls DRAM, SRAM, ROM, or peripherals
  - generates up to 8 chip selects
  - wait states
- DMA controller
  - 4 channels, 2 internal, 2 external
  - transfer between external and internal memory, I/O
  - cycle-steal and burst-mode

- 16-bit integrated timer unit
  - measures input pulse widths
  - generates PWM, complementary PWM
  - phase counting mode
- timing pattern controller
  - can be used in conjunction with DMAC and ITU to generate arbitrary pulse trains with no load on CPU
  - non-overlapping mode available
  - applications include multi-phase motor control, serial communications (SPI, microwire, etc.)
- watchdog timer
- serial comm. interface
  - 2 channels
  - async/synch options
  - internal baud rate generator
- A/D converter
  - 10 bits, 8 channels
  - external trigger, reference source
- digital I/O ports
  - up to 40 lines, but all shared with other I/O functions
- on-chip RAM/ROM
  - SH7032 (ROM-less) has 8K RAM
  - SH7034 has 64K PROM plus 4K RAM
  - single-cycle access to on-chip RAM/ROM

1/21/99 Microcontroller Seminar -- ASH 5

1/21/99 Microcontroller Seminar -- ASH 6

1/21/99 Microcontroller Seminar -- ASH 6
SH-1: Key Design Point

- All peripherals and memory are mapped to one common address space
  - 28-bit effective address space is divided into 16 equally sized sectors
    - 28 bits because top 4 bits of a 32-bit address are don’t care
  - actual usage of allocated space per sector varies; undesignated areas alias to designated areas
  - processor bus behavior (i.e., 8 or 16 bit mode, wait states, multiplexed addr, data) determined by which address is accessed
    - details of behavior can be modified by setting memory-mapped registers in the BSC
- Vector table is located at bottom of memory
  - hence there must always be a ROM mapped to CS0

Memory organization of the SH-1

Output address: Output from address pins A21–A0
Ignored: Only valid when the address multiplex function is being used in the DRAM space (area 1); not output in other cases. When not output, becomes shadow.
Area selection: Decoded to become chip select signals CS0—CS7 for areas 0–7

Basic bus width selection:
- Not output externally, but used for basic bus width selection
  - When 0 (H’00000000–H’7FFFFFFF), the basic bus width is 8 bits.
  - When 1 (H’80000000–H’FFFFFFF), the basic bus width is 16 bits.

Ignored: Always ignored, not output externally
Function allocation in the SH-1

<table>
<thead>
<tr>
<th>Area</th>
<th>Address</th>
<th>Assignable Memory</th>
<th>Capacity (Linear Space)</th>
<th>Bus Width</th>
<th>CS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H'0000000-0xFFFFFFF</td>
<td>On-chip ROM*</td>
<td>64 kB</td>
<td>32</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>H'1000000-1xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8/16*</td>
<td>CS0</td>
</tr>
<tr>
<td>2</td>
<td>H'2000000-2xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8</td>
<td>CS1</td>
</tr>
<tr>
<td>3</td>
<td>H'3000000-3xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8</td>
<td>CS2</td>
</tr>
<tr>
<td>4</td>
<td>H'4000000-4xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8</td>
<td>CS3</td>
</tr>
<tr>
<td>5</td>
<td>H'5000000-5xFFFFFFF</td>
<td>On-chip supplying modules</td>
<td>512 B</td>
<td>8/16*</td>
<td>—</td>
</tr>
<tr>
<td>6</td>
<td>H'6000000-6xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8/16*</td>
<td>CS4</td>
</tr>
<tr>
<td>7</td>
<td>H'7000000-7xFFFFFFF</td>
<td>Multiplied I/O</td>
<td>4 MB</td>
<td>—</td>
<td>CS5</td>
</tr>
<tr>
<td>8</td>
<td>H'8000000-8xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>8</td>
<td>CS6</td>
</tr>
<tr>
<td>9</td>
<td>H'9000000-9xFFFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>CS7</td>
</tr>
<tr>
<td>10</td>
<td>H'A000000-A'FFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>—</td>
</tr>
<tr>
<td>11</td>
<td>H'B000000-B'FFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>—</td>
</tr>
<tr>
<td>12</td>
<td>H'C000000-C'FFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>H'D000000-D'FFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>H'E000000-E'FFFFFF</td>
<td>External memory²</td>
<td>4 MB</td>
<td>16</td>
<td>—</td>
</tr>
<tr>
<td>15</td>
<td>H'F000000-F'FFFFFF</td>
<td>On-chip RAM</td>
<td>8 kB*</td>
<td>32</td>
<td>—</td>
</tr>
</tbody>
</table>

BSC registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbr.</th>
<th>R/W</th>
<th>Initial Value</th>
<th>Address*¹</th>
<th>Bus width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus control register</td>
<td>BCR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFA0</td>
<td>8,16,32</td>
</tr>
<tr>
<td>Wait state control register 1</td>
<td>WCR1</td>
<td>R/W</td>
<td>H'FFFF</td>
<td>H'5FFFFFA2</td>
<td>8,16,32</td>
</tr>
<tr>
<td>Wait state control register 2</td>
<td>WCR2</td>
<td>R/W</td>
<td>H'FFFF</td>
<td>H'5FFFFFA4</td>
<td>8,16,32</td>
</tr>
<tr>
<td>Wait state control register 3</td>
<td>WCR3</td>
<td>R/W</td>
<td>H'FB00</td>
<td>H'5FFFFFA6</td>
<td>8,16,32</td>
</tr>
<tr>
<td>DRAM area control register</td>
<td>DCR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFA8</td>
<td>8,16,32</td>
</tr>
<tr>
<td>Parity control register</td>
<td>PCR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFAA</td>
<td>8,16,32</td>
</tr>
<tr>
<td>Refresh control register</td>
<td>RCR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFAC</td>
<td>8,16,32¹²</td>
</tr>
<tr>
<td>Refresh timer control/status register</td>
<td>RTCSR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFAE</td>
<td>8,16,32¹²</td>
</tr>
<tr>
<td>Refresh timer counter</td>
<td>RTCNT</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFFB0</td>
<td>8,16,32¹²</td>
</tr>
<tr>
<td>Refresh time constant register</td>
<td>RTCOR</td>
<td>R/W</td>
<td>H'00FF</td>
<td>H'5FFFFFB2</td>
<td>8,16,32¹²</td>
</tr>
</tbody>
</table>
Registers of the BSC

- BCR
  - determines basic signal configuration (DRAM mode on area 1, mux’d adr/data on area 6, warp mode, RD# duty cycle, byte access style

- WCR[1:3]
  - configures wait states for normal, DMA accesses

- DCR
  - configures basic DRAM access timing (dual CAS, precharge time, FPM, CAS duty cycle)

- RCR
  - configures DRAM refresh timing (CBR, self-refresh, wait states)

- RTCSR, RTCNT
  - configures refresh period based on system clock

- PCR
  - configures parity checking modes

PFC

- Pin function controller
  - determines functions of multiplexed pins
  - almost every I/O pin has multiple functions
  - must initialize to select proper mode prior to using any peripheral function
  - many basic services, such as some chip select lines, RD#, WR#, CAS/RAS can be turned on or off by writing to this register
  - fastest way to wedge the system is to write the wrong info to these registers
PFC registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbreviation</th>
<th>R/W</th>
<th>Initial Value</th>
<th>Address</th>
<th>Access Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A I/O register</td>
<td>PAIOR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFC4</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Port A control register 1</td>
<td>PACR1</td>
<td>R/W</td>
<td>H'3302</td>
<td>H'5FFFFC8</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Port A control register 2</td>
<td>PACR2</td>
<td>R/W</td>
<td>H'FF95</td>
<td>H'5FFFFCA</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Port B I/O register</td>
<td>PBDR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFC6</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Port B control register 1</td>
<td>PBCR1</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFCC</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Port B control register 2</td>
<td>PBCR2</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFCE</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Column address strobe pin control register</td>
<td>CASCR</td>
<td>R/W</td>
<td>H'FFFFF</td>
<td>H'5FFFFEE</td>
<td>8, 16, 32</td>
</tr>
</tbody>
</table>

- P{A,B}CR{1,2} configures which mux’d function to use
- P{A,B}IOR configures, on a pin by pin basis, input or output mode if digital I/O mode is selected
- CASCR determines function of mux’d CS1,3 and CASL,H lines

Digital I/O

- Referred to as parallel I/O in the hardware manual
  - any of port A or B can be configured to be input or output
  - port C is input only
  - ports A and B are each 16 bits wide at max, port C is 8 bits wide at max
  - port C is mux’d with analog input, and automagically functions as analog input when A/D is activated
- PADR, PBDR, and PCDR are R/W registers that will either reflect the data on the pin, or contain the data to be driven on pin, depending on pin configuration
Interrupt Controller

- 16 priority levels
- up to 255 interrupt vectors
  - about 43 are actually assigned to interrupt sources
  - vector table starts at memory location 0, each vector consisting of a 32-bit pointer to code to handle the interrupt
  - power on reset and manual reset each has two vectors, the first for setting the PC, the second for setting the SP
- 8 external interrupt inputs
- IRQOUT line to indicate presence of internal interrupts
- edge or level triggered, programmable by software

Interrupts

- on interrupt,
  - status register is pushed (32-bit)
  - return PC is pushed (32-bit)
  - note that SP should always be a multiple of 4
- interrupt response latency
  - latency dependant upon how fast long word accesses to memory occur
  - min. latency is 11 cycles (8 cycles for internal pipes to flush, plus 3 cycles for vector fetch and stack pushes)
  - max. latency is $12 + 2(SR, PC, vector access times) + \text{fetch time}$ for first handler instruction
  - typ. latency around 1 to 2 us using slow memory at 12 MHz
- see table 4.10 in hardware manual for precise behavior of machine on a wide range of conditions
Interrupt Controller Hardware

Interrupt Processing Flow Chart
Interrupt Controller Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbr.</th>
<th>R/W</th>
<th>Address*</th>
<th>Initial Value</th>
<th>Bus width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt priority register A</td>
<td>IPRA</td>
<td>R/W</td>
<td>H'5FFFF84</td>
<td>H'0000</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Interrupt priority register B</td>
<td>IPRB</td>
<td>R/W</td>
<td>H'5FFFF86</td>
<td>H'0000</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Interrupt priority register C</td>
<td>IPRC</td>
<td>R/W</td>
<td>H'5FFFF88</td>
<td>H'0000</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Interrupt priority register D</td>
<td>IPRD</td>
<td>R/W</td>
<td>H'5FFFF8A</td>
<td>H'0000</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Interrupt priority register E</td>
<td>IPRE</td>
<td>R/W</td>
<td>H'5FFFF8C</td>
<td>H'0000</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>Interrupt control register ICR</td>
<td>R/W</td>
<td>H'5FFFF8E</td>
<td>H'0000</td>
<td>8, 16, 32</td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1. H'8000 when pin NMI is high, H'0000 when pin NMI is low.

- ICR sets NMI detection parameters and edge or level trigger of external interrupt sources

Programming the Vector Table

- Vector table is located in FLASH ROM
  - requires erasing part of ROM monitor and reprogramming
  - utilities will be provided to assist in this risky process
  - once your handler is debugged, it is recommended to burn it into some unused sector of FLASH so as to avoid accidental corruption
### ITU Register Cross-Reference Matrix

<table>
<thead>
<tr>
<th>Register Setting</th>
<th>Register</th>
<th>TDIR</th>
<th>TMR2</th>
<th>TMR1</th>
<th>TCNT</th>
<th>TRFR</th>
<th>TRCR</th>
<th>TIOR0</th>
<th>TCR0</th>
<th>TIOR1</th>
<th>TRCR1</th>
<th>TRFR1</th>
<th>TIOR2</th>
<th>TRCR2</th>
<th>TRFR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Mode</td>
<td>Sync</td>
<td>MDF</td>
<td>FDIR</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
</tr>
<tr>
<td></td>
<td>Sync</td>
<td>MDF</td>
<td>FDIR</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
</tr>
<tr>
<td></td>
<td>Sync</td>
<td>MDF</td>
<td>FDIR</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
</tr>
<tr>
<td></td>
<td>Sync</td>
<td>MDF</td>
<td>FDIR</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
<td>PWM</td>
<td>Comp</td>
</tr>
</tbody>
</table>

See tables at end of ITU section in hardware manual for complete matrix.

---

### Simple ITU Example

1. TCNT value
2. Counter cleared by GR compare match
3. Time
4. STR0–STR4
5. IMF
6. Select counter clock
7. Counting mode selection
8. Start counting
9. Stop counting
10. Free-running counter
11. Periodic counter
12. Select output compare register
13. Select counter clear source
14. Set period
15. Periodic counter
16. Free-running counter
ITU output modes

- TCNT value
- Counter cleared at GRB compare match
- Time
- Toggle output

ITU measuring intervals

- TCNT value
- Counter cleared by TIOCB input
- Time
- Input selection
- Select input-capture input
- Start counting
- Capture
ITU in PWM mode

1. Select counter clock
2. Select counter clear source
3. Set GRA
4. Set GRB
5. Select PWM mode
6. Start counting

ITU complementary PWM mode

- Non-overlapping complementary PWM outputs
ITU in phase counting mode

- Use quadrature to figure out direction of rotation in shaft encoders

Direct Memory Access Controller

- 4-channel
  - 2 internal
  - 2 external
- 64K transfers at a time max.
- Addressing modes
  - single and dual address modes (for DREQ activated devices)
  - address increment of -2, -1, 0, 1, 2
- Transfer request modes
  - external from DREQ
  - internally generated signals from A/D, ITU, SCI
- Cycle steal and burst mode options
- Allows transfers to happen with no CPU loading
DMAC block diagram

DMAC: DMA operation register
SARn: DMA source address register
DARn: DMA destination address register
TCRn: DMA transfer count register
CHCRn: DMA channel control register
DEIn: DMA transfer-end interrupt requested to CPU n: 0-3

DMAC registers

<table>
<thead>
<tr>
<th>Channel</th>
<th>Name</th>
<th>Abbreviation</th>
<th>R/W</th>
<th>Initial Value</th>
<th>Address</th>
<th>Access Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DMA source address register 0</td>
<td>SAR0</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF40</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA destination address register 0</td>
<td>DAR0</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF44</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer count register 0</td>
<td>TCR0</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF4A</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA channel control register 0</td>
<td>CHCR0</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFF4E</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer-end interrupt request to CPU 0</td>
<td>DEI0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DMA source address register 1</td>
<td>SAR1</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF50</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA destination address register 1</td>
<td>DAR1</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF54</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer count register 1</td>
<td>TCR1</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF5A</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA channel control register 1</td>
<td>CHCR1</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFF5E</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer-end interrupt request to CPU 1</td>
<td>DEI1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DMA source address register 2</td>
<td>SAR2</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF60</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA destination address register 2</td>
<td>DAR2</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF64</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer count register 2</td>
<td>TCR2</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF6A</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA channel control register 2</td>
<td>CHCR2</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFF6E</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer-end interrupt request to CPU 2</td>
<td>DEI2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DMA source address register 3</td>
<td>SAR3</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF70</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA destination address register 3</td>
<td>DAR3</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF74</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer count register 3</td>
<td>TCR3</td>
<td>RW</td>
<td>Undefined</td>
<td>H'5FFFF7A</td>
<td>16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA channel control register 3</td>
<td>CHCR3</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFF7E</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td></td>
<td>DMA transfer-end interrupt request to CPU 3</td>
<td>DEI3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shared DMA operation register</td>
<td>DMOCR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFF48</td>
<td>8, 16, 32</td>
</tr>
</tbody>
</table>
### DMAC Usage

1. Initial settings (SAR, DAR, TCR, CHCR, DMAOR)
2. Transfer (1 transfer unit); TCR→TCR, SAR and DAR updated
3. DEI interrupt request (when IE = 1)

**Notes:**
1. In auto-request mode, transfer begins when NMIF, AE, and TE are all 0 and the DE and DME bits are set to 1.
2. DREQ = level detection in burst mode (external request), or cycle steal mode.
3. DREQ = edge detection in burst mode (external request), or auto request mode in burst mode.

### Timing Pattern Controller

- 4 bits x 4 TPC units digital output
- Typically used in conjunction with ITU and DMAC
- Used to synthesize pulse trains whose details are determined by ITU timing and a sequence of bits stored in memory
  - Good for doing serial protocols
TPC Block Diagram

TPC Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbreviation</th>
<th>R/W</th>
<th>Initial Value</th>
<th>Address*1</th>
<th>Access Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port B control register 1</td>
<td>PBCR1</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFCC</td>
<td>8, 16</td>
</tr>
<tr>
<td>Port B control register 2</td>
<td>PBCR2</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFCE</td>
<td>8, 16</td>
</tr>
<tr>
<td>Port B data register</td>
<td>PBDR</td>
<td>R/W</td>
<td>H'0000</td>
<td>H'5FFFFC2</td>
<td>8, 16</td>
</tr>
<tr>
<td>TPC output mode register</td>
<td>TPMR</td>
<td>R/W</td>
<td>H'FF</td>
<td>H'5FFFFF0</td>
<td>8, 16</td>
</tr>
<tr>
<td>TPC output control register</td>
<td>TPCR</td>
<td>R/W</td>
<td>H'FF</td>
<td>H'5FFFFF1</td>
<td>8, 16</td>
</tr>
<tr>
<td>Next data enable register B</td>
<td>NDERB</td>
<td>R/W</td>
<td>H'00</td>
<td>H'5FFFFF2</td>
<td>8, 16</td>
</tr>
<tr>
<td>Next data enable register A</td>
<td>NDERA</td>
<td>R/W</td>
<td>H'00</td>
<td>H'5FFFFF3</td>
<td>8, 16</td>
</tr>
<tr>
<td>Next data register A</td>
<td>NDRA</td>
<td>R/W</td>
<td>H'00</td>
<td>H'5FFFFF0/</td>
<td>8, 16</td>
</tr>
<tr>
<td>Next data register B</td>
<td>NDRB</td>
<td>R/W</td>
<td>H'00</td>
<td>H'5FFFFF0/</td>
<td>8, 16</td>
</tr>
</tbody>
</table>
TPC usage flow chart

1/21/99 Microcontroller Seminar -- ASH

TPC + DMA + ITU example

1/21/99 Microcontroller Seminar -- ASH
A/D Converter

- 8 channels, multiplexed
- 10 bits/channel
- Integrated S/H with internal or external sample trigger
- Successive Approximation
- Vref tied to 2.5V on SH1WH => 2.44 mV/LSB
- 11.2 us per channel conversion rate
- Can trigger DMAC for background sampling
- Scan mode capable (sample from each channel round-robin style)
- polled or interrupt mode for A/D completion

A/D Converter Block Diagram
A/D converter registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Abbreviation</th>
<th>R/W</th>
<th>Initial Value</th>
<th>Address*</th>
<th>Access Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D data register A (high)</td>
<td>ADDRAH</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE0</td>
<td>8, 16</td>
</tr>
<tr>
<td>A/D data register A (low)</td>
<td>ADDRAL</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE1</td>
<td>16</td>
</tr>
<tr>
<td>A/D data register B (high)</td>
<td>ADDRBH</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE2</td>
<td>8, 16</td>
</tr>
<tr>
<td>A/D data register B (low)</td>
<td>ADDRBL</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE3</td>
<td>16</td>
</tr>
<tr>
<td>A/D data register C (high)</td>
<td>ADDRCH</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE4</td>
<td>8, 16</td>
</tr>
<tr>
<td>A/D data register C (low)</td>
<td>ADDRCL</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE5</td>
<td>16</td>
</tr>
<tr>
<td>A/D data register D (high)</td>
<td>ADDRDH</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE6</td>
<td>8, 16</td>
</tr>
<tr>
<td>A/D data register D (low)</td>
<td>ADDRDL</td>
<td>R</td>
<td>H'00</td>
<td>H'05FFFEE7</td>
<td>16</td>
</tr>
<tr>
<td>A/D control/status register</td>
<td>ADCSR</td>
<td>R/(W)*2</td>
<td>H'00</td>
<td>H'05FFFEE8</td>
<td>8, 16</td>
</tr>
<tr>
<td>A/D control register</td>
<td>ADCR</td>
<td>RW</td>
<td>H'7F</td>
<td>H'05FFFEE9</td>
<td>8, 16</td>
</tr>
</tbody>
</table>

Embedded RAM

- 8K of fast RAM
  - 1 cycle access time read/write
  - 32-bit wide datapath
- Use for inner loops and time-critical processing
  - like a user managed cache
- Located from 0xF000000 to 0xF002000
Power Down Modes

- Power down modes entered by executing SLEEP instruction
  - sleep or standby set by changing a bit in the SBYCR
- sleep mode
  - clock runs, CPU halts, all peripherals run
  - exit through interrupt, DMA address error, power on reset, manual reset
- standby mode
  - clock halts, all functions stop
  - exit through NMI, power on reset, or manual reset

Building a system

- Minimal SH-1 system
  - SH-1 + embedded ROM/RAM, crystal
- SH1WH system
  - SH-1
  - 8 MB DRAM
  - 128K battery backed SRAM
  - 1 MB ISP FLASH
  - D/A converter
  - RTC
  - RS-232 tranceivers
  - Extra digital I/O
  - Switching regulator
SH1WH RTC

- Dallas Semiconductor DS1670 system controller/RTC
- Serial interface to RTC
  - CLK, DIN/DOUT, CS
- Power fail manager
  - turns off SRAM CS, resets processor on low Vcc condition
- 3-channel, 8-bit A/D converter
  - used for temperature sensor
- Wake-up alarm clock
- Watchdog timer

RTC registers, serial interface

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16 SECONDS</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>15 MINUTES</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>24 HOURS</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>24 HOURS ALARM</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>10 MINUTES ALARM</td>
<td>0</td>
</tr>
</tbody>
</table>

1/21/99 Microcontroller Seminar -- ASH
D/A converter

- **LTC1454L D/A converter**
  - 12-bit resolution
  - 1.220V reference + x2 amplifier => 2.44V effective full-scale
  - power-on reset to 0
  - 14 us typ. settling time, 1 V/us typ. slew rate
D/A converter

FLASH ROM

- 1 MB, 3.3V only boot-block ROM with autoshutdown
- embedded algorithms for erase and program
  - writing a special sequence to the FLASH causes erasing and programming to occur
- like all FLASH, an entire sector must be erased before being written to
  - erase defaults to ‘1’
  - programming will clear respective bits to ‘0’
  - possible to program over if all you’re doing is setting bits to 0
- drivers provided for your convenience in using the FLASH
  - not recommended to muck around too much with the boot block, as losing the boot block means you’ll have to desolder the ROM and burn it in a programmer
SH1WH Memory Map

SRAM and FLASH Memory Map
Development Environment

- Host environment
  - GNU development environment
    - gcc, make, binutils, etc.
      - cross-compiler to SH-COFF format can be built for a large number of platforms
    - all software developed under Linux 2.0.32 with emacs as editor
      - DOS version of development environment to be released soon
    - java applet called JTerm which runs under windows is used to upload code binaries and interact with monitor
      - any serial terminal program can do interaction, but upload protocol is proprietary (basically straight binary plus header and trailer bytes)
      - uses Sun javax.comm extensions, only supports windows and Sun OS platforms but will be part of java spec soon
    - communication via serial port, 38400 N81

- Target environment
  - Interactive ROM monitor available at all times
    - manipulate and examine memory, registers, code
    - base level diagnostics
  - User settable power-on environment
    - can go straight to user code in battery-backed SRAM or FLASH for stand-alone operation
    - can go straight into monitor for debugging operations
    - panic mode to drop into monitor in case user code fails
  - FLASH burning, RTC manipulation, FPGA programming all provided as uploadable binaries
  - Code development starts with a set of template files which contains necessary code to initialize SP, jump to main; you just need to add your code and compile
Development Environment

- Things to avoid doing:
  - Don’t bash the FLASH, particularly the ROM monitor area
  - Don’t set the RTC watchdog timer unless you have code to turn it off
  - Doing either of the above will require desoldering components or pins off of the board to recover from