Overview

- The MCF5272 ColdFire is a 32-bit, 66 MHz processor based on the Version 2 ColdFire Core
- 4K Internal SRAM
- 16K Internal ROM
- 1K Instruction Cache
- Fully static operation with stop and sleep mode
- Individual module clock enables
- Very rapid response times to interrupts from low power and sleep modes
- Software-controlled disable of external clock input for virtually zero power consumption
- Operating voltage of 3.3 v

Peripherals

- IEEE 802.3 compliant 10/100 Fast Ethernet Controller (FEC), with dedicated DMA
- USB 1.1 device controller and transceiver
- 4 TDM ports
- PLIC module – Intended for ISDN designs
- QSPI – 16 stacked transfers
- SDRAM controller
- 3 PWM outputs
- 2 UARTs – baud rates up to 5 Mbps
- 1-channel DMA
- 8 chip selects
- 16-bit general purpose I/Os
- 4 16-bit timers and SW watchdog timer
- 16 general purpose 32-bit registers, A0-A7 and D0-D7
- 32-bit Program Counter
- 8-bit Condition Code Register
- Data registers can be accessed as 8, 16, or 32 bit values and can be used as index registers
- A0-A6 can be used as software stack pointers, index registers, or base address registers
- A7 is the system Stack Pointer
Condition Code Register

- Bits 7:5 are reserved and should always be written to 0
- X – Assigns the value of a carry bit
- N – Negative flag
- Z – Zero condition code bit
- V – Overflow condition code bit
- C – Carry condition code bit

Supervisor Programming Model

- Supervisor functions, performed by the MOVEC instruction, enable programmers to implement Operating System functions such as memory and I/O control.
- It enables configuring CPU attributes such as cache enable.

The Status Register SR

- The SR register provides processor status, interrupt priority mask, and other control bits.
- The Supervisor can read/write the entire SR register, but the user is limited to read/write of SR[7:0]

<table>
<thead>
<tr>
<th>Bit</th>
<th>Supervisor Programming Model</th>
<th>The Status Register SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Trace enable</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Supervisor vs. User state</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Master Interrupt State</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Interrupt Priority Mask</td>
<td></td>
</tr>
</tbody>
</table>
Vector Base Address Register

- The VBR is an address register which indicates where in memory the processor vector table is stored
- The ColdFire processor supports a 1024 vector table aligned on any 1 MB address boundary
- The Vector table contains 256 entries, the first 64 of which are defined by Motorola and the remaining 192 are user defined vector addresses

<table>
<thead>
<tr>
<th>Field</th>
<th>Function</th>
<th>vector table base address</th>
<th>256 entries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parity</td>
<td>0000_0000_0000_0000_0000_0000_0000_0000</td>
<td>1024 entries</td>
</tr>
</tbody>
</table>

![Figure 3A: Vector Base Register (VBR)]

Exception Processing

- The ColdFire processor inhibits interrupts during the first instruction of all exception handlers
- This enables the programmer to disable interrupts by raising the interrupt mask level in the status register
- There are many sources for exception, however the following usually indicate a fatal error:

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0 = 0000_0000</td>
<td>0000</td>
<td>Reset</td>
</tr>
<tr>
<td>X0 = 0000_0001</td>
<td>0001</td>
<td>System Reset</td>
</tr>
<tr>
<td>X0 = 0000_0010</td>
<td>0010</td>
<td>Illegal Instruction</td>
</tr>
<tr>
<td>X0 = 0000_0011</td>
<td>0011</td>
<td>Illegal Instruction, int 0</td>
</tr>
<tr>
<td>X0 = 0000_0100</td>
<td>0100</td>
<td>Access Violation</td>
</tr>
<tr>
<td>X0 = 0000_0101</td>
<td>0101</td>
<td>Access Violation, int 0</td>
</tr>
<tr>
<td>X0 = 0000_0110</td>
<td>0110</td>
<td>Software Breakpoint</td>
</tr>
<tr>
<td>X0 = 0000_0111</td>
<td>0111</td>
<td>Software Breakpoint, int 0</td>
</tr>
<tr>
<td>X0 = 0000_1000</td>
<td>1000</td>
<td>Numeric Exception</td>
</tr>
<tr>
<td>X0 = 0000_1001</td>
<td>1001</td>
<td>Numeric Exception, int 0</td>
</tr>
<tr>
<td>X0 = 0000_1010</td>
<td>1010</td>
<td>Numeric Exception, int 0</td>
</tr>
<tr>
<td>X0 = 0000_1011</td>
<td>1011</td>
<td>Numeric Exception, int 0</td>
</tr>
</tbody>
</table>

![Figure 4: Exception Handling]
Exceptions

- Access Error
  - Caused by an error when accessing memory
- Address Error
  - Caused by addressing an odd instruction address (bit 0 of address is set)
- Illegal Instruction
  - Executing illegal opcodes such as 0x0000 and 0x4AFC
- Divide by Zero
- Privilege Violation
  - Attempted execution of a supervisor mode when in user mode

Interrupts

- There are four interrupt control registers ICR1-4, which are used to assign interrupt levels to interrupt sources
- The ISR register allows for reading the instantaneous values of an interrupt source
- The PITR specifies transition level of interrupt
- The PIWR specifies which interrupts can bring the CPU out of sleep mode
- The PIVR specifies which vector number is returned in response to an interrupt acknowledge cycle
- Interrupt Priorities are set through IPL, but if they all have the same IPL, priority is established through:
### Interrupt Priority

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ALRM</td>
<td>External alarm signal</td>
</tr>
<tr>
<td>1</td>
<td>RSC</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>TSC</td>
<td>Timer channel</td>
</tr>
<tr>
<td>3</td>
<td>ESC</td>
<td>External scan</td>
</tr>
<tr>
<td>4</td>
<td>CSC</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>OIC</td>
<td>System time out</td>
</tr>
<tr>
<td>6</td>
<td>TIM0, TIM1, TIM2, TIM3</td>
<td>Timer channels (see below)</td>
</tr>
<tr>
<td>7</td>
<td>I/O</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>PLL</td>
<td>PLL clock</td>
</tr>
<tr>
<td>9</td>
<td>SSA</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>ISSA</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>NISA</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>UFR</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>SCI</td>
<td>Serial communication interface</td>
</tr>
<tr>
<td>14</td>
<td>RC</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>ERI</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>SPI</td>
<td>Serial peripheral interface</td>
</tr>
<tr>
<td>17</td>
<td>SPS</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>PSPI</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>ECB</td>
<td>External bus controller</td>
</tr>
<tr>
<td>20</td>
<td>ECH</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>CMU</td>
<td>Central multiprocessor</td>
</tr>
<tr>
<td>22</td>
<td>ICR</td>
<td>Interrupt controller register</td>
</tr>
<tr>
<td>23</td>
<td>ICR2</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Interrupt Controller Registers

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>ICR1</td>
<td>Interrupt controller register</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Interrupt Control Register ICR1

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

17
### Interrupt Vector Table

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Bit 4-6</th>
<th>Source</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>0000</td>
<td>Reserve</td>
<td>User Software Interrupt</td>
</tr>
<tr>
<td>05</td>
<td>0001</td>
<td>INT1</td>
<td>External Interrupt 1</td>
</tr>
<tr>
<td>06</td>
<td>0010</td>
<td>INT2</td>
<td>External Interrupt 2</td>
</tr>
<tr>
<td>07</td>
<td>0011</td>
<td>INT3</td>
<td>External Interrupt 3</td>
</tr>
<tr>
<td>08</td>
<td>0100</td>
<td>INT4</td>
<td>External Interrupt 4</td>
</tr>
<tr>
<td>09</td>
<td>0101</td>
<td>TIM0</td>
<td>Timer 0</td>
</tr>
<tr>
<td>0A</td>
<td>0110</td>
<td>TIM1</td>
<td>Timer 1</td>
</tr>
<tr>
<td>0B</td>
<td>0111</td>
<td>TIM2</td>
<td>Timer 2</td>
</tr>
<tr>
<td>0C</td>
<td>1000</td>
<td>UART0</td>
<td>UART 0</td>
</tr>
<tr>
<td>0D</td>
<td>1001</td>
<td>UART1</td>
<td>UART 1</td>
</tr>
<tr>
<td>0E</td>
<td>1010</td>
<td>UART2</td>
<td>UART 2</td>
</tr>
<tr>
<td>0F</td>
<td>1011</td>
<td>PL0</td>
<td>PL0 (only private)</td>
</tr>
</tbody>
</table>

### Power Management

- **In sleep mode** the processor allows any interrupt condition to wake up the processor.
- Because it is a static design, sleep mode simply disables the clock to the processor’s modules. When an interrupt occurs, execution is simply resumed from the point where the processor was placed into sleep mode.
- **Stop mode** is achieved by setting a bit in the control register, which disables both the core and the peripherals.
- In stop mode, the processor consumes very little power and is brought out of the stop mode through an external interrupt event.
Local Memory - SRAM

- The MCF5272 provides a local memory with the following parameters:
  - 4K SRAM organized as 1K x 32 bits
  - Sing-cycle access
  - Programmable base address
  - Ideal for use as system stack, or for storing critical code or data

<table>
<thead>
<tr>
<th>Address (in MBAR)</th>
<th>Name</th>
<th>Width</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x400</td>
<td>CCSR</td>
<td>32</td>
<td>Core control register</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x420</td>
<td>ACR</td>
<td>32</td>
<td>Access control register</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x424</td>
<td>ACR</td>
<td>32</td>
<td>Access control register</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x500</td>
<td>RMBAR</td>
<td>32</td>
<td>ROM base address register</td>
<td>Uninitialized unless set</td>
</tr>
<tr>
<td>0x504</td>
<td>RMBAR</td>
<td>32</td>
<td>ROM base address register</td>
<td>Uninitialized unless set</td>
</tr>
</tbody>
</table>

MBAR

- The supervisor-level MBAR register provides a base address for the internal peripherals.
- The MBAR(V) bit is cleared by the system at reset time in order to prevent incorrect references before MBAR is set.
- All internal peripheral registers occupy a single relocatable memory block along 64K boundaries.

![MBAR Register Diagram](image)

Figure 6.2: Memory Base Address Register (MBAR)
The SRAM Base Address Register (RAMBAR) determines the address of the internal SRAM as well as other parameters:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:9</td>
<td>SRAM base address</td>
</tr>
<tr>
<td>10:15</td>
<td>Virtual address space</td>
</tr>
</tbody>
</table>

After reset, the contents of SRAM are undefined and needs to be initialized before use.

Instruction Cache:
- 1 K direct-mapped cache
- Single-cycle access on cache hits
- 16 byte line-fill buffer
- Organized as 64 lines of 16 bytes, and consists of a 64-entry tag array
- CACR (W/IA) forces the entire cache to be marked as invalid
- CPUSHL can invalidate a single cache line
- Reset clears the CACR and disables the instruction cache
### Cache Programming Model

<table>
<thead>
<tr>
<th>Address (using MOVC)</th>
<th>Name</th>
<th>Width</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>CACR</td>
<td>32</td>
<td>Cache control register</td>
<td>D0000</td>
</tr>
<tr>
<td>126</td>
<td>ADR0</td>
<td>32</td>
<td>Access control register 0</td>
<td>D0000</td>
</tr>
<tr>
<td>127</td>
<td>ADR1</td>
<td>32</td>
<td>Access control register 1</td>
<td>D0000</td>
</tr>
</tbody>
</table>

**Register:**
- **Name:** Cache control register
- **Width:** 32 bits
- **Description:** Cache control register
- **Reset Value:** D0000

---

### Cache Programming Model

#### Register 1:
- **Name:** Cache control register
- **Width:** 32 bits
- **Description:** Cache control register
- **Reset Value:** D0000

#### Register 2:
- **Name:** Access control register 0
- **Width:** 32 bits
- **Description:** Access control register 0
- **Reset Value:** D0000

#### Register 3:
- **Name:** Access control register 1
- **Width:** 32 bits
- **Description:** Access control register 1
- **Reset Value:** D0000

---

### Cache Programming Model

#### Register 4:
- **Name:** Cache control register
- **Width:** 32 bits
- **Description:** Cache control register
- **Reset Value:** D0000

---

### Cache Programming Model

#### Register 5:
- **Name:** Cache control register
- **Width:** 32 bits
- **Description:** Cache control register
- **Reset Value:** D0000

---

### Figure 4.1: Cache Control Register (CACR)

- **Description:** Cache control register
- **Width:** 32 bits
- **Reset Value:** D0000

---

### Figure 4.2: Access Control Registers (ADRs)

- **Name:** Access control register
- **Width:** 32 bits
- **Description:** Access control register
- **Reset Value:** D0000

---

### Figure 4.3: Cache Control Register (CACR)
System Integration Module

- The System Integration Module provides overall control and arbitration of the bus and serves as the interface between the ColdFire core and the internal peripheral devices.
- It provides the following features:
  - Module base address register (MBAR)
  - Interrupt Controller
  - Chip Select module and SDRAM controller interface
  - System protection (watchdog timer)
  - Pin Assignment Register (PAR)
  - Power Management
  - Bus Arbitration

Chip Select Module

- The ColdFire provides:
  - 8 dedicated chip selects
  - Address masking for memories ranging from 4K to 2 GB
  - Programmable wait states and port sizes
  - Programmable address setup and hold times
  - SDRAM controller interface with CS7#
  - Global chip select functionality
  - CS0# used to access external boot ROM, in conjunction with BUSW[1:0]
### Chip Select Module

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Chip ID</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>CS Base Register 0</td>
<td>0x0000</td>
<td>0x0000_0000</td>
</tr>
<tr>
<td>0x0004</td>
<td>CS Base Register 1</td>
<td>0x0004</td>
<td>0x0004_0000</td>
</tr>
<tr>
<td>0x0008</td>
<td>CS Base Register 2</td>
<td>0x0008</td>
<td>0x0008_0000</td>
</tr>
<tr>
<td>0x000C</td>
<td>CS Base Register 3</td>
<td>0x000C</td>
<td>0x000C_0000</td>
</tr>
</tbody>
</table>

### CS Base Address Register

- **Description:**
  - The CS Base Address Register is used to set the base address for the chip select module.
  - It allows for fine-grained control over the memory space allocated to the chip select module.
  - The register is typically used in systems where multiple chips share the same address space.

### CS Base Address Register

- **Field:**
  - **B1:** Chip Select
  - **B0:** Chip Enable
  - **A[7:0]:** Address

- **Description:**
  - The CS Base Address Register is used to set the base address for the chip select module.
  - It allows for fine-grained control over the memory space allocated to the chip select module.
  - The register is typically used in systems where multiple chips share the same address space.

- **Note:**
  - The CS Base Address Register is used to set the base address for the chip select module.
  - It allows for fine-grained control over the memory space allocated to the chip select module.
  - The register is typically used in systems where multiple chips share the same address space.

### CS Base Address Register

- **Field:**
  - **A[7:0]:** Address

- **Description:**
  - The CS Base Address Register is used to set the base address for the chip select module.
  - It allows for fine-grained control over the memory space allocated to the chip select module.
  - The register is typically used in systems where multiple chips share the same address space.
CS Option Register

<table>
<thead>
<tr>
<th>CS Option Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Option Register</td>
</tr>
<tr>
<td>CS Option Register</td>
</tr>
<tr>
<td>CS Option Register</td>
</tr>
</tbody>
</table>

CS Option Register

<table>
<thead>
<tr>
<th>CS Option Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Option Register</td>
</tr>
<tr>
<td>CS Option Register</td>
</tr>
<tr>
<td>CS Option Register</td>
</tr>
</tbody>
</table>

End of Lecture 1
UART Module

- The 5272 contains two independent UARTs
- Each UART can be clocked by the CPU, or by an external source
- They each provide extensive interrupt control and baud rate selections, as well as full-duplex receive/transmitter functionality

UART

- Features of the UART are:
  - 24 byte FIFO receiver with FIFO timeout
  - 24 byte FIFO transmitter
  - Independently programmable receiver and transmitter clock sources
  - Programmable data format, from 5-8 data bits, with various parity features and different numbers of stop bits
  - Each channel can be programmed for automatic echo, local loop-back, or remote loop-back mode
  - Automatic wake-up mode for multi-drop applications
- UART registers are only accessible as bytes

UART Programming Model
UART Mode Register 1

- The UART Mode Register 1 (UMR1n) sets the configuration of the UART register such as parity mode and parity type.
- It can be read or written when the mode register points to it.
- After UMR1n has been read or written, the pointer then points to UMR2n.

UART Mode Register 2

- The UMR2n further configures the UART by setting configuration such as the channel mode (normal, echo, or loop-back), stop bit length control.
- It can be read or written when the mode register points to it, which is the case after a read/write of UMR1n.

UART Command Register

- The UART Command Registers UCRn supply commands to the UART module.
- Only multiple commands that do not conflict can be specified at one time.
- For example, RESET UART and ENABLE UART cannot be specified at the same time.
The UART Receive Buffer contains one serial shifter and a 24-byte FIFO. The CPU reads from the top of the stack while the RxDO shifts and updates from the bottom. RB contains the character in the receiver.
UART Transmitter Buffer

- The Transmitter also consists of a 24 byte buffer and a transmit shift register
- The transmitter automatically inhibits writing of any more characters while the FIFO is full

UART – Multidrop

- The MCF5272 UART has an advanced feature enabling it to operate in a wake-up mode for multi-drop or multi-processor applications
- In this mode a master can transmit an address character followed by a block of data characters targeted for one of up to 256 slaves
- The slaves have their receivers disabled, but they still continue to monitor the master’s data stream
The MCF5272 provides up to 48 General Purpose IO ports. The GPIO ports are multiplexed with other module signals and must be selected appropriately. To avoid indeterminate read values and reduce power consumption, internal pull-up resistors are active immediately after a reset.

### GPIO registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Access</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0</td>
<td>Port A Control Register (PAC0)</td>
<td>Read/Write</td>
<td>8-bit</td>
</tr>
<tr>
<td>PA1</td>
<td>Port A I/O Direction Register (PA0DIR)</td>
<td>Read/Write</td>
<td>8-bit</td>
</tr>
<tr>
<td>PA2</td>
<td>Port A I/O Data Register (PA0DAT)</td>
<td>Read/Write</td>
<td>8-bit</td>
</tr>
<tr>
<td>PA3</td>
<td>Port A I/O Data Register (PA0DAT)</td>
<td>Read/Write</td>
<td>8-bit</td>
</tr>
</tbody>
</table>

**Figure 17-1: Port A Control Register (PA0C0)**
Queued SPI

- Programmable queue to support up to 16 transfers without user intervention
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chipselect lines to control up to 15 devices
- Baud rates from 129.4 Kbps to 33 Mbps at 66 MHz
- Programmable delays before and after transfers
The QSPI uses a dedicated 80 byte SRAM to provide for:

- 16 command control bytes
- 16 data write bytes
- 16 data read bytes

Each byte corresponds to one queue entry from 0x00 to 0x0F.

Clocking and Data Transfer Example

SDRAM Controller

- The MCF5272 SDRAM controller provides a glueless interface to a variety of SDRAM devices
- Data bus width sizes of 16 or 32
- 16- to 256-Mbit device support
The 5272 provides a one-channel DMA controller which supports memory-to-memory transfers that can be used for block-data moves. The DMA Mode Register controls various DMA operations, in particular that of addressing modes.

**Figure 10-1. DMA Mode Register (DMR)**
DMA Address Registers

- The DMA Source Address Register (DSAR) provides a 32-bit address which the DMA controller drives onto the internal bus for all the channel's read accesses.
- In the same fashion, the DMA Destination Address Register (DDAR) provides the address which the DMA controller provides to the bus for all write operations.
- The DMA Byte Count Register (DBCR) provides the number of bytes which need to be transferred.

Ethernet Module

- Full compliance with IEEE 802.3 standard.
- Supports three different physical interfaces:
  - 100 Mbps Media Independent Interface (MII)
  - 10 Mbps MII
  - 10 Mbps seven-wire interface
- 448 byte on-chip transmit and receive FIFO, which works in conjunction with the DMA engine.

Fast Ethernet Controller

- The Fast Ethernet Controller is designed to work with little host intervention.
- As soon as the driver enables the FEC transmitter, the FEC fetches the first FEC transmit buffer descriptor (TxBD), and if there is a frame ready to be transferred, the DMA transmit of the data buffer starts immediately.

Figure 11-3. Ethernet Frame Format

NOTE: Short To frames are potted automatically by the HCS48F2
When the transmit FIFO contains 512 bits of data, the FEC asserts the E_TxEN and starts transmitting the preamble sequence, the start-of-frame delimiter, and then the frame data.

- If the line is busy, the controller defers transmission (carrier sense is active)
- If the carrier sense goes inactive, the controller waits to verify that it stays inactive for 60 bit times, and if so, the transmission begins after waiting an additional 36 bit times
- If a collision occurs, the FEC follows the specified back-off procedures and retransmits until the retry limit is reached
- The FEC stores the first 64 bytes of the transmit frame in internal RAM so it doesn’t need to be retrieved
USB

- USB uses four wires, two for data and two for power, and operates at a full data rate of 12Mbps
- The MCF5272 USB includes the following features
  - Full compliance with USB 1.1
  - Supports either internal or external USB transceiver
  - Programmable 512 byte receive and transmit FIFO buffers
  - USB device controller with protocol control and administration for up to eight endpoints, 16 interfaces, and 16 configurations
  - Independent interrupts for each endpoint
  - Remote wakeup

Timer Module

- The Timer module has four identical general-purpose 16-bit timers and a software watchdog timer
- Maximum period of 4 seconds at 66 MHz
- 15-ns resolution
- Programmable clock sources
- Free run and restart modes

Timer

- A timer can be configured to count until a reference is reached, at which point it can either start a new time count immediately, or continue to run.
- Upon reaching a reference value, the timer can issue an interrupt.
- Also, timers 0 and 2 are capable of outputting a signal on the timer outputs TOUT0 and TOUT1, which can be either an active-low pulse or a toggle of the current output.
- Timers are also capable of latching counter values when the corresponding input capture edge detector detects a predefined transition, and this too can generate an interrupt.
Mechanical Data

Electrical Characteristics

Portions of this power point presentation may have been taken from relevant users and technical manuals. Original content Copyright © 2005 - Babak Kia.