Course Administration

- **Instructor:** Aleksandar Milenkovic
  - milenka@ece.uah.edu
  - www.ece.uah.edu/~milenka
  - EB 217-L
  - Mon. 5:30 PM – 6:30 PM,
  - Wen. 12:30 – 13:30 PM
- **URL:** [http://www.ece.uah.edu/~milenka/cpe421-05F](http://www.ece.uah.edu/~milenka/cpe421-05F)
- **TA:** Joel Wilder
- **Labs:** Lab #5 is on. Hw2 due is 11/02/05.
- **Test II:** 11/09/05 (MSP430 & related).
- **Text:** Microprocessor Systems Design:
  - 68000 Hardware, Software, and Interfacing
- **Review:** M68K (Chapter 1; Chapter 2; Chapter 3),
  - MSP430 (Introduction, Arch., Basic Clock System, WDT, Low Power Modes, Digital I/O), Timers
- **Today:** MSP 430 UART; M68000 Hw (Chapter 4)
Serial Communication

Serial I/O Interface

Functional Units

- Translates data between the internal computer form and the form in which it is transmitted over the data link.
- Translates the TTL-level signals processed by the ACIA into a form suitable for the transmission path.
Asynchronous Serial Interface

- Asynchronous
  - Transmitted and received data are not synchronized over any extended period
  - No synchronization between receiver and transmitter clocks
- Serial
  - Usually character oriented
  - Data stream divided into individual bits at the transmitter side
  - Individual bits are grouped into characters at the receiving side
- Information is usually transmitted as ASCII-encoded characters
  - 7 or 8 bits of information plus control bits

Asynchronous Serial Interface, cont’d

- MARK level (or OFF, or 1-state, or 1-level)
  - This is also the idle state (before the transfer begins)
- SPACE level (or ON, or 0-state, or 0-level)
- One character:
  - Start bit: space level
  - Data bits
  - Optional parity bit
  - Optional stop bit
Asynchronous Serial Interface, cont’d

- 12 possible basic formats:
  - 7 or 8 bits of data
  - Odd, even, or no parity
  - 1 or 2 stop bits
  - Others exist also: no stop bits, 4/5/6 data bits, 1.5 stop bits, etc.

Example: Letter M = ASCII 74D = 10011012 (even parity)

Receiver Clock Timing

- For N=9 bits (7 data + parity + stop) maximum tolerable error is 5% (assume that the receiver clock is slow -- [T + δt] instead of T)
  \[ T/2 > (2N+1)\delta t/2 \]
  \[ \delta t/2 < 1/(2N+1) \]
  \[ \delta t/T < 100/(2N+1) \] as a percentage
RS-232 Interface Standard

- Bi-polar:
  - +3 to +12V (ON, 0-state, or SPACE condition)
  - -3 to –12V (OFF, 1-state, or MARK condition)
- Modern computers accept 0V as MARK
- “Dead area” between –3V and 3V is designed to absorb line noise
- Originally developed as a standard for communication between computer equipment and modems
- From the point of view of this standard:
  - MODEM: data communications equipment (DCE)
  - Computer equipment: data terminal equipment (DTE)
- Therefore, RS-232C was intended for DTE-DCE links (not for DTE-DTE links, as it is frequently used now)

RS-232 Interface Standard

- Each manufacturer may choose to implement only a subset of functions defined by this standard
- Two widely used connectors: DB-9 and DB-25
- Three types of link
  - Simplex
  - Half-duplex
  - Full-duplex
- Basic control signals
  - RTS (Request to send): DTE indicates to the DCE that it wants to send data
  - CTS (Clear to send): DCE indicates that it is ready to receive data
  - DSR (Data set ready): indication from the DCE (i.e., the modem) that it is on
  - DTR (Data terminal ready): indication from the DTE that it is on
RS-232 Interface Standard, another example

- DTR (Data terminal ready): indication from the DTE that it is on.

RS-232 Interface Standard

- DB-25 connector is described in the book; let’s take a look at DB-9.
RS-232 Interface Standard
Example: 9 to 25 pin cable layout for asynchronous data

<table>
<thead>
<tr>
<th>Description</th>
<th>Signal</th>
<th>9-pin DTE</th>
<th>25-pin DCE</th>
<th>Source DTE or DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Detect</td>
<td>CD</td>
<td>1</td>
<td>8</td>
<td>from Modem</td>
</tr>
<tr>
<td>Receive Data</td>
<td>RD</td>
<td>2</td>
<td>3</td>
<td>from Modem</td>
</tr>
<tr>
<td>Transmit Data</td>
<td>TD</td>
<td>3</td>
<td>2</td>
<td>from Terminal/Computer</td>
</tr>
<tr>
<td>Data Terminal Ready</td>
<td>DTR</td>
<td>4</td>
<td>20</td>
<td>from Terminal/Computer</td>
</tr>
<tr>
<td>Signal Ground</td>
<td>SG</td>
<td>5</td>
<td>7</td>
<td>from Modem</td>
</tr>
<tr>
<td>Data Set Ready</td>
<td>DSR</td>
<td>6</td>
<td>6</td>
<td>from Modem</td>
</tr>
<tr>
<td>Request to Send</td>
<td>RTS</td>
<td>7</td>
<td>4</td>
<td>from Terminal/Computer</td>
</tr>
<tr>
<td>Clear to Send</td>
<td>CTS</td>
<td>8</td>
<td>5</td>
<td>from Modem</td>
</tr>
<tr>
<td>Ring Indicator</td>
<td>RI</td>
<td>9</td>
<td>22</td>
<td>from Modem</td>
</tr>
</tbody>
</table>

The Minimal RS-232 Function

- DTE to DCE in simplex mode
- DTE to DTE in simplex mode
The Minimal RS-232 Function

DTE to DCE in full-duplex mode

DTE to DCE in full-duplex mode

DTE to DCE with remote control

DTE to DTE with remote control

CPE/EE 421/521 Microcomputers
Handshaking Between RTS and CTS

Null Modem

Null-modem simulates a DTE-DCE-DCE-DTE circuit
USART Peripheral Interface

- Universal Synchronous/Asynchronous Receive/Transmit (USART) peripheral interface supports two modes
  - Asynchronous UART mode (User manual, Ch. 13)
  - Synchronous Peripheral Interface, SPI mode (User manual, Ch. 14)

- UART mode:
  - Transmit/receive characters at a bit rate asynchronous to another device
  - Connects to an external system via two external pins URXD and UTXD (P3.4, P3.5)
  - Timing is based on selected baud rate (both transmit and receive use the same baud rate)

 UART Features

- 7- or 8-bit data width; odd, even, or non-parity
- Independent transmit and receive shift reg.
- Separate transmit and receive buffer registers
- LSB-first data transmit and receive
- Built-in idle-line and address-bit communication protocols for multiprocessor systems
- Receiver start-edge detection for auto-wake up from LPMx modes
- Programmable baud rate with modulation for fractional baud rate support
- Status flags for error detection
- Independent interrupt capability for transmit and receive
Initialization Sequence & Character Format

- Initialization Sequence
  - Set SWRST bit
  - Initialize all USART registers with SWRST = 1
  - Enable USART module via the MEx SFRs (URXEx and/or UTXEx)
  - Clear SWRST via software (releases the USART for operation)
  - Optional: enable interrupts via IEx SFRs

- Character format

---

Alex Milenkovich
C Examples, UART 2400

```
//****************************************************************************
// MSP-FET430P140 Demo - USART UART 2400 Ultra-low Power Echo ISR, 32kHz ACLK
// Description: Echo a received character, RX ISR used. In the Mainloop USART1
// is made ready to receive one character with interrupt active. The Mainloop
// waits in LPM3. The USART ISR forces the Mainloop to exit LPM3 after
// receiving one character which echo's back the received character.
// ACLK = UCLK1 = LFXT1 = 32768, MCLK = SMCLK = DCO~ 800kHz
// Baud rate divider with 32768Hz XTAL @2400 = 32768Hz/2400 = 13.65 (000Dh)
// An external watch crystal is required on XIN XOUT for ACLK
//****************************************************************************

#include <msp430x14x.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT

    P3SEL |= 0xC0; // P3.6,7 = USART1 option select

    ME2 |= UTXE1 + URXE1; // Enable USART1 TXD/RXD

    UCTL1 |= CHAR; // 8-bit character
    UCTL1 |= U2STOP; // U2-stop character

    UBR01 = 0x0D; // 32kHz/2400 - 13.65
    UBR11 = 0x00;

    UMCTL1 = 0x6B; // Modulation

    UCTL1 &= ~SWRST; // Initialize USART state machine

    IE2 |= URXIE1; // Enable USART1 RX interrupt

    // Mainloop
    for (;;)
    {
        _BIS_SR(LPM3_bits + GIE); // Enter LPM3 w/interrupt

        while (!(IFG2 & UTXIFG1)); // USART1 TX buffer ready?

        TXBUF1 = RXBUF1; // TXBUF1 = RXBUF1

        _BIC_SR_IRQ(LPM3_bits); // Clear LPM3 bits from 0(SR)
    }
}

UART1 RX ISR will for exit from LPM3 in Mainloop
interrupt(USART1_VECTOR) void usart1_rx (void)
{
    // RXBUF1 gets a new character
}
```

---

CPE/EE 421
Microcomputers

THE 68000 CPU HARDWARE MODEL

Instructor: Dr Aleksandar Milenkovic
Lecture Notes

Lecture 19
THE 68000 CPU HARDWARE MODEL
Chapter 4

- 68000 interface
- Timing diagram
- Minimal configuration using the 68000

68000 Interface

- M68000: 64 pins, arranged in 9 groups:
  - Address Bus: \( A_{01} - A_{23} \)
  - Data Bus: \( D_{00} - D_{15} \)
  - Asynchronous bus control: \( AS^*, R/W^*, UDS^*, LDS^*, DTACK^*, BERR^* \)
  - Synchronous bus control: \( E, VPA^*, VMA^* \)
  - Bus arbitration control: \( BR^*, BG^*, BGACK^* \)
  - Function code: \( FC0, FC1, FC2 \)
  - System control: \( CLK, RESET^*, HALT^* \)
  - Interrupt control: \( IPL0^*, IPL1^*, IPL2^* \)
  - Miscellaneous: \( Vcc(2), Gnd(2) \)

- Legend:
  - XX: Input
  - XX: Output
  - XX: Input/Output
68000 Interface, cont’d

- Classification of pins based on function
  - SYSTEM SUPPORT PINS
    - Essential in every 68000 system (power supply, clock, ...)
  - MEMORY AND PERIPHERAL INTERFACE PINS
    - Connect the processor to an external memory subsystem
  - SPECIAL-PURPOSE PINS
    (not needed in a minimal application of the processor)
    - Provide functions beyond basic system functions

- Terminology
  - Asterisk following a name: indicates the signal is active low
  - “Signal is asserted” means signal is placed in its active state
  - “Signal is negated” means signal is placed in its inactive state

System Support Pins

- Power Supply
  - Single +5V power supply: 2 Vcc pins and 2 ground pins

- Clock
  - Single-phase, TTL-compatible signal
  - Bus cycle: memory access, consists of a minimum 4 clock cycles
  - Instruction: consists of one or more bus cycles

- RESET*
  - Forces the 68000 into a known state on the initial application of power:
    - supervisor’s A7 is loaded from memory location $00 0000
    - Program counter is loaded from address $00 0004
  - During power-up sequence must be asserted together with the HALT* input for at least 100 ms.
  - Acts also as an output, when processor executes the instruction RESET (used to reset peripherals w/out resetting the 68000)
System Support Pins, cont’d

- **HALT***
  - In simple 68000 systems can be connected together with **RESET***
  - Can be used:
    - by external devices to make the 68000 stop execution after current bus cycle (and to negate all control signals)
    - to single-step (bus cycle by bus cycle) through program
    - to rerun a failed bus cycle (if memory fails to respond correctly) in conjunction with the bus error pin, **BERR***
  - It can be used as an output, to indicate that the 68000 found itself in situation from which it cannot recover (HALT* is asserted)

Memory and Peripheral Interface Pins

- **Address Bus**
  - 23-bit address bus, permits $2^{23}$ 16-bit words to be addressed
  - Tri-state output pins (to permit devices other then the CPU to take a control over it)
  - Auxiliary function:
    - supports vectored interrupts
    - Address lines $A_{01}, A_{02}, A_{03}$ indicate the level of the interrupt being serviced
    - All other address lines are set to a high level

- **Data Bus**
  - Bi-directional 16-bit wide data bus
    - During a CPU read cycle acts as an input
    - During a CPU write cycle acts as an output
  - Byte operations: only $D_{00}$-$D_{07}$ or $D_{08}$-$D_{15}$ are active
  - Interrupting device identifies itself to the CPU by placing an interrupt vector number on $D_{00}$-$D_{07}$ during an interrupt acknowledge cycle
Memory and Peripheral Interface Pins, cont’d

- **AS***
  - When asserted, indicates that the content of the address bus is valid.

- **R/W***
  - Determines the type of a memory access cycle
    - CPU is reading from memory: R/W* = 1
    - CPU is writing to memory: R/W* = 0
    - If CPU is performing internal operation, R/W* is always 1
    - When CPU relinquishes control of its busses, R/W* is undefined

- **UDS*** and **LDS***
  - Used to determine the size of the data being accessed
  - If both UDS* and LDS* are asserted, word is accessed
  - **R/W*** **UDS*** **LDS***
    - 010: write lower byte (D00 – D07: data valid, replicated on D8–D15)
    - 011: write word (D00 – D15: data valid)
    - 101: read upper byte (D00 – D07: invalid, D8–D15 – data valid)

---

Memory and Peripheral Interface Pins, cont’d

- **DTACK*** (Data Transfer Acknowledge)
  - Handshake signal generated by the device being accessed
  - Indicates that the contents of the data bus is valid
  - **If DTACK*** is not asserted, CPU generates wait-states until DTACK goes low or until an error state is declared.
  - **When DTACK*** is asserted, CPU completes the current access and begins the next cycle
  - DTACK* has to be generated a certain time after the beginning of a valid memory access (timer supplied by the system designer).
Memory and Peripheral Interface Pins, cont’d

Special-Function Pins of the 68000

- **BERR** *(Bus Error Control)*
  - Enables the 68000 to recover from errors within the memory system

- **BR*, BG*, BGACK** *(Bus Arbitration Control)*
  - Used to implement multiprocessor systems based on M68000

- **FC0-FC2** *(Function Code Output)*
  - Indicate the type of cycle currently being executed
  - Becomes valid approximately half a clock cycle earlier than the contents of the address bus

- **IPL0*-IPL2** *(Interrupt Control Interface)*
  - Used by an external device to indicate that it requires service
  - 3-bit code specifies one of eight levels of interrupt request
### Special-Function Pins of the 68000, cont’d

<table>
<thead>
<tr>
<th>Function Code Output</th>
<th>Processor Cycle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC2     FC1     FC0</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>0 0 1</td>
<td>User data</td>
</tr>
<tr>
<td>0 1 0</td>
<td>User program</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Undefined, reserved</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Supervisor data</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Supervisor program</td>
</tr>
<tr>
<td>1 1 1</td>
<td>CPU space (interrupt acknowledge)</td>
</tr>
</tbody>
</table>

---

Using the 68000’s function code outputs

![Diagram of 68000 Function Code Outputs](image)

**Figure 4.8**
The 68000 is not fully asynchronous because its actions are synchronized with a clock input:
- It can prolong a memory access until an ACK is received, but it has to be in increments of one clock cycle.

Interpreting the Timing Diagram
Timing Diagram of a Simple Flip-Flop
Idealized form of the timing diagram

Actual behavior of a D flip-flop

General form of the timing diagram

An alternative form of the timing diagram
The Clock

- A microprocessor requires a clock that provides a stream of timing pulses to control its internal operations.
- A 68000 memory access takes a minimum of eight **clock states** numbered from clock state S0 to clock state S7.

A memory access begins in clock state S0 and ends in state S7.
The most important parameter of the clock is the duration of a cycle, $t_{\text{CYC}}$.

At the start of a memory access, the CPU sends the address of the location it wishes to read to the memory.
Address Timing

- We are interested in **when** the 68000 generates a new address for use in the current memory access.
- The next slide shows the relationship between the new address and the state of the 68000’s clock.

Initially, in state S0 the address bus contains the old address.

In state S1 a new address becomes valid for the remainder of the memory access.
The time at which the contents of the address bus change can be related to the edges of the clock.

Address Timing

- Let’s look at the sequence of events that govern the timing of the address bus
- The “old” address is removed in state S0
- The address bus is floated for a short time, and the CPU puts out a new address in state S1
The old address is removed in clock state S0 and the address bus floated.

The designer is interested in the point at which the address first becomes valid. This point is $t_{\text{CLAV}}$ seconds after the falling edge of S0.
The memory needs to know when the address from the CPU is valid. An address strobe, AS*, is asserted to indicate that the address is valid.

**Address and Address Strobe**

- We are interested in the relationship between the time at which the address is valid and the time at which the address strobe, AS*, is asserted.
- When AS* is active-low it indicates that the address is valid.
- We now look at the timing of the clock, the address, and the address strobe.
AS* goes active low after the address has become valid.
AS* goes inactive high before the address changes.

AS* goes low in clock state S2.
The Data Strobes

- The 68000 has two data strobes LDS* and UDS*. These select the lower byte or the upper byte of a word during a memory access.
- To keep things simple, we will use a single data strobe, DS*.
- The timing of DS* in a read cycle is the same as the address strobe, AS*.

The data strobe, is asserted at the same time as AS* in a read cycle.
The Data Bus

- During a read cycle the memory provides the CPU with data.
- The next slide shows the data bus and the timing of the data signal.
- Note that valid data does not appear on the data bus until near the end of the read cycle.
Analyzing the Timing Diagram

- We are going to redraw the timing diagram to remove clutter.
- We aren’t interested in the signal paths themselves, only in the relationship between the signals.

We are interested in the relationship between the clock, AS*/DS*, and the data in a read cycle.
The earliest time at which the memory can begin to access data is measured from the point at which the address is first valid.
Calculating the Access Time

- We need to calculate the memory’s access time
- By knowing the access time, we can use the appropriate memory component
- Equally, if we select a given memory component, we can calculate whether its access time is adequate for a particular system

Data from the memory is latched into the 68000 by the falling edge of the clock in state S6.
Data must be valid \( t_{\text{DICL}} \) seconds before the falling edge of S6.

We know that the time between the address valid and data valid is \( t_{\text{acc}} \).
The address becomes valid \( t_{CLAV} \) seconds after the falling edge of S0.

From the falling edge of S0 to the falling edge of S6:
- the address becomes valid
- the data is accessed
- the data is captured
The falling edge of S0 to the falling edge of S6 is three clock cycles

$3 \times t_{\text{cyc}} = t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}}$
Timing Example

- 68000 clock 8 MHz \( t_{\text{CYC}} = 125 \text{ ns} \)
- 68000 CPU \( t_{\text{CLAV}} = 70 \text{ ns} \)
- 68000 CPU \( t_{\text{DICL}} = 15 \text{ ns} \)
- What is the minimum \( t_{\text{acc}} \)?
  - \( 3 \cdot t_{\text{CYC}} = t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}} \)
  - \( 375 = 70 + t_{\text{acc}} + 15 \)
  - \( t_{\text{acc}} = 290 \text{ ns} \)

A 68000 Read Cycle
Extended Read Cycle

One bus cycle with four wait states

DTACK* did not go low at least 20ns before the falling edge of state S4

▲ Designer has to provide logic to control DTACK*

Memory Timing Diagram

▲ The 6116 static memory component
  ▲ 2K x 8bit memory – byte-oriented!
  ▲ Two 6116’s configured in parallel to allow word accesses
  ▲ Eleven address inputs
Assumptions:
- R/W* is high for the duration of the read cycle
- OE* is low
Figure 4.20 Connecting The 6116 RAM to a 68000 CPU Timing Diagram

Timing Example

- 68000 clock 8 MHz
  - 68000 CPU
  - What is the minimum $t_{\text{acc}}$?
    - $3 \times t_{\text{CYC}} > t_{\text{CLAV}} + t_{\text{acc}} + t_{\text{DICL}}$
    - $375 > 70 + t_{\text{acc}} + 15$
    - $t_{\text{acc}} < 290 \text{ ns}$ (or $t_{\text{AA}}$ from the timing diagram, access time)

- For the 12.5MHz version of 68000
  - 68000 CPU
  - $3 \times 80 > 55 + t_{\text{acc}} + 10$
  - $t_{\text{acc}} < 175 \text{ ns}$

- Remember, maximum $t_{\text{AA}}$ for the 6116 RAM was 200 ns
68000 Write Cycle

- 68000 transmits a byte or a word to memory or a peripheral
- Essential differences:
  - The CPU provides data at the beginning of a write cycle
  - One of the bus slaves (see later) reads the data
- In a read cycle DS* and AS* were asserted concurrently
  This will be not a case here!
- Reason for that: 68000 asserts DS* only when the contents of data bus have stabilized
  - Therefore, memory can use UDS*/LDS* to latch data from the CPU

Simplified write cycle timing diagram

In a write cycle: UDS*/LDS* is asserted one cycle after AS*
Follow this sequence of events in a write cycle:
- Address stable
- AS* asserted
- R/W* brought low
- Data valid
- DS* asserted

Figure 4.23

Write Cycle Timing Diagram of a 6116 RAM

- Write recovery time (min 10ns)
- Address valid to end of write (min 120ns)
- Write pulse width (min 90ns)
- Address setup time (min 20ns)

Figure 4.24
Write Cycle Timing Diagram of a 6116 RAM, cont’d

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write cycle time</td>
<td>tWC</td>
<td>90</td>
<td>150</td>
</tr>
<tr>
<td>Chip select low to end of write</td>
<td>tCWL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write recovery time</td>
<td>tWR</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Address valid to end of write</td>
<td>tAW</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Address setup time</td>
<td>tAS</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Write pulse width</td>
<td>tWP</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>Data setup time</td>
<td>tDS</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Data hold time</td>
<td>tDH</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

- Write cycle ends with either CS* or WE* being negated (CS* and WE* internally combined)
- An address must be valid for at least $t_{AS}$ nanoseconds before WE* is asserted
- Must remain valid for at least $t_{WR}$ nanoseconds after WE* is negated
- Data from the CPU must be valid for at least $t_{DW}$ nanoseconds before WE* is negated
- Must remain valid for at least $t_{DH}$ nanoseconds after the end of the cycle