CPE/EE 421
Microcomputers
Instructor: Dr Aleksandar Milenkovic
Lecture Note
S18

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
- TA: Joel Wilder
- Labs: Lab #4 is on. Hw #2 is posted.
- Test I: Graded. Solutions are in scr/.
- 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)
- Today: Recap Timers, USART
Timer_A MSP430x1xx

- **Purpose**
  - The Timer A and B systems on the MSP are a versatile means to measure time intervals. The timers can measure the timing on incoming signals or control the timing on outgoing signals. This function is necessary to meet arbitrary timing requirements from outside components, and the ability is useful in phase locking scenarios.

- **Features**
  - 16-bit counter with 4 operating modes
  - Selectable and configurable clock sources (internal - ACLK, SMCLK; external – INCLK, TBCLK)
  - Three (or five) independently configurable capture/compare registers with configurable inputs
  - Three (or five) individually configurable output modules with 8 output modes
  - Multiple, simultaneous, timings; multiple capture/compares; multiple output waveforms such as PWM signals; and any combination of these.
  - Interrupt capabilities
    - each capture/compare block individually configurable
Capture and Compare Registers

- **What is a capture?**
  - A record of the timer count when a specific event occurs. The capture modules of the timers are tied to external pins of the MSP. When the control registers of timer A and the specific capture compare module have been properly configured, then the capture will record the count in the timer when the pin in question makes a specific transition (either from low to high or any transition). This capturing event can be used to trigger an interrupt so that the data can be processed before the next event. In combination with the rollover interrupt on Capture module 0, you can measure intervals longer than 1 cycle.

- **Compare**
  - The inverse of a capture. While capture mode is used to measure the time of an incoming pulse width modulation signal (a signal whose information is encoded by the time variation between signal edges), compare mode is used to generate a pulse width modulation (PWM) signal. When the timer reaches the value in a compare register, the module will give an interrupt and change the state of an output according to the other mode bits. By updating the compare register numbers, you change the timing of the signal level transitions.
**Timer_A Counting Modes**

**Stop/Halt Mode**
Timer is halted with the next +CLK

**UP Mode**
Timer counts between 0 and CCR0

**UP/DOWN Mode**
Timer counts between 0 and CCR0 and 0

**Continuous Mode**
Timer continuously counts up

---

**Timer_A 16-bit Counter**

- **TACLK**
- **ACLK**
- **SMCLK**
- **INCLK**
- **MC0**
- **MC1**
- **ID0**
- **ID1**
- **SSEL0**
- **SSEL1**

**Input Select**

**Input Divider**

**Mode Control**

**CLR**

**TAIE**

**TAIFG**

---

Page 11-12, User’s Manual
Timer_A Capture Compare Blocks

Timer_A Output Units

Operational Conditions

<table>
<thead>
<tr>
<th>OMx2</th>
<th>OMx1</th>
<th>OMx0</th>
<th>Function</th>
<th>Operational Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Output Mode</td>
<td>Oux signal is set according to Oux bit</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Set</td>
<td>EQUx sets Oux signal clock synchronous with timer clock</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>PWM Toggle/Reset</td>
<td>EQUx toggles Oux signal, reset with EQU0, clock sync. with timer clock</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Toggle</td>
<td>EQUx toggles Oux signal, clock synchronous with timer clock</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
<td>EQUx resets Oux signal clock synchronous with timer clock</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>PWM Toggle/Reset</td>
<td>EQUx toggles Oux signal, set with EQU0, clock synchronous with timer clock</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PWM Set/Reset</td>
<td>EQUx resets Oux signal, set with EQU0, clock synchronous with timer clock</td>
</tr>
</tbody>
</table>
Timer_A Continuous-Mode Example

Example shows three independent HW event captures. CCRx "stamps" time of event - Continuous-Mode is ideal.

Timer_A PWM Up-Mode Example

Example shows three different asymmetric PWM-Timings generated with the Up-Mode.
Timer_A PWM Up/Down Mode Example

Example shows Symmetric PWM Generation - Digital Motor Control

CPE/EE 421/521 Microcomputers

C Examples, CCR0 Contmode ISR, TA_0 ISR

```c
#include <msp430x14x.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT
    P1DIR |= 0x01; // P1.0 output
    CCTL0 = CCIE; // CCR0 interrupt enabled
    CCR0 = 50000;
    TACTL = TASSEL_2 + MC_2; // SMCLK, contmode

    _bis_SR_register (LPM0_bits + GIE); // Enter LPM0 with interrupt

    // void TimerA(void) interrupt [TIMERA0_VECTOR]
    {
        P1OUT ^= 0x01; // Toggle P1.0
        CCR0 += 50000; // Add Offset to CCR0
    }
}
```
C Examples, CCR0 Upmode ISR, TA_0

#include <msp430x14x.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  P1DIR |= 0x01;  // P1.0 output
  CCTL0 = CCIE; // CCR0 interrupt enabled
  CCR0 = 1000-1;
  TACTL = TASSEL_1 + MC_1; // ACLK, upmode

  #include <msp430x14x.h>

  void Timer_A (void)
  {
    P1OUT ^= 0x01; // Toggle P1.0
  }

  #pragma vector=TIMERA0_VECTOR
  #pragma vector=TIMERA1_VECTOR

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

  // Timer A0 interrupt service routine

  void main(void)
  {
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT
    P1DIR |= 0x01; // P1.0 output
    CCTL0 = CCIE; // CCR0 interrupt enabled
    CCR0 = 1000;
    TACTL = TASSEL_2 + MC_2; // SMCLK, Contmode

    #pragma vector=TIMERA1_VECTOR
    __interrupt void Timer_A(void)
    {
      switch( TAIV )
      {
        case  2:  // CCR1
          {
            P1OUT ^= 0x01; // Toggle P1.0
            CCR1 += 50000; // Add Offset to CCR1
          }
          break;
        case  4: // CCR2 not used
          break;
        case 10: break; // overflow not used
      }
    }

    #pragma vector=TIMERA3_VECTOR
    // Timer_A3 Interrupt Vector (TAIV) handler
    _interrupt void Timer_A(void)
    {
      switch( TAIV )
      {
        case  2: // CCRI
          {
            P1OUT ^= 0x01; // Toggle P1.0
            CCRI = 50000; // Add offset to CCRI
          }
          break;
        case  4: // CCR2 not used
          break;
        case 10: break; // overflow not used
      }
    }

    M. Buccini
    Texas Instruments, Inc
    October 2003
    Built with IAR Embedded Workbench Version: 1.24B
    December 2003
    Updated for IAR Embedded Workbench Version: 2.21B

}
C Examples, PWM, TA1-2 upmode

```c
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;  // Stop WDT
    P1DIR |= 0x0C;        // P1.2 and P1.3 output
    P1SEL |= 0x0C;        // P1.2 and P1.3 TA1/2 options
    CCR0 = 512-1;         // PWM Period
    CCTL1 = OUTMOD_7;     // CCR1 reset/set
    CCR1 = 384;           // CCR1 PWM duty cycle
    CCTL2 = OUTMOD_7;     // CCR2 reset/set
    CCR2 = 128;           // CCR2 PWM duty cycle
    TACTL = TASSEL_2 + MC_1;  // SMCLK, up mode
    _BIS_SR(LPM0_bits);         // Enter LPM0
}
```

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 $\text{4D} = 1001101_2$ (even parity)

Least significant bit
Receiver Clock Timing

For $N=9$ bits ($7$ data + parity + stop) maximum tolerable error is 5% (assume that the receiver clock is slow -- $[T + \delta t]$ 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 DCE in full-duplex mode**

**DTE to DTE in simplex mode**

**DTE to DTE in full-duplex mode**
The Minimal RS-232 Function

DTE to DCE with remote control

<table>
<thead>
<tr>
<th>DTE</th>
<th>2</th>
<th>3</th>
<th>7</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCE</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DTE to DTE with remote control

<table>
<thead>
<tr>
<th>DTE</th>
<th>2</th>
<th>3</th>
<th>7</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTE</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Handshaking Between RTS and CTS

<table>
<thead>
<tr>
<th>DTE</th>
<th>2</th>
<th>3</th>
<th>7</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Null Modem

- Null-modem simulates a DTE-DCE-DCE-DTE circuit

![Null Modem Diagram]

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

USART Block Diagram: UART mode
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**

```
// Example C code for UART 2400
#include <msp430x14x.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT
    UBR01 = 0x0D; // 32k/2400 - 13.65
    UBRRH = 0x00;
    UMCTL1 = 0x6B;     // Modulation
    UTXRCTL |= SSEL0; // UCLK = ACLK
    IE2 |= URXIE1;     // Enable USART1 RX interrupt

    // Mainloop
    for (;;)
    {
        // RX ISR
        if (URXIFG1 != 0)
        {
            TXBUF1 = RXBUF1; // Send received character
            UTXIFG1 = 0;     // Clear TX interrupt flag
        }
    }
}
```