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/compare; 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.
- Continuous Mode: Timer continuously counts up.

**UP/DOWN Mode**
- Timer counts between 0 and CCR0 and 0FFFFh.

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

Timer_A Capture Compare Blocks

**Timer_A Continuous-Mode Example**

Timer_A PWM Up-Mode Example

Example shows three different asymmetric PWM-Timings generated with the Up-Mode.
**C Examples, PWM, TA1-2 upmode**

- **Description:** Example shows Symmetric PWM Generation - Digital Motor Control
- **Notes:**
  - MSP-FET430P140 Demo - Timer_a PWM TA1-2 upmode
  - Texas Instruments, Inc
  - M.Buccini

---

**C Examples, CCR0 Contmode ISR, TA_0**

- **Description:** C Examples, CCR0 Contmode ISR, TA_0
- **Notes:** Proper use of TAIV interrupt vector generator demonstrated.
- **Code:**

```c
#include <msp430x14x.h>

#pragma vector=TIMERA0_VECTOR

Timer_A0中断服务程序

void main(void)
{
    _BIS_SR(LPM0_bits); // 进入LPM0模式，开启中断
    CCR0 = 50000; // CCR0 PWM周期
    P1DIR |= 0x01; // P1.0输出
    WDTCTL = WDTPW + WDTHOLD; // 停止WDT
    case  2: // CCR1
        switch( TAIV )
        {
            break;
        }
        CCR1 += 50000; // Add Offset to CCR1
        P1OUT ^= 0x01; // Toggle P1.0
    }

// Timer A0中断服务程序示例

int main(void)
{
    int i;

    for( i = 0; i < 10000; i++ )
    {
        // Timer A0中断服务程序示例
    }
}
```

---

**C Examples, CCR1 Contmode ISR, TA_1**

- **Description:** C Examples, CCR1 Contmode ISR, TA_1
- **Notes:**
  - MSP-FET430P140 Demo - Timer_A Toggle P1.0,
  - default DCO frequency used for TACLK.

---

**Serial Communication**

- **Description:**
  - Serial Communication
  - Texas Instruments, Inc
  - M.Buccini

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **January 2004**
  - September 2003
  - Texas Instruments, Inc
  - M.Buccini

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Built with IAR Embedded Workbench Version: 1.26B

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Texas Instruments, Inc
  - M. Buccini

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Built with IAR Embedded Workbench Version: 1.26B

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Texas Instruments, Inc
  - M. Buccini

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Built with IAR Embedded Workbench Version: 1.26B

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Texas Instruments, Inc
  - M. Buccini

---

**Notes:**

- **Update for IAR Embedded Workbench Version:** 2.21B
- **December 2003**
  - Built with IAR Embedded Workbench Version: 1.26B
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

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)
  \[ \frac{T}{2} > \frac{2N+1}{2} \]
  \[ \frac{\delta t}{T} < 100 \frac{2N+1}{2N} \] 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

- DB-25 connector is described in the book; let’s take a look at DB-9.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Carrier Detection</td>
<td>6</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>2</td>
<td>Receive Data</td>
<td>7</td>
<td>Request to Send</td>
</tr>
<tr>
<td>3</td>
<td>Transmitted Data</td>
<td>8</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>4</td>
<td>Data Terminal Ring Indicator</td>
<td>9</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

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</th>
<th>25-pin</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

- TxD
- RxD
- RTS
- CTS

DCE

- TxD
- RxD
- RTS
- CTS

DTE to DCE with remote control

- TxD
- RxD
- RTS
- CTS

DTE

- TxD
- RxD
- RTS
- CTS

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

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**

C Examples, UART 2400

```c
#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
  UTCTL1 |= SSEL0; // UCLK = ACLK
  UBR01 = 0x0D; // 32k/2400 = 13.65 (000Dh)
  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;            // RXBUF1 to TXBUF1
  }

  // UART1 RX ISR will for exit from LPM3 in Mainloop
  interrupt[UART1RX_VECTOR] void usart1_rx (void)
  {
    _BIC_SR_IRQ(LPM3_bits);// Clear LPM3 bits from 0(SR)
  }
```

(M. Buccini
Texas Instruments, Inc
October 2003
Built with IAR Embedded Workbench Version: 1.26B
January 2004
Updated for IAR Embedded Workbench Version: 2.21B

---

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