![Texas Instruments]

Implementing a UART Function With TimerA3

Mark Buccini   Mixed Signal Products

ABSTRACT

This application report describes how to use timer_A3 to implement a UART function. The included examples are specifically for the MSP430x11x(1) family, but they can be adapted to any MSP430 family member incorporating timer_A. The UART function uses hardware features of the timer_A3, and software. The implementation is half-duplex, event-driven, and it supports an 8N1 protocol at baud rates from 1200 to 115200, and faster.

Introduction

Asynchronous serial communication can be added to an MSP430x11x(1) application by using hardware features of the integrated timer_A3 module. This report provides several UART function examples demonstrated with an RS232 interface between an MSP430F1121 Flash MCU and a PC serial port (see Figure 1). A description of how to use timer_A3 hardware that provides automatic start-bit detection, baud-rate generation, and data-bit latching is detailed. Hardware features of timer_A3 greatly reduce software and CPU overhead typically associated with microcontroller software UART implementations. Hardware features of timer_A3 also allow the UART to operate as a background function simultaneously with other real-time system tasks.
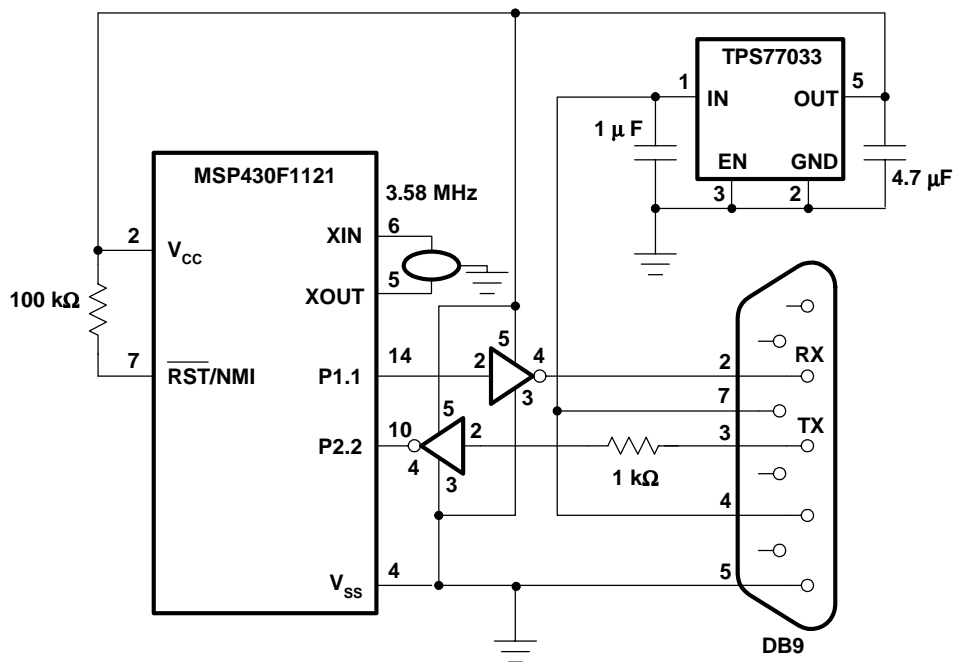

Figure 1.   MSP430x11x(1) UART Demonstration Circuit

MSP430x11x(1) Timer_A UART Description

As shown in Figure 1, an MSP430F1121 communicates serially with another system, in this report a PC using an RS232 interface. Characters are exchanged between the two systems over three lines: receive, transmit, and common ground. The character protocol used is 8N1: 8 data bits, no parity and one stop bit. Users can modify the UART function to support other protocols and baud rates, including parity and a 9-th bit addressing. The UART function described uses a capture-compare register 0 (CCR0), one of the three available with timer_A3. CCR0 is used for start-bit detection, baud-rate generation, and data-bit latching. The other two capture-compare registers are free for other tasks. The selection of CCR0 is arbitrary. Any or all CCRx registers can be used for the UART function. Port pins P1.1 and P2.2 are peripheral-option selections associated with timer_A3 CCR0 (on MSP430x11x(1) derivatives). P1.1 is selected for transmit, P2.2 for receive. Peripheral-options are selected for a pin using the peripheral-option select registers, P1SEL and P2SEL. Because P1.1 is used as an output, this pin must be configured as an output using the port-1 direction register (P1DIR). P2.2 is required to operate as an input. This is the default for an MSP430 port pin. Timer_A3 is configured to run in continuous mode, allowing timer resources to be available for other functions simultaneously with the UART. CPU register R4 is used for RXTXData—a buffer that shifts in or out UART data bits. CPU register R5 is used for BitCnt, a bit-tracking register. The selection of R4 and R5 is arbitrary. Any CPU registers or RAM bytes can be used for these functions.

In receive mode, the capture-compare control register 0 (CCTL0) is initially configured such that CCR0 captures on the falling edge of receive pin P2.2. As the receive line idles high, a falling edge indicates the beginning of the start-bit. When the UART function is ready to receive data, no overhead is put on the CPU even though the function is ready to receive a character at any time. CPU resources are only exercised after a start-bit falling edge occurs on P2.2. A falling edge on P2.2 captures the current value of the free running timer_A3 counter register (TAR) in CCR0 independent of any other run-time activity. Capture is done by timer_A3 hardware, not by software. An interrupt is also issued to the CPU. The latency of the interrupt is not of great concern as the exact time the falling edge triggered the interrupt is stored in CCR0, independent of other activities. After start-bit edge detection, software reconfigures CCTL0 such that CCR0 is in a compare mode with the first compare to occur in the middle of the first data bit. A 1.5 bit offset is added to CCR0, positioning the next compare to the middle of the first data bit. Received data are hardware latched in timer_A3 synchronized capture-compare input (SCCI). SCCI is a readable latch in CCTL0. In the UART function, SCCI captures the logic level on the input P2.2 synchronously with the CCR0 compare. The UART function receives latched data from SCCI. Software does not test P2.2 directly.

```
RX_Bit      bit.w   #SCCI,&CCTL0            ; Get bit waiting in SCCI

            rrc.b   RXTXData               ; Store received bit
```

After the first data bit, a 1-bit length offset is added to CCR0 to position the next capture in the middle of each bit. Eight consecutive data bits are latched and received from SCCI by software into RXTXData bit-by-bit.

Transmit-mode tasks are simpler because the MSP430 determines when to transmit data, and no start-bit edge detect is required. Data stored in the RXTXData buffer are transmitted on pin P1.1 using timer_A CCR0 output hardware.

CCR0 is preconfigured in compare mode to transmit the next bit using output mode-control bits in CCTL0. Mode-control bits are preconfigured to RESET mode (logical 0), or a SET mode (logical 1) to occur on the next compare in CCR0. When compare occurs, at the beginning of a transmitted data bit, CCR0 hardware automatically outputs the preconfigured data bit to P1.1 and an interrupt is issued. With CCR0 hardware automatically outputting data bits on P1.1, software interrupt latency and bit-timing concerns are reduced. It is not necessary for software to prepare the CCR0 output latch at an exact time, only prior to the next bit. The UART function software does not directly output data on P1.1, but instead preloads the output data bit in CCR0 output latch hardware.

```
TX_Bit      rra.b   RXTXData              ; LSB is shifted to carry

            jc      TX_Mark               ; Jump if bit = 1

TX_Space    bis.w   #OUTMOD2,&CCTL0       ; TX Space

            reti                          ;

TX_Comp     bic.w   #CCIE,&CCTL0          ; All Bits RX, disable interrupt

TX_Mark     bic.w   #OUTMOD2,&CCTL0       ; TX Mark

            reti                          ;
```

Before each bit output, software rotates RXTXData to expose the bit into carry. The appropriate jump is made and CCTL0 output mode is prepared and a 1-bit length offset is added to CCR0.

Baud-rate Calculation

Timer_A3 CCR0 is used for baud-rate generation. Based on the required baud rate, an interval Bitime is calculated. Bitime is the length in timer_A3 counts between individual bits and the interval at which timer_A3 latch receives in and transmits out data bits (see Figure 2). Bitime is calculated as the timer_A3 clock source divided by the baud-rate.
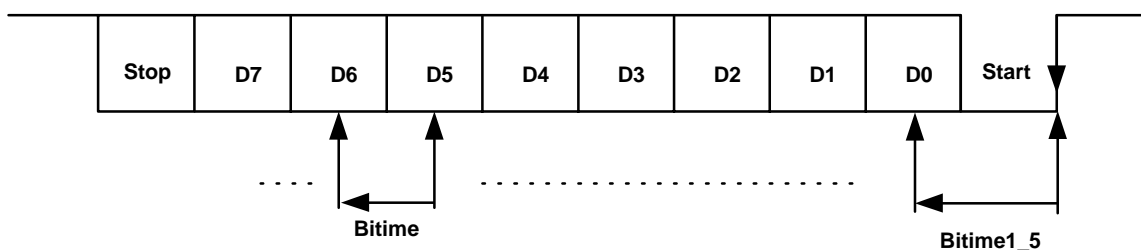

Figure 2.   UART8N1 Character

Timer_A3 has several available clock sources and dividers (see the device specific data sheet). Available clock sources for the MSP430x11x(1) timer_A3 module include the auxiliary clock (ACLK), subsystem clock (SMCLK) and two external clocks.

Example: Consider a 9600-baud-rate with the timer_A3 clock source selected as the auxiliary ACLK, which is configured to be the same as a 3.579545-MHz XTAL.

Bittime = 3 579 545 / 9600 = 372.9 ~ 373

Actual baud-rate = 3 579 545 / 373 = 9597

As only an integer value of Bitime can be added to CCR0, the value 373 is used. Assuming 9600-baud and a 3.579545-MHz clock source, the error of rounding Bitime to the nearest integer less than 0.03% per bit.

Software Overhead

A firmware engineer might be concerned by the system overhead of using a hardware/software UART as opposed to a dedicated serial port. The UART function described in this report uses the hardware features of timer_A3 to minimize the burden on the CPU. Timer_A3 hardware records the start-bit edge and latches individual data bits in and out automatically. The software required has a maximum of 26 cycles for each received or transmitted bit, including the interrupt service routine. The overhead is a function of the CPU clock (MCLK) and baud-rate. Using a 3.58-MHz MCLK for the 9600-baud example above, the overhead is calculated as:

Software overhead = (26 CPU cycles) $\times$ (9600BPS) / 3 579 545 = 6.9%

Demonstration Circuit

The example circuit in Figure 1 is powered directly by a PC serial port with regulation from a 3.3-V TPS76033 low-dropout voltage regulator. For demonstration purposes, the serial port interface is implemented using two TI SN74AHC1G04 inverters. If a fully-compliant RS232 interface is required, integrated circuits such as TI's low-power 3-V MAX2331 can be used. Reset is pulled high, and a 3.58-MHz ceramic resonator is used for clock generation.

Example Code 11x1_uart1.s43

The included example 11x1_uart1.s43 uses the circuit in Figure 1 and provides a basic echo function. A character is received from the PC and echoed back. During initialization, port pins are configured and all clocks are synchronized to the LFXT1 crystal oscillator. LFXT1 is configured to operate in high-speed (HF) mode. If, as in the example, the MCLK source is an external HF crystal, the crystal must be stable or the oscillator failsafe mode will automatically use the DCOCLK for MCLK. The OSCFAULT can be polled to ensure the crystal is stable before selecting this clock source for MCLK.

```
SetupBC     bis.b   #XTS,&BCSCTL1           ; ACLK = LFXT1 HF XTAL

SetupOsc    bic.b   #OFIFG,&IFG1            ; Clear OSC fault flag

            mov.b   #0FFh,R15

SetupOsc1   dec.b   R15                     ; Ddelay to ensure startup

            jnz     SetupOsc1

            bit.b   #OFIFG,&IFG1           ; OSC fault flag set?
```

```
         jnz     SetupOsc                    ;

         bis.b   #SELM1+SELM0,&BCSCTL2   ; (CPU) MCLK = LFXT1
```

The oscillator failsafe mode is described in *MSP430x1xx Family* User's Guide (SLAU049). The Mainloop calls the UART receive ready subroutine RX_Ready, then waits in low-power mode 0 (LPM0) with the CPU off. Only timer_A3 and ACLK are active. Even with the CPU turned off in the Mainloop, the UART receive-function operates interrupt-driven in the background. After the UART function receives a complete character into RXTXData, the UART interrupt handler returns the CPU to active in the Mainloop. The transmit subroutine is called next and echoes back the received character in RXTXData. The loop repeats, waiting for another character to be received. The example uses coding techniques optimized for speed. Inside the CCR0_ISR, BitCnt is used for auto-increment indirect addressing, to direct program flow to the exact section of the ISR required for handling the received or transmitted bit. Software is not required to poll flags or decrement registers to determine which action to take. Auto-increment addressing is used with a look-up table to direct program flow immediately.

```
         add.w   #Bitime,&CCR0       ; Bitime till next bit

         br      @BitCnt+            ; Branch To Routine
```

The advantage of auto-increment addressing is speed and efficiency in terms of the CPU cycles required for handling the ISR, at the expense of the code words required for the look-up table.

Example 11x1_uart2.s43—Using the DCO for Baud Generation

1. The example 11x1_uart2.s43 demonstrates how to implement a high baud-rate UART function even from MSP430 ultralow-power modes that use only a single 32 768 watch crystal. The MSP430's independent on-chip high-speed digitally-controlled oscillator (DCO) can be used for baud-rate generation. The DCO is a 100-kHz to 5-MHz and faster, digitally-tunable RC-type oscillator that starts and is stable in less than 6µs. With a fast DCO start, high baud rates are possible even from ultralow-power modes that require the DCO off—the falling edge of a start-bit is used as an interrupt which enables the DCO in less than 6 µs. Example 11x1_uart2.s43 provides a solution for a 9600-baud UART using the DCO for baud-rate generation. A 32 768 watch crystal is used for XTAL, replacing the 3.58-MHz resonator in Figure 1. The DCO is set to 1 228 800 Hz and used to drive the subsystem clock (SMCLK) which is configured as the source for timer_A3. The DCO is tuned to a high-speed value using a software frequency-locked loop (FLL). The software FLL integrates faster SMCLK (DCO clocks) over periods of the slower ACLK (derivatives of the 32 768-Hz crystal). The DCO clock is adjusted until the target frequency is reached (1 228 800 Hz). The DCO clock remains stable at the set frequency as long as V_{CC} and temperature are stable. The software FLL can be called at anytime to retune the DCO. See the current data sheet, *MSP430x11x1 Mixed Signal Microcontroller,* SLAS241, ref [1] for DCO specifications. In this example the Mainloop waits in LPM3—typically less than 2 µA —with the UART function fully ready to receive. The Mainloop echo operation is the same as the previous example. This example uses the BitCnt register simply to track the character-receiving process as opposed to using auto-increment as in the previous example.

TEXAS INSTRUMENTS

Example 11x1_uart3.s43—A 32-kHz Crystal for Baud Generation

In some cases a 32 768-Hz watch crystal is required as the direct clock source for timer_A3. A UART function is still possible, but at slower baud rates to prevent bit-timing errors. Example 11x1_uart3.s43 provides a solution for a 2400-baud UART, assuming that a 32 768-Hz watch crystal is used for XTAL (replacing the 3.58-MHz resonator in Figure 1). No software FLL is required to set the DCO, because the crystal generates the UART bit timing. When in active mode, the CPU is configured to execute code at the faster DCO clock rate—the CPU does not execute code at 32 768 Hz. Also in this example, RAM locations are used for RXTXData and BitCnt, as opposed to the CPU registers used in the other examples.

References

2. *MSP430x11x1 Mixed Signal Microcontroller,* Data Sheet, Texas Instruments Literature Number SLAS241.
3. *MSP430x1xx Family, User's Guide*, Texas Instruments Literature Number SLAU049.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Copyright © 2002, Texas Instruments Incorporated