Decide TV IR Remote Control Signals Using Timer_A3

Mark Buccini / Stefan Schauer

MSP430

ABSTRACT

This application report describes the use of Timer_A3 to decode RC5 and SIRC TV IR remote control signals. The decoder described in this report is interrupt-driven and operates as a background function using specific features the Timer_A3. Only a small portion of the MSP430 CPU’s nonreal-time resources is used. Specific hardware bit-latching capabilities of the Timer_A3 module are used for real-time decoding of the IR data signal, independent and asynchronous to the CPU. CPU activity and power consumption are kept to an absolute minimum level. The Timer_A3 decoder implementation also allows other tasks to occur simultaneously if required. The solutions provided are written specifically for MSP430x11x(1) and MSP430x12x derivatives, but can be adapted to any other MSP430 incorporating Timer_A3.

Contents

Introduction ................................................................. 2
Clock Selection ............................................................. 3
Demonstration Circuit ..................................................... 3
Decoding Software .......................................................... 4
RC5 Protocol .................................................................. 4
Example 11x1_rc5.243 ....................................................... 5
SIRC Protocol ................................................................. 6
Example 11x1_SRC.s43 ....................................................... 6
UART Software ............................................................... 7
PC Monitor ................................................................... 7
References .................................................................. 7
Appendix A – 11x1_SIRC.s43.txt ........................................ 8
Appendix B – 11x1_RC5.s43.txt ........................................... 15

List of Figures

1 IR Decoder Demonstration Circuit ........................................ 2
2 Inverted RC5 Data Packet as Seen by the MSP430 .................. 5
3 Inverted SIRC Data Packet as Seen by the MSP430 .............. 6

Trademarks are the property of their respective owners.
Introduction

Adding TV remote-control decoding capability to an MSP430 application is a low-cost method of enabling IR wireless communication. IR decoding capability can be added to an MSP430x11x(1) application using one Timer_A3 capture/compare register, less than 200 bytes of code, and an external sensor. See the demonstration circuit in Figure 1. For demonstration purposes, the decoded IR information packets are transmitted serially to a PC and a LED illuminates on P1.0 if a channel + code is received.

Figure 1. IR Decoder Demonstration Circuit
Both RC5 and SIRC protocols transmit packets of information serially using a 40-kHz modulated IR carrier. A logical 1 indicates the presence of 40-kHz modulated IR, and a 0 indicates its absence. The 40-kHz modulation is used to filter out natural forms of IR present from sources such as sunlight or office fluorescent lights. While 40 kHz is the most common modulation frequency, some systems use other frequencies in the 32-kHz to 64-kHz range. In order to decode RC5 or SIRC signals, the 40-kHz modulation must first be removed to expose the actual data bits in the serial packet. A simple three-pin VISHAY TSOP1840SS3V 3-V integrated sensor is used in this report to amplify, filter, and demodulate the IR signal, providing a clean logic-level output with only the serial data present. With no 40-kHz IR modulation present, the sensor output is high; when 40-kHz IR is present, the output is low. Thus, the sensor also has the effect of inverting the transmitted data in addition to removing the modulation. The IR-sensor output is connected directly to MSP430x11x(1) input pin P1.2. P1.2 is configured by software as a capture/compare function for Timer_A3 capture/compare register 1 (CCR1) using the port 1 option-select (P1SEL) register. Using the capture/compare features of Timer_A3 enables much easier decoding of the IR data. CCR1 does the IR data receive and transmit bit latching in hardware, independent of CPU and other system activity. The IR decoding is done as a background task using minimum CPU resources. To communicate the received IR data, a UART is implemented using CCR0. The UART transmit function is configured on P1.1.

**Clock Selection**

Both RC5 and SIRC packet timings are relatively slow (>1 ms/bit) compared to the operation of the MSP430. The demonstration circuit uses a common 32,768-Hz watch crystal as the source for the auxiliary clock (ACLK), which is also selected as the Timer_A3 clock source. With this clock source, Timer_A3 has a resolution of 30.5 µs—more than enough accuracy to resolve either RC5 or SIRC protocols with no bit errors. The on-chip digitally-controlled oscillator (DCO) is used at the default frequency of approximately 1 Mhz for the CPU master clock (MCLK). As the 32,768 Hz watch crystal sources the clock for Timer_A3 and IR decoder function, CPU speed is not critical. The CPU only needs to operate sufficiently fast to manage the tasks required. Using the slower ACLK for the IR decoder and the faster DCO for the CPU, both ultralow-power standby and fast-burst code execution are enabled. The MSP430 CPU must not be clocked from the slow watch crystal to eliminate the occurrence of long interrupt latency and higher power consumption. The CPU must be clocked fast, but in short interrupt-driven intervals, to conserve power and allow rapid noncompromised program execution.

**Demonstration Circuit**

The demonstration circuit is powered directly by a PC serial port with regulation from a 3.3-V TPS76033 low-dropout voltage regulator. One low-power LED is used in the circuit on P1.0 to indicate if a channel + command has been received. A serial port interface on P1.1 is implemented using a TI SN74AHC1G04 inverter. If a fully compliant RS232 interface is required, integrated circuits such as TI's low-power 3-V MAX3221 can be used. Reset is pulled high and a 32,768 Hz watch crystal is used for clock generation. No phase-shift capacitors are required if a watch crystal is used, as these are integrated in the MSP430 clock buffer.
Decoding Software

Two software examples are included. Example 11x1_rc5.s43 decodes the RC5 protocol, and 11x1_sirc.s43 decodes SIRC. The Mainloop of both examples is short and operates identically. Only the background IR decoder software is unique.

```
Mainloop call    #IR_Ready          ; Ready IR decoder
    bis.w   #LPM3,SR           ; Enter LPM3, stop, save power
    call    #TXIR_2_PC         ; TX received command
    call    #LED_Disp          ; Test for Channel +
    jmp     Mainloop           ;
```

The IR decoder function is enabled in Mainloop by calling the IR_Ready subroutine. Next, software in the Mainloop sets bits in the CPU status register (SR) to put the system into low-power mode 3 (LPM3). In LPM3, the CPU and DCO are off, but Timer_A3 is still counting from the ACLK with CCR1 interrupt logic fully active. Even though the system is in LPM3, the Timer_A3 driven decoder will run interrupt-driven in the background. The architecture of the MSP430 automatically enables the CPU and DCO when any enabled interrupt is requested. The DCO starts and becomes stable in less than 6 µs. Short burst events can be processed efficiently. Additionally, when an enabled interrupt occurs, the system automatically saves the original SR on the stack and clears the SR low-power bits inside of the interrupt service routine. After the interrupt service routine has been processed, the reti (return from interrupt) instruction pops the original SR off the stack. The system returns to the previous state prior to the interrupt service routine—unless the SR on the stack is modified inside of the interrupt service routine.

In this report, after a complete IR packet has been received using the background CCR1 interrupt service routine, software returns the CPU to active in Mainloop by clearing the LPM3 bits from the SR saved on the stack. This is a convenient way of managing the Mainloop with true event-driven programming. The received IR data packet is converted to four ASCII characters and transmitted to a PC using 2400 baud 8N1 UART protocol by calling the TXIR_2_PC subroutine. The four ASCII characters are preceded by a carriage return and line-feed character. The active Mainloop completes by calling the LED_Disp subroutine. The LED_Disp subroutine will set P1.0 to power the LED if the IR data packet is a channel + command. The Mainloop repeats, waiting in LPM3 for the next IR data packet.

RC5 Protocol

The RC5 protocol is a type of Manchester encoded data packet. Manchester data is unique in that a data is signified by a transition in the middle of the bit. A 1 is received by the MSP430 (after inversion by the IR sensor) as a high-to-low transition, and a 0 as a low-to-high transition. The RC5 IR packet consists of 14 bits: two start bits (S1, S0), one control bit (C), five address bits (A4 to A0), and a six bit command code (C5 to C0). The entire 14-bit packet is received MSB first, starting with two start bits. Figure 2 shows the RC5 packet as received by the MSP430 after demodulation and inversion from the IR sensor. The start bits are always transmitted as 1. The control bit toggles whenever a new key is received. The five address bits represent 32 different potential addresses of the equipment for which the packet is intended. The six command bits represent 64 commands that can be transmitted. The bit period for RC5 is 1.78 ms long, with half of that period high and the other half low. The duration for the complete 14-bit packet is approximately 25 ms.
Example 11x1_rc5.243

Two CPU registers are used in the 11x1_RC5.s43 example. IRData (R6) receives the RC5 IR packet, and IRBit (R7) is used as a temporary counter to track the IR data bits as they are received. The choice of R6 and R7 is arbitrary—any two CPU registers or RAM bytes can be used. Two variables are defined: Bit_50 is 1/2 of an RC5 1.78-ms bit length in Timer_A3 clocks (same as ACLK), and Bit_75 is 1/4 of a bit length. As a 32,768-Hz watch crystal is used in this report to generate the ACLK, BIT_50 and Bit_75 are as follows:

```
Bit_50  equ     29 ; 890 µs @ 32768 ACLK
Bit_75  equ     44 ; 1348 µs @ 32768 ACLK
```

The subroutine IR_Ready enables CCR1 to capture the Timer_A3 counter register (TAR) and request an interrupt on the falling edge of the IR sensor output, as configured on P1.2. TAR is captured on the falling edge and automatically stored in CCR1 and a TA1_ISR interrupt is requested. Inside of the TA1_ISR, software determines if the interrupt was triggered from a capture or compare. As a capture was the source of the first interrupt, BIT_75 (¼ of a bit length) is added directly to CCR1, which stored the exact time the IR sensor output edge fell in the middle of the first bit. Because the stored CCR1 capture occurred in the middle of the first bit, adding ¼ of a bit length will effectively offset CCR1 to the middle of the first half of the next bit. CCR1 is now reconfigured by software to compare mode. The next CCR1 interrupt is now timed for the middle of the first half of the second start bit. As configured, when the CCR1 compare occurs, the logic level present at P1.2 will be latched into the register’s synchronous capture compare input (SCCI) latch. SCCI provides the very important feature of enabling CCR1 hardware to capture and store the logic level on P1.2 with the exact timing generated from Timer_A3, irrespective of other system or CPU activities. In this application, the CPU is actually off while the IR decoder is enabled. Even with the CPU off, the logic level of P1.2 will be captured and stored in SCCI exactly when the CCR1 compare occurs. Software does not directly read P1.2; instead it reads the latched data in SCCI after the event. Software will recover received data from SCCI bit by bit, shifting the data into the storage register IRData.
After each bit has been recovered, CCR1 is reconfigured to capture on both rising and falling edges. The next capture will occur and self-synchronize on the next bit’s midpoint transition. To insure the data packet is decoding properly, the CCR2 interrupt is also enabled and loaded with a value of 1/2 the bit length of the compare stored in CCR1. The next bit transition received should occur in approximately 1/4 of a bit length if the packet is decoding properly. Inside of a normally occurring CCR1_ISR CCR1 edge capture, CCR2 interrupt is cleared by software. If CCR1_ISR does not capture a normally occurring edge and clears the CCR2 interrupt, CCR2_ISR will reset the decoder assuming an overrun error. In normal operation, the balance of the bits are recovered and received into IRData.

**SIRC Protocol**

The SIRC protocol uses a data packet with an encoding scheme of variable bit length. The length of a bit determines its logical value. The start bit is 2.4 ms of modulated IR, a 600-µs 0, and a 1.2-ms 1. All data bits, excluding the start bit, also include a 600-µs sync pulse, or lack of IR presence. The total length of a received 0, including the sync pulse, is therefore 1200 µs, and the total length of a 1 is 1800 µs. A complete SIRC packet consists of the start bit and 12 data bits. The 12 data bits are comprised of a seven-bit command code (C6 to C0) and a five-bit device code (D4 to D0). The SIRC protocol sends data LSB first. C0 is the first bit received following the start bit.

Figure 3 shows the SIRC data packet as received by the MSP430 after demodulation and inversion from the IR decoder.

![Figure 3. Inverted SIRC Data Packet as Seen by the MSP430](image)

**Example 11x1_SRC.s43**

Three CPU registers are used in the 11x1_SRC.s43 example: IRData (R6) receives the IR data, IRBit (R7) is used to track bits as they are received, and IRlength (R8) is used to store the length of the data bits as they are received. Three variables are defined: IR_Mid is 1/2 of an SIRC bit length in Timer_A3 clocks, IR_Start is approximately 2.3 ms in Timer_A3 clocks (the minimum length of a valid start bit), and IR_Start2 is approximately 2.5 ms in Timer_A3 clocks (the maximum length of a valid start bit).
The subroutine IR_Ready enables CCR1 to capture on a falling edge from the IR sensor connected to P1.2. The MSP430 system is then placed in LPM3 with only ACLK and Timer_A3 active. On the first falling edge, indicating the beginning of the start bit, a CCR1 capture occurs capturing TAR into CCR1 and requesting interrupt TA1_ISR. The count in TAR is automatically captured and stored in CCR1 by hardware, no software is required. CCR1 is stored by software in IRLength after the capture. After the first falling edge, CCR1 capture edge is switched to a rising edge that will capture at the end of the start bit. The length of a data bit is calculated by subtracting the current TAR capture stored in CCR1 from the previous saved in IRLength. If the start bit length received is not between 2.3 ms and 2.5 ms, the decoder software will reset assuming that an error has occurred. A valid start bit should be approximately 2.4 ms. Inside the TA1_ISR, the IRBit is used to count down the 12 data bits as they are received. Each data bit is calculated by comparing its bit length to IR_Mid. The length of IR_Mid is 1500 μs, which is 1/2 the difference between a 1 and a 0. A bit length greater than IR_Mid is decoded as a 1, a length less than IR_Mid is decoded as a 0. Hardware capturing of Timer_A3 insures that software interrupt latency does not effect the accuracy of the captured timer value and the calculated bit length. Software does not directly read Timer_A3, but instead, the latched timer value in CCR1. The system stack is also used to temporarily save the current CCR1 value to be subtracted from the previous in IRLength.

```
IR_ST_Test  push.w &CCR1 ; Save CCR1 count to stack
        sub.w   IRLength,0(SP) ; Time length last capture
        ir.Bit   cmp.w   #IR_Mid,0(SP) ; C=1 if IR RXed bit = 1
        ir.Shift  rrc.w   IRData ; Carry ->IRData
```

With CCR1 hardware capturing the Timer_A3 value exactly when edges occur on P1.2, other real-time activities can occur simultaneously with the IR decoder. The IR decoder software runs interrupt-driven in the background. Data are shifted into IRData bit-by-bit under software control after each bit has been received.

**UART Software**

The UART function is implemented with CCR0 and uses two CPU registers: RXTXData (R4) and BitCnt (R5). A complete description of the UART function is provided in a separate application report, please see the References section of this report.

**PC Monitor**

A standard PC terminal program can be used to receive the serial data packets transmitted from the demonstration circuit. The 16-bit IRData are right justified and sent as four ASCII characters. The terminal program must be set at 2400 baud 8N1.

**References**

1. *MSP430x11x1 Mixed Signal Microcontroller* datasheet, Texas Instruments literature number SLAS241
2. *MSP430x1xx Family Users Guide*, Texas Instruments literature number SLAU049
3. *Implementing a UART Function with Timer_A3*, Texas Instruments literature number SLAA078
4. *Photo Module for PCM Remote Control System*, VISHAY literature number 82052
Appendix A  11x1_SIRC.s43.txt

; THIS PROGRAM IS PROVIDED "AS IS". TI MAKES NO WARRANTIES OR
; REPRESENTATIONS, EITHER EXPRESS, IMPLIED OR STATUTORY,
; INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
; FOR A PARTICULAR PURPOSE, LACK OF VIRUSES, ACCURACY OR
; COMPLETENESS OF RESPONSES, RESULTS AND LACK OF NEGLIGENCE.
; TI DISCLAIMS ANY WARRANTY OF TITLE, QUIET ENJOYMENT, QUIET
; POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY
; INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE PROGRAM OR
; YOUR USE OF THE PROGRAM.
;
; IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, INCIDENTAL,
; CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY
; THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED
; OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT
; OF THIS AGREEMENT, THE PROGRAM, OR YOUR USE OF THE PROGRAM.
; EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF
; REMOVAL OR REINSTALLATION, COMPUTER TIME, LABOR COSTS, LOSS
; OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, OR LOSS OF
; USE OR INTERRUPTION OF BUSINESS. IN NO EVENT WILL TI’S
; AGGREGATE LIABILITY UNDER THIS AGREEMENT OR ARISING OUT OF
; YOUR USE OF THE PROGRAM EXCEED FIVE HUNDRED DOLLARS
; (U.S.$500).
;
; Unless otherwise stated, the Program written and copyrighted
; by Texas Instruments is distributed as "freeware". You may,
; only under TI’s copyright in the Program, use and modify the
; Program without any charge or restriction. You may
; distribute to third parties, provided that you transfer a
; copy of this license to the third party and the third party
; agrees to these terms by its first use of the Program. You
; must reproduce the copyright notice and any other legend of
; ownership on each copy or partial copy, of the Program.
;
; You acknowledge and agree that the Program contains
; copyrighted material, trade secrets and other TI proprietary
; information and is protected by copyright laws,
; international copyright treaties, and trade secret laws, as
; well as other intellectual property laws. To protect TI’s
; rights in the Program, you agree not to decompile, reverse
; engineer, disassemble or otherwise translate any object code
; versions of the Program to a human-readable form. You agree
; that in no event will you alter, remove or destroy any
; copyright notice included in the Program. TI reserves all
; rights not specifically granted under this license. Except
; as specifically provided herein, nothing in this agreement
; shall be construed as conferring by implication, estoppel,
; or otherwise, upon you, any license or other right under any
; TI patents, copyrights or trade secrets.
;
; You may not use the Program in non-TI devices.
;
#include "msp430x11x1.h"

;***************************************************************************
;   MSP-FET430X110 Demo - Decode SIRC IR Remote Control / TX to PC @ 2400
;***************************************************************************
;
; Description: Decode 12-bit SIRC format IR packet using Timer_A.
; Timer_A CCR1 is used to decode IR packet, capture mode to measure IR bit
; length. Received packet is TXed to PC using Timer_A CCR0 as a UART
; function. Packet sent as four ACII bytes, preceded by a CR and LF
; character. P1.0 is set if channel+ is RXed, reset if not. IR data are
; received LSB first. Start, 12-bits of data.
; D4-D3-D2-D1-D0-C6-C5-C4-C3-C2-C1-C0-Start
;
; Demonstrate with IR monitor - TX IRData as CR, LF, 4 ASCII Bytes
;
; MSP430F1121
;          ---------------
;          \       |      XIN|--
;           |      | 32kHz
;           --| RST XOUT|--
;          |      |     |
; IR Receiver->|P1.2/CCR1 P1.0|-- LED0
;           | P1.1|-- 2400 8N1
;
; Bit pattern as seen at MSP430
; Start = 2.4ms low ~ 79 @ 32kHz ACLK
; 1 = 1.2ms low
; 2 = 0.6ms low
; sync = 0.6ms high
SLAA134

; sync sync sync sync
; ---+ ++++ ++++ ++++ ++++
; |
; \-----\\-----+- -+-+-+-+-++
; ^ 0  ^ 1  ^ 0  ^ Start ^

; CPU registers used
#define RXTXData R4
#define BitCnt R5
#define IRData R6
#define IRBit R7
#define IRlength R8

; Conditions for 2400 Baud SW UART, ACLK = 32768
Bitime_5 equ 06      ; .5 bit length + small adj.
Bitime equ 014; 427us bit length ~ 2341 baud

IR_Mid equ 49      ; 1500us @ 32768Hz ACLK
IR_Start equ 75    ; 2300us @ 32768Hz ACLK
IR_Start2 equ 82   ; 2500us @ 32768Hz ACLK

LED0 equ 001h       ; LED0 on P1.0
TXD equ 002h        ; TXD on P1.1
IRIN equ 004h       ; IR input on P1.2
Ch_up equ 16
Ch_dwn equ 17
LF equ 0ah           ; ASCII Line Feed
CR equ 0dh           ; ASCII Carriage Return

; M. Buccini
; Texas Instruments, Inc
; July 2001

;*****************************************************************************
;----------------------------------------------------------------------------

ORG 0F000h     ; Program Start

RESET    mov.w #0300h,SP       ; Initialize 'x112x stackpointer
call     #Init_Sys         ; Initialize System Peripherals

10  Decode TV IR Remote Control Signals Using TimerA3
Mainloop    call    #IR_Ready               ; Ready IR decoder
           bis.w   #LPM0,SR                ; Enter LPMx, stop, save power
           call    #TXIR_2_PC              ; TX received command
           call    #LED_Disp               ; Test for Channel +/-
           jmp     Mainloop                ;

;---------------------------------------------------------------
Init_Sys;   Initialize System Peripherals
;---------------------------------------------------------------
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop Watchdog Timer
SetpTA      mov.w   #TASSEL0+MC1,&TACTL     ; ACLK, continuous
SetupC0      mov.w   #OUT,&CCTL0             ; TXD Idle as Mark
SetupP1      bis.b   #IRIN+TXD,&P1SEL        ; P1.2 CCR1, P1.1 CCR0
            bis.b   #LED0+TXD,&P1DIR        ; P1.0, TXD outputs
            bic.b  #LED0,&P1OUT            ; P1.0, low, LED off
            eint                            ;
            ret                             ; Return from subroutine
;---------------------------------------------------------------
IR_Ready;   Subroutine prepares to receive 12-bit SIRC into IRData buffer
;---------------------------------------------------------------
clr.w   IRData                  ;
clr.w   IRlength                ;
mov.b   #14,IRBit               ; Two start edges and 12 data bits
SetupC1    mov.w   #CM1+SCS+CAP+CCIE,&CCTL1    ; CAP CCI1A, falling edge, int
ret         ; Return from subroutine
;---------------------------------------------------------------
TXIR_2_PC;  Subroutine to send CR, LF and IRData as four ASCII bytes to PC
; R15 used as working register and not saved
;---------------------------------------------------------------
mov    #CR,RXTXData            ; CR to UART buffer
call   #TX_Byte                ; CR -> PC/user
mov    #LF,RXTXData            ; LF to UART buffer
call   #TX_Byte                ; CR -> PC/user
;---------------------------------------------------------------
TX_Word_ASCII; TX Word from IRData as four ASCII bytes
swpb   IRData                  ; IRData = 3412
call   #TX_Byte_ASCII          ;
swpb   IRData                  ; IRData = 1234

Decode TV IR Remote Control Signals Using TimerA3
TX_Data_ASCII: TX Byte from IRData in two ASCII bytes

    mov.b  IRData,R15               ; transmit ..x. of value
    call   #NUM_ASCIR
    mov.b  IRData,R15               ; transmit ...x of value
    jmp    NUM_ASCIA

NUM_ASCIR      rrc.b  R15                     ; 1. and 3. pass
                 rrc.b  R15                     
                 rrc.b  R15                     
                 rrc.b  R15                     

NUM_ASCIA      and.b  #0fh,R15                ; 2. and 4. pass
                 add.b  #030h,R15
                 cmp.b  #03ah,R15
                 jlo    NUM_End
                 add.b  #039h,R15

NUM_End        mov.b  R15,RXTXData            ; load transmit buffer, FALL

TX_Data: Subroutine to TX Data from RXTXData Buffer using CCR0 UART

    mov.w   &TAR,&CCR0                ; Current state of TA Counter
    add.w   #Bitime,&CCR0             ; Some time till first bit
    bis.w   #0100h, RXTXData          ; Add mark stop bit to RXTXData
    rla.w   RXTXData                  ; Add space start bit
    mov.w   #10h,BitCnt               ; Load Bit Counter, 8 data + SP
    mov.w   #OUTMOD0+CCIE,&CCTL0      ; TXD = mark = idle
    TX_Wait     tst.w   BitCnt             ; Wait for TX completion
                jnz     TX_Wait
                ret

TA0_ISR; RXTXData Buffer holds UART Data

    add.w   #Bitime,&CCR0             ; Time to Next Bit
    UART_TX    bic.w   #OUTMOD2,&CCTL0   ; TX Mark
                rra.w   RXTXData                 ; LSB is shifted to carry
                jc      TX_Test
    TX_Space   bis.w  #OUTMOD2,&CCTL0   ; TX Space
TX_Test   dec.w BitCnt      ; All bits sent (or received)?
jnz TX_Next   ; Next bit?
bic.w #CCIE,&CCTL0  ; All Bits TX/RX, Disable Int.
TX_Next     reti

;----------------------------------------------------------------------------------------------------------------------
TAX_ISR;   Common ISR - CCR1-4 and overflow
;----------------------------------------------------------------------------------------------------------------------
add.w &TAIV,PC  ; Add Timer_A offset vector
reti  ; CCR0 - no source
jmp TA1_ISR    ; CCR1
; jmp TA2_ISR    ; CCR2
; reti  ; CCR3 - not used
; reti  ; CCR4 - not used
;TA_over  reti  ; TA overflow - not used

TA1_ISR    mov.w #CM0+SCS+CAP+CCIE,&CCTL1  ; CAP CCI1A,rising edge, int
IR_ST_Test push.w &CCR1      ; Temp save to stack CCR1 count
sub.w IRlength,0(SP)  ; Time length last capture
cmp.b #14,IRBit     ; First falling edge?
jeq IR_Next  ; Jump
jmp IR_Bit       ; Jump
; cmp.w #IR_Start2,0(SP)  ; Start bit > 2.5ms
; jge IR_error       ; Jump-- > IRlength > 2.5ms
jmp IR_Bit       ; Jump--
cmp.w #IR_Start,0(SP)  ; Start bit minimum ~ 2.5ms
jge IR_Next  ; Jump-- IRlength > 2.3ms
IR_error    incd.w SP  ; Clean up stack
call #IR_Ready  ; ERROR - restart RX sequence
reti  ; Return from interrupt

IR_Bit      cmp.w #IR_Mid,0(SP)  ; C=1 if IR RXed bit = 1
IR_Shift    rrc.w IRData     ; Carry ->IRData
IR_Next     mov.w &CCR1,IRlength  ; Save captured edge
incd.w SP  ; Clean up stack
dec.b IRBit
jnz IR_Cont  ; Jump-- not last bit

IR_Comp     clr.w &CCTL1     ; Disable CCR1
rrc.w IRData ; 12-bit IRData right justified
rrc.w IRData ;

Decode TV IR Remote Control Signals Using TimerA3
Decode TV IR Remote Control Signals Using TimerA3

```assembly
rrc.w IRData ;
rrc.w IRData ;
and.w #0FFFh,IRData ; Isolate 12-bit packet
mov.w #GIE,0(SP) ; Decode Byte = Active in Mainloop
IR_Cont reti ;
;

;---------------------------------------------
LED_Disp; LED0 (P1.0) set if IRData = Channel+ code (16)
;---------------------------------------------
and.w #07Fh,IRData ; Isolate 7-bit command code
LED_off bic.b #01h,&P1OUT ; LED0 off
LED0_tst cmp.w #Ch_up,IRData ; Test for Channel+ (32)
jne LED_exit ;
bis.b #01h,&P1OUT ; LED0 on
LED_exit ret ; Return from subroutine ;
;

;---------------------------------------------
; Interrupt Vectors Used
;---------------------------------------------
ORG 0FFFEh ; MSP430 RESET Vector
DW RESET ;
ORG 0FF2h ; Timer_A0 Vector
DW TA0_ISR ;
ORG 0FF0h ; Timer_AX Vector
DW TAX_ISR ;
END
```
Appendix B  11x1_RC5.s43.txt

; THIS PROGRAM IS PROVIDED “AS IS”. TI MAKES NO WARRANTIES OR
; REPRESENTATIONS, EITHER EXPRESS, IMPLIED OR STATUTORY,
; INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
; FOR A PARTICULAR PURPOSE, LACK OF VIRUSES, ACCURACY OR
; COMPLETENESS OF RESPONSES, RESULTS AND LACK OF NEGLIGENCE.
; TI DISCLAIMS ANY WARRANTY OF TITLE, QUIET ENJOYMENT, QUIET
; POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY
; INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE PROGRAM OR
; YOUR USE OF THE PROGRAM.
;
; IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, INCIDENTAL,
; CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY
; THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED
; OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT
; OF THIS AGREEMENT, THE PROGRAM, OR YOUR USE OF THE PROGRAM.
; EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF
; REMOVAL OR REINSTALLATION, COMPUTER TIME, LABOR COSTS, LOSS
; OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, OR LOSS OF
; USE OR INTERRUPTION OF BUSINESS. IN NO EVENT WILL TI’S
; AGGREGATE LIABILITY UNDER THIS AGREEMENT OR ARISING OUT OF
; YOUR USE OF THE PROGRAM EXCEED FIVE HUNDRED DOLLARS
; (U.S.$500).
;
; Unless otherwise stated, the Program written and copyrighted
; by Texas Instruments is distributed as “freeware”. You may,
; only under TI’s copyright in the Program, use and modify the
; Program without any charge or restriction. You may
; distribute to third parties, provided that you transfer a
; copy of this license to the third party and the third party
; agrees to these terms by its first use of the Program. You
; must reproduce the copyright notice and any other legend of
; ownership on each copy or partial copy, of the Program.
;
; You acknowledge and agree that the Program contains
; copyrighted material, trade secrets and other TI proprietary
; information and is protected by copyright laws,
; international copyright treaties, and trade secret laws, as
; well as other intellectual property laws. To protect TI’s
; rights in the Program, you agree not to decompile, reverse
; engineer, disassemble or otherwise translate any object code
; versions of the Program to a human-readable form. You agree
; that in no event will you alter, remove or destroy any
; copyright notice included in the Program. TI reserves all
; rights not specifically granted under this license. Except
; as specifically provided herein, nothing in this agreement
; shall be construed as conferring by implication, estoppel,
; or otherwise, upon you, any license or other right under any
; TI patents, copyrights or trade secrets.
;
; You may not use the Program in non-TI devices.
;
#include "msp430x14x.h"
;
****************************************************************************
; MSP-FET430X110 Demo - Decode RC5 IR Remote Control / TX to PC @ 2400
;
; Description: Decode 12-bit bi-phase RC5 format IR packet using Timer_A.
; Timer_A CCR1 is used to decode IR packet, capture mode to detect mid-bit
; edge and compare mode to latch data bit. CCR2 is used for decoder
; over-run detection. Received packet is TXed to PC using Timer_A CCR0 as
; a UART function. Packet sent as four ACII bytes, preceded by a CR and LF
; character. P1.0 is set if channel+ is RXed, reset if not.
; IR data are received MSB first. 2 Start, C and 11-bits of data.
; S1-S2-C-A4-A3-A2-A1-A0-C5-C4-C3-C2-C1-C0
;
; Demonstrate with IR monitor - TX IRData as CR, LF, 4 ASCII Bytes
;
;    MSP430F1121
;    ----------------------------
;    /|\    XIN-   
;    |   |  32kHz
;    --|RST   XOUT- |
;    |   |   |
; IR Receiver-->|P1.2/CCR1   P1.0|---> LED
;    |       P1.1|---> 2400 8N1
;
; Bit pattern as seen at MSP430
;
; 1.78ms
; ---- ---- ---- ---- ---- ----

Decode TV IR Remote Control Signals Using TimerA3
Decode TV IR Remote Control Signals Using TimerA3

; CPU registers used
#define      RXTXData    R4
#define      BitCnt      R5
#define      IRData      R6
#define      IRBit       R7

; Conditions for 2400 Baud SW UART, ACLK = 32768
Bitime_5   equ    06           ; .5 bit length + small adj.
Bitime     equ    014          ; 427us bit length ~ 2341 baud

LED0       equ     001h        ; LED0 on P1.0
TXD        equ     002h        ; TXD on P1.1
IRIN       equ     004h        ; IR input on P1.2
Bit_50     equ     29          ; 890 us @ 32768 ACLK
Bit_75     equ     44          ; 1348 us @ 32768 ACLK
Ch_up      equ     32          ;
Ch_dwn     equ     33          ;
LF          equ     0ah         ; ASCII Line Feed
CR          equ     0dh         ; ASCII Carriage Return

; M. Buccini
; Texas Instruments, Inc
; July 2001
;-----------------------------------------------------------------------
;-----------------------------------------------------------------------
ORG     0F000h            ; Program Start

;-----------------------------------------------------------------------
RESET       mov.w   #0300h,SP               ; Initialize 'x112x stackpointer
call    #Init_Sys               ; Initialize System Peripherals

Mainloop    call    #IR_Ready               ; Ready IR decoder
            bis.w   #LPM3,SR                ; Enter LPMx, stop, save power
call    #TXIR_2_PC              ; TX received command
call    #LED_Disp               ; Test for Channel +/-
jmp     Mainloop                ;

;---------------------------------------------------------------------------

Init_Sys; Initialize System Peripherals

;---------------------------------------------------------------------------

StopWDT  mov.w  #WDTPW+WDTHOLD,&WDTCTL  ; Stop Watchdog Timer
SetupTA  mov.w  #TASSEL0+MC1,&TACTL     ; ACLK, continuous
SetupC0  mov.w  #OUT,&CCTL0             ; TXD Idle as Mark
SetupP1  bis.b  #IRIN+TXD,&P1SEL        ; P1.2 CCR1, P1.1 CCR0
          bis.b  #LED0+TXD,&P1DIR        ; P1.0, TXD outputs
          bic.b  #LED0,&P1OUT            ; P1.0, low, LED off
          eint                            ;
          ret                             ; Return from subroutine

;---------------------------------------------------------------------------

IR_Ready; Subroutine to prepare to receive 12-bit RC5 into IRData

;---------------------------------------------------------------------------

clr.w   IRData                  ;
mov.w   #014,IRBit              ; 12 data + 1 start + completion
SetupC1  mov.w  #CM1+SCS+CAP+CCIE,&CCTL1    ; CAP CCI1A, falling edge, int
ret

;---------------------------------------------------------------------------

TXIR_2_PC; Subroutine to send CR, LF and IRData as four ASCII bytes to PC
;         R15 used as working register and not saved

;---------------------------------------------------------------------------

mov    #CR,RXTXData            ; CR to UART buffer
call   #TX_Byte                ; CR --> PC/user
mov    #LF,RXTXData            ; LF to UART buffer
call   #TX_Byte                ; CR --> PC/user

TX_Word_ASCII; TX Word from IRData as four ASCII bytes
swpb   IRData                  ; IRData = 3412
call   #TX_Byte_ASCII          
swpb   IRData                  ; IRData = 1234

TX_BYTE_ASCII; TX Byte from IRData as two ASCII bytes
mov.b  IRData,R15              ; transmit ..x. of value
call   #NUM_ASCIR              
mov.b  IRData,R15              ; transmit ...x of value
jmp    NUM_ASCIA               

NUM_ASCIR   rrc.b R15          ; 1. and 3. pass
           rrc.b R15
           rrc.b R15
           rrc.b R15

NUM_ASCIA   and.b #0fh,R15    ; 2. and 4. pass
           add.b #030h,R15
           cmp.b #03ah,R15
           jlo    NUM_End
           add.b #039,R15

NUM_End     mov.b R15,RXTXData ; load transmit buffer, FALL

TX_BYTE;   Subroutine to TX Byte from RXTXData Buffer using CCR0 UART

mov.w   &TAR,&CCR0           ; Current state of TA Counter
add.w   #Bitime,&CCR0        ; Some time till first bit
bis.w   #0100h, RXTXData     ; Add mark stop bit to RXTXData
rla.w   RXTXData             ; Add space start bit
mov.w   #10,BitCnt           ; Load Bit Counter, 8 data + SP
mov.w   #OUTMOD0+CCIE,&CCTL0 ; TXD = mark = idle
TX_Wait   tst.w   BitCnt      ; Wait for TX completion
           jnz     TX_Wait
           ret

TA0_ISR    ; RXTXData Buffer holds UART Data.

add.w   #Bitime,&CCR0        ; Time to Next Bit
UART_TX   bic.w   #OUTMOD2,&CCTL0 ; TX Mark
           rra.w   RXTXData ; LSB is shifted to carry
           jc     TX_Test ; Jump --> bit = 1
TX_Space  bis.w   #OUTMOD2,&CCTL0 ; TX Space
TX_Test   dec.w   BitCnt      ; All bits sent (or received)?
           jnz    TX_Next ; Next bit?
           bic.w   #CCIE,&CCTL0 ; All Bits TX/RX, Disable Int.
TX_Next   reti

TAX_ISR;   Common ISR - CCR1-4 and overflow
; Add Timer_A offset vector
add.w &TAIV,PC
reti ; CCR0 - no source
jmp TA1_ISR ; CCR1
jmp TA2_ISR ; CCR2
;
reti ; CCR3
;
reti ; CCR4
; TA_over
reti ; Return from overflow ISR
;
TA1_ISR bit.w #CAP,&CCTL1
jc RX_edge ; Jump -> Edge captured
;
RX_Bit dec.w IRBit
jz RX_Comp ; Test of end of packet
RX_Cont bit.w #SCCI,&CCTL1 ; Carry = Data bit in SCCI
rlc.w IRData ; Carry -> IRData
mov.w #CM1+CM0+CAP+CCIE+SCS,&CCTL1 ; CAP CCIIA,both edges, int
push.w &CCR1 ; Max time till next edge
add.w #Bit_50,0(SP)
pop.w &CCR2
mov.w #CCIE,&CCTL2 ; Enable timeout interrupt
reti
;
RX_Comp clr.w &CCTL1 ; Disable CCR1
and.w #0FFFh,IRData ; Isolate 12-bit packet
mov.w #GIE,0(SP) ; Decode = Active in Mainloop
reti
;
RX_edge clr.w &CCTL2 ; Disable CCR2 timeout
mov.w #CCIE,&CCTL1 ; Compare mode w/ int.
add.w #Bit_75,&CCR1 ; Time to middle of data bit
reti
;
TA2_ISR clr.w &CCTL2 ; Disable CCR2 timeout
call #IR_Ready ; ERROR - restart RX sequence
reti ; Return from interrupt
;
;-----------------------------------------------------------------------------------
LED_Disp; LED0 (P1.0) set if IRData = Channel+ code (32)
;-----------------------------------------------------------------------------------
and.w #03Fh, IRData ; Isolate 6-bit command code

LED_off bic.b #01h, &P1OUT ; LED0 off

LED0_tst cmp.w #Ch_up, IRData ; Test for Channel+ (32)
jne LED_exit ;

bis.b #01h, &P1OUT ; LED0 on

LED_exit ret ; Return from subroutine

;--------------------------------------------------------------
;
; Interrupt Vectors Used
;
;--------------------------------------------------------------

org 0FFFEh ; MSP430 RESET Vector
dw RESET ;

org 0FFF2h ; Timer_A0 Vector
dw TA0_ISR ;

org 0FFF0h ; Timer_AX Vector
dw TAX_ISR ;

end
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 © 2001, Texas Instruments Incorporated