

# **Use of Two MSP430s to Enhance Segment Lines for Larger LCDs**

Stefan Schauer

MSP430

## **ABSTRACT**

This application report explains a technique that uses two MSP430™ microcontrollers to expand the maximum number of LCD segments that can be driven with a single controller. Additional benefits of the technique include increases in available memory, data storage (RAM), and I/O control lines. These benefits are realized without the need for an external LCD driver circuit. The use of a second MSP430 controller also maintains the ultra-low power feature. The report includes a circuit schematic, software, and timing diagrams.

The MSP430F4xx and MSP430F6xx series have integrated LCD with support for up to 320 segments and 8-mux modes.

## **Contents**

1	Introduction .....	2
2	General Information About LCDs .....	2
3	LCD Module Integrated Into the MSP430 .....	3
4	Requirements for LCD Expansion With a Second MSP430 .....	4
5	Program Flow .....	8
Appendix A Listings .....		10

## **List of Figures**

1	4-Mux Waveform Drive Example Schematic .....	3
2	Example of 4-Mux Waveform Drive .....	4
3	Timing of the Signal Used for Synchronization .....	5
4	Interconnections Between the Two Controllers Necessary for LCD Expansion .....	5
5	Timing for Asynchronous Operation .....	6
6	Timing for Synchronous Operation .....	7
7	+Connecting a Large LCD to Two MSP430s.....	8
8	Connecting a Large LCD to Two MSP430s .....	9

## **List of Tables**

1	Pins Needed to Display 80 Segments With Different Multiplex Modes .....	4
2	Resources Required by the Controllers for LCD Expansion .....	8

## 1 Introduction

With the increasing performance of microcontrollers, the complexity of the functions they perform has grown. This often requires that more information be communicated to the user. If this information is displayed on an LCD, more pins are necessary to control the LCD. But microcontrollers with integrated LCD drivers, such as the MSP430, suffer from a limited number of available pins, and only a restricted number of segments can be displayed.

When facing this problem during development, the system designer has some alternatives:

- Reduce the amount of information displayed.
- Organize the information in a more hierarchical way, if this is possible.
- Use an external LCD driver, or use an LCD module with integrated controller. These modules are expensive and require additional pins for their control. Additionally, the communication between the microcontroller and the LCD should be programmed. Often, a serial protocol should be handled, which requires a lot of software overhead and CPU resources.
- Use a second MSP430, such as an MSP430P311. This method is described in this application report.

## 2 General Information About LCDs

The liquid crystal display should be driven with alternating voltage. A dc drive would destroy the liquid crystal. The frequency of the ac drive is low, in the 30 Hz to 100 Hz range. The particular manufacturer's data sheets for the LCD should specify this frequency range.

Different methods have been developed to control the LCD. These methods are a compromise between number of segments, number of pins of the display, driving source, LCD contrast, and temperature range.

Multiplex modes are usually used to reduce the number of segment pins required.

### 3 LCD Module Integrated into the MSP430

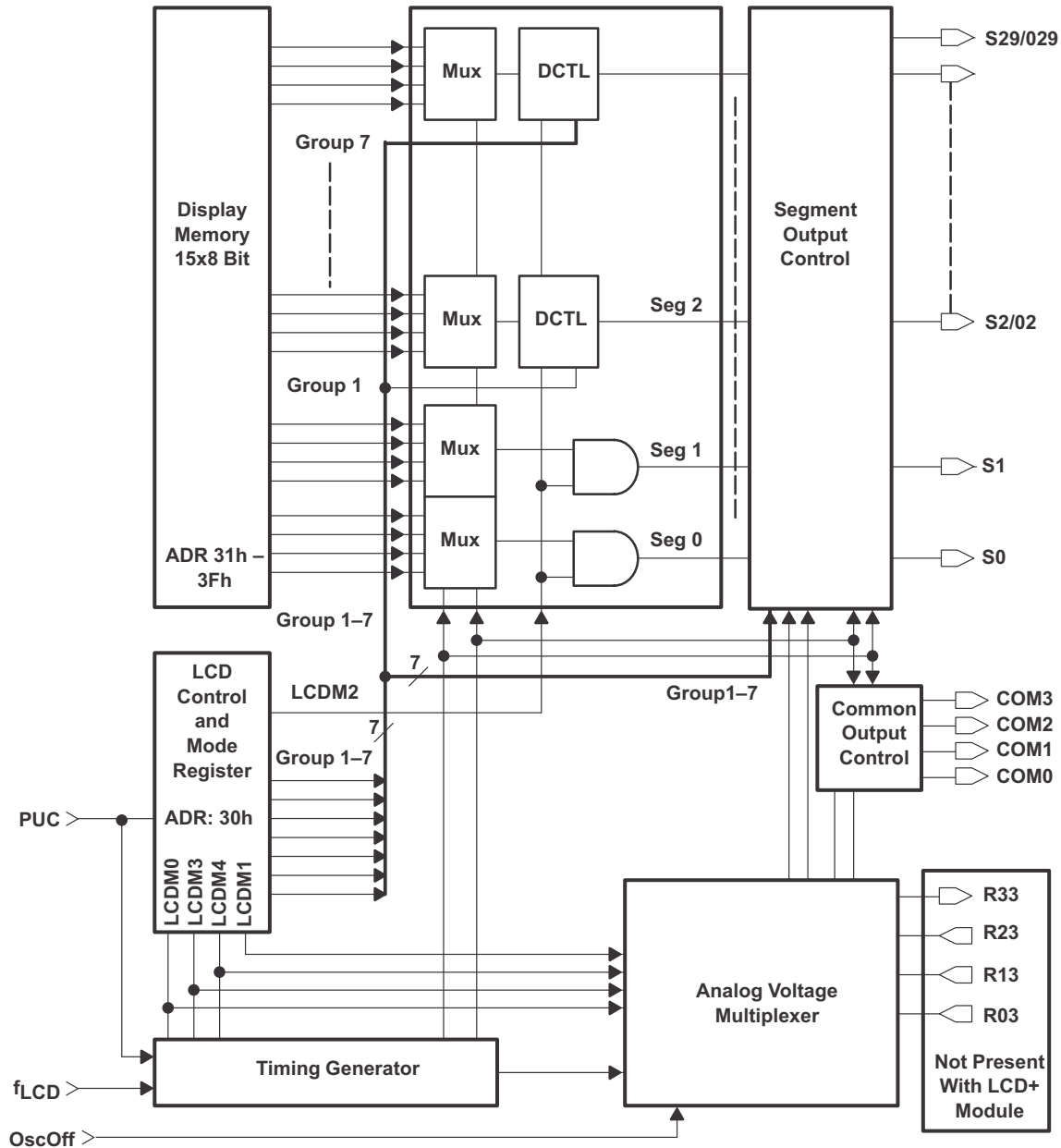


Figure 1. 4-Mux Waveform Drive Example Schematic

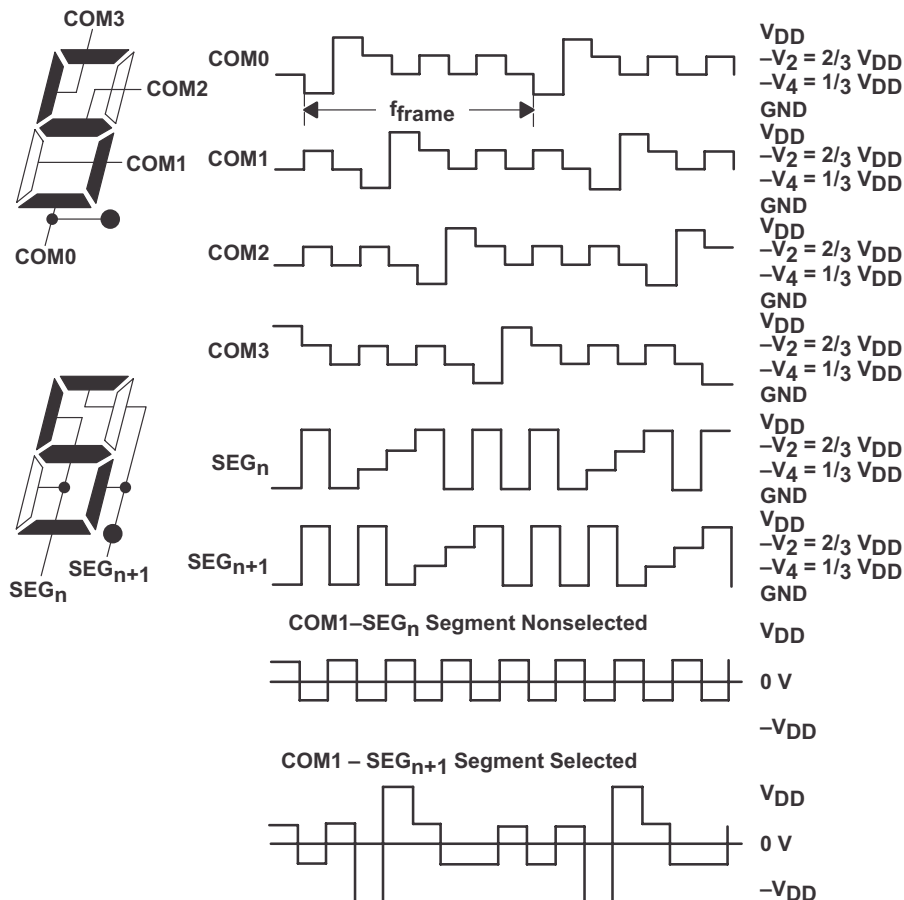
The MSP430 family's LCD module supports four driving methods:

- Static
- 2-mux or 1/2 duty, 1/2 bias
- 3-mux or 1/3 duty, 1/3 bias
- 4-mux or 1/4 duty, 1/3 bias

Increasing the multiplex rate reduces the number of pins required. The continuous reduction in the pin count is demonstrated in [Table 1](#) by an application using 80 segments.

**Table 1. Pins Needed to Display 80 Segments With Different Multiplex Modes**

Multiplexing Mode	Number of COM Lines	Number of Pins
Static	1	81
2-mux	2	42
3-mux	3	30
4-mux	4	24


**Figure 2. Example of 4-Mux Waveform Drive**

## 4 Requirements for LCD Expansion With a Second MSP430

Assuming that the 4-mux driving method is being used for the high segment count, the two controllers should be synchronized so that the two LCD timing generators run synchronously. This is also necessary with the 2-mux and 3-mux driving methods, but not with the static mode. The common lines of the LCD are connected to only one controller, and the segment lines are driven by both controllers.

### 4.1 How to Synchronize the Two Timing Generators

A special LCD module function is used for the synchronization. The segment lines are combined into groups of four and can be switched to digital outputs. If the digital output should be switched, the whole nibble, which corresponds to the segment line, should always be set or reset. Otherwise, the information stored in the LCD memory is shifted out at the pins controlled by the LCD timing generator. Using this function, the start of an LCD clock frame is detected without any additional hardware (such as a comparator).

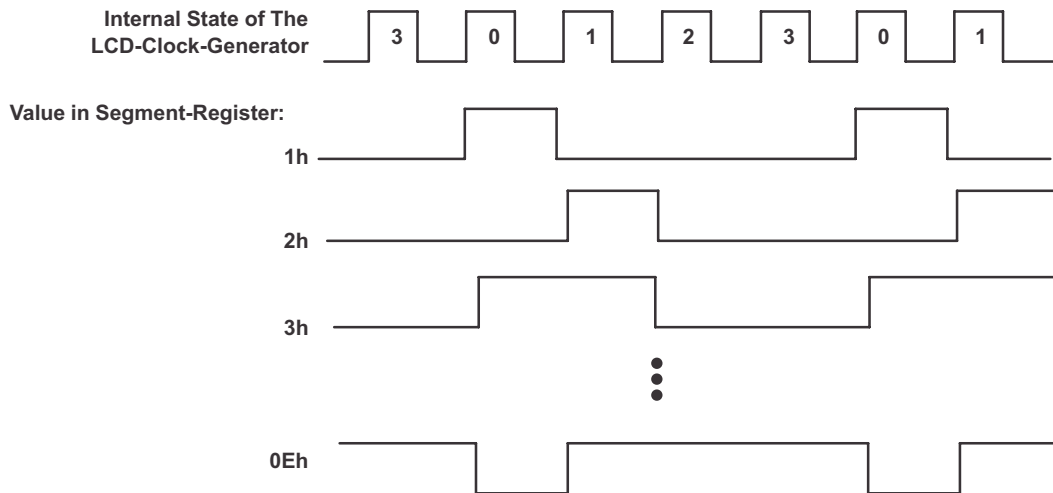


Figure 3. Timing of the Signal Used for Synchronization

Figure 3 shows the timing for a segment line in digital output mode. The value is set to the corresponding nibble of the control register.

Using this special feature, the timing generators in the two MSP430s can be easily synchronized as described in Section 4.2.

#### 4.2 How the Synchronization Operates

Both MSP430s run off the same ACLK. One controller (master) generates the ACLK and feeds it through the XBUF output to the XIN of the other controller (slave). Only one crystal is needed.

The second signal connection between the two controllers is a one-segment signal connected to a port pin of the slave. This port pin should have interrupt capability.

Also at the slave, a segment signal is fed back to another port pin with interrupt capability.

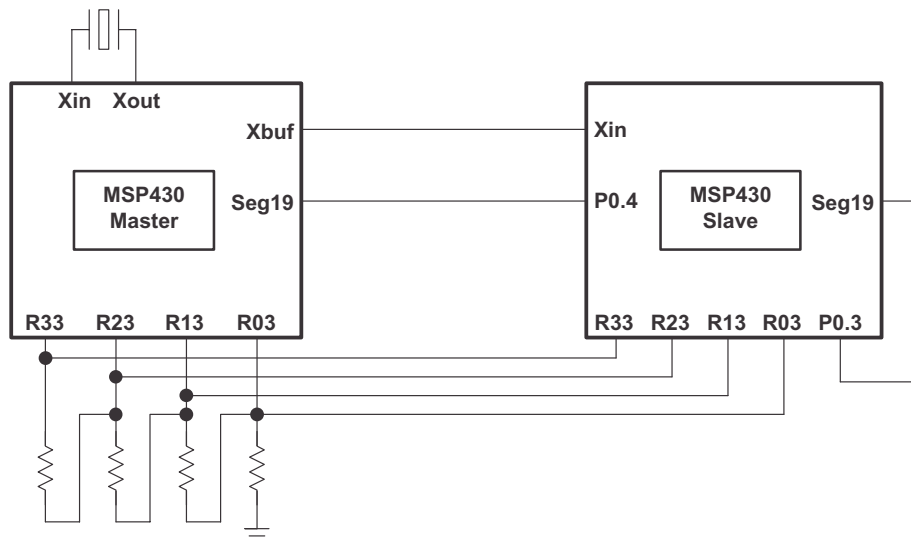


Figure 4. Interconnections Between the Two Controllers Necessary for LCD Expansion

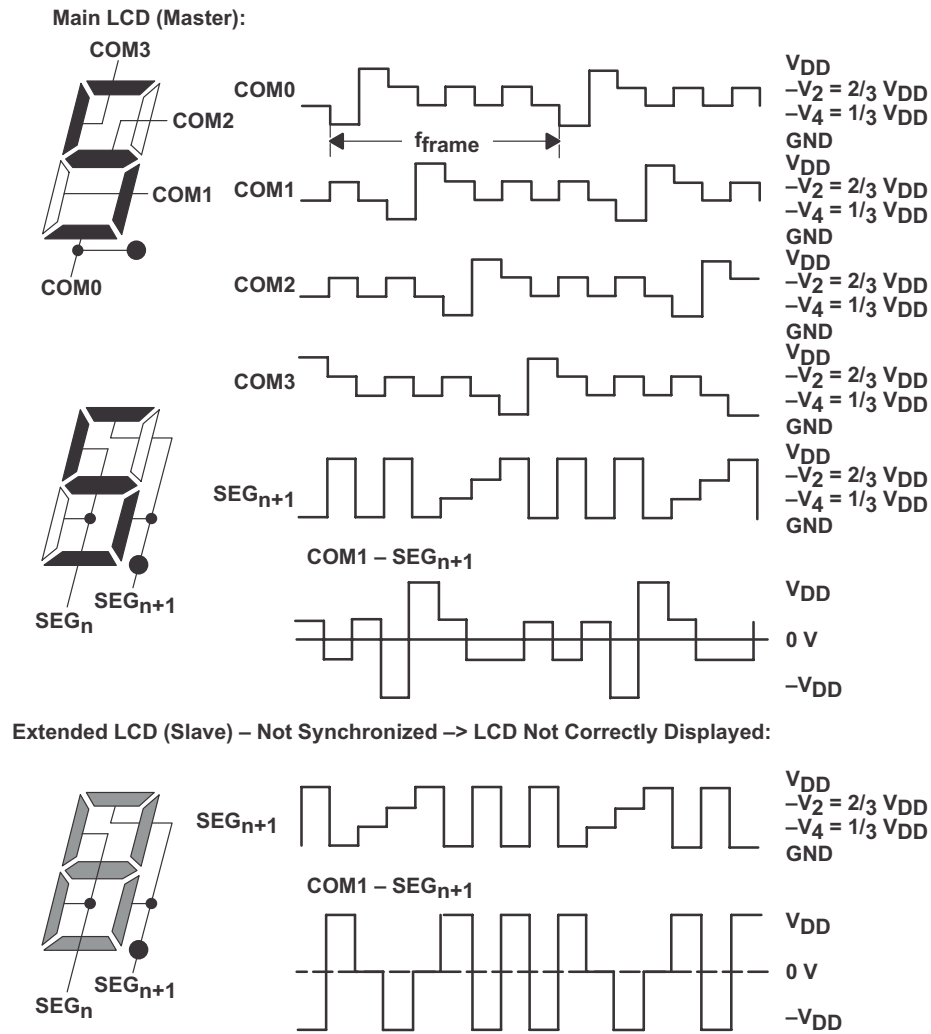
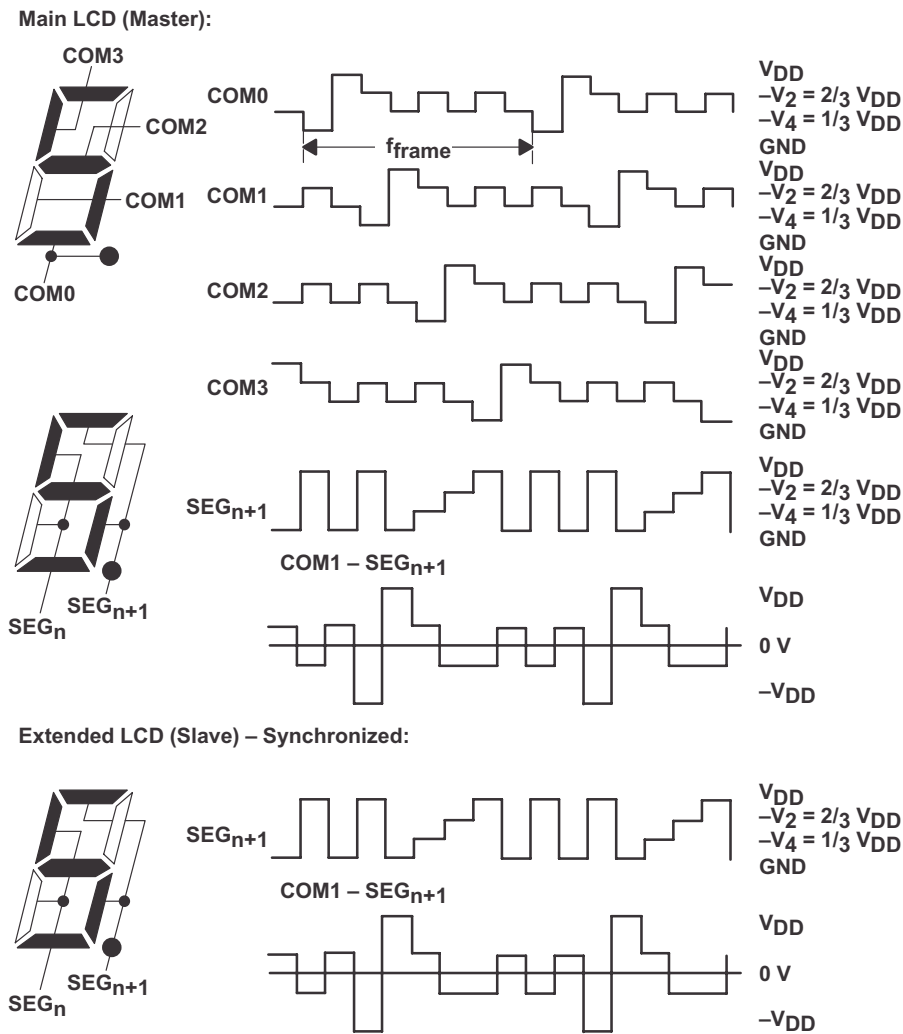
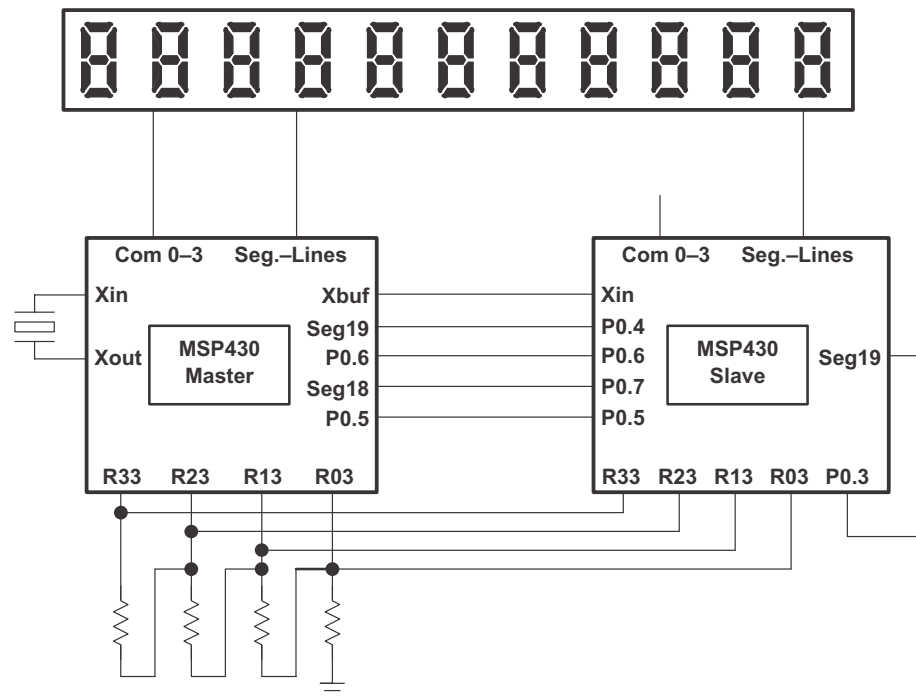


Figure 5. Timing for Asynchronous Operation



**Figure 6. Timing for Synchronous Operation**

The master and slave LCD timing generators are started, and the LSB of the nibble corresponding to the appropriate segment line is set in both MSP430s. Both LCD modules generate the signals as described before. Once the first rising edge of the slave's segment signal is detected, the slave stops its LCD timing generator. The slave waits for the rising edge on the master's segment signal. When this edge is detected, the slave starts its LCD timing generator again and the two LCD modules run synchronously.


**Figure 7. +Connecting a Large LCD to Two MSP430s**
**Table 2. Resources Required by the Controllers for LCD Expansion**

Master	Slave
Four segment Lines should be switched to output One is used for synchronization One is used for serial data transmission <sup>(1)</sup>	Four segment lines should be switched to output One is used for synchronization
Crystal is at XIN and XOUT XBUF should be switched to ACLK	No crystal is used ACLK source is XBUF from master
Common signals COMx should be connected to the LCD	Common signals COMx are unused
R33, R23, R13, and R03 should be connected to resistors	R23, R13, and R03 should be connected to pins of the master
	P0.3 and P0.4 are used for synchronization
P0.5 and P0.6 used for serial data transmission <sup>(1)</sup>	P0.5, P0.6, and P0.7 are used for serial data transmission <sup>(1)</sup>

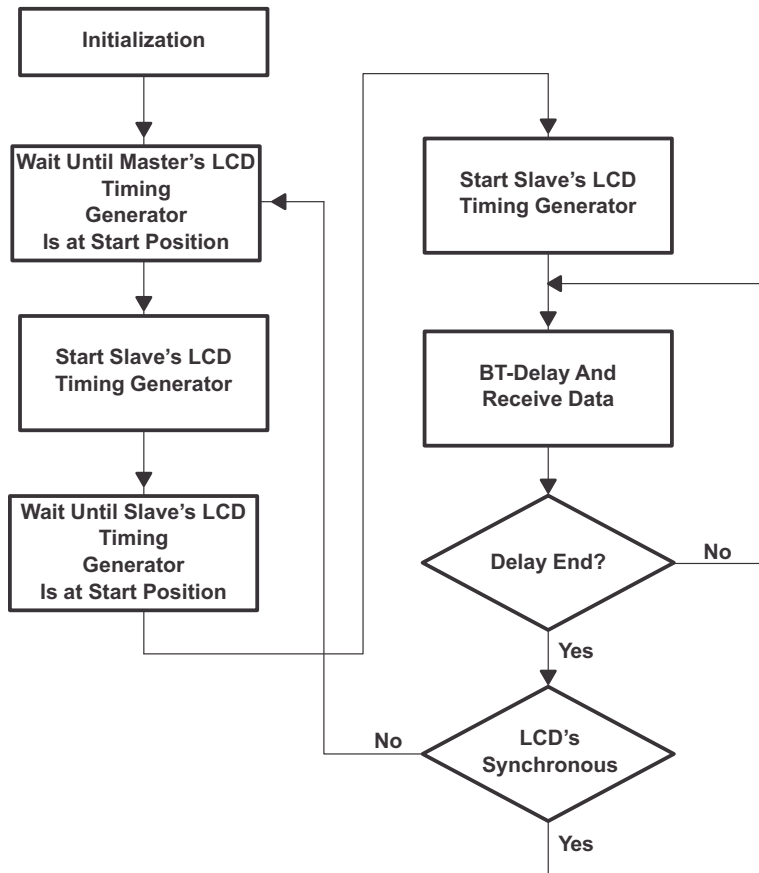
<sup>(1)</sup> Different communication methods can be implemented to lower the use of MSP430 resources.

## 5 Program Flow

Two MSP430EVK320s are used (MSP430P325) in the example described in this application note.

- Master
  - Segments 18 to 21 should be switched to output (initialization)
  - Wait until ACLK is running
  - ACLK should be switched to XBUF (initialization)
  - Transmit data to slave if needed
- Slave
  - Segments 18 to 21 should be switched to output (initialization)
  - Receive data from master if needed, or prepare own data to display
  - Synchronize LCD timing generator with master LCD timing generator
  - After a delay, the slave checks if the timing is still synchronous. A synchronization cycle is started again if master and slave are asynchronous.





**Figure 8. Connecting a Large LCD to Two MSP430s**

## Appendix A Listings

### A.1 *Master.asm*

```

; *****
; *** LCD demo software for LCD-Expansion.asm software
; *** This software shows the functionality equal to the EVK/STK
; *** Copyright Texas Instruments Inc. 1999
; *****
;*** Set this variable to '1' for the use on the Simulator ***
SIM      .set    0          ; 1 = Simulator
                                ; 0 = STK/EVK

SP_orig  .set    003DEh     ; stackpointer
lcd_req  .set    001h       ; lcd request
rx_req   .set    002h       ; Rx request
        .if SIM = 0
USER_END .set    003FFh     ; RAM endaddress
PRG_orig .set    00240h     ; Program Memory startaddress
lcd_ival .set    080h*128   ; 128*1/2 lcd shift interval 1/2 sec for STK
        .else
USER_END .set    0FFFFh     ; Memory endaddress
PRG_orig .set    00C000h    ; Program Memory startaddress
lcd_ival .set    1*128      ; 1*1/2 lcd shift interval 1/2 sec for Sim
        .endif
;--- RAM allocation
        .bss  runoff,1,220h ; runoff control flag register
        .bss  dummy,1      ;
        .bss  lcd_timer,2  ; lcd interval timer
        .bss  txt_ori,2    ; pointer for LCD text tables origin
        .bss  txt_idx,2    ; pointer for LCD text tables index
        .bss  lcd_ext,14   ; memory for extended LCD
Adr        .set    R11      ; Pointer for LCD-Address
        .bss  Data, 1
        .bss  SCount, 1
DataSize  .set    08
;--- Control register definitions
IE1        .equ    0h
IE2        .equ    01h
IFG1       .equ    02h
IFG2       .equ    03h
ME1        .equ    04h
ME2        .equ    05h
;--- Port register definitions
P0IN       .equ    010h
P0OUT      .equ    011h
P0DIR      .equ    012h
P0IFG      .equ    013h
P0IES      .equ    014h
P0IE       .equ    015h
P0IN_0     .set    001h
P0IN_6     .set    040h
P0OUT_5    .set    020h
P0DIR_5    .set    020h
P0OUT_6    .set    040h
P0DIR_6    .set    040h
P0IFG_6    .set    040h
P0IE_6     .set    040h
P0IES_6    .set    040h
LCDCTL     .equ    030h
LCD1       .equ    031h
BTME       .set    080h      ; BT module enable
BTIE       .set    080h      ; BT intrpt enable
BTIFG      .equ    080h      ; BT intrpt flag
P0IE0      .set    004h      ; P0.0 intrpt enable
P0IFG0     .set    004h      ; P0.0 intrpt flag

```

```

PODIR0    .set    001h        ; P0.0 direction
POIES0    .set    001h        ; P0.0 edge select
BTCTL     .equ    040h        ; BT control
TCCTL     .equ    042h        ; Address of Timer/Counter control register
TCPLD     .equ    043h        ; Address of Timer/Counter pre-load register
TCDAT     .equ    044h        ; Address of Timer/Counter
SCF10     .equ    050h
SCF11     .equ    051h
SCFQCTL   .equ    052h
AIN        .equ    0110h
AEN        .equ    0112h
ACTL       .equ    0114h
ADAT       .equ    0118h
ADIFG     .equ    04h
WDTCTL    .equ    0120h
WDTHold   .equ    80h
WDT_wrkey .equ    05A00h
CBCTL     .equ    053h
GIE        .equ    08h
;*****
; Reset : Initialize processor
;*****
        .sect "MAIN",PRG_orig
RESET
        MOV     #SP_orig,SP                ; initialize stackpointer
        MOV     #(WDTHold+WDT_wrkey),&WDTCTL ; Stop Watchdog Timer
;--- Clear Special Function Registers
        MOV.B   #08h,IE1 ; ! Monitor !
        CLR.B   IE2
        CLR.B   IFG1
        CLR.B   IFG2
;--- Prepare LCD, Basic timer and XBUF
        Mov.b   #01h,&CBCTL                ; ACLK at XBUF
        CALL   #show_clr                  ; clear LCD
        mov.b   #010h,&LCDCTL+10          ; Seg19 = 0001
        MOV.B   #09Fh,&LCDCTL             ; LCD : Analog generator on
                                                ; Low impedance of AG
                                                ; 4Mux active
                                                ; Group5 = dig.Output
        MOV.B   #050h,&BTCTL              ; Basic Timer: SSEL=0 DIV=0 Reset=1
                                                ; ACLK
                                                ; 32768/2 = 128*128Hz (7.8ms/128)
                                                ; LCD frame frequency @4Mux: 64Hz
        BIC.B   #040h,&BTCTL              ; Basic Timer reset disabled
        BIC.B   #BTIFG,&IFG2              ; clear basic timer intrpt flag
WaitACLK
        BIT.B   #BTIFG,&IFG2              ; test basic timer intrpt flag
        JZ     WaitACLK                   ; wait till ACLK is running
        BIC.B   #BTIFG,&IFG2              ; clear basic timer intrpt flag
        BIS.B   #BTIE,&IE2                ; enable basic timer intrpt
        MOV     #lcd_ival,lcd_timer       ; load SW lcd timer
        CALL   #init_ser                  ; prep for serial transmit
;*****
; Mainloop
;*****
mainloop
        CALL   #init_txt_MSP              ; initialize shift text
;--- idle loop : check runoff control bits
        EINT
idle
        BIT.B   #lcd_req,runoff           ; time slice over ?
        JZ     testAck
        BIC.B   #lcd_req,runoff           ; yes : clear control bit and execute
        CALL   #shift_txt                 ; shift text 1 digit left
testAck
        BIT.B   #rx_req,runoff            ; rx time slice over ?

```

```

        JZ     idle
        BIC.B  #rx_req,runoff      ; yes : clear control bit and execute
        BIT.B  #P0IFG_6,&P0IFG    ; test if ACK
        JZ     idle
        CALL  #ser_tx              ; transmit next bit
        jmp   idle

;-----
; subroutines for serial transmit
;-----
init_ser
        bic.b  #P0OUT_6,&P0OUT      ;set output
        bis.b  #P0DIR_5+ P0DIR_6,&P0DIR ;switch to output
        bis.b  #P0OUT_5,&P0OUT      ;set strobe
        mov   #lcd_ext,Adr          ;Adr.-Pointer
        mov.b @Adr+,Data            ;init Transmit Register
        clr.b SCount               ;Counter for transmitted Bits
        ret

;*****
;transmit serial Data
;*****
ser_tx
        bic.b  #P0OUT_5,&P0OUT      ;clear first bit flag
;Byte Transmitted?
        cmp.b  #DataSize,Scount     ;hole Data received ?
        jne   nextBit               ;No -> jump
        mov.b @Adr+,Data            ;get Data
        clr.b SCount                ; Counter for received Bits
        cmp   #digit-digit1+lcd_ext+1,Adr ; End of LCD Memory?
        jne   nextBit               ; No -> jump
        mov   #lcd_ext,Adr          ; Init Address-Pointer
        mov.b @Adr+,Data            ; get Data
        bis.b  #P0OUT_5,&P0OUT      ; set first bit flag
nextBit
        rra.b  Data                  ; next bit to output
        jnc   ser_tx1
        bis.b  #0Fh,&LCDCTL+0Ah      ; Seg18 = Data output -> 1
        jmp   ser_tx2
ser_tx1
        bic.b  #0Fh,&LCDCTL+0Ah      ; Seg18 = Data output -> 0
ser_tx2
        inc.b  SCount
        bis.b  #P0OUT_6,&P0OUT      ; send Strobe
        bic.b  #P0OUT_6,&P0OUT      ; idle
        ret

;-----
; subroutines : show text or values on LCD
;-----
txt_MSP .byte  cdp, cdp, cdp, cdp, cdp, cdp
        .byte  cM, cS, cP, c4, c3, c0, 0
        ;":::::MSP430",0           ; ':' = ' '(blank)
txt_MSPe
        .even
digit1  .set  7                      ; digits at first display
digit   .set  14                     ; digits at both displays
;--- initialize pointers for shift text
init_txt_MSP
        MOV   #txt_MSP,txt_ori        ; load text table pointer
        MOV   txt_ori,txt_idx        ; load text table index and fall into
;--- shift variable text from texttable over LCD (uses RAM txt_ori, txt_idx)
shift_txt
        PUSH  r5                      ; save R5
        PUSH  r6                      ; save R6
        MOV   #digit,r6               ; init digit pointer (outward loop)
        MOV   txt_idx,r5              ; load index pointer and
        CMP   #txt_MSPe,r5            ; check if data end
        JNE   shift_txt2

```

```

        MOV     txt_ori,txt_idx           ; yes : set index to origin of string
shift_txt1
        CMP     #txt_MSPE,r5             ; check if data end
        JNE     shift_txt2
        MOV     txt_ori,r5               ; yes : set pointer to origin of str
shift_txt2
        MOV.b   @r5+,r7                  ; move char to LCD (tab autoinc.)
        CMP     #digit1+1,R6
        JHS     shift_lcd2              ; select for first or second LCD
        MOV.b   r7,LCD1-1(r6)
        jmp     shift_txt3
shift_lcd2
        MOV.    b r7,lcd_ext-digit1-1(r6)
shift_txt3
        DEC     r6                       ; increment digit pointer
        JNZ     shift_txt1              ; loop if LCD isn't set completely
        INC     txt_idx                  ; increment index pointer and exit
        POP     r6                       ; restore R6
        POP     r5                       ; restore R5
        RET
;--- clear LCD
show_clr
        MOV     #9,r5                    ; clear display memory
show_clr1
        MOV.b   #0,LCD1-1(r5)
        DEC     r5
        JNZ     show_clr1
        RET
; * * * * *
; Basic Timer Interrupt routine
; * * * * *
Int_BT   ; Basic Timer 128Hz*128 (7.8ms/128)
        BIS.B  #rx_req,runoff           ; set RX request control bit
        DEC    lcd_timer                 ; decrement SW lcd-timer
        JNZ    Int_BT_end                ; !0 : no action
        MOV    #lcd_ival,lcd_timer       ; =0 : load again and
        BIS.B  #lcd_req,runoff           ; set lcd request control bit
Int_BT_end
; all other interrupts
Int_P0_0 ; P0.0
Int_P0_1 ; P0.1 (SW UART)
Int_WDT_T ; Watchdog / Timer
Int_ADC ; ADC
Int_P_27 ; Port0, bits 2..7
        RETI
; * * * * *
; LCD Definitions
; * * * * *
LCD_TYPE
;STK/EVK LCD
a        .equ  01h
b        .equ  02h
c        .equ  10h
d        .equ  04h
e        .equ  80h
f        .equ  20h
g        .equ  08h
h        .equ  40h
;--- character definitions
;LCD_Tab
c0       .equ  a+b+c+d+e+f              ; displays "0"
c1       .equ  b+c                       ; displays "1"
c2       .equ  a+b+d+e+g                 ; displays "2"
c3       .equ  a+b+c+d+g                 ; displays "3"
c4       .equ  b+c+f+g                   ; displays "4"
c5       .equ  a+c+d+f+g                 ; displays "5"

```

```

c6      .equ    a+c+d+e+f+g      ; displays "6"
c7      .equ    a+b+c            ; displays "7"
c8      .equ    a+b+c+d+e+f+g   ; displays "8"
c9      .equ    a+b+c+d+f+g     ; displays "9"
cdp     .equ    0                ; displays ":" blank
csc     .equ    g                ; displays ";" -
cle     .equ    a+d+e+f         ; displays "<" [
ceq     .equ    d+g             ; displays "=" ]
cge     .equ    a+b+c+d         ; displays ">" ]
cqu     .equ    a+b+e+g         ; displays "?"
cat     .equ    a+b+d+e+f+g     ; displays "@"
cA      .equ    a+b+c+e+f+g     ; displays "A"
cB      .equ    c+d+e+f+g       ; displays "B" b
cC      .equ    a+d+e+f         ; displays "C"
cD      .equ    b+c+d+e+g       ; displays "D" d
cE      .equ    a+d+e+f+g       ; displays "E"
cF      .equ    a+e+f+g         ; displays "F"
cG      .equ    a+c+d+e+f+g     ; displays "G"
cH      .equ    b+c+e+f+g       ; displays "H"
cI      .equ    b+c             ; displays "I"
cJ      .equ    b+c+d+e         ; displays "J"
cK      .equ    0               ; displays "K"
cL      .equ    d+e+f           ; displays "L"
cM      .equ    a+b+c+e+f       ; displays "M"
cN      .equ    c+e+g           ; displays "N" n
cO      .equ    c+d+e+g         ; displays "O" o
cP      .equ    a+b+e+f+g       ; displays "P"
cQ      .equ    0               ; displays "Q"
cR      .equ    e+g             ; displays "R" r
cS      .equ    a+c+d+f+g       ; displays "S"
cT      .equ    d+e+f+g         ; displays "T" t
cU      .equ    c+d+e           ; displays "U" u
cV      .equ    0               ; displays "V"
cW      .equ    0               ; displays "W"
cX      .equ    0               ; displays "X"
cY      .equ    b+c+d+f+g       ; displays "Y"
cZ      .equ    a+b+d+e+g       ; displays "Z" 2
;LCD_Tab_End
        .even    ; succeeding .text sections must be even aligned!
; * * * * *
; Interrupt vectors
; * * * * *
        .sect    "Int_Vect",USER_END-31
        .word    Int_P_27        ; Port0, bit 2 to bit 7
        .word    Int_BT          ; Basic Timer
        .word    RESET           ; no source
        .word    RESET           ; no source
        .word    RESET           ; no source
        .word    Int_ADC         ; EOC from ADC
        .word    RESET           ; no source
        .word    RESET           ; no source
        .word    RESET           ; no source
        .word    RESET           ; no source
        .word    Int_WDT_T       ; Watchdog/Timer, Timer mode
        .word    RESET           ; no source
        .word    Int_P0_1        ; Address of UART handler
        .word    Int_P0_0        ; P0.0
        .word    RESET           ; NMI, Osc. fault
        .word    RESET           ; POR, ext. Reset, Watchdog
        .sect    "PW",USER_END-33
        .word    #0AA55h        ; Password for EVK autorun

```

## A.2 Slave.asm

```

;*****
; LCD - EXPANSION -- SLAVE --
;*****
; Assembler-Program to synchronize a second MSP430 (Slave) to a
; first (Master) e.g. to synchronize Common and Segment lines of
; both controllers.
;
; The Master have generate the Clock at the XBUF for the Slave's XIN.
; The Seg20 of the Master and the Slave are switched to digital
; output and via port pins scanned.
; After a rising edge of the Slave's Seg20 the fLCD of the Slave
; will be stopped.
; If now a rising edge of the Master's Seg 20 detected the Frequency
; for the LCD (fLCD) is started again.
; Now both LCD-Drivers are running synchronously.
; After a delay of 2 seconds this state is controlled and if
; necessary a new synchronization will be started.
_CPU_      .set      3          ; ID32X
           .include  "STD_DEF.ASM" ; file is included with the
                                           ; Simulator for the MSP430
sim        .set      0          ; 1 : Simulator
                                           ; 0 : EVK/STK

           .if SIM = 0
USER_END   .set      003FFh     ; RAM end address
PRG_orig   .set      00240h     ; Program Memory start address
           .else
USER_END   .set      0FFFFh     ; Memory end address
PRG_orig   .set      00C000h    ; Program Memory start address
           .endif

STACK     .set      3DEh       ;start of system stack
STATUS    .set      R10
Adr        .set      R11       ;Pointer for LCD-Address
digit2    .set      7          ;number of digits at 2. LCD
DataSize  .set      08
           .bss      Data, 1,220h
           .bss      SCount, 1

;**** Initialization:
           .sect "Main", PRG_orig

Start
           Mov       #STACK,SP          ; initialize system stack pointer
           Clr       STATUS
           mov       #WDTPW+WDTHOLD,&WDTCTL ; WDT is on hold

; clear all interrupt enable
           mov.b     #08h,&IE1          ; ! Monitor !
           clr.b     &IE2
           clr       Adr
           CALL      #show_clr          ; clear LCD
           MOV.B    #09Fh,&LCDCTL       ; LCD : Analog generator on
                                           ; Low impedance of AG
                                           ; 4Mux active
                                           ; Group5 = dig.Output
                                           ; seg 18 - 21
           mov.b     #010h,&LCDCTL+10   ; Seg19 = 0001
           MOV.B    #055h,&BTCTL       ; Basic Timer : SSEL=0 DIV=0 Reset=1
                                           ; ACLK
                                           ; 32768/256 = 128Hz (7.8ms)
                                           ; LCD frame frequency @4Mux: 64Hz
           BIC.B    #040h,&BTCTL       ; Basic Timer reset disabled
           clr.b     &IFG1             ; clear all interrupt flags
           clr.b     &IFG2

                                           ; init LCD and Serial receive
           bic.b     #P0DIR_3+P0DIR_4+P0DIR_5+P0DIR_6+P0DIR_7,&P0DIR ;reset state
                                           ; P0.3 , P0.5, P0.6 , P0.7 and P0.4 = input
           bic.B    #P0IES_3+P0IES_4,&P0IES ; edge select =low/high
           bic.b     #P0IFG_3+P0IFG_4,&P0IFG ; clear int. flag

```

```

        bis.b  #P0IE_3,&P0IE          ; enable intrr. P0_bit 3
        eint                                     ; general interrupt enable
                                           ; init for serial receive of data
        mov   #LCDCTL+1,Adr             ; Adr.-Pointer
        clr.b SCount                   ; Counter for received Bits
;Part for receiving serial data
;*****
waitData
        bit.b  #P0IFG_6,&P0IFG        ; wait till flag
        jz    waitData
;Bit to data register
GetBit
FrameStart
        bit.b  #P0IN_5,&P0IN          ; start of frame?
        jz    GetBit1                ; input low -> jump
        mov   #LCDCTL+1,Adr             ; Adr.-Pointer to start
        clr.b SCount                   ; Counter for received Bits
GetBit1
        bit.b  #P0IN_7,&P0IN          ; test input
        rrc.b Data                     ; shift bit into Data
        inc.b  SCount
        cmp.b  DataSize,Scount         ; hole Data received ?
        jne   Ack                     ; No -> jump
        mov.b  Data,0(Adr)             ; write Data
        clr.b  SCount                 ; Counter for received Bits
        inc   Adr                     ; inc Adr.-Pointer
        cmp   #(LCDCTL+1+digit2+1),Adr ; End of LCD Memory?
        jne   Ack
        mov   #LCDCTL+1,Adr           ; Adr.-Pointer to start
Ack
        jmp   waitData                ; jmp idle loop
;*****
;**** Interrupt Service Routine for interrupt caused by P0IFG.27:
P0INT27
        push  R9                      ; SAVE R9 on Stack
        cmp  #01h,STATUS              ; 1 = Wait to Slave Seg19 edge
        jeq  RUN_LCD1
        cmp  #02h,STATUS              ; compare Status
        jeq  TESTBT1                  ; 2 = Basic Timer Intr.
        inc  STATUS                   ; Status = 0
        bis.b #BTHOLD+BTDIV,&BTCTL    ; stop LCD (flcd)
        bic.b #P0IE_3,&P0IE          ; disable intrr. P0_bit 3
        bis.b #P0IE_4,&P0IE          ; enable intrr. P0_bit 4
        jmp  END
RUN_LCD1
        bic.b #BTHOLD,&BTCTL          ; run BTCNT1 (flcd)
        bic.b #P0IE_4,&P0IE          ; disable P0IE_4
        inc  STATUS                   ; Seg19 Low/High (Slave)
        bis.b #07h,&BTCTL             ; BT interval = 2.0 sec.
        jmp  END1
;-----
TESTBT1
        mov.b  &BTCNT1,R9             ; save BTCNT1 to R9
        bic.b  #080h,R9               ; maskiere BTCNT1 Q0-Q6
        cmp.b  #040h,R9               ; flcd = ACLK/128
        jne   NOTSYNC                ; jmp if not synchron
        bit.b  #P0IN_3,&P0IN          ; If Seg20 of Slave = high
        jz   NOTSYNC
        bic.b  #P0IE_4,&P0IE          ; disable P0IE_4
END1
        clr.b  &BTCNT2                ; Clear BTCNT2(Counter for Interr.)
        bic.b  #BTIFG,&IFG2           ; Clear BTIFG
        bis.b  #BTIE,&IE2             ; enable BTIE
        jmp  END
NOTSYNC
        bic.b  #P0IFG_3+P0IFG_4,&P0IFG ; Clear Intrr. Flag P0.3+4
        bis.b  #P0IE_3,&P0IE          ; enable P0IE_3
        bic.b  #BTIE,&IE2             ; disable BTIE
        clr   STATUS
END
        bic.b  #P0IFG_3+P0IFG_4,&P0IFG ; clear int. flag 3 and 4

```



```

        pop     R9                ; restore R9
        reti

;*****
;--- clear LCD
show_clr
        MOV     #9,r5            ; clear display memory
show_clr1
        MOV.b  #0,LCD1-1(r5)
        DEC    r5
        JNZ    show_clr1
        RET

;*****
;**** Interrupt Service Routine for interrupt caused by BTIFG:
BTINT
        bic.b  #P0IFG_3+P0IFG_4,&P0IFG ; Clear Interrupt Flag P0.3+4
        bis.b  #P0IE_4,&P0IE           ; enable intrr. P0_bit 4 (Master)
        bic.b  #BTIE,&IE2             ; Disable BT Interrupt
NOINT   reti                    ; return from interrupt
;**** Interrupt Vector Addresses:
        .sect "Int_Vect", USER_END-31 ;PUC/reset address
        .word  P0INT27              ; P0IFG.27
        .word  BTINT                ; BTIFG
        .word  NOINT
        .word  NOINT
        .word  NOINT
        .word  NOINT                ; ADCIFG
        .word  NOINT
        .word  NOINT
        .word  NOINT
        .word  NOINT
        .word  NOINT                ; WDTIFG
        .word  NOINT
        .word  NOINT                ; P0IFG.1
        .word  NOINT                ; P0IFG.0
        .word  NOINT                ; RSTI/OFIFG
        .word  Start                ; PUC/Reset
        .sect "PW",USER_END-33
        .word  #0AA55h              ; Password for EVK autorun

```

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions of TI 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)