ЕМИТЕР 6/2013 - Изработка на работна околина и проект во IAR развојната апликација (2)

Додаток 2: Програма за генерирање прекин во секоја секунда, напишана во асемблер.

1. asm.s

```
        NAME    main

 #include "VIC_Addr.h"
 #include "TIMER0_Addr.h"
 #include "UART1.h"

        PUBLIC  __iar_program_start
        PUBLIC  __vector
        PUBLIC  __vector_0x14

        SECTION IRQ_STACK:DATA:NOROOT(3)
        SECTION CSTACK:DATA:NOROOT(3)

        SECTION .intvec : CODE (2)
        CODE32
        ARM
        ARM

__vector:
        ldr     pc, =__iar_program_start
        ldr     pc, =UndefinedInsHandler
        ldr     pc, =SoftwareInt
        ldr     pc, =PrefetchAbort
        ldr     pc, =DataAbort
__vector_0x14:
        DC32    0                               ; - Reserved, used by NXP MCU's for storage of
CRC.
        ldr      pc, [pc, #-0xFF0]              ; - IRQ Handler (reads vector address from
VicVecAddr)
;         ldr      pc, IRQ_ISR_Address          ; - If non Vectored features of VIC are used
this is the way how ISR is called.
        ldr     pc, =FIQHandler

UndefinedInsHandler
        b     UndefinedInsHandler
SoftwareInt
        b     SoftwareInt
PrefetchAbort
        b     PrefetchAbort
DataAbort
        b     DataAbort
IRQ_ISR_Address
        DCD   IRQ_ISR_Function
FIQHandler
        b     FIQHandler


        SECTION .text : CODE (2)
        CODE32
__iar_program_start
main
;
; Add initialization needed before setup of stackpointers here.
;
; LPC2148 Errata
; Date: August 5, 2005
; Document Release: Version 1.0
; Device Affected: LPC2148
; Incorrect read of data from SRAM after Reset and MAM is not enabled or partially enabled
MAM.1
; Init MAM before acsses to SRAM
MAMCR    DEFINE 0xE01FC000                  ; MAM Control Register
MAMTIM   DEFINE 0xE01FC004                  ; MAM Timing register
        ldr   r0,=MAMCR
        ldr   r1,=MAMTIM
        ldr   r2,=0
```

```
        str   r2,[r0]
        ldr   r2,=7
        str   r2,[r1]
        ldr   r2,=2
        str   r2,[r0]

; Initialize the stack pointers.
; The pattern below can be used for any of the exception stacks:
; FIQ, IRQ, SVC, ABT, UND, SYS.
; The USR mode uses the same stack as SYS.
; The stack segments must be defined in the linker command file,
; and be declared above.
;
; --------------------
; Mode, correspords to bits 0-5 in CPSR
MODE_MSK DEFINE 0x1F                        ; Bit mask for mode bits in CPSR

USR_MODE DEFINE 0x10                        ; User mode
FIQ_MODE DEFINE 0x11                        ; Fast Interrupt Request mode
IRQ_MODE DEFINE 0x12                        ; Interrupt Request mode
SVC_MODE DEFINE 0x13                        ; Supervisor mode
ABT_MODE DEFINE 0x17                        ; Abort mode
UND_MODE DEFINE 0x1B                        ; Undefined Instruction mode
SYS_MODE DEFINE 0x1F                        ; System mode

        mrs   r0,cpsr                       ; Original PSR value
        bic   r0,r0,#MODE_MSK               ; Clear the mode bits
        orr   r0,r0,#IRQ_MODE               ; Set IRQ mode bits
        msr   cpsr_c,r0                     ; Change the mode
        ldr   sp,=SFE(IRQ_STACK)            ; End of IRQ_STACK

        bic   r0,r0,#MODE_MSK               ; Clear the mode bits
        orr   r0,r0,#SYS_MODE               ; Set System mode bits
        msr   cpsr_c,r0                     ; Change the mode
        ldr   sp,=SFE(CSTACK)               ; End of CSTACK
// Initialize memory remapping to SRAM
        ldr   r0, MEMMAP
        mov   r1, #10b
        str    r1,  [r0]                                ; - Remap exception vectors to SRAM
(0x40000000). This way vector 0x00000008 will be remaped to 0x40000008
MEMMAP
        DCD   0xE01FC040

// ====================
// Initialize UART1 controller
// ====================
PINSEL0 EQU 0xE002C000
// Enable access to devision latch
        ldr   r0, =U1LCR
        mov   r1, #10000000b
        str   r1, [r0]
// Set devision value
// Devisor = PCLK / (16*BaudRate); PCLK = SYS_CLK/4 = 3000000; BaudRate = 9600; Devisor =
19.53125
        ldr   r0, =U1DLL          ; Devisor LSB
        mov   r1, #19
        str   r1, [r0]
        ldr   r0, =U1DLM          ; Devisor MSB
        mov   r1, #0
        str   r1, [r0]
// Enable FIFO's
        ldr   r0, =U1FCR          ; Enable FIFO's
        mov   r1, #0
        str   r1, [r0]
// Configure transmission parameters:
        ldr   r0, =U1LCR
        mov   r1, #00000011b      ; - DLAB          = 0  (disable)
                                  ;   BreakControl  = 0  (disable)
                                  ;   ParitySelect  = 00 (N/A)
                                  ;   ParityEnable  = 0  (disable)
                                  ;   StopBitSelect = 0  (1 stop bits)
                                  ;   WordLength    = 11 (8 bit character length)
        str   r1, [r0]
// Select Rx/Tx functionality for peripheral I/O pins
        ldr   r0, =PINSEL0
//                1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
//                5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
        mov   r1, #00000000000001010000000000000000b
        str   r1, [r0]
// ====================
```

```
// Initialize Vectored Interrupt Controller
// ===================
// Assign all interrupt channels to IRQ
// VICIntSelect  =  0;
        ldr   r0, =VICIntSelect ;
        mov   r1, #0
        str   r1, [r0]
// Disable all interrupts
// VICIntEnClear = 0xFFFFFFFF;
        ldr   r0, =VICIntEnClear ;
        mov   r1, #0xFFFFFFFF
        str   r1, [r0]
// Clear all software interrupts
// VICSoftIntClear = 0xFFFFFFFF;
        ldr   r0, =VICSoftIntClear ;
        mov   r1, #0xFFFFFFFF
        str   r1, [r0]
// VIC registers can be accessed in User or privileged mode
// VICProtection = 0;
        ldr   r0, =VICProtection ;
        mov   r1, #0
        str   r1, [r0]
// Clear interrupt
// VICVectAddr = 0;
        ldr   r0, =VICVectAddr ;
        mov   r1, #0
        str   r1, [r0]

// Clear address of the Interrupt Service routine (ISR) for non-vectored IRQs.
// VICDefVectAddr = 0;
        ldr   r0, =VICDefVectAddr ;
        mov   r1, #0
        str   r1, [r0]
// Clear address of the Interrupt Service routine (ISR) for vectored IRQs.
// VICVectAddr0  = 0
        ldr   r0, =VICVectAddr0 ;
        mov   r1, #0
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr1
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr2
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr3
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr4
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr5
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr6
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr7
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr8
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr9
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr10
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr11
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr12
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr13
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr14
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectAddr15
        str   r1, [r0]          ; - addr(r1) = 0
// Disable all vectored IRQ slots
// VICVectCntl0..15  = 0
        ldr   r0, =VICVectCntl0 ;
        mov   r1, #0
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl1
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl2
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl3
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl4
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl5
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl6
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl7
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl8
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl9
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl10
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl11
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl12
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl13
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl14
        str   r1, [r0], #4      ; - addr(r1) = 0, r0=r0+4, r1 = VICVectCntl15
        str   r1, [r0]          ; - addr(r1) = 0
// Assign slot 0 to TIMER0 interrupts
        ldr   r0, =VICVectCntl0
        ldr   r1, =(VIC_TIMER0|0x20)
        str   r1, [r0]
// Set address of TIMER0 interrupt routine.
        ldr   r0, =VICVectAddr0
        ldr   r1, =TimerISR
```

```
                str   r1, [r0]
        // Enable interrupts from slot 0
                ldr   r0, =VICIntEnable
                mov   r1, #(1<<VIC_TIMER0)
                str   r1, [r0]

        // ===================
        // Configure TIMER0
        // ===================
        // Disable counting
        // T0TCR=0;
                ldr   r0, =T0TCR
                mov   r1, #0
                str   r1, [r0]
        // Clear interrupts flags
        // T0IR=0xFF;
                ldr   r0, =T0IR
                mov   r1, #0xFF
                str   r1, [r0]
        // Clear timer counter
        // T0TC=0;
                ldr   r0, =T0TC
                mov   r1, #0
                str   r1, [r0]
        // Set Prescaler Value
        // T0PR = Timer0Config.Prescaler - 1;
                ldr   r0, =T0PR
                mov   r1, #1000
                str   r1, [r0]
        // Clear prescaler timer counter
        // T0PC=0;
                ldr   r0, =T0PC
                mov   r1, #0
                str   r1, [r0]
        // Set Interrupt on match
                ldr   r0, =T0MCR
                mov   r1, #11b
                str   r1, [r0]
        // Init Compare modules
                ldr    r0, =T0MR0             ; - Set compare value for module 0 to 30000. When TC
reaches this value IRQ will be generated.
                ldr   r1, =3000
                str   r1, [r0]

                ldr   r0, =T0MR1          ; - Set compare value for module 1 to 0.
                mov   r1, #0
                str   r1, [r0]

                ldr   r0, =T0MR2          ; - Set compare value for module 2 to 0.
                mov   r1, #0
                str   r1, [r0]

                ldr   r0, =T0MR3          ; - Set compare value for module 3 to 0.
                mov   r1, #0
                str   r1, [r0]
        // Reset Capture modules
        // T0CCR=0;
                ldr   r0, =T0CCR
                mov   r1, #0
                str   r1, [r0]
        // Reset External Compare module
        // T0EMR=0;
                ldr   r0, =T0EMR
                mov   r1, #0
                str   r1, [r0]
        // Enable IRQ in CPU
                mrs   r0, CPSR
                bic   r0, r0, #0x80
                msr   CPSR_cf, r0
        // Enable TIMER0 counting
                ldr   r0, =T0TCR
                mov   r1, #1
                str   r1, [r0]
        // Continue looping while TIMER0 generates IRQ reqest.
        MainLoop
                nop
                nop
                nop
                nop
                B MainLoop                    ; - Loop forever.
```

```
// ==========================================
// ==========================================
IRQ_ISR_Function
TimerISR
        stmdb r13!,{r0,r1}            ; - Save context (all used registers on IRQ stack)
// Clear interrupts flags (ACK that interrupt is processed  )
// T0IR=0xFF;
        ldr   r0, =T0IR
        mov   r1, #0xFF
        str   r1, [r0]
// Send character over UART1
        ldr   r1, Count
        add   r1, r1, #0x30           ; - Convert to ASCII
        ldr   r0, =U1THR
        str   r1, [r0]                ; - Send character over UART1

        ldr   r0, =U1THR
        mov   r1, #'\n'
        str   r1, [r0]

        ldr   r0, =U1THR
        mov   r1, #'\r'
        str   r1, [r0]

        ldr   r1, Count
        adds  r1, r1, #1
        cmp   r1, #10
        moveq r1, #0
        str   r1, Count
// ACK Interrupt
// VICVectAddr = 0;
        ldr   r0, =VICVectAddr
        mov   r1, #0
        str   r1, [r0]
// Restore content of registers and return from ISR
        ldmfd r13!,{r0,r1}            ; - return old values of registers
        subs   pc, lr, #4             ; - exit from IRQ context

// ISR Variable
Count
        DC32  0x0
// ==========================================
// ==========================================
        END
```

2. TIMER0_Addr.h

```
/*****************************************************************************
 **
 ** TIMER0
 **
 *****************************************************************************/
T0IR EQU 0xE0004000
T0TCR EQU 0xE0004004
T0TC EQU 0xE0004008
T0PR EQU 0xE000400C
T0PC EQU 0xE0004010
T0MCR EQU 0xE0004014
T0MR0 EQU 0xE0004018
T0MR1 EQU 0xE000401C
T0MR2 EQU 0xE0004020
T0MR3 EQU 0xE0004024
T0CCR EQU 0xE0004028
T0CR0 EQU 0xE000402C
T0CR1 EQU 0xE0004030
T0CR2 EQU 0xE0004034
T0CR3 EQU 0xE0004038
T0EMR EQU 0xE000403C
T0CTCR EQU 0xE0004070
```

3. UART1.h

```
U1FCR  EQU  0xE0010008
U1LCR  EQU  0xE001000C
U1MCR  EQU  0xE0010010
U1LSR  EQU  0xE0010014
U1MSR  EQU  0xE0010018
U1SCR  EQU  0xE001001C
```

```
U1ACR  EQU  0xE0010020
U1FDR  EQU  0xE0010028
U1TER  EQU  0xE0010030
U1THR  EQU  0xE0010000
U1RBR  EQU  0xE0010000
U1DLL  EQU  0xE0010000
U1DLM  EQU  0xE0010004
```

4. VIC_Addr.h

```
VICIRQStatus EQU     0xFFFFF000
VICFIQStatus EQU     0xFFFFF004
VICRawIntr EQU       0xFFFFF008
VICIntSelect EQU     0xFFFFF00C
VICIntEnable EQU     0xFFFFF010
VICIntEnClear EQU    0xFFFFF014
VICSoftInt EQU       0xFFFFF018
VICSoftIntClear EQU  0xFFFFF01C
VICProtection EQU    0xFFFFF020
VICVectAddr EQU      0xFFFFF030
VICDefVectAddr EQU   0xFFFFF034
VICVectAddr0 EQU     0xFFFFF100
VICVectAddr1 EQU     0xFFFFF104
VICVectAddr2 EQU     0xFFFFF108
VICVectAddr3 EQU     0xFFFFF10C
VICVectAddr4 EQU     0xFFFFF110
VICVectAddr5 EQU     0xFFFFF114
VICVectAddr6 EQU     0xFFFFF118
VICVectAddr7 EQU     0xFFFFF11C
VICVectAddr8 EQU     0xFFFFF120
VICVectAddr9 EQU     0xFFFFF124
VICVectAddr10 EQU    0xFFFFF128
VICVectAddr11 EQU    0xFFFFF12C
VICVectAddr12 EQU    0xFFFFF130
VICVectAddr13 EQU    0xFFFFF134
VICVectAddr14 EQU    0xFFFFF138
VICVectAddr15 EQU    0xFFFFF13C
VICVectCntl0 EQU     0xFFFFF200
VICVectCntl1 EQU     0xFFFFF204
VICVectCntl2 EQU     0xFFFFF208
VICVectCntl3 EQU     0xFFFFF20C
VICVectCntl4 EQU     0xFFFFF210
VICVectCntl5 EQU     0xFFFFF214
VICVectCntl6 EQU     0xFFFFF218
VICVectCntl7 EQU     0xFFFFF21C
VICVectCntl8 EQU     0xFFFFF220
VICVectCntl9 EQU     0xFFFFF224
VICVectCntl10 EQU    0xFFFFF228
VICVectCntl11 EQU    0xFFFFF22C
VICVectCntl12 EQU    0xFFFFF230
VICVectCntl13 EQU    0xFFFFF234
VICVectCntl14 EQU    0xFFFFF238
VICVectCntl15 EQU    0xFFFFF23C

/*****************************************************************************
 **
 **  VIC Interrupt channels
 **
 *****************************************************************************/
#define VIC_WDT          0  /* Watchdog                          */
#define VIC_SW           1  /* Software interrupts               */
#define VIC_DEBUGRX      2  /* Embedded ICE, DbgCommRx           */
#define VIC_DEBUGTX      3  /* Embedded ICE, DbgCommTx           */
#define VIC_TIMER0       4  /* Timer 0 (Match 0-3 Capture 0-3)   */
#define VIC_TIMER1       5  /* Timer 1 (Match 0-3 Capture 0-3)   */
#define VIC_UART0        6  /* UART 0  (RLS, THRE, RDA, CTI)     */
#define VIC_UART1        7  /* UART 1  (RLS, THRE, RDA, CTI, MSI) */
#define VIC_PWM0         8  /* PWM 0   (Match 0-6 Capture 0-3)   */
#define VIC_I2C          9  /* I2C 0   (SI)                      */
#define VIC_SPI         10  /* SPI 0   (SPIF, MODF)              */
#define VIC_SSP         11  /* SSP                               */
#define VIC_PLL         12  /* PLL lock (PLOCK)                  */
#define VIC_RTC         13  /* RTC      (RTCCIF, RTCALF)         */
#define VIC_EINT0       14  /* External interrupt 0 (EINT0)      */
#define VIC_EINT1       15  /* External interrupt 1 (EINT1)      */
#define VIC_EINT2       16  /* External interrupt 2 (EINT2)      */
#define VIC_EINT3       17  /* External interrupt 3 (EINT3)      */
#define VIC_AD0         18  /* A/D converter 0                   */
#define VIC_I2C1        19  /* I2C 1                             */
#define VIC_BOD         20  /* Brown out detect                  */
```

```
#define VIC_AD1         21  /* A/D converter 1                    */
#define VIC_USB         22  /* USB Low and High priority          */
```