```
;*******************************************************************************
;
;   Project:        Osmibitovy citac a vypocet kvadratickeho polynomu
;   Filename:       C8POL_11.asm
;   Date:           12.05.2010
;   File Version:   1.1
;   Charset:        ascii
;   Tab Stop:       4
;   Search Mark:    #>
;
;   Hardware:       Explorer PIC18F87J11
;   Tools:          IDE MPLAB v8.46
;
;   Author:         Petr Kozak
;   Company:        CVUT FEL
;
;   Status:         Released
;
;*******************************************************************************
;   Popis:
;   Program pro pripravek Explorer2 s procesorem PIC18F87J11 realizujici
;   osmibitovy citac s intervalem 600ms,jehoz vystup je zobrazen na LED portu
;   PORTD procesoru. Ovlivnovan stavem tlacitek S1 (prepinani citaci/stop) a S2
;   reset (nulovani) citace.
;   Vystupem, v promenne, je vypocet polymonu z prave aktualni hodnoty citace.

;*******************************************************************************
;#> BEGIN: Historie
;-------------------------------------------------------------------------------
;   - 12.05.2010 peko, v1.1
;   Zamezeni problikavani nejnizsiho bitu pri stisknutem tlac. RA5
;   (S2 - nulovani).
;
;   - 07.05.2010 peko, v1.0
;   Prvotni verze, vytvoreni.
;
;-------------------------------------------------------------------------------
;#< END: Historie
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Debug
;-------------------------------------------------------------------------------
; Promenna pro ladeni
;-------------------------------------------------------------------------------
;   #define DEBUG

;*******************************************************************************
;#> BEGIN: Konfigurace HW
;-------------------------------------------------------------------------------
    LIST P=18F87J11
    #include <P18F87J11.INC>

    CONFIG  FOSC  = HS          ;HS oscilator
    CONFIG  WDTEN = OFF         ;WATCHDOG zakazan
    CONFIG  XINST = OFF
;-------------------------------------------------------------------------------
;#< END: Konfigurace HW
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Promenne
;-------------------------------------------------------------------------------
    UDATA_ACS

    ;---------------------------------------------------------------------------
    ;#> BEGIN: Zasobnik kontext
    ;   Pomocne promenne pro ulozeni kontextu v preruseni pro Timer1
    ;---------------------------------------------------------------------------
        STACK_WREG          res 1
        STACK_STATUS        res 1
        STACK_BSR           res 1
```

```
    ;-------------------------------------------------------------------------
    ;#> BEGIN: Promenne pro polynom a citac
    ;   Vypocet polynomu a citac
    ;-------------------------------------------------------------------------
        VARIABLE_X_TMP      res 1   ; lokalni kopie promenne VARAIBLE_X
                                    ; pouzita v SolveResult
        VARIABLE_X          res 1   ; promenna citace x, ktera je zvetsovana
                                    ; v Timer0Interrupt (Timer0)

        VARIABLE_RESULT_TMP res 3   ; pomocna promenna pro vypocet polynomu
                                    ; pouzita v SolveResult
        VARIABLE_RESULT     res 3   ; vysledek vypoctu polynomu 24b
                                    ; CONST_K2 * VARIABLE_X * VARAIBLE_X +
                                    ; CONST_K1 * VARIABLE_X +
                                    ; CONST_K0

    ;-------------------------------------------------------------------------
    ;#> BEGIN: Button context
    ;   Debounce tlacitek RB0(S1) a RA5(S2) a jejich aktualni stav.
    ;-------------------------------------------------------------------------
        BUTTON_STATE        res 1   ; stav tlacitek, dle bitu
                                    ; CONST_BUTTON_BIT_RB0_S1
                                    ; CONST_BUTTON_BIT_RA5_S2
        BUTTON_RB0_S1       res 1   ; registr pro debounce RB0 (S1)
        BUTTON_RA5_S2       res 1   ; registr pro debounce RA5 (S2)

    ;-------------------------------------------------------------------------
    ;#> BEGIN: Switch FSA
    ;   Uchovani stavu automatu prepinace - citani/stop
    ;   Zakodovani po dvojic bitu dle CONST_SWITCH_FSA_BIT_(BWAIT|COUNT)
    ;-------------------------------------------------------------------------
        SWITCH_FSA_STATE    res 1

    ;-------------------------------------------------------------------------
    ;#> BEGIN: Debug
    ;   Promenne pro ladici ucely
    ;-------------------------------------------------------------------------

    #ifdef DEBUG
        ;   porty pro DEBUG, cteni tlacitek
        PORTA_DBG           res 1
        PORTB_DBG           res 1

        ;   pro overeni Timer1 testovanim
        VARIABLE_TIMER1_DBG res 1
    #endif


;-------------------------------------------------------------------------
;#< END: Promenne
;*************************************************************************

;*************************************************************************
;#> BEGIN: Konstanty
;-------------------------------------------------------------------------
        #define     F   1

    ;-------------------------------------------------------------------------
    ;#> BEGIN: Automat pro prepinac
    ;   Preddefinovani automatu pro prepinac. Prechazi se dle stavu tlacitka
    ;   RB0(S1). CONST_SWITCH_FSA_BIT_COUNT - bit urcuje zda je povoleno
    ;   citani (log. 1). CONST_SWITCH_FSA_BIT_BWAIT - bit urcuje na jakou
    ;   hodnotu (log. uroven) se ceka, aby automat presel do dalsiho stavu.
    ;-------------------------------------------------------------------------
        #define     CONST_SWITCH_FSA_INIT       b'11001001'
        #define     CONST_SWITCH_FSA_BIT_COUNT  0
        #define     CONST_SWITCH_FSA_BIT_BWAIT  1

    ;-------------------------------------------------------------------------
    ;#> BEGIN: Tlacitka
    ;   Maska pro debounce tlacitek, ty jsou zjistovany v Timer1. Pozice bitu
    ;   v promenne BUTTON_STATE odpovidajici danym tlacitkum.
    ;-------------------------------------------------------------------------
```

```
        #define     CONST_BUTTON_DEBOUNCE_MASK  b'00000111'
        #define     CONST_BUTTON_BIT_RB0_S1     0
        #define     CONST_BUTTON_BIT_RA5_S2     1

    ;-------------------------------------------------------------------------------
    ;#> BEGIN: Konstanty polynomu
    ;   Definicni konstanty pro vypocet polynomu
    ;   VARIABLE_RESULT = CONST_K2 * VARIABLE_X * VARIABLE_X +
    ;                     CONST_K1 * VARIABLE_X +
    ;                     CONST_K0
    ;-------------------------------------------------------------------------------
        #define     CONST_K2        d'07'
        #define     CONST_K1        d'10'
        #define     CONST_K0        d'09'

    ;-------------------------------------------------------------------------------
    ;#> BEGIN: Timer0
    ;   Prednastaveni Timer0 pro periodu 600ms pri fosc 10MHz
    ;   Preruseni Timer0 = 4*32*46875/10MHz = 600ms
    ;-------------------------------------------------------------------------------
        ; ridici registr T0CON
        ; 0   - bit7 - Stop Timer0
        ; 0   - bit6 - 16bit citac
        ; 0   - bit5 - interni hodiny
        ; 0   - bit4 - "ignorovan"
        ; 0   - bit3 - pouzita preddelicka
        ; 100 - bit2-0 - preddelicka 1:32
        #define     CONST_TIMER0_T0CON      b'00000100'

        ; registry TMR0H a TMR0L
        ; delicka - Timer0 => 0xFFFF - 46875 = 0x48E4
        #define     CONST_TIMER0_TMR0H      h'48'
        #define     CONST_TIMER0_TMR0L      h'E4' +1

    ;-------------------------------------------------------------------------------
    ;#> BEGIN: Timer1
    ;   Prednastaveni Timer1 pro periodu cca 10ms pri fosc 10MHz
    ;   Preruseni Timer1 = 4*8*3125/10MHz = 10ms
    ;-------------------------------------------------------------------------------
        ; ridici registr T1CON
        ; 1   - bit7 - 16bit citac
        ; 0   - bit6 - neni pouzit Timer1 oscilator
        ; 11  - bit5-4 - predelicka 1:8
        ; 0   - bit3 - Timer1 osc. je vyputy
        ; 0   - bit2 - "ignorovan"
        ; 0   - bit1 - interni hodiny
        ; 0   - bit0 - Stop Timer1
        #define     CONST_TIMER1_T1CON      b'10110000'

        ; registry TMR1H a TMR1L
        ; delicka - Timer1 => 0xFFFF - 3125 = 0xF3CA
        #define     CONST_TIMER1_TMR1H      h'F3'
        #define     CONST_TIMER1_TMR1L      h'CA' +1

;-------------------------------------------------------------------------------
;#< END: Konstanty
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Vektor zacatku programu po resetu
;-------------------------------------------------------------------------------
;   Odskok na hlavni fce Main
;*******************************************************************************
    ORG 0x0000

        goto    Main

;*******************************************************************************
;#> BEGIN: Vektor preruseni s vysokou prioritou
;-------------------------------------------------------------------------------
;   Zpracovani preruseni s vysokou prioritou odskokem na Timer0Interrupt
;*******************************************************************************
    ORG 0x0008
```

```
        goto    Timer0Interrupt

;*******************************************************************************
;#> BEGIN: Vektor preruseni s nizkou prioritou
;-------------------------------------------------------------------------------
;       Zpracovani preruseni s nizkou prioritou odskokem na Timer1Interrupt
;*******************************************************************************
    ORG 0x0018

        goto    Timer1Interrupt

;*******************************************************************************
;#> BEGIN: Main
;-------------------------------------------------------------------------------
;   Inicializace HW procesoru (Timer0, Timer1, porty). Nastaveni promennych
;   pro vypocet polynomu. Inicializace osetreni stavu tlacitek. Nastaveni
;   preruseni pro jednotlive casovace.
;
;   Vstup:  neni
;   Vystup: neni
;   Modifikuje:
;-------------------------------------------------------------------------------
Main:
        ; inicializace procesoru
        call    Timer0Init
        call    Timer1Init
        call    PortInit
        call    ButtonInit
        call    SolveInit
        call    OutputVariableX
        call    SolveResult
        call    SwitcherInit
        call    InterruptInit

        ; spusteni ulohy
        call    TimerStart
Loop:
        nop
        bra     Loop

;-------------------------------------------------------------------------------
;#< END: Main
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Timer0Init
;-------------------------------------------------------------------------------
;   Inicializace casovace Timer0 na 600ms.
;
;   Vstup: neni
;   Vystup: neni
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
Timer0Init:

        movlw   CONST_TIMER0_T0CON      ; rezim Timer0
        movwf   T0CON
        movlw   CONST_TIMER0_TMR0H      ; nastaveni delicky
        movwf   TMR0H
        movlw   CONST_TIMER0_TMR0L
        movwf   TMR0L

        return
;-------------------------------------------------------------------------------
;#< END: Timer0Init
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Timer0Interrupt
;-------------------------------------------------------------------------------
;   Zpracovani pozadavku s vysokou prioritou - nastaveno na Timer0.
;   Zvetseni promenne pro citac VARIABLE_X a zobrazeni hodnoty na vystupu,
```

```
;   vypocet polynomu - neni-li zakazano citani nebo pozadavek na nulovani.
;   Vyuziva se stinovych registru.
;
;   Vstup:  neni
;   Vystup: neni
;   Modifikuje:
;-------------------------------------------------------------------------------
Timer0Interrupt:

        ; obnoveni delicky pro Timer0 pri priruseni
        movlw   CONST_TIMER0_TMR0H
        movwf   TMR0H
        movlw   CONST_TIMER0_TMR0L
        movwf   TMR0L

        ; je povoleno citani?
        btfss   SWITCH_FSA_STATE, CONST_SWITCH_FSA_BIT_COUNT
        bra     Timer0Interrupt_skipCount

        ; je pozadavek na nulovani?
        btfss   BUTTON_STATE, CONST_BUTTON_BIT_RA5_S2
        bra     Timer0Interrupt_skipCount

        ; zvyseni hodnoty promenne citace
        incf    VARIABLE_X

        ; a jeji vystup na LED
        call    OutputVariableX

        ; a vypocet polynomu
        call    SolveResult

    Timer0Interrupt_skipCount:
        ; vynulovani priznaku osetreneho preruseni
        bcf     INTCON, TMR0IF

        retfie  FAST
;-------------------------------------------------------------------------------
;#< END: Timer0Interrupt
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Timer1Init
;-------------------------------------------------------------------------------
;   Inicializace casovace Timer1 na cca 10ms.
;
;   Vstup: neni
;   Vystup: VARIABLE_TIMER1_DBG (DEBUG)
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
Timer1Init:

        movlw   CONST_TIMER1_T1CON      ; rezim Timer1
        movwf   T1CON
        movlw   CONST_TIMER1_TMR1H      ; nastaveni delicky
        movwf   TMR1H
        movlw   CONST_TIMER1_TMR1L
        movwf   TMR1L

    #ifdef DEBUG
        ; podmineny preklad pro otestovani casovace
        clrf    VARIABLE_TIMER1_DBG
    #endif

        return
;-------------------------------------------------------------------------------
;#< END: Timer1Init
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: Timer1Interrupt
;-------------------------------------------------------------------------------
;   Zpracovani preruseni s nizkou prioritou - nastaveno na Timer1.
```

```
;   Nacteni stavu tlacitek, vyhodnoceni automatu pro prepinac. Je-li pozadavek
;   na nulovani, je promenna citace nastavena na nulu.
;
;   Vstup:  neni
;   Vystup: neni
;   Modifikuje:
;------------------------------------------------------------------------------
Timer1Interrupt:

        ; ulozeni registru
        movff   STATUS, STACK_STATUS
        movff   WREG, STACK_WREG
        movff   BSR, STACK_BSR

        ; obnoveni delicky pro Timer1 pri priruseni
        movlw   CONST_TIMER1_TMR1H
        movwf   TMR1H
        movlw   CONST_TIMER1_TMR1L
        movwf   TMR1L

    #ifdef DEBUG
        ; podmineny preklad pro otestovani casovace
        incf    VARIABLE_TIMER1_DBG
    #endif

        ; cteni stavu tlacitka S1 pro automat citace
        call    CheckButtonS1

        ; vypocet noveho stavu pro automat citace
        call    SwitcherFSA

        ; tlacitko S2 pro vynulovani citace
        call    CheckButtonS2

        ; je pozadavek na nulovani?
        btfsc   BUTTON_STATE, CONST_BUTTON_BIT_RA5_S2
        bra     Timer1Interrupt_skipClear

        clrf    VARIABLE_X
        call    OutputVariableX
        call    SolveResult

    Timer1Interrupt_skipClear:
        ; vynulocani priznaku osetreneho preruseni
        bcf     PIR1, TMR1IF

        ; obnoveni registru
        movff   STACK_BSR, BSR
        movff   STACK_WREG, WREG
        movff   STACK_STATUS, STATUS

        retfie
;------------------------------------------------------------------------------
;#< END: Timer1Interrupt
;******************************************************************************

;******************************************************************************
;#> BEGIN: PortInit
;------------------------------------------------------------------------------
;   Nastaveni HW portu. PORTD bude vystupni. PORTA<5> a PORTB<0> vstupni.
;
;   Vstup:  neni
;   Vystup: neni
;   Modifikuje: WREG, STATUS
;------------------------------------------------------------------------------
PortInit:

        ; PORTD vystupni
        clrf    WREG
        movwf   PORTD       ; vystup PORTD = 0x00
        movwf   TRISD

        ; PORTA, port RA5 vstupni
```

```
        bsf     TRISA, RA5
        bsf     WDTCON, ADSHR
        bsf     ANCON0, PCFG4
        bcf     WDTCON, ADSHR

        ; PORTB, port RB0 vstupni
        bsf     TRISB, RB0

        return
;-------------------------------------------------------------------------------
;#< END: PortInit
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: OutputVariableX
;-------------------------------------------------------------------------------
;   Zobrazeni vystupu promenne citace VARIABLE_X na PORTD.
;
;   Vstup:  VARIABLE_X
;   Vystup: neni
;   Modifikuje: STATUS
;-------------------------------------------------------------------------------
OutputVariableX:

        movff   VARIABLE_X, PORTD
        return
;-------------------------------------------------------------------------------
;#< END: OutputVariableX
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: ButtonInit
;-------------------------------------------------------------------------------
;   Nastaveni priznaku pro tlacitka. Jelikoz je na tlacitkach opacna
;   logika, je promenna nastavena na log. 1.
;
;   Vstup: neni
;   Vystup: BUTTON_STATE, BUTTON_RB0_S1, BUTTON_RA5_S2
;           PORTA_DBG (DEBUG), PORTB_DBG (DEBUG)
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
ButtonInit:

        setf    BUTTON_STATE, F
        movlw   CONST_BUTTON_DEBOUNCE_MASK
        movwf   BUTTON_RB0_S1
        movwf   BUTTON_RA5_S2

    #ifdef DEBUG
        bsf     PORTB_DBG, RB0
        bsf     PORTA_DBG, RA5
    #endif

        return
;-------------------------------------------------------------------------------
;#< END: ButtonInit
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: CheckButtonS1
;-------------------------------------------------------------------------------
;   Cteni vstupniho hodnoty z portu PORTB RB0 {tlacitko S1). Vyuziva se masky
;   CONST_BUTTON_DEBOUNCE_MASK, aby se rozhodlo, zda je hodnota platna -
;   osetreni zakmitu tlacitka.
;
;   Vstup:  PORTB, PORTB_DBG (DEBUG), BUTTON_RB0_S1
;   Vystup: BUTTON_STATE, BUTTON_RB0_S1
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
CheckButtonS1:

        rlncf   BUTTON_RB0_S1, F
```

```
        ; aktualni hodnota tlacitka
    #ifdef DEBUG
        btfsc   PORTB_DBG, RB0
    #else
        btfsc   PORTB, RB0
    #endif

        ; ulozeni stavu do promenne
        bsf     BUTTON_RB0_S1, 0

        ; je hodnota v promenne?
        movlw   CONST_BUTTON_DEBOUNCE_MASK
        andwf   BUTTON_RB0_S1
        bz      CheckButtonS1_0             ; 000? -> tlacitko stisknuto
        cpfseq  BUTTON_RB0_S1               ; 111? -> tlacitko uvolneno
        bra     CheckButtonS1_exit          ; jinak konec

    CheckButtonS1_1:
        ; zmena stavu tlacitka
        bsf     BUTTON_STATE, CONST_BUTTON_BIT_RB0_S1
        bra     CheckButtonS1_exit

    CheckButtonS1_0:
        ; zmena stavu tlacitka
        bcf     BUTTON_STATE, CONST_BUTTON_BIT_RB0_S1

    CheckButtonS1_exit:
        return

;-------------------------------------------------------------------------------
;#< END: CheckButtonS1
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: CheckButtonS2
;-------------------------------------------------------------------------------
;   Cteni vstupniho hodnoty z portu PORTA RA5 {tlacitko S2). Vyuziva se masky
;   CONST_BUTTON_DEBOUNCE_MASK, aby se rozhodlo, zda je hodnota platna -
;   osetreni zakmitu tlacitka.
;
;   Vstup:  PORTA, PORTA_DBG (DEBUG), BUTTON_RA5_S2
;   Vystup: BUTTON_STATE, BUTTON_RA5_S2
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
CheckButtonS2:

        rlncf   BUTTON_RA5_S2, F

        ; aktualni hodnota tlacitka
    #ifdef DEBUG
        btfsc   PORTA_DBG, RA5
    #else
        btfsc   PORTA, RA5
    #endif

        ; ulozeni stavu do promenne
        bsf     BUTTON_RA5_S2, 0

        ; je hodnota v promenne?
        movlw   CONST_BUTTON_DEBOUNCE_MASK
        andwf   BUTTON_RA5_S2
        bz      CheckButtonS2_0         ; 000? -> tlacitko stisknuto
        cpfseq  BUTTON_RA5_S2           ; 111? -> tlacitko uvolneno
        bra     CheckButtonS2_exit      ; jinak konec

    CheckButtonS2_1:
        ; zmena stavu tlacitka
        bsf     BUTTON_STATE, CONST_BUTTON_BIT_RA5_S2
        bra     CheckButtonS2_exit

    CheckButtonS2_0:
        ; zmena stavu tlacitka
        bcf     BUTTON_STATE, CONST_BUTTON_BIT_RA5_S2
```

```
    CheckButtonS2_exit:
        return

;-------------------------------------------------------------------------------
;#< END: CheckButtonS1
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: SwitchInit
;-------------------------------------------------------------------------------
;   Inicializace stavoveho automatu pro prepinac.
;
;   Vstup: neni
;   Vystup: SWITCH_FSA_STATE
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
SwitcherInit:

        movlw   CONST_SWITCH_FSA_INIT
        movwf   SWITCH_FSA_STATE

        return
;-------------------------------------------------------------------------------
;#< END: SwitcherInit
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: SwitcherFSA
;-------------------------------------------------------------------------------
;   Stavovy automat pro prepinac. Prechod do noveho stavu rotaci o dva bity
;   vpravo. Stav automatu v SWITCH_FSA_STATE, rizeny tlacitkem S1.
;
;   Vstup:  SWITCH_FSA_STATE, BUTTON_STATE (S1)
;   Vystup: SWITCH_FSA_STATE
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
SwitcherFSA:

        ; stav tlacitka RB0 (S1)?
        btfsc   BUTTON_STATE, CONST_BUTTON_BIT_RB0_S1
        bra     SwitcherFSA_buttonOff

        ; tlacitko RB0 (S1) je zmacknuto (log. 0)
    SwitcherFSA_buttonOn:
        btfss   SWITCH_FSA_STATE, CONST_SWITCH_FSA_BIT_BWAIT
        bra     SwitcherFSA_changeState
        bra     SwitcherFSA_exit

        ; tlacitko RB0 (S1) neni znacknuto (log. 1)
    SwitcherFSA_buttonOff:
        btfsc   SWITCH_FSA_STATE, CONST_SWITCH_FSA_BIT_BWAIT
        bra     SwitcherFSA_changeState
        bra     SwitcherFSA_exit

        ; zmena stavu automatu
        ; WREG - race-condition
    SwitcherFSA_changeState:
        movff   SWITCH_FSA_STATE, WREG
        rrncf   WREG, 0
        rrncf   WREG, 0
        movwf   SWITCH_FSA_STATE

    SwitcherFSA_exit:
        return
;-------------------------------------------------------------------------------
;#< END: SwitcherFSA
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: InterruptInit
;-------------------------------------------------------------------------------
;   Nastaveni prerusovaciho systemu. Timer0 ma nastavenu vyssi prioritu
```

```
;   a Timer1 je nastaven na nizsi prioritu.
;
;   Vstup: neni
;   Vystup: neni
;   Modifikuje:  STATUS
;-------------------------------------------------------------------------------
InterruptInit:

        bsf     RCON, IPEN          ; povoleni urovni preruseni High/Low
        bsf     INTCON, GIEH        ; povol - vysoka priorita
        bsf     INTCON, GIEL        ; povol - nizka priorita
        bsf     INTCON, TMR0IE      ; Timer0 - preruseni pretecenim
        clrf    INTCON2
        bsf     INTCON2, TMR0IP     ; Timer0 - vysoka priorita
        clrf    PIE1
        bsf     PIE1, TMR1IE        ; Timer1 - preruseni pretecenim
        clrf    IPR1
        bcf     IPR1, TMR1IP        ; Timer1 - nizka priorita

        return
;-------------------------------------------------------------------------------
;#< END: InterruptInit
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: TimerStart
;-------------------------------------------------------------------------------
;   Spusteni casovacu Timer0 a Timer1.
;
;   Vstup:  neni
;   Vystup: neni
;   Modifikuje:
;-------------------------------------------------------------------------------
TimerStart:

        bsf     T1CON, TMR1ON
        bsf     T0CON, TMR0ON

        return
;-------------------------------------------------------------------------------
;#< END: TimerStart
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: SolveInit
;-------------------------------------------------------------------------------
;   Nastaveni promenne citace na definovanou hodnotu.
;
;   Vstup:  neni
;   Vystup: VARIABLE_X
;   Modifikuje: STATUS
;-------------------------------------------------------------------------------
SolveInit:

        clrf    VARIABLE_X          ; VARIABLE_X <- 0

        return
;-------------------------------------------------------------------------------
;#< END: SolveInit
;*******************************************************************************

;*******************************************************************************
;#> BEGIN: SolveResult
;-------------------------------------------------------------------------------
;   Vypocet polynomu.
;   VARIABLE_RESULT = CONST_K2 * VARIABLE_X * VARIABLE_X +
;                     CONST_K1 * VARIABLE_X +
;                     CONST_K0
;   Vztah pro vypocet je nasledujici (Hornerovo schema) TODO.
;
;   VARIABLE_RESULT = \
;       ( CONST_K2 * VARIABLE_X + CONST_K1 ) * VARIABLE_X + CONST_K0
;
```

```
;   Vysledek je ulozen do promenne VARIABLE_RESULT. Pro mezi vypocet je
;   pouzita promenna VARIABLE_RESULT_TMP. A take promenna VARIABLE_X je
;   zastoupena ve vypoctu promennou VARIABLE_X_TMP. Duvodem je moznost
;   volani fce i ve zdroji nizkeho preruseni a zamezeni race-condition.
;
;   Vstup: VARIABLE_X
;   Vystup: VARIABLE_RESULT
;   Modifikuje:  WREG, STATUS
;-------------------------------------------------------------------------------
SolveResult:

        ; vytvoreni lokalni kopie
        ; z VARIABLE_TMP do VARIABLE_X_TMP
        movff   VARIABLE_X, VARIABLE_X_TMP

        ; vynulovani pomocne promenne pro vysledek
        clrf    VARIABLE_RESULT_TMP +0
        clrf    VARIABLE_RESULT_TMP +1
        clrf    VARIABLE_RESULT_TMP +2

        ; CONST_K2(8bit) * VARIABLE_X_TMP(8bit) + CONST_K1(8bit)
        ;                                 => VARIABLE_RESULT_TMP(16bit)
        ;-----------------------------------------------------------------------
            ; CONST_K2 * VARIABLE_X_TMP => PRODH:PRODL
        movlw   CONST_K2
        mulwf   VARIABLE_X_TMP
            ; CONST_K1 + (CONST_K2*VARIABLE_X_TMP)[0]
            ;                                       => VARIABLE_RESULT_TMP[0]
        movlw   CONST_K1
        movff   PRODL, VARIABLE_RESULT_TMP +0
        addwf   VARIABLE_RESULT_TMP +0
            ; "Carry" + (CONST_K2*VARIABLE_X_TMP)[1]
            ;                                       => VARIABLE_RESULT_TMP[1]
        clrf    WREG
        movff   PRODH, VARIABLE_RESULT_TMP +1
        addwfc  VARIABLE_RESULT_TMP +1

        ; VARIABLE_RESULT_TMP(16bit) * VARIABLE_X_TMP(8bit) + CONST_K0(8bit)
        ;                                 => VARAIBLE_RESULT_TMP(24bit)
        ;-----------------------------------------------------------------------
            ; VARIABLE_X_TMP * VARIABLE_RESULT_TMP[1]
            ;                                       => PRODH:PRODL
        movff   VARIABLE_X_TMP, WREG
        mulwf   VARIABLE_RESULT_TMP +1
            ; (VARIABLE_X_TMP * VARIABLE_RESULT_TMP[1])[0]
            ;                                       => VARIABLE_RESULT_TMP[1]
        movff   PRODL, VARIABLE_RESULT_TMP +1
            ; (VARIABLE_X_TMP * VARIABLE_RESULT_TMP[1])[1]
            ;                                       => VARIABLE_RESULT_TMP[2]
        movff   PRODH, VARIABLE_RESULT_TMP +2
            ; VARIABLE_X_TMP * VARIABLE_RESULT_TMP[0]
            ;                                       => PRODH:PRODL
        mulwf   VARIABLE_RESULT_TMP +0
            ; CONST_K0 + (VARIABLE_X_TMP * VARIABLE_RESULT_TMP[0])[0]
            ;                                       => VARIABLE_RESULT_TMP[0]
        movlw   CONST_K0
        movff   PRODL, VARIABLE_RESULT_TMP +0
        addwf   VARIABLE_RESULT_TMP +0
            ; "Carry" + (VARIABLE_X_TMP * VARIABLE_RESULT_TMP[0])[1]
            ;       + VARIABLE_RESULT_TMP[1]    => VARIABLE_RESULT_TMP[1]
        movff   PRODH, WREG
        addwfc  VARIABLE_RESULT_TMP +1
            ; "Carry" + VARIABLE_RESULT_TMP[2]  => VARIABLE_RESULT_TMP[2]
        clrf    WREG
        addwfc  VARIABLE_RESULT_TMP +2

        ; zkopirovani vysledku
        movff   VARIABLE_RESULT_TMP +0, VARIABLE_RESULT +0
        movff   VARIABLE_RESULT_TMP +1, VARIABLE_RESULT +1
        movff   VARIABLE_RESULT_TMP +2, VARIABLE_RESULT +2

        return
;-------------------------------------------------------------------------------
```

```
;#< END: SolveResult
;*******************************************************************************

;*******************************************************************************
;#> Program: END
;*******************************************************************************
        END
```