Programfejlesztés PIC mikrovezérlőkre II.

Írta: Molnár Zsolt

BMF KVK MAI 2007. március 28.

Tartalomjegyzék

1. Bevezetés

Ez az útmutató a **Programfejlesztés PIC mikrovezérlőkre I. c. mérés** útmutatóján alapul, annak **folyatatása**. Ebben a segédletben az MPLAB **szimulátorának további lehetőségeit** ismerheti meg, illetve **újabb perifériák** kezeléséről ejtünk szót. Az MPLAB IDE ismertetését, a programfejlesztés alapvető lépéseinek, és a szimulátor legfontosabb funkcióinak bemutatását a fenti útmutatóban találja.

A **mérés sikeres elvégzését** nagymértékben **segíti** a mérést előkészítő előadások anyagának elsajátítása, illetve a Programfejlesztés PIC mikrovezérlőkre I. c. mérés elvégzése.

A segédlet megírásakor az **MPLAB** legfrissebb elérhető verziója a **7.50**-es.

A mérés során a **18F452** típussal dolgozunk, mivel a későbbiekben ezt a típust tartalmazó demonstrációs panelt fogjuk használni.

2. Mintafeladatok megoldásának ismertetése

A következőkben a mérés előző részéhez képest **magasabb szinten ismerheti meg** a rendelkezésre álló **szimulációs** lehetőségeket. A mintapéldák olyan perifériákat is használnak, amelyekkel eddig nem foglalkoztunk.

A mérésen megszerzett **ismeretek önálló munkával**, részben a laboratóriumi gyakorlatokon, részben azon kívüli tevékenységgel, a Microchip oldaláról és az Internet más részeiből letölthető mintapéldák és a súgó tanulmányozásával **bővíthetőek**.

2.1. példa

Írjunk programot, amely a BANK0 60h…7Fh területét indirekt címzés felhasználásával átmásolja a BANK1 180h…19Fh területére! A feladat megoldásához kövessük az **alábbi lépéseket**!

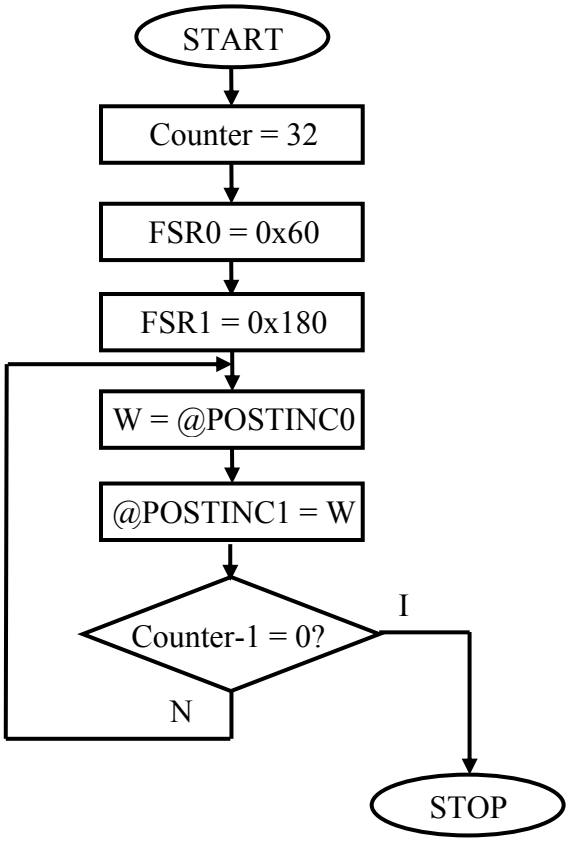
a. **Tervezzük** meg a programot folyamatábra segítségével!
b. **Hozzunk létre projektet és a kódot**, majd végezzük el a fordítást!
c. **Ellenőrizzük** a program működőképességét szimulációval!

1.a. A program **megtervezése**

Indirekt címzésnél az **FSR** regiszter **tartalma által megcímzett regiszter az operandus**. A PIC18-as családnál különleges lehetőség indirekt címzés esetén a cím **automatikus inkrementálása és dekrementálása**. **Ehhez külön regiszterek** állnak rendelkezésre, amelyeket megcímezve, az FSR értéke a felhasználása előtt vagy után inkrementálódik illetve dekrementálódik (lásd katalógus 4.12 fejezet).

Jelen esetben a **forrás** címének tárolásához **FSR0**-át, a **cél** címének tárolásához pedig **FSR1**-et használjuk. Mivel az indirekt címzés regiszterei az adatmemória teljes egészét képesek megcímezni, ezért a két **memóriabank** közötti **váltásra nincs szükség**. A programban felhasználjuk az **automatikus cím-inkrementálás** lehetőségét, ehhez **POSTINC0**, illetve **POSTINC1** regisztereket kell megcímezni, amelyek az adatmozgató művelet végrehajtása

után növelik a hozzájuk tartozó FSR regiszter értékét. Az **áthelyezendő memóriaterület hosszát** az N = végcím – kezdőcím + 1 képlettel számíthatjuk, amelyet a Counter változóban helyezünk el. A program **folyamatábrája** az alábbiak szerint alakul:



1.b. Hozza létre a projektet és a forrásfájlt az előzőekben tanultak alapján! Végezze el a fordítást, és az esetleges szintaktikai hibák javítását!

A forrásfájl egy lehetséges változatát a következőkben közöljük.

```
;*******************************************************************************************
;* feladat_3.asm
;*******************************************************************************************
;* A program indirekt címzés felhasználásával átmásolja
;* a BANK0 60h…7Fh területét a BANK1 180h…19Fh területre
;*******************************************************************************************
;
        list p=18f452

        include "p18f452.inc"          ; A processzorfüggő deklarációkat tartalmazó include fájl

Counter         equ 0x50               ; Változók elhelyezése az általános felhasználású területre
```

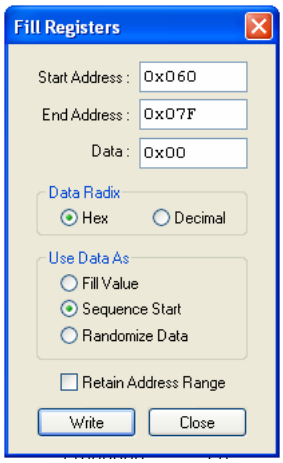
```
Rst_vect                        ; Reset vektor, tápbekapcsoláskor, resetkor innen indul
        org     0x0000          ; a program végrehajtása

        goto    Start

Start                           ; A program kezdete
        org     0x0040

        movlw   .32             ; A Counter változó feltöltése:
        movwf   Counter         ; 32 bájtot kell átmásolni

        lfsr    FSR0, 0x60      ; A másolandó terület kezdőcíme
        lfsr    FSR1, 0x180     ; A feltöltendő terület kezdőcíme

Next
        movf    POSTINC0, 0     ; A soron következő másolandó adat betöltése W-be
        movwf   POSTINC1        ; W-nek a soron következő helyre való kiírása
        decfsz  Counter, 1      ; A ciklusszámláló csökkentése
        goto    Next            ; Ismétlés, ha a ciklusszámláló nem nulla

Stop                            ; Kész a másolás, egyhelyben járás
        goto    Stop

        end
```

1.c. A program **vizsgálata**

A program működőképességének vizsgálatához futtatás előtt **a forrás memóriaterületet fel kell töltenünk** a cél memóriaterülettől eltérő adatokkal, hogy az átmásolás követhető legyen. Ezen kívül **érdemes figyelemmel kísérn**i a speciális funkciójú regiszterek közül az **FSR0** és **FSR1** regisztereket, hogy megfigyelhessük az automatikus inkrementálás működését. Ehhez nyissuk meg a nyomkövető ablakot (*View→Watch*), és helyezzük el benne FSR0-át és FSR1-et.

A forrásterület feltöltését elvégezhetjük kézi úton is, egyszerűbb azonban a **feltöltést automatizálni**. Nyissuk meg a fájlregiszterek ablakát (*View→File Registers*). Kattintsunk jobb gombbal az ablak felett, és a megjelenő menüben válasszuk ki a regiszterek feltöltése (*Fill Registers…*) parancsot. A felugró ablakban a megadott kezdőcímtől megadott végcímig állandó értékkel, 00h-tól FFh-ig szekvenciálisan változó adatokkal, vagy véletlenszerű adatokkal tölthető fel a memória.

Töltsük fel a 60h…7Fh területet szekvenciálisan változó adatokkal. Ehhez a következőképpen kell kitölteni a mezőket:



Futtassa a programot lépésenkénti üzemmódban! **Figyelje meg** az indirekt címzésre használt regiszterek (FSR0 és FSR1) változását, valamint a másolási műveletet!

2.2. példa

Írjunk megszakításon alapuló időzítő rutint, amely a TIME regiszterben tárolt 8 bites előjel nélküli értéket felhasználva $t = \text{TIME} \cdot 10$ ms időközönként adott tevékenységeket futtat le. A feladat megoldásához használjuk TIMER1-et, az órajel legyen 4 MHz! A tevékenység az egyszerűség kedvéért legyen PORTB 0. bitjének invertálása. A feladat megoldásához kövessük az **alábbi lépéseket**!

a. Tervezzük meg a programot!
b. Hozzunk létre projektet és a kódot, majd végezzük el a fordítást!
c. Ellenőrizzük a program működőképességét szimulációval!

2.a. A katalógus segítségével **állítsa össze** a PORTB, a TIMER1, és a megszakításrendszer inicializálásához **szükséges beállítások listáját**!

2.b. Hozza létre a projektet és a kódot!

Végezze el a fordítást, és az esetleges szintaktikai hibák javítását!

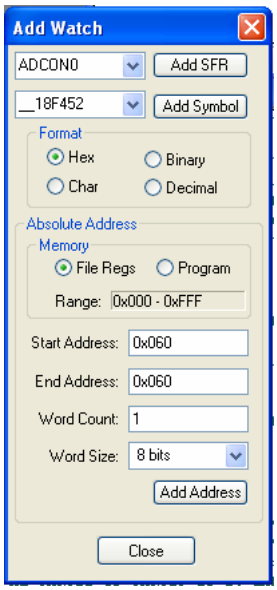
A forrásfájl egy lehetséges változatát a következőkben közöljük.

```
;*******************************************************************************
;* feladat_4.asm
;*******************************************************************************
;* A program TIMER1 megszakításának használatával PORTB0 állapotát
;* TIME*10ms-onként invertálja. A program TIME=0-ra nem működik!
;* A PicDem2 Plus panel órajele 4MHz, a programban ezt vettük alapul
;*******************************************************************************

        list p=18f452

        include "p18f452.inc"          ; A processzorfüggő deklarációkat tartalmazó include fájl

TIME            equ 0x60               ; Változók elhelyezése az általános felhasználású
TIME_WORK       equ 0x70               ; regiszterek területén

Rst_vect                               ; Reset vektor, tápbekapcsoláskor, resetkor innen indul
        org     0x0000                 ; a program végrehajtása

        goto    Start

Int_vect                               ; Megszakítás vektor. Ha engedélyezett megszakítás van,
        org     0x0008                 ; innen folytatódik a program végrehajtása
                                       ; Megszakítás-forrás azonosítása:
        btfss   PIE1, TMR1IE           ; ha TMR1IE és TMR1IF is 1, akkor TIMER1-től jött
        reset                          ; megszakítás, különben nem.
        btfss   PIR1, TMR1IF           ; Mivel csak TIMER1 megszakítása van engedélyezve,
        reset                          ; ezért ha nem tőle jött megszakítás, akkor resetelünk
        bra     TMR1_ISR               ; Ugrás TIMER1 megszakításának kiszolgáló rutinjára

Start                                  ; A program kezdete
        org     0x0040
                                       ; PORTB beállítása
        bcf     LATB, 0
;       bcf     TRISB, 0               ; Itt ez lenne az egyszerűbb megoldás, a portláb irányának
                                       ; beállítására, az alábbi maszkolás
        movlw   0xFE                   ; több bit egyszerre történő beállítására hatékony
        andwf   TRISB, 1               ; PORTB0-t kimenetté konfiguráljuk

                                       ; Timer1 beállítása
        bsf     T1CON, TMR1ON          ; TIMER1 engedélyezése
        movlw   0xD8                   ; TIMER1 feltöltése úgy, hogy 10ms után csorduljon túl:
        movwf   TMR1H                  ; (65536-TMR1Preload)*(1/1MHz)=10 ms, innen:
        movlw   0xF0                   ; TMR1Preload = 65536 - 10ms*1MHz = 55536 = 0xD8F0
        movwf   TMR1L                  ; TMR1L írásával TMR1H pufferbe írt érték is töltődik.
                                       ; A feltöltési sorrend fontos!
                                       ; Megszakítások beállítása
        bcf     PIR1, TMR1IF           ; TIMER1 megszakítás flagjének törlése
        bsf     PIE1, TMR1IE           ; TIMER1 megszakítás engedélyezése
        bsf     INTCON, PEIE           ; Periféria megszakítások engedélyezve
        bsf     INTCON, GIE            ; Globális megszakítás-engedélyezés

        movff   TIME, TIME_WORK        ; TIME másolása TIME_WORK-be, hogy TIME tartalma
                                       ; ne sérüljön

Stop                                   ; Örök helyben járás, itt lehetne a főprogram
        goto    Stop
```

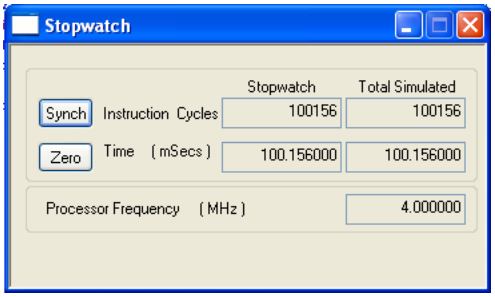
```
        org     0x0200              ; TIMER1 megszakításának kiszolgáló rutinja

TMR1_ISR                            ; Timer1 megszakításának kiszolgálása
        movlw   0xD8                ; Timer1 feltöltése az előbbiekben kiszámított értékkel
        movwf   TMR1H
        movlw   0xF0
        movwf   TMR1L
        decfsz  TIME_WORK, 1
        goto    No_activity
        btg     PORTB, 0            ; Tevékenységek (most csak RB0 invertálása)
                                    ; Ide jöhetnének egyéb tevékenységek...
        movff   TIME, TIME_WORK     ; TIME_WORK feltöltése TIME értékével
No_activity
        bcf     PIR1, TMR1IF        ; Timer1 megszakítása kiszolgálva, megszakítás bitet töröljük
        retfie                      ; Visszatérés megszakításból (mindig "retfie" paranccsal!)

        end
```

2.c. Végezzük el a program **működőképességének vizsgálatát**!

A **nyomkövető ablakban** (*View→Watch*) jelenítsük meg PORTB, TIME és TIME_WORK regisztereket! Mivel PORTB egy speciális funkciójú regiszter, így a szokásos módon hozzáadatjuk a listához. TIME és TIME_WORK általános felhasználású területen vannak. Mivel definiált szimbólumok, így az *Add Symbol* gombbal adhatjuk őket hozzá a listához. Ha a memóriacella szimbólumként nem definiált, akkor a listához adásához a nyomkövető ablak feletti jobb kattintás után megjelenő menüből a hozzáadás (*Add…*) funkciót kiválasztva egy ablak ugrik elő. Ennek alsó részén (*Absolute Address*) adhatunk a listához általános célú regisztert. Példaként a következő ábra a TIME, 0x60 című regiszter hozzáadásához szükséges kitöltést mutatja (a hozzáadást a cím hozzáadása (*Add Address*) gombra kattintva végezhetjük el):



Nyissa meg a stopper ablakot (*Debugger→StopWatch*)! **Helyezzen el töréspontot** a „btg PORTB, 0” programsorra! **Töltse fel** TIME regiszter (0x60) értékét a kívánt értékkel! **Futtassuk a programot**, szimulációval ellenőrizzük számításaink helyességét! Az alábbi ábrán a stopper ablaka látható, Time mezőjében a $TIME = 10$ esetén előálló ($10 \cdot 10\text{ms} = 100ms$) ütemidővel.



2.3. példa

Hozzunk létre a szimulátorban előállított külső gerjesztést, amely az alábbi feltételek szerint **működteti PORTB-t!**

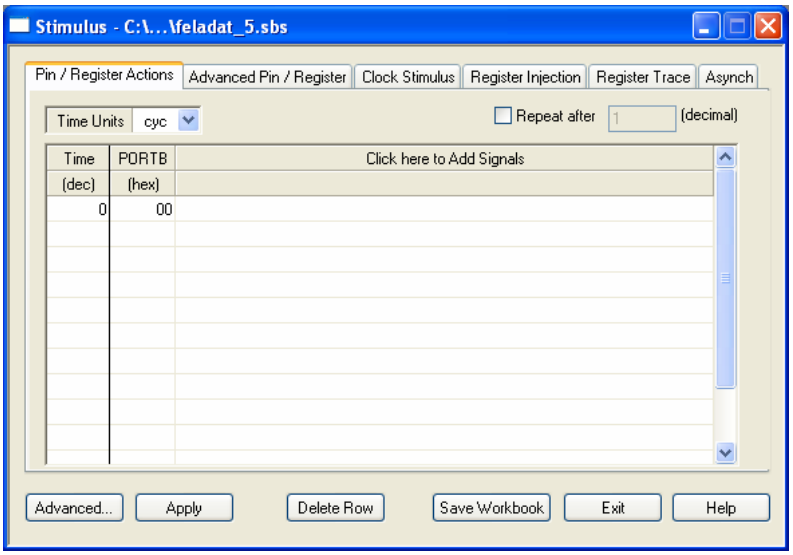
Bekapcsolás után álljon elő a következő szekvencia (minden portláb 0 szintről induljon):

- **RB0 10 gépi ciklusonként váltson szintet** (20 gépi ciklus periódusidejű szimmetrikus négyszögjel)!
- **50 gépi ciklus** késleltetés **után 100 ciklus ideig RB1**-en jelenjen meg **40 gépi ciklus periódusidejű** szimmetrikus **négyszögjel**!
- **RB2** váltson **magas szintre RB1 magas szintre váltása után 15 gépi ciklussal**!
- **RB3**-on álljon elő egy **1 gépi ciklus idejű pozitív impulzus**, **RB2 magas szintre váltása után 10 gépi ciklussal**!

Hozzon létre egy új projektet, de ne adjon hozzá forrást! **Nyissa meg** a logikai analizátor ablakot, adja a jellistához RB0, RB1, RB2 és RB3 lábakat!

Nyissunk új gerjesztés-vezérlő ablakot (*Debugger→Stimulus→New Workbook*)! A kezdeti feltétel (RB0…RB3 kezdeti jelszintje alacsony) biztosításához a *Pin/Register Actions* fülön

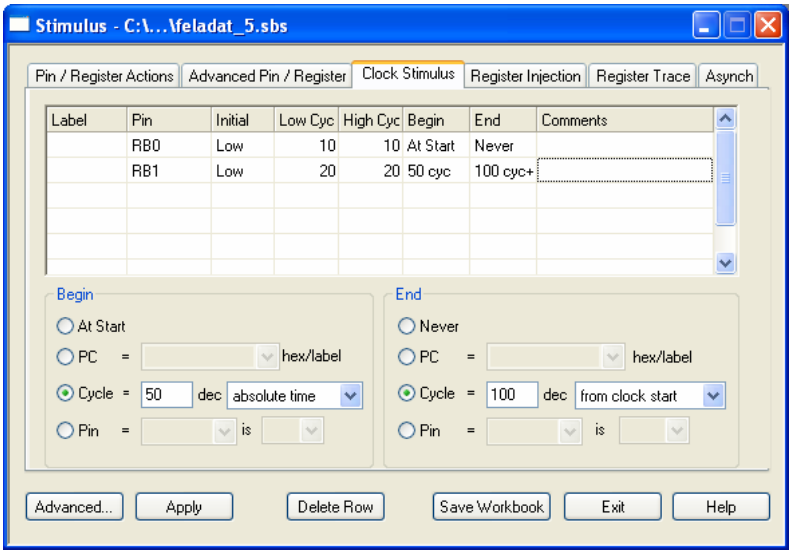
állítsuk be, hogy 0 időpillanatban PORTB összes bitje álljon alacsony szintre! Az idő (*Time*) oszlopban adjuk meg az időpontot (*0*). A jelek hozzáadása mezőn (*Click here to Add Signals*) való kattintással a felugró listából válasszuk ki PORTB-t, majd értékét állítsuk 0-ra. A táblázat kitöltése az alábbi ábra szerint történhet.



Állítsuk elő RB0-on a 20 ciklus periódusidejű szimmetrikus négyszögjelet! Ehhez váltsunk a *Clock Stimulus* fülre. A láb (*Pin*) oszlop első sorára kattintva válasszuk ki RB0-át!
Kezdeti szintje (*Initial*) alacsony (*Low*). Az alacsony szint (*Low Cyc*) ideje 10 ciklus, magas szinté (*High Cyc*) is ugyanannyi.
A gerjesztés kezdete (*Begin*) a bekapcsolás vagy az alaphelyzetbe állítás (*At Start*), és amíg a szimuláció tart, ne fejeződjön be (*End: Never*).

RB1 gerjesztésének beállításánál is hasonlóképpen járunk el. Egyetlen különbséget a kezdeti és a befejező időpont megadása jelenti.
A feladat szerint a gerjesztés kezdete (*Begin*) 50 ciklusnál (50 cyc) van, amelyet a lista alatt a *Begin* mezőben a ciklusszám (*Cycle*) kiválasztása, majd az 50-es érték megadásával lehetséges beállítani.
Befejezése a kezdetéhez képest 100 ciklussal később van, amelyet a befejezés (*End*) mezőben a ciklusszám (*Cycle*) kiválasztása, majd az 100-as érték megadásával, és a kezdetnek, mint viszonyítási alapnak (*from clock start*) kiválasztásával lehetséges beállítani.

A táblázat kitöltése az alábbi ábra szerint történhet.



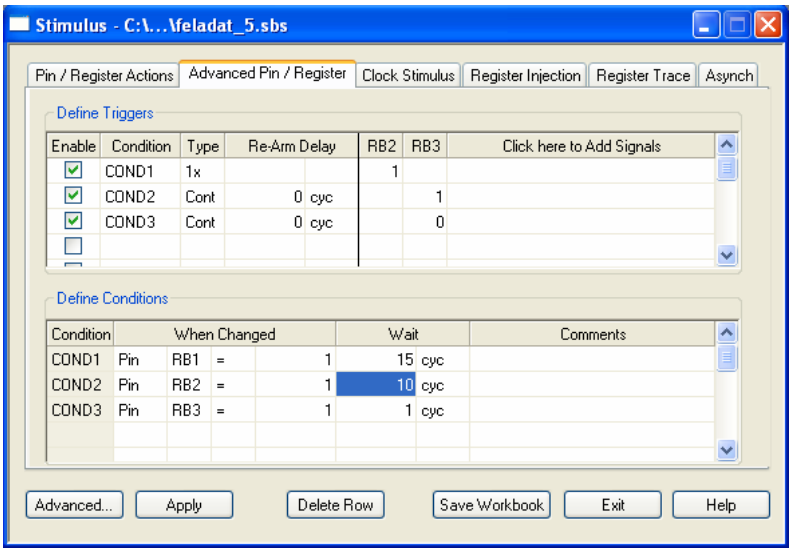
Végül a **feltételes gerjesztéseket** kell beállítani, az *Advanced Pin/Register* fülön. Először hozzuk létre a feltételeket!

Kattintsunk az alsó táblázatban (*Define Conditions*) a *COND1* melletti mezőre. Mivel lábat szeretnénk gerjeszteni, válasszuk a *Pin* beállítást. A következő mezőben állítsuk be RB1-et, mivel a magas szintbe (1) váltása után 15 ciklussal (*Wait: 15 cyc*) kell tevékenységeket végezni.

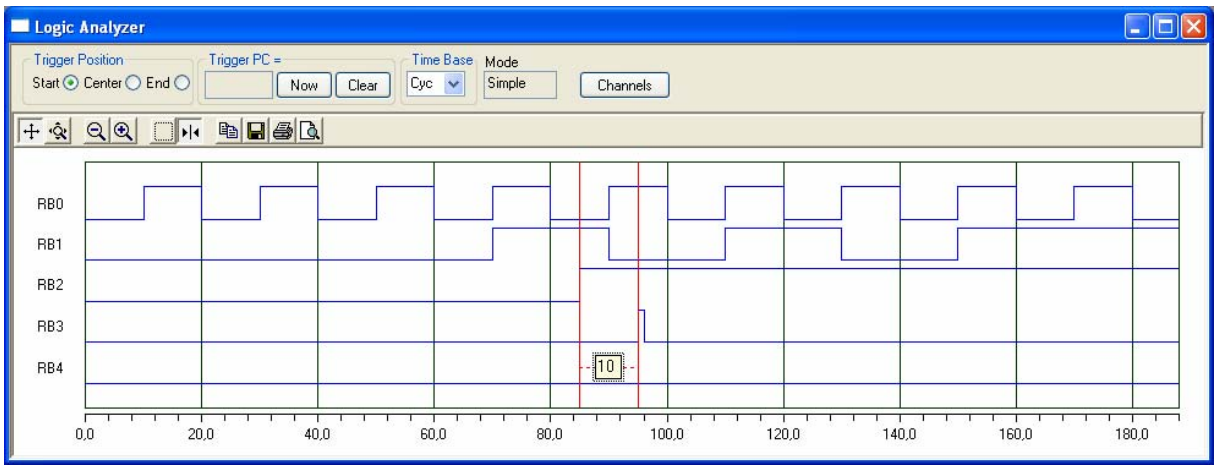
COND2 és *COND3* feltételeket **hozza létre a feladatkiírás és az alábbi ábra szerint**! A feltételek megadása után ki lehet tölteni a felső táblázatot (*Define Triggers*). A típus (*Type*) oszlopban megadható, hogy egyszeri (*1x*) vagy folyamatosan figyelendő (*Cont*) triggerről van-e szó. Az újraélesedési idő (*Re-Arm Delay*) oszlopban megadható, hogy ha a trigger bekövetkezett, utána mennyi idővel kezdődjön el ismét a feltétel keresése.

A táblázat jobb felében a *Click here to Add Signals* mezőre kattintva, megadhatjuk a gerjesztendő lábakat vagy regisztereket.

A feladatban definiált feltételek szerint kitöltött táblázat az alábbiakban látható.



Végezze el a szimulációt egyszerű léptetéssel (*Step*)! A gerjesztés időfüggvénye az alábbi ábrán látható. **Ellenőrizze**, hogy az időfüggvény megfelel-e a feladatkiírásnak!



2.4. példa

A közölt forráslista és a katalógus alapján **elemezze a következő program működését**, majd **végezze el** szimulációval a **vizsgálatát**!

```
;*******************************************************************************
;* feladat_6.asm
;*******************************************************************************
;* A program soros porton megszakítással vesz, ha beérkezett egy bájt,
;* visszaküldi eggyel megnövelve. Beállítások: 9600, 8, N, 1
;*******************************************************************************

        list p=18f452

        include "p18f452.inc"          ; A processzorfüggő deklarációkat tartalmazó include fájl

Rst_vect                               ; Reset vektor, tápbekapcsoláskor, resetkor innen indul
        org     0x0000                 ; a program végrehajtása

        goto    Start

Int_vect                               ; Megszakítás vektor. Ha engedélyezett megszakítás van,
        org     0x0008                 ; innen folytatódik a program végrehajtása

        btfss   PIE1, RCIE             ; USART-tól jött a megszakítás?
        goto    Other_Int              ; Nem, máshonnan, hibakezelés.
        btfss   PIR1, RCIF             ; USART-tól jött a megszakítás?
        goto    Other_Int              ; Nem, máshonnan, hibakezelés.

        movlw   06h                    ; Vételi hiba ellenőrzése
        andwf   RCSTA, W
        btfss   STATUS, Z
        goto    Rcv_Error              ; Vételi hiba (túlfutási vagy kerethiba)

        movf    RCREG, W               ; Vett adat beolvasása
        incf    WREG                   ; Megnövelése
        movwf   TXREG                  ; Adási regiszterbe töltése
        goto    ISR_End                ; Vége a megszakítás kiszolgálásának

Rcv_Error
        bcf     RCSTA, CREN            ; A hibák törlése az USART ki/be kapcsolásával
        bsf     RCSTA, CREN
        goto    ISR_End                ; Vége a megszakítás kiszolgálásának

Other_Int
        goto    Other_Int              ; Mivel csak USART vételi megszakítása engedélyezett, és
                                       ; nem onnan jött megszakítás, ezért megszakítjuk a
                                       ; programvégrehajtás szekvenciáját
ISR_End
        retfie                         ; Visszatérés a megszakításból.

Start                                  ; A program kezdete
        org     0x0050

        bcf     TRISC, 6               ; RC6/TX kimenet
```

```
        bsf     TXSTA, BRGH     ; Magas bitsebesség kiválasztása
        movlw   .25             ; SPBRG feltöltése (9600bps 4MHz órajel esetén)
        movwf   SPBRG           ; Katalógus képlete alapján:
                                ; 9600bps=4MHz/(16(SPBRG+1)), innen SPBRG=25

        bsf     RCSTA, SPEN     ; Soros port engedélyezése
        bsf     RCSTA, CREN     ; Folyamatos vétel engedélyezése
        bsf     TXSTA, TXEN     ; Adás engedélyezése

        bcf     PIR1, RCIF      ; Vételi megszakítás törlése
        bsf     PIE1, RCIE      ; Vételi megszakítás engedélyezése
        bsf     INTCON, PEIE    ; Periféria megszakítások engedélyezése
        bsf     INTCON, GIE     ; Globális megszakítás engedélyezés

Stop                            ; Egyhelyben járás, várakozás megszakításra
        goto    Stop

        end
```

A szimulációhoz használja az alább közölt **gerjesztési fájlt** (*Register Injection*), a TXREG értékét **naplózza** (*Register Trace*)! **Értelmezze a gerjesztési fájl tartalmát** a súgó segítségével!

„usart.txt”:

```
wait 20 ms
30 31 32

wait 10 ms
"BMF KVK "

rand 10 50 ms
"MAI"
```

3. Mérési feladatok

1. Végezze el a házi feladat 2. pontja szerinti program szimulációját, és az esetleges hibakeresését és javítását!
2. Végezze el a házi feladat 3. pontja szerinti program szimulációját, és az esetleges hibakeresését és javítását!
3. Oldja meg a mérésvezető által kiadott feladatot, és végezze el a vizsgálatát szimulációval!

4. Házi feladat

1. Korábbi tanulmányai, a mérést előkészítő előadásokon elhangzottak, és a katalógus alapján készüljön fel a következő témákból:
   - PIC18F452
     - Indirekt címzés
     - Időzítők
     - Megszakítások
     - USART és A/D
   - MPLAB szimulátor
     - Stopperóra
     - Gerjesztések (órajelhez kötött, feltételes)

2. Tervezzen és írjon programot PIC18F452-re, amely az USART-ra érkező 0…7 közötti ASCII karaktereknek megfelelően bekapcsolja PORTB 0…7. bitjét, a többit pedig kioltja. (Ha nem a tartományba eső karakter érkezik, ne történjen a porton változás.)

3. Tervezzen és írjon programot PIC18F452-re, amely az AN0 analóg bemenet értékétől függően a következő táblázat szerint működik (referencia a tápfeszültség).

Bemeneti feszültség AN0-on	RB0, RB1 értéke
$U_{be} < 0{,}25 \cdot U_{ref}$	0, 0
$0{,}25 \cdot U_{ref} \le U_{be} < 0{,}5 \cdot U_{ref}$	1, 0
$0{,}5 \cdot U_{ref} \le U_{be} < 0{,}75 \cdot U_{ref}$	0, 1
$0{,}75 \cdot U_{ref} \le U_{be}$	1, 1