# Mérési jegyzőkönyv a

3.

laboratóriumi gyakorlatról

**A mérés helyszíne:** QBP110

**A mérés időpontja:** 2016. *március 30.*

**A mérést végezték:** Herjavecz Zoltán, SDE19 mérőcsoport

**A mérést vezető oktató neve:** Széll András

sde19\_3.docx

## Az elvégzett mérési feladatok

# Sávlezárást jelző villanó fénysor

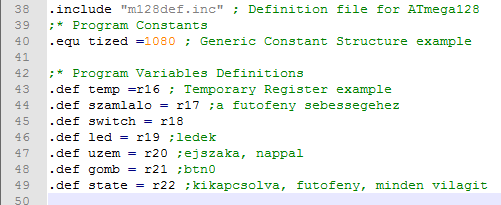
A fénysort a LED-oszlopok jelentik, egy oszlopban mindig csak egy LED világít, így fusson a fény végig rajtuk.

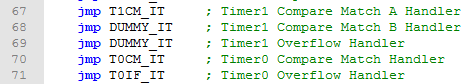
* a léptetés sebessége a SW0…SW3 kapcsolók állásától függően történjen 100 ms időalappal 15 lépésben (100ms…1,5 sec). A kapcsolókat „normál” sorrendben, sorrendhelyesen kezelje (SW0 = LSB).
* a világítás fényerejét az optikai bemenet függvényében fél fényerőre visszaveszi (éjszakai üzem). Nappal a két egymás melletti oszlop egyformán világít (tehát a LED-ek párosával gyulladnak ki és alszanak el), éjszakai üzem során csak a baloldali LED soron futnak végig a fények.
* a készüléket a BT0 nyomógomb segítségével lehet ki-bekapcsolni, három üzemmódja van: kikapcsolva, futófény, minden LED világít

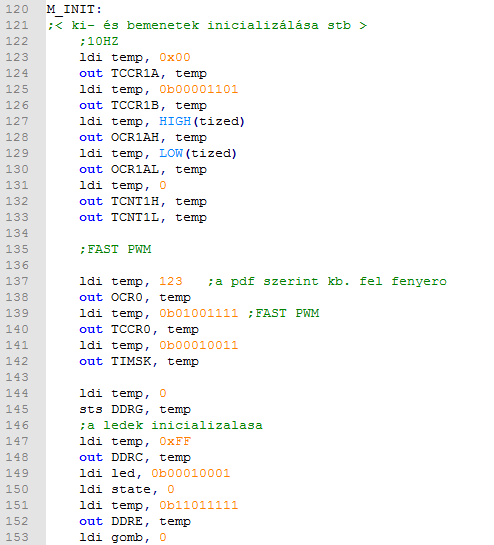
# Megoldás

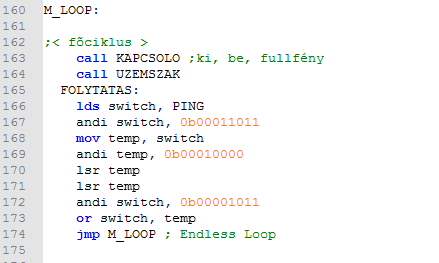
* A feladat első pontját úgy oldottam meg, ahogy az ml2\_avr jegyzetben le van írva a 16 bites Timer/counter1 perifériával, ezzel előállítottam egy 100 ms-os alapot, amin szoftveres időzítővel számolok a switchek miatt
* a világítás fényerejétől függően változik, hogy mennyi LED világít vagy sem. A fél fényerőt szintén a jegyzet alapján készítettem el, egy PWM jelet előállítva a Timer/Counter0 periférivál.
* A btn-nel történő ki/bekapcsolást a main loopban, interruptok használata nélkül implementáltam.

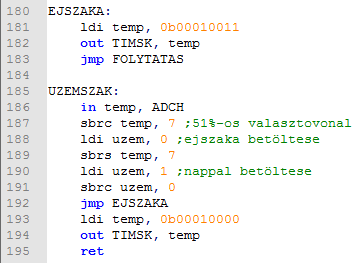
# Forráskód (annak magyarázata)

Létrehoztam egy konstanst, hogy amivel 10 Hz-re osztom le az órajelet, ezzel segítve a 0,1 sec-undumonként befutó IT-t, a többi regiszterért a kommentek beszélnek.

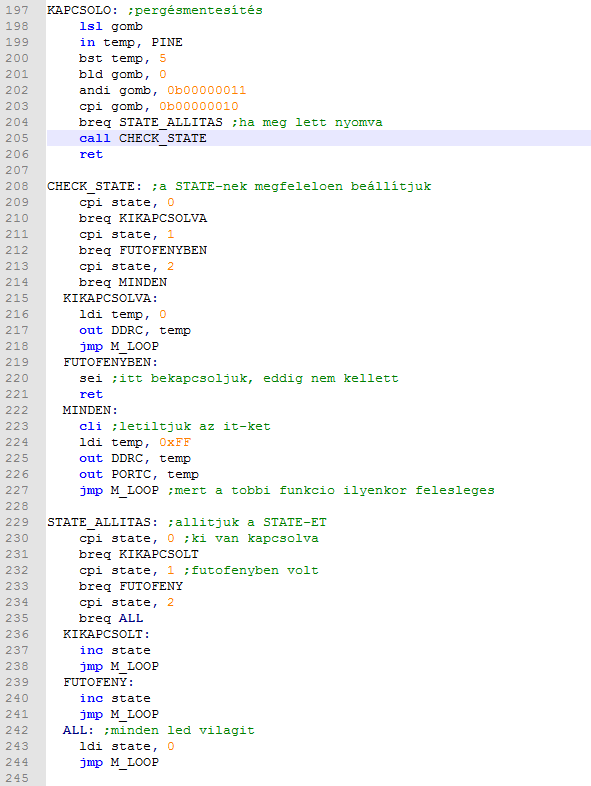
Bejegyeztem az IT szubrutinokat a megszakítási vektortáblába.

Először a jegyzetnek megfelelően konfigurálom a Timer0 perifériát, aztán a Fast PWM jelhez (amitől várhatóan fele olyan erősen világít a LED), aztán beállítom az I/O-kat, illetve inicializálom a regisztereket. A globális interruptot nem itt, hanem máshol engedélyezem, hogy a kikapcsolt illetve folyamatosan világító állapot tökéletesen működjön, ne fussanak közben be interruptok.

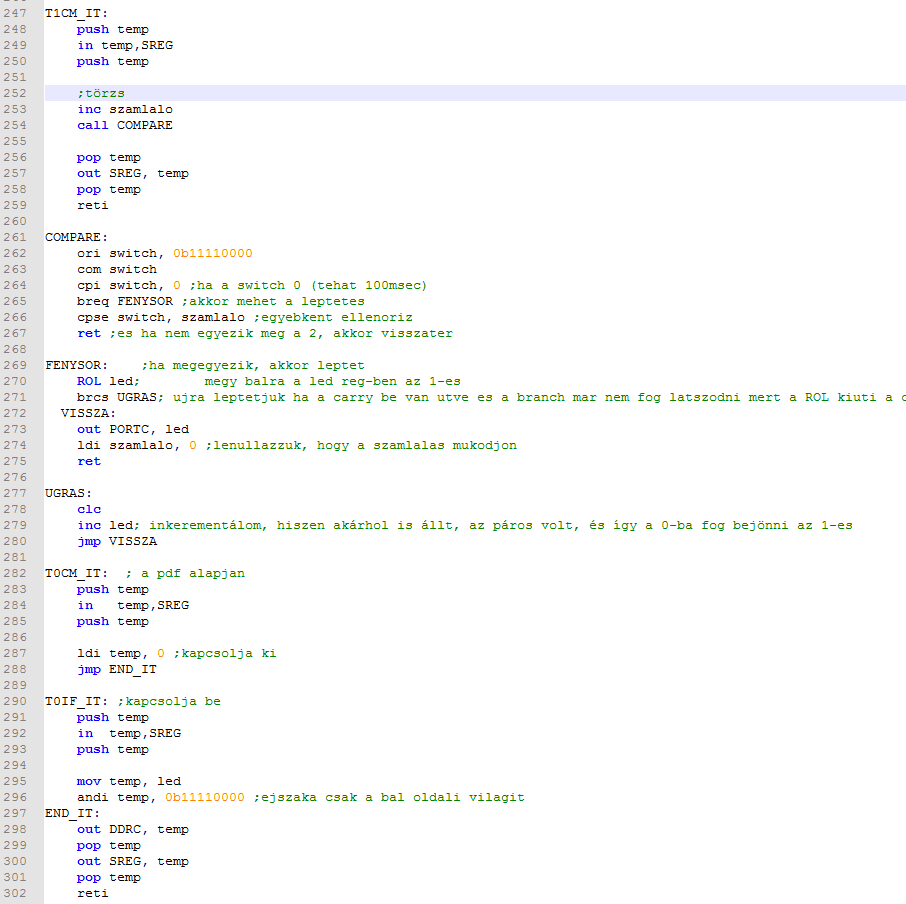
Mint látható a M\_LOOP-ból összesen két szubrutint hívok meg közvetlenül, ezeket ott elmagyarázom, illetve itt kezelem le a switchek beolvasását, megfelelő helyre helyezését.



Itt olvasom be a fényérzékelő állapotát, illetve töltöm be a regiszterbe, hogy éppen nappal vagy éjszaka van-e. Ha éjszaka van, akkor bekapcsolom a PWM megszakításait is, hogy a led gyengébben világítson, ha pedig nappal van, akkor kikapcsolom azt.

Itt pergésmentesítem a gombot első lépésben, kapcsolónak neveztem el a szubrutint, mert ez kapcsolja ki/be a programot kvázi. Ha meg lett nyomva a gomb, akkor beállítom a state regisztert, jelezve hogy most milyen funkciót kell produkáljon a program. Ez lefut minden egyes gombnyomás érzékelése után.

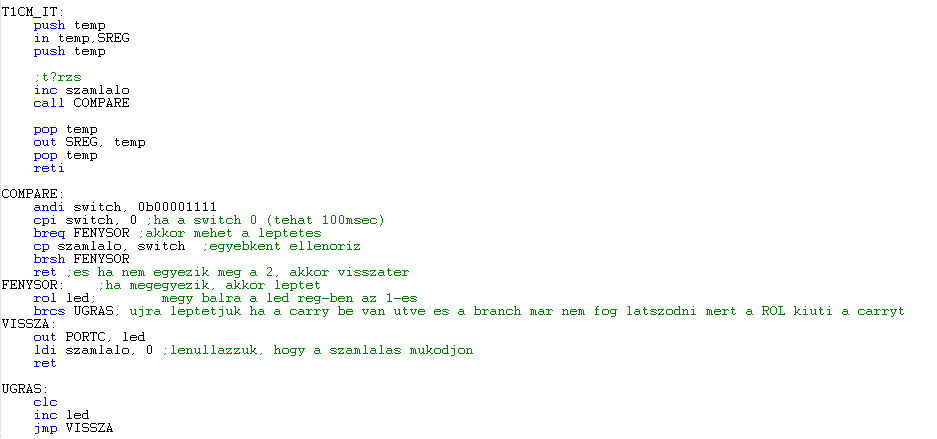
A CHECK\_STATE-ben konfigurálom be a programot a state regiszter alapján, hogy az adott funkciónak megfelelően működjön. A KIKAPCSOLVA illetve MINDEN címkék után azért ugrok rögtön vissza az M\_LOOP-ra, ugyanis akkor teljesen felesleges minden mást olvasni/állítani, ugyanis nem függ semmitől. A futófényben pedig engedélyezem a globális interruptokat, hogy az a funkció működjön, hiszen azt interruptok, hardver időzítő állítja elő.

A T1CM\_IT interrupt fut le 100 msec-onként, itt növelem a szoftver számlálót, és nullázom a switchek állásától függően, hiszen a switchek állásáig számol csak el. A switchekről beolvasott értékek komplementerét veszem, hiszen azok fordítva „működnek”, és a számlálós összehasonlítás miatt jobb így. A többihez pedig a kommentek bőségesen elegek. A következő ami érdekes lehet, az a Timer0 pedig a PWM jelet állítja elő, amivel a ledek fényereje lesz gyengébb. A bekapcsolásnál beletettem egy ANDI-t, hogy csak a bal oldali fénysor világítson, más érzésem szerint nem szorul magyarázatra.

# Szimuláció

Az utolsó verzió, amelyben minden funkció implementálva van, már nem működik ISIS proteussal. Korábban még szimplán a nappali futófény működött.

# Órai munka



A fent látható részét módosítani kellett a kódnak, így majdnem tökéletesen fut, kivéve amikor az SW2 van beállítva, akkor ugyanis nem egyenletes a futás.

Hibák:

T1CM\_IT:

push temp

in temp,SREG

push temp

;törzs

inc szamlalo

call COMPARE

pop temp

out SREG, temp

pop temp

reti

COMPARE:

ori switch, 0b11110000

com switch

cpi switch, 0 ;ha a switch 0 (tehat 100msec)

breq FENYSOR ;akkor mehet a leptetes

cpse switch, szamlalo ;egyebkent ellenoriz

ret ;es ha nem egyezik meg a 2, akkor visszater

FENYSOR: ;ha megegyezik, akkor leptet

ROL led; megy balra a led reg-ben az 1-es

brcs UGRAS; ujra leptetjuk ha a carry be van utve es a branch mar nem fog latszodni mert a ROL kiuti a carryt

VISSZA:

out PORTC, led

ldi szamlalo, 0 ;lenullazzuk, hogy a szamlalas mukodjon

ret

UGRAS:

clc

inc led

jmp VISSZA

1. Itt az egyik hiba a breq utasítással volt, ugyanis előfordult, hogy túlfutott a számláló, amit nem ellenőriztem és ezért meg kellett várni, hogy túlcsorduljon a számláló.
2. a másik hiba az volt, hogy az IT-ben hívtam meg a COMPARE-t, ami miatt előfordult olyan állapot, hogy a switch-ről beolvasott értékeket még nem rendeztem a helyükre és ezért nem mindig egyenletesen futottak a ledek.

A javított teljes kód:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Feladat:

;\* R?vid le?r?s:

;

;\* Szerz?k:

;\* M?r?csoport: <merocsoport jele>

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* "AVR ExperimentBoard" port assignment information:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* LED0(P):PortC.0 LED4(P):PortC.4

;\* LED1(P):PortC.1 LED5(P):PortC.5

;\* LED2(S):PortC.2 LED6(S):PortC.6

;\* LED3(Z):PortC.3 LED7(Z):PortC.7 INT:PortE.4

;\*

;\* SW0:PortG.0 SW1:PortG.1 SW2:PortG.4 SW3:PortG.3

;\*

;\* BT0:PortE.5 BT1:PortE.6 BT2:PortE.7 BT3:PortB.7

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* AIN:PortF.0 NTK:PortF.1 OPTO:PortF.2 POT:PortF.3

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* LCD1(VSS) = GND LCD9(DB2): -

;\* LCD2(VDD) = VCC LCD10(DB3): -

;\* LCD3(VO ) = GND LCD11(DB4): PortA.4

;\* LCD4(RS ) = PortA.0 LCD12(DB5): PortA.5

;\* LCD5(R/W) = GND LCD13(DB6): PortA.6

;\* LCD6(E ) = PortA.1 LCD14(DB7): PortA.7

;\* LCD7(DB0) = - LCD15(BLA): VCC

;\* LCD8(DB1) = - LCD16(BLK): PortB.5 (1=Backlight ON)

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m128def.inc" ; Definition file for ATmega128

;\* Program Constants

.equ tized =1080 ; Generic Constant Structure example

;\* Program Variables Definitions

.def temp =r16 ; Temporary Register example

.def szamlalo = r17

.def switch = r18

.def led = r19

.def uzem = r20

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Reset & Interrupt Vectors

.cseg

.org $0000 ; Define start of Code segment

jmp RESET ; Reset Handler, jmp is 2 word instruction

jmp DUMMY\_IT ; Ext. INT0 Handler

jmp DUMMY\_IT ; Ext. INT1 Handler

jmp DUMMY\_IT ; Ext. INT2 Handler

jmp DUMMY\_IT ; Ext. INT3 Handler

jmp DUMMY\_IT ; Ext. INT4 Handler (INT gomb)

jmp DUMMY\_IT ; Ext. INT5 Handler

jmp DUMMY\_IT ; Ext. INT6 Handler

jmp DUMMY\_IT ; Ext. INT7 Handler

jmp DUMMY\_IT ; Timer2 Compare Match Handler

jmp DUMMY\_IT ; Timer2 Overflow Handler

jmp DUMMY\_IT ; Timer1 Capture Event Handler

jmp T1CM\_IT ; Timer1 Compare Match A Handler

jmp DUMMY\_IT ; Timer1 Compare Match B Handler

jmp DUMMY\_IT ; Timer1 Overflow Handler

jmp DUMMY\_IT ; Timer0 Compare Match Handler

jmp DUMMY\_IT ; Timer0 Overflow Handler

jmp DUMMY\_IT ; SPI Transfer Complete Handler

jmp DUMMY\_IT ; USART0 RX Complete Handler

jmp DUMMY\_IT ; USART0 Data Register Empty Hanlder

jmp DUMMY\_IT ; USART0 TX Complete Handler

jmp DUMMY\_IT ; ADC Conversion Complete Handler x

jmp DUMMY\_IT ; EEPROM Ready Hanlder

jmp DUMMY\_IT ; Analog Comparator Handler

jmp DUMMY\_IT ; Timer1 Compare Match C Handler

jmp DUMMY\_IT ; Timer3 Capture Event Handler

jmp DUMMY\_IT ; Timer3 Compare Match A Handler

jmp DUMMY\_IT ; Timer3 Compare Match B Handler

jmp DUMMY\_IT ; Timer3 Compare Match C Handler

jmp DUMMY\_IT ; Timer3 Overflow Handler

jmp DUMMY\_IT ; USART1 RX Complete Handler

jmp DUMMY\_IT ; USART1 Data Register Empty Hanlder

jmp DUMMY\_IT ; USART1 TX Complete Handler

jmp DUMMY\_IT ; Two-wire Serial Interface Handler

jmp DUMMY\_IT ; Store Program Memory Ready Handler

.org $0046

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* DUMMY\_IT interrupt handler -- CPU hangup with LED pattern

;\* (This way unhandled interrupts will be noticed)

;< t?bbi IT kezel? a f?jl v?g?re! >

DUMMY\_IT:

ldi r16, 0xFF ; LED pattern: \*-

out DDRC, r16 ; -\*

ldi r16, 0xA5 ; \*-

out PORTC, r16 ; -\*

DUMMY\_LOOP:

rjmp DUMMY\_LOOP ; endless loop

;< t?bbi IT kezel? a f?jl v?g?re! >

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* MAIN program, Initialisation part

.org $004B;

RESET:

;\* Stack Pointer init,

; Set stack pointer to top of RAM

ldi temp, LOW(RAMEND) ; RAMEND = "max address in RAM"

out SPL, temp ; RAMEND value in "m128def.inc"

ldi temp, HIGH(RAMEND)

out SPH, temp

M\_INIT:

;< ki- ?s bemenetek inicializ?l?sa stb >

ldi temp, 0x00

out TCCR1A, temp

ldi temp, 0b00001101

out TCCR1B, temp

ldi temp, HIGH(tized)

out OCR1AH, temp

ldi temp, LOW(tized)

out OCR1AL, temp

ldi temp, 0

out TCNT1H, temp

out TCNT1L, temp

ldi temp, 0b00010000

out TIMSK, temp

ldi temp, 0

sts DDRG, temp

;a ledek inicializalasa

ldi temp, 0xFF

out DDRC, temp

ldi led, 0b00010001

out PORTC, led

sei

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* MAIN program, Endless loop part

M\_LOOP:

;< f?ciklus >

lds switch, PING

andi switch, 0b00011011

mov temp, switch

andi temp, 0b00010000

lsr temp

lsr temp

andi switch, 0b00001011

or switch, temp

call COMPARE

in temp, ADCH

sbrc temp, 7 ;51%-os valasztovonal

ldi uzem, 0

sbrs temp, 7

ldi uzem, 1

jmp M\_LOOP ; Endless Loop

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Subroutines, Interrupt routines

COMPARE:

andi switch, 0b00001111

cpi switch, 0 ;ha a switch 0 (tehat 100msec)

breq EGYEZES ;akkor egybe allitom

FOLYTAT:

cp szamlalo, switch ;egyebkent ellenoriz

brsh FENYSOR

ret ;es ha nem egyezik meg a 2, akkor visszater

FENYSOR: ;ha megegyezik, akkor leptet

rol led; megy balra a led reg-ben az 1-es

brcs UGRAS; ujra leptetjuk ha a carry be van utve es a branch mar nem fog latszodni mert a ROL kiuti a carryt

VISSZA:

out PORTC, led

ldi szamlalo, 0 ;lenullazzuk, hogy a szamlalas mukodjon

ret

UGRAS:

clc

inc led

jmp VISSZA

EGYEZES:

ldi switch, 1

jmp FOLYTAT

T1CM\_IT:

push temp

in temp,SREG

push temp

;torzs

inc szamlalo

pop temp

out SREG, temp

pop temp

reti

Illetve a kezdetleges állapothoz képest még átírtam, hogy ne a switchek komparáltjával hasonlítsuk össze őket. De ez eddig csak a villanó fénysor funkció volt, most következik a teljes program.

FONTOS!! Az eredeti kódban a túlbonyolított feladat megoldása van, ami PWM-mel dolgozik, miközben a félfényerő csak arra vonatkozott, hogy csak az egyik ledsor világítson.

Sikerült a teljes funkcionalitást életre kelteni, a következő a kód:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Feladat:

;\* R?vid le?r?s:

;

;\* Szerz?k:

;\* M?r?csoport: <merocsoport jele>

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* "AVR ExperimentBoard" port assignment information:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* LED0(P):PortC.0 LED4(P):PortC.4

;\* LED1(P):PortC.1 LED5(P):PortC.5

;\* LED2(S):PortC.2 LED6(S):PortC.6

;\* LED3(Z):PortC.3 LED7(Z):PortC.7 INT:PortE.4

;\*

;\* SW0:PortG.0 SW1:PortG.1 SW2:PortG.4 SW3:PortG.3

;\*

;\* BT0:PortE.5 BT1:PortE.6 BT2:PortE.7 BT3:PortB.7

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* AIN:PortF.0 NTK:PortF.1 OPTO:PortF.2 POT:PortF.3

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* LCD1(VSS) = GND LCD9(DB2): -

;\* LCD2(VDD) = VCC LCD10(DB3): -

;\* LCD3(VO ) = GND LCD11(DB4): PortA.4

;\* LCD4(RS ) = PortA.0 LCD12(DB5): PortA.5

;\* LCD5(R/W) = GND LCD13(DB6): PortA.6

;\* LCD6(E ) = PortA.1 LCD14(DB7): PortA.7

;\* LCD7(DB0) = - LCD15(BLA): VCC

;\* LCD8(DB1) = - LCD16(BLK): PortB.5 (1=Backlight ON)

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m128def.inc" ; Definition file for ATmega128

;\* Program Constants

.equ tized =1080 ; Generic Constant Structure example

;\* Program Variables Definitions

.def temp =r16 ; Temporary Register example

.def szamlalo = r17 ;a futofeny sebessegehez

.def switch = r18

.def led = r19 ;ledek

.def uzem = r20 ;ejszaka, nappal

.def gomb = r21 ;btn0

.def state = r22 ;kikapcsolva, futofeny, minden vilagit

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Reset & Interrupt Vectors

.cseg

.org $0000 ; Define start of Code segment

jmp RESET ; Reset Handler, jmp is 2 word instruction

jmp DUMMY\_IT ; Ext. INT0 Handler

jmp DUMMY\_IT ; Ext. INT1 Handler

jmp DUMMY\_IT ; Ext. INT2 Handler

jmp DUMMY\_IT ; Ext. INT3 Handler

jmp DUMMY\_IT ; Ext. INT4 Handler (INT gomb)

jmp DUMMY\_IT ; Ext. INT5 Handler

jmp DUMMY\_IT ; Ext. INT6 Handler

jmp DUMMY\_IT ; Ext. INT7 Handler

jmp DUMMY\_IT ; Timer2 Compare Match Handler

jmp DUMMY\_IT ; Timer2 Overflow Handler

jmp DUMMY\_IT ; Timer1 Capture Event Handler

jmp T1CM\_IT ; Timer1 Compare Match A Handler

jmp DUMMY\_IT ; Timer1 Compare Match B Handler

jmp DUMMY\_IT ; Timer1 Overflow Handler

jmp T0CM\_IT ; Timer0 Compare Match Handler

jmp T0IF\_IT ; Timer0 Overflow Handler

jmp DUMMY\_IT ; SPI Transfer Complete Handler

jmp DUMMY\_IT ; USART0 RX Complete Handler

jmp DUMMY\_IT ; USART0 Data Register Empty Hanlder

jmp DUMMY\_IT ; USART0 TX Complete Handler

jmp DUMMY\_IT ; ADC Conversion Complete Handler

jmp DUMMY\_IT ; EEPROM Ready Hanlder

jmp DUMMY\_IT ; Analog Comparator Handler

jmp DUMMY\_IT ; Timer1 Compare Match C Handler

jmp DUMMY\_IT ; Timer3 Capture Event Handler

jmp DUMMY\_IT ; Timer3 Compare Match A Handler

jmp DUMMY\_IT ; Timer3 Compare Match B Handler

jmp DUMMY\_IT ; Timer3 Compare Match C Handler

jmp DUMMY\_IT ; Timer3 Overflow Handler

jmp DUMMY\_IT ; USART1 RX Complete Handler

jmp DUMMY\_IT ; USART1 Data Register Empty Hanlder

jmp DUMMY\_IT ; USART1 TX Complete Handler

jmp DUMMY\_IT ; Two-wire Serial Interface Handler

jmp DUMMY\_IT ; Store Program Memory Ready Handler

.org $0046

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* DUMMY\_IT interrupt handler -- CPU hangup with LED pattern

;\* (This way unhandled interrupts will be noticed)

;< t?bbi IT kezel? a f?jl v?g?re! >

DUMMY\_IT:

ldi r16, 0xFF ; LED pattern: \*-

out DDRC, r16 ; -\*

ldi r16, 0xA5 ; \*-

out PORTC, r16 ; -\*

DUMMY\_LOOP:

rjmp DUMMY\_LOOP ; endless loop

;< t?bbi IT kezel? a f?jl v?g?re! >

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* MAIN program, Initialisation part

.org $004B;

RESET:

;\* Stack Pointer init,

; Set stack pointer to top of RAM

ldi temp, LOW(RAMEND) ; RAMEND = "max address in RAM"

out SPL, temp ; RAMEND value in "m128def.inc"

ldi temp, HIGH(RAMEND)

out SPH, temp

M\_INIT:

;< ki- es bemenetek inicializalasa stb >

;10HZ

ldi temp, 0x00

out TCCR1A, temp

ldi temp, 0b00001101

out TCCR1B, temp

ldi temp, HIGH(tized)

out OCR1AH, temp

ldi temp, LOW(tized)

out OCR1AL, temp

ldi temp, 0

out TCNT1H, temp

out TCNT1L, temp

;FAST PWM

;ldi temp, 123 ;a pdf szerint kb. fel fenyero

;out OCR0, temp

;ldi temp, 0b01001111 ;FAST PWM

; out TCCR0, temp

ldi temp, 0b00010000

out TIMSK, temp

ldi temp, 0

sts DDRG, temp

;a ledek inicializalasa

ldi temp, 0xFF

out DDRC, temp

ldi led, 0b00010001

ldi state, 0

ldi temp, 0b11011111

out DDRE, temp

ldi gomb, 0

ldi temp, 0b01100010

out ADMUX, temp

ldi temp, 0b11100111

out ADCSRA, temp

sei

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* MAIN program, Endless loop part

M\_LOOP:

;< fOciklus >

cpi gomb, 0b00000010 //TIMER0\_IT-bol frissitve

brne CONTINUE

call STATE\_ALLITAS ;ha meg lett nyomva, kikapcsolva, futofeny, minden vilagit

CONTINUE:

call CHECK\_STATE

call UZEMSZAK

FOLYTATAS:

lds switch, PING

andi switch, 0b00011011

mov temp, switch

andi temp, 0b00010000

lsr temp

lsr temp

andi switch, 0b00001011

or switch, temp

call COMPARE

jmp M\_LOOP ; Endless Loop

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Subroutines, Interrupt routines

COMPARE:

andi switch, 0b00001111

cpi switch, 0 ;ha a switch 0 (tehat 100msec)

breq EGYEZES ;akkor egybe allitom

FOLYTAT:

cp szamlalo, switch ;egyebkent ellenoriz

brsh FENYSOR

ret ;es ha nem egyezik meg a 2, akkor visszater

FENYSOR: ;ha megegyezik, akkor leptet

rol led; megy balra a led reg-ben az 1-es

brcs UGRAS; ujra leptetjuk ha a carry be van utve es a branch mar nem fog latszodni mert a ROL kiuti a carryt

VISSZA:

mov temp, led //a maszkolas miatt

cpi uzem, 0

breq EJSZAKA

EJVISSZA:

out PORTC, temp

ldi szamlalo, 0 ;lenullazzuk, hogy a szamlalas mukodjon

ret

UGRAS:

clc

inc led; inkerementalom, hiszen akarhol is allt, az paros volt, es igy a 0-ba fog bejonni az 1-es

jmp VISSZA

EGYEZES:

ldi switch, 1

jmp FOLYTAT

EJSZAKA:

andi temp, 0x0F

JMP EJVISSZA

UZEMSZAK:

in temp, ADCH

;out PORTC, temp

cpi temp, 64

brlo NAPPAL

BACK:

cpi temp, 64

brsh EJSZAK

BACKEJ:

;sbrc temp, 6 ;51%-os valasztovonal

;ldi uzem, 0 ;ejszaka betoltese

;sbrs temp, 6

;ldi uzem, 1 ;nappal betoltese

; jmp EJSZAKA

; ldi temp, 0xFF

; out DDRC, temp

ret

NAPPAL:

ldi uzem, 1

jmp BACK

EJSZAK:

ldi uzem, 0

jmp BACKEJ

GOMB\_BEOLVASAS: ;pergesmentesites

lsl gomb

in temp, PINE

bst temp, 5

bld gomb, 0

andi gomb, 0b00000011

ret

CHECK\_STATE: ;a STATE-nek megfeleloen beallitjuk

cpi state, 0

breq KIKAPCSOLVA

cpi state, 1

breq FUTOFENYBEN

cpi state, 2

breq MINDEN

KIKAPCSOLVA:

ldi temp, 0

out PORTC, temp

ret

FUTOFENYBEN:

ret

MINDEN:

ldi temp, 0xFF

out DDRC, temp

out PORTC, temp

ret ;mert a tobbi funkcio ilyenkor felesleges

STATE\_ALLITAS: ;allitjuk a STATE-ET

ldi gomb, 0

cpi state, 0 ;ki van kapcsolva

breq KIKAPCSOLT

cpi state, 1 ;futofenyben volt

breq FUTOFENY

cpi state, 2

breq ALL

KIKAPCSOLT:

inc state

ret

FUTOFENY:

inc state

ret

ALL: ;minden led vilagit

ldi state, 0

ret

T1CM\_IT:

push temp

in temp,SREG

push temp

;torzs

inc szamlalo

call GOMB\_BEOLVASAS

pop temp

out SREG, temp

pop temp

reti

T0CM\_IT: ; a pdf alapjan

push temp

in temp,SREG

push temp

ldi temp, 0 ;kapcsolja ki

jmp END\_IT

T0IF\_IT: ;kapcsolja be

push temp

in temp,SREG

push temp

mov temp, led

andi temp, 0b11110000 ;ejszaka csak a bal oldali vilagit

END\_IT:

out DDRC, temp

pop temp

out SREG, temp

pop temp

reti

Az eredeti kódhoz képest változtatni kellett az opto érzékelő kezelésén, inicializálásán, a pergésmentesítés ellenőrző át lett helyezve az IT rutinba, illetve rengeteg helyen jmp-pal tértem vissza call után, tehát a stack pointer teljesen rossz helyre mutatott állandóan. Illetve az ejszakai mód kezelése kicsit máshogy történik, most a napszaktól függűen a PORTC-re való kiírás előtt maszkolom a ledeket.

Folyamatábra:

