Mikropočítačové Systémy
MIPS
Distribuované vnorené počítačové systémy
Distributed Embedded Computer System
(Microcontrollers)

Prednáška 12.
Mix.

Venované všetkým tým, ktorí si prečítali knihu TOM-a PHILIPS-a ĽUDSTVO ....
a poznajú zákon zachovania „čohokoľvek"

(správny výsledok dostaneme až vtedy, keď urobíme niekoľko chýb)
A AJ NAPRIEK
tomu sa pokúsia urobiť niečo nové, niečo naviac ...

1

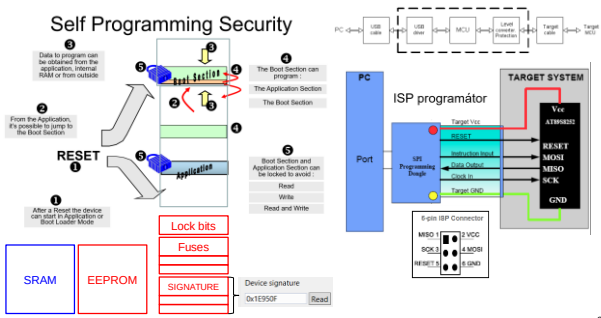
KL

Má 294 strán a my sme z nich prelistovali len niekoľko. Získali sme prehľad o možnostiach ATMEGA 238P a základné vedomosti ako použiť ARDUINO UNO a niekoľkých „programov".

Ak budeme chcieť navrhnúť niečo pre prax a má to prežiť (byť úspešné), musí to vedieť pracovať aj v zarušenom prostredí, musí to mať schopnosť adaptovať sa na zmeny prostredia, ... a v neposlednom rade to musí odolať útokom vykrádačov dobrých nápadov.

Ináč povedané za málo peňazí veľa muziky.
To čo nasleduje, by mohlo byť náplňou na semester.

2



0x1E vyrobené Atmel-om

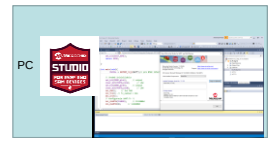
3

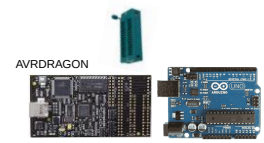
Na poslednom cvičení sme identifikovali RC člen ako regulovanú sústavu a niektorí aj navrhli P regulátor.
Z KL ATMEGA 328P sa dá vyčítať, že OSC môže byť
Kryštál, napr.: 16MHz. Jeho cena (0,3Eura) je len o rád menšia ako cena procesora ATMEGA 328P (< 3Eura).
Plocha kryštálu nie je zanedbateľná.

Pri veľkých sériách, zníženie ceny o percentá a pracnosti výroby hrá veľkú úlohu. Z tohto dôvodu sa pokúsime vynechať (jednoducho) – nepoužiť kryštálový oscilátor a použiť zabudovaný RC oscilátor, ktorý je naviac aj kalibrovaný.

T.j. musíme si podrobnejšie preštudovať kapitolu KL: **8. System Clock and Clock Options**
27. Memory Programming
. . . a niekoľko ďalších kapitol.

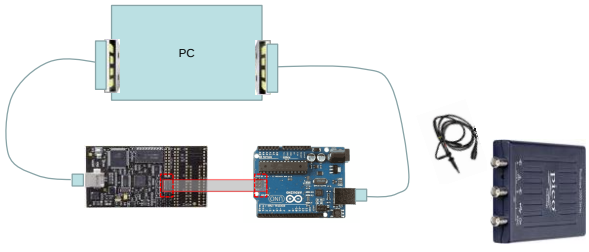
Ako časom zistíme, problém je v tom, že ext. Osc. Je 16MHz a interný RC je 8MHz, a treba ho kalibrovať.
Použijeme.





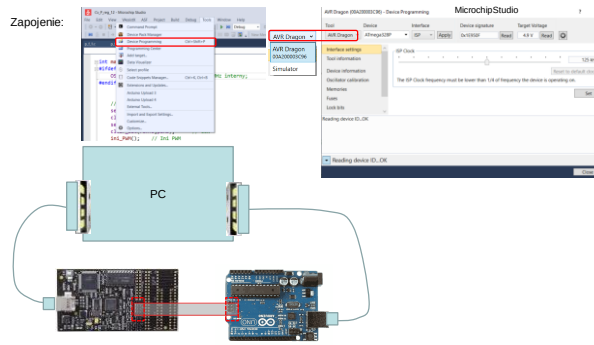
4

Zapojenie:



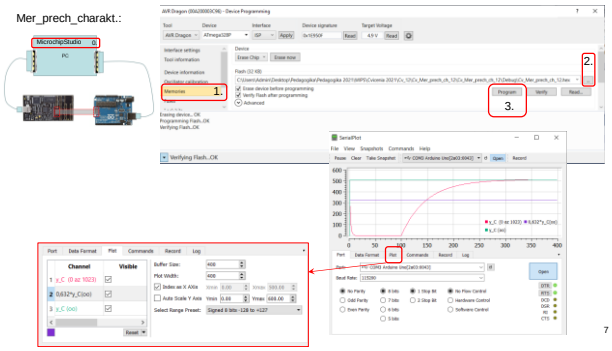
5

Zapojenie:



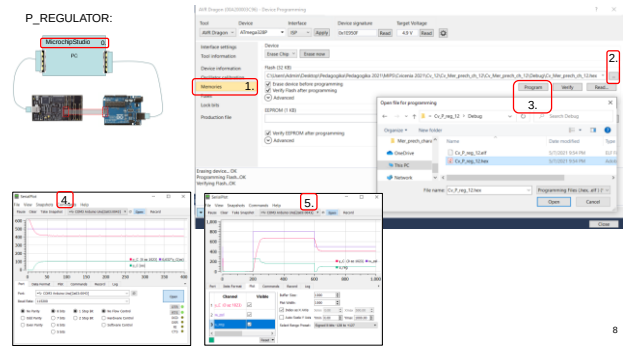
6

Mer_prech_charakt.:



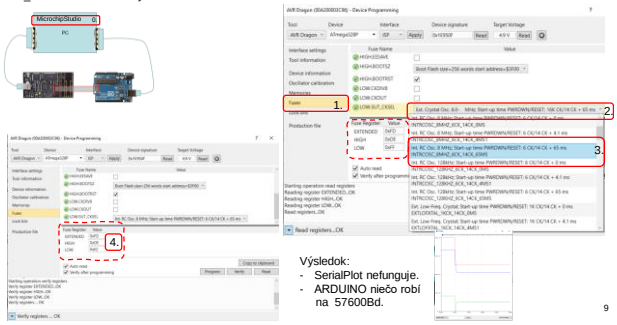
7

P_REGULATOR:



8

P_REGULATOR:Ext. Kryst. Osc. 16 MHz ⟶ Int. RC. Osc. 8MHz



Výsledok:
- SerialPlot nefunguje.
- ARDUINO niečo robí na 57600Bd.

9

P_REGULATOR:Ext. Kryst. Osc. 16 MHz ⟶ Int. RC. Osc. 8MHz

P R E P R O G R A M U J E M E, pridáme časti programu:

```
/* p_f_1.h*/                    1.
...
#ifdef avr_dragon_1
#define F_CPU 8000000UL
#else
#define F_CPU 16000000UL
#endif
```

Výsledok: Nefunguje.

Pridáme

a) RC kalibračnú konštantu.

b) Pridáme funkciu

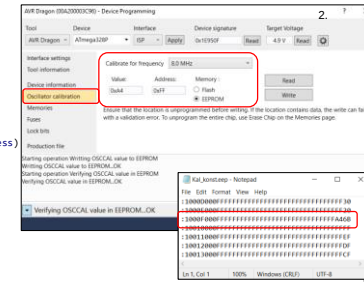
```
char EEPROM_rd_B(unsigned int uiAddress)
```

c) Vyčítame RC kalibračnú konštantu

d) Uložíme do EEPROM na adresu 0x??.

e) Doplníme main program o

```
OSCCAL = EEPROM_rd_B(0x??);
// pre 8MHz interny;
```



10

P_REGULATOR:Ext. Kryst. Osc. 16 MHz ⟶ Int. RC. Osc. 8MHz

P R E P R O G R A M U J E M E, pridáme časti programu:

```
/* p_f_1.h*/                    1.
...
#ifdef avr_dragon_1
#define F_CPU 8000000UL
#else
#define F_CPU 16000000UL
#endif
```

Výsledok: Nefunguje.

Pridáme

a) RC kalibračnú konštantu.

b) Pridáme funkciu (do p_f_1.c)

```
char EEPROM_rd_B(unsigned int uiAddress)
```

c) Vyčítame RC kalibračnú konštantu

d) Uložíme do EEPROM na adresu 0x???.

e) Doplníme main program o

```
OSCCAL = EEPROM_rd_B(0x???);
// pre 8MHz interny;
```

f) Opäť preprogramujeme.

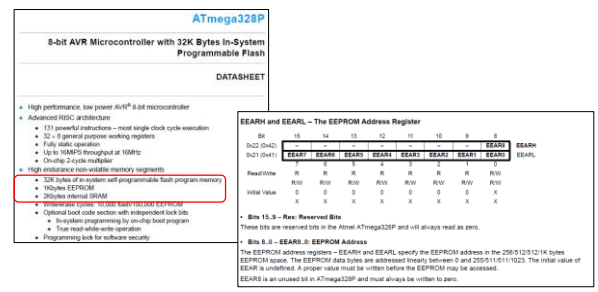
g) A funguje.

```
#ifdef avr_dragon_2                2.
 char EEPROM_rd_B(unsigned int uiAddress){
   while(EECR & (1 << EEPE));//WDR();
   EEAR = uiAddress;
   set_bit(EECR,EERE);
   return EEDR;
}
#endif
```

```
int main(void){                    3.
#ifdef avr_dragon_2
OSCCAL = EEPROM_rd_B(0xff);
// pre 8MHz interny;
#endif
. . .
```

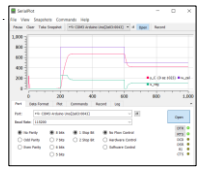
11

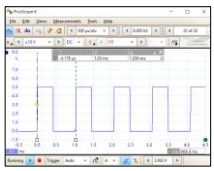
Zaujímavosti KL:



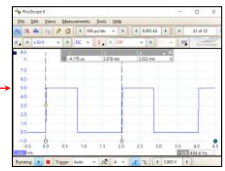
12

Teraz už všetko opäť funguje, ale . .





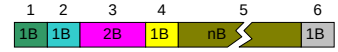
PWM pôvodne teraz

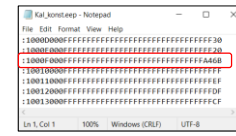


13

Intel HEX formát. [IntelHexFormat.pdf]

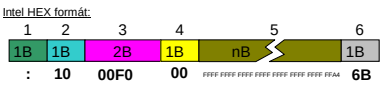
Intel HEX je formát súboru, ktorý obsahuje hexadecimálny kód programu alebo iných dát určených pre mikroprocesory a pamäte. Je to jeden z najstarších formátov používaných v tejto oblasti. V podstate sa jedná o textový súbor kde každý riadok predstavuje postupnosť hexadecimálnych hodnôt . Existujú 3 typy IHEX súborov a to 8B, 16B a 32B, ktoré sa líšia maximálnym počtom dát v jednom riadku . Každý riadok pozostáva zo šiestich častí:

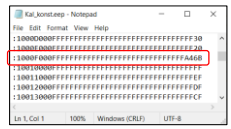




14

Intel HEX formát:





1.Štart znak, jeden znak „:".

2.Počet n Byte-ov dát v riadku, dvojica hexadecimálnych znakov.

3.Adresa určujúca posunutie prvého byte-u dát v riadku, dve dvojice hexadecimálnych znakov.

4. Typ záznamu s hodnotami od 00 do 05, dvojica hexadecimálnych znakov.

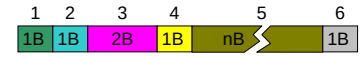
- 00 dáta v riadku sú určené pre pamäť mikroprocesora.
- 01 koniec súboru, typicky má tvar :00000001FF.
- 02 začiatok rozšírenej pamäti; dáta za týmto riadkom majú adresu posúvanú nie od začiatku 0x0000 hexa ale od 0x10000 hexa (pre pamäte väčšie ako 64KB) .
- Ostatné hodnoty sa vyskytujú iba pri 32byt-ovom type IHEX súboru a znamenajú ďalšie rozšírenie pamäte.

5. Dáta, 2*n hexadecimálnych znakov

6. Kontrolná suma = negácia {(suma všetkých dvojíc znakov v riadku, okrem prvého prevedených na binárnu hodnotu) mod (256)} + 1, (ak je výsledkom číslo 0x100, potom kontrolná suma = 0x00), dvojica hexadecimálnych znakov.

15

Príklad, kontrolná suma:



:10010000 21 46 01 36 01 21 47 01 36 00 7E FE 09 D2 19 01 40

: - prvý znak v riadku, začiatok záznamu.

10 - záznam obsahuje 10 hexa = 16 decimálne Byte-ov dát .

0100 - dáta v riadku majú začínať od adresy 0x0100.

00 - dáta v riadku sú určené pre zápis do pamäte mikroprocesora.

21 46 ... D2 19 01 - 32 znakov = 16B dát.

40 - 0x10 + 0x01 + 0x00 + 0x00 + 0x21 + ... + 0x19 + 0x01 = 0x3C0;

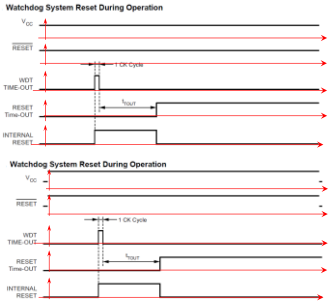
0x3C0 modulo 0x100 = 0xC0;

negácia 0xC0 = 0x3F;

0x3F + 1 = 0x40.

16

WDT:

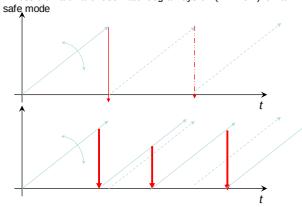


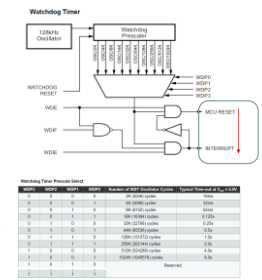
WDT: Začneme tradične: Niečím iným. Na obr. je priebeh signálov odpovedajúci externému RST-u. Treba upresniť „počiatok".

17

WDT:

- Clocked from separate on-chip oscillator
- 3 operating modes
  - Interrupt
  - System reset
  - Interrupt and system reset
- Selectable time-out period from 16ms to 8s
- Possible hardware fuse watchdog always on (WDTON) for fail-safe mode





18

Povolíme WDT: Fuse WDTON „naprogramujeme“ ? A začnú sa diať čudné veci. „RST“.

Watchdog Timer Configuration

WDTON(1)	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt mode	Interrupt
1	1	0	System reset mode	Reset
1	1	1	Interrupt and system reset mode	Interrupt, then go to system reset mode
0	x	x	System reset mode	Reset

Note: 1. WDTON fuse set to "0" means programmed and "1" means unprogrammed.

WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

The sequence for clearing WDE and changing time-out configuration is as follows:
1. In the same operation, write a logic one to the watchdog change enable bit (WDCE) and WDE. A logic one must
be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and watchdog prescaler bits (WDP) as desired, but with the
WDCE bit cleared. This must be done in one operation.

MCUSR – MCU Status Register

The MCU status register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 3 – WDRF: Watchdog System Reset Flag**

This bit is set if a watchdog system reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

MCUSR – MCU Status Register

The MCU status register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 3 – WDRF: Watchdog System Reset Flag**

This bit is set if a watchdog system reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a brown-out reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an external reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a power-on reset occurs. The bit is reset only by writing a logic zero to the flag.