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 chyb)
A AJ NAPRIEK
tomu sa pokúsia urobiť niečo nové, niečo naviac ...

1

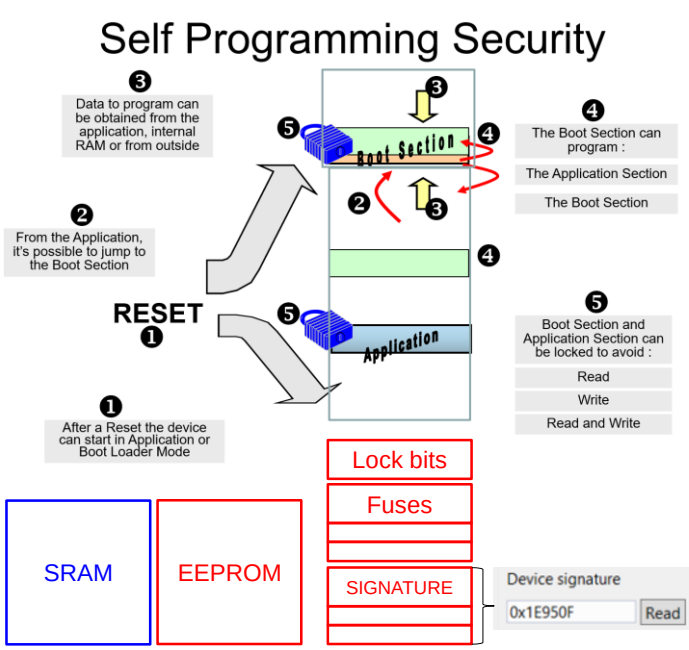
KL

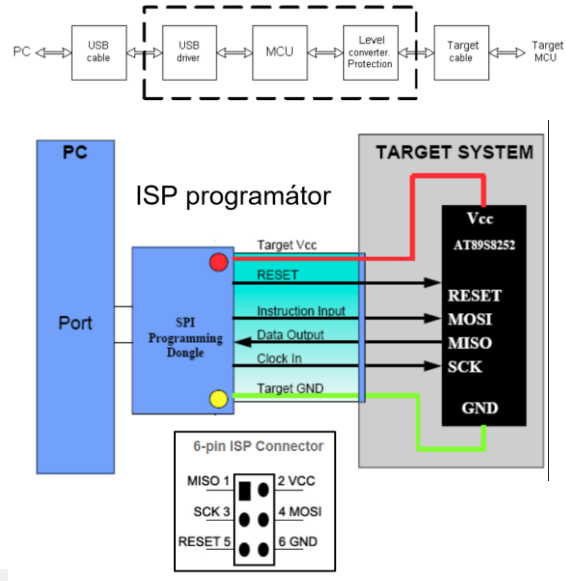
Má 294 strán a my sme z nich prelistovali len niekoľko. Získali sme prehľad o možnostiach ATMEGA 238P a základné vedomosti ako použiť ARDUINO UNO a niekoľkých „programov“.

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

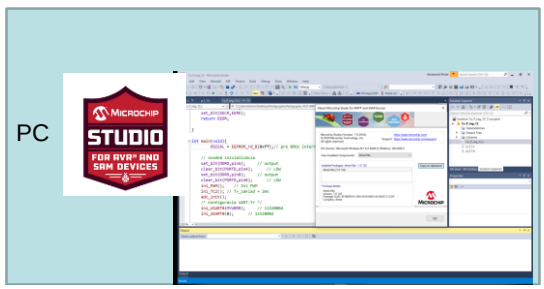
3

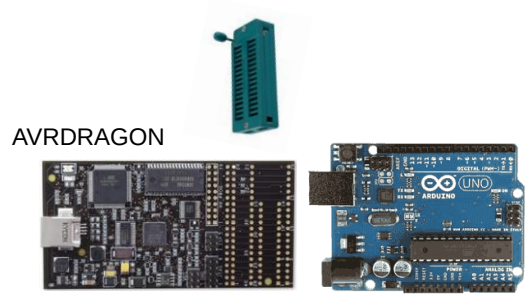
Na poslednom cvičení sme identifikovali RC člen ako regulovanú sústavu a niektorí aj navrhli P regulátor.
Z KL ATMEGA 328P sa dá vyčítať, že OSC môže byť
Kryštál, napr.: 16MHz. Jeho cena (0,3Eura) je len o rád menšia ako cena procesora ATMEGA 328P (< 3Eura).
Plocha kryštálu nie je zanedbateľná.

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.

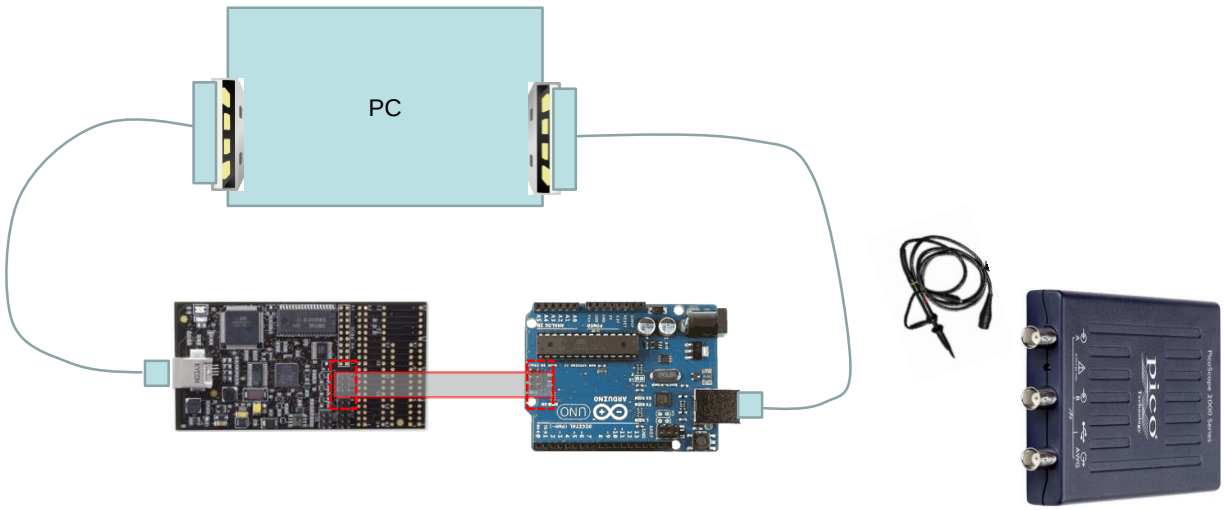
Ako časom zistíme, problém je v tom, že ext. Osc. Je 16MHz a interný RC je 8MHz, a treba ho kalibrovať.
Použijeme.





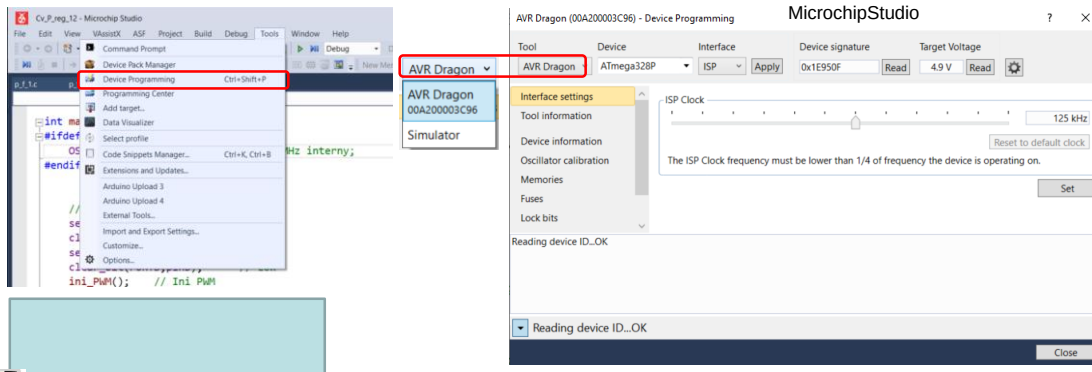
4

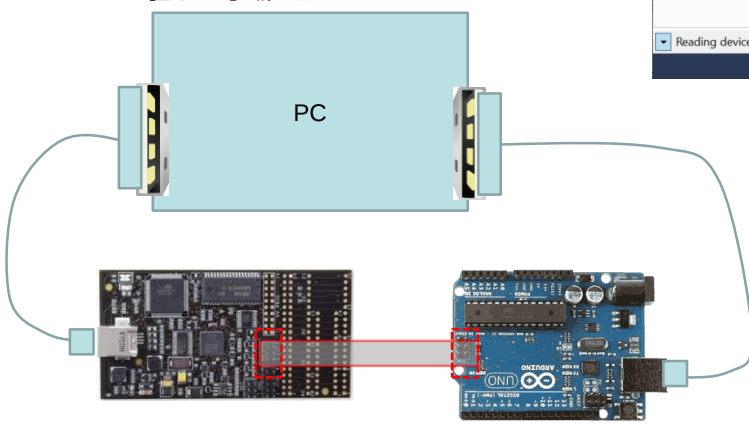
Zapojenie:



5

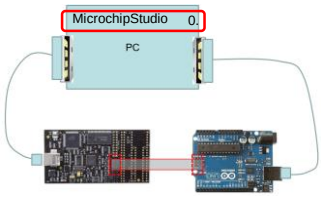
Zapojenie:

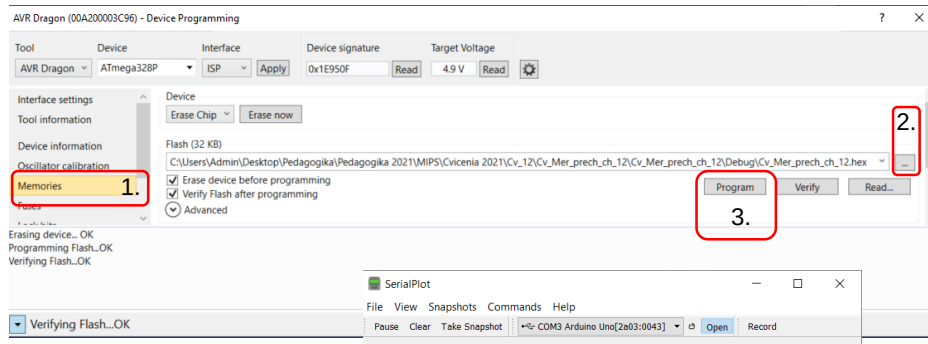


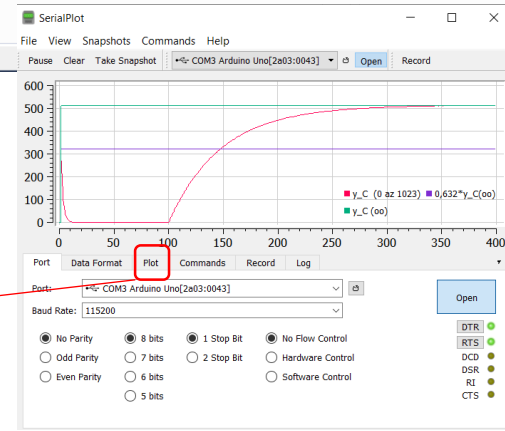


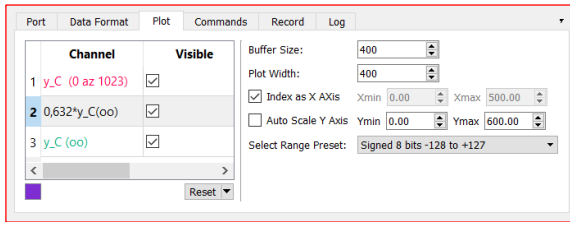
6

Mer_prech_charakt.:



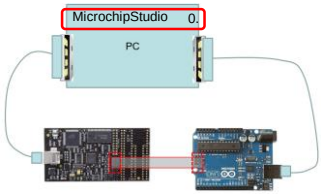


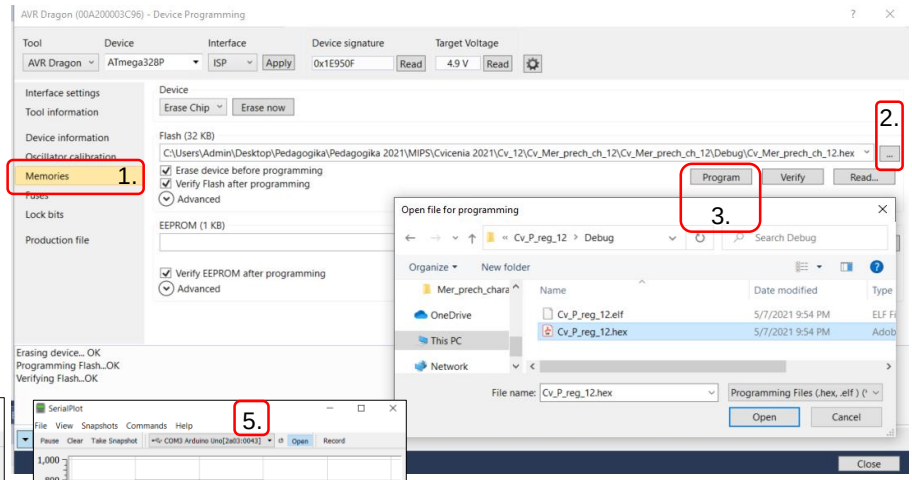


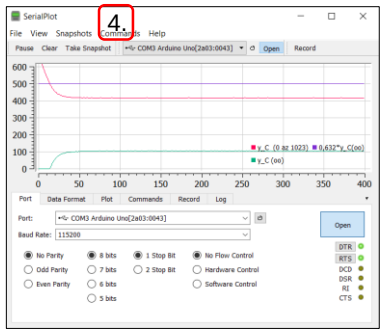


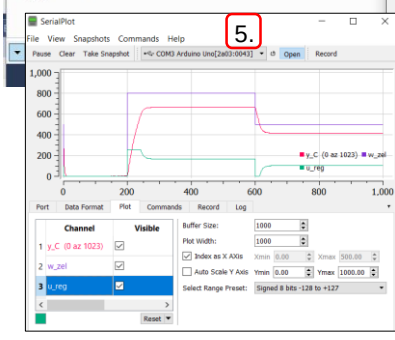
7

P_REGULATOR:



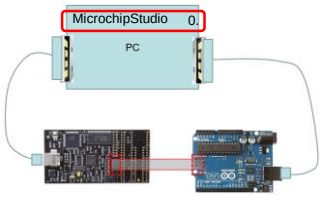


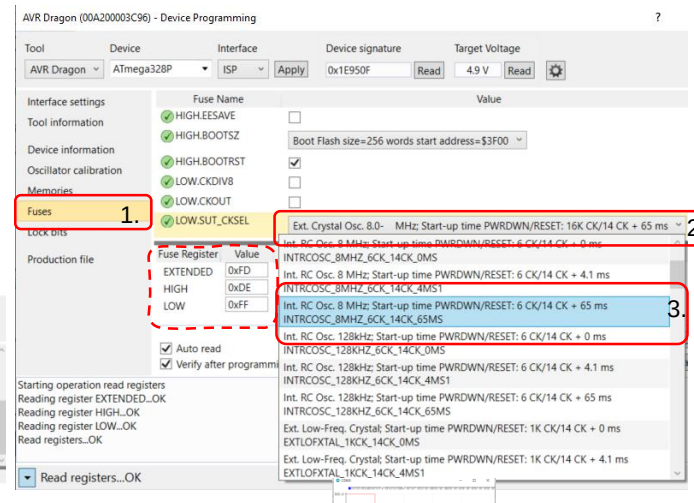


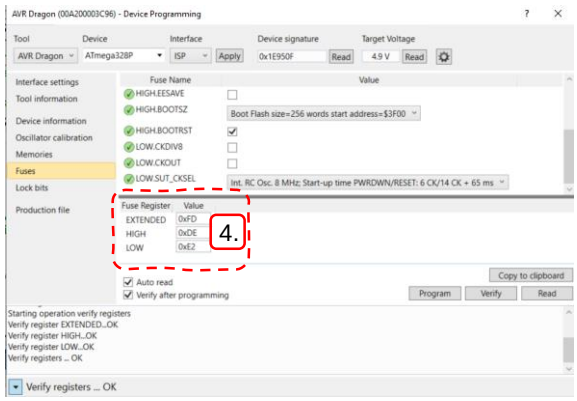


8

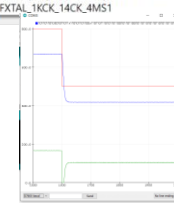
P_REGULATOR:Ext. Kryst. Osc. 16 MHz ⟶ Int. RC. Osc. 8MHz







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



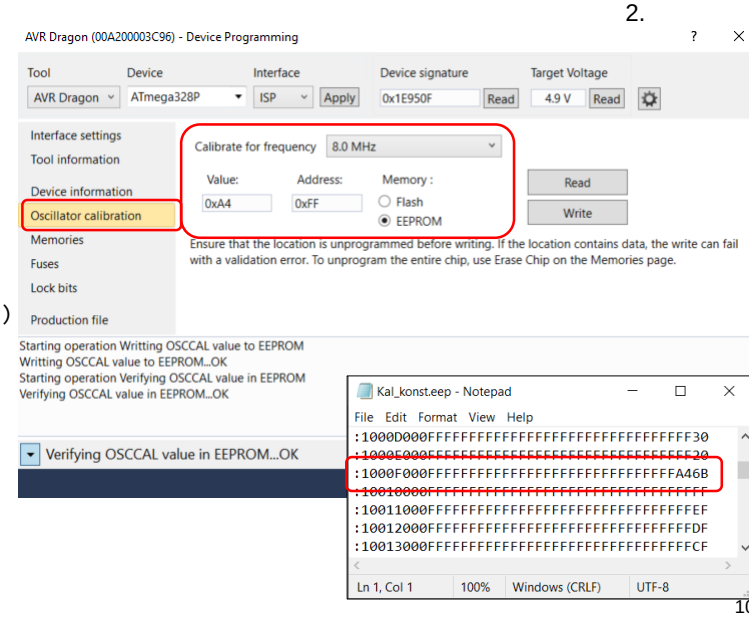
9

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:

1.
```
/* p_f_1.h*/
...
#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;
```

2.



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:

1.
```
/* p_f_1.h*/
...
#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.

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

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

11

Zaujímavosti KL:

ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security

EEARH and EEARL – The EEPROM Address Register

Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	–	–	–	EEAR8	EEARH
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the Atmel ATmega328P and will always read as zero.

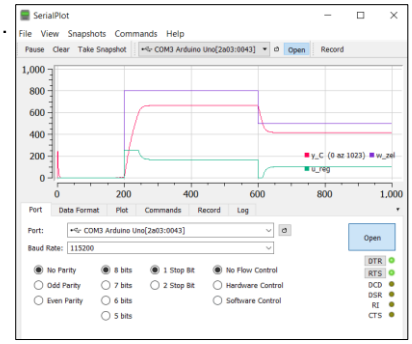
- **Bits 8..0 – EEAR8..0: EEPROM Address**

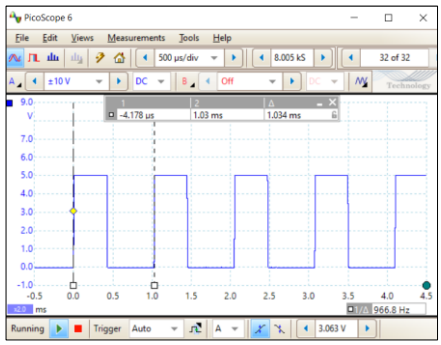
The EEPROM address registers – EEARH and EEARL specify the EEPROM address in the 256/512/512/1K bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255/511/511/1023. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

EEAR8 is an unused bit in ATmega328P and must always be written to zero.

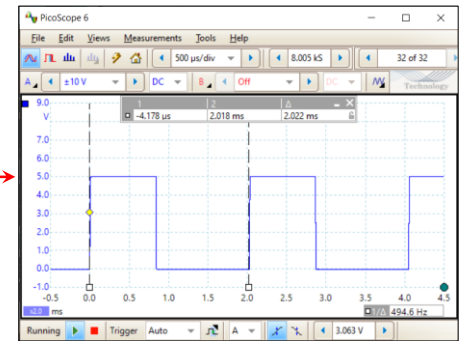
12

Teraz už všetko opäť funguje, ale . .





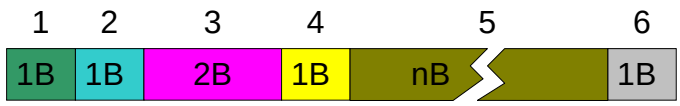
PWM
pôvodne
teraz

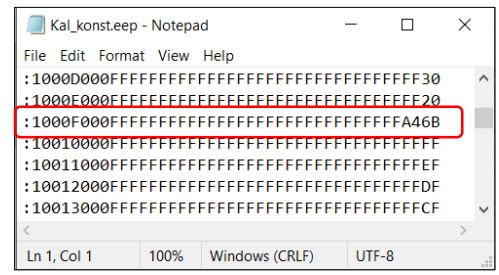


13

Intel HEX formát. [IntelHexFormat.pdf]

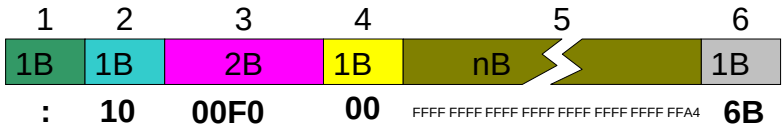
Intel HEX je formát súboru, ktorý obsahuje hexadecimálny kód programu alebo iných dát určených pre mikroprocesory a pamäte. Je to jeden z najstarších formátov používaných v tejto oblasti. V podstate sa jedná o textový súbor kde každý riadok predstavuje postupnosť hexadecimálnych hodnôt . Existujú 3 typy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

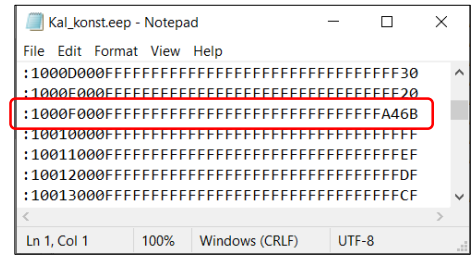




14

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.

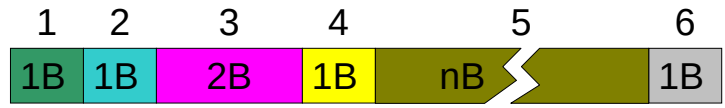
4. Typ záznamu s hodnotami od 00 do 05, dvojica hexadecimálnych znakov.

- 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.

3D - 0x10 + 0x01 + 0x00 + 0x00 + 0x21 + ... + 0x19 + 0x01 = 0x3C0;

0x3C0 modulo 0x100 = 0xC0;

negácia 0xC0 = 0x3F;

0x3F + 1 = 0x40.

16

WDT