PICKIT2 je velice zdařilý programátor od fy Microchip. Kromě programování procesorů může sloužit jako jednoduchý osciloskop, logický analyzátor a vysílač/přijímač UARTu (RS232). PICKIT byl vydán v roce 2008. Dnes je bohužel jeho podpora dávno ukončena, do programu se nepřidávají nové součástky. Protože ale se jedná o naprosto famózní zařízení, upravil jsem pro vás devicefile tak, aby programátor PICKIT2 programoval i náš procesor 33EV32GM002.

PICKIT si vyrobíme podle návodu v <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT2/PICKIT2_konstrukce.doc> nebo v <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT_Juranek/PICKIT2J_konstrukce.docx>

Program PICKIT Utility nainstalujeme podle návodu v <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT_Juranek/PICKIT2J_pouziti.docx>

Budeme instalovat file <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT2/Pickit_Utility/PICkit%202%20v2.61.00%20Setup%20A.zip>

V tomto návodu je platné téměř všechno kromě popisu programování procesoru, ten najdete v tomto file ( to je ten file, který právě teď čtete). S PICKITem uděláme všechny testy, které jsou uvedeny v návodu. Taktéž platí použití jako UART TOOL a osciloskop a Logic Analyzer.

Podle návodu <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT_Juranek/PICKIT2J_pouziti.docx>

hlavně **zkontrolujte, zda vám souhlasí programovací napětí na pinu MCLR.** Pokud bude špatné, zničíte si procesor.

**Odlišnost:** do adresáře, ve kterém máme nainstalovanou Pickit Utility

( dejme tomu C:\Program Files (x86)\Microchip\PICkit\_2\_v2 **) musíme nakopírovat mnou upravený PK2DeviceFile.dat .** Prostě ten jejich smažete a na jeho místo nakopírujete tento: <http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT_pro_33EV32/PK2DeviceFile.dat>

Dále u W10 musíte nainstalovat Net Framework 2.0.5

Pro pořádek instalace ještě jednou

Postupujeme podle http://ozeas.sdb.cz/panska/mikroproc/PICKIT2/PICKIT\_pro\_33EV32/Pickit\_33EV.docx

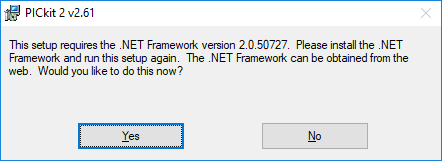
stáhneme soubor

PICkit 2 v2.61.00 Setup A.zip

unzip

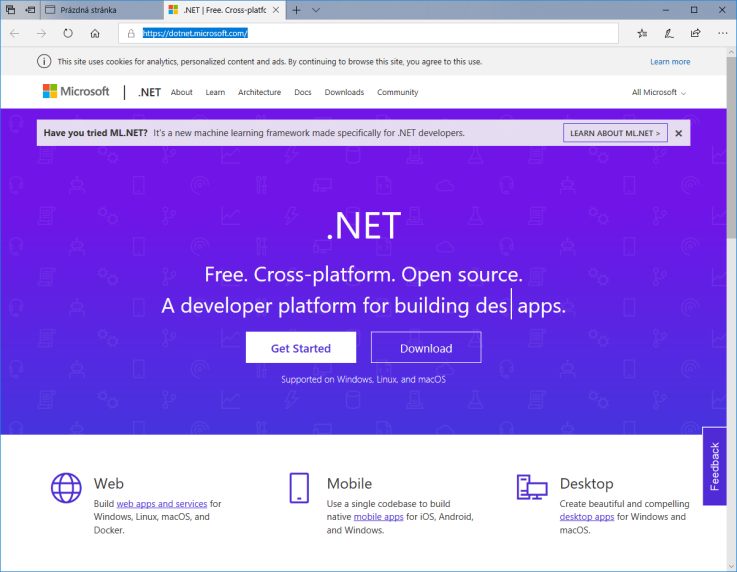
cvakáme na setup.exe

Objeví se



Když cvakneme na YES, zobrazí se nějaká obecná stránka

<https://dotnet.microsoft.com/>



Po cvaknutí na download to nabízí .NET Framework 4.7.2

**Ale to my nechceme**.

My opravdu potřebujeme Net Framework 2.0.5

Stáhneme ho z

<https://www.microsoft.com/en-us/download/details.aspx?id=6523>

Stáhnul se soubor NetFx64.exe , po spuštění si tento soubor ještě něco stáhnul z Internetu, nakonec bylo nutno udělat restart počítače. Po této akci již setup.exe pro PickitUtility funguje.

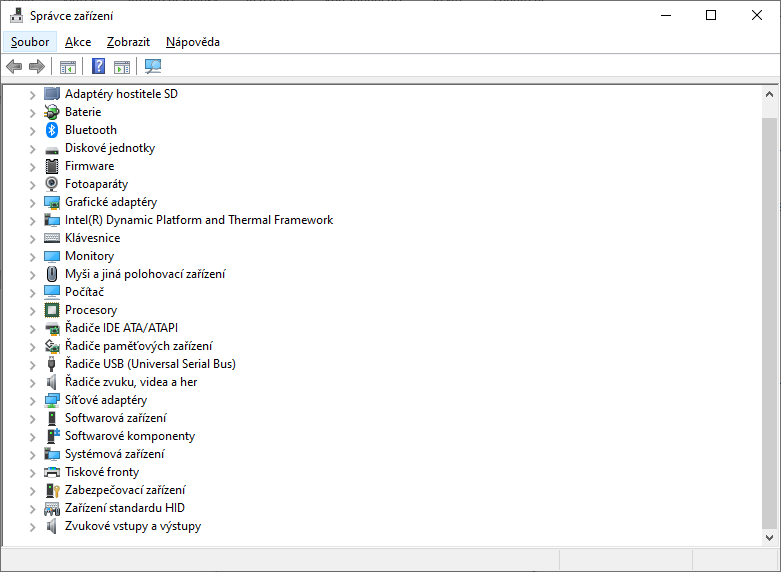
Možná se to dá udělat deseti jinými způsoby, nevím,

**Nezapomeneme přepsat devicefile !**

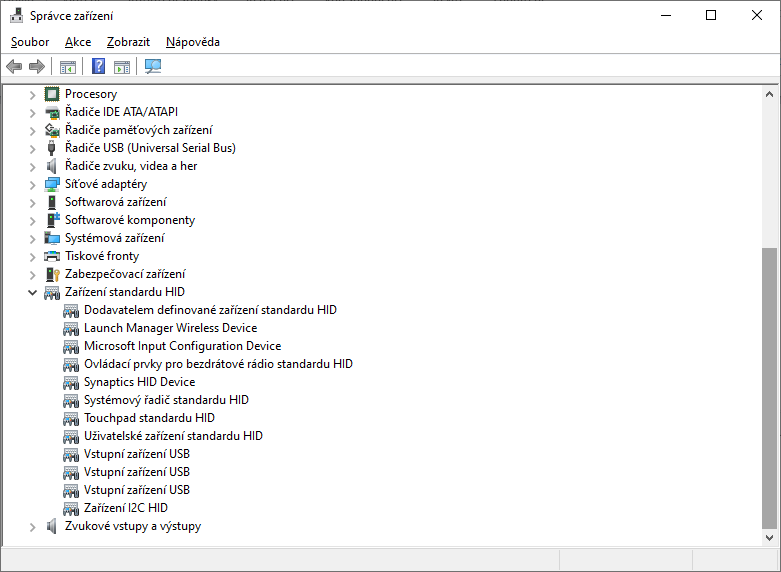
Jak poznáme, že hardware funguje správně.

W10 – nastavení (zubaté kolo) - systém – o systému (potáhnout dolů posuvník) – Správce zařízení

Objeví se nějaké takovéto okno:



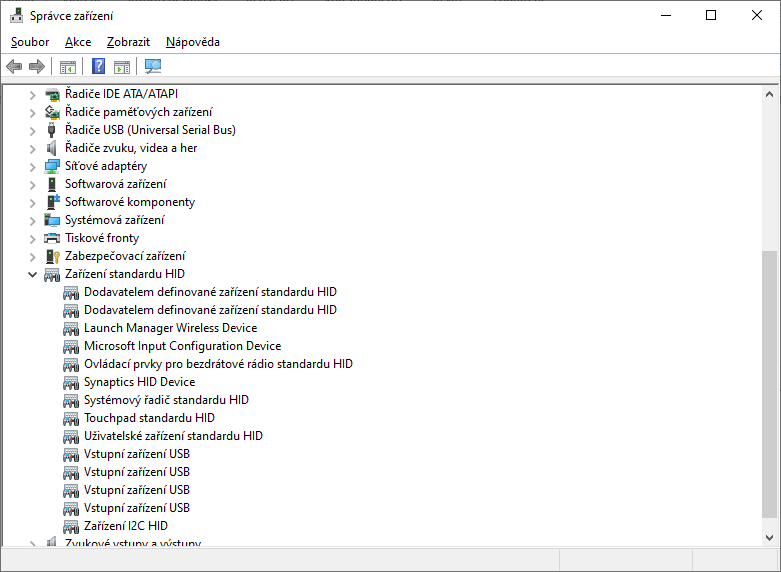
To, co nás zajímá, jsou Zařízení standardu HID . Rozklikneme.



Je jich tam docela dost. Když připojíme PICKIT, jedno zařízení přibude.

V našem případě přibylo jedno „dodavatelem definované….“ . Tím poznáte, že hardware PICKITu je v pořádku.

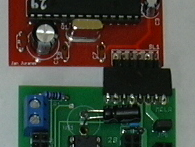
Zkusíme to několikrát – připojíme PICKIT, přibude v okně zařízení. Odpojíme PICKIT, zařízení zmizí.



No a teď již můžeme pustit PICKIT Utility.

**Použití jako programátor**

pokud používáte programátor pro naprogramování bastldesky <http://ozeas.sdb.cz/panska/mikroproc/33EV32/bastldeska/procesor/> , musíte připojit PICKIT k bastldesce tak, aby konektor PICKITu byl uprostřed

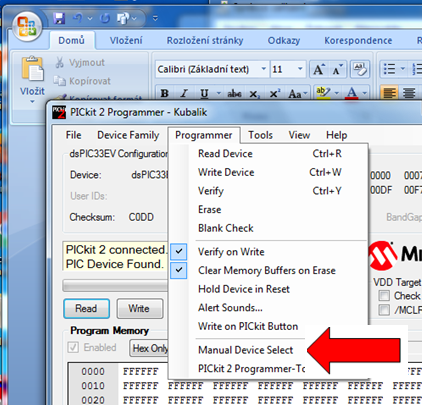




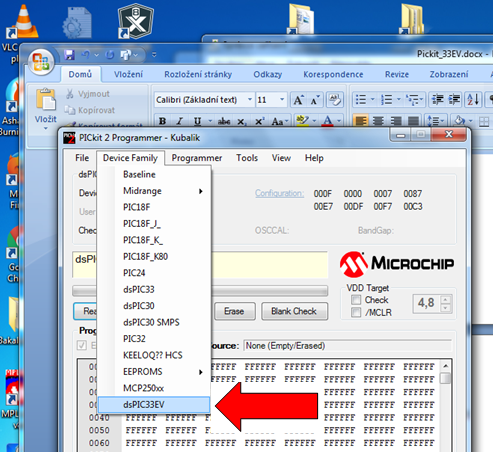
Je to proto, že ve škole máme programovátko PRESTO, které má osm pinů. Vaše bastldeska se tedy může připojit k PICKITu i k PRESTu

Pustíme PICKIT Utility. Start trvá dost dlouho, na ikonku **necvakejte** vícekrát.

V nastavení programátoru **NEBUDE** zaškrtnuto Manulal Device Select

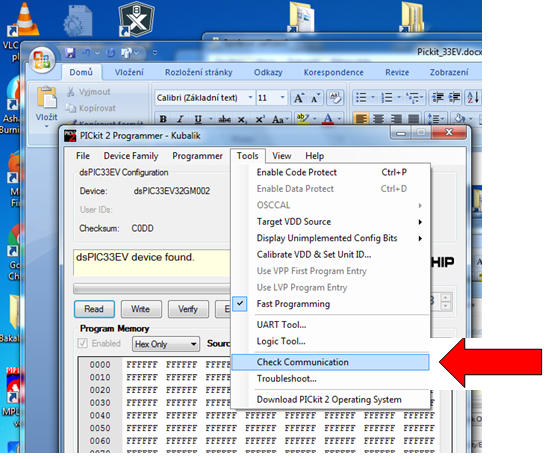


Dále nastavíme rodinu součástky –Device Family - na dsPIC33EV

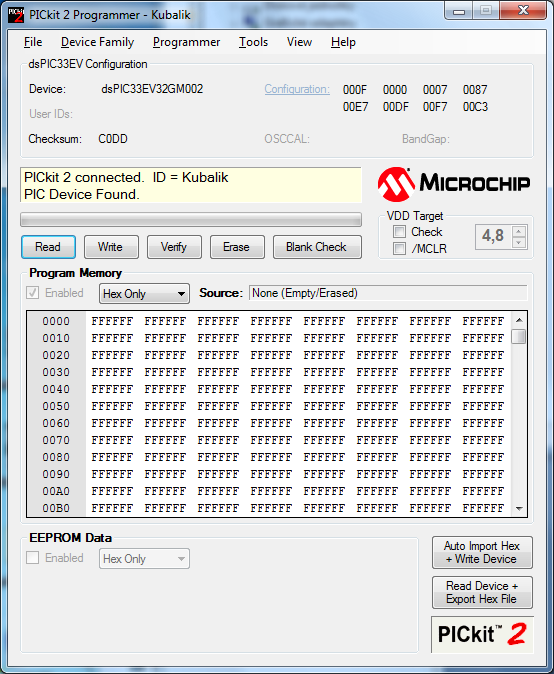


Pokud tady nenajdete dsPIC33EV, znamená to, že jste si nepřepsali devicefile. Tak zpátky na stromy a přepsat.

Pokud PICKIT automaticky nenajde součástku, zkusíme udělat Tools- Check Communication



Objeví se



Nahoře v okně se objeví : Device: dsPIC33EV32GM002

Pokud ne, máte něco špatně

Tady je nutno rozlišit dvě věci: Pickit2 connected – to znamená, že programátor PICKIT je připojen k počítači a pracuje s programem. Device: dsPIC33EV32GM002 znamená, že k programátoru je připojen procesor a programátor procesor našel.

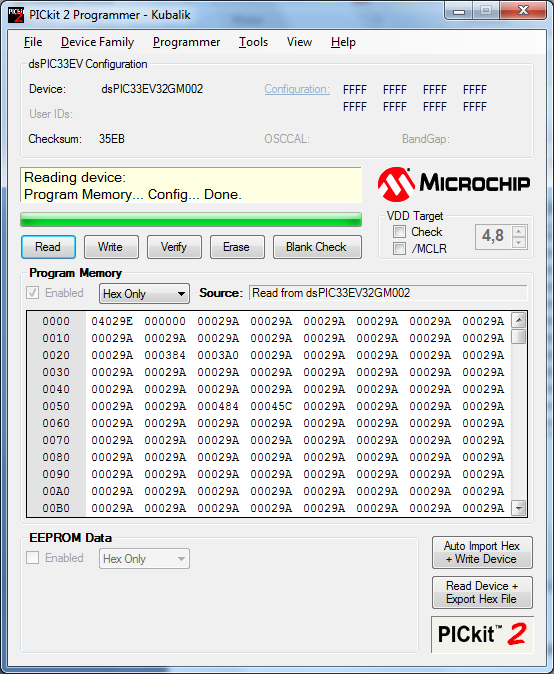
Dále máme k dispozici následující tlačítka:

Read – přečte paměť procesoru

Erase – smaže celý procesor

Tohle je snad jasné, prostě to zkuste. V okně uprostřed vidíte obsah programové paměti EEPROM procesoru. Můžete tam ručně zapsat nějaké kódy a procesor naprogramovat. **Před programováním musí být procesor smazán – tlačítko Erase.**

Po přečtení paměti vypadá okno např. takto:



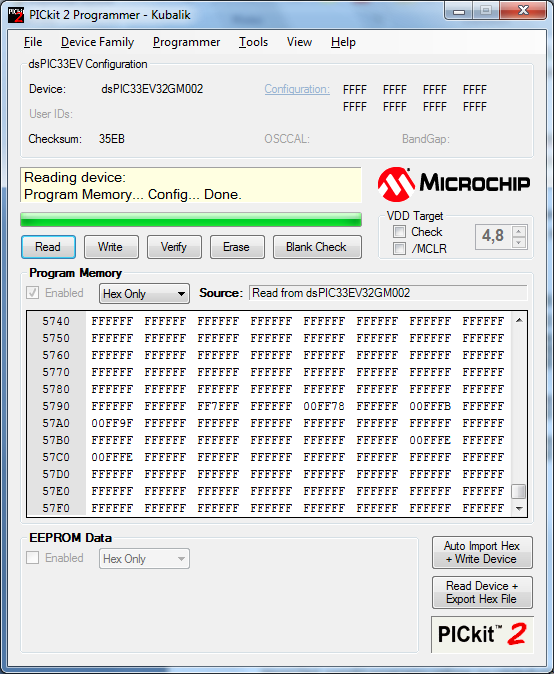
Na levé straně vidíme adresy buněk, v okně je jejich obsah. V řádku máme celkem 16 buněk, proto adresa druhého řádku je o 16 větší, tedy 0x0010 . V okně vidíme pohromadě vždy tři buňky. Ta nejvyšší je tzv. fantom, její adresa se nepočítá ( podrobněji viz strana 35 datasheetu, kapitola 4.1.1 ).

PICKIT je vývojově stár asi 15 let. Jeho vývoj již Microchip dávno ukončil. Abychom mohli programovat i náš procesor 33EV32GM002, musel jsem pro vás program upravit. Některé věci ovšem úplně upravit nešlo. Nejdůležitější z nich jsou konfigurační slova – config words.

Config words u starších procesorů jsou oddělena v úplně jiné části paměti. U našeho procesoru následují hned za pamětí programu ( viz datasheet, str. 11, FIGURE 4-1: PROGRAM MEMORY MAP FOR dsPIC33EV32GM00X/10X DEVICES)

**V mnou upraveném programovátku programujeme i čteme najednou UserProgramFlashMemory i DeviceConfiguration. Má to za následek některá zavádějící hlášení programu, ale nic jiného jsem s tím nebyl schopen udělat**.

Horní část paměti programu vidíme na následujícím obrázku.



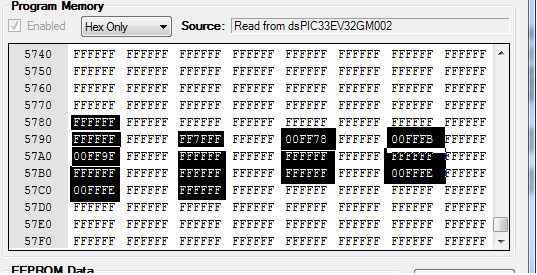
Paměť programu končí na adrese 0x578F . Od adresy 0x5780 je prostor pro konfigurační slova. Ta ovšem nejsou všude, ale jenom na některých adresách

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |



I v okně PICKITu vidíme na některých z těchto adres čísla odlišná od 0xffffff .

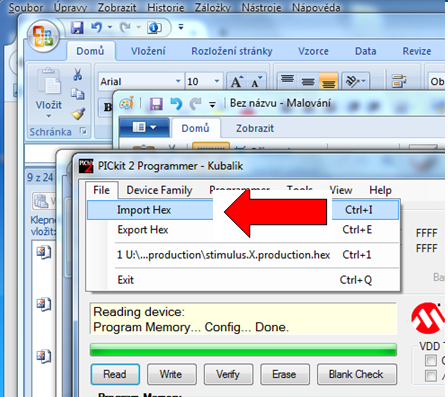
Na následujícím obrázku máte konfigurační slova zvýrazněna.



**Vlastní programování procesoru**

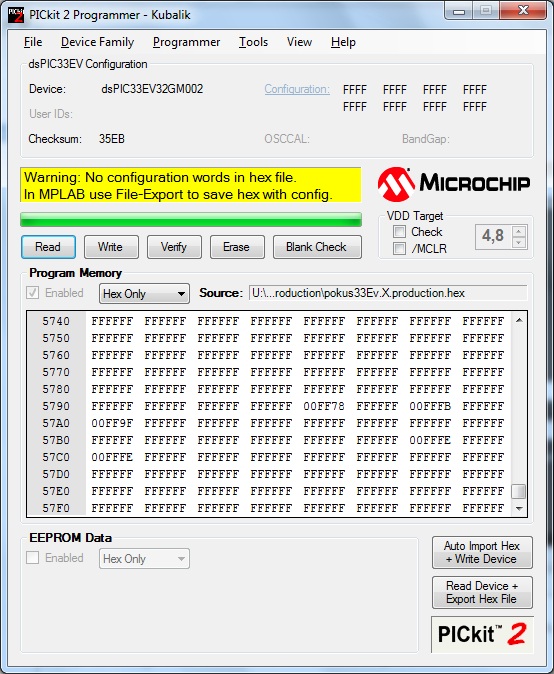
Nejdříve musíme program přeložit – např. pomocí programu MPLABX. Tím získáme přeložený \*.hex file s programovým kódem. Tento kód nyní importujeme do PICKITu

File – Import Hex



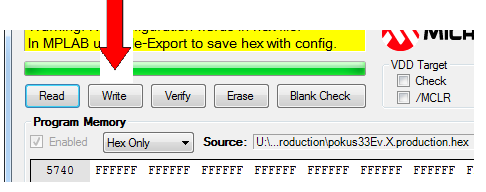
Najdeme složku, ve které máme file s přeloženým programem ( \*.hex ), a zadáme jeho adresu s celou cestou.

Po importu file se objeví hláška:

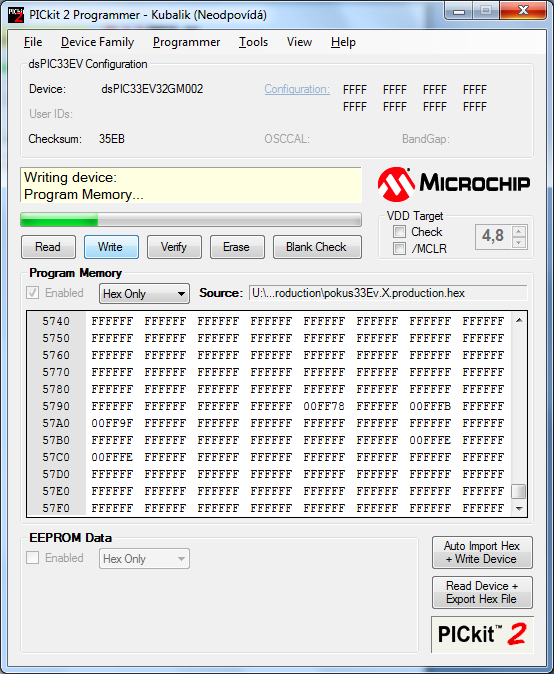


Neděste se. Je to způsobeno právě tím, že program i konfigurační slova načítáme najednou v jednom bloku. To, zda máme nebo nemáme konfigurační slova zjistíme tak, že se podíváme na adresy 0x5780 a následující, zda máme alespoň v některých buňkách jiné slovo než 0xfffff. Pokud ano, je vše v pořádku.

No a zapisujeme data do procesoru – Write

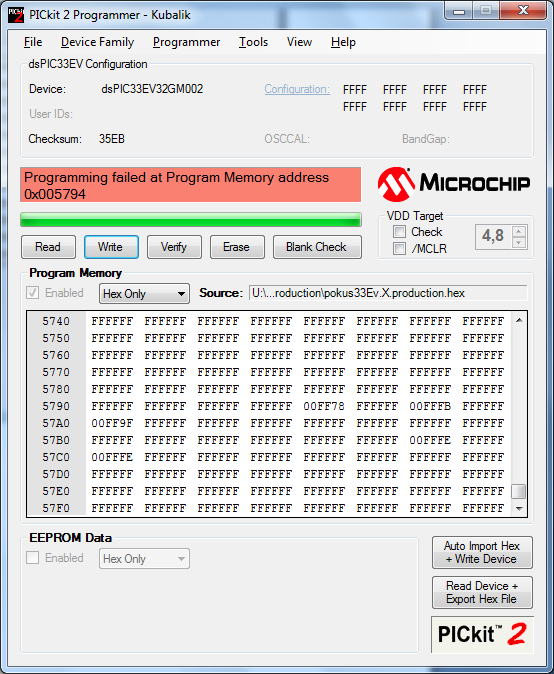


Sledujeme postupující zápis



Zápis se většinou tak na 1 minutu zastaví (tedy postupující zelený pruh se zastaví), nechte tomu čas.

A nakonec se objeví:



Tato hláška znamená, že

**procesor je naprogramován správně.**

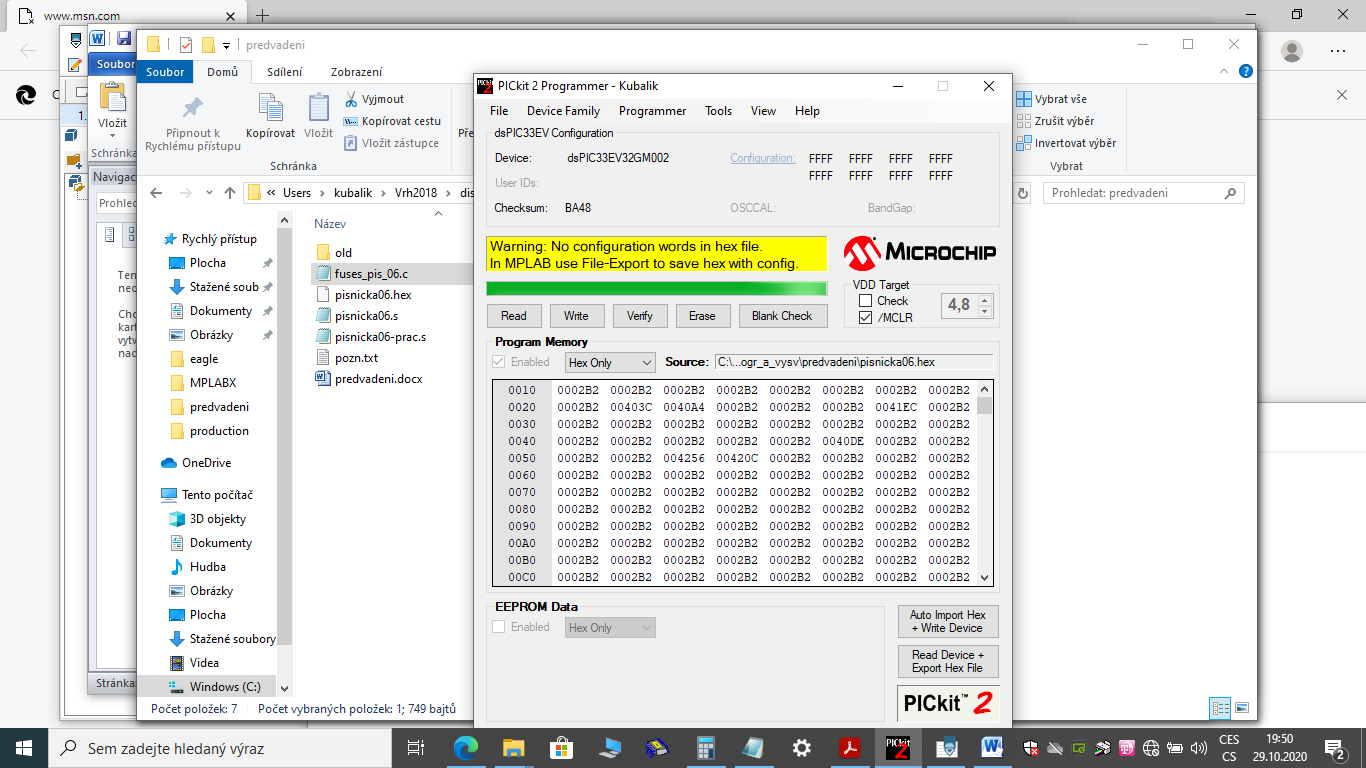
Problém by nastal, kdyby chybná adresa byla menší než 0x5780 . Potom by skutečně k chybě došlo. Chybová hláška je způsobena tím, že v oblasti konfiguračních slov se programují pouze 2 byte, nikoli tři. Třetí byte konfiguračního slova je čten jako 00, a pak je pochopitelně odlišný od hodnoty 0xff , kterou jsme se pokoušeli programovat. Přečtěte si paměť (Read) a podívejte se, jaká hodnota se přečetla.

**Takže ještě jednou – jak programovat.**

V okně „program memory“ v dřívějších obrázcích máte vždy adresy ke konci paměti. V následujících obrázcích zobrazíme adresy od začátku.

Fille – import hex – zadáme cestu k \*.hex file

V okně se objeví něco jako:



Zde si hlavně všimneme, že na adresách počínaje 0000 **NEJSOU** čísla 0xffff

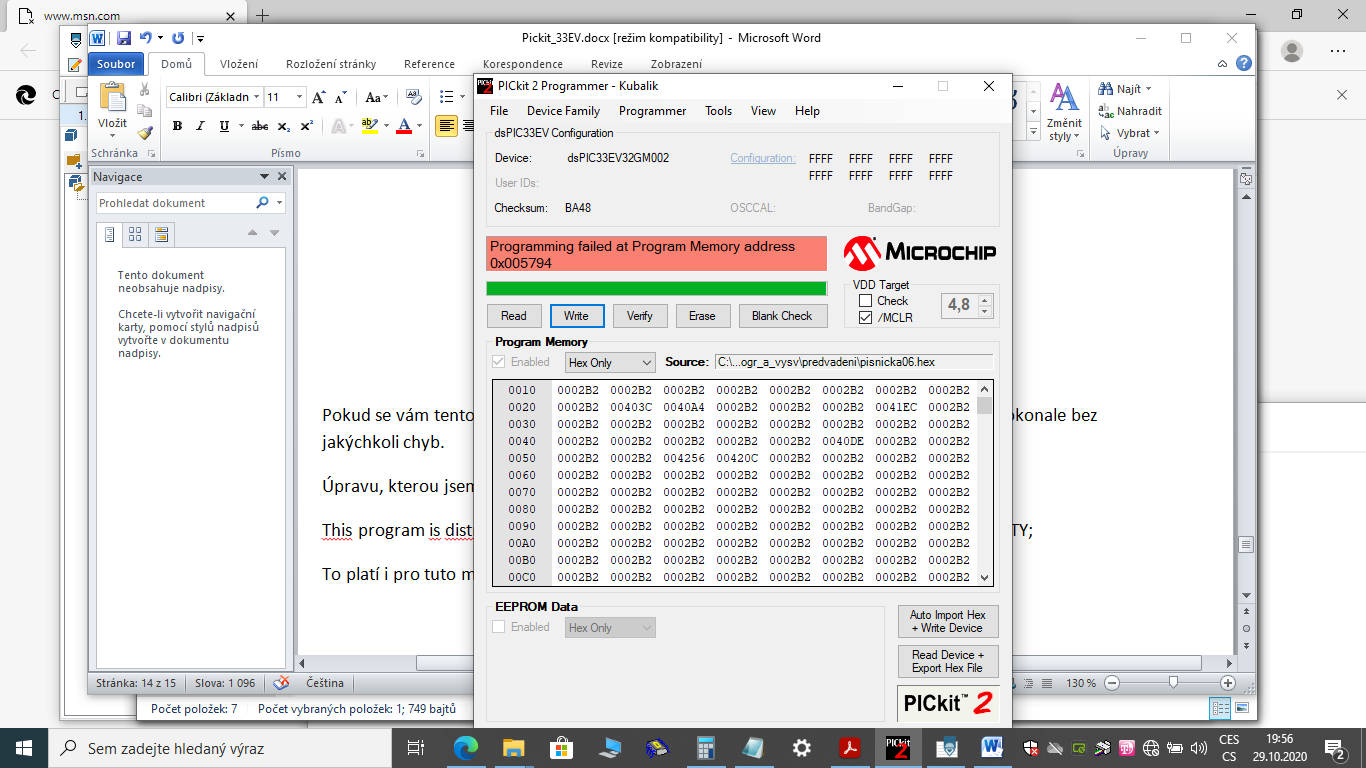
Pokud máte v okně samá čísla 0xffff , neimportovali jste správný \*.hex file

Běžná chyba je následující: smažeme procesor – ERASE – a programujeme. Ale po ERASE se v okně objeví 0xfff , protože PICKIT přečetl smazaný procesor.

Musíme tedy NEJDŘÍVE smazat, potom import hex , potom WRITE

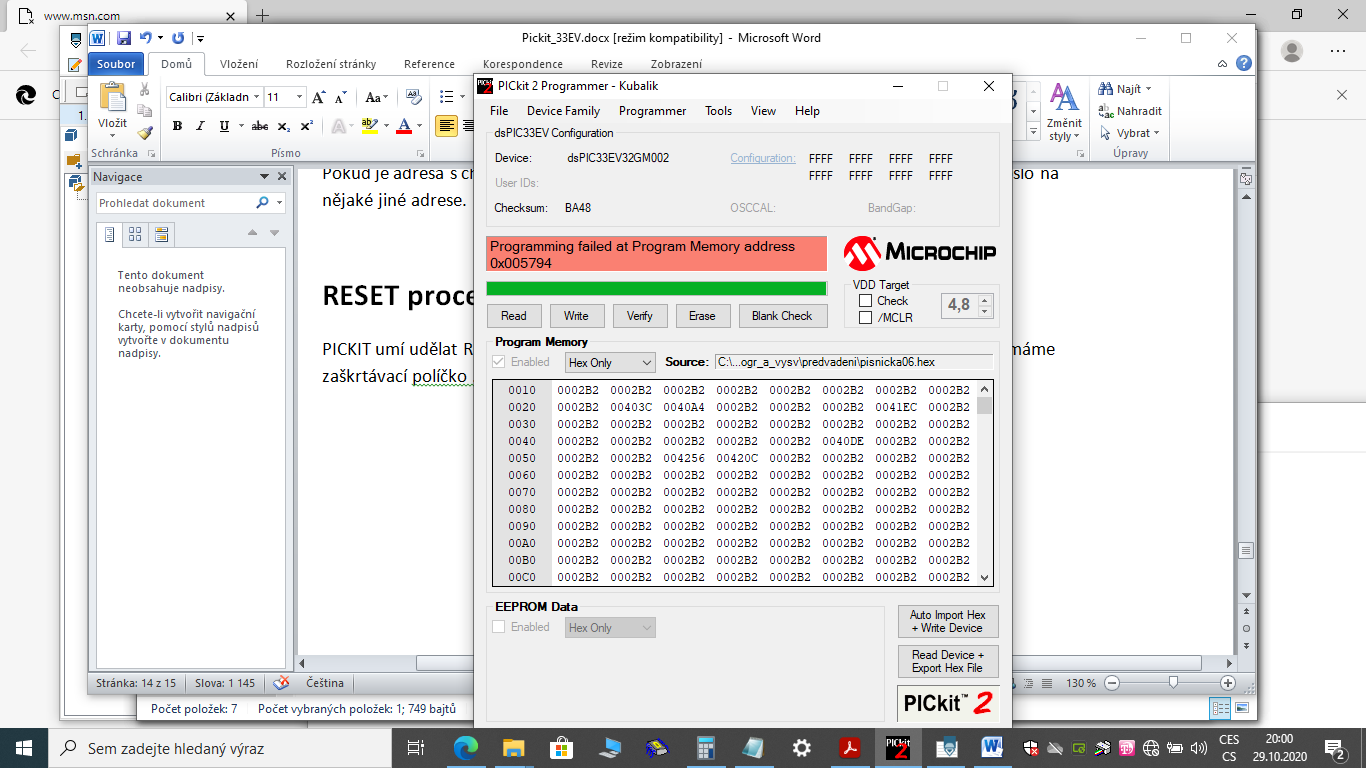
Příkaz WRITE sám procesor také smaže předtím, než začne programovat.

Po naprogramování se objeví



Pokud je adresa s chybou 0x005794 , je to v pořádku. Problém je v případě, že k chybě došlo na nějaké jiné adrese.

**RESET procesoru**

PICKIT umí udělat RESET procesoru. To se děje pomocí pinu 1 – MCLR. U PICKITu k tomu máme zaškrtávací políčko /MCLR. Na obrázku políčko není zaškrtnuto, to znamená, že procesor pracuje normálně. Pokud toto políčka zaškrtnete, děláte neustále RESET procesoru, a procesor pochopitelně nepracuje. 

Pokud se vám tento PICKIT nelíbí, kupte si PICKIT3 nebo PICKIT4. Ten pracuje naprosto dokonale bez jakýchkoli chyb.

Úpravu, kterou jsem udělal, jsem udělal pod licencí GNU. K ní bývá připojen text:

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;

To platí i pro tuto moji úpravu.

Pokud si chcete hrát s devicefile, potřebujete

Devicefile editor

<https://sites.google.com/site/pk2devicefileeditor/>

devicefile, které jsem použil k úpravám, je na

<http://www.auelectronics.com/forum/index.php/topic,413.msg1167.html#msg1167>

já jsem dodělal procesory PIC16F1708 PIC16F1704 dsPIC33EV32GM002