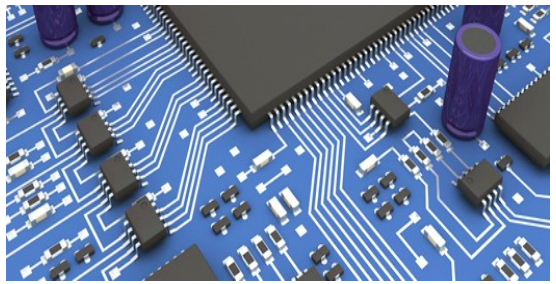
Aplikace Embedded systémů v Mechatronice



MPLAB® X IDE

Michal Bastl

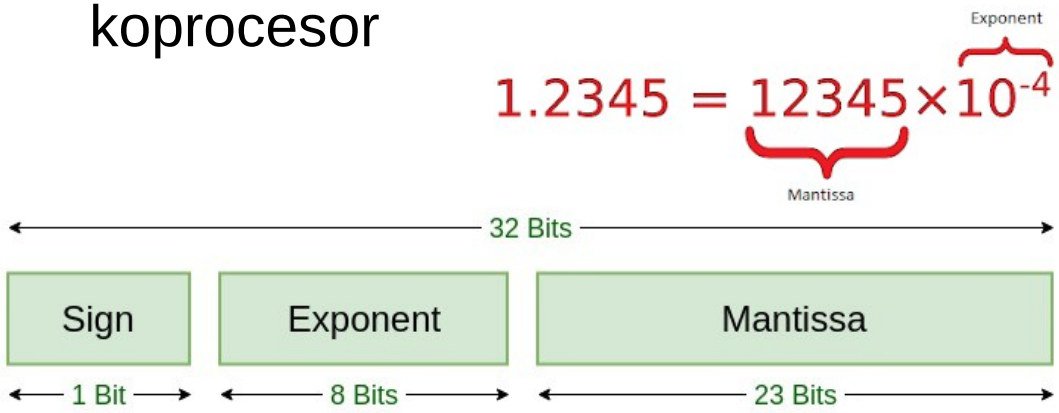
Reprezentace čísel

- Celočíselná reprezentace
- S pevnou řádovou čárou (fixed point)

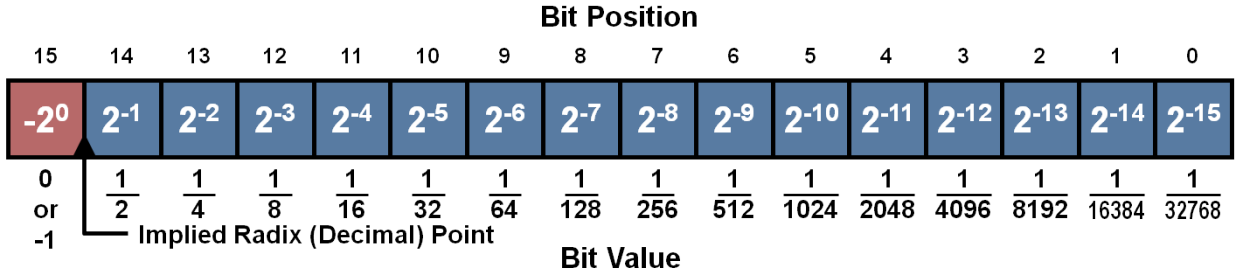
- S plovoucí řádovou čárkou (floating point)
  - Jsou popsána ve standardu IEEE 754
  - Pro práci s těmito čísly slouží FPU(floating point unit), také matematický koprocesor

čísla s pevnou řádovou čárkou jsou často používaná na DSP (digital signal processor) procesorech. Jsou vhodná pro rychlé výpočty jako FFT, FIR filtry apod.

Na obrázku dole je zobrazen tzv formát Q15 první bit reprezentuje znaménko. Následují zlomky za desetinou čárkou.



Single Precision
IEEE 754 Floating-Point Standard



0x8000 = -1.000000000000
0x7FFF = 0.999969482422

Reprezentace čísel

8-bit procesor PIC18 nemá ani FPU ani DSP jednotku

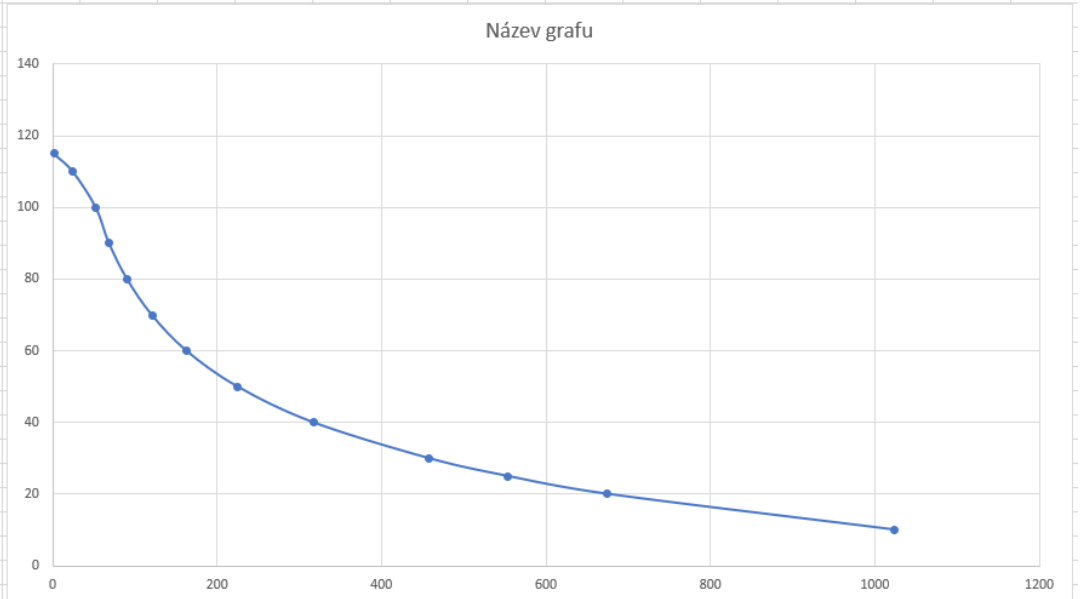
- Pokud použijete float, dojde k jeho výpočtu pomocí SW, což je velké množství instrukcí, ale někdy nám to nemusí vadit.
- Jinak můžeme využít různých triků, jak hodnotu uchovávat.
- Například pi=3,14159 mohu uložit bez čárky 31415, ale musím si pamatovat, kde je.
- Hodnota pí může být také 22/7 = 3.142857; nebo přesněji 3217*10000/1024 = 31416 což umožňuje vydělit pomocí bitového posuvu >> 10.

- **Převod z AD převodníku:**
- Referenční napětí je 3.3V, ale to je také 3300mV
- Pak mohu klidně použít long adc = ADC * 3300;
- Výsledek v mV pak získám jako ADC>>10

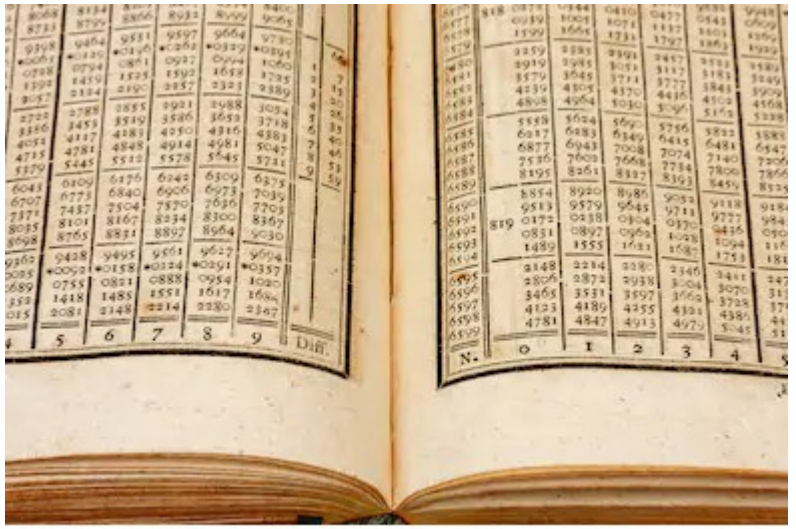
LookUP

Jedná se o hojně využívanou metodu v embedded systémech. V našem případě ji budeme využívat k rychlému určení nelineární funkce. Hodí se však i k přepočtům závislostí, pro kterou funkci neznáme, ale máme hodnoty z naměřených (experimentálních) dat. Závislosti odporu na teplotě čidla, odbuzovací charakteristika elektrického stroje a tak podobně... v jednoduché podobě se jedná o tabulku s hodnotami X a Y. Tabulka má nějakou konečnou hodnotu, nemůže být tedy libovolně přesná. Mezi body, které znám je třeba provádět interpolaci. Nejpoužívanější je lineární.

1023	10
674	20
553	25
457	30
317	40
225	50
163	60
121	70
91	80
69	90
53	100
25	110
2	115



Charakteristika termistoru



Logaritmické tabulky

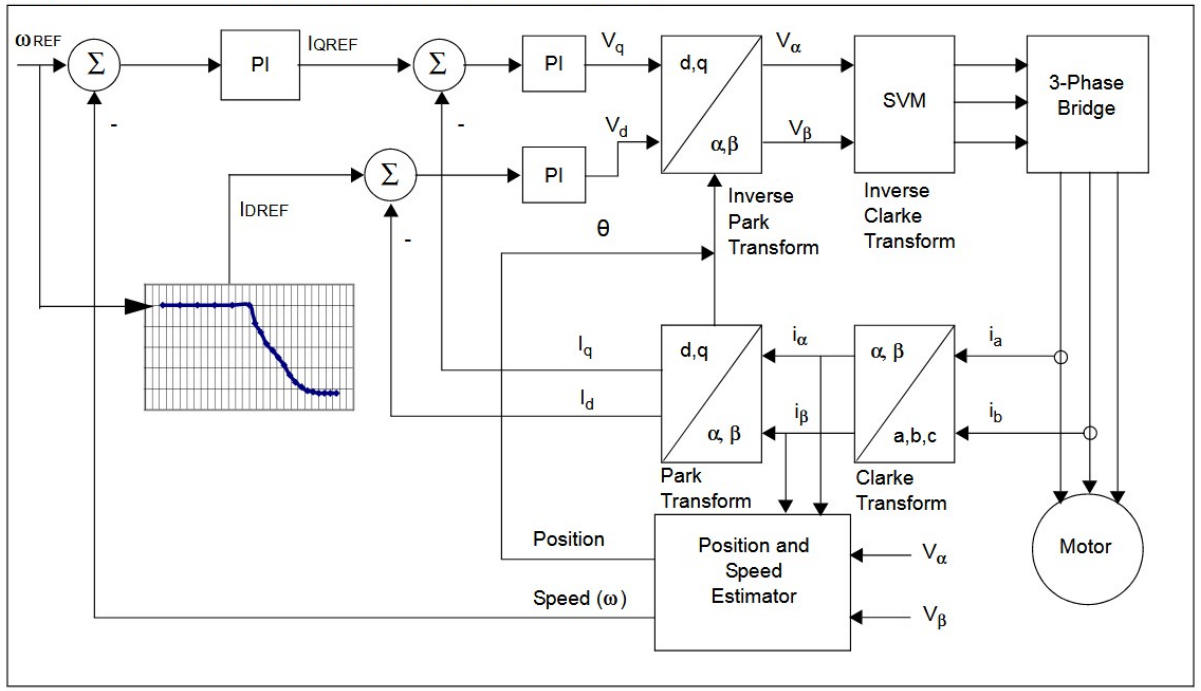
LookUP

Vhodné je pro definici tabulky použít strukturu, která představuje hodnoty x,y.

Tabulku pak tvoří pole těchto bodů.

Na obrázku LookUP tabulka v profesionálním FOC řízení střídavého motoru:

FIGURE 6: VECTOR CONTROL BLOCK DIAGRAM




```
typedef struct
{
    int x;
    int y;
} point
```

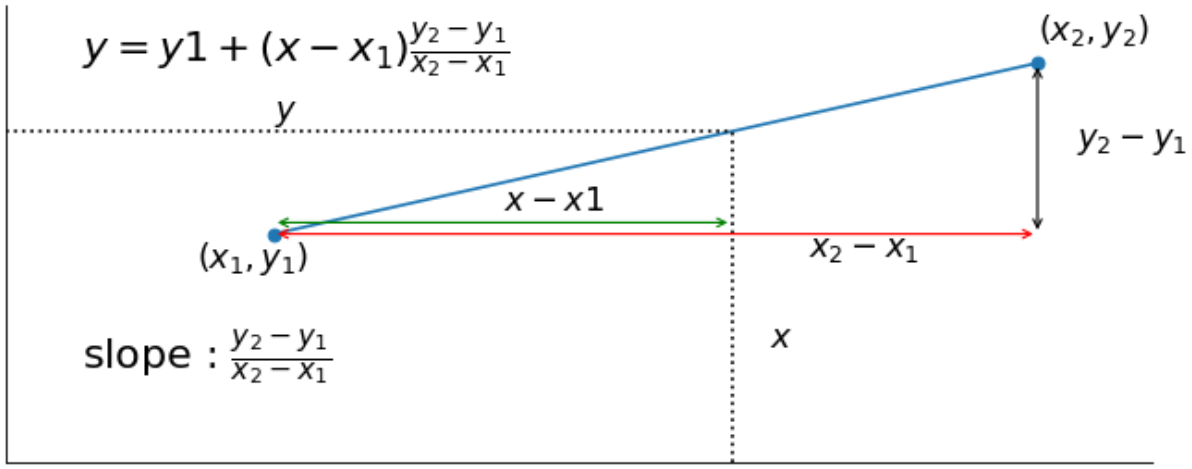
```
const point tabulka[18] =
{
        {0, 0},
        {15, 46},
        {30, 86},
        {45, 114},
        {60, 126},
        {75, 122},
        {90, 101},
        {105, 67},
        {120, 23},
        {135, -23},
        {150, -67},
        {165, -101},
        {180, -122},
        {195, -126},
        {210, -114},
        {225, -86},
        {240, -46},
        {255, 0},
};
```

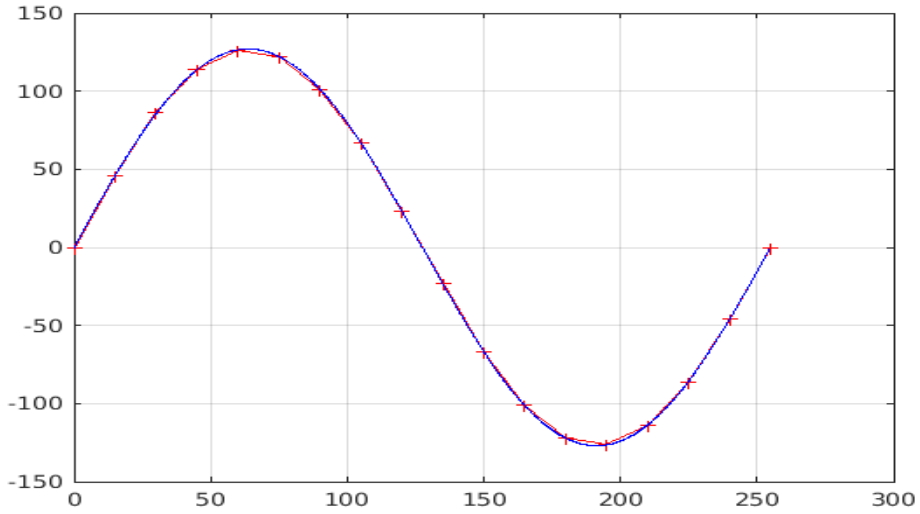
LookUP

V programu poté používám tabulku tak, že napřed naleznu dva sousední body, kde je třeba interpolovat.
To lze udělat různě, jednoduše třeba prochlazením hodnot x.
Lineární interpolace je potom snadná je však dobré přemýšlet na rozsahu proměnných.
Jak tedy dosáhnout lepší přesnosti výpočtu?

```
int y;
long tmp;
tmp = (x - x1);
tmp *= (y2 - y1);
tmp /= (x2 - x1);
y = y1 + tmp;
```

Používat floating point není dobrý nápad, potom je mi ta tabulka k ničemu, nic jsem nezrychlil.

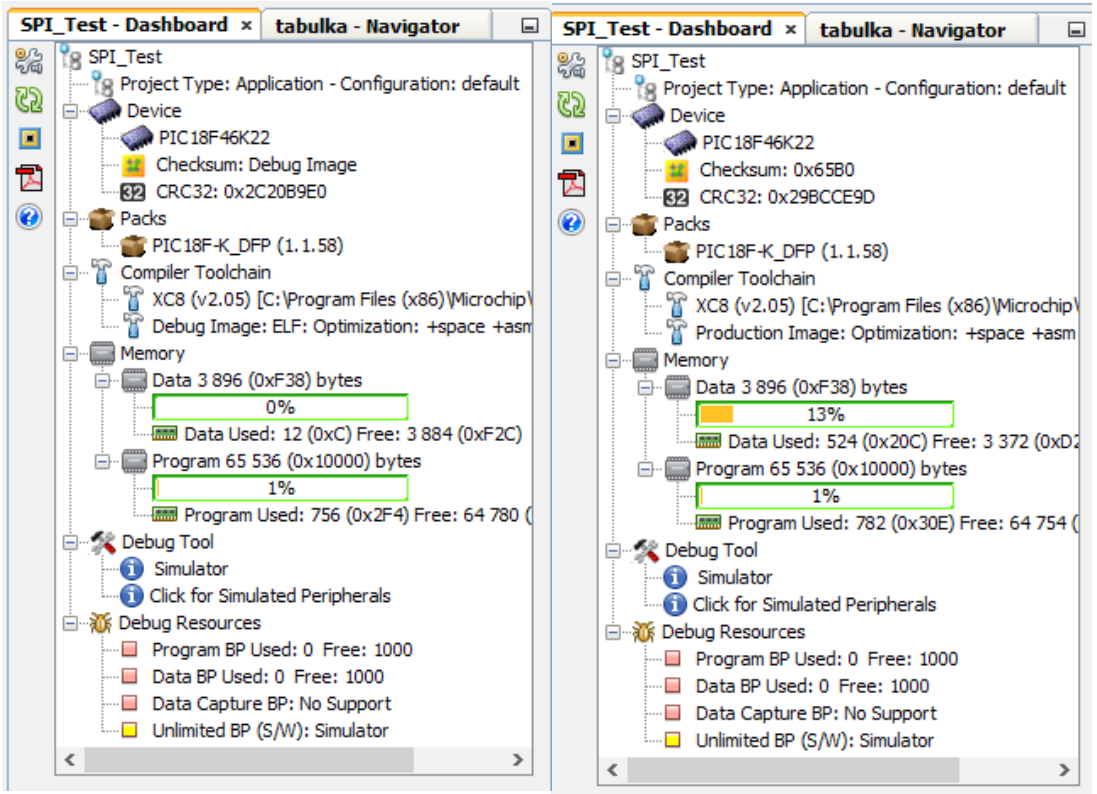




LookUP

- Mezi použitím slovíčka const a mezi jeho vynecháním je zásadní rozdíl.
- Kompilátor je nucen tuto tabulku brát jako proměnnou.
- Nemůže tedy předpokládat, že hodnoty se po celou dobu programu nemůžou měnit.

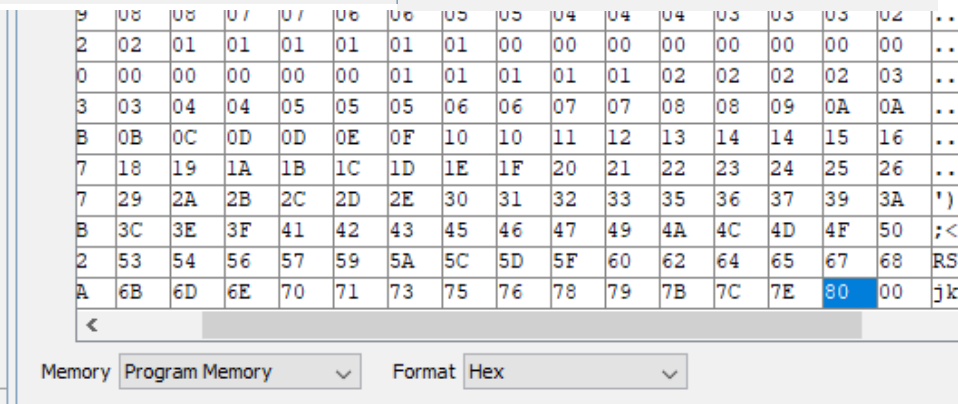
- Nemá tedy možnost jí optimalizovat.
- Vpravo je vidět rozdíl v zabrané RAM.
- Při použití const byla tabulka uložena do paměti programu.
- Paměť programu je typicky větší a je to tedy přesně co chceme.




```
0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x20,0x21,0x22,0x23,0x24,0x25,
0x26,0x27,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x30,0x31,0x32,0x33,0x35,0x36,0x37,0x39,
0x3a,0x3b,0x3c,0x3e,0x3f,0x41,0x42,0x43,0x45,0x46,0x47,0x49,0x4a,0x4c,0x4d,0x4f,
0x50,0x52,0x53,0x54,0x56,0x57,0x59,0x5a,0x5c,0x5d,0x5f,0x60,0x62,0x64,0x65,0x67,
0x68,0x6a,0x6b,0x6d,0x6e,0x70,0x71,0x73,0x75,0x76,0x78,0x79,0x7b,0x7c,0x7e,0x80,
};
```

Tabulka je v paměti programu: program memory

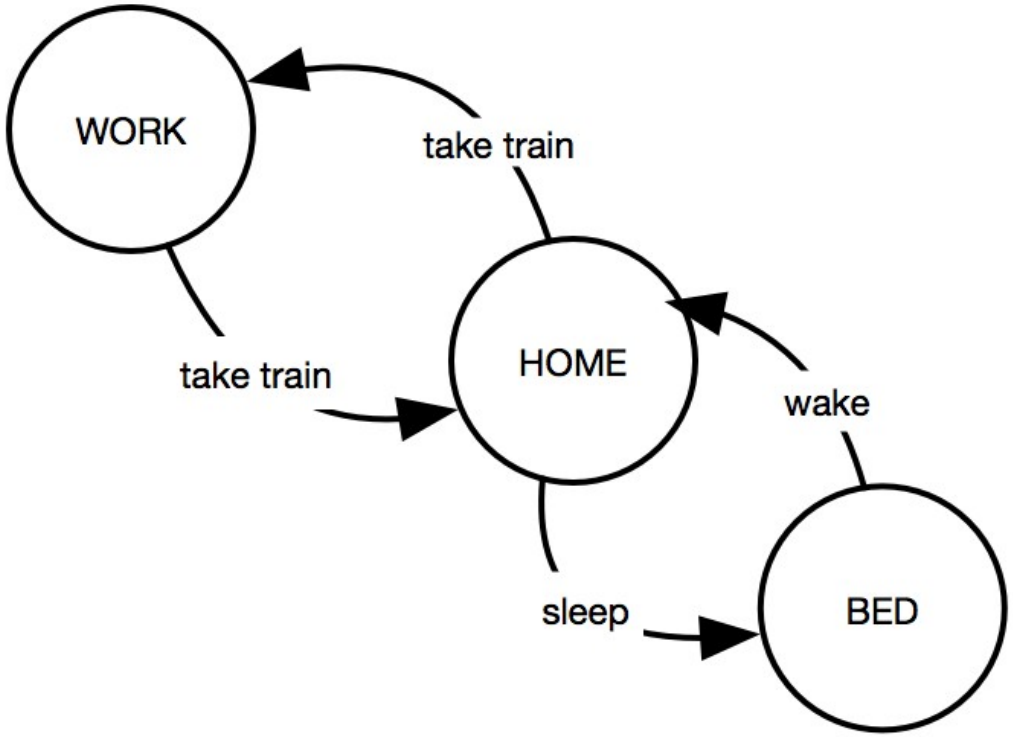
```
init(void){
```



Stavové automaty

Volný výklad konečného stavového automatu FSM:

- Stavový automat je abstraktní popis problému, který jej uzavírá do tzv. Stavů. Lze je samozřejmě popsat matematicky.
- Podstatné však je, že funkcionalitu zařízení uzavírám do stavů a k přechodům z jednoho stavu do druhého používám události.
- Stavový automat má vždy počáteční stav, ve kterém se nachází po uvedení do provozu.
- Kolem automatů existuje rozsáhlá teorie, důležitá je však základní myšlenka.
- Není přesně určeno jak se potom daný automat naprogramuje.

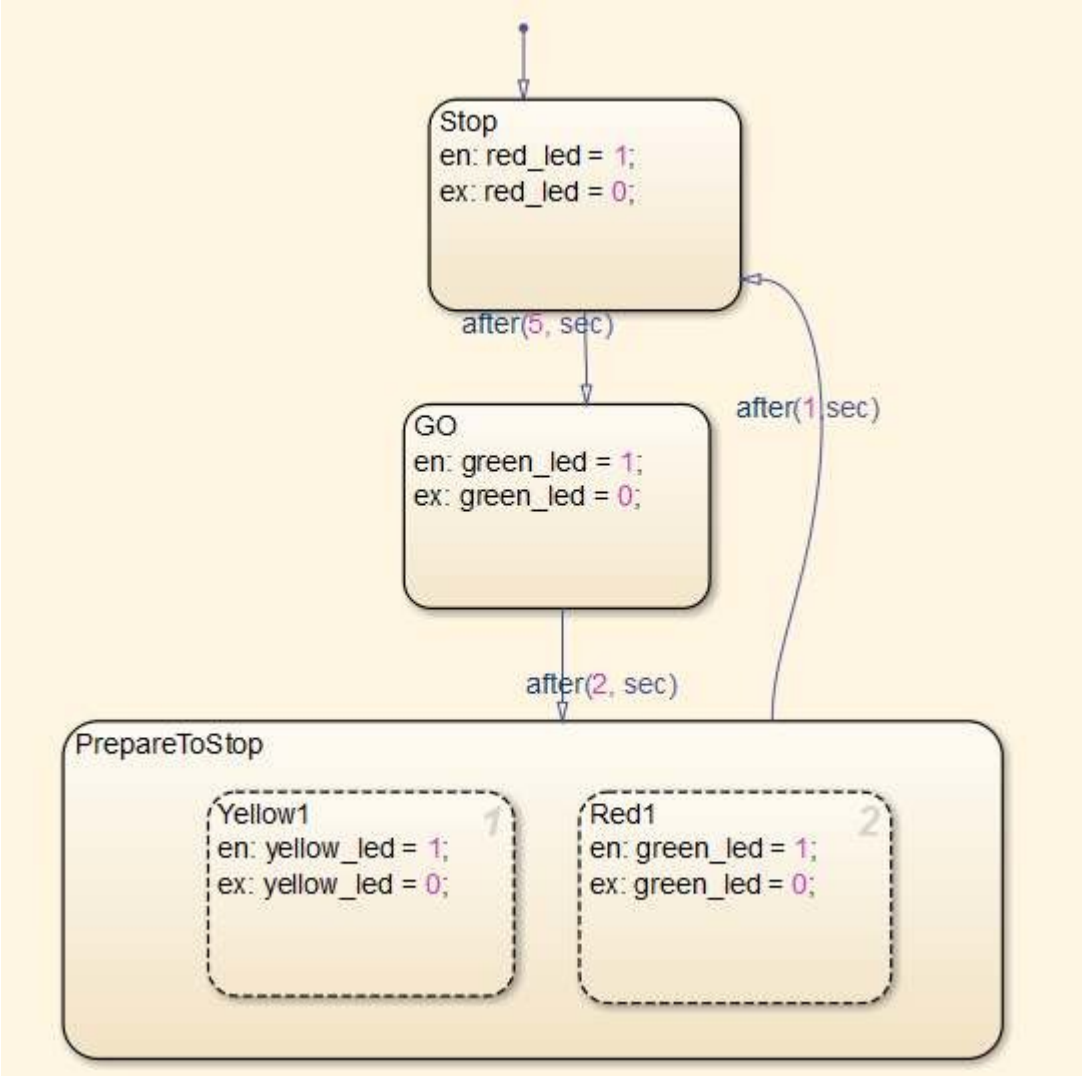


Stavové automaty

- Popis pomocí stavového automatu je tedy převod komplexních systému, do abstraktního popisu pomocí stavů.
- Výhodou je právě to, že tato abstrakce umožňuje snadné ověření funkce, zda takto vytvořený automat skutečně myslí na vše (proofability)
- řešení pomocí stavového automatu je tedy jednoduší ověřit tzv verifikovat

- Možností jak FSM programovat je několik, jednoduché automaty lze dělat pomocí switch-case, což je poněkud nepřehledné, ale často se používá. Dále pak lze použít různý koncept tabulek a struktur a pointery.
- Případně objektově orientovaný přístup OOP

Stateflow:



Semestrální projekt

Projekt naleznete i s jeho popisem na githubu:

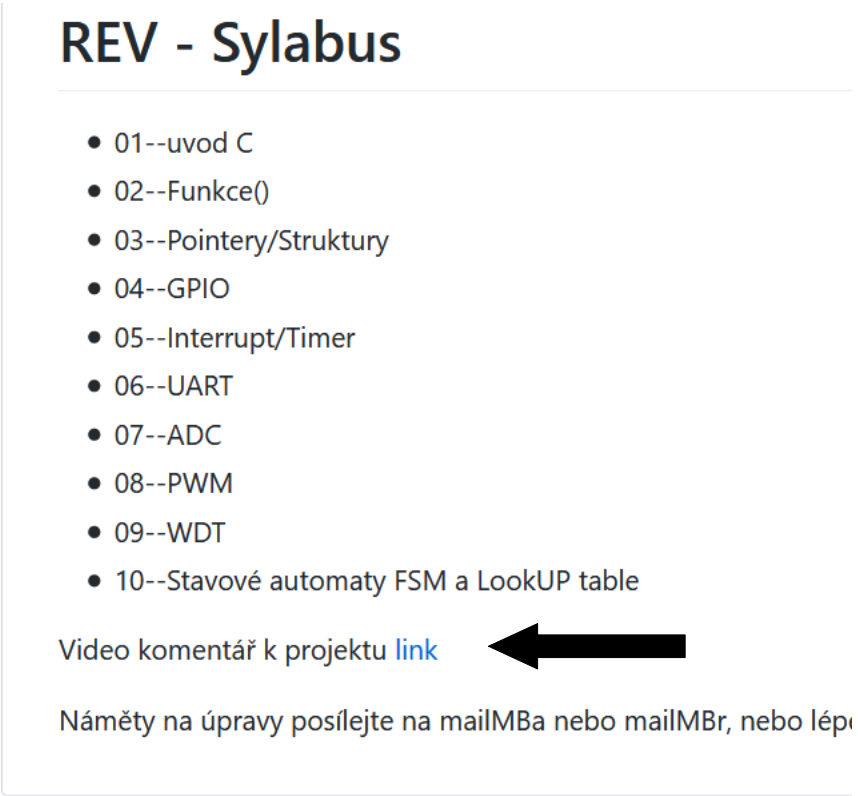
- Naprogramujete stavový automat, kde většina stavů charakterizuje základní práci s periferiemi.
- Poslední stav je vždy jednoduchá hra.
- Zadání si vygenerujete dle video-návodu. Vstupem je číslo kitu.

- Doporučujeme nepodcenit časovou náročnost projektu. Cca 20h

Odevzdání do e-learningu: odevzdává se celý projekt Prijmeni_Jmeno_final.zip

Čas na vypracování je 3 týdny!
Nepoužívejte knihovnu REV_basic (spíše Vám uškodí)!
LCD knihovnu k displeji používejte dle libosti.

Na github :
BUT-FME-REV/02_cv_zadani/
Je link na video



Slovo závěrem

- Kit lze většinu roku zapůjčit, také lze koupit cenově dostupný kit od Microchipu s modernějším PIC18.
- Takové kity obsahují i debuger/programátor, nic dalšího již nepotřebujete.
- Pokud již nemáte co objevovat na jednoduchém PIC18
- Jsou zde třeba dspic33
- Ve vodách 32bitů je potom mnohem více živo výborné jsou:

Microchip:
PIC32MK, PIC32MZ,
ATSAM4s, ATSAMD51,
ST:
stm32f4, stm32l4,
Texas instruments:
TM4C123, TM4C129

...a mnoho dalších

Curiosity Nano Evaluation Kit: