MKS3x3 – aktualizace interpretu BASIC

MKS3x3 je mnou vytvořený projekt (Open Source), který začal kdysi vyrobením prvního prototypu SW24X3, ale pak to nějak pro nedostatek času ustrnulo na mrtvém bodu a dále se nerozvíjelo. Nyní jsem měl opět trochu času „si hrát“ a vytvořil novou desku se 3 relé.

Jak mi čas dovolí, budu postupně doplňovat, rozšiřovat, cizelovat, vylepšovat, aktualizovat stávající jádro BASIC interpretu a pokud bude zájem, doplňovat hw o I/O komponenty (displeje, tlačítka, senzory, výstupy, RF modulátory, přijímače atd.).

Výhoda tohoto systému je především v tom, že pro jeho zprovoznění nepotřebujete žádný speciální, drahý programátor a nahrávání „pokusných“ programů je svižné přímo do RAMky procesoru bez degradace Flash paměti. Programy je možné editovat i přímo přes USB konzolu.

Možnosti dsPIC30F3013, 16bitový DSP procesor s DSP jádrem a instrukcemi, 12bitový A/D.

Bude-li mít někdo zájem, mohu mu poslat jak naprogramovaný dsPIC30F3013, tak i aktuální zdrojový program pro jeho další úpravu, rozšíření, vylepšení a hrátky. Byl bych rád, kdyby se pro tento projekt časem našlo pár nadšenců, podobně jako pro Arduino apod.

Rozšíření BASIC ze dne 29.června 2020

1. Podmíněný skok

110 IF <podmínka> THEN 320 <CR> … podmíněný skok na řádek 320 (pokud řádek 320 v programu chybí, program nebude spuštěn – LED RUN po spuštění programu pouze blikne).

Každá aritmetická a logická operace nastavuje příznak ZERO. Pokud je výsledek nulový, je ZERO=1, jinak je 0. Příznak lze testovat v podmínce takto:

115 IF **@Z=1** THEN 200 … podmínka splněna (skok na řádek 200), byl-li výsledek poslední ALU operace nulový.

116 IF **@Z=0** THEN 300 … podmínka splněna (skok na řádek 300), byl-li výsledek poslední ALU operace nenulový.

Do podmínkových skoků jsou nyní implementovány i konstanty:

110 IF P1<1500 THEN 320
120 IF P2=300 THEN 350
130 IF P3>0 THEN 200

POZN: Všechny proměnné i konstanty jsou typu WORD (Unsigned Integer), rozsah 0 až 65535.

2. Přiřazovací příkazy pro proměnné P0 až P9

Načtení stavu trimrů P1 a P2 a analogových vstupů AN2 a AN3 (nyní i pro 12bitový A/D)
(Původní vstupy AN0 až AN3 odpovídaly 7mi bitovému A/D s průměrováním vzorků)

400 P1=#4 <CR> … načte AN0 (stav trimru P1) v plném 12ti bitovém rozlišení.

401 P2=#5 <CR> … načte AN1 (stav trimru P2) v plném 12ti bitovém rozlišení.

402 P3=#6 <CR> … načte AN2 v plném 12ti bitovém rozlišení.

403 P4=#7 <CR> … načte AN3 v plném 12ti bitovém rozlišení (max. 4095).

3. Aritmetické a logické operace s jednou či více proměnnými

400 P1** … druhá mocnina (totéž co P1=P1*P1).

410 P1*3 … totéž co P1=P1*3, číslo může být v rozsahu 0 až 65535.

420 P1/5 … viz P1=P1/5, číslo může být v rozsahu 1 až 65535.

430 P2<< … rotace P2 vlevo o 1 místo (RLNC).

440 P2<4 … logický posun P2 vlevo o 4 místa (Logic Shift Left).

450 P3>> … rotace P3 vpravo o 1 místo (RRNC).

460 P3>8 … rotace P3 vpravo o 8 míst (Logic Shift Right).

470 P4! … logická negace P4 (NOT P4).

480 P5&7 … AND (viz P5=P5 AND 7).

490 P5|2 … OR (viz P5=P5 OR 2).

500 P5^4 … XOR (viz P5=P5 XOR 4).

510 P2=P4&P3 … logický součin P4 s P3. Výsledek ulož do P2 (P2=P4 AND P3).

520 P1=P4&31 … logický součin P4 s číslem 31. Výsledek ulož do P1.

530 P6=P7|P8 … logický součet P7 s P8 a výsledek ulož do P6 (P6=P7 OR P8).

540 P6=P7|128 … logický součet P7 s číslem 128 a výsledek ulož do P6 (P6=P7 OR 128).

550 P3=P1^P2 … logický exkluzivní součet (XOR) P1 s P2. Výsledek ulož do P3.

560 P3=P1^2 … logický exkluzivní součet P1 s číslem 2, výsledek ulož do P3 (P3=P1 XOR 2).

570 P4=P4>8 … rotace P4 vpravo o 8 míst (Logic Shift Right).

580 P4=P4>P5 … rotace P4 vpravo o P5 míst (Logic Shift Right).

590 P2=P1<4 … logický posun P1 vlevo o 4 místa a výsledek ulož do P2 (Logic Shift Left).

600 P2=P1<P4 … logický posun P1 vlevo o P4 míst a výsledek ulož do P2 (Logic Shift Left).

610 P7=P3>> … rotace P3 vpravo o 1 místo (RRNC) a výsledek ulož do P7.

620 P8=P2<< … rotace P2 vlevo o 1 místo (RLNC) a výsledek ulož do P8.

630 P1=P2*30 … vynásobí P2 s číslem 30 a výsledek uloží do P1.

640 P1=P3/40 … vydělí P3 číslem 40 a výsledek uloží do P1.

650 P1=P4+3 … k P4 přičte číslo 3 a výsledek uloží do P1.

660 P1=P5-7 … od P5 odečte číslo 7 a výsledek uloží do P1.

POZN: Mezi proměnnými, znaménky a indexy nesmí být žádný jiný znak ani mezera, jinak bude příkaz špatně vyhodnocen.

4. Příkaz READ – vstupy z konzole

100 READ1(P2) … program čeká na vložení znaku z konzole (UART1) do proměnné P2.

110 READ2(P3) … program čeká na vložení znaku z konzole (UART2) do proměnné P3.

120 READ1LN(P1) … program čeká na vstup čísla (UART1). Číslo se do proměnné P1 uloží až po odentrování, tj. odeslání znaku 0x0D. Číslo je automaticky přepočteno z ASCII znaků. Pokud nebude znaku 0x0D předcházet žádný jiný znak, bude do proměnné uloženo 0.

130 READ2LN(P1) … program čeká na vstup čísla (UART2). Číslo se do proměnné P1 uloží až po odentrování, tj. odeslání znaku 0x0D. ASCII znaky odesílané na port jsou převáděny na číslo, tj. 0x31 na 1, 0x32 na 2 … 0x39 na 9. Více znaků po sobě, předchozí číslo násobeno 10, tj. odesláním 0x31 0x32 0x33 0x0D bude v proměmné uloženo 123 (dekadicky).

5. Přidány nové řídící registry T0 až T9

Přiřazení přímých číselných hodnot a konstant řídícím registrům T0 až T9

T0 až T3 jsou časovače, každých 10ms je stav snížen o 1.

Při dočasování do 0 (vynulování) T0 sepne relé Re1.

Při dočasování do 0 (vynulování) T1 vypne relé Re1.

Při dočasování do 0 (vynulování) T2 vypne relé Re2.

Při dočasování do 0 (vynulování) T3 vypne relé Re3.

Registry T4 až T9 zatím bez specifické funkce – možno zatím využít jako rezervní proměnné (úschova stavu proměnných P0 až P9).

100 T1=1000 <CR> … přiřazení konstanty 1000 do T1 (za 10 vteřin se T1 vynuluje a vypne relé Re1 – program musí v té době běžet – jakékoliv násilné zastavení programu vypne všechna relé).

110 T2=21000 <CR> … přiřazení konstanty 21000 do T2 (za 210 vteřin se T2 vynuluje a vypne relé Re2 – program musí v té době běžet – jakékoliv násilné zastavení programu vypne všechna relé).

120 T3=200 <CR> … přiřazení konstanty 200 do T3 (za 2 vteřiny se T3 vynuluje a vypne relé Re3 – program musí v té době běžet – jakékoliv násilné zastavení programu vypne všechna relé).

130 T0=800 <CR> … přiřazení konstanty 800 do časovače T0 (za 8 vteřin se T0 vynuluje a současně zapne relé Re1 – program musí v té době běžet – jakékoliv zastavení programu časování vypne – nuluje časovače).

Zápis proměnných do registrů T0 až T9

400 T1=P7 <CR> … hodnotu proměnné P7 zkopíruj do T1, P7 beze změny.
410 T2=P2 <CR> … hodnotu proměnné P2 zkopíruj do T2, P2 beze změny.
420 T3=P1 <CR> … hodnotu proměnné P1 zkopíruj do T3, P1 beze změny.

Čtení registrů T0 až T9 do proměnných P0 až P9

500 P3=T0 <CR> … hodnotu časovače T0 zkopíruj do proměnné P3.
510 P4=T1 <CR> … hodnotu časovače T1 zkopíruj do proměnné P4.
520 P2=T9 <CR> … hodnotu registru T9 zkopíruj do proměnné P2.