USB3x3 – sekvenční automat s USB portem pro nahrávání programů

- **ovládání:** **4800**/9600bps – viz příkaz RKfg3, 8 bitů bez parity, 1 nebo 2 stop-bity
- **typy příkazů**: jednoznakový dotaz (vrátí stav vstupů IN1 až IN3), ovládací příkazy pro výstupy a příkazy nastavovací
- **jednoznakový dotaz č.1**: po přijetí znaku **!** ihned vrátí stav vstupů 1 až 3 a stav, zda modul čeká na dokončení poslední operace či nikoliv.

  Příkaz vrátí zpět:

  &000R* … žádný ze vstupů není aktivní (led LD1,2,3 nesvítí) a modul je Ready
  &000B* … hlásí neaktivní vstupy, modul čeká na dokončení operace (Busy)
  &100R* … vstup IN1 je aktivní (svítí LD1)
  &010R* … vstup IN2 je aktivní (svítí LD2)
  &001R* … vstup IN3 je aktivní (svítí LD3)
  &110R* … vstupy IN1 a IN2 jsou aktivní (svítí LD1 a LD2)
  &101R* … vstupy IN1 a IN3 jsou aktivní (svítí LD1 a LD3)
  &011R* … vstupy IN2 a IN3 jsou aktivní (svítí LD2 a LD3)
  &111R* … všechny vstupy jsou aktivní (svítí LD1, LD2 a LD3), modul Ready
  &111B* … všechny vstupy aktivní, modul je Busy – čeká na dokončení operace

- **jednoznakový dotaz č.2**: pokud se zpracovává interní program, pak po přijetí znaku **?** ihned vrátí zpět okamžitý stav vstupů IN1 až IN3 zakončený * (není-li aktivní žádný vstup, vrátí * , aktivní IN1 a IN3, vrátí **13***, apod.). Pokud se program nezpracovává, vrátí vždy jen *
- **příkazy ovládající výstupy – viz 1.2 a 1.3**:

  R<čísla_výstupů>=X,Y;

  R<čísla_výstupů>=X;

- **příkazy řídící chod modulu – viz 1.4**:

  RUN=0; RUN=1; RGoOn;

- **konfigurační příkazy – viz 1.5 až 1.7**:

  RESET=Y; RESET=N; RKfg1=0; RKfg1=1; RKfg2=0; RKfg2=1; RKfg3=0; RKfg3=1;

- **příkazy pracující s pamětí interního programu – viz 1.8**:

Rx[JMN12345ABCDEFWXYZ]=y;

1. **Ovládání výstupů - příkaz s jedním parametrem**

Obecně: **R<čísla_výstupů>=X;**

Kde **čísla výstupů** musí být v rozmezí 1 až 5, jinak je příkaz ignorován, přičemž 1 až 3 jsou výstupy určené pro relé, 4 a 5 pro led.

X je buď čas (pokud je X 2 až 999999) ve vteřinách nebo stav (1 , 0) – zapnuto, vypnuto

Příklady: **R1=1;** … zapne relé Re1
R123=1; … zapne všechna relé
R23=0; … vypne Re2 a Re3
R1=2; … za 2 vteřiny přepne relé Re1
R2=120; … za 2 minuty přepne Re2
R145=10; … za 10 vteřin přepne Re1 a červenou i modrou led

2. **Ovládací příkazy se dvěma parametry**

Obecně: **R<čísla_výstupů>=X,Y;**

Čísla výstupů – viz 1.2.

X je čas (1 až 999999) vteřin a Y počáteční stav (1 / 0) – zapnuto / vypnuto

Příklady: **R1=1,1;** … zapne relé Re1 a za vteřinu vypne
R12=1,0; … vypne Re1 a Re2 a za vteřinu je zapne
R23=20,1; … zapne Re2 a Re3 a za 20 vteřin obě vypne
R1=2,1; … zapne Re1 a za 2 vteřiny vypne
R2=60,1; … zapne Re2 a za minutu vypne

3. Řídící příkazy RUN a RgoOn

- příkazem **RUN=1;** bezprostředně po přijetí spustíme vykonávání vnitřního programu od prvního uloženého příkazu (tj. Ra…). Pokud není první příkaz platný nebo je type END, je provádění programu zastaveno a svítí žlutá led LD8. Běh programu je signalizován svitem zelené led LD9. Ihned po přijetí příkazu je do počítače odesláno **running***
- příkazem **RUN=0;** zastavíme provádění interního programu (pokud běží) a rozsvítí se žlutá led LD8. Do počítače se vrátí řetězec **stop*** a dále do počítače nebudou odesílány žádné změny na vstupech IN1 až IN3
- nastane-li v průběhu aktivního běhu programu na vstupech IN1 až IN3 událost, přenese se ihned do počítače příslušné číslo aktivovaného vstupu, např. 1 pro vstup IN1, atd.
- rozsah proudu každého ze vstupů IN1-3 by měl být v rozsahu 3 až 11mA (nepřekračujte)
- do počítače je rovněž možné posílat i stavy deaktivace vstupů (zhasnutí led LD1 až LD3), to je možné zapnutím reakce na obě hrany, nastavení provedeme příkazem **RESET=Ys**
- naopak, pokud nám stačí jen informace o sepnutí vstupu, odešleme **RESET=Ns**
- příkazem **RGoOn;** se přeruší všechny časovací operace a nulují všechny požadavky na čekání (je provedeno okamžité přerušení probíhajících časovacích operací a ukončení čekání na všechny vstupní události). Tento příkaz má význam především u krokování či trasování interního programu, kde je nastaveno buď dlouhé čekání na provedení nějaké operace nebo se čeká na vstupní událost, která není fyzicky ošetřena (chybějící čidlo na jednom z programem používaných vstupů apod.)

4. Konfigurační příkaz RESET – odesílání znaků jen při běhu programu

- po příkazu **RESET=Y;** bude při uvolnění vstupu **IN1** odeslán znak **A**, při uvolnění **IN2** odeslán znak **B** a při uvolnění **IN3** odeslán znak **C**
- po příkazu **RESET=N;** nebude při uvolnění vstupu **IN1** odeslán znak **A**, resp. **B** pro **IN2** apod. Odesílat se budou pouze číslovky 1 až 3 korespondující s aktivací příslušného vstupu IN1 až IN3.
- odesílání znaků je povoleno jen při běhu programu (po RUN=1;) a svítí-li zelená LD9
- po odeslání **RESET=Ns** bude do PC vráceno **L=N***
- po odeslání **RESET=Ys** bude do PC vráceno **L=Y***

Manuální aktivace/deaktivace alarmu

- po stisku **SET** se přepne režim modulu (START programu/ STOP) – viz příkaz RUN
- provádění programu je indikováno zelenou led LD9
- pokus od spuštění programu (START) - do počítače se vrátí řetězec **TEST=Ys***
- pokus od vypnutí programu (STOP) - do počítače se vrátí řetězec **TEST=Ns***
- **spustit program je možné, jen pokud svítí žlutá LD8 (modul v režimu STOP)**
- **zastavit program je možné, jen pokud svítí zelená LD9 (running)**

5. Konfigurační příkaz RKfg1=X; – potvrzování ukončení časování

- v některých případech potřebujeme navázat na uskutečnění předchozí operace, příkladem může být situace, kdy z počítače postupně přepínáme jednotlivé výstupy tak, aby přepnutí jednoho plynule navazovalo na následující, tj. po ukončení jedné operace byla ihned zahájena operace následující. Modul má pro tyto případy implementovanou funkci odeslání informace po ukončení časování a uskutečnění operace (viz ovládací příkazy)
- pokud nastavíme **RKfg1=1;** pak po každém ukončení časování bude zpět vrácen řetězec identifikující ukončení / provedení operace (**T1e* , T2e* , T3e* , T4e* , T5e***)
- naopak po nastavení **RKfg1=0;** nebudou tyto identifikace T1e* … nikdy odeslány
- každá změna RKfg1 je uložena do EEPROM
- po odeslání **RKfg1=0;** bude do PC vráceno **C1=0***
- po odeslání **RKfg1=1;** bude do PC vráceno **C1=1***

Příklad chování modulu po příkazu **RKfg1=1;**

R1=120,1; **...** zapne relé Re1 a po 2 minutách vypne a odešle do počítače **T1e***

Příklad chování modulu po příkazu **RKfg1=0;**

R1=120,1; **...** zapne relé Re1 a po 2 minutách vypne (žádná zpráva)

6. Příkaz RKfg2=X; – blokování příjmu při RUN=1 (interní program běží)

- **RKfg2=1;** pokud běží interní program, přijímání dat z PC je blokováno
- **RKfg2=0;** přijatá data z USB portu budou zpracovávána vždy

7. Konfigurační příkaz RKfg3=X; – nastavení přenosové rychlosti

- **RKfg3=1;** nastavení přenosové rychlosti na 4800bps (uloženo do EEPROM)
- **RKfg3=0;** nastavení přenosové rychlosti na 9600bps (uloženo do EEPROM)

8. Ukládání příkazů (instrukcí) do interní programové paměti

Uložen je každý platný příkaz, u něhož je po **R** písmenko malé abecedy v rozsahu **a** až **z**. Toto písmeno určuje pořadí příkazu v paměti, kam bude uložen. Takový příkaz nebude po odeslání z počítače vykonán, nýbrž pouze uložen na odpovídající paměťovou pozici.

Po spuštění programu (zapnutí modulu, příkazem RUN či stiskem tlačítka SET) je vždy jako první provedena instrukce na první pozici paměti – viz Ra….

Obecně: **Rx[instrukce];**

Kde **x** je písmenko malé abecedy v rozsahu **a** až **z**.

Instrukce – určuje význam dle níže uvedeného seznamu:

Seznam platných instrukcí

A==1 … čekání na vstupní událost na vstupu A, program bude v čekací smyčce, dokud nebude splněna podmínka IN1=1 (led LD1 bude svítit).

B==1 … totéž pro vstup B (IN2)

C==1 … totéž pro vstup C (IN3)

A==0 … čekání na vstupní událost na vstupu A, dokud nebude splněna podmínka IN1=0.

B==0 … platí pro vstup B, čekání do splnění podmínky IN2=0 (led LD2 nesvítí)

C==0 … totéž pro vstup C

AB==1 … čekej, dokud nenastane IN1=1 a IN2=1, přičemž události mohou přijít asynchronně

AC==1 … čekej, dokud nenastane IN1=1 a IN3=1, přičemž události mohou přijít asynchronně

Jsou možné různé kombinace až tří vstupů.

Příklad: **RcA==1;**

Na třetí programovou pozici (řádek programu) je uložena podmínka běhu A==1, tj. modul zde bude čekat na splnění podmínky vstupu A (IN1=1)

N=X … nastav opakování smyčky, X může být maximálně 199. Pokud je X=0 je naopak smyčka ukončena, počitadlo průchodů o 1 sníženo a pokud je nenulové, je návratová adresa první adresou smyčky, jinak je proveden následující příkaz.

N=0 … konec smyčky (význam viz N=X). Pokud není smyčka prováděna (byla ukončena nebo nebyla definována), je program ukončen, zastaven (svítí žlutá led LD8)

Příklad: **RaN=0;**

Na první programovou pozici je uloženo N=0, což značí ukončení provádění programu, neboť smyčka dosud nebyla definována.

N=1 (obecně N=X, kde X>0) … nastavení nového počátku smyčky bez ohledu, zda je již program ve smyčce vykonáván či nikoliv. Předchozí počáteční adresa smyčky bude nenávratně zapomenuta, bude přepsána novou hodnotou. Stejně tak nový počet X přepíše stávající počítadlo průchodů.

N=0,0 … jiná syntaxe N=0, význam stejný
N=0,1 … jiná syntaxe N=0, význam stejný

NX=0,Y … viz N=0 (význam konec smyčky, počitadlo průchodů sníženo o 1 a pokud je rovno 0, je smyčka ukončena), smyčka může být ukončena rovněž z jiné příčiny, viz splnění podmínky X a Y dle následujícího seznamu:

NA=0,0 … ukonči smyčku, pokud je vstup **A==0** (IN1=0)
NA=0,1 … ukonči smyčku, pokud je vstup **A==1** (IN1=1)
NB=0,0 … ukonči smyčku, pokud je vstup **B==0** (IN2=0)
NB=0,1 … ukonči smyčku, pokud je vstup **B==1** (IN2=1)
NC=0,0 … ukonči smyčku, pokud je vstup **C==0** (IN3=0)
NC=0,1 … ukonči smyčku, pokud je vstup **C==1** (IN3=1)

NAB=0,0 … ukonči smyčku, pokud je vstup **A** nebo **B==0**
NAB=0,1 … ukonči smyčku, pokud je vstup **A** nebo **B==1**
NAC=0,0 … ukonči smyčku, pokud je vstup **A** nebo C==0
NAC=0,1 … ukonči smyčku, pokud je vstup **A** nebo **C==1**
NBC=0,0 … ukonči smyčku, pokud je vstup **B** nebo **C==0**
NBC=0,1 … ukonči smyčku, pokud je vstup **B** nebo **C==1**

NABC=0,0 … ukonči smyčku, pokud je vstup **A** nebo **B** nebo **C==0**
NABC=0,1 … ukonči smyčku, pokud je vstup **A** nebo **B** nebo **C==1**

N&AB=0,0 … ukonči smyčku, pokud je vstup **A** a současně i **B==0**
N&AB=0,1 … ukonči smyčku, pokud je vstup **A** a současně i **B==1**
N&AC=0,0 … ukonči smyčku, pokud je vstup **A** a současně i **C==0**
N&AC=0,1 … ukonči smyčku, pokud je vstup **A** a současně i **C==1**
N&BC=0,0 … ukonči smyčku, pokud je vstup **B** a současně i **C==0**
N&BC=0,1 … ukonči smyčku, pokud je vstup **B** a současně i **C==1**

N&ABC=0,0 … konec smyčky, pokud jsou všechny vstupy **A,B,C==0**
N&ABC=0,1 … konec smyčky, pokud jsou všechny vstupy **A,B,C==1**

Příklad: **RfNA=0,1;**

Na šestém programovém řádku (f) je uložena podmínka pro ukončení cyklu. Splněno bude buď při nulovém počitadle průchodů nebo splnění podmínky vstupu **A==1** (IN1=1)

1=X … viz příkaz R1=X, kapitola 1.2 – ovládání výstupů

Příklad: **Rb23=1;**

Na druhém programovém řádku (b) je uložen příkaz, kterým zapneme relé Re2 a Re3.

1=X,Y … viz příkaz R1=X,Y, kapitola 1.3 – ovládání výstupů, dvouparametrový příkaz.

Příklad: **Rd2=12,1;**

Na čtvrtém programovém řádku (d) je uložen příkaz, kterým zapneme Re2 a za 12 vteřin vypneme.

J=x … instrukce nepodmíněného skoku na řádek programu pod písmenkem **x** (x v rozsahu a až z)

JA=x … instrukce podmíněného skoku, A určuje podmínku, při jejímž splnění je skok proveden. Pokud podmínka splněna není, pokračuje program na následujícím řádku.

Příklad: **ReJ=h;**

Na pátém programovém řádku (e) je nepodmíněný skok (Jump) na osmý řádek (h).

Seznam podmínek skoků:

A … splněna, pokud je vstup A (IN1==1)

B … splněna, pokud je vstup B (IN2==1)

C … splněna, pokud je vstup C (IN3==1)

D … splněna, pokud je vstup A (IN1==0)

E … splněna, pokud je vstup B (IN2==0)

F … splněna, pokud je vstup C (IN3==0)

0 … viz nepodmíněný skok

1 … splněna, pokud je sepnuté relé Re1

2 … splněna, pokud je sepnuté relé Re2

3 … splněna, pokud je sepnuté relé Re3

4 … splněna, pokud je aktivní výstup č.4

5 … splněna, pokud je aktivní výstup č.5

Z … skok, pokud je čítač smyčky roven nule

Y … dekrementuj čítač smyčky a proveď skok, pokud je čítač smyčky roven nule

X … skok, pokud je čítač smyčky neroven nule

W … dekrementuj čítač smyčky a proveď skok, pokud je čítač smyčky neroven nule

Příklad: **RkJD=a;**

Na jedenáctém programovém řádku (k) je podmíněný skok (Jump) na začátek programu (a).
Pokud bude v okamžiku provádění instrukce **JD=a** splněno IN1==0, pak bude skok proveden.
V opačném případě bude program pokračovat na dvanáctém řádku.

9. Interní uživatelská paměť EEPROM

Kromě paměti instrukcí popsané v předchozí kapitole je možné pracovat i s uživatelskou EEPROM pamětí pro ukládání dat. Instrukce zápisu bude ihned po přijetí provedena.

K dispozici je 9 pamětí, přičemž první paměťová buňka má specielní význam pro blokování autonomního spouštění programu po zapnutí napájení. Pokud bude mít paměť číslo 1 nulový obsah, nebude program nikdy spuštěn po zapnutí i přesto, že bude zaveden. Hodnota paměti číslo 1 nemá vliv na spouštění tlačítkem SET nebo příkazem RUN.

Paměť je přístupná pomocí instrukcí zápisu do paměti (Memory Write):

Přímý zápis čísla: **RMWx=číslo;**

Kde x musí být číslice v rozsahu 1 až 9 a určuje paměťovou pozici (číslo paměti)

Inkrementace obsahu paměti - nepřetéká: **RMWx++;**

Dekrementace obsahu paměti - nepodtéká: **RMWx--;**

Příklady:

RMW8=100;… do paměti č.8 bude uloženo 100

RMW1=1; … do paměti č.1 bude uložena 1 (po zapnutí modulu bude program spuštěn)

RMW8++; … obsah paměti č.8 zvýšen o 1, tj. aktuální hodnota 101, viz první příklad

RMW1--; … obsah paměti č.1 snížen o 1, tj. akt. hodnota 0 (blokování autostartu)

RMW1--; … hodnota zůstává na 0 (obsah nepodtéká do 255), paměťovou buňku s nulovým obsahem lze pouze čítat nebo přepsat přímým zápisem čísla

RMW9=255;… paměť číslo 9, maximální hodnota, viz paměťová buňka typu **byte**

RMW2=256;… paměť číslo 2, hodnota = 0, viz paměťová buňka typu **byte**

RMW1=150;… do paměti č.1 bude uloženo 150 (autostart nebude blokován)
Bude-li MW předcházet písmeno malé abecedy, instrukce provedena nebude, nýbrž jen uložena do programové paměti. Pomocí **RaMW1--;** tak je možné počítat, kolikrát se program či určitá jeho část či programová smyčka vykonala.

Například chceme, aby se program vykonal právě 10x (vždy 1x po zapnutí napájení)

Do paměti č.1 uložíme číslo 10 jednorázově příkazem **RMW1=10;**

Po každém zapnutí napájení se program spustí a než bude ukončen, provede instrukci **RxMW1--;**

Po jedenáctém zapnutí napájení se již program neprovede (paměť č.1 bude obsahovat 0)

Příklad: na osmý programový řádek uložíme instrukci dekrementující obsah paměti č.1 takto:

RhMW1--;