Dokumentace k projektu **Simulace v CW: Konfigurovatelný čítač** do předmětu IMP, 2015

Zadání RTC, jazyk assembler HCS08

Jan Jileček, xjilec00

Popis implementace

Podle zadání projektu jsem vypracoval čítač s více režimy. Programoval jsem v jazyku assembler HCS08, zadání mi také specifikovalo použití Real-Time counteru.
První režim je STOP, ve kterém je čítač v pozastaveném stavu. Po zapnutí prvním přepínačem se přepne do režimu START, ve kterém se v závislosti na poloze dalších přepínačů přepne do režimu SET nebo CITANI.

Detaily

- Při režimu čítání jsou indexy nibblů zobrazeny v pořadí 3 2 1 0, nebo také little endian. V režimu set jsou nibbly zadávány v režimu 0 1 2 3, neboli big endian. V zadání je zmíněn zápis v little endianu i big endianu, proto jsou implementovány oba, jeden v režimu set a druhý v režimu čítání.
- Do 16bitové proměnné *CNT* je uložena hlavní hodnota. Další proměnné, určující nibbly, jsou *CNTNbPrvniNibble, CNTNbDruhyNibble, CNTNbTretiNibble a CNTNbCtvrtyNibble*.
- Levý displej je namapován na port B. pravý displej na port D a DILSwitch na port E.
- Nastavení hodin je v proměnné *RTCClockSetting*. Hodiny jsou nastaveny na 16ms, protože kód v realitě běžel přibližně 10x pomaleji.
- Obsluha přerušení má název *clockInterruptService* a obstarává hlavní část čítání a blikání.
- Popis dalších důležitých proměnných je v kódu u jejich deklarace ve formě komentáře.

Implementace

Program začíná inicializací proměnných, následuje vypnutí watchdogu a nastavení Data Direction a Data bitů pro porty periferií. Také je povoleno přerušení a následuje skok do hlavní smyčky.

mainLoop

Zde je volána subrutina nactiDILSwitch.

nactiDILSwitch

Pokud je na DILSwitchi nastaven bit 7, program se přepne do režimu START. Pokud je bit nenastaven, zůstává v režimu STOP.

rezimStop

Uloží se do proměnné číslo současného režimu (0), bude potřeba v obsluze přerušení. Nastaví se Data a DD jednotlivých portů na původní hodnoty (kromě bitu 7 na DILSwitch, co slouží pro zapínání, ten je ponechán ve stávající poloze). Je také vynulován CNT.

rezimStart

Zde je podle bitu 6 skočeno na režim Set nebo Citani.

rezimCitani

Uloží se číslo režimu (3), zapnou se hodiny, následuje zpomalení. Následně je zálohována proměnná CNT a čeká se na přerušení.

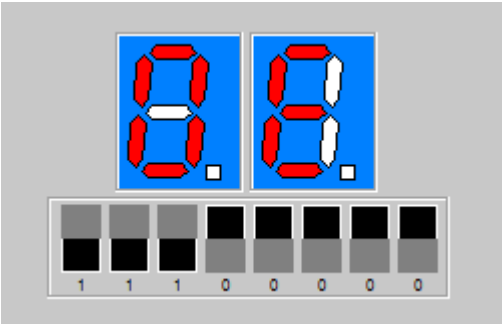
clockInterruptService, režim Čítání

Zde následují testy, pokud je režim čítání, je v závislosti na bitu 5 na DILSwitchi provedena buď inkrementace, nebo dekrementace.

Po dokončení subrutiny zvys/snizCNT následuje rozsáhle testování nibblů. Pomocí proměnné *compareCNT* a funkcí XOR(EOR) a AND je zjištěno, jaké bity se změnily. Proměnná CNT je nahrána do H:X, kde je postupně testován registr H a poté X na změny. V závislosti na tom, ve kterém nastaly

změny, jsou testovány i jejich nibbly. Je určen naposledy změněný nibble a na základě toho je uložen jeho index, který bude následně zobrazen na levém displeji. Nejméně významný nibble má index 0.

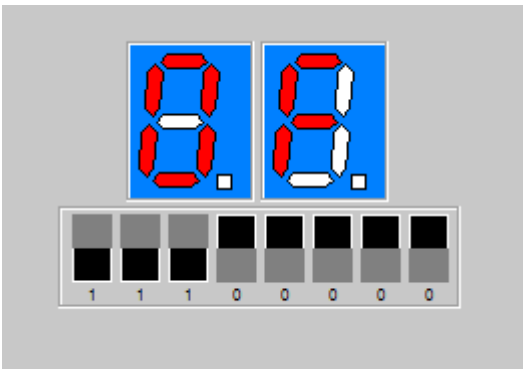
V tom případě vypadá např. číslo 0x000E následovně:



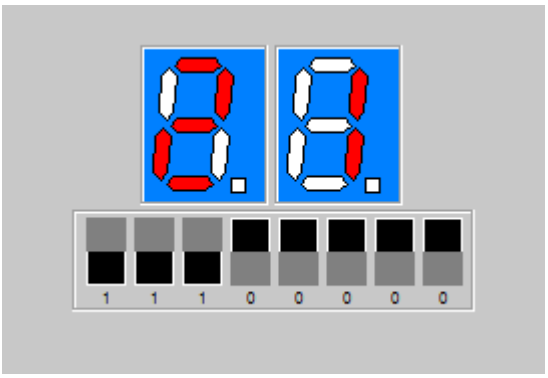
V závislosti na zvoleném nibblu je uloženo a zobrazeno i číslo na pravý displej.

Přechod z 255 na 256, demonstrace změny nejrychleji se měnícího nibblu.

CNT	255	int
CN	0	int
compareCNT	255	int
myH	0	unsigned char
myX	255	unsigned char
CNTNbPrvniNibble	0	unsigned char
CNTNbDruhyNibble	0	unsigned char
CNTNbTretiNibble	15	unsigned char
CNTNbCtvrtyNibble	15	unsigned char
currentNibble	0	unsigned char
currentNibbleValue	0	unsigned char
currentDisplay	1	unsigned char
currentNumber	15	unsigned char



CNT	256	int
CN	0	int
compareCNT	256	int
myH	1	unsigned char
myX	0	unsigned char
CNTNbPrvniNibble	0	unsigned char
CNTNbDruhyNibble	1	unsigned char
CNTNbTretiNibble	0	unsigned char
CNTNbCtvrtyNibble	0	unsigned char
currentNibble	0	unsigned char
currentNibbleValue	0	unsigned char
currentDisplay	1	unsigned char
currentNumber	1	unsigned char



zvysCNT

do registru H:X je načtena proměnná CNT. Následuje porovnání, zda již není dosaženo horní hranice 0xFFFF, v tom případě je skočeno na rutinu *trikratBlikniAZastav.* Jinak je využito instrukce AIX přičtení jedničky a uložení H:X zpět do CNT.

snizCNT

do registru H:X je načtena proměnná CNT. Následuje porovnání, zda již není dosaženo dolní hranice 0x0000, v tom případě je skočeno na rutinu *trikratBlikniAZastav.* Jinak je využito instrukce AIX k odečtení jedničky a uložení H:X zpět do CNT.

trikratBlikniAZastav

Tento mód má číslo 5. Pokud je proměnná *pocetBliknuti* rozdílná od nuly, je snížena a čeká se na přerušení, kde je dále obsluhováno blikání. Jinak se přepne DILSwitch do stavu vše vypnuto (0x00) a je skočeno do *rezimStop.*

clockInterruptService, mód trikratBlikniAZastav

Pokud je detekován tento mód v obsluze přerušení, je skočeno na návěstí, ve kterém se pomocí instrukce COM (NOT) notuje stav displejů. Další volání nastává až při dalším přerušení cca za 1 sekundu (podle nastavení RTC), tudíž je vytvořen dojem bliknutí.

rezimSet

V případě nastavení bitu 6 DILSwitche na pozici 0 je zapnut tento režim. Je zde volána subrutina *zmenHodnotuAktivnihoDosplejei,* ve které je řešeno veškeré chování režimu Set. Následuje čekání na přerušení, kde je řešeno již popsané bliknutí. Na počátku je aktivní pouze levý displej, pravý displej se inicializuje až při přepnutí se na něj.

zmenHodnotuAktivnihoDispleje

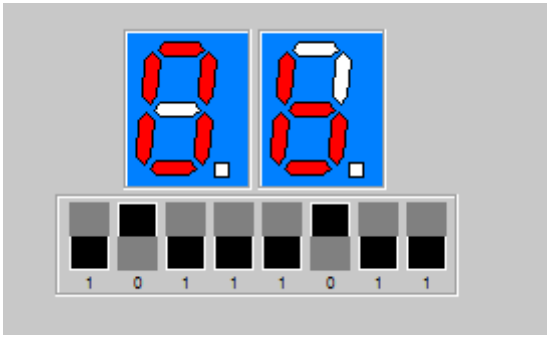
nejprve je zjištěn stav přepínače 4, který určuje zda bude pracováno s levým nebo pravým displejem, tzn. buď s indexem nibblu, nebo s jeho hodnotou. Následně je načtena hodnota dolních 4 bitů přepínače DILSwitch, které jsou dále použity.

Hodnota načtená z přepínačů je ihned zobrazena na aktivním displeji (zobrazování, segmenty displeje atp. řešeno níže). Následně je určen index nibblu na levém displeji a hodnota rovnou uložena. Při jakékoliv změně je přímo uložena hodnota na zobrazena na pravém displeji do právě aktivního nibblu, jehož index je zobrazen na levém displeji. Změna pouze určitého nibblu je zajištěna v návěstích *ulozeniPrvniNibble, ulozeniDruhyNibble* atd. Je řešeno pomocí operací AND, OR a LEFTSHIFT. Nová hodnota je uložena do CNT.

V tomto režimu má nejvyšší nibble hodnotu 0, neboli big endian. Příklad:

(předpokládáme CNT 0x0000)

```
currentNibble          0 unsigned char
currentNibbleValue    11 unsigned char
currentDisplay         1 unsigned char
currentNumber         11 unsigned char
```




```
A   2
HX  B000   SP  10AB
SR  60     Status  VHINZC
PC  1B94
```

H:X je B000, je v něm teď uloženo CNT

displayNumberDef

Má za úkol načíst právě uložené číslo (v závislosti na přepínačích to může být hodnota nibblu nebo jeho index) a podle toho se přepne na načtení segmentů pomocí *zobrazJedna, zobrazDva* atd.

zobrazJedna, zobrazDva

Načte odpovídající hodnotu ze statové tabulky na začátku programu a uloží ji do proměnné *currentNumberSegments*. Tuto proměnnou používá subrutina *displayIt.*

displayIt

Zobrazí segmenty na aktivním displeji. Načte proměnnou *currentNumberSegments* a uloží na port specifikovaný konstantou *XXDisplayAdr.*

Analýza paměťových nároků aplikace

RAM: Code: 1000

Flash: Data: 0

Velikost kódu: cca 650