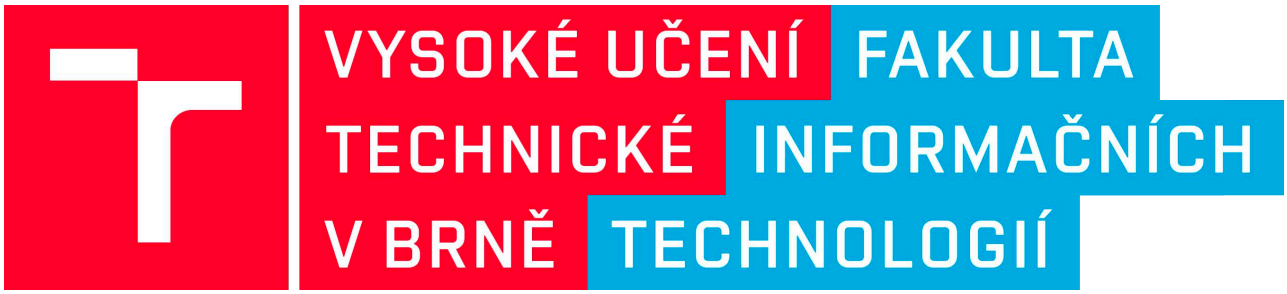
Vysoké Učení Technické v Brně




# Simulace v CW: Konfigurovatelný OCT čítač

Autor: Jan Pawlus, xpawlu00@stud.fit.vutbr.cz

# Obsah

# 1 Úvod

Tato dokumentace popisuje projekt do předmětu IMP v kontextu vývoje, funkčnosti a paměťových nároků výsledné aplikace.

# 2 Program z pohledu dat

## 2.1 Proměnné, konstanty a makra

Program je založen na několika proměnných, konstantách a makrech. Samotný čítač je založen na poli `byte` jménem `CNTN`, které reprezentují jednotlivé `nibble` a dohromady dává jako celek čítač `CNT`. Pro výběr v poli `CNTN` slouží `byte index`, což je proměnná indexující aktuálně zobrazovaný `nibble`. Dále je zadefinováno konstantní pole typu `byte` s názvem `Number`, ve kterém jsou uloženy hexadecimální hodnoty reprezentující jednotlivé číslice pro výpis na sedmisegmentový displej. Následuje další proměnná typu `byte cycleCount`, do které se zaznamenává, kolikrát čítač změnil stav. Poslední proměnná typu `byte` je `final`, která udává, zda se již čítač dostal do finálního stavu.

Program obsahuje několik maker, které je výhodně použít, jelikož nezabírají paměť `RAM`. Takto jsou definovány volativní ukazatele ukazující na adresy do paměti, ze kterých čtou hodnoty oba sedmisegmentové displeje `leftDisplay` a `rightDisplay`) a `DILSwitch`, který slouží k nastavení čítače. Jednotlivé bity čítače jsou popsány takto:

Table 1 – 8-bitový DILSwitch

| **Bit** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Význam** | Start | Count | Dir | Select | Data[3] | Data[2] | Data[1] | Data[0] |

Kolekci maker doplňují definované hexadecimální hodnoty znázorňující stavy přepínače `DILSwitch` (`ModeUp` - čítání nahoru, `ModeOn` - zapnuté hodiny a přerušení, `ModeRight` - výběr pravého displeje při režimu `Set`).

## 2.2 Funkce

Hlavní funkcí je samozřejmě `main()`, která obsahuje nekonečnou smyčku, kde se odehrává řídící logika aplikace z pohledu stavů jednotlivých přepínačů `DILSwitch`. Zde se nastavuje modul `RTC`, pokud má být čítač spuštěn (`Start == 1`). Je třeba nakonfigurovat přerušení a jak často se má generovat. Smyslem přerušení z `RTC` modulu je samotné čítání. V obsluze přerušení se tedy buď změní stav čítače (funkce `count()` pokud `Count == 1`), nebo je zavolána funkce `flash()` (`Count == 0`), která zařídí blikání aktuálně vybraného displeje (bit `Select`) v módu `Set`.

Program má pomocné funkce jako například `setDisplay`, která po zavolání nastaví na levý sedmisegmentový displej proměnnou `index` a na pravý hodnotu `CNTN[index]`, nebo `getState()`, která zjistí hodnotu určitého bitu `DILSwitch` zamaskováním všech ostatních bitů. Dále `setIndex()`, která bude rozebrána dále. V neposlední řadě `defaultState()` a `finalState()`, které uvedou čítač do počátečního, resp. finálního stavu.

### 2.3 Paměťová náročnost

Ze souboru `Project.map` je možné vyčíst paměťovou náročnost aplikace.

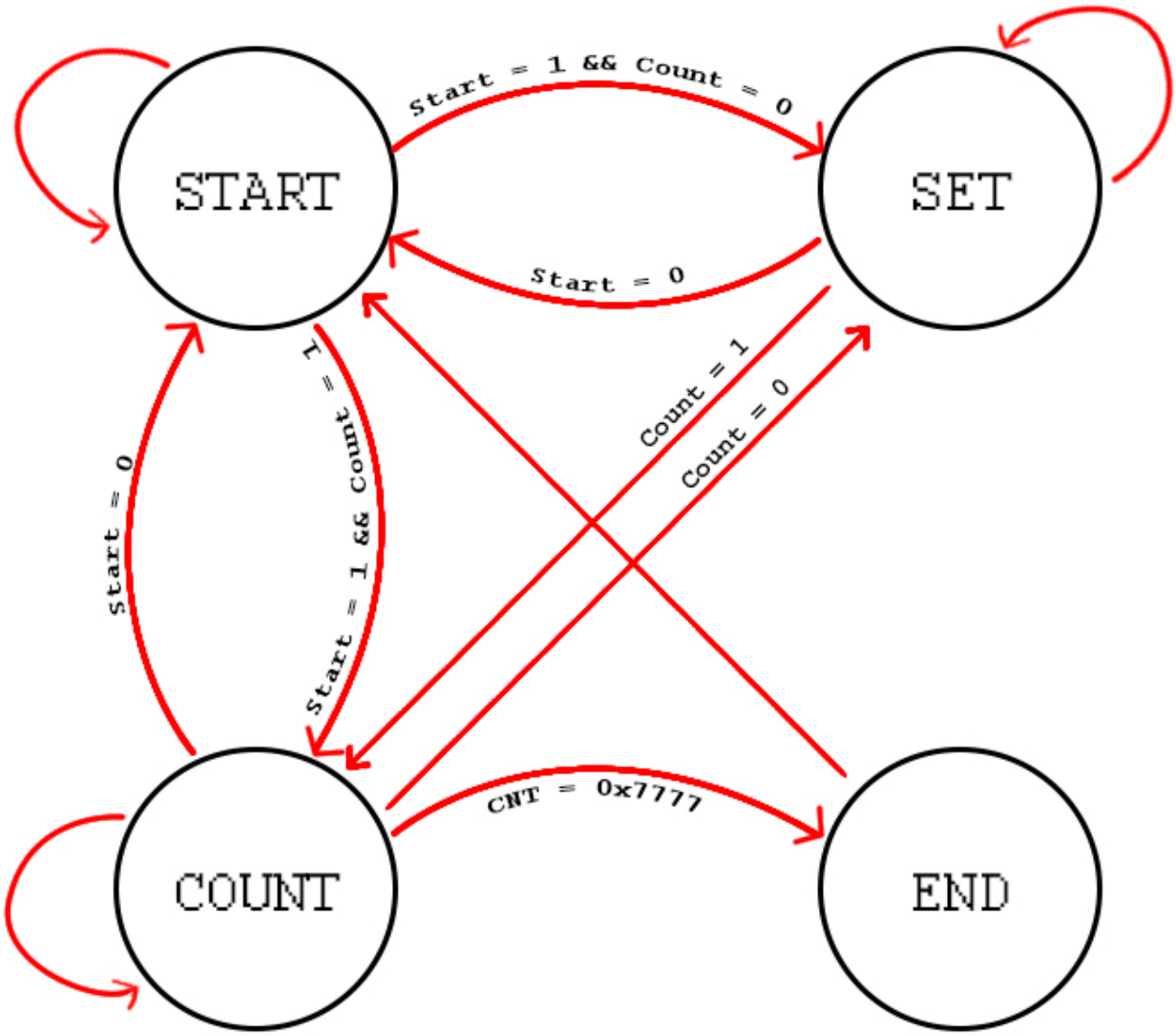
Table 2 – Paměťová náročnost

| **Typ** | RAM | ROM | Řádky |
|---|---|---|---|
| **Velikost** | 138 B | 1293 B | 214 |

Do paměti RAM se ukládají globální proměnné, lokální proměnné a zásobník. Do paměti ROM pak konstanty a samotný kód. Na základě dostupného místa v hardwaru lze dělat podle potřeby z konstant globální proměnné (a naopak), tím se sníží (zvýší) náročnost na ROM a zvýší (sníží) náročnost na RAM.

## 3 Program z pohledu funkčnosti a kódu

### 3.1 Stavový automat



## 3.2 Chování programu

Program začíná v defaultním stavu a na základě přepínačů se mění jeho chování. Bit `Start` nastavuje hodinový signál do `RTC` modulu a aktivuje přerušení. Bit `Count` ovlivňuje, jestli program čítá (směr je dán hodnotou bitu `Dir`), nebo je ve stavu `Set`, který na základě bitu `Select` dovede nastavovat hodnotu indexovaného `CNTN`, resp. samotný index. Nastavovaná hodnota/index se přečtou ze spodních 4 bitů `DILSwitch`. Čítání probíhá v obsluze přerušení od `RTC`, stejně jako blikání sedmisegmentového displeje v režimu `Set`.

Pokud se přepínač `Start` nastaví v jakémkoliv bodě do nuly, vypne se hodinový signál v `RTC` a vše je nastaveno do defaultního stavu. Pokud program dosáhne maximální hodnoty (Každé `CNTN` je v hodnotě 0111 značící maximální hodnotu v osmičkové soustavě), sedmisegmentový displej ve třech cyklech obsluhy přerušení změní všechny bity na displeji do nuly, resp. konečných hodnot (blikání) a vše se nastaví do defaultního stavu.

## 3.3 Zobrazování na displejích

Jednou z otázek bylo, co přesně zobrazovat na sedmisegmentových displejích. Dle zadání je to index nejpomaleji měnícího se `nibble` a jeho hodnoty, toto je ale dost nejednoznačné - nejpomaleji se mění pokaždé třetí `nibble` a na displeji by se v takovém případě dlouhou dobu nic neměnilo. Proto jsem se rozhodl zobrazovat index a hodnotu nejpomaleji měnícího se `nibble`, jehož hodnotou není nula a zároveň sedm. po každé změně stavu čítače tedy proběhne funkce `setIndex`, která v cyklu projede všechny `nibble` a nastaví největší možný index odpovídající výše zmíněným kritériím.

```
void setIndex() {
  byte i = 0;
  index = 0;
  for (i = 0; i <= 3; i++) {
    if ((CNTN[i] != 0 && CNTN[i] != 7) && i > index)
      index = i;
  }
}
```

## 3.4 Čítání

Čítání probíhá v obsluze přerušení od modulu `RTC`. Pokud nastane přerušení a bit`Count` obsahuje hodnotu 1, čítač změní stav na základě směru čítání (bit `Dir`). Následuje ukázka pro čítání směrem nahoru (pro směr dolů je kód stejný, pouze naopak).

```
// pokud je citac na maximalni hodnote, prejde do finalState
if (CNTN[0] == 7 && CNTN[1] == 7 && CNTN[2] == 7 && CNTN[3] == 7) {
  cycleCount = 3;
  final = 1;
  finalState();
}
```

```
// jinak cita
else {
  for (i; i <= 3; i++) {
    if (i == 0) {
      CNTN[i]++;
      cycleCount++;
    }
    else if (CNTN[i-1] == 8) {
      CNTN[i]++;
      CNTN[i-1] = 0;
    }
  }
  setIndex();
  setDisplay();
}
```

### 3.5 Set

Do stavu `Set` se program dostane, pokud je bit `Start` v jedničce a `Count` v nule. Je znázorněn funkcí `set()`, která je popsána takto:

```
void set() {
  byte temp = DILSwitch & 0xF;
  // spodni 4 bity DILSwitch nenulove - je mozne nastavit CNTN
  if (temp != 0) {
    // pokud se nastavuje hodnota CNTN, tj. pravy display
    if (GetState(ModeRight)) {
      // posledni tri bity dat (hodnota muze nabyvat maximalne 7)
      temp = DILSwitch & 0x7;
      // pokud se hodnota zmenila
      if (CNTN[index] != temp) {
        CNTN[index] = temp;
        setDisplay();
      }
    }
    // pokud se nastavuje index, tj. levy display
    else {
      // posledni dva bity dat (index muze nabyvat maximalne 3)
      temp = DILSwitch & 0x3;
      // pokud se hodnota zmenila
      if (index != temp) {
        index = temp;
        setDisplay();
      }
    }
  }
}
```