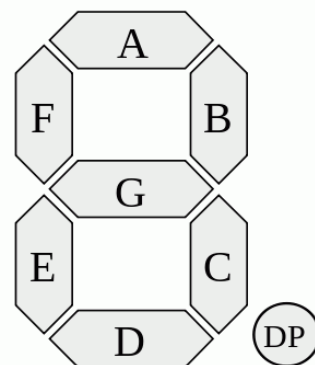




LED displej na posuvném registru

1 Zadání

- Vytvořte driver pro dvojici posuvných registrů SCT2024 na zásuvné desce. Driver bude v nezávislém modulu pro oddělený překlad. Bude obsahovat minimálně globální funkce `sct_init()` pro inicializaci hardwaru a `sct_led(uint32_t value)` pro nastavení svitu jednotlivých segmentů a LED dle bitových pozic hodnoty `value`. Driver otestujte **(1b)**.
- Vytvořte sedmisegmentové znaky pro číslovky 0 až 9 na jednotlivých pozicích a funkci pro zobrazení čísla v rozsahu 000-999. Displej bude ukazovat hodnoty od 0 do 999 s krokem 111 **(1b)**.
- Vytvořte nový projekt, který bude realizovat stejnou funkci s využitím HAL knihoven **(1b)**.
- Doplňte obsluhu rotačního enkodéru, pomocí které budete měnit hodnotu zobrazovanou na sedmisegmentovém displeji. Minimální hodnota enkodéru na displeji bude 0, maximální 150 **(1b)**.
- Hodnocena bude i úprava zdrojového kódu, zejména odsazování bloků.



2 Návod

2.1 Základní seznámení

- Vytvořte si pracovní kopii svého repozitáře z Githubu (Git Clone), příp. aktualizujte repozitář ze serveru (Git Pull).
- Základní strukturu první části můžete využít (tj. okopírovat) z prvního cvičení, příp. lze založit nový projekt stejným postupem a přidat knihovny CMSIS. Po okopírování projektu je potřeba jej přidat do STM32CubeIDE pomocí File / Import / General / Existing Projects into Workspace. **Každý dokončený bod commitujte.**

2.2 Driver pro posuvný registr

- Driver se bude skládat ze souborů `sct.c` a `sct.h`, které založíte pomocí File / New / Source/Header File.
- Zadejte si funkce, resp. jednoduchá makra pro nastavení jednotlivých signálů posuvných registrů `sct_nla()`, `sct_sdi()`, `sct_clk()` a `sct_noe()`. Lze zapsat např. takto:

```
#define sct_nla(x) do { if (x) GPIOB->BSRR = (1 << 5); else GPIOB->BRR = (1 << 5); } while (0)
```
- Funkce `void sct_init(void)` provede povolení hodin příslušných portů (`RCC_AHBENR_GPIOB`), inicializaci příslušných signálů jako výstupních (`GPIO_MODER_MODERx_0`), zápis nul do posuvných registrů (volání `sct_led(0)`) a aktivování výstupu pomocí `/OE` (volání `sct_noe(0)`).
- Funkce `void sct_led(uint32_t value)` provede zápis do posuvných registrů tak, aby jednotlivé bity 0 až 31 hodnoty `value` odpovídaly jednotlivým sedmisegmentovkám a LED diodám na desce. Je třeba bity z `value` postupně vysouvat na SDI (směrem doprava, tj. začít LSB) a následně generovat puls na CLK. Po odeslání všech 32 bitů provést zápis do výstupních registrů pulzem na `/LA`. Detaily naleznete v datasheetu SCT2024, zejména na str. 5.
- Pro otestování použijte konstantu `0x7A5C36DE`. Na displeji se zobrazí „bYE“ a LED se střídavě rozsvítí.



2.3 Zobrazení číslovek a hodnoty

- Vytvořte funkci `void sct_value(uint16_t value)`, která na displej zapíše hodnotu `value`.
- Pro překlad číslovek na odpovídající segmenty připojené na posuvné registry je vhodné použít tabulku, která může vypadat takto:

```
static const uint32_t reg_values[3][10] = {
{
    //PCDE-----GFAB @ DIS1
    0b0111000000000111 << 16,
    0b0100000000000001 << 16,
    0b0011000000000101 << 16,
    0b0110000000000101 << 16,
    0b0100000000000110 << 16,
    0b0110000000000110 << 16,
    0b0110000000000110 << 16,
    0b0100000000000011 << 16,
    0b0111000000000111 << 16,
    0b0110000000000111 << 16,
},
{
    //----PCDEGFAB---- @ DIS2
    0b0000011101110000 << 0,
    0b0000010000010000 << 0,
    0b0000011101110000 << 0,
    0b0000011010110000 << 0,
    0b0000010011010000 << 0,
    0b0000011011100000 << 0,
    0b0000011111100000 << 0,
    0b0000010000110000 << 0,
    0b0000011111110000 << 0,
    0b0000011011110000 << 0,
},
{
    //PCDE-----GFAB @ DIS3
    0b0111000000000111 << 0,
    0b0100000000000001 << 0,
    0b0011000000000101 << 0,
    0b0110000000000101 << 0,
    0b0100000000000110 << 0,
    0b0110000000000110 << 0,
    0b0111000000000110 << 0,
    0b0100000000000011 << 0,
    0b0111000000000111 << 0,
    0b0110000000000111 << 0,
},
};
```

- Jednotlivé číslovky přidávejte pomocí OR do 32bitové proměnné, např. pro pozici stovek lze použít:
`reg |= reg_values[0][value / 100 % 10];`
- Otestujte v hlavní smyčce pomocí čítání od 0 do 999 s krokem 111. Proveďte commit pracovní kopie do Gitu.

2.4 Použití HAL knihoven

- Založte nový projekt přes File / New / STM32 Project / Board Selector / NUCLEO-F030R8. Budeme využívat HAL knihovny, proto ponechte Targeted Project Type na STM32Cube. Potvrďte inicializaci všech periférií do výchozího nastavení.



Mikrokontroléry a embedded systémy – cvičení

- Nastavte příslušné piny připojené na SCT registr jako výstupní (levé tlačítko na pinu, zvolit GPIO_Output) a pojmenujte je (pravé tlačítko, Enter User Label, vložit název). Použijte pojmenování SCT_NLA, SCT_SDI, SCT_CLK a SCT_NOE. Vygenerujte kód (stačí uložit CubeMX soubor, příp. Project / Generate Code).
- Do projektu okopírujte drivery sct.c a sct.h z první části cvičení. Tento krok je v STM32CubeIDE poněkud problematický, zřejmě nejsnazší cesta je použít drag&drop se stisknutým Ctrl, soubor sct.c přetáhnout na složku Src, sct.h na Inc nového projektu. Na dotaz ke způsobu importu zvolit Copy files.
- Veškerou inicializaci zajišťuje kód vygenerovaný CubeMX. Ve funkci `sct_init()` proto zůstane pouze volání `sct_led(0)`. Definice `#define` přístupů na piny smažte. Mezi inkudované soubory přidejte main.h.
- Funkci `sct_led()` upravte tak, aby využívala HAL funkce pro přístup ke GPIO se symbolickými názvy definovanými v main.h, např. takto:

```
HAL_GPIO_WritePin(SCT_CLK_GPIO_Port, SCT_CLK_Pin, 1);
```
- V souboru main.c doplňujte vlastní kód vždy mezi komentáře `USER CODE BEGIN` a `USER CODE END`. Pokud toto pravidlo porušíte, bude váš kód smazán při přegenerování projektu z CubeMX.
- Zobrazení „bYE“ vložte do sekce `USER CODE 2`, místo prázdné smyčky budeme využívat funkci `HAL_Delay()`:

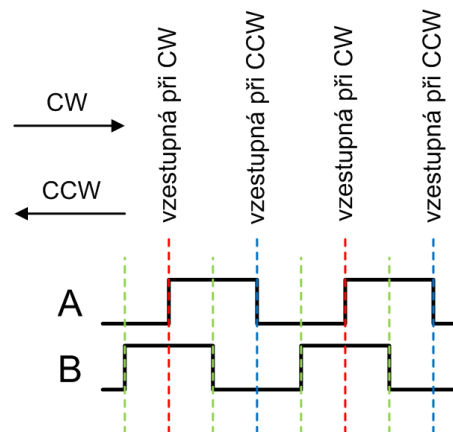
```
sct_init();  
sct_led(0x7A5C36DE);  
HAL_Delay(1000);
```
- Podobně doplňte testování čísel v hlavní smyčce, tj. v sekci `USER CODE 3`.
- Proveďte commit pracovní kopie do Gitu.

2.5 Rotační enkodér

- O čítání pulsů z rotačního enkodéru se bude starat hardware. V CubeMX vyberte časovač TIM1 a přepněte Combined Channels na Encoder Mode. Tím se aktivují piny PA8 a PA9, na které je enkodér připojený.
- Aby enkodér čítal správným směrem, je potřeba invertovat jeden ze signálů – pro kanál 1 nastavte Polarity na Falling Edge. Protože chceme čítat do hodnoty 150, nastavte toto omezení – Counter Period (AutoReload) na 150.
- Vygenerujte kód.
- Čítač v režimu enkodéru je potřeba spustit. Do sekce `USER CODE 2` přidejte volání:

```
HAL_TIM_Encoder_Start(&htim1, htim1.Channel);
```

- V hlavní smyčce zakomentujte testovací zobrazení a místo toho vložte kód pro aktualizaci displeje aktuální hodnotou čítače, kterou lze získat voláním `__HAL_TIM_GET_COUNTER(&htim1)`. Nekonečnou smyčku doplňte krátkým čekáním (cca 50ms).
- Proveďte commit pracovní kopie do Gitu, uložte repozitář pomocí Git Push.





Mikrokontroléry a embedded systémy – cvičení

