Maticová klávesnice a kódový zámek

1 Zadání

- Vytvořte jednoduchou aplikaci, která bude blikat jednou z LED diod na desce. Využijte prosté čekání. Otestujte na vývojové desce, ověřte ladění krokováním. Aktivujte Serial Wire Viewer, přesměrujte standardní výstup (1b).
- Připojte a implementujte maticovou klávesnici. Stisknutá tlačítka vypisuje v ladicím okně (1b).
- Vytvořte jednoduchý kódový zámek, který po zadání sekvence 7932# rozsvítí LED na desce. Doplňte timeout (cca 3s), po kterém se stavový automat kódového zámku vrátí do výchozího stavu (2b).

2 Návod

2.1 Základní seznámení

- Vytvořte si pracovní kopii svého repozitáře z Githubu (Git Clone), příp. aktualizujte repozitář ze serveru (Git Pull).
- Založte nový projekt přes File / New / STM32 Project / Board Selector / **NUCLEO-F429ZI**. Budeme využívat HAL knihovny, proto ponechte Targeted Project Type na STM32Cube. Potvrďte inicializaci všech periferií do výchozího nastavení. Je vhodné deaktivovat nepoužívané periferie, zejména vypnout USB_OTG_FS (Mode: Disable) a ETH, aby projekt pro začátek nebyl zbytečně komplikovaný.
- Pokud máte k dispozici pouze **NUCLEO-F030R8**, postupujte podle poznámek na konci dokumentu.

2.2 Rozblikání LED

- Pro blikání LED použijeme v nekonečné smyčce kód:

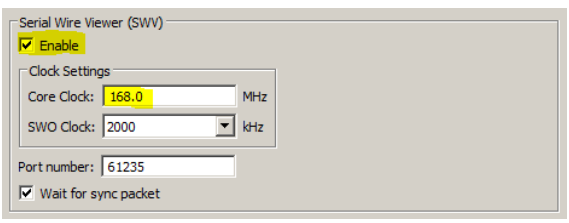
```
HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
HAL_Delay(250);
```

- Kompletní popis HAL funkcí knihovny STM32Cube najdete na:

  https://www.st.com/en/embedded-software/stm32cubef4.html

  https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics

- Serial Wire Viewer umožňuje s využitím Instrumentation Trace Macrocell (ITM) jinou formu přesměrování standardního výstupu, která nevyžaduje připojený UART. Možnosti ladění přes SWV jsou obrovské, základní přehled viz http://blog.atollic.com/learn-serial-wire-viewer-debugging-on-cortex-m.
- SWV je třeba aktivovat na nastavení debuggeru a zvolit správnou frekvenci jádra: Run – Debug Configurations – Debugger – Serial Wire Viewer.



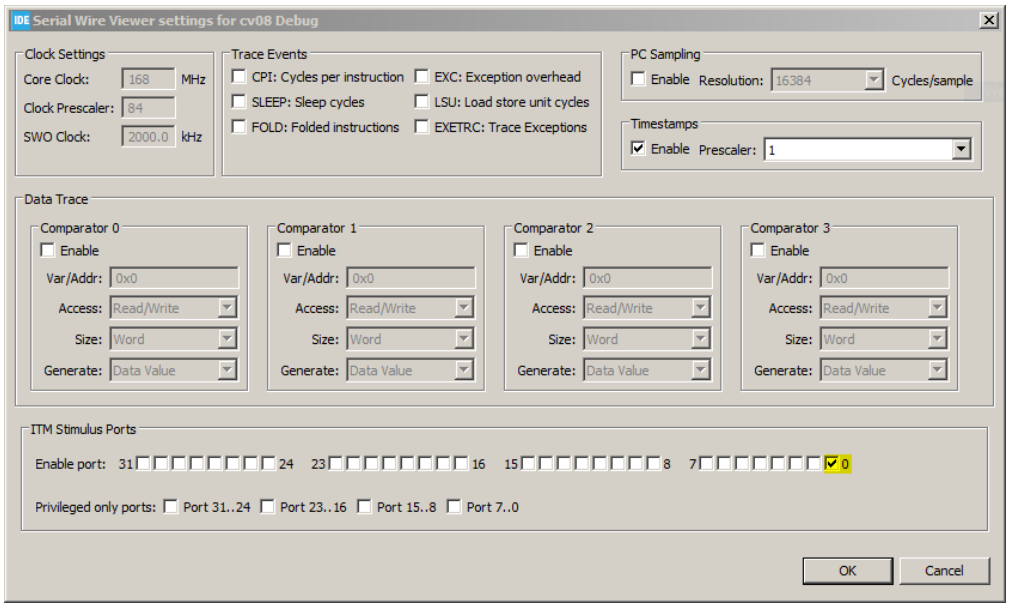
- Standardní výstup je třeba přesměrovat, lze definovat vlastní funkci _write() nebo použít dodanou (syscalls.c) a definovat funkci __io_putchar(), která bude volat ITM_SendChar():

```
int __io_putchar(int ch)
{
   ITM_SendChar(ch);
   return 0;
}
```

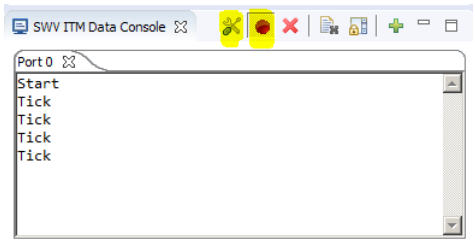
- Mezi inkludované soubory v sekci USER CODE Includes přidáme standardní knihovnu:

```
#include <stdio.h>
```

- Po spuštění ladění je třeba zobrazit okno konzole Window – Show View – SWV – SWV ITM Data Console a v konfiguraci povolit reakci ITM Stimulus Port na port 0.



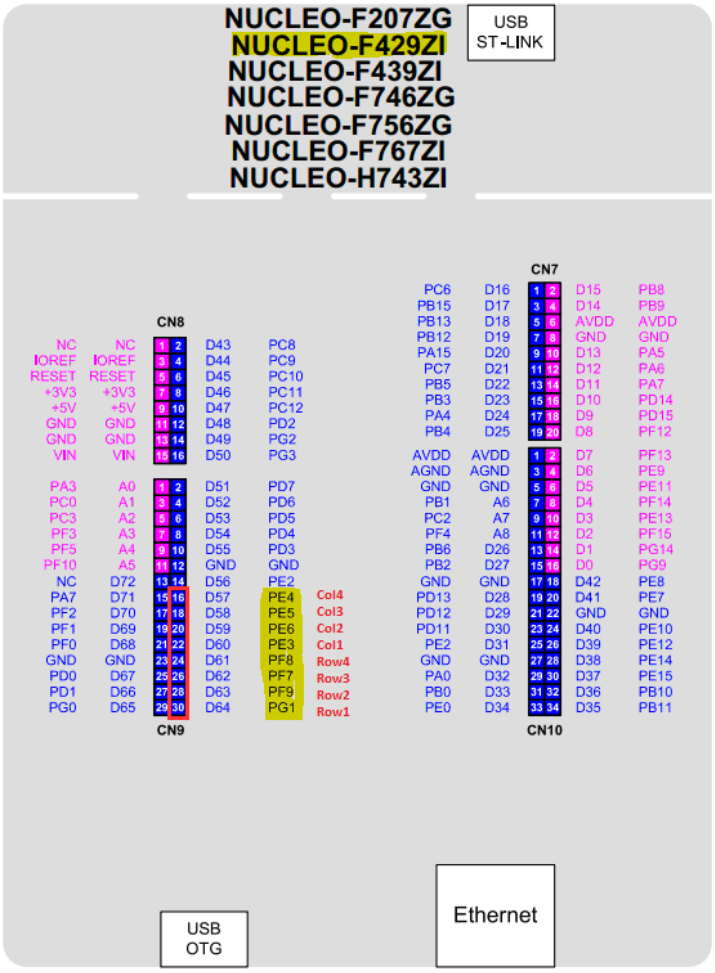
- Nyní již můžeme kdekoliv v kódu používat printf(). Veškerý výstup je třeba zakončovat znakem \n, který kromě odřádkování provádí vyprázdnění bufferu, tj. okamžité odeslání obsahu ke zobrazení.
- V konzoli se po se po spuštění (Start Trace) vypisují texty odeslané přes printf().



2.3 Maticová klávesnice

- Klávesnice je zapojena do matice 4x4, při pohledu na klávesnici jsou vývody zapojeny v pořadí řádek 1 až 4, sloupec 1 až 4. Skenování klávesnice budeme řešit postupným uzemňováním jednotlivých řádků (přes otevřený kolektor) a následným čtením stavu sloupců (s aktivovaným pullupem).

- Jedním z vhodných míst pro připojení klávesnice na vývojový kit jsou piny PG1 až PE4.



- V STM32CubeMX nakonfigurujeme Row1-4 (PG1, PF9, PF7, PF8) jako GPIO_Output, dále Col1-4 (PE3, PE6, PE5, PE4) jako GPIO_Input. Všechny signály správně pojmenujeme. V záložce Configuration je třeba výstupní piny přepnout na Output Open Drain, vstupním aktivovat Pull-up.
- Dále aktivujeme časovač TIM3, který bude sloužit ke skenování – zvolíme Clock Source Internal Clock. Časovač je řízen taktem 84MHz (připojení na sběrnici APB1), skenování bude probíhat s periodou 10ms na sloupec, takže vhodná předdělička je 8400-1=8399, AutoReload nastavit 100-1=99 a povolit. V záložce NVIC povolíme přerušení časovače.
- Vytvoříme obsluhu pro skenování. Do main() sekce USER CODE 2 přidáme povolení přerušení časové základny:

```
HAL_TIM_Base_Start_IT(&htim3);
```

- Dále vytvoříme samotnou obsluhu časové základny v sekci USER CODE 4. Kostra kódu skenování maticové klávesnice by mohla vypadat např. takto:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  static int row = 0;
  static const int keyboard[4][4] = {
    {  1,  2,  3, 21 },
    {  4,  5,  6, 22 },
    {  7,  8,  9, 23 },
    { 11,  0, 12, 24 },
  };

  if (key == -1) {
    if (HAL_GPIO_ReadPin(Col1_GPIO_Port, Col1_Pin) == GPIO_PIN_RESET) key = keyboard[row][0];
    if (HAL_GPIO_ReadPin(Col2_GPIO_Port, Col2_Pin) == GPIO_PIN_RESET) key = keyboard[row][1];
    // ... atd ...
  }

  HAL_GPIO_WritePin(Row1_GPIO_Port, Row1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(Row2_GPIO_Port, Row2_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(Row3_GPIO_Port, Row3_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(Row4_GPIO_Port, Row4_Pin, GPIO_PIN_SET);
  switch (row) {
    case 0: row = 1; HAL_GPIO_WritePin(Row2_GPIO_Port, Row2_Pin, GPIO_PIN_RESET); break;
    case 1: row = 2; HAL_GPIO_WritePin(Row3_GPIO_Port, Row3_Pin, GPIO_PIN_RESET); break;
    // ... atd ...
  }
}
```

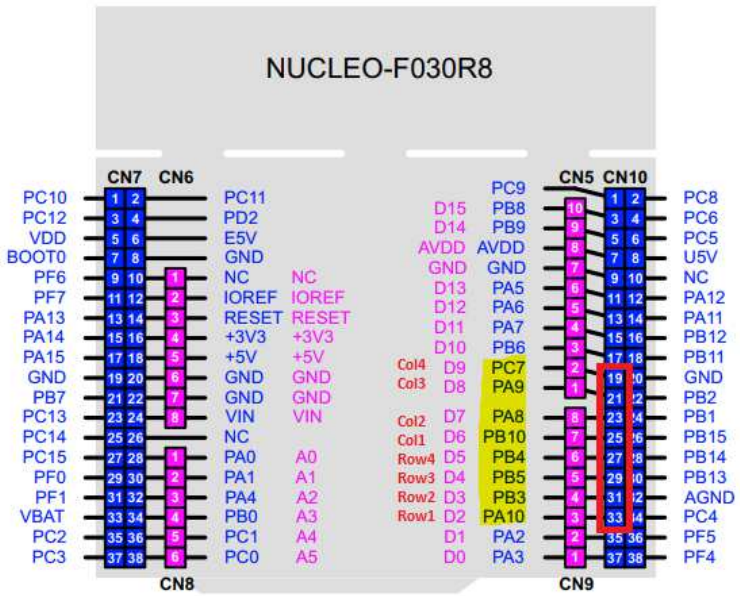
- Obsluha je „obrácená“, tj. nejdříve se testují sloupce pro řádek aktivovaný v předchozím kroku a následně se vybere řádek pro příští krok.
- Do statické volatilní proměnné key (definované v USER CODE PV – private variables) je zapsán scan-kód stisknuté klávesy 0 až 9, příp. 11 pro *, 12 pro #, 21 až 24 pro A až D. Klávesa je obsloužena v hlavním kódu a do proměnné key je zapsána hodnota -1, která značí, že může být přečtena další klávesa.
- Pro otestování funkce je výborně použitelný výpis printf() přes SWV, popsaný výše.
- V případě přiřazené nové hodnoty do proměnné key tedy je vhodné si zároveň poslat zprávu do okna ladění, např. ve tvaru „stisknuto: 5“.

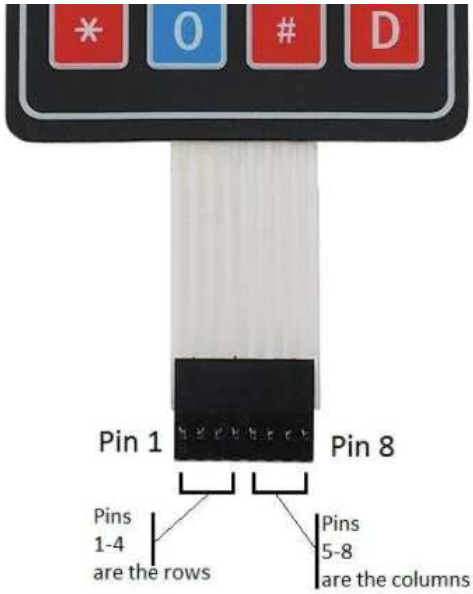
2.4 Kódový zámek

- Do kódu uložte jako pole konstant požadovaný kód, tj. { 7, 9, 3, 2, 12 }.
- Udržujte si informaci o správně zadané pozici, tj. výchozí hodnota bude 0, při postupném zadávání kódu se bude kontrolovat oproti požadovanému a při kompletní sekvenci se rozsvítí LED na desce.
- Pro pomoc s laděním je opět vhodné využít printf() výpisů do debuggeru, např. informaci o poslední správně zadané pozici.
- Timeout je nejjednodušší implementovat pomocí funkce HAL_GetTick(), která vrací počet milisekund od zapnutí desky. Pomocí kontroly rozdílu mezi HAL_GetTick() a uloženou hodnotou lze snadno timeout určit a obsloužit v rámci hlavní smyčky stavového automatu. Při timeoutu se nastaví poslední správně zadaná pozice na 0.
- Na závěr proveďte commit pracovní kopie do Gitu, uložte repozitář pomocí Git Push.

2.5 Modifikace zadání pro NUCLEO-F030R8

- Serial Wire Viewer (SWV) není u jádra Cortex-M0 podporován. Použijte standardní přesměrování výstupu na UART a terminál v PC.
- Vhodným místem pro připojení klávesnice na vývojový kit jsou piny PA10 až PC7.





- Ke skenování je možné opět využít časovač TIM3, vzhledem k rozdílné frekvenci interních hodin je ale třeba příslušně upravit předdělička. Frekvence jádra i čítače je 48MHz, takže vhodná předdělička je 4800-1=4799.