OR domača naloga – STM32H750

Morsejev prevajalnik v C s HAL knjižnico

Tilen Bezgovšek

vpisna številka: 63110302

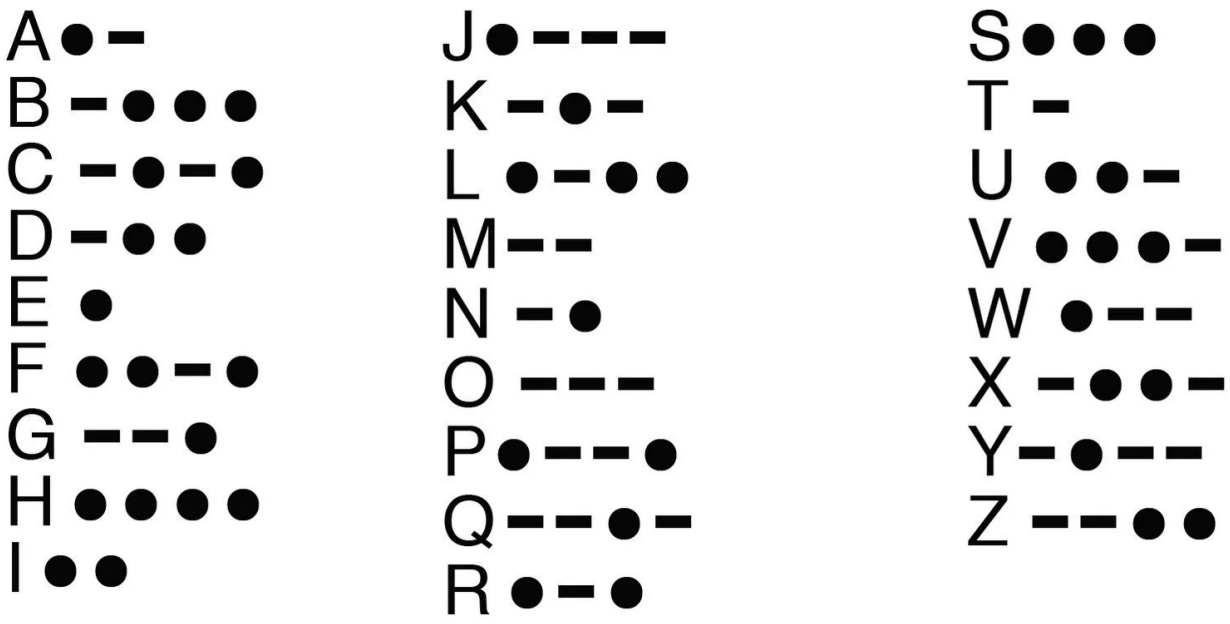
8.2.2024

Uvod in osnovni pregled delovanja

Osnovno idejo za projekt sem dobil iz pripravljene teme za to domačo nalogo; Morsejev oddajnik. Namesto oddajnika sem za projekt izbral Morsejev prevajalnik.

Prevajalnik deluje preko modrega gumba na plošči STM32H750B-DK. Krajši pritisk gumba predstavlja simbol ‘.’, daljši pritisk gumba pa simbol ‘-’. Če se zaporedje teh dveh simbolov ujema z zaporedjem, ki predstavlja eno izmed črk angleške abecede, se ta črka prenese po serijski UART povezavi. Če se zaznano zaporedje simbolov ne ujema z nobeno črko, se ne prenese nič. Če eno sekundo ni pritiska gumba, se po UART povezavi prenese en presledek.

Osnovna legenda za Morsejevo abecedo:



Kot osnovo za projekt sem uporabil pripravljen začetni projekt na GitHub: https://github.com/LAPSyLAB/ORLab-STM32H7/tree/main/STM32H750B-DK_C_Basic

Za delo sem uporabljal razvojno okolje STM32CubeIDE 1.13.1.

Pridobitev dodatnih informacij

Za delo z modrim gumbom na plošči in s knjižnico HAL sem potreboval nekaj dodatnih informacij, ki sem jih dobil iz dokumentov za ploščo.

Prvi uporaben dokument je bil UM2488 User manual.

life.augmented

UM2488

User manual

Discovery kits with STM32H745XI and STM32H750XB MCUs

Podpoglavje 6.16 nam pove, da je alternativna funkcija za moder gumb “wakeup”.

6.16 Buttons and LEDs

The black button (B2), located on the board top side, is the reset of the STM32H745XIH6 and STM32H750XBH6 microcontrollers. Refer to Figure 4. STM32H745I-DISCO and STM32H750B-DK Discovery board top layout. The blue button (B1), located on the top side, is available for use as a digital input or as a wakeup alternate function.

Table 7 summarizes the assignment of the control ports to the LED indicators.

Table 7. Assignment of control ports to LEDs

Reference	Color	Name	Comment
B1	Blue	USER	Wakeup alternate function
B2	Black	RESET	-

V poglavju 8 je dodelitvena tabela IO pin-ov. Tukaj vidimo, da ima pin PC13 oznako “WAKEUP”. Torej pin PC13 ustreza modremu gumbu na plošči.

Pin name	Pin number	Signal or label	Comment
PC6	F14	SDIO1_D6	-
PC7	F13	SDIO1_D7	-
PC8	E13	SDIO1_D0	-
PC9	E14	SDIO1_D1	-
PC10	A13	SDIO1_D2	-
PC11	B13	SDIO1_D3	-
PC12	C12	SDIO1_CK	-
PC13	E3	WAKEUP	-
PC14	C2	OSC32_IN	-
PC15	C1	OSC32_OUT	-

Naslednji uporaben dokument je bil UM2217 User manual, ki opisuje HAL knjižnico. Sicer je v začetnem projektu veliko dela s HAL že narejenega, ampak potreboval sem nekaj dodatnih funkcij.

life.augmented

UM2217

User manual

Description of STM32H7 HAL and low-layer drivers

Najprej sem za branje gumba pogledal podpoglavje 35.2.4, kjer je opisana funkcija HAL_GPIO_ReadPin.

HAL_GPIO_ReadPin

Function name

GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)

Function description

Reads the specified input port pin.

Parameters

- **GPIOx:** where x can be (A..K) to select the GPIO peripheral.
- **GPIO_Pin:** specifies the port bit to read. This parameter can be GPIO_PIN_x where x can be (0..15).

Return values

- **The:** input port pin value.

Zadnja informacija, ki sem jo pridobil iz tega dokumenta je o funkciji HAL_GetTick iz poglavja 7. Ta funkcija vrne sistemski čas v milisekundah. Ta čas začne teči, ko se izvede funkcija HAL_InitTick. To funkcijo kliče funkcija HAL_Init(), vsakič ko se sistem resetira ali funkcija HAL_RCC_ClockConfig(), če je ura kdaj ponastavljena.

HAL_GetTick

Function name

uint32_t HAL_GetTick (void)

Function description

Provides a tick value in millisecond.

Return values

- **tick:** value

Notes

- This function is declared as __weak to be overwritten in case of other implementations in user file.

Podroben opis programske kode

Ker uporabljam nekaj dodatnih funkcij za delo z nizi, sem na začetku moral dodati ‘include’ za knjižnico string.h.

Nato sem pred glavno zanko v programu dodal dve funkciji za delovanje časovnika.

```
int startTimer() {
    return HAL_GetTick();
}
int stopTimer(int startTime) {
    int endTime = HAL_GetTick();
    int diff = endTime - startTime;
    return diff;
}
```

Funkcija startTimer() samo vrne trenutni sistemski čas v milisekundah, funkcija stopTimer() pa za podan čas vrne razliko med tem časom in trenutnim časom.

Dodal sem abecedo v obliki Morsejevih simbolov in simbola za kratek ter dolg pritisk.

```
char alphabet[26][5] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.",
                        "....", "..", ".---", "-.-", ".-..", "--", "-.",
                        "---", ".--.", "--.-", ".-.", "...", "-", "..-",
                        "...-", ".--", "-..-","-.--", "--.."};
char dot = '.';
char dash = '-';
```

Zdaj je vse pripravljeno za glavno zanko.

Glavna zanka programa vsebuje na začetku definicijo in inicializacijo treh enostavnih ‘int’ spremenljivk za merjenje časa in niz za shranjevanje zaznanih simbolov (pritiskov gumba). Na začetku je niz prazen. V C-ju je niz definiran enostavno kot tabela znakov in se vedno konča z znakom ‘\0’. Ker so vsa zaporedja simbolov, ki predstavljajo črke v angleški abecedi največ dolžine 4, potrebujemo za shranjevanje niza simbolov samo 4 znake. Ker moramo imeti še prostor za znak ‘\0’, je tabela znakov dolžine 5.

```
int timer1 = startTimer();
int timer2;
int timer2elapsed;
char symbol[5] = "";
```

Zdaj je na vrsti druga, ugnezdena ‘while’ zanka. V tej zanki se pritiski gumba zaznajo in shranijo v spremenljivko “symbol”. Po izstopu iz zanke se niz interpretira in pošlje po UART povezavi.

Izstopni pogoj za to zanko je količina pretečenega časa. Ko preteče ena sekunda brez pritiska gumba, izstopimo iz zanke.

Uporabljata se dva časovnika, eden za merjenje časa v tej zanki brez aktivacije gumba (timer1), drugi pa za merjenje, kako dolgo je gumb aktiviran (timer2). Meja med kratkim in dolgim pritiskom je 200 milisekund.

Za branje gumba se uporablja funkcija HAL_GPIO_ReadPin:

```
while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 1)
```

Za dodajanje znaka na konec niza se uporablja funkcija strncat:

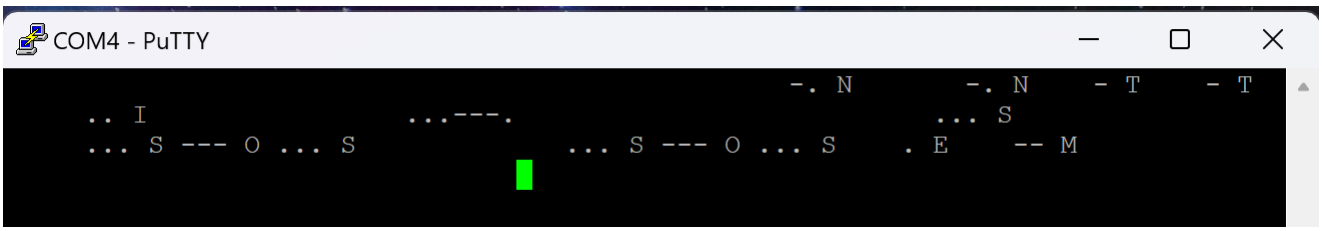
```
strncat(symbol, &dash, 1);
```

Za interpretacijo črk se po izstopu iz ugnezdene zanke uporablja funkcija strcmp v zanki, kjer primerjamo niz z vsakim zaporedjem abecede. Če se ujema, shranimo indeks v tabeli abecede. Tako lahko enostavno to število prištejemo k 65 in dobimo pravo črko. Za samo pošiljanje po UART povezavi sem uporabil že prisotno kodo v pripravljenem projektu.

```
snprintf(SendBuffer,BUFSIZE,"%c", 65 + j);
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);
```

Prejemanje znakov po UART

Na računalniku, ki je povezan preko USB s ploščo sem uporabil program PuTTY za branje znakov preko UART povezave.



V terminalu lahko vidimo črke in zaporedja kratkih oziroma dolgih pritiskov gumba.

Priloga: celotna dodana koda

```
int startTimer() {
     return HAL_GetTick();
 }
 int stopTimer(int startTime) {
     int endTime = HAL_GetTick();
     int diff = endTime - startTime;
     return diff;
 }
 char alphabet[26][5] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.",
                         "....", "..", ".---", "-.-", ".-..", "--", "-.",
                         "---", ".--.", "--.-", ".-.", "...", "-", "..-",
                         "...-", ".--", "-..-","-.--", "--.."};
 char dot = '.';
 char dash = '-';
 /* USER CODE END 2 */
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
       // detect button and time duration
       // add '.' or '-' or nothing based on button duration
       int timer1 = startTimer();
       int timer2;
       int timer2elapsed;
       char symbol[5] = "";
       while (stopTimer(timer1) < 1000) {
             timer2 = 0;
             while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 1) {
                   timer1 = startTimer();
                   if (timer2 == 0) {
```

```
                timer2 = startTimer();
            }
        }
        if (timer2 == 0) {
            timer2elapsed = 0;
        } else {
            timer2elapsed = stopTimer(timer2);
        }
        if (timer2elapsed > 200) {
            strncat(symbol, &dash, 1);
        } else if (timer2elapsed > 0) {
            strncat(symbol, &dot, 1);
        }
}
snprintf(SendBuffer,BUFSIZE," %s ", symbol);
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);

// decode symbol and send
int j = -1;
for (int i = 0; i < 26; i++) {
        if (strcmp(symbol, alphabet[i]) == 0) {
            j = i;
        }
}

if (strcmp(symbol, "") == 0) {
        snprintf(SendBuffer,BUFSIZE," ");
        HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);
} else if (j != -1) {
        snprintf(SendBuffer,BUFSIZE,"%c", 65 + j);
        HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);
}
```