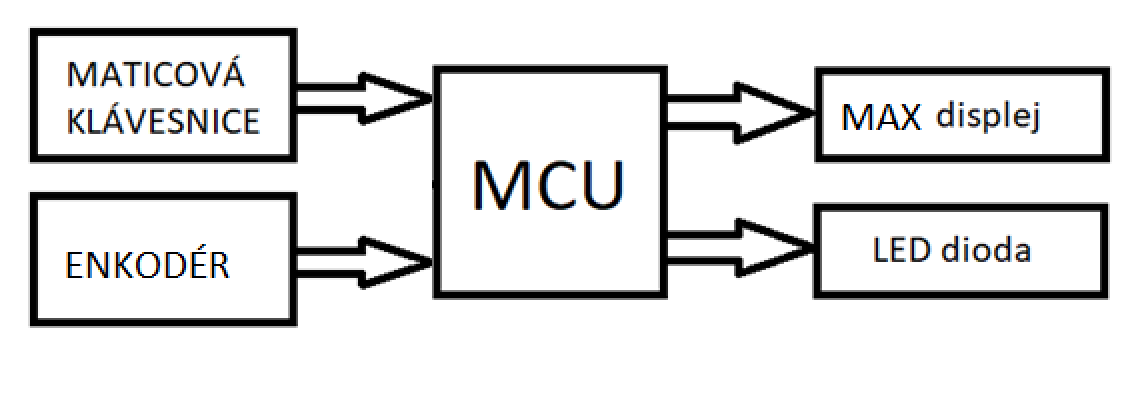
**BLOKOVÉ SCHÉMA:**



**PROGRAM:**

|  |
| --- |
| #include "stm8s.h" |
|  | #include "milis.h" |
|  | #include "stm8\_hd44780.h" |
|  | #include "stdio.h" |
|  | #include "keypad.h" |
|  | #include "swspi.h" |
|  |  |
|  | void process\_keypad(void); |
|  | void max\_init(void); |
|  | void max\_posli(void); |
|  |  |
|  |  |
|  | uint16\_t lasttime=0; |
|  |  |
|  | void main(void){ |
|  | CLK\_HSIPrescalerConfig(CLK\_PRESCALER\_HSIDIV1); // 16MHz z interního RC oscilátoru |
|  | GPIO\_Init(GPIOB,GPIO\_PIN\_7,GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); |
|  | GPIO\_Init(GPIOG,GPIO\_PIN\_3,GPIO\_MODE\_IN\_PU\_NO\_IT); |
|  | init\_milis(); |
|  | keypad\_init(); |
|  | max\_init(); |
|  | max\_posli(); |
|  |  |
|  | while (1){ |
|  | process\_keypad(); // obsluhujeme klávesnici |
|  | if(GPIO\_ReadInputPin(GPIOG,GPIO\_PIN\_3)==RESET){ |
|  | GPIO\_WriteHigh(GPIOB,GPIO\_PIN\_7); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // pravidelně sleduje stav klávesnice a reaguje na stisknutí kláves |
|  | void process\_keypad(void){ |
|  | static uint8\_t minule\_stisknuto=0; // ukládáme poslední stav klávesnice |
|  | static uint16\_t last\_time=0; |
|  | uint16\_t stisknuto; |
|  |  |
|  | if(milis()-last\_time > 20){ // každých 20ms ... |
|  | last\_time = milis(); |
|  | stisknuto=keypad\_scan(); // ... skenujeme klávesnici |
|  | if(minule\_stisknuto == 0 && stisknuto != 0){ // pokud byla minule uvolněná a teď je něco stisknuto |
|  | minule\_stisknuto = stisknuto; |
|  | // zachytili jsme stisk klávesy - můžeme na to zareagovat |
|  |  |
|  | if(stisknuto < 10){ |
|  | max7219\_posli(DIGIT0, stisknuto); |
|  | } |
|  | } |
|  | else if(stisknuto == 15){ |
|  | GPIO\_WriteHigh(GPIOB,GPIO\_PIN\_7); |
|  | } |
|  | else if(stisknuto == 16){ |
|  | GPIO\_WriteLow(GPIOB,GPIO\_PIN\_7); |
|  | } |
|  | } |
|  | if(stisknuto == 0){minule\_stisknuto=0;} |
|  | } |
|  | void max\_init(void){ |
|  | GPIO\_Init(CS\_GPIO,CS\_PIN,GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); |
|  | GPIO\_Init(CLK\_GPIO,CLK\_PIN,GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); |
|  | GPIO\_Init(DATA\_GPIO,DATA\_PIN,GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); |
|  | // nastavíme základní parametry budiče |
|  | max7219\_posli(DECODE\_MODE, DECODE\_ALL); // zapnout znakovou sadu na všech cifrách |
|  | max7219\_posli(SCAN\_LIMIT, 7); // velikost displeje 8 cifer (počítáno od nuly, proto je argument číslo 7) |
|  | max7219\_posli(INTENSITY, 5); // volíme ze začátku nízký jas (vysoký jas může mít velkou spotřebu - až 0.25A !) |
|  | max7219\_posli(DISPLAY\_TEST, DISPLAY\_TEST\_OFF); // |
|  | max7219\_posli(SHUTDOWN, DISPLAY\_ON); // zapneme displej |
|  | } |
|  |  |
|  | void max\_posli(void){ |
|  | max7219\_posli(DIGIT0,8); |
|  | max7219\_posli(DIGIT1,7); |
|  | max7219\_posli(DIGIT2,6); |
|  | max7219\_posli(DIGIT3,5); |
|  | max7219\_posli(DIGIT4,4); |
|  | max7219\_posli(DIGIT5,3); |
|  | max7219\_posli(DIGIT6,2); |
|  | max7219\_posli(DIGIT7,1); |
|  | } |
|  |  |
|  | // pod tímto komentářem nic neměňte |
|  | #ifdef USE\_FULL\_ASSERT |
|  |  |
|  | /\*\* |
|  | \* @brief Reports the name of the source file and the source line number |
|  | \* where the assert\_param error has occurred. |
|  | \* @param file: pointer to the source file name |
|  | \* @param line: assert\_param error line source number |
|  | \* @retval : None |
|  | \*/ |
|  | void assert\_failed(u8\* file, u32 line) |
|  | { |
|  | /\* User can add his own implementation to report the file name and line number, |
|  | ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/ |
|  |  |
|  | /\* Infinite loop \*/ |
|  | while (1) |
|  | { |
|  | } |
|  | } |
|  | #endif |

}

**ZÁVĚR:**

Programoval jsem mikrokontrolér STMS208RB v prostředí STVD. Aplikace je absolutně zbytečná a nemá žádné využití ve skutečném životě. Funkčnost je pochybná a je velice náchylná na špatné povětrnostní podmínky, vlhkost a okolní teplotu.