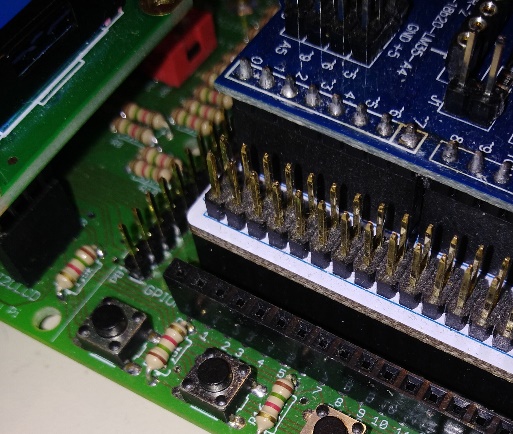
Multifunctionshield mit SPI



SCLK

MOSI

Anschluss:

Mikrocontroller

PB\_15

PB\_14

PB\_13

PB\_5

MOSI

SCLK

SS

Multifunctionshield

D8

D7

D4

PB\_3

D3 Summer

PA\_0

A0 Analog In

Brücken

Telegrammaufbau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Dp | g | f | e | d | c | b | a | x | x | x | x | D0 | D1 | D2 | D3 |

Zeitablaufdiagramm: SS=0; anzeige.write(tausender(eingabe)); SS=1;

dp g f e d c b a x x x x D0 D1 D2 D3

MOSI

SS

SCLK

Anzeige:

a

b

c

d

e

f

g

dp

a

b

c

d

e

f

g

dp

a

b

c

d

e

f

g

dp

a

b

c

d

e

f

g

dp

D3 D2 D1 D0

Beschreibung:

Die Baugruppe verfügt unter anderem über:

* 4-stellige Siebensegmentanzeige

Achtung: Da sowohl das Poti auf dem Multifunctionshield als auch das Poti auf dem Baseshield mit PA\_0 verbunden sind muss eines von beiden Potis in Mittelstellung sein!!!!!

Funktion: Der analoge Einstellwert soll 4-stellig angezeigt werden.

Globale Deklarationen:

* Siebensegmenttabelle seg7[10]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0xC0 | 0xF9 | 0xA4 | 0xB0 | 0x99 | 0x92 | 0x82 | 0xF8 | 0x80 | 0x90 |

* SS als Digitalausgang an PB\_5 (D4)
* anzeige als SPI an PB\_15, PB\_14, PB\_13
* summer als Digitalausgang an PB\_3 (D3)
* ain als Analogeingang an PA\_0 (A0)
* eingabe als int

Hauptprogramm:

Initialisierungen:

* summer=1
* SPI: anzeige.format(*Bitzahl,0bXY*);
  + X: Polarität: \_\_\_\_\_\_\_\_\_
  + Y: Abtasten: \_\_\_ Flanke
  + Bitzahl: \_\_\_\_ Datenbits

Endlosschleife:

* eingabe= (int)(ain\*4096); //eingabe = 0 .. 4095
* Tausenderstelle ausgeben
* 5ms warten
* Hunderterstelle ausgeben
* 5ms warten
* Zehnerstelle ausgeben
* 5ms warten
* Einerstelle ausgeben
* 5ms warten

Hinweise:

int hunderter(int e)  
{  
 return (seg7[(e%1000)/100]<<8)+0b00000010;  
}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Dp | g | f | e | d | c | b | a | x | x | x | x | D0 | D1 | D2 | D3 |
| seg7[(e%1000)/100] | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

0b00000010 aktiviert die hunderterstelle  
(e%1000)/100 berechnet den Wert der Hunderterstelle  
seg7[(e%1000)/100] holt den Siebensegmentcode aus dem Array  
(seg7[(e%1000)/100]<<8) verschiebt den Siebensegmentcode um 8 Binärstellen nach links, so dass der Siebensegmentcode in den Bits 15 .. 8 zu liegen kommt.

Ebenso für Einer, Zehner und Tausender

Aufgabe: Schreiben Sie das Programm und testen Sie.