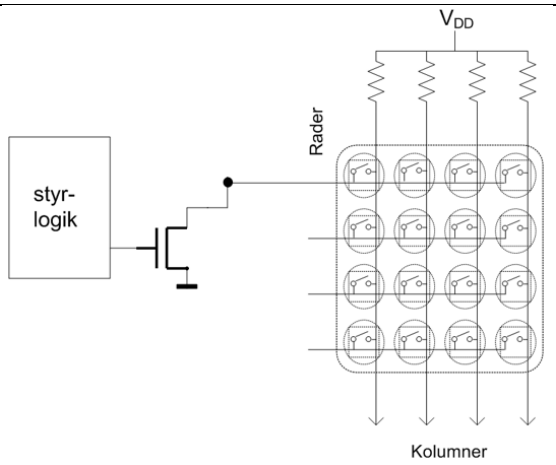
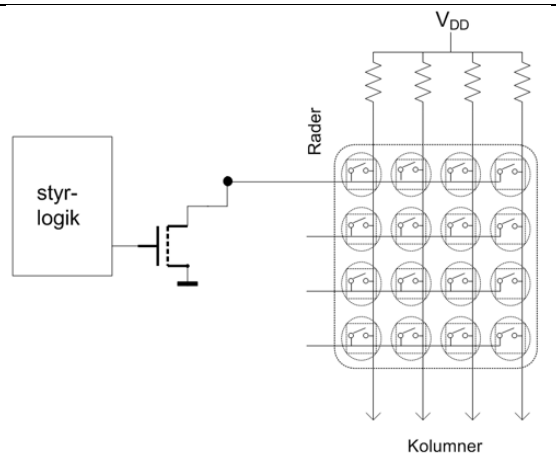
Syften:

- Visa exempel GPIO in/ut som förberedelse på lab2 (keypad).
- Visa implementeringsdetaljer för programmering av ASCII-display.

Kort introduktion:

Vi vill göra en ”kortslutningssäker” lösning för koincidensavsökning av tangentbord:
Utgångarna ska därför vara OPEN-DRAIN

En rad aktiveras med ’0’, styrlogiken kommer då att jorda raden.	I annat fall kommer ”obestämd” nivå att ligga på raden.
	

Då en rad aktiverats och brytaren är öppen, kommer följdaktligen en hög nivå att läsas från kolumnerna.
Då en rad aktiverats och brytaren är sluten, kommer kolumnen att kopplas till jord, och därför läses en låg nivå för kolumnen. Om flera brytare är slutna kommer alla dess kolumner att indikeras.

Jämför distanslab 1.4
Skapa en tangentbordsrutin som söker av tangentbordet och placerar kolumnvärde för varje rad i en 16 bitars int och returnerar denna. Antag att port D kopplats till keypad (enligt laborationsuppgiften).

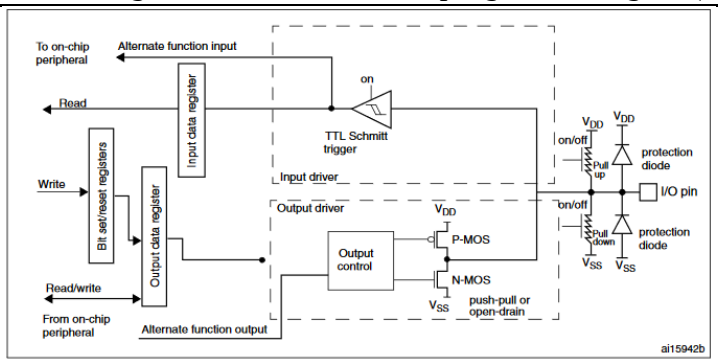
```
Algoritm: keyb_alt_ctrl
Keyb_status = 0;
Aktivera rad 4, placera kolumnvärdena i Keyb_status( b0..b3)
Aktivera rad 3, lägg till kolumnvärdena i Keyb_status( b4..b7)
Aktivera rad 2, lägg till kolumnvärdena i Keyb_status( b8..b11)
Aktivera rad 1, lägg till kolumnvärdena i Keyb_status( b12..b14)
returnera Keyb_status
```

Lösning:

```
/* Port D */
#define      PORT_D_BASE     0x40020C00
#define      GPIO_D_MODER    ((volatile unsigned int *)   (PORT_D_BASE))
#define      GPIO_D_OTYPER   ((volatile unsigned short *)(PORT_D_BASE+0x4))
#define      GPIO_D_OSPEEDR  ((volatile unsigned int *)   (PORT_D_BASE+0x8))
#define      GPIO_D_PUPDR    ((volatile unsigned int *)   (PORT_D_BASE+0xC))
#define      GPIO_D_IDRLOW   ((volatile unsigned char *)(PORT_D_BASE+0x10))
#define      GPIO_D_IDRHIGH  ((volatile unsigned char *)(PORT_D_BASE+0x11))
#define      GPIO_D_ODRLOW   ((volatile unsigned char *)  (PORT_D_BASE+0x14))
#define      GPIO_D_ODRHIGH  ((volatile unsigned char *)  (PORT_D_BASE+0x15))
void init_app( void )
{
   /* PORT D
   b15-b12 used for output to rows
   b11-b8  used for input from columns
   b7-b0  used for output to 7-seg display
   */
   *GPIO_D_MODER   = 0x55000000;
   *GPIO_D_PUPDR   = 0x00550000;      /* Input, pull up */
   *GPIO_D_OTYPER  = 0x00000000;       /* outputs are push/pull */
   *GPIO_D_ODRHIGH = 0;
}

unsigned short keyb_alt_ctrl(void)
{
   int row;
   unsigned short pad;
   *GPIO_D_ODRHIGH = ~0x80;  pad = (unsigned short)    (*GPIO_D_IDRHIGH & 0xF);
   *GPIO_D_ODRHIGH = ~0x40;  pad |= (unsigned short)  ((*GPIO_D_IDRHIGH <<4) & 0x00F0);
   *GPIO_D_ODRHIGH = ~0x20;  pad |= (unsigned short)  ((*GPIO_D_IDRHIGH <<8) & 0x0F00);
   *GPIO_D_ODRHIGH = ~0x10;  pad |= (unsigned short)  ((*GPIO_D_IDRHIGH <<12) & 0xF000);
   *GPIO_D_ODRHIGH =  0xFF;
   return pad^0xFFFF;  /* Returnera inverterat mönster */
}
```

”Routa” en portpinne – Alternativ funktion



Exempel:
Man vill konfigurera det synkrona gränssnittet SPI1 (*Serial Peripheral Interface)* . Gränssnittet använder tre portpinnar, med funktioner: MISO, MOSI och SCK. Visa hur dessa konfigureras tillsammans med portpinnarna PA5, PA6 och PA7.

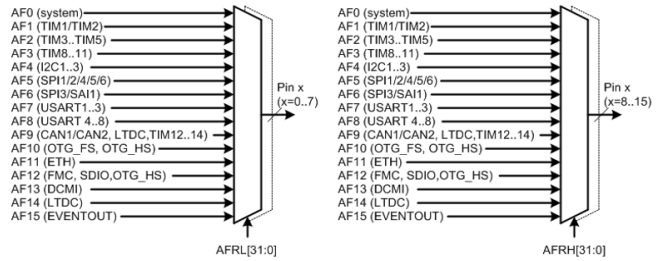
Lösning:

I databladet sid 58, läser vi ut:

PA5 : SPI1_SCK (out)
PA6: SPI1_MISO (in)
PA7: SPI1_MOSI (out)

Pin number						Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP64	WLCSP90	LQFP100	LQFP144	UFBGA176	LQFP176						
21	G8	30	41	P4	51	PA5	I/O	TTa	(4)	SPI1_SCK/ … TIM2_CH1_ETR/ TIM8_CH1N/ EVENTOUT	ADC12_IN5/DAC_OUT2
22	H8	31	42	P3	52	PA6	I/O	FT	(4)	SPI1_MISO / … DCMI_PIXCLK / TIM3_CH1 / TIM1_BKIN/ EVENTOUT	ADC12_IN6
23	J8	32	43	R3	53	PA7	I/O	FT	(4)	SPI1_MOSI/ … ETH_MII_RX_DV / TIM1_CH1N / ETH_RMII_CRS_DV/ EVENTOUT	ADC12_IN7

I referenshandboken sid 270 (bild till höger), ser vi att SPI1 använder AF5 (Alternativ funktion 5)



Av bilden till höger framgår de bitfält som ska initieras:
PA5: AFRL5=0101 (5)
PA6: AFRL6=0101
PA7: AFRL7=0101

8.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..I/J/K)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRLy:** Alternate function selection for port x bit y (y = 0..7)
These bits are written by software to configure alternate function I/Os

AFRLy selection:

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

Vi förutsätter att klockor för Port A och SPI1 är startade.

Vi har: PA5 och PA7 är output, push-pull 50MHz
PA6, input, pullup.

Konfigurering av AF-register:

```
GPIOA->afrl       |=    0x55500000;    /* AF5 SPI1/2 -> PA5, PA6, PA7 */
```

GPIO:

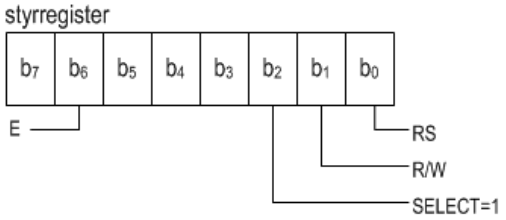
```
GPIOA->moder      |=    0x00008800;    /* PA5 and PA7 output */
GPIOA->otyper     &=    ~0x00A0;       /* PA5 and PA7 push pull */
GPIOA->ospeedr    |=    0x00008800;    /* PA5 and PA7, 50 MHz */
GPIOA->pupdr      |=    0x0001000;     /* PA6 pull-up */
```

Synkronisering

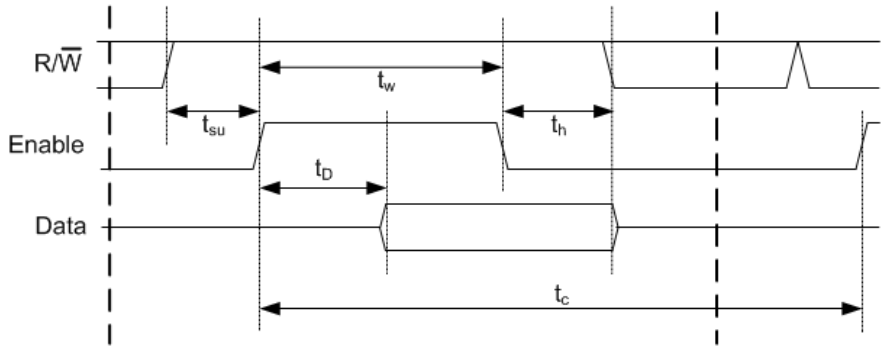
Koncentrera på ’lågnivå’, Koppla kod till tidsdiagram. Resonera kring mycket korta fördröjningar (nano-sekunder). En maskininstruktion tar c:a 6 nanosekunder.

Förmodligen enklast att göra någom PPT att prata om här.
Repetera syftet med E-signal, att synkronisera den yttre enheten.

Bitar i styrregistret:



Tidsdiagram:




```
unsigned char ascii_read_controller( void )
{
   unsigned char c;
   ascii_ctrl_bit_set( B_E );
   delay_250ns();  /* max 360 ns */
   delay_250ns();
   c = *GPIO_E_IDRHIGH;
   ascii_ctrl_bit_clear( B_E );
   return c;
}
unsigned char ascii_read_status()
{
   unsigned char c;
   *GPIO_E_MODER   = 0x00005555;   /* b15-8 are inputs, 7-0 are outputs */
   ascii_ctrl_bit_set( B_RW );
   ascii_ctrl_bit_clear( B_RS );
   c = ascii_read_controller(  );
   *GPIO_E_MODER      = 0x55555555;   /* all bits outputs */
   return c;
}
void ascii_wait_ready(void)
{
   while( ( ascii_read_status() & 0x80)== 0x80 );
   delay_micro( 8 );      /* se not sid 16, behövs antagligen inte... */
}
```