GPIO - General Purpose Input Output

Content:

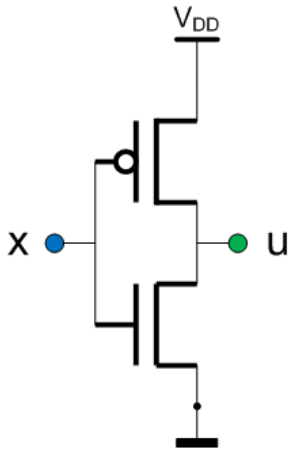
- Digital IO
- Ideala och verkliga signaler
- Bitvis in- och utmatning
- Anslutning - fysiskt gränssnitt
- F407 - GPIO-modul – tillämpningar
- Programmering av enkelt tangentbord

Läsanvisningar:

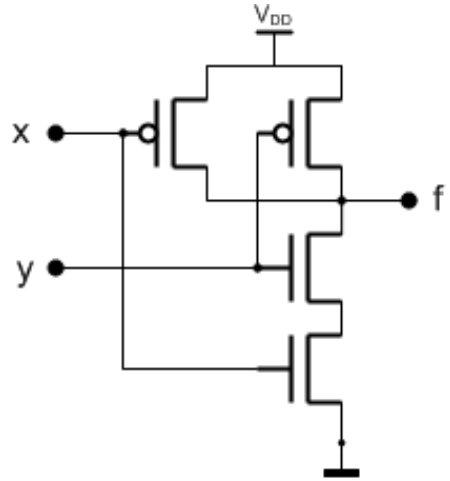
- Arbetsbok kapitel 4

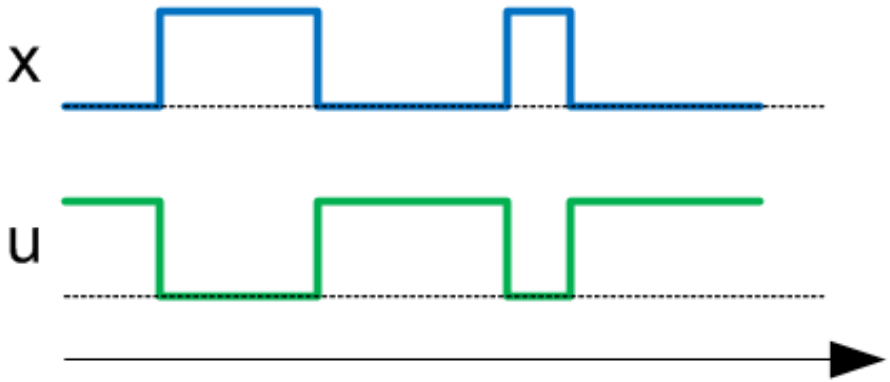
Ideal gates – ideal signals!

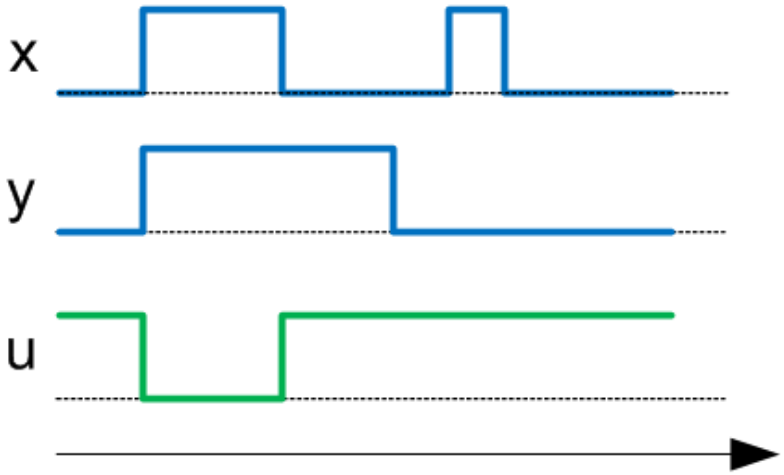
NOT



NAND

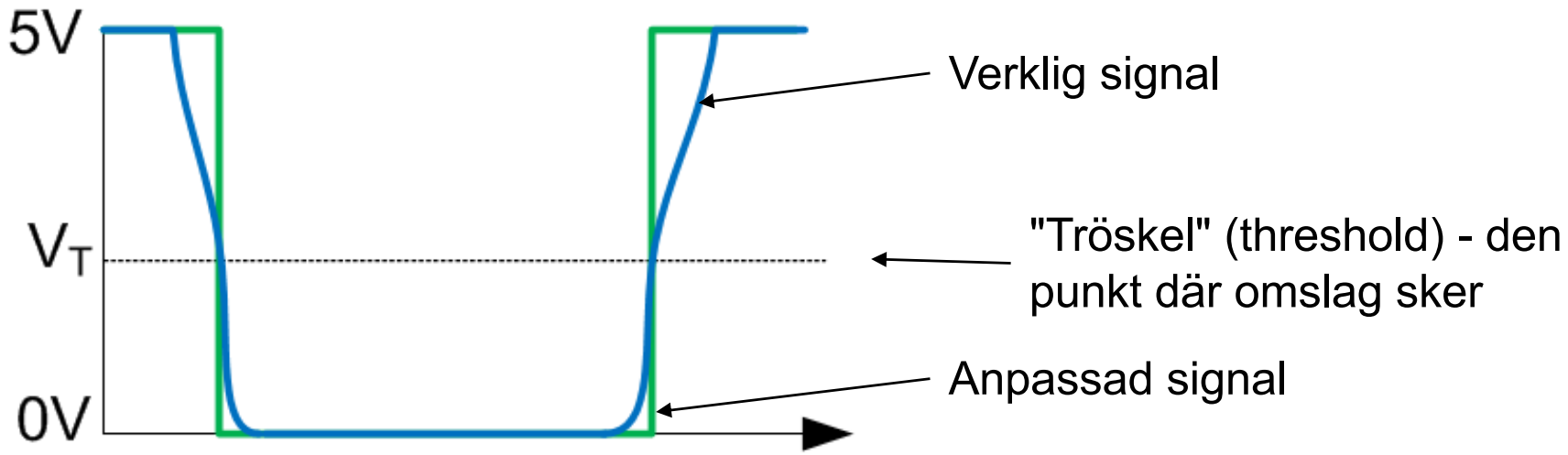






Real signals- logic levels and customization

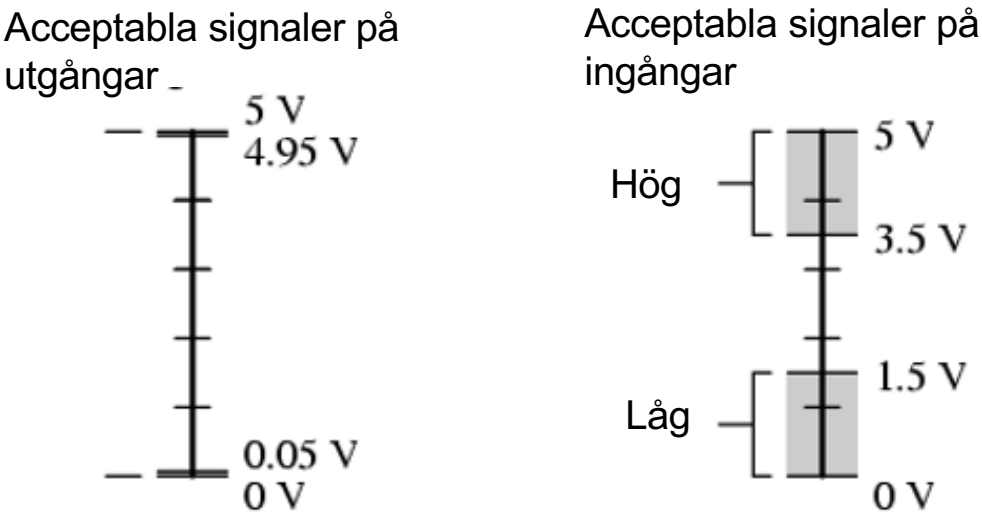
Verkliga signaler måste anpassas till distinkta CMOS-nivåer representerande logikvärdena '0' och '1'



Ideal and real gates

Hos ideala kretsar är tröskelnivån den samma.
Hos verkliga kretsar kan det vara spridning.
Tillverkarna specificerar "säkra" intervall för hög respektive låg nivå.

För CMOS-kretsar med matningsspänning 5 Volt gäller:



Skillnaden kallas ***störmarginal***:

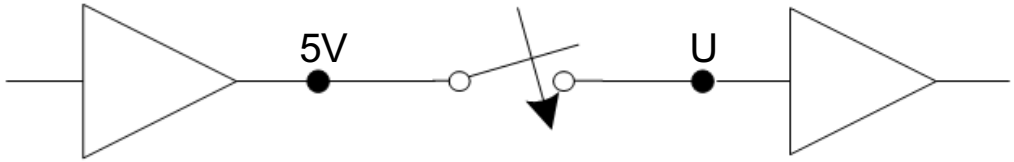
Hög nivå 4,95-3,5 = 1,4 Volt

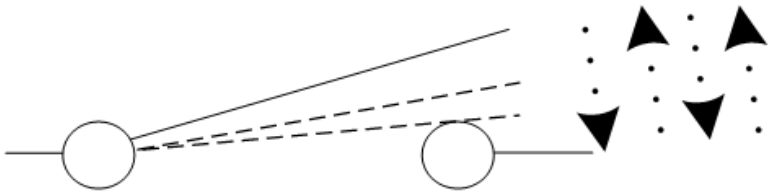
Låg nivå: 1,5-0,05 = 1,45 Volt

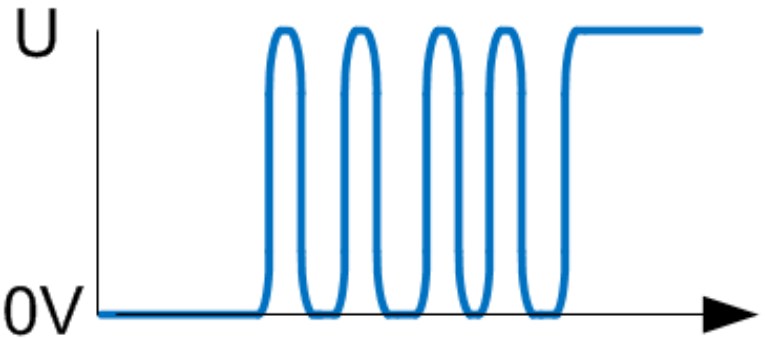
En signal mellan de angivna intervallen är ***obestämd***.

Bounce

Då en mekanisk omkopplare sluts kan den studsa flera gånger mot det slutande blecket innan den stabiliseras i slutet läge. Detta kan generera ett pulståg och kallas för "kontaktstudsar".



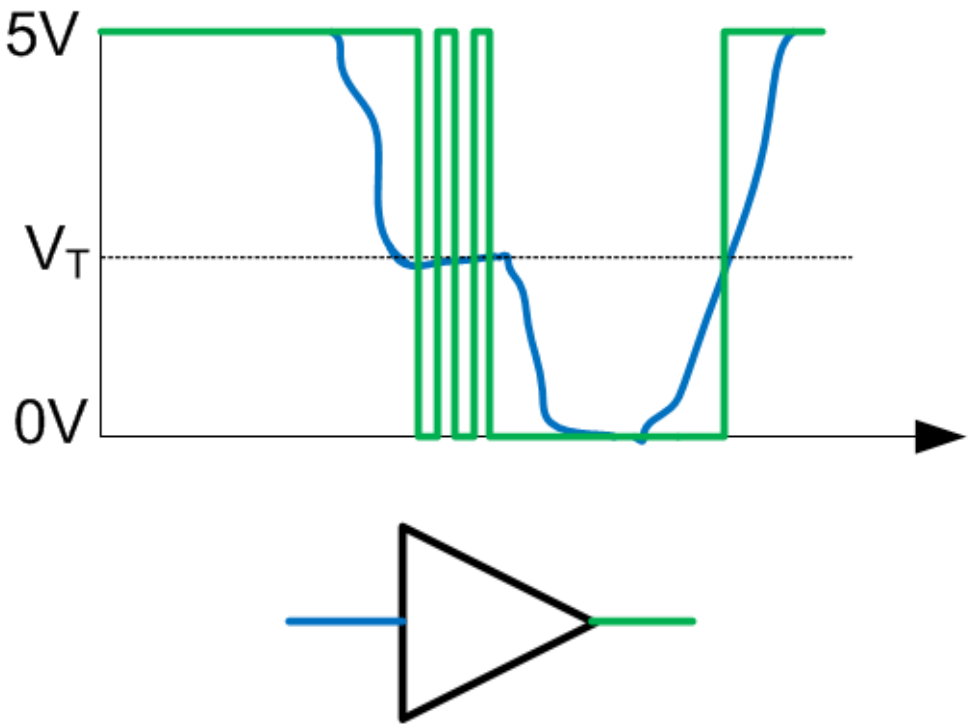


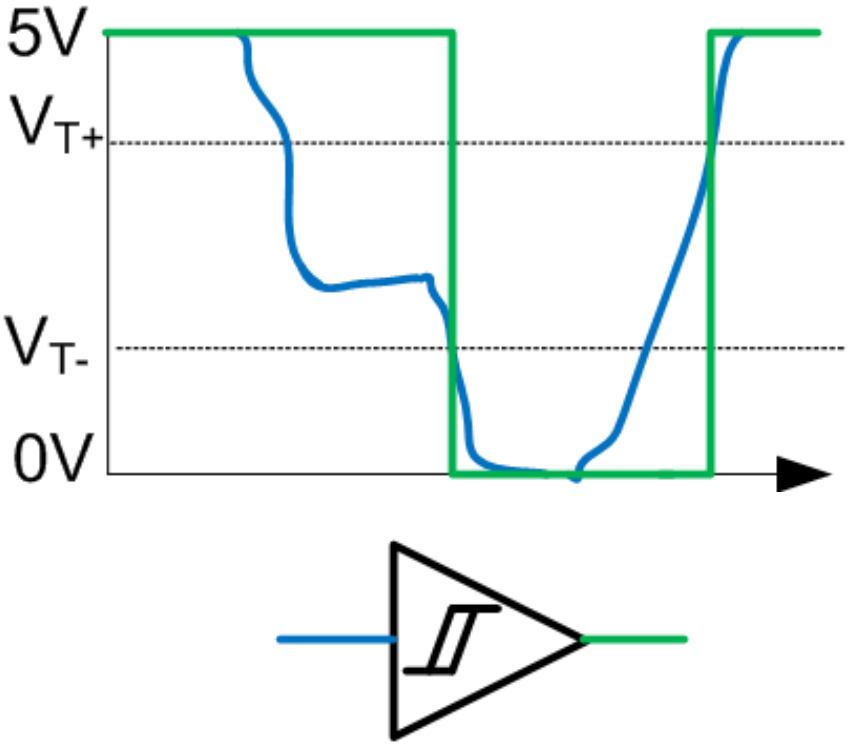


”Schmitt-trigger”

Brusiga insignaler kan generera många oönskade omslag

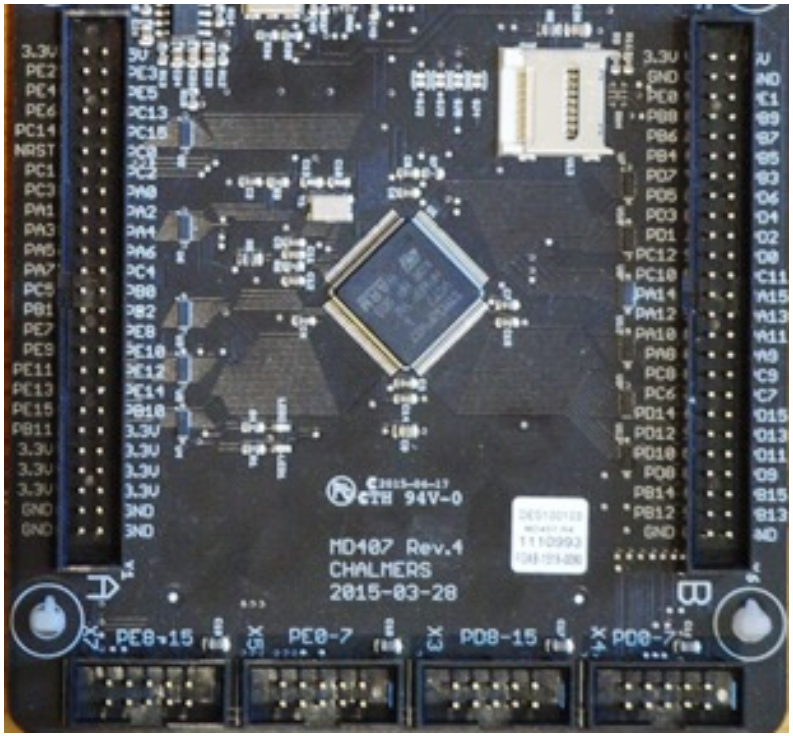
Med fastlagda tröskelnivåer
VT+ för omslag från 0 till 1
och VT- från 1 till 0



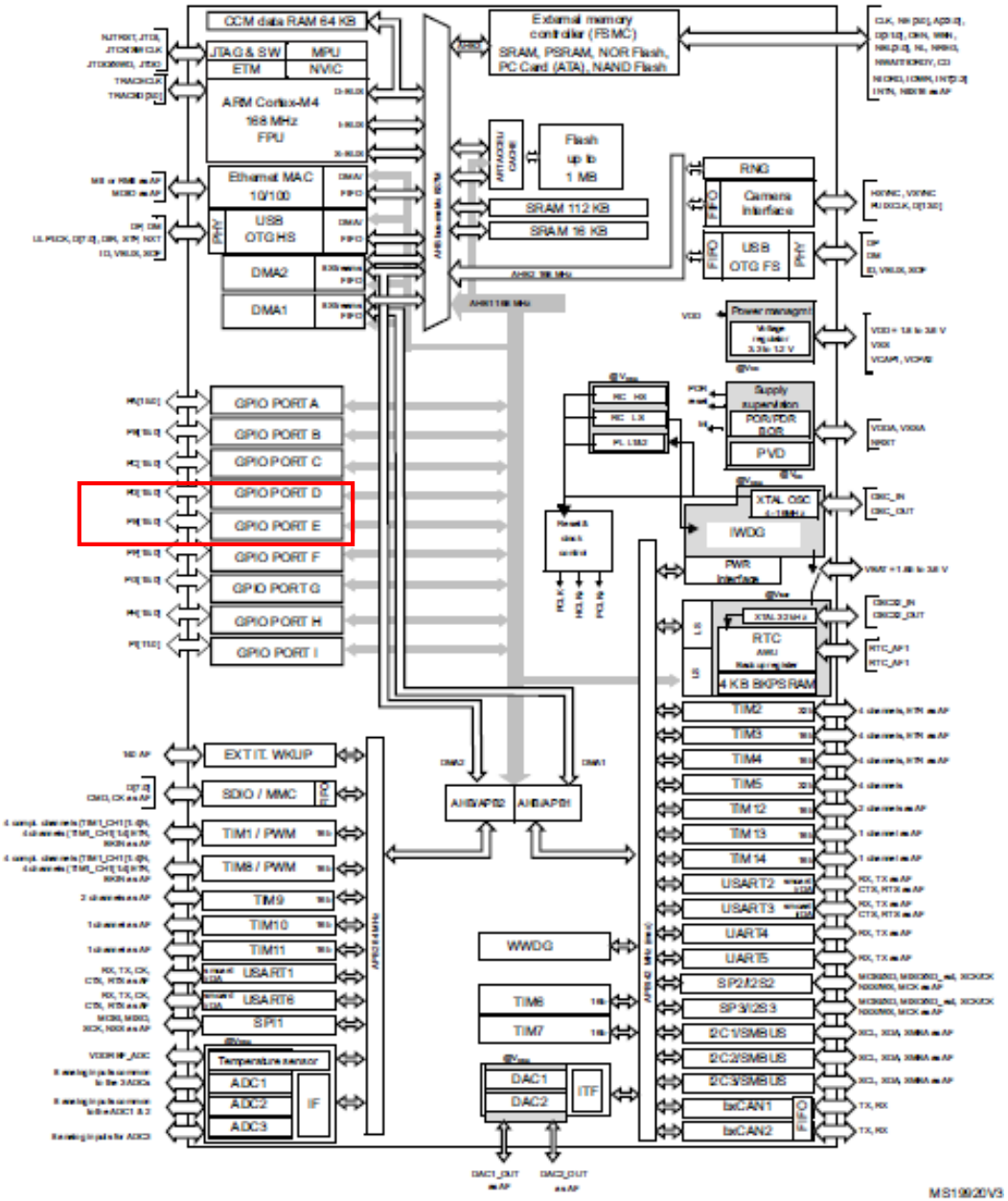


Connections

STM32F407VGT7: Större delen av kretsens 100 pinnar har programmerbar funktion och organiserats i portar (A-E).



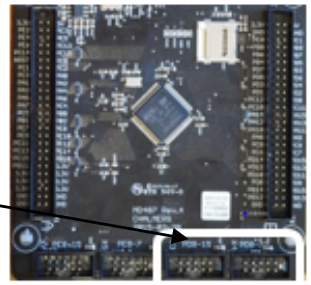
Hos MD407 används portar D och E för generell IO (32 pinnar) medan övriga portar i olika utsträckning används till förutbestämda funktioner.



GPIO-port, programmer's view

För varje port finns en uppsättning register där respektive pinnes funktion kan konfigureras. Registren kan läsas eller skrivas med *byte*, *halfword* eller *word*-operationer.

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																																	GPIO_MODER
4																																	GPIO_OTYPER
8																																	GPIO_OSPEEDR
0xC																																	GPIO_PUPDR
0x10																																	GPIO_IDR
0x14																																	GPIO_ODR
0x18																																	GPIO_BSRR
0x1C																																	GPIO_LCKR
0x20																																	GPIO_AFRL
0x24																																	GPIO_AFRH



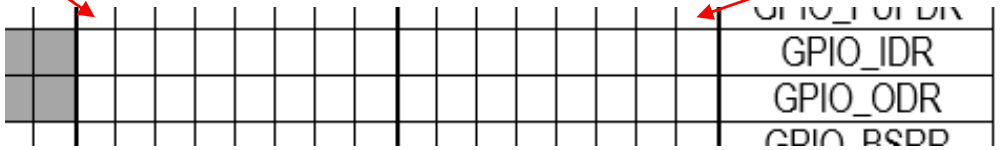
Port D – 16 pinnar
IDR : Input data register
ODR: Output data register

De 16 pinnarna i en port kan konfigureras individuellt i *mode register* (MODER) för någon av funktionerna:

- **00: digital ingång**
- **01: digital utgång**
- *10: alternativ funktion*
- 11: analog funktion

nu behandlar vi pinnarnas funktion som digital IO, dvs. de första två alternativen.

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x0	pin15		pin14		pin13		pin12		pin11		pin10		pin9		pin8		pin7		pin6		pin5		pin4		pin3		pin2		pin1		pin0		GPIO_MODER
	0x03								0x02								0x01								0x00								



EXAMPLE 1

Skriv en assemblerfunktion `app_init` som sätter upp port D bitar 0-7 som en 8–bitars utport och port D bitar 8-15 som en 8-bitars inport.

GPIO

General Purpose Input Output
GPIO A: 0x40020000
GPIO B: 0x40020400
GPIO C: 0x40020800
GPIO D: 0x40020C00
GPIO E; 0x40021000

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																																	GPIO_MODER
4																																	GPIO_OTYPER
8																																	GPIO_OSPEEDR
0xC																																	GPIO_PUPDR
0x10																																	GPIO_IDR
0x14																																	GPIO_ODR
0x18																																	GPIO_BSRR
0x1C																																	GPIO_LCKR
0x20																																	GPIO_AFRL
0x24																																	GPIO_AFRH

Ledning: Det finns en beskrivning av GPIO, port D i Quick-Guide (sidan 19).
Där framgår också att portens basadress är 0x40020C00.

Den önskade funktionen får vi om MODE-registret initieras enligt:

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	GPIO_MODER
	0				0				0				0				5				5				5				5				

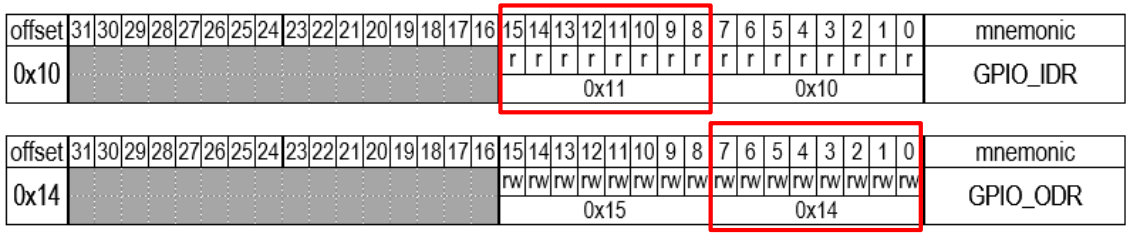
Vilket kan skrivas: `M[0x40020C00] = 0x00005555`

EXAMPLE 1

Verifiera `app_init` genom att, i C, skriva ett enkelt testprogram som läser från inporten och skriver till utporten.

```
void startup(void) __attribute__((naked)) __attribute__((section (".start_section")) );
void startup ( void )
{
__asm (
          " LDR R0,=0x2001C000\n"       /* set stack */
          " MOV SP,R0\n"
          " BL main\n"                  /* call main */
          ".L1: B .L1\n"                /* never return */
          ) ;
}
```

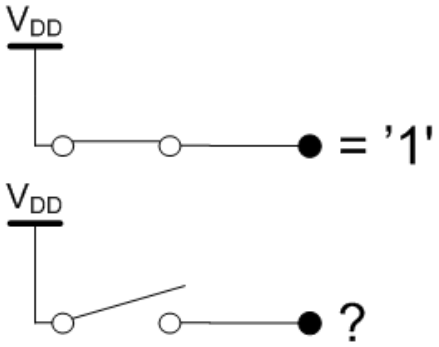
GPIO D = 0x40020C00



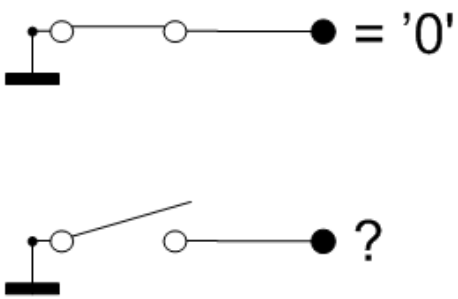

```
...app_init...

void main(void)
{
  unsigned char c;
  app_init();
  while(1){
    c = (unsigned char) *(( unsigned char *) 0x40020C11 );
    * ( (unsigned char *) 0x40020C14) = c;
  }
}
```

Vi löser på tavlan ...

Digital input

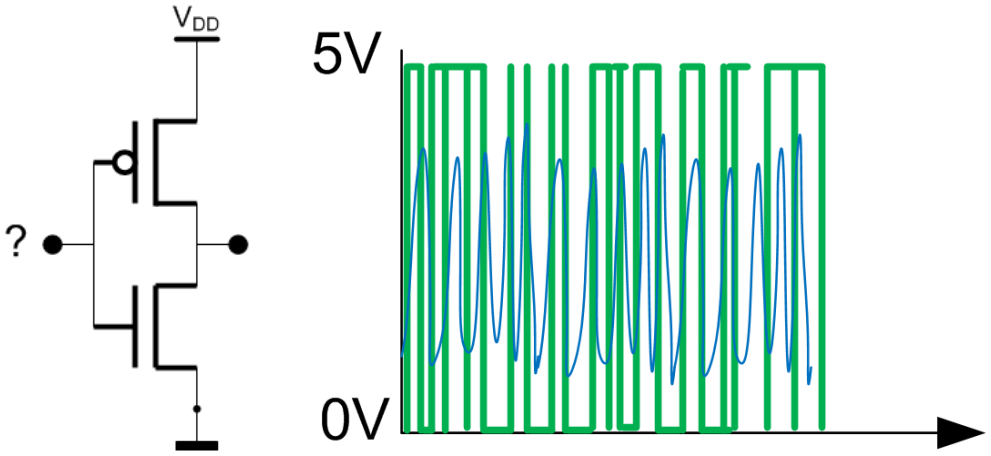


Ger ”säker” etta då brytaren är sluten, annars är signalen obestämd



Ger ”säker” nolla då brytaren är sluten, annars är signalen obestämd

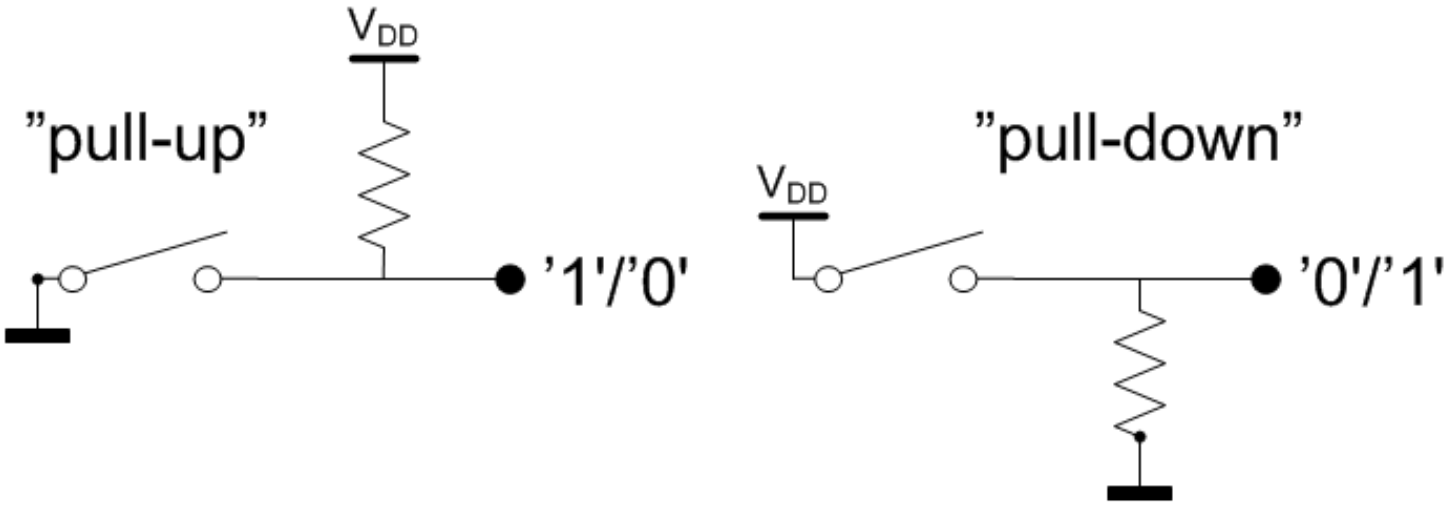
En *obestämd* signal lämnar ingången i ett ”flytande” tillstånd



I ogynsamma fall kan detta resultera i en självsvängande krets som drar mycket ström och dessutom kan orsaka störningar på andra kretselement

”pull-up” or ”pull-down”

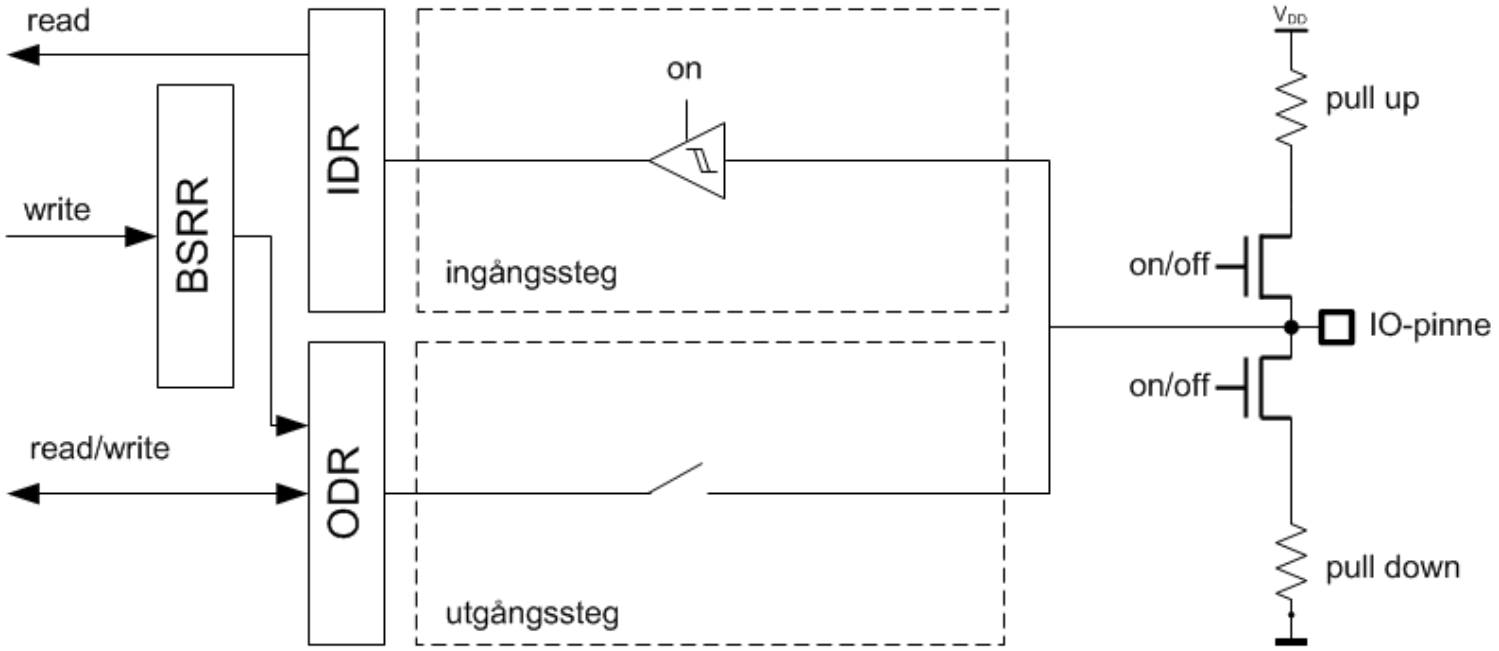
För att säkerställa en stabil nivå på ingången kan man koppla in ett motstånd till antingen V_{DD} eller GND.



Oavsett vilken lösning vi väljer kan vi alltid avgöra om brytaren är öppen eller stängd.

IO-pins input configuration

Vi kan programmera ”pull-up” eller ”pull-down”-funktion, inget externt motstånd behövs.



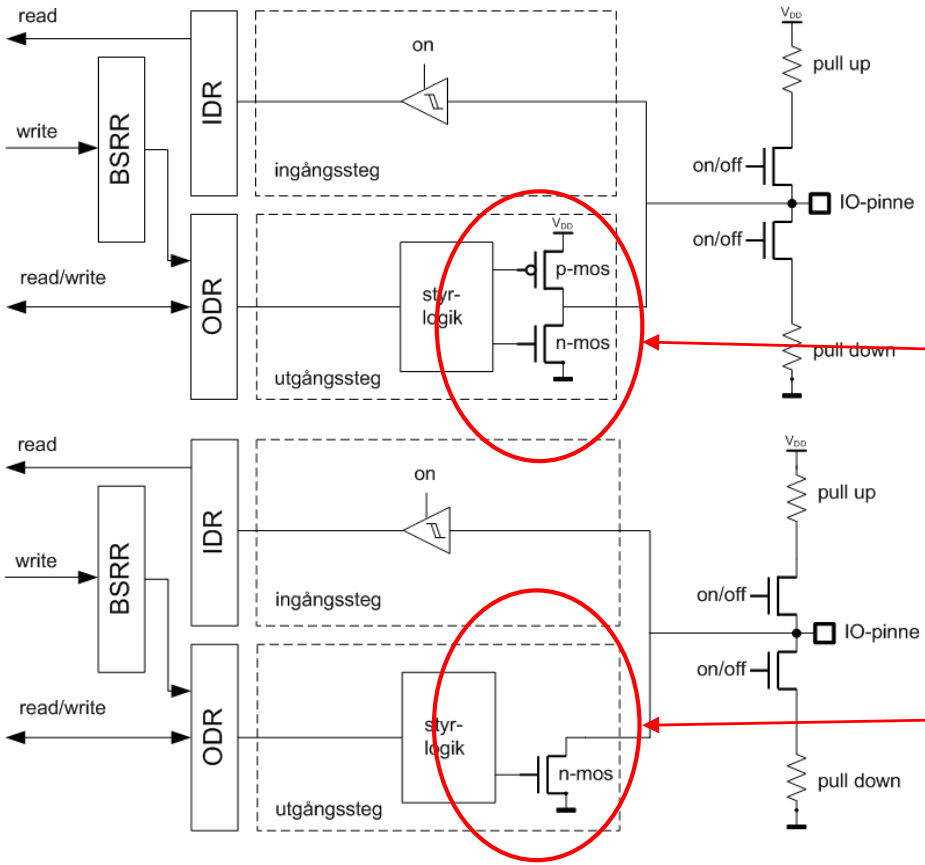
För varje portpinne används 2 bitar i **PUPDR** för att konfigurera pinnen enligt:

- 00: floating
- 01: pull-up
- 10: pull-down
- 11: reserverad

GPIO Pull Up Pull Down Register

offset	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0	mnemonic
0x0C	pin15	pin14	pin13	pin12	pin11	pin10	pin9	pin8	pin7	pin6	pin5	pin4	pin3	pin2	pin1	pin0	GPIO_PUPDR
	0x0F				0x0E				0x0D				0x0C				

IO-pins output configuration



"push-pull" steg

då motsvarande bit i OTYPER är 0

"open drain" steg

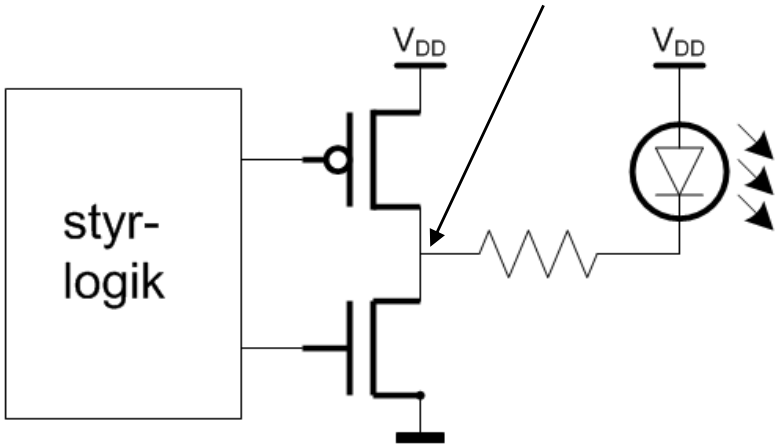
då motsvarande bit i OTYPER är 1

GPIO Output Type Register

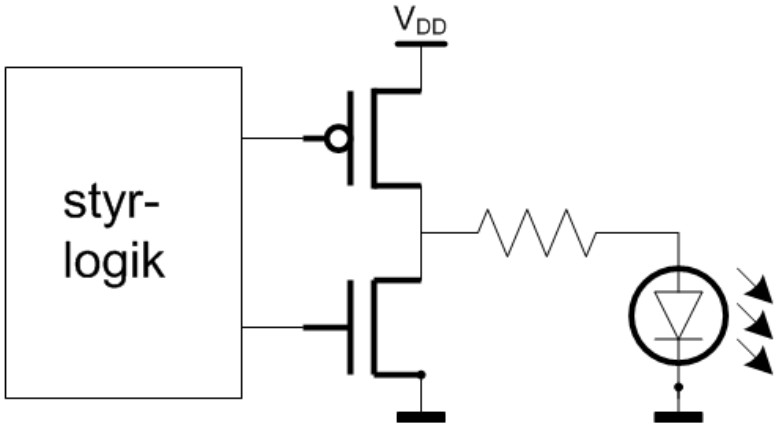
offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x04																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	GPIO_OTYPER
																	0x05								0x04								

”push-pull”

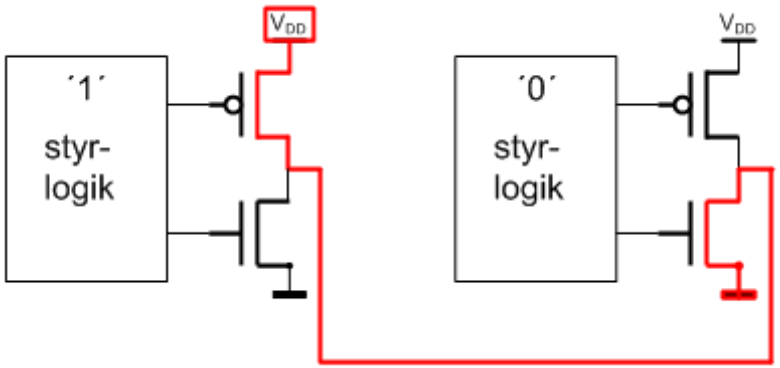
Ljusdioden tänds då utgången är 0.



Ljusdioden tänds då utgången är 1.



Utgångar från flera push-pull steg får inte kopplas samman, eftersom det kan leda till kortslutning mellan V_{DD} och GND.

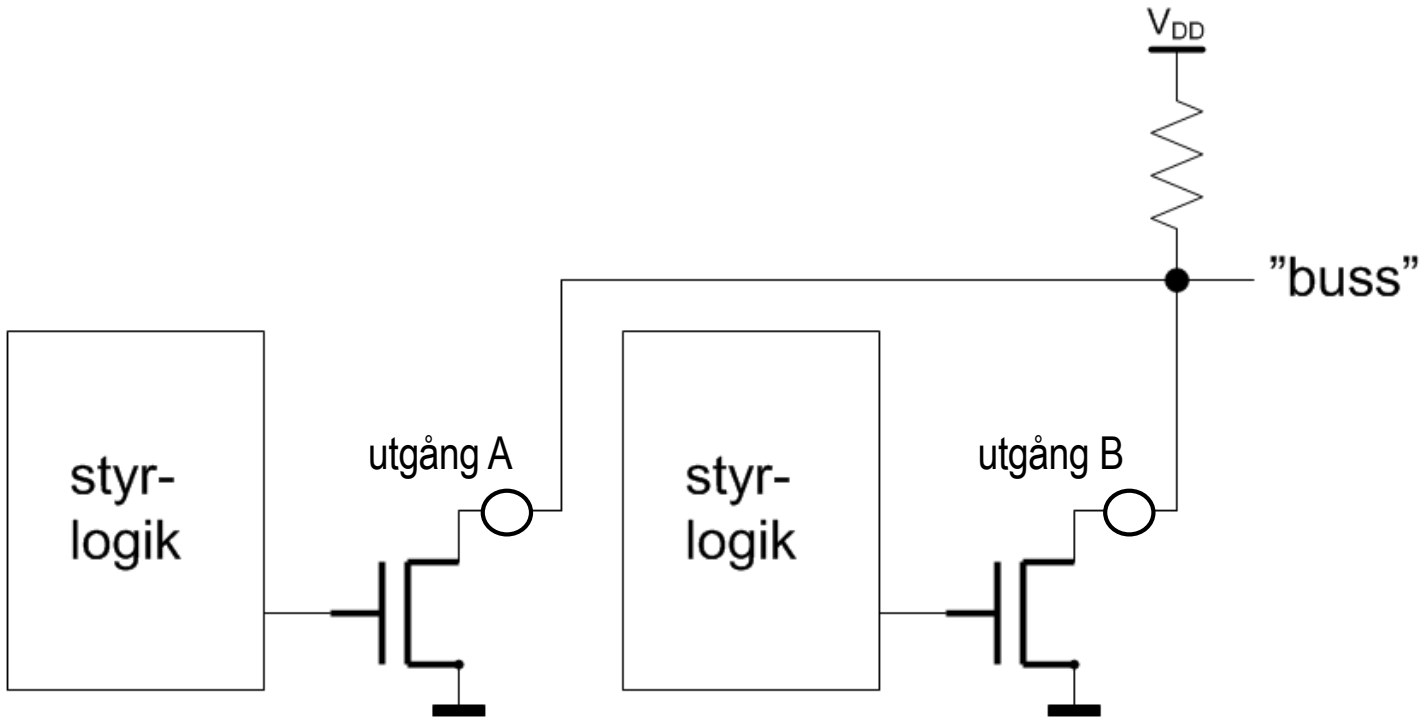


”open drain”

Utgångar från flera open-drain steg kan kopplas samman utan problem.

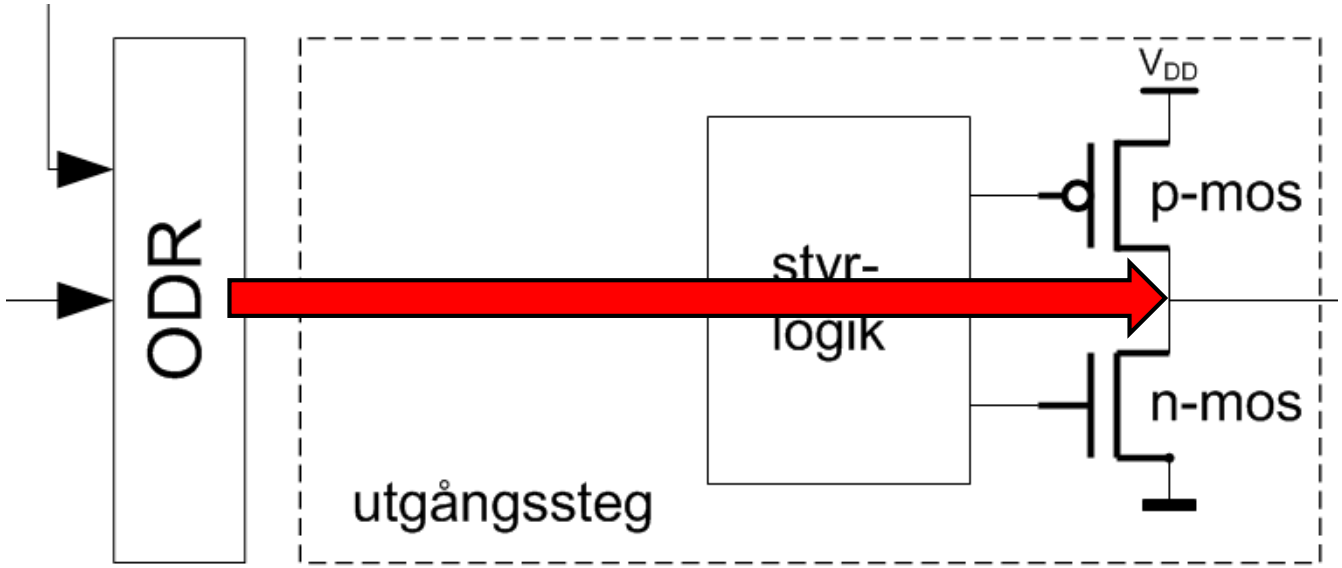
Nivån på den gemensamma "bussen" bestäms av att:

- Om alla utgångar är '1' är också bussnivån '1'
- Om någon utgång är '0' är också bussnivån '0'



”output speed"

Bestämmer hur ofta registrets innehåll överförs till utgångssteget. Ju lägre frekvens desto mindre strömförbrukning.



För varje portpinne används 2 bitar i OSPEEDR för att konfigurera pinnen enligt:

- 00: Low speed, 2 MHz
- 01: Medium speed, 25 MHz
- 10: Fast speed, 50MHz
- 11: High speed, 100 MHz

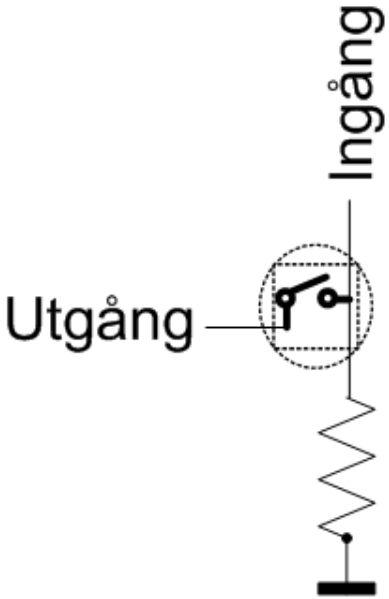
Om hastigheten är 50 MHz eller mer måste en så kallad "IO-kompensationscell" aktiveras

GPIO Output SPEED Register

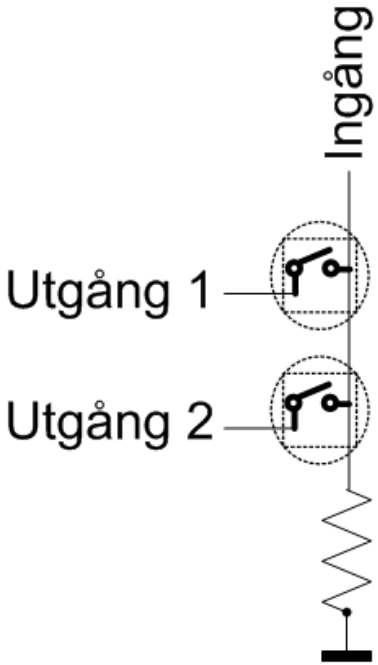
offset	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0	mnemonic
0x08	pin15	pin14	pin13	pin12	pin11	pin10	pin9	pin8	pin7	pin6	pin5	pin4	pin3	pin2	pin1	pin0	GPIO_OSPEEDR
	0x0B				0x0A				0x09				0x08				

Multiplex function

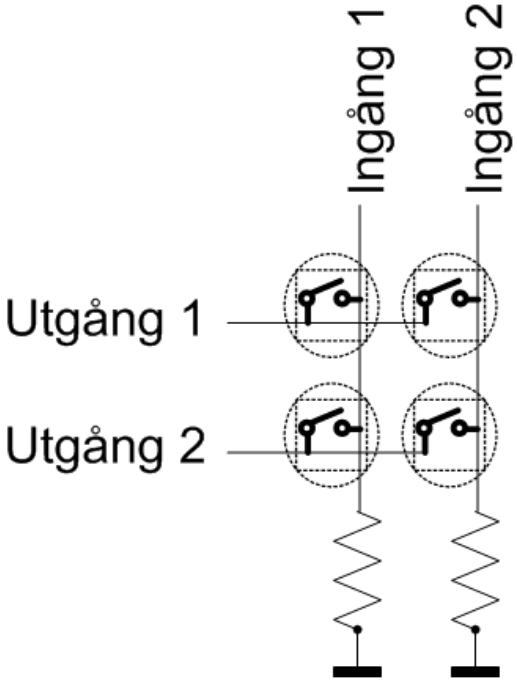
Använd samma ingång för flera olika funktioner.



1 funktion
2 anslutningar



2 funktioner
3 anslutningar



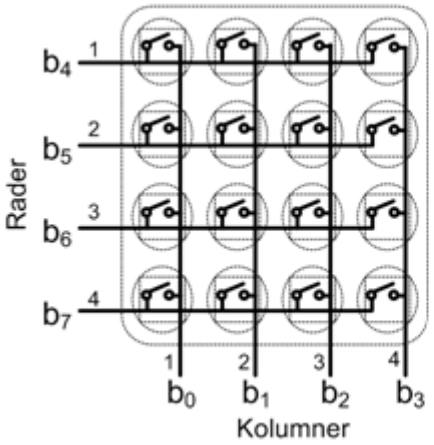
4 funktioner
4 anslutningar

EXAMPLE 2 Routine for scanning a keyboard.

```
unsigned char keyb( void );
```

Funktionen ska avsöka tangentbordet en gång.

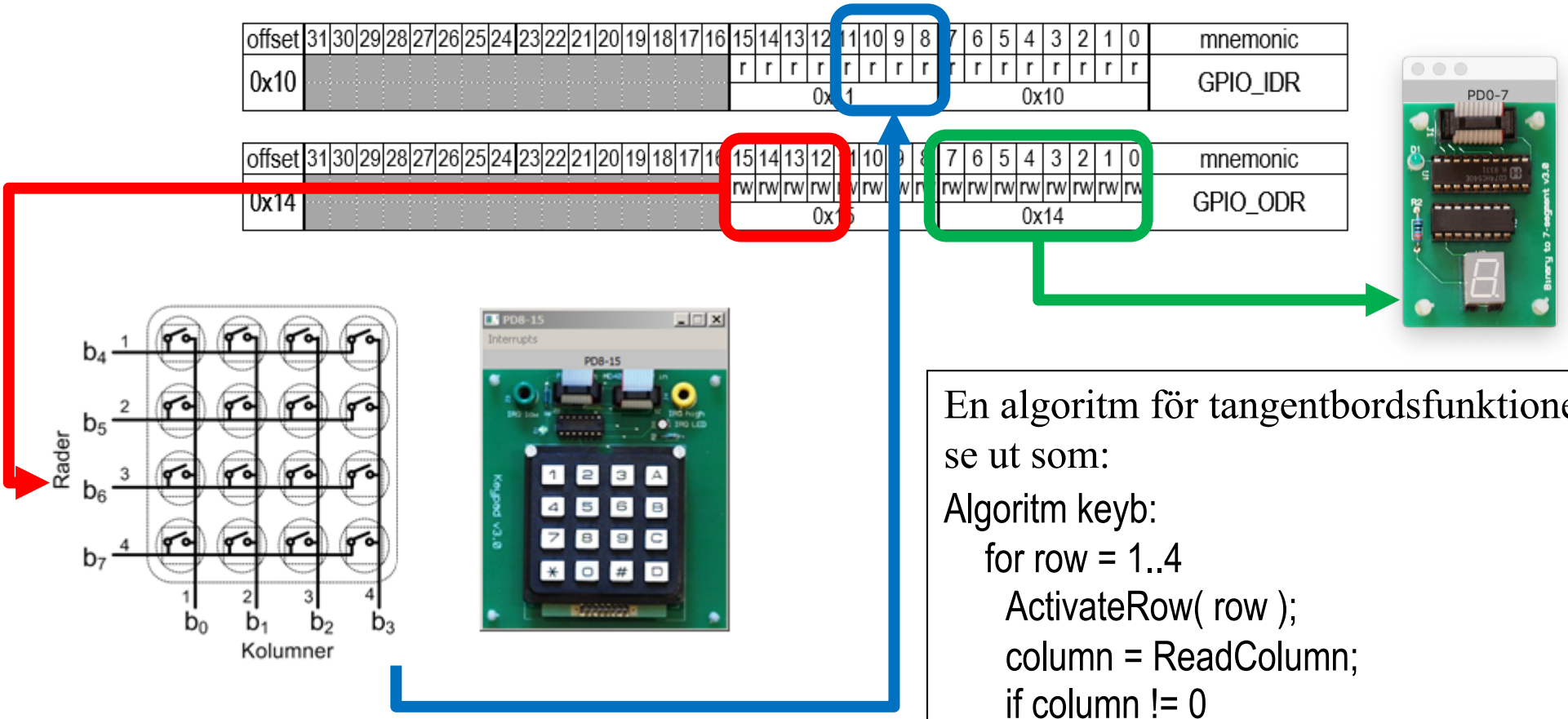
- Om ingen tangent är nedtryckt ska funktionen returnera värdet 0xFF.
- Om någon tangent är nedtryckt ska dess tangentkod returneras.
- Om flera tangenter är nedtryckta är valet av tangentkod, bland dessa, godtyckligt.



En algoritm för tangentbordsfunktionen kan se ut som:

```
Algoritm keyb:
    for row = 1..4
      ActivateRow( row );
      column = ReadColumn;
      if column != 0
        keyb  = keyValue [pressed key ];
keyb = 0xFF;
```

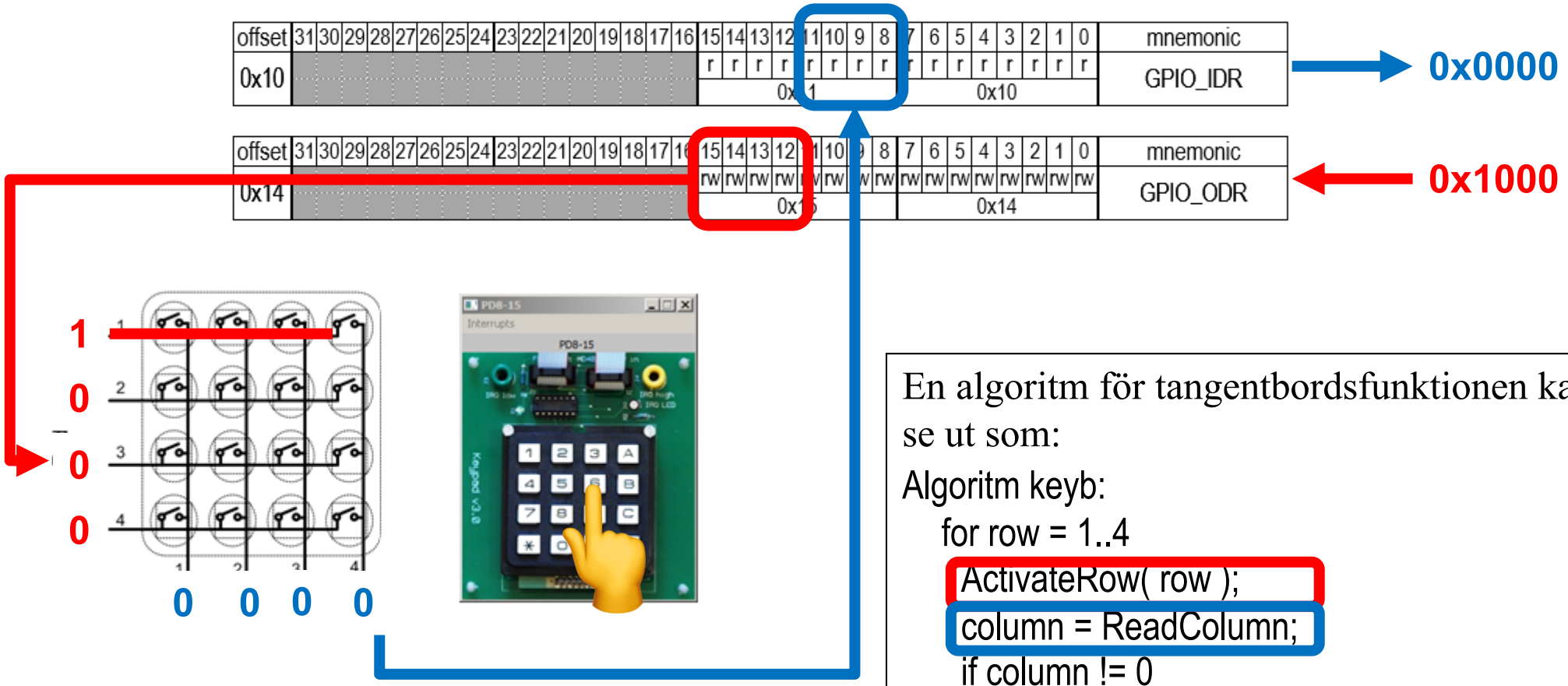
Scan Keyboard



En algoritm för tangentbordsfunktionen kan se ut som:

```
Algoritm keyb:
    for row = 1..4
      ActivateRow( row );
      column = ReadColumn;
      if column != 0
        keyb  = keyValue [pressed key ];
keyb = 0xFF;
```

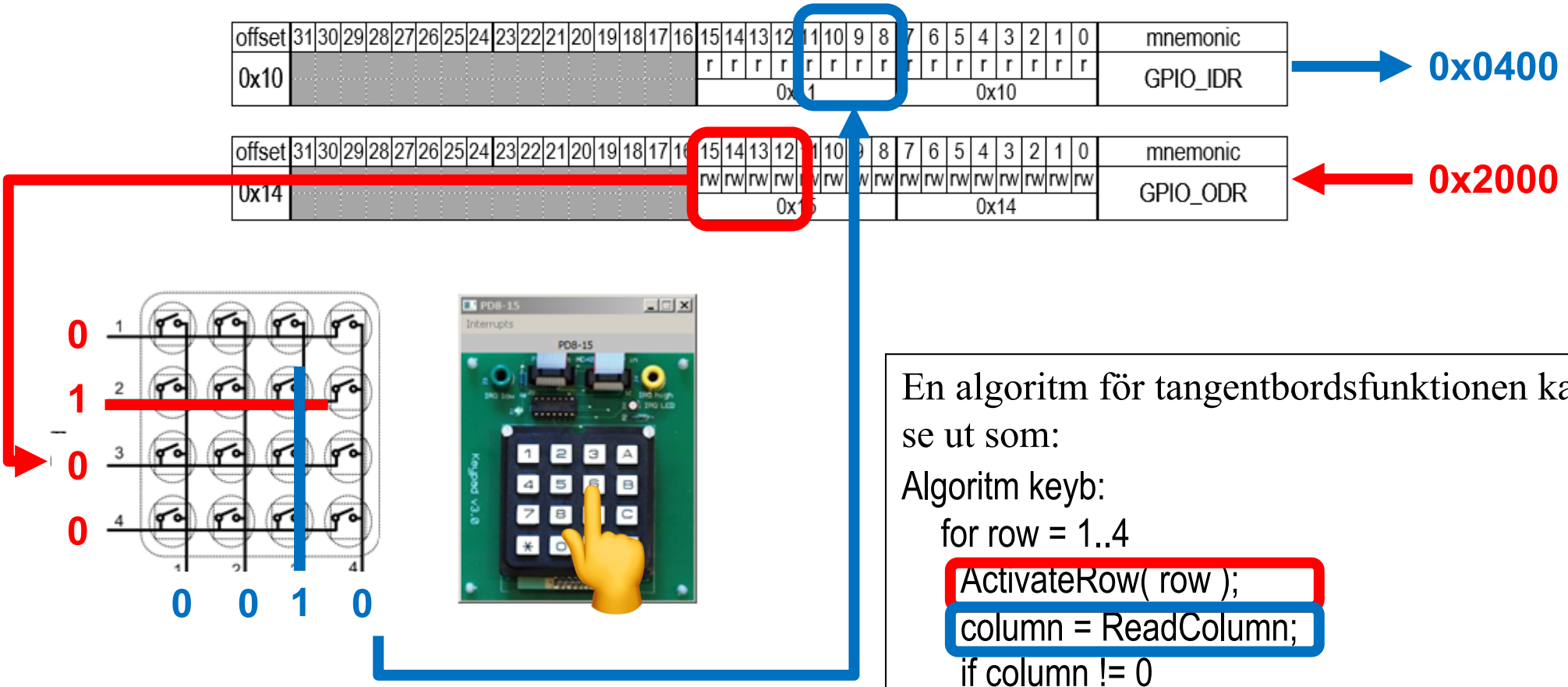
Scan Keyboard (1)



En algoritm för tangentbordsfunktionen kan se ut som:

```
Algoritm keyb:
    for row = 1..4
      ActivateRow( row );
      column = ReadColumn;
      if column != 0
        keyb  = keyValue [pressed key ];
keyb = 0xFF;
```

Scan Keyboard (2)



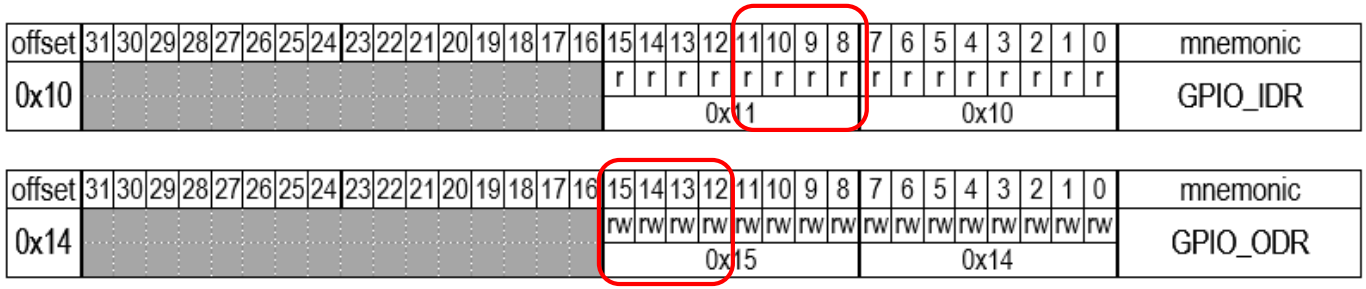
En algoritm för tangentbordsfunktionen kan se ut som:

```
Algoritm keyb:
    for row = 1..4
      ActivateRow( row );
      column = ReadColumn;
      if column != 0
         keyb  = keyValue [pressed key ];
keyb = 0xFF;
```

EXAMPLE 2

Vi förutsätter nu att Port D (8-15) initierats enligt Uppgift 18 i arbetsboken...

Observera att vi bara använder de 8 mest signifikanta pinnarna i porten,
4 av dessa är ut-pinnar, de andra 4 är in-pinnar...



Följande definitioner är nu behändiga...

```
#define  GPIO_D             0x40020C00
#define  GPIO_MODER         ((volatile unsigned int *)          (GPIO_D))
#define  GPIO_OTYPER        ((volatile unsigned short *)        (GPIO_D+0x4))
#define  GPIO_PUPDR         ((volatile unsigned int *)          (GPIO_D+0xC))
#define  GPIO_IDR_LOW       ((volatile unsigned char *)         (GPIO_D+0x10))
#define  GPIO_IDR_HIGH      ((volatile unsigned char *)         (GPIO_D+0x11))
#define  GPIO_ODR_LOW       ((volatile unsigned char *)         (GPIO_D+0x14))
#define  GPIO_ODR_HIGH      ((volatile unsigned char *)         (GPIO_D+0x15))
```

Vi löser på tavlan ...

BSRR – synchronized update

Kan användas för en synkroniserad ändring av flera utgångars pinnar.

bit 0..15: bit reset

bit 16-31: bit set.

BSRR (bit set/reset register)

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	mnemonic
0x18	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	GPIO_BSRR
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0	

”Bitlock” function

Då porten har programmerats kan konfigurationen låsas som skydd mot ofrivillig eller otillåten manipulation av portens inställningar. Detta sker genom att en speciell sekvens skrivs till detta register.

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	mnemonic
0x1C																LCKK	GPIO_LCKR
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0	

Alternative function – ”route” internal circuits

AFRL (alternate function low register)

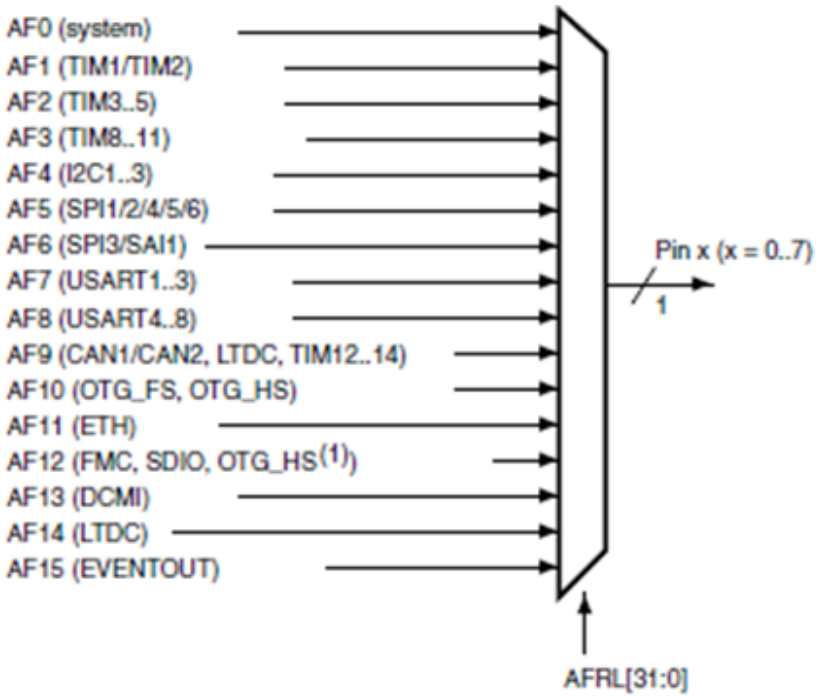
offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x20	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]				GPIO_AFRL

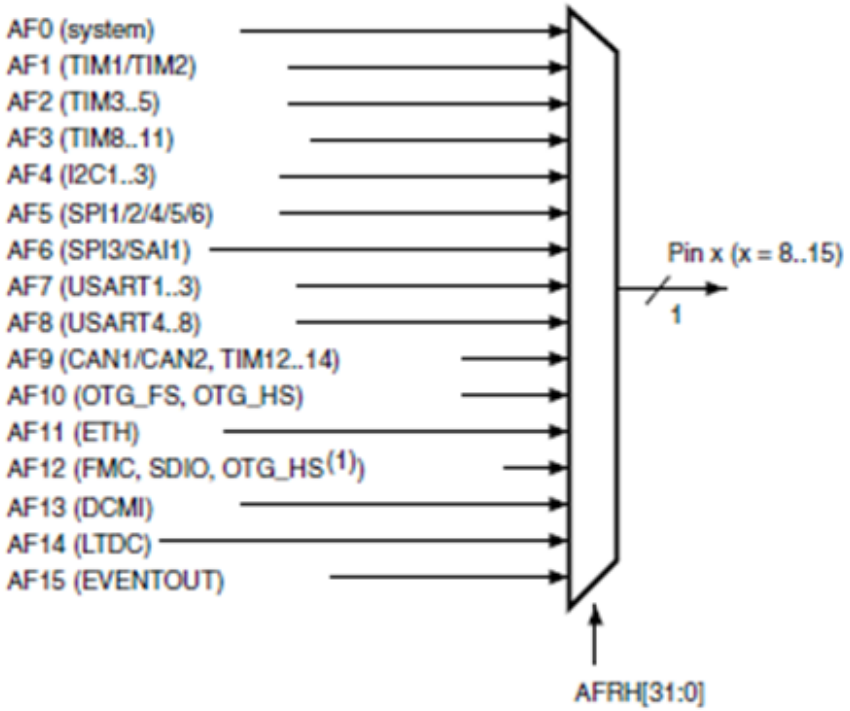
AFRH (alternate function high register)

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic
0x24	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]				GPIO_AFRH

De 16 pinnarna i en port kan konfigureras individuellt i *mode register* (MODER) för någon av funktionerna:

- 00: digital ingång
- 01: digital utgång
- **10: alternativ funktion**
- 11: analog funktion





Analog function

De 16 pinnarna i en port kan konfigureras individuellt i *mode register* (MODER) för någon av funktionerna:

- 00: digital ingång
- 01: digital utgång
- 10: alternativ funktion
- **11: analog funktion**