Port & Bit-manipulation i Arduino:

Links til afsnit i dokumentet:

Indledning:

Hvordan er det lige, man kan arbejde med bits og bytes i programmeringssproget C ??

Fx 0x30 or'et med 0xB5

Eller hvordan laver man et tal om til Ascii tal ?? Fx 0x08 'or' 0x30 = 38h = 8 Ascii

Hidtil har vi arbejdet "the Arduino way", som på mange måder er lidt hobby-præget.

Med dette kompendium kommer vi til at arbejde direkte med registre.

Fx funktionen ”digitalWrite”. Compileren koder det om til mange bytes, fordi der kaldes en – skjult subrutine, der skal regne ud, hvilken bit i hvilken port, der skal skrives til. Det tager lang tid, - og kan godt gøres betydelig kortere!!
Se fx linket her

Arduino er født med en version af højniveausproget “C”, der er beregnet til at håndtere Boardets I/O-pins som individuelle pins med en digitalWrite-funktion. Det sker med en digitalWrite- eller digitalRead-funktionen.

Men i vores tekniske tilgang til uC’er ved vi godt, at der bag disse pinnumre gemmer sig nogle porte, med portpins fra 0 til 7.

Vi vil godt nogle gange direkte kunne håndtere en 8-bit port, eller direkte kunne manipulere enkelte portbit uden digitalWrite-funktionen.

Både fordi det er praktisk, men også fordi det er betydelig hurtigere end med 1 pin ad gangen ”The Arduino way”.

Heldigvis er der i Arduino”C” compileren implementeret kommandoer, så det kan lade sig gøre. Det er både praktisk, men også betydelig hurtigere end ”The Arduino way”.

Eksempel, digitalWrite():

Den funktion, der kaldes med fx ordren `digitalWrite(9,HIGH);` skal jo først beregne hvilken bit og i hvilken port, der skal sættes høj. Det tager tid !!

Her sammenlignes ordrer, the arduino-way og med direkte portmanipulation:

```
digitalWrite(9,HIGH);          // funktionen tager 57 clock cycles
digitalWrite(9,LOW);           // 57 clock cycles

bitSet(PORTB,1);               // 2 clock cycles
bitClear(PORTB,1);             // 2 clock cycles
bitWrite(PORTB,0,value);       // 3 clock cycles
```

Resultatet er ca. 25 gange hurtigere kode!

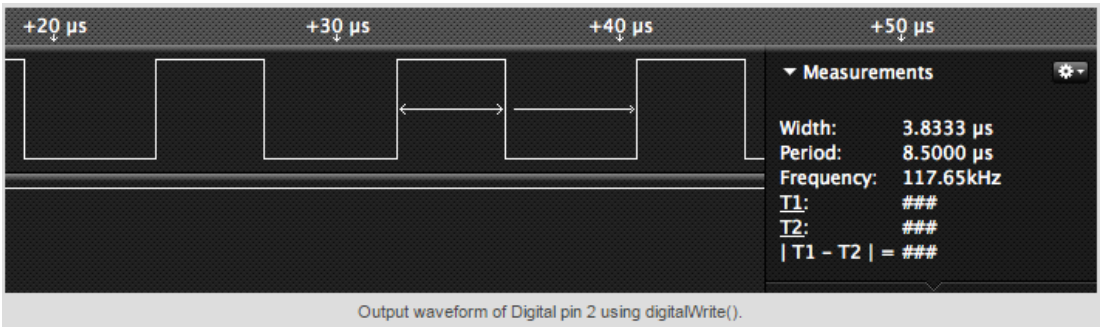
Det understøttes af følgende kodeeksempler med grafer fra Scoopet:

Først ”the Arduino way”:

```
int outPin = 2;                 // Use digital pin 2 as output
void setup()
```

```
{
  pinMode(outPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(outPin, HIGH);   // sets output high
  digitalWrite(outPin, LOW);    // sets output low
}
```

Og grafen målt på pin 2



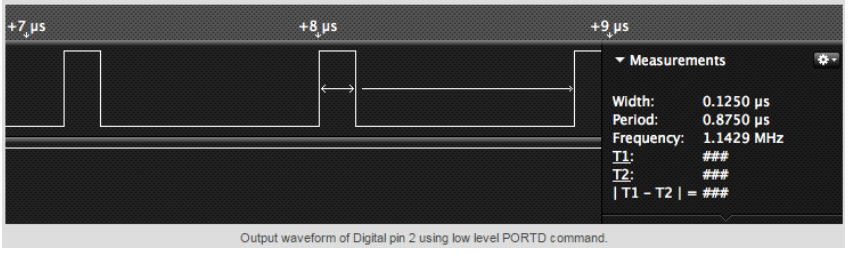
Output waveform of Digital pin 2 using digitalWrite().

Dernæst samme program med direkte bit manipulation i et portregister.

```
#define PD2 2
int outPin = 2;                 // Use digital pin 2 as output

void setup()
{
  pinMode(outPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  bitSet(PORTD,2);      // sets output bit 2 high
  bitClear(PORTD,2);    // sets output bit 2 low
}
```

Resultatet bliver en meget kortere puls.

Mellemrummet mellem pulsene kommer af, at programmet jo skal loope til loopens start.



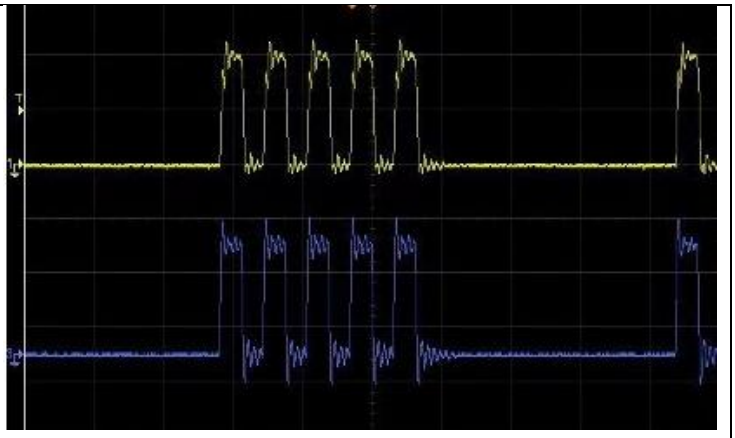
Output waveform of Digital pin 2 using low level PORTD command.

Ovenstående taget fra: http://skpang.co.uk/blog/archives/323

Det ses også, at der indgår nogle andre koder, - bitSet og bitClear - end det vi hidtil har set.

Her ses et andet eksempel:

```
void loop()
{
  PORTD = B11111111;
  PORTD = B00000000;
  PORTD = B11111111;
  PORTD = B00000000;
  PORTD = B11111111;
  PORTD = B00000000;
  PORTD = B11111111;
  PORTD = B00000000;
  PORTD = B11111111;
  PORTD = B00000000;
}
```



Kilde: (Ret god !!!) http://tronixstuff.com/2011/10/22/tutorial-arduino-port-manipulation/

Et andet problem kunne være følgende situation:

```
digitalWrite(10, HIGH);
digitalWrite(11, HIGH);


PORTB |= B1100
```

Resultatet vil være, at Pin 10 går høj flere mikrosekunder før Pin 11.

Dette kan give problemer for tidsfølsomt ekstern digital elektronik.

Her sættes begge Pins høj på nøjagtig same tid.

Det ses, at der indgår nogle andre koder, end dem vi hidtil har set. Det er netop det, dette kompendium drejer sig om!!

Sammenhængen mellem Arduinos Pinnumre og processorens Portnumre

En af de ting, der skal håndteres ved direkte portmanipulation er, at i Arduinoens verden er pin-numre forskellige fra uC-ens port-betegnelser.

Uno'ens Pinnumre er vist på Boardet.

Og sammenhængen til controllerens porte og portpins er vist herunder:

Her til højre ses pin- og port-konfigurationen på den benyttede processor, Atmega328.

PD – digital pins 7 til 0 (Port D)
PB – digital pins 13 til 8 (Port B)
PC – analog pins 5 til 0 (Port C)

Atmega168 Pin Mapping

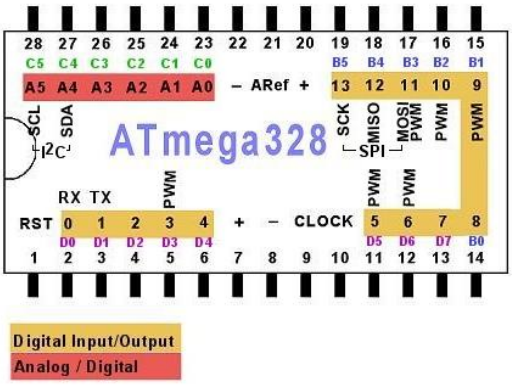
Arduino function			Arduino function
reset	(PCINT14/RESET) PC6	1	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	analog input 0
VCC	VCC	7	GND
GND	GND	8	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	digital pin 9 (PWM)
		15	PB1 (OC1A/PCINT1)
		16	PB2 (SS/OC1B/PCINT2)
		17	PB3 (MOSI/OC2A/PCINT3)
		18	PB4 (MISO/PCINT4)
		19	PB5 (SCK/PCINT5)
		20	AVCC
		21	AREF
		22	GND
		23	PC0 (ADC0/PCINT8)
		24	PC1 (ADC1/PCINT9)
		25	PC2 (ADC2/PCINT10)
		26	PC3 (ADC3/PCINT11)
		27	PC4 (ADC4/SDA/PCINT12)
		28	PC5 (ADC5/SCL/PCINT13)

Sammenhængen vist på en lidt anden måde:

Port B har pin **B0 to B5**

Port C har pin **C0 to C5**

Port D har Pin **D0 to D7**



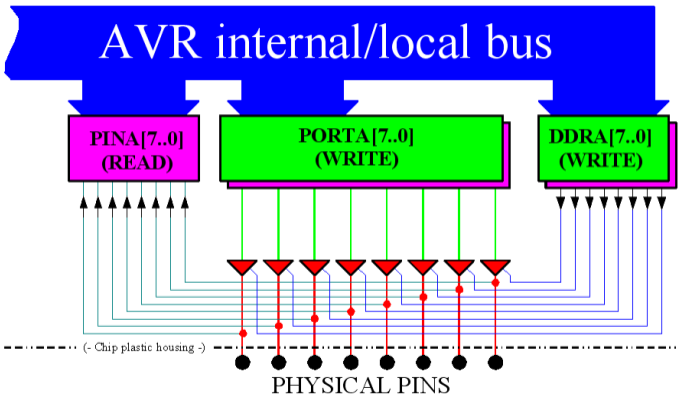
Bemærk, at man normalt ikke kan bruge pin 0 og 1 som output – fordi de bruges til kommunikation til PC-en!

Portregistre:

Se: https://iamsuhasm.wordpress.com/tutsproj/avr-gcc-tutorial/

Først ses lidt på, hvordan portene og de interne styreregistre er opbygget i Atmega328P:

Her et billede af opbygningen af en ports tilhørende registre og den interne bus.



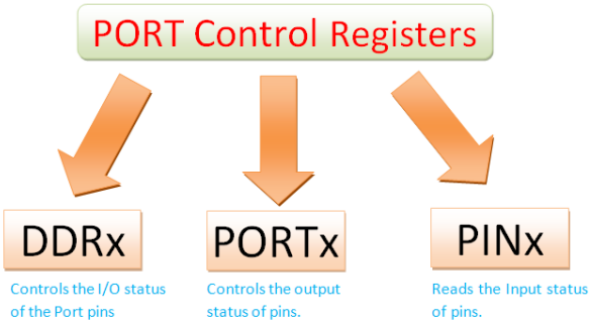
Kilde: http://www.uio.no/studier/emner/matnat/fys/FYS3240/v14/forelesninger/l4---uc_basic_and_c_basic.pdf

Hver port i microcontrolleren og dets pins kontrolleres af 3 SFR, - special function registre, hvis navne også er forud-definerede variable i Arduino-C-sproget.

DDR-registeret – DataDirectionRegisteret bestemmer om de tilhørende pins er input eller output.

Ved at skrive til **PORT-registeret** kontrollerer man om pins er høje eller lave.

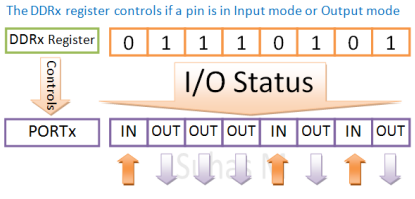
Og fra Input-registreret **PIN-x** kan et program læse om en input pin er Høj eller Lav.



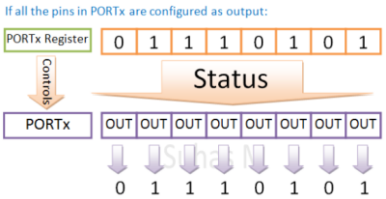
x kan være B, C eller D

Kilde: https://iamsuhasm.wordpress.com/tutsproj/avr-gcc-tutorial/

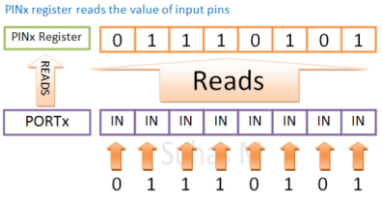
Her en anden oversigt !!



Et 0 i DDR-registeret bestemmer at denne pin er input



En værdi skrevet til PORTx registeret sendes direkte ud på porten



Input fra pins læses i PINx registeret

Dvs. At de tre I/O-port manipulation kommandoer er:

DDRx – "replacement" for pinMode()
PORTx – "replacement" for digitalWrite()
PINx – "replacement" for digitalRead()

PINx er egentlig et register kun beregnet til at læse en pin eller en port. Men den er lavet med XOR, dvs. at man ved at skrive et '1' til registeret, vil den tilhørende pin toggle.

This uses a relatively little-known feature of the hardware, which actually has "bit flipping" built-in. The "PIND" register is normally used for input, i.e. for reading the state of a pin as an input signal. But you can also write to that register. When you do, it will be used to *flip* output pins, *but only for the bits which were set to 1*. ?? It's essentially a built-in XOR.

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn.

I det følgende er vist eksempler på hvordan man kan manipulere en 8-bit port, en SFR mm:

Skriv til hel Port eller SFR-register	
	Eksempler på at skrive til hel port / SFRregister (Kilde: [1])
`PORTD = B11111111;  // = 255`	
`PORTD = 0b00000000; // = 0`	
`PORTD = 0x3A;       // = 3Ah`	
`DDRD = 0b00101010;`	
`DDRD = 0x2A;`	`// Pin 2 i PortD er output.`
`DDRD = (1<<PORTD2);`	`//  Note# [2]`
`DDRD = DDRD | B11111100;`	`// Set bit 7 only.`
`PORTC |= 0x80;`	`// Definer en variabel`
`byte pinState = B00000000;`	

Bemærk at man ikke kan kontrollere pin 0 og 1 i portD og samtidig bruge den serielle debug-monitor.

Port-manipulation på normal vis

[1] http://little-scale.blogspot.dk/2013/05/teensy-basics-5-port-manipulation.html
[2] Sæt pin 2 til 7 til output, uden at ændre pin 0 & 1, som er RX & TX

```
// Digital 0 to 7 set as outputs, then on/off using digitalWrite()

void setup()
{
  for (int a=0; a<8; a++)
  {
    pinMode(a, OUTPUT);
  }
}

void loop()
{
  for (int a=0; a<8; a++)
  {
    digitalWrite(a, HIGH);
  }
  for (int a=0; a<8; a++)
  {
    digitalWrite(a, LOW);
  }
}
```

Eks. Med direkte manipulation:

```
// Arduinokode til at køre knightrider, made by Kenneth !!

void setup() {
  DDRD = 0xFF;
}

void loop() {
  while (1) {
     for(PORTD = 0x01; PORTD != 0; PORTD <<= 1) // Skift alle bit 1 til
        delay(50);                              // venstre

     for (PORTD = 0x80; PORTD != 0; PORTD >>= 1) // Skift 1 bit til højre
        delay(50);
  }
}
```

```
int delayTime = 333;            //It's better coding style not to have any
                                // hard-coded constants
byte portD_HIGH = B11111000;
```

```
byte portD_LOW =  B00000000;

void loop()
{
    PORTD = portD_HIGH;         //sets pins High
    delay(delayTime);           //wait
    PORTD = portD_LOW;          //sets pins to Low
    delay(delayTime);           //waits another 333 ms
}
```

```
void binaryCount()
{
  for (int z=0; z<256; z++)
  {
    PORTD = z;
    delay(100);
  }
  PORTD=0;
}
```

Array Eksempel

```
// Example 43.5
// tronixstuff.com/tutorials > chapter 43
// John Boxall - October 2011
// inputs and outputs

byte segments[] = {
  B01111110, B00110000, B01101101, B01111001, B00110011, B01011011, B01011111,
B01110000, B01111111, B01111011};

// digital pins 7~0 connected to display pins dp,a~g

void setup()
{
  DDRB = B00000000; // set PORTB (digital 13~8) to inputs
  DDRD = B11111111; // set PORTD (digital 7~0) to outputs
}

void disp(int z)
{
  PORTD = segments[z];
}

void loop()
{
    disp(PINB);
```

```
    delay(100);
}
```
// Fra: http://tronixstuff.com/2011/10/22/tutorial-arduino-port-manipulation/

Læs hel Port:	En port læses fra dets PIN-register.
`Tal=PIND;`	Læs hele PORTD fra PIN-registeret til Variabel Erstatning for at bruge funktionen digitalRead() 8 gange.

`if(PIND == B00000000) {`
`do something`
`}`	

`char my_var = 0;` `my_var = PIND;`	Læs Port D til variable.

Manipuler et enkelt bit i en SFR eller i en variabel:

bitSet	Forklaring Bemærk: Bit tælles fra bit 0 til 7

`bitSet(x, bitPosition)`
`bitSet(PORTD, 4);`
`bitSet(TCCR1B, CS12);`
`bitSet(TCCR1B, CS10);`
 	
`byte pinState = B00000000;`
`bitSet(pinState, 2);`
	Syntax:
	I variable x sættes et '1' i en given bitPosition
	
	Set bit 4 i portD
	I SFR TCCR1B i timeren sættes bit CS12 og CS10
	
	//sæt 3. pin høj
	//pinState = B00000100
	
	Virker også på selvdefinerede variable.

bitClear	

`bitClear(x, bitPosition)`
 	
`bitClear(PORTD, 4);`
	I variable x cleares en given bitPosition
	Clr bit 4 i portD

`byte pinState = B11111111;`
 	
`bitClear(pinState, 2);`
	`Initializer alle pins til HIGH`
	`Sæt den tredie pin LOW`
	
	`pinState = B11111011`

bitWrite	

`bitWrite(x, bitPosition, value)`
 	
`bitWrite(PORTB, 3, 1);`
`bitWrite(PORTB, 3, 0);`
	`Skriv 0 eller 1 i bitpos. i variablen x.`
	
	`Sæt bit 3 i portB høj`
	`Clear bit 3 i portB`

`byte pinState= 0;`
 	
`bitWrite(pinState, 0, HIGH);`
`bitWrite(pinState, 3, HIGH);`
`bitWrite(pinState, 0, LOW);`
	`//initializer alle pins til LOW (B00000000)`
	`//pinState = B00000001`
	`//pinState = B00001001`
	`//pinState = B00001000`

bitRead(x, n)	Syntax:`var = bitRead(x, bitPosition)` Variablen var får værdien 0 eller 1 afhængig af en given bitposition i tallet x (bit 0 til 7 fra højre)
`byte a = bitRead(x, 5);`	`Læs bit 5 i reg x`

`if (bitRead(PORTE,4) == LOW) { do }`
 	
`if (bitRead(PORTE,4 {`
`}`
	`Hvis bit 4 i Port E er lav, så`
	
	`Hvis læste bit er sand, så`

`bitRead(2,1);` `bitRead(4,1);`	`Giver 1 fordi 2 2 er binær 10` `Giver 0.`

`byte b = 0x55;`
`for (int i = 0; i < 8; i++)`
`{`
`Serial.println(bitRead(b, i));`
`}`	

BitRead eksempel Sæt en pin / portbit høj eller lav

// Led0 til 7 mangler at blive defineret

```
void loop()
{
    for (int i=0; i<256; i++)
    {
        digitalWrite(led0, bitRead(i, 0));
        digitalWrite(led1, bitRead(i, 1));
        digitalWrite(led2, bitRead(i, 2));
        digitalWrite(led3, bitRead(i, 3));
        digitalWrite(led4, bitRead(i, 4));
        digitalWrite(led5, bitRead(i, 5));
        digitalWrite(led6, bitRead(i, 6));
        digitalWrite(led7, bitRead(i, 7));
        delay(time);
    }
}
```

bit(x) **bit(bitPosition)**	

`PORTD |= bit(4);`
`PORTD ^= bit(4);`
	`I PORTD Or-es Bit4 med 1`
	`Bit 4 XOR-es med 1`
	`// XOR, => Toggle`

`#define LEDPIN 4`
 	
`void setup() {`
`  DDRD |= bit(LEDPIN);`
`}`
 	
`void loop() {`
`  PORTD ^= bit(LEDPIN);`
`  Delay(100);`
`}`
	`bit(0) er 1, bit(1) er 2, bit(2) er 4, osv. 3`

Toggle Pin eller bit	
`if (bitRead(X, 4) == 0) bitSet(X, 4);`	`// toggle bit 4 i variablen x` `// toggle bit 4 i variablen x`

[3] https://www.inkling.com/read/arduino-cookbook-michael-margolis-2nd/chapter-3/recipe-3-12

`else bitClear(X, 4);`	
`X = X ^ bit(4);` `X ^= bit(4);`	`// flip bit 4 i x` `// flip bit 4 i x`

Toggle Pin-eksempel -

```
#define LEDPIN 4                // toggle port bit

void setup () {
  pinMode(LEDPIN, OUTPUT)
}

byte onOff;

void loop () {
  if (onOff == 0)
     onOff = 1;
  else
     onOff = 0;
  digitalWrite(LEDPIN, onOff);  // The Arduino way
  delay(100);
}
```

PINx [4]	
`PIND = bit(4);` `PIND = bit(4);`	`Flip bit 4 i port D` `Set igen, = clear !!`

Bit test: Teste om bit er høj eller lav

If bit_is_set / clear	

`if(bit_is_set(TIFR1,OCF1A)) {`
`  flag = 1; Do something`
`}`
	Tjekker om bit OCF1A i register TIFR1 er sat!

`if(bit_is_clear(TIFR1,OCF1A)) {`	Tjekker om register, bit er 0.

[4] http://jeelabs.org/2010/08/27/flippin%E2%80%99-bits/

`    flag = 1; Do something` `}`	

`// Send a byte over the IR LED`
`void send_byte(int bits) {`
`  for (int i = 7; i >= 0; i--)`
`  {`
`    if (BIT_IS_SET(i, bits)) {`
`      WriteOne();Do something`
`    } else {`
`      WriteZero();Do something else`
`    }`
`  }`
`}`
	Eksempel på subrutine.
	
	// Kilde: # [5]

`val = bit_is_clear(PINB, BUTTON_PIN)`
 	
`buttonState = bit_is_clear(PINB, BUTTON_PIN);`
 	
`if(bit_is_clear(PIND,0))`
`{`
`    //  hvis pin D0 er lav så do …!`
`}`
	Tjekker om register, bit er 0.
	
	`read input value and store it in val`
	
	`Tjek om Button på PortD.0 er trykket, dvs. lav`

`if (PINB == 0x01) {`
`      do something;`
`    }`
 	
`if (PINC & 0x01) {`
`      do something;`
`    }`
	Tjek om PortB er lig 0b00000001

Definering af værdier i en variabel	Tal-typen skal defineres – evt. på forhånd.

`Tal = 0b10010100;`
`Tal =0x94;`
`Tal = 148;`
	`Binær`
	`Hex`
	`Decimal`

`Tal = bit(3);`	`// Bit 3 i tallet er sat, = 8`

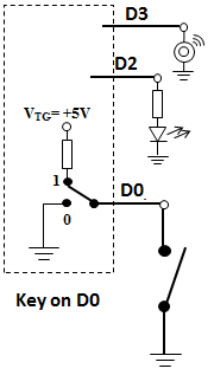
[5] http://adrianlombard.wordpress.com/2009/01/18/arduino-irobot-infrared-communication/

	`// ( Bit 0 til 7 fra højre`

`Tal = (1 << n);`
 	
`Tal = ((1<<7) | (1<<4) | (1<<2));`
	`// Tal = værdi, der kun har et 1 på bit n's plads.`

`byte a = bit(0);`
 	
`byte b = bit(2);  // 4 i decimal` [6]
	`Declare variable "a" and assigns a number with the bit 0 set to 1, 00000001 (1 in decimal)`
	`(bit can be from 0 to 7)`

Intern Pull Up på input-pins:



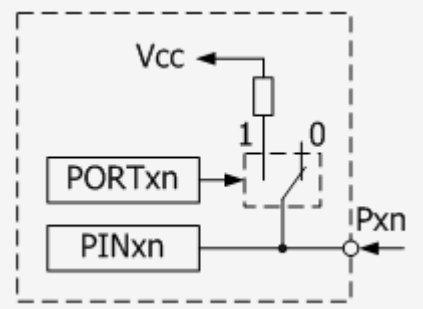
D0 = Arduino-Pin 0
D2 = Arduino-Pin 2
D3 = Arduino-Pin 3

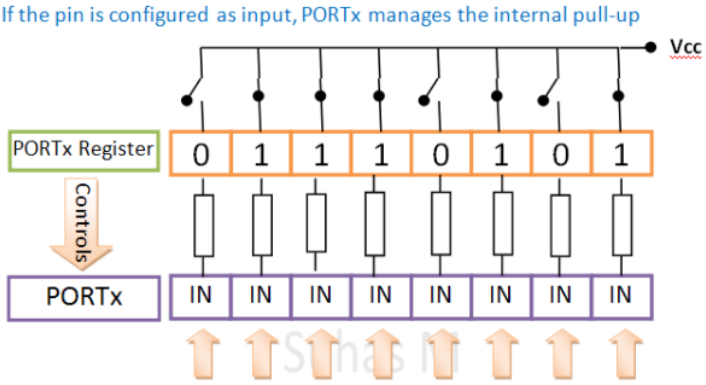
Internt kan man forbinde inputpins til en 20 K Pullup-modstand.

```
DDRD |= B00001100;      // D2 og D3 er OUTPUTs
PORTD |= B00000011;     // turn on pull-ups for D0 og D1


pinMode(pin, INPUT);      // set pin to input
digitalWrite(pin, HIGH);  // turn on pullup resistors

pinMode(2, INPUT_PULLUP);  // Input and pullup!!

PORTD = 0b00001111;      // Pull-ups enabled in the pins
0,1,2
                         // and 3 and pull-ups disabled in
                         // pins 4,5,6 and 7
```

Kilde # [7]

Her et billede af pull-up modstande for pins defineret som input.

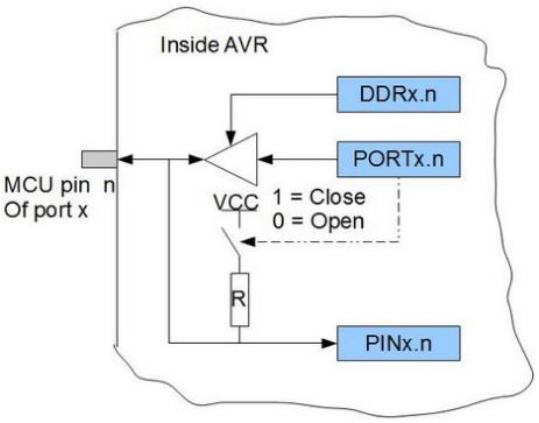




[6] http://wiring.org.co/reference/bit_.html
[7] http://hekilledmywire.wordpress.com/2011/02/23/direct-port-manipulation-using-the-digital-ports-tutorial-part-3/

Kilde # [8]

Her endnu et billede på, hvordan man kontrollerer pins vha. registre.



http://www.embedds.com/controlling-avr-io-ports-with-avr-gcc/

Switch case:

```
pins = PINB & 0x7;              // read pins 8/9/10
switch (pins) {
   case 0:
                                // no switches set
     break;
   case 1:
     switch1_func();
     break;
   case 2:
     switch2_func();
     break;
   case 3:                      // switches one and two.
     switch1_func();
     switch2_func();
   case 4:                      // switch 3 makes 1&2 into "don't care"
   case 5:
   case 6:
   case 7:
     switch3_func();
     break;
   }
}
```

Random

[8] https://iamsuhasm.wordpress.com/tutsproj/avr-gcc-tutorial/

```
void loop(){
  int i=random(4);
  digitalWrite(ledPin[i], HIGH);
  while(digitalRead(buttonPin[i]) == LOW)
  { //do nothing
  }
  digitalWrite(ledPin[i], LOW);
}
```

Random fra 0 til 3

randNumber = random(300);

random number from 0 to 299

randNumber = random(10, 20);
from 10 to 19

Se: https://www.arduino.cc/en/Reference/Random

_BV macroen:

Man kan angive en enkelt pin ved at bruge en indbygget makro, _BV. De øvrige bits efterlades uændret.
Kommandoen `_BV(x)` virker som `(1 << x)`

`_BV()` (_BV står for for Bit-Value)

```
PORTD |= _BV(3);
PORTD &= ~_BV(3);
```
pin, bit 3 i port D sættes high
Pin Low.

```
PORTC |= _BV(0);
```
Set bit 0 only.

```
PORTC &= ~(_BV(1));
```
Clear bit 1 only.

```
PORTC ^= _BV(7);
```
Toggle bit 7 only.

```
PORTC |= (_BV(0) | _BV(2);
```
Set bits 0 og 2

Boolsk algebra Se fx: The Bitmath Tutorial in the Playground

Bitwise AND

```
int a =  92;    // i binær: 0000000001011100
int b = 101;    // i binær: 0000000001100101
int c = a & b;  // 0000000001000100, eller 68 i decimal.
```

En af de mest brugte funktioner af bitwise AND er at udvælge bestemte bit fra et tal. Det kaldes ofte for ” Masking”.

Eksempel. Man ønsker at manipulere mindst betydende bit i variablen x, og gemme resultatet i variablen y. Det kan gøres med følgende kode:

```
int x = 5;       // binary: 101
int y = x & 1;   // now y == 1
x = 4;           // binary: 100
y = x & 1;       // now y == 0

PORTC &= 0x01;                  // Bitwise "And" alle bit i PortC
                                // med 01h


PORTC &= ~0x01;                 // Clear bit 0 only.
                                // ~ betyder inverter, derfor "And-es "
                                //  portC med 0b11111110
```

Bitwise OR

Bitwise OR bruges ofte til at sikre sig, at en given bit er sat til 1.

|

```
int a =  92;    // in binary: 0000000001011100
int b = 101;    // in binary: 0000000001100101
int c = a | b;  // 0000000001111101, or 125 in decimal.

b = a | 1;      // laveste bit i variablen a sættes til 1.

PORTC |= 0x01;          // Clear bit 0 only.
PORTC |= 0x80;          // Set bit 7 only.
```

Bitwise XOR

En anden måde at se på bitwise XOR er, at hver bit i resultatet er 1 hvis input-bittene er forskellige, og 0 hvis de er ens.

^

EX-OR operatoren ^ bruges ofte til at toggle nogle af bittene i et udtryk

```
int x = 12;     // binary: 1100
int y = 10;     // binary: 1010
int z = x ^ y;  // binary: 0110, or decimal 6

y = x ^ 1;      // toggle the lowest bit in x
PORTC ^= 0x01;          // Toggle bit 0 only.
```

Bitwise NOT

Bitwise NOT operatoren er ~. Bitwise NOT ændrer hver bit til det modsatte

~

```
int a = 103;    // binary:  0000000001100111
int b = ~a;     // binary:  1111111110011000 = -104
```

Bemærk: Det negative tal -104 er et resultat af, at man i et heltal, Integer bruger mest betydende bit som sign bit, (fortegnsbit). Hvis det er 1, er tallet negativ. Denne måde at kode positive og negative tal på hedder two's complement.

Bemærk, at for en given heltal x, ~x er det same som -x-1.

Bit Shift Operators

Der er 2 bit shift operatorer i C++: left shift operator << og right shift operator >>.

Disse operatorer bevirker, at alle bit i den venstre operand skiftes til venstre eller til højre et antal gange, specificeret i den højre operand.

```
int a = 5;        // binary: 0000000000000101
int b = a << 3;   // binary: 0000000000101000, = 40 i dec.
int c = b >> 3;   // binary: 0000000000000101, = 5 igen.
```

Når man skifter en værdi x , y gange, (x << y), vil de bit, der ryger ud over venstre kant tabes.

```
int a = 5;        // binary: 0000000000000101
int b = a << 14;  // binary: 0100000000000000 – den første 1 i 101 forsvinder.
```

Hvis man skifter x til højre med y bit (x >> y), og den mest betydende bit i x er 1, afhænger resultatet af hvilken data-type, x er. Hvis det er af typen int, er den højeste bit fortegnsbit, sign bit, og bestemmer om x er negativ eller positiv. I dette tilfælde kopieres signbittet ind i de lavere bit.

```
int x = -16;     // binary: 1111111111110000
int y = x >> 3;  // binary: 1111111111111110
```

Dette kaldes sign extension, og er ofte ikke hvad man ønsker.

I stedet kan man bruge typen unsigned int .

```
int x = -16;               // binary: 1111111111110000
int y = unsigned(x) >> 3;  // binary: 0001111111111110
```

```
int x = 1;      // binary: 0000000000000001
x <<= 3;        // binary: 0000000000001000
x |= 3;         // binary: 0000000000001011 - because 3 is 11 in binary
x &= 1;         // binary: 0000000000000001
x ^= 4;         // binary: 0000000000000101 - toggle using bit mask 100
```

```
x ^= 4;          // binary: 0000000000000001 - toggle with mask 100 again
```

Der er ingen kortere måde at håndtere bitwise NOT operator ~;
Her er det nødvendig at skrive:

```
x = ~x;          // toggle all bits in x and store back in x
```

Bitwise operators versus boolean operators

Bitwise AND operatoren & er ikke det samme som boolsk AND.

Bitwise & opererer uafhængig på de enkelte bit i operanden. Herimod konverterer && begge operanter til en boolsk værdi, (true==1 eller false==0), og returnerer derefter enten et enkelt true eller false værdi.

Derimod er 4 && 2 == true, og true er numerisk lig 1. Dette er fordi 4 ikke er 0, og 2 ikke er nul, så begge er boolsk lig true.

Logisk AND, &&

Læs to pins, og udfør hvis begge er høje.

```
if (digitalRead(2) == HIGH  &&
digitalRead(3) == HIGH) {
  Do Something ...
}
```

Logisk OR, ||

Sandt hvis enten x eller y er sand.

```
if (x > 0 || y > 0) {
  // ...
}
```

Logisk NOT, !

Sand hvis x er falsk, dvs. hvis x er 0.

```
if (!x) {
  // ...
}
```

Sammenlignings-operatorer:

En oversigt over, hvordan de forskellige test-operatorer skrives.

```
x == y (x is equal to y)
x != y (x is not equal to y)
x <  y (x is less than y)
x >  y (x is greater than y)
x <= y (x is less than or equal to y)
x >= y (x is greater than or equal to y)
```

Større end, mindre end, >, <

If-test

```
if (someVariable > 50)
{
  // do something here
}
```

Test størrelsen af en variabel.
Hvis testen er sand, udføres programmet mellem { og }

```
if (x >= 12) digitalWrite(LEDpin, 1);
```

Her er programmet kun på 1 linje.

Flere kode-eksempler

```
int ledPin = 13;
byte portD_HIGH = B11111000;
byte portD_LOW =  B00000000;

void setup()
{
  pinMode(ledPin, OUTPUT);
  DDRD = B11111100;
}

void loop()
{
  for(int i=0; i<10; i++){
    digitalWrite(ledPin, HIGH);
    PORTD = portD_HIGH;
    delay(i);
    digitalWrite(ledPin, LOW);
    PORTD = portD_LOW;
    delay(10-i);
  }
  for(int i=10; i>0; i--){
    digitalWrite(ledPin, HIGH);
    PORTD = portD_HIGH;
    delay(i);
    digitalWrite(ledPin, LOW);
    PORTD = portD_LOW;
    delay(10-i);
  }

}
```

http://www.instructables.com/id/Beginning-Arduino-Ports-Pins-and-Programming/?ALLSTEPS

Værdi af indbyggede konstanter

```
#define CS10    0
#define CS11    1
#define CS12    2
#define WGM12   3
#define WGM13   4
#define ICES1   6
#define ICNC1   7
```

Kilder:
http://tronixstuff.com/2011/10/22/tutorial-arduino-port-manipulation/
http://www.hobbytronics.co.uk/arduino-tutorial4-cylon
http://dubworks.blogspot.dk/2012/06/intra-to-port-manipulation.html
http://tronixstuff.wordpress.com/2011/10/22/tutorial-arduino-port-manipulation/
http://arduino.cc/forum/index.php?topic=123697.0
http://forum.arduino.cc/index.php/topic,44771.0.html
http://arduino.cc/forum/index.php?topic=90239.0