#define – smart hjælp til Debugging af kode

Når der skrives kode, er det smart, at sende variabel-værdier til debug-vinduet på PC-en, for fx at kontrollere, om et resultat blev, som forudset.

Men i det endelige program er der ikke nødvendigvis tilsluttet en PC med debugvindue. Derfor skal eller bør den del af kildeteksten, der bruges til debugging - jo fjernes inden den fegentlig ungerende kode uploades til microcontrolleren.

Men vha. et trick med funktionerne ”**#define** ” **#if**, **#ifdef** og **#endif** kan man lave betinget compilering.

Dvs. at ens kildetekst bliver oversat til maskinsprog ud fra nogle betingelser. På den måde kan man meget let styre hvad compileren laver af kode i udviklingsfasen – og i den endelige version.

Det kan kaldes betinget compilering, på engelsk: ” Conditional Compiling ”

Det kan styres vha. funktionen **#define**

Men #define kan bruges til flere ting:

Omdøbning af funktioner

Man kan bruge #define til at definere et ord til at betyde noget andet.

Det skal ske i sektionen før setup()

Fx:

```
#define udskriv  Serial.print (x)
```

Dvs. at man så i stedet for Serial.print(x) blot kan skrive

```
udskriv;
```

Et par flere eksempler:

```
#define PRINTDEC(x) Serial.print (x, DEC)
#define PRINTLN(x)  Serial.println (x)
#define pin13_on  bitSet(PORTB,5)
#define pin13_off  bitClear(PORTB,5)

#define myLED 8
```

Senere kan man så skrive:

```
digitalWrite ( myLED, 1);
```

Her er en lille testkode

```
/* Test af #define
Valle / 19/5-2022
*/

#define pin13_on bitSet(PORTB, 5)
#define pin13_off bitClear(PORTB, 5)

void setup() {
pinMode (13, OUTPUT);
}

void loop() {
pin13_on;
delay(1000);
pin13_off;
delay(1000);
}
```

Brug af #define ved debugging:

Her er der 2 muligheder, **#if** og **#ifdef**

1: **#if:**

Før setup() kan man fx definere et ord, her ordet DEBUG , - til at være fx 0 eller 1

```
#define DEBUG 1
```

I setup() kan man skrive:

```
#if DEBUG > 1  // Compileres kun hvis sand

Serial.begin(9600);

#endif
```

Og i loop()

```
#if DEBUG == 1   // Compileres kun hvis sand

Serial.println("hej fra #1");
```

```
#endif
```

Dvs. Compileren tester om ordet DEBUG er lig værdien 1. Hvis sandt, medtager compileren kode til serial kommunikation og Arduinoen vil sende besked til debugvinduet.

Den måde, man så kan styre om debug-koden skal med er at ændre #define DEBUG 1 til #define DEBUG 0

Det vil jo også betyde, at man fx andre steder i koden tjekke om programmet kommer "forbi" ved at tildele DEBUG værdien 2, 3 osv.

Man kan også simpelt bruge testen #if DEBUG.

```
#if DEBUG  // sand hvis DEBUG er alt andet end 0.
```

2: **#ifdef**

Der er også en anden mulighed.

Når man skriver #define DEBUG 0 eller #define DEBUG 1, bliver ordet DEBUG defineret. Altså kendes ordet DEBUG af compileren.

Før setup()

```
#define DEBUG 1    // Ordet DEBUG er defineret, og har en værdi !
```

I koden tester compileren så om DEBUG er defineret vha. **#ifdef**

I setup()

```
#ifdef DEBUG  // Sandt hvis ordet DEBUG er defineret,
Serial.begin(9600);
#endif
```

Og i loop()

```
#ifdef DEBUG
Serial.print("Hej");
#endif
```

Metoden her til at styre om den serielle del skal medtages i kompileringen er ved at udkommentere linjen

```
// #define DEBUG 1
```

Så er den jo ikke ” defineret ” og ikke kendt af kompileren.

Eksempel:

```
// Før setup():

#define DEBUG 1  //  Udkommenter denne linje, hvis ikke debug-mode

// I setup():

#ifdef DEBUG   // eller brug: #if DEBUG == 1
   Serial.Begin(9600);
   // Do stuff
#endif

// I loop() kan fx følgende placeres:

#ifdef DEBUG
   Serial.print("test ");
   Serial.print(i);
   // Do more stuff if DEBUG == 1
#endif
```

Samlet kode med flere eksempler:

```
// test af #define
// Valle, 17/5-2022

#define DEBUG 2

void setup() {
  #ifdef DEBUG
  Serial.begin(9600);
  #endif
}

void loop() {

  #if DEBUG == 1
   Serial.println("hej DEBUG = 1");
#endif
//
```

```
#if DEBUG == 2
   Serial.println("hej DEBUG = 2");
#endif

#if DEBUG == 3
   Serial.println("hej DEBUG = 3");
#endif

#if DEBUG > 0
Serial.println("hej fra loop");
#endif
delay(1000);
}
```

Strukturen kan også udvides med #elif, #ifndef (*if not def*)

#if tests if a condition is true. #else provides an alternative case for an #if, in case it is not true.

The #if condition is evaluated only at compile time. The "if" is evaluated at run time.

Basically, the #if construct allows you to control/change which code the compiler "sees".

Ps:

En #define må vist ikke være den første kodelinje i et program:

Derfor kan man definere en dummy-var i den første linje

```
byte nonsense_var = 0;  //this line solves everything!
```

Eller man kan vist bare sætte et semikolon i 1. kodelinje. ??

Kilde:
https://www.interviewsansar.com/difference-between-preprocessor-if-and-ifdef/
https://forum.arduino.cc/t/toggling-debug-code/47041/5
https://stackoverflow.com/questions/28177544/defining-variables-within-if-else-endif-in-arduino-ide