Arduino Programming Cheat Sheet

Primäre Quelle: Arduino Language Reference
https://www.arduino.cc/reference/en/

Struktur & Ablauf

Grundstruktur

```
void setup() {
} // Einmalig bei Sketchstart

void loop() {
} // Endlos wiederholte Ausführung
```

Kontrollstrukturen

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Schleife beenden
continue;  // Zu nächster Iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x; // x muss Rückgabe ensprechen
return;   // Für void-Rückgabe
```

Definition von Funktionen

```
<ret.-Typ> <Name>(<Parameter>) { ... }
z.B. int double(int x) {return x*2;}
```

Operatoren

Standardoperatoren

=	zuweisen		
+	addieren	-	subtrahieren
*	multiplizieren	/	teilen
%	modulo		
==	gleich	!=	ungleich
<	kleiner	>	größer
<=	kleinergleich		
>=	größergleich		
&&	und	||	oder
!	nicht		

Verkürzte Zuweisungsoperatoren

++	inkrementieren um 1
--	dekrementieren um 1
+=	erhöhen um..
-=	vermindern um..
*=	multiplizieren mit..
/=	teilen durch..
&=	bitweises 'und'
|=	bitweises 'oder'

Bitweise Operatoren

&	bitweises 'und'	|	bitweises oder
^	bitweises 'xor'	~	bitweises nicht
<<	linksschieben	>>	rechtsschieben

Pointerzugriff

&	Referenz: Pointer erhalten
*	Dereferenz: Pointer folgen

Variablen, Arrays und Datentypen

Datentypen

bool	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	identisch mit float
void	als Rückgabetyp: keine Rückgabe

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numerische Konstanten

123	dezimal
0b01111011	binär
0173	oktal - base 8
0x7B	hexadezimal - base 16
123**U**	Vorzeuichenlos
123**L**	Datentyp 'long'
123**UL**	Datentyp 'unsigned long'
123.**0**	Datentyp 'float'
1.23**e**6	1.23*10^6 = 1230000

Modifikatoren

static	persistent zw. Aufrufen
volatile	im RAM (z.B. für ISR)
const	read-only
PROGMEM	im Flash-Speicher

Arrays

```
byte myPins[] = {2, 4, 8, 3, 6};
int myInts[6];   // Array mit 6 Zahlen
myInts[0] = 42;  // Zuweisung 1.Zahl
                 // in myInts
myInts[5] = 12;  // Zuweisung 6.Zahl
                 // (Index von 0..5!)
```

Arduino Programming Cheat Sheet

Primäre Quelle: Arduino Language Reference
https://www.arduino.cc/reference/en/

Standardfunktionen

Pin Input/Output

```
Digital I/O
// pins 0-13 und A0-A5
pinMode(pin, mode)
// mode: INPUT, OUTPUT
   oder INPUT_PULLUP
int digitalRead(pin)
digitalWrite(pin,{HIGH|LOW})

Analog In - pins A0-A5
int analogRead(pin)
analogReference(
{DEFAULT|INTERNAL|EXTERNAL})

PWM Out
// nur pins 3,5,6,9,10,11
analogWrite(pin, value)
// value: 0-255
```

Zusätzliche I/O-Befehle

```
tone(pin, freq_Hz, dauer)
noTone(pin)
shiftOut(dataPin, clockPin,
  {MSBFIRST|LSBFIRST},val)
shiftIn(dataPin, clockPin,
  {MSBFIRST|LSBFIRST})
unsigned long pulseIn(pin,
  {HIGH|LOW},[timeout_usec])
```

Zeiten und Verzögerungen

```
unsigned long millis()
// Überlauf nach 50 Tagen
unsigned long micros()
// Überlauf nach 70 Min.
delay(msec)
delayMicroseconds(usec)
```

Mathematische Befehle

```
min(x, y)  max(x, y) abs(x)
sin(rad)   cos(rad)  tan(rad)
sqrt(x)    pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH,
    toL, toH)
```

Zufallszahlen

```
randomSeed(seed) // long/int
long random(max) // 0 bis max-1
long random(min, max)
```

Bits und Bytes

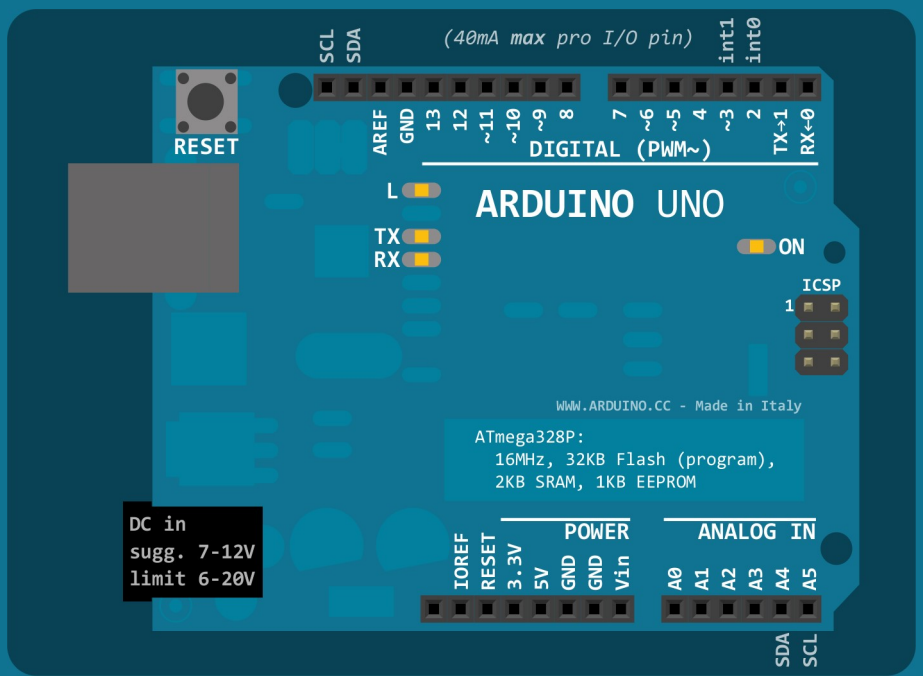
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn) // bitn:0=LSB 7=MSB
```

Typumwandlung

```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

Externe Interrupts

```
attachInterrupt(interpt, func,
 {LOW|CHANGE|RISING|FALLING})
detachInterrupt(interpt)
interrupts()
noInterrupts()
```



Bibliotheken

Serial - Kommunikation mit Terminal

```
begin(long speed)  // Bis zu 115200
end()
int available()  // #bytes verfügbar
int read()   // -1 falls keine Daten
int peek()   // Lesen ohne Entfernen
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Event für Daten
```

SoftwareSerial.h - Komm. auf bel. Pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Bis zu 115200
listen()       // Nur 1 Zuhörer
isListening()  // gleichzeitig mgl.
read, peek, print, println, write
  // Identisch zu Oben
```

EEPROM.h - Zugriff auf Eepromspeicher

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Zugriff als array
```

Servo.h - *Servos steuern*

```
attach(pin, [min_usec, max_usec])
write(angle)  // 0° bis 180°
writeMicroseconds(uS)
   // 1000-2000; 1500 ist Mitte
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - *I²C Kommuniktion*

```
begin()     // Master verbinden
begin(addr) // Slave verb. mit addr
requestFrom(address, count)
beginTransmission(addr) // Schritt 1
send(byte)              // Schritt 2
send(char * string)
send(byte * data, size)
endTransmission()       // Schritt 3
int available() // #bytes verfügbar
byte receive()  // Nächst. Byte lesen
onReceive(handler)
onRequest(handler)
```


Übersetzung / Redesign
Werner Mager
version: 2024-05-30
https://github.com/wmager/Arduino-Cheat-Sheet-De

Basierend auf:
- Engl. Original: Mark Liffiton
- Original 2: Gavin Smith
- SVG-Version: Frederic Dufourg
- Arduino board drawing: Fritzing.org