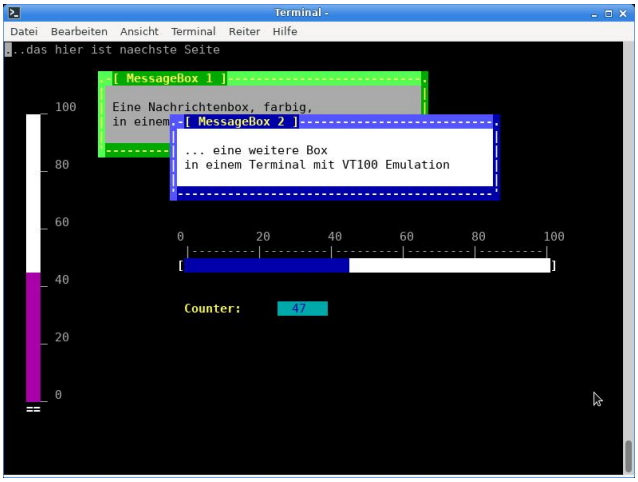
Arduino: extended_serial.h / extended_serial.cpp



extended_serial.h / extended_serial.cpp ist eine Arduino-Library zur Erweiterung des originalen HardwareSerial von Arduino.

Diese Library stellt 2 neue Klassen zur Verfügung:

- extendedSerial
- AnsiOut

In diesem Dokument gilt im Folgenden:

AsciiZ : nullterminierter ASCII-String ('\0' abgeschlossen)

Klasse extendedSerial

extendedSerial ist eine Arduino-Klasse für serielle Ausgaben, die erweiterte Methoden im Vergleich zu HardwareSerial bietet. Mit extendedSerial werden u.a ESC-Sequenzen für farbige Ausgaben sowie zur Cursorpositionierung gesendet, die auf einer ANSI-Konsole (VT100 kompatibel) wie bspw. PuTTY interpretiert werden.

Für eine, im Vergleich zu HardwareSerial, vereinfachte und zusammengefasste Ausgabe wurde ein sehr abgespecktes, aber auch sehr ressourcensparendes printf implementiert (dieses abgespeckte printf ist / war der Hauptgrund für die Entstehung von extendedSerial).

Grundsätzlich werden, wie auch bei HardwareSerial, Steuerzeichen (Ascii-Code < 32) akzeptiert, die auch auf Nicht-ANSI-Terminals funktionieren.

Die Klasse extendedSerial ist plattformunabhängig, sodass sie auf allen Plattformen funktionsfähig ist, die ein HardwareSerial implementieren.

Des weiteren erbt extendedSerial die Stream-Klasse von Arduino und so sind gewohnte Funktionen wie print und println ebenfalls verfügbar.

extendedSerial wurde auf AVR-, STM- und CH32V003 Systemen getestet.

Anmerkung zu CH32V003:

> Der Arduino-Core von CH32V003 ist in einigen Teilen noch fehlerbehaftet und im Falle von HardwareSerial sind nicht alle Funktionen komplett implementiert. Es fehlen sämtliche Lesefunktionen, sodass alle Eingabefunktionen auf dem seriellen Port über HardwareSerial leider nicht funktionieren. Hier wird auf eine speziell für den CH32V003 verfügbare Library „v003_uart“ verwiesen.

Im Folgenden werden Konstante, Variable, Konstruktoren und Methoden von extendedSerial beschrieben.

Konstante

```
constexpr uint8_t black         = 0;
constexpr uint8_t blue          = 1;
constexpr uint8_t green         = 2;
constexpr uint8_t cyan          = 3;
constexpr uint8_t red           = 4;
constexpr uint8_t magenta       = 5;
constexpr uint8_t brown         = 6;
constexpr uint8_t grey          = 7;
constexpr uint8_t darkgrey      = 8;
constexpr uint8_t lightblue     = 9;
constexpr uint8_t lightgreen    = 10;
constexpr uint8_t lightcyan     = 11;
constexpr uint8_t lightred      = 12;
constexpr uint8_t lightmagenta  = 13;
constexpr uint8_t yellow        = 14;
constexpr uint8_t white         = 15;
```

Variable

```
char printfkomma;
```

bestimmt Nachkommateil bei printf-Platzhalter %k

```
uint8_t textattr;
```

beinhaltet das Farbattribut (Hintergrund- / Vordergrundfarbe), mit der das nächste Zeichen ausgegeben wird.

Konstruktor extendedSerial

```
extendedSerial(HardwareSerial &s);
```

benötigt als Argument bei der Instanziierung den seriellen Port, über den die Ausgaben erfolgen sollen.

Hinweis:

Bei den meisten ATmega und CH32V003 ist nur "Serial" verfügbar, während bei einigen STM32-Varianten zudem Serial1 oder auch Serial2 vorhanden sind.

Beispiel für den Konstruktor:

```
// Objektinstanziierung
extendedSerial    myout(Serial);
```

Methoden extendedSerial

begin

```
void begin(unsigned long baud);
```

Initialisiert den seriellen Port mit der Baudrate, über den die Ausgaben erfolgen.

Beispiel:

```
myout.begin(115200);
```

my_putchar

```
void my_putchar(char ch);
```

Sendet ein einzelnes Zeichen über den seriellen Port, wird u.a. von printf verwendet und ist somit die zentrale Methode von extendedSerial.

Beispiel:

```
myout.my_putchar('R');
```

clrscr

```
void clrscr(void);
```

Löscht den Bildschirminhalt der ANSI-Konsole mit der durch settextattr oder setbkcolor gesetzten Hintergrundfarbe:

Beispiel:

```
myout.setbkcolor(blue);
myout.clrscr();            // es erscheint ein blauer, leerer Bildschirm
```

gotoxy

```
void gotoxy(uint8_t x, uint8_t y);
```

Positioniert den Textcursor, an dem die nächste Konsolenausgabe erfolgen wird.

Beispiel:

```
myout.gotoxy(5,4);
```

settextcolor

```
void settextcolor(uint8_t col);
```

Setzt die Textvordergrundfarbe, mit der die nächste Textausgabe erfolgen wird.

Beispiel:

```
myout.setbkcolor(blue);
myout.settextcolor(yellow);
myout.printf(" Text mit gelber Schrift und blauem Hintergrund ");
```

setbkcolor

```
void setbkcolor(uint8_t col);
```

Setzt die Farbe des Texthintergrundes, mit der die naechste Textausgabe
erfolgen wird.

Beispiel siehe settextcolor.

settextattr

```
void settextattr(uint8_t attr);
```

Mit settextattr lassen sich Text- und Hintergrundfarbe gleichzeitig setzen. Hierbei muss beachtet werden, dass das obere Nibble (Datenbits D4 bis D7) der Hintergrundfarbe, das untere Nibble (Datenbits D0 bis D3) der Textfarbe entspricht.

Somit gibt die höherwertige Ziffer eines **`unsigned char`** die Texthintergrundfarbe, die niederwertige Ziffer die Textfarbe an.

Mit **`myout.settextattr(0x2e);`** wird die Hintergrundfarbe 2 (laut Farbkonstanten ist das grün) und die Vordergrundfarbe 14 (entspricht hexadezimal "e", laut Farbkonstanten ist das gelb) für die nächste Zeichenausgabe gesetzt.

Beispiel:

```
myout.settextattr(0x2e);
myout.printf(" Hallo Welt in gelb-gruen ");
```

printf

```
void printf(const char *s,...);
```

Das hier implementierte printf ist eine sehr reduzierte Variante der Standardfunktion printf, dessen Zweck es ist, äußerst ressourcensparend mit dem Flashspeicher umzugehen. printf wurde zu diesem Zweck ein neuer Platzhalter %k hinzugefügt, mittels diesem es möglich ist, Pseudokommazahlen auszugeben.

Hinweis:

Gleitkommazahlen (float / double) werden NICHT unterstützt.

Platzhalter	Datentyp und Darstellung
%d	Integer als dezimale Zahl ausgeben
%k	(Pseudo-Kommazahl) Integerwert mit eingefügtem Kommapunkt ausgeben. Der Wert in Variable printfkomma bestimmt, an welcher Position des Integerwertes ein Dezimalpunkt eingefügt wird.
%c	Ausgabe eines char als Ascii-Zeichen
%s	Zeichenkette (String) ausgeben
%x	Integer als hexadezimale Zahl ausgeben
%%	Ausgabe des Prozentzeichens

Beispiele:

```
myout.printfkomma = 2;
myout.printf("\n\r Ausgabe einer Pseudokommazahl: %k", 2035);
```

angezeigt wird:

20.35

```
myout.printf("\n\r Ascii-Zeichen %c hat den Wert %xh = %dd", 'a', 'a','a');
```

angezeigt wird:

Ascii-Zeichen a hat den Wert 61h = 97d

Klasse AnsiOut

AnsiOut ist eine Klasse für optisch ansprechendere und leichter zu realisierende Ausgaben auf einer ANSI-Konsole. Sie beinhaltet Methoden zur koordinatengetreuen Ausgabe von einzelnen Zeichen sowie vertikal oder linksbündig ausgerichteten Strings.

Zudem lassen sich Text- und Messageboxen sowie Fortschrittsbalken sehr einfach erzeugen.

Konstruktor AnsiOut

```
AnsiOut(extendedSerial &ext);
```

Benötigt als Argument &ext eine Referenz auf eine bestehende Instanz von extendedSerial, auf der Ausgaben zu erfolgen haben.

Beispiel:

```
extendedSerial myout(Serial);

AnsiOut        ansi(myout);
```

Strukturen und ihre Instanzen

Um einen einfachen Umgang bei der Verwendung von Text-, Messageboxen und Progressbars (Fortschrittsbalken) zu haben, wurden für Textboxen und Progressbars jeweils eigene Strukturen angelegt. Von diesen Strukturen erzeugte Instanzen beinhalten das Erscheinungsbild und Position einer Textbox oder eines Fortschrittbalkens.

Diese Strukturen sind: **txbox_t** und **progbar_t.**

Hieraus werden die Variablen tbox und progbar instanziiert.

```
txbox_t tbox;
progbar_t progbar;
```

txbox_t - Strukturmitglieder

```
typedef struct
{
  int x1, y1;             // linkes oberes Eck der Textbox / Messagebox
  int x2, y2;             // rechtes unteres Eck der Textbox / Messagebox
  uint8_t upborderattr;   // Farbattribut des oberen Rahmenteils
  uint8_t dwnborderattr;  // Farbattribut unterer Rahmenteil
  uint8_t txtfieldattr;   // Farbattribut inneres Textfeld
  uint8_t titelattr;      // Farbattribut Rahmentitels
  char *boxtitel;         // Zeiger auf einen AsciiZ des Rahmennamens
} txbox_t;
```

Beispiel siehe msgbox.

progbar_t - Strukturmitglieder

```
typedef struct
{
  uint8_t  x;           // x-Koordinate, ab der der Balken angezeigt wird
  uint8_t  y;           // y-Koordinate
  uint16_t anz;         // Anzahl Zeichen aus denen der Balken besteht
  uint16_t percentage;  // Anzuzeigende Prozentzahl
  uint8_t  barattr;     // Farbattribut des Balkens,
                        // Highnibble == Farbe des Balkens
                        // Lownibble == Farbe der nicht gezeichneten
                        // Prozentzahl des Balkens
  uint8_t  barendattr;  // Farbattribut der Begrenzungszeichen des Balkens
  uint8_t  drawboth;    // 1 : zeichnet erreichte und nicht erreichte
                        //     Prozentzahl
                        // 0 : nur erreichte Prozentzahl
  uint8_t markstep;     // Schrittweite, der Skaleneinteilung in Zeichen
  int16_t startw;       // Beschriftung: Startwert der Skala
  int16_t stepw;        // Schrittweite des Wertes fuer eine Markierung
  uint8_t scaleattr;    // Farbattribut der Skala
} progbar_t;
```

Beispiel siehe progressbar.

Methoden AnsiOut

putcharxy

```
void putcharxy(uint8_t x, uint8_t y, char ch);
```

putcharxy ist die zentrale Ausgabemethode von AnsiOut. Mit putcharxy kann ein beliebiges Zeichen an einer beliebigen Textkoordinate auf der ANSI-Konsole angezeigt werden.

Beispiel:

```
myout.settextattr(0x4e);     // gelbe Schrift auf rotem Grund setzen
ansi.putcharxy(14, 3, 'R');  // Buchstaben 'R' an Koordinate x,y = 14,3 ausgeben
```

vertikalout

```
void vertikalout(uint8_t x, uint8_t y, uint8_t col, uint8_t max, char *cptr);
```

Mit vertikalout kann ein AsciiZ-String vertikal ausgegeben werden. Bei der Definition von vertikalout wurde bewusst eine Steuerung der Methode über eine Strukturinstanz nicht implementiert, weil die Praxis gezeigt hat, dass der Umgang, selbst mit einer relativ langen Argumentliste, einfacher ist.

Die Bedeutung der Methodenargumente:

x,y	: Textkoordinate, ab der ein Text vertikal ausgegeben wird
col	: das Farbattribut, mit dem der Text ausgegeben wird
max	: die maximale Anzahl an Zeichen, die ausgegeben wird. Erreicht die Zeichenausgabe diesen maximalen Wert, wird die Ausgabefunktion auch dann beendet, wenn das Stringende noch nicht erreicht war
*cptr	: Zeiger auf den auszugebenden AsciiZ-String

Beispiel:

```
ansi.vertikalout(14,8, 0x4e, 30, " Hallo Welt ");
```

Ausgabe:
ein vertikal angezeigter "Hallo Welt"-Text mit gelber Schriftfarbe auf rotem Grund.

txtleftout

```
void txtleftout(int xorg, int yorg, char *s);
```

Mit **txtleftout** ist es möglich, einen AsciiZ-String, der innerhalb des Strings die Steuerzeichen **\n** und insbesondere **\r** verwendet, linksbündig auszugeben. Da linksbündig hier nicht die erste Stelle einer Zeile bedeutet, entspricht das faktisch einer Einrückung eines Textes.

Die Argumente von txtleftout sind:

xorg :	x-Koordinate die den linken Rand der Textausgabe bezeichnet
yorg :	y-Koordinate ab der die Textausgabe erfolgt
*s :	AsciiZ-String der ausgegeben wird.

Beispiel:

```
ansi.txtleftout(5,3, "\n\rDieses ist ein Text, der vom Textrand"
                     "\n\r5 Zeichen eingerueckt ist.\n\r"
                     "\n\rEs erfolgte eine Leerzeile, bevor dieser Text"
                     "\n\rhier ausgegeben wurde!");
```

txtbox

```
void txtbox(const txbox_t *box);
```

txtbox "zeichnet" eine leere Textausgabebox, die im Erscheinungsbild durch eine Referenz einer Instanz der Struktur txbox_t beschrieben ist.

Als Referenz kann die durch die Instanziierung eines Objekts erfolgte Variable tbox fungieren, aber ebenso auch jede andere dieser Struktur:

```
AnsiOut::txbox_t mybox2;
```

Beschreibung der Struktur txbox_t siehe "txbox_t - Strukturmitglieder"

Beispiel:

```
// Verwendung der bereits durch das Objekt bereitgestellten Variable tbox

char s[] = "[ Titel leeres Fenster ]";

ansi.tbox.x1 = 4;                 // Ausgabeposition x
ansi.tbox.y1 = 3;                 // Ausgabeposition y
ansi.tbox.x2 = 40;                // Endeposition x
ansi.tbox.y2 = 8;                 // Endeposition y
ansi.tbox.txtfieldattr = 0x70;    // Attribut des Fensters, grauer Hintergrund,
                                  // schwarze Schrift
ansi.tbox.upborderattr = 0xae;    // obere Rahmenteil, hellgruener Hintergrund,
                                  // gelbe Schrift
ansi.tbox.dwnborderattr = 0x2e;   // unterer Rahmenteil, dunkelgruener
                                  // Hintergrund, gelbe Schrift
ansi.tbox.boxtitel = s;           // Beschriftungstext im String s
ansi.tbox.titelattr = 0x2e;       // Farbattribut der Beschriftung

ansi.txtbox(&ansi.tbox);          // Fenster anzeigen
```

Hier wird ein "Fenster“ erzeugt, das einen aus Ascii-Zeichen bestehenden Rahmen zeichnet. Dieser Rahmen wird mit einem einheitlichen Farbattribut für den linken und oberen, sowie einem weiteren Farbattribut für den rechten und unteren Rahmenteil, gezeichnet. Das Fenster selbst hat wiederum ein eigenes Farbattribut. Zudem ist dieses Fenster mit dem Text "[Titel leeres Fenster]" beschriftet.

```
.-[ Titel leeres Fenster ]----------.
|                                    |
|                                    |
|                                    |
|                                    |
'------------------------------------'
```

msgbox

```
void msgbox(const txbox_t *box, const char *s);
```

msgbox "zeichnet" eine Nachrichtenausgabebox, die im Erscheinungsbild durch eine Referenz einer Instanz der Struktur txbox_t beschrieben ist und gibt in dieser Box einen AsciiZ-String aus.

Als Referenz kann die durch die Instanziierung eines Objekts erfolgte Variable tbox fungieren, aber ebenso auch jede andere dieser Struktur:

```
AnsiOut::txbox_t mybox2;
```

Beschreibung der Struktur txbox_t siehe "txbox_t - Strukturmitglieder"

Beispiel:

```
char s[] = "[ NTC 10k ]";

ansi.tbox.x1 = 4;                  // Ausgabeposition x
ansi.tbox.y1 = 2;                  // Ausgabeposition y
ansi.tbox.x2 = 44;                 // Endeposition x
ansi.tbox.y2 = 10;                 // Endeposition y
ansi.tbox.txtfieldattr = 0x70;     // Attribut des Fensters, grauer Hintergrund,
                                   // schwarze Schrift
ansi.tbox.upborderattr = 0xae;     // obere Rahmenteil, hellgruener Hintergrund,
                                   // gelbe Schrift
ansi.tbox.dwnborderattr = 0x2e;    // unterer Rahmenteil, dunkelgruener
                                   // Hintergrund, gelbe Schrift
ansi.tbox.boxtitel = s;            // Beschriftungstext im String s
ansi.tbox.titelattr = 0x2e;        // Farbattribut der Beschriftung

ansi.msgbox(&ansi.tbox, "\n\rArduino mit AnsiOut \n\r"
                        "230400 bd 8N1 \n\r"
                        "Temperatursensor NTC 10k\n\n\r"
                        "15.03.2026 by R. Seelig");
```

Hier wird ein "Fenster“ erzeugt, welches einen aus Ascii-Zeichen bestehenden Rahmen zeichnet. Dieser Rahmen wird mit einem einheitlichen Farbattribut für den linken und oberen, sowie einem weiteren Farbattribut für den rechten und unteren Rahmenteil, gezeichnet. Das Fenster selbst hat wiederum ein eigenes Farbattribut. Zudem ist dieses Fenster mit dem Text "[NTC 10k]" beschriftet.

Das Fenster selbst zeigt den folgenden Inhalt an:

```
.-[ NTC 10k ]--------------------------.
|                                      |
| Arduino mit AnsiOut                  |
| 230400 bd 8N1                        |
| Temperatursensor NTC 10k             |
|                                      |
| 15.03.2026 by R. Seelig              |
|                                      |
'--------------------------------------'
```

progressbar / vprogressbar

```
void progressbar(const progbar_t *pb);
void vprogressbar(const progbar_t *pb)
```

Die Funktion **progressbar** zeigt einen konfigurierbaren horizontalen Fortschrittsbalken an, **vprogressbar** einen vertikalen. Diese Progressbar kann auch als universelle Balkenanzeige (Bargraph) zur Darstellung von Messwerten, wie z. B. Temperatur, Luftdruck oder Spannung, verwendet werden.

Das Erscheinungsbild sowie die Ausgabeposition werden durch eine Referenz auf eine Instanz der Struktur progbar_t festgelegt.

Als Referenz kann die durch die Instanziierung eines Objekts bereitgestellte Variable progbar verwendet werden, aber ebenso auch jede andere dieser Struktur.

```
AnsiOut::progbar_t myprogbar2;
```

Beschreibung der Struktur siehe "progbar_t - Strukturmitglieder"

Grundsätzlich stellt der Balken einen Wert im Bereich von 0 bis 100 % dar, unabhängig davon, aus wie vielen Zeichen er besteht.

Der dargestellte Prozentwert muss vom Anwender entsprechend auf den gewünschten Wertebereich (z. B. Temperatur oder Spannung) umgerechnet werden.

Beispiel:

```
myout.gotoxy(1,1); myout.printf("Progressbar-Demo, gleich geht's weiter");

// Erscheinungsbild der Progressbar
ansi.progbar.x = 14;
ansi.progbar.y = 4;
ansi.progbar.anz = 51;
ansi.progbar.percentage = 0;
ansi.progbar.barattr = 0xf1;        // Balken Blau-Weiss
ansi.progbar.barendattr = 0x0f;     // Farbe der Begrenzung
ansi.progbar.drawboth = 1;          // komplette Progressbar zeichnen

for (int i = 0; i < 101; i++)
{
  myout.gotoxy(1,4);
  myout.printf("Time: %k s", i);
  ansi.progbar.percentage = i;
  ansi.progressbar(&ansi.progbar);
  delay(100);
}

myout.settextattr(0x07);
myout.gotoxy(1,6);
myout.printf("Hier geht's jetzt weiter...");
```

progressbar / vprogressbar

```
void progbarscala(const progbar_t *scale);
void vprogbarscala(const progbar_t *scale);
```

Mit progbarscala / vprogbarscala kann eine Progressbar beschriftet werden, wenn diese als universelle Balkenanzeige (Bargraph) genutzt werden soll.

progbarscala / vprogbarscala verwendet für das Erscheinungsbild und die Ausgabeposition dieselbe Referenz auf eine Instanz der Struktur progbar_t wie progressbar (Struktur progbar_t beinhaltet Elemente, die nicht von progbarscala, jedoch von progressbar ausgewertet werden).

Als Referenz kann die durch die Instanziierung eines Objekts bereitgestellte Variable progbar verwendet werden, aber ebenso auch jede andere dieser Struktur.

```
AnsiOut::progbar_t myprogbar2;
```

Beschreibung der Struktur siehe "progbar_t – Strukturmitglieder"

Beispiel (Bargraph zeigt Temperaturwert an):

```
int temp;
int scaletemp;

ansi.progbar.x = 18;
ansi.progbar.y = 4;
ansi.progbar.anz = 61;
ansi.progbar.percentage = 0;
ansi.progbar.barattr = 0xf4;       // Balken rot-weiss
ansi.progbar.barendattr = 0x0f;    // Farbe der Begrenzung
ansi.progbar.drawboth = 1;         // komplette Progressbar zeichnen
ansi.progbar.markstep = 10;        // Schrittweite der Skaleneinteilung
ansi.progbar.startw = -10;         // Startwert der Skala
ansi.progbar.stepw = 10;           // Schrittweite der Werte
ansi.progbar.scaleattr = 0x07;     // Farbattribut der Skala

temp= 238;                         // Temperaturvorgabewert entspricht 23.8 oC

myout.gotoxy(1, ansi.progbar.y);
myout.printf("Temp.: %k oC", temp);  // Temperaturwert anzeigen

// Skalierung: 23.8 oC auf einer Skala von -10.0 bis +50.0 entsprechen
// einem auf prozentualen Anteil 39.6%  (Skala zeigt Werte von 0..100% an)

scaletemp= map(temp, -100, 500, 0, 100);

ansi.progbar.percentage= scaletemp;
ansi.progressbar(&ansi.progbar);   // Bargraph anzeigen
ansi.progbarscala(&ansi.progbar);  // Skala anzeigen
```