# AVR-Terminal

## Einleitung

Die Applikation AVR-Terminal wurde entwickelt, um die Arbeit mit den ATMEL-Mikrocontrollern Atmega32 und Atmega328p in den ICT Modulen M121 und M242 zu unterstützen. Die Applikation übernimmt dabei folgende Aufgaben:

* Darstellung von Ausgaben in einer Mikrokontroller Anwendung: In vielen Fällen ist die Tracing die einzig brauchbare Möglichkeit eine Mikrokontroller-Anwendung zu debuggen. Was ist Tracing? In einer Windows-Konsole Applikation bedeutet tracen, dass wir in unser Programm Anweisungen einbauen, welche eine Zeichenkette auf der Konsole schreiben. So hinterlässt unsere Applikation eine Spur (Trace), welche wir nachverfolgen können. Anhand dieser Spur können wir nachvollziehen, ob unser Programm richtig funktioniert.

Nun hat ein Mikrokontroller aber keinen Bildschirm, auf welchen wir schreiben könnten. Der Mikrocontroller kann keinen Trace anzeigen. Wir müssen die Trace-Daten deshalb an ein Gerät schicken, welche das Tun kann. In unserem Fall benutzen wir die COM Schnittstelle, um die Daten an den PC zu schicken. Auf dem PC werden die Daten in der Applikation AVR-Terminal angezeigt. Mehr zum Thema «Tracen» ist im Abschnitt Tracing beschrieben.

* Senden von Benutzereingaben und Test-Stimuli an die Mikrokontroller Applikation: In einer Windows-Anwendung sind wir uns gewohnt, mit eine Anwendung zu interagieren. Wir können dazu Tastatur und Maus verwenden. Wir können einer Anwendung auch Parameter über die Kommandozeile mitgeben. Auch diesbezüglich sind wir mit unseren Mikrocontrollern eingeschränkt. Wir haben keine Eingabe-Geräte die wir benutzen könnten. Wir können der Mikrokontroller-Anwendung aber über die COM Schnittstelle Daten schicken. Die Anwendung kann diese Daten auswerten und entsprechend reagieren. Im Abschnitt «Benutzereingaben» sind die Möglichkeiten des AVR-Teminals genauer beschrieben.
* Interaktion mit der Hardwareperipherie des Mikrokontrollers: Die Mikrokontroller der Familie ATmegaXX verfügen über eine Vielzahl von Hardware-Funktionen. All diese Funktionen sind in einer umfangreichen Spezifikation beschrieben. Wir können diese Funktionen mit einem C-Programm nutzen.   
  Diese Hardware abreitet eigentlich völlig unabhängig von irgend einem C-Programm. Wir könnten also diese Funktionen auch ausprobieren, ohne eine C-Programm schreiben zu müssen. Das ist nützlich, wenn es darum geht, eine Hardwarefunktion kennen zu lernen. Wir könnten direkt auf einzelne Register zugreifen und die Auswirkungen beobachten.  
  Die Applikation AVR-Terminal stellt eine Register-View zur Verfügung, welche genau das ermöglicht. Mehr zum diesem Thema im Abschnitt «Register-View»
* Macro-Expansion: Wenn wir Hardware programmieren, spielen Makros eine gewisse Rolle. Der C-Präprozessor expandiert solche Makros zu C-Code. Im Modul M242 habe ich ein bisschen genauer beschrieben, wie dieses «expandieren» funktioniert. Die Applikation AVR-Terminal beinhaltet einen Präprozessor, mit welchem unabhängig von einen C-Programm geübt werden kann. Mehr dazu im Abschnitt «Makro-expansion»

## Verbindung zum Board:

Das Board ATmega328 verfügt über einen USB-Stecker. Über diesen Stecker kann eine serielle Verbindung zwischen Board und PC hergestellt werden. Ist der Board-Treiber auf dem PC installiert, so wird im Geräte-Manager ein neuer COM-Port sichtbar, sobald das Board mit dem USB-Kabel am PC angeschlossen wird.

Diese serielle Schnittstelle wird verwendet, um eine Applikation auf das Board zu laden. Für den Build-Prozess müssen wir den COM-Port in eine Konfigurationsdatei schreiben. In einer laufenden Anwendung wird dieselbe Schnittstelle aber auch verwendet, um Daten mit dem PC auszutauschen.

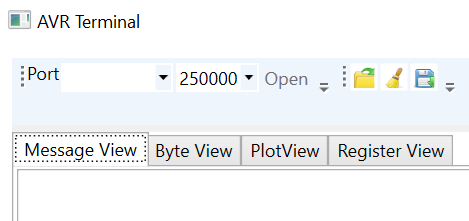
Damit die Applikation AVR-Terminal Daten, die vom Board geschickt wurden, empfangen kann, muss der entsprechende COM Port selektiert und geöffnet werden. Zudem muss die richtige Kommunikationsgeschwindigkeit ausgewählt werden. In Abbildung 1 sind die entsprechenden Elemente der Toolbar dargestellt.



Abbildung : Ausschnitt Toolbar AVR-Terminal

1) Combobox für die Auswahl des Ports

2) Combobox für die Auswahl der Baudrate (Kommunikationsgeschwindigkeit. Für ATmega328 ist 250000 passend

3) Button für das Öffnen und schliessen des COM-Ports. Achtung: Damit eine neue Anwendung auf das Board geladen werden kann, muss die Applikation AVR-Terminal den Port freigeben. Es existiert eine Automatismus, der versucht, selbständig festzustellen, ob der Port geschlossen werden soll. Es kann aber sein, dass das manuell gemacht werden muss.

## Tracen

Es geht beim Tracen darum, Daten der Mikrocontroller-Applikation (Firmware) am PC darzustellen. Für den AVR Terminal existieren unterschiedliche Möglichkeiten der Darstellung.

### Raw Data Tracing

Das Raw-Data Tracing ist die primitivste, aber auch effizienteste Möglichkeit, den Programmablauf der Firmware aufzuzeichnen. Dabei werden in der Firmware einzelne Bytes mit der Funktion PutChar(uint8\_t b) auf die serielle Schnittstelle geschrieben. Im AVR Terminal werden diese Bytes in hexadezimaler Form im Fenster Byte-View angezeigt. Für jede Stelle im Programm müssen Sie ein anderes Zeichen wählen!

### Formatted Traces

Das Aufspüren der Stellen im Programm anhand einzelner Hex-Zahlen ist mühsam. Viel schöner wäre es, einen lesbaren String am Bildschirm zu sehen. In eingebetteten Systemen hat das Übertragen von ganzen Strings an den PC aber zwei Haken.

* Die Strings brauchen Platz im Arbeitsspeicher. Der Atmega328 verfügt über 2K RAM (2^11 Bytes). Darin müssen alle Datenstrukturen der Applikation Platz finden. Würden wir 32 Zeichenketten mit einer maximalen Länge von 32 Bytes für Tracing verwenden, wäre schon die Hälfte des Speichers für «Debugging» weg.
* Das Übertragen von vielen Bytes braucht Zeit. Im Vergleich zum Arbeitsgeschwindigkeit des Prozessors ist die Übertragung von Daten auf der seriellen Schnittstelle langsam. Jede Art von Traces wird das Verhalten der Firmware beeinflussen. An gewissen Stellen im Programm werden schon wenige Zugriffe auf die serielle Schnittstelle dazu führen, dass das Programm nicht mehr funktioniert. Ganze Strings vom Microcontroller an den PC zu schicken ist also auch aus diesem Grund keine Option.

Um diese zwei Probleme zu umschiffen habe ich das Makro TRACE(«string») gemacht. Der Build-Prozess ersetzt das Makro Trace im Code durch einen Funktionsaufruf. Anstelle des Strings wird aber nur eine Zahl auf die serielle Schnittstelle geschrieben. Gleichzeitig erstellt der Build-Prozess eine .json-Datei, welche die Meldungen und die dazugehörenden Zahlen (Ids) enthält.

Normale Trace-Meldungen werden im AVR-Terminal im Tab «Message-View» angezeigt. Damit die Meldungen richtig angezeigt werden können, muss aber die entsprechende .json-Datei geöffnet wernden.

Im TRACE Makro können auch Argumente angegeben werden. Das Makro TRACE(«count = %8», i) wird eine Meldung der Form «count = 196» ausgeben. Das Zeichen %8 ist also ein Platzhalter für die Variable i. Diese Variable wird als 8-Bit Variable vom Mikrocontroller an den PC übertragen.

Die Trace-Meldung «@plot» wird vom AVR-Terminal speziell interpretiert. Eine solche Trace-Meldung soll ein numerisches Argument haben. @plot Trace-Strings werden nicht in der Message View dargestellt, sondern als ein neuer Messpunkt in einer Datenreihe interpretiert. Eine solche Datenreihe wird im Fenster «Plot-View» als Graph dargestellt. Die Skalierungen der Achsen können zu Laufzeit angepasst werden.  
Auf diese Weise ist es möglich Messungen des Mikrocontrollers graphisch darzustellen. Wir können ein einfaches «Scope» simulieren (analog zu Scope in Matlab). Auf der X-Achse haben wir die Zeit. Auf der Y-Achse den aktuellen Messwert.

## Benutzereingaben:

TBD:

## Register View:

TBD: