



RMB AVR Board

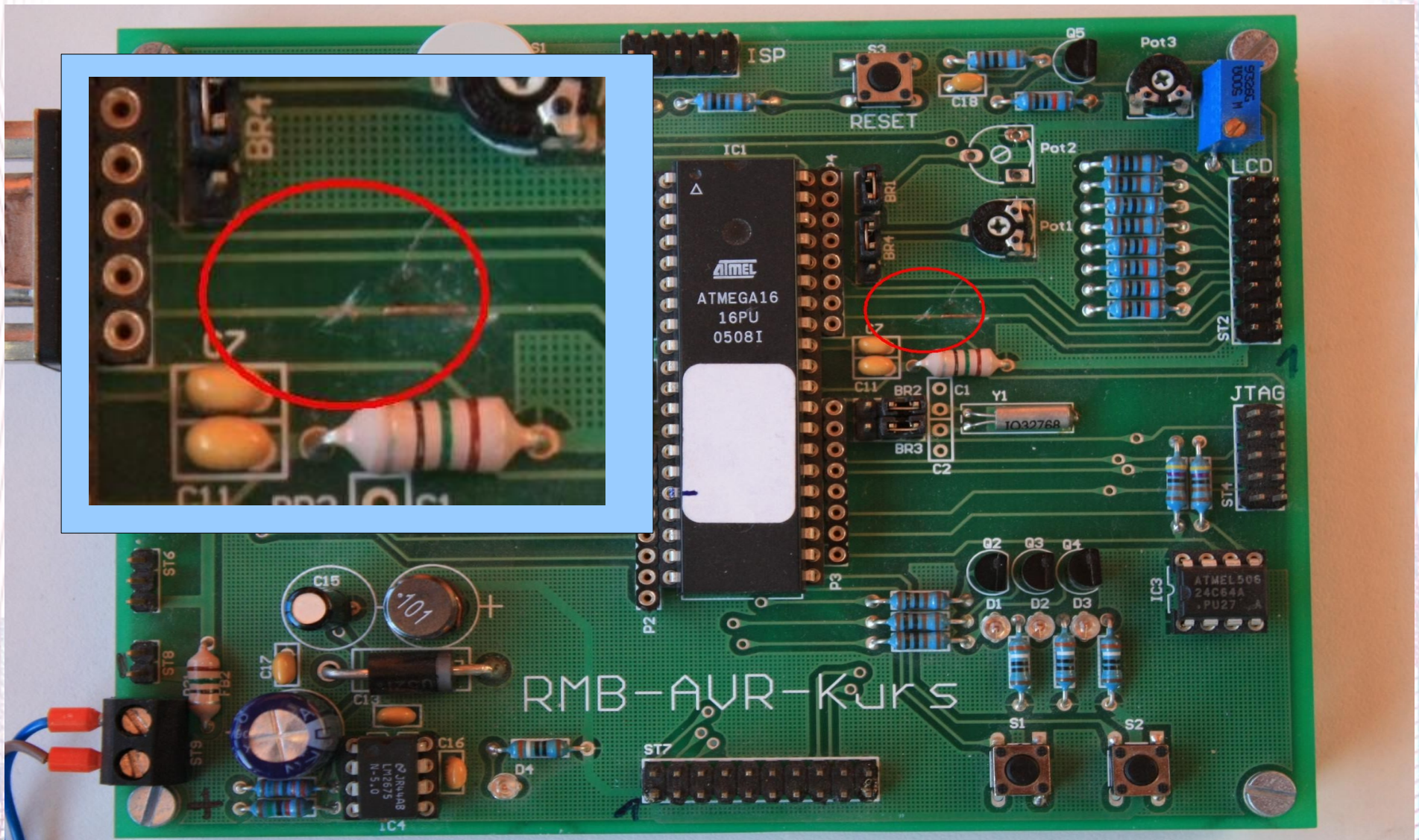
Das RMB AVR Board - ein vergessener Schatz



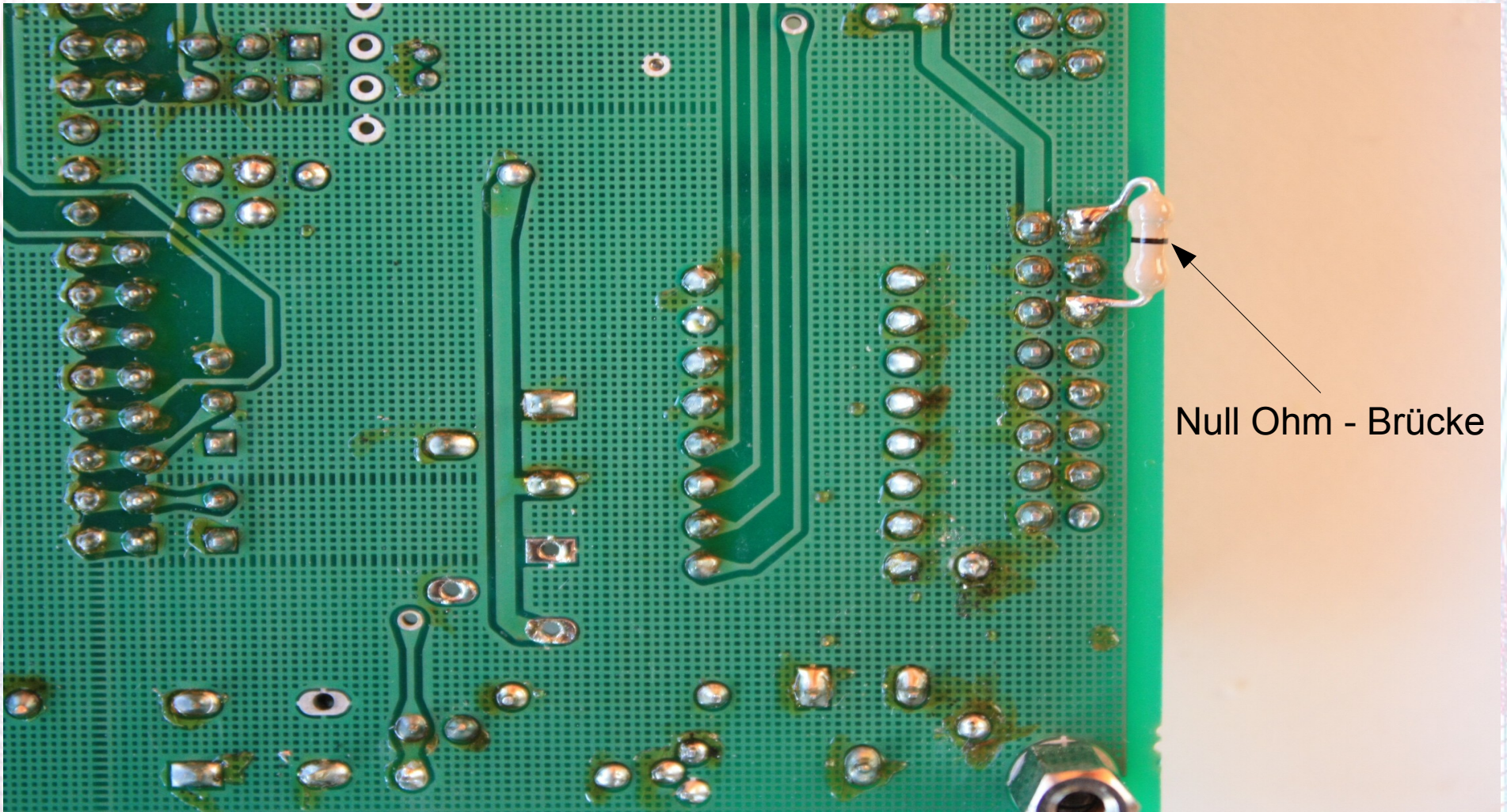
LCD Ansteuerung

- Zur Nutzung der BASCOM Hochsprachen Befehle zur LCD Ansteuerung ist eine Hardwareänderung erforderlich
- Siehe BASCOM Handbuch unter „Attaching an LCD Display“ und Beschreibung LCD S.4 Table 2 Interface Signals.
- Die WE- Leitung vom ATMEGA zum LCD-Display unterbrechen
- Die Unterbrechung prüfen (Piepser o.ä)
- Brücke am ST2 von Pin 1 (Masse) auf Pin 5 (WE)

LCD Ansteuerung Unterbrechung der WE- Leitung



LCD Ansteuerung Brücke auf der Rückseite





LCD Ansteuerung

Beispiel LCD.BAS

Config Lcd = 16 * 2

Config Lcdbus = 4

Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 ,
Db6 = Portc.4 , Db7 = Portc.5 , E = Portb.4 , Rs = Portb.5

Upperline

Lcd "Hello World"

Lowerline

Lcd "Hallo Welt"

End



LCD Ansteuerung

- Konfiguration des LCD in den Compiler-Einstellungen (Options – Compiler – LCD) oder im Deklarationsteil des Quelltextes
- Fülle von Befehle zur LCD Ansteuerung vorhanden: CLS, ShiftCursor left/right, Home, Locate, Cursor....
- Es können diverse LCD angesteuert werden bis 20*4 Zeichen
- Farbdisplays bei <http://www.display3000.de/>



Ein/Ausgänge

Konfiguration der Ports

- Benötigte Ports als Ausgang konfigurieren:
 $\text{PORTX.Y} = \text{Output}$
- Benötigte Ports als Eingang konfigurieren:
 $\text{PINX.Y} = \text{Input}$
- PullUp einschalten: $\text{PortX.Y} = 1$
- Der interne Pullup zieht den Eingang bei nicht betätigtem Schalter über einen hochohmigen Widerstand auf U_{VCC}
- Daher werden die Schalter über Massepotential ausgewertet, 0 - Aktiv
- Siehe RMB-Atmel-Kurs Teil 11 oder
http://de.wikipedia.org/wiki/Open_circuit



Ein/Ausgänge

Beispiel PORTS.BAS

Config Pina.7 = Output

Config Pind.6 = Input

Portd.6 = 1

'* Grüne LED

'* S1

'* Pullup einschalten

Do

If Pind.6 = 0 Then

Porta.7 = 1

Else

Porta.7 = 0

End If

Waitms 100

Loop

END



Ein/Ausgänge

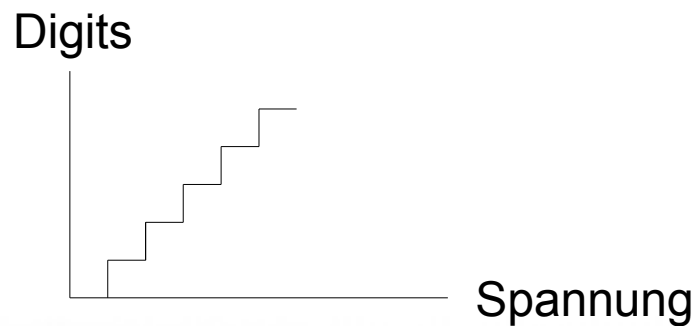
Interessante Befehle für Ports

- Debounce zur Kontaktentprellung
- Alias zur leichteren Ansteuerung, siehe Quelltext
- Toggle zum Ein/Ausschalten
- Hintergrund-Beleuchtung für LCD an Port D.4



AD-Wandler

- Auflösung $2^{10} = 1024$ Stufen (Digits)
- Maximale Messspannung 5V
- Pro Stufe $5 \text{ (Volt)} / 1024 = 0,0048\text{V}$
- Referenzspannung auf Versorgungsspannung einstellen
- 8 Kanäle massebezogene Eingänge
- Wandlungsdauer $125\mu\text{s}$, Ergebnis schwankt, siehe ADC1.BAS
- Mittelwertbildung zur Abhilfe, siehe ADC2.BAS





AD-Wandler

Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc

Dim Digits As Word
Dim Voltage As Single

Do
Cls
Digits = Getadc(2)
Upperline
Lcd "Digits:" ; Digits
Voltage = Digits * 0.0049
Lowerline
Lcd "Spg:" ; Voltage
Waitms 20
Loop

End

'* Enthält Wandlungsergebnis
'* In Spannung umgerechnet

'* Poti 1 auf dem RMB Board

'* Umrechnen in eine Spannung

'* Wartepause, sonst ist nichts zuerkennen



Timer/Counter/Interrupt

- Timer/Counter sind Zähler im Prozessor
- Zählen den internen Taktgeber (Prozessortakt) oder externe Ereignisse (Eingänge PB0 und PB1 laut Datenblatt ATMEGA S.56)
- Zwei 8bit (256) und ein 16bit (65536) Timer/Counter
- Auswertung durch ständige Kontrolle des aktuellen Zählwertes (Polling) oder durch Interrupt beim Erreichen eines eingestellten Wertes
- Der Prozessortakt kann über einen Teiler (Prescaler) verringert werden

Config Pina.7 = Output

Config Timer0 = Timer , Prescale = 1024

'* Als Ausgang konfigurieren, Grüne LED

'* Prozessortakt zählen

Dim W As Word

Timer0 = 0

Cls

Upperline

Lcd "Timer V1"

'* Zähle von Null beginnend

Start Timer0

Do

W = Timer0

Lowerline

Lcd "W: " ; W

If W = 255 Then

Stop Timer0

Porta.7 = 1

Timer0 = 0

Waitms 1

Porta.7 = 0

Start Timer0

End If

'* Aktuellen Zählerstand auslesen...

'* ... und ausgeben.

'* Zähle von Null auf 255 und beginne von vorn

'* Zum neu einstellen kurz anhalten

'* Grüne LED einschalten

'* Zähle von Null beginnend

'* Wartezeit, sonst sieht man die LED nicht blitzen....

'* Grüne LED ausschalten

'* Timer starten

Loop

End

'* Hauptprogramm



Timer

- Prescaler (Teiler) -Stufen 1, 8, 64, 256 und 1024
- Bei einer Teilung von 256 und einer Taktfrequenz von 16MHz zählt Timer in genau einer Sekunde auf $\text{Takt}/256=62500$.
- Daraus folgt, um einen Überlauf des Timers nach einer Sekunde zu erreichen und damit einen Interrupt auszulösen, muss der Timer mit einem Zählwert von 3036 ($65536 - 62500$) starten.
- Interrupt -Unterbrechungsanforderung durch interne Schaltungsteile (z.B. AD-Wandlung fertig, Ergebnis steht bereit, Serielle Daten werden empfangen...)
- Reaktion auf externe Ereignis , Eingänge PD2 (INT0) und PD3 (INT1)



Timer

Config Timer1 = Timer , Prescale = 256

Enable Timer1

On Timer1 Isr_timer1

Enable Interrupts

Timer1 = 3036

Cls

Start Timer1

Upperline

Lcd "Timer V2"

Do

NOP

Loop Until Pind.6 = 0

Lowerline

Lcd "Programmende!"

End

Isr_timer1:

Stop Timer1

Toggle Porta.5

Timer1 = 3036

Start Timer1

Return

'* Interrupt Timer 1 freigeben

'* Wenn Timer 1 fertig - Interrupt! Springe zu..

'* Interrupts global freigeben

'* Timer Ablauf starten

'* Mach mal Pause

'* Hauptprogramm

'* Sprungziel bei einem Interrupt

I²C mit EEPROM

- Von Philips entwickelt zur seriellen Inter IC Communication
- Zweidraht-Bus mit Takt (SCL) und Daten (SDA)
- Vielzahl von IC's (AD/DA-Wandler, Speicher) und Sensoren verfügbar, Hot-Plug fähig
- Nachteil: Langsam, kurze Leitungslängen
- Jedes Device am I2C Bus hat eine Adresse, wird beim 24C64 durch A0-A2 bestimmt, Master-Device = 0 (Controller)
- Das erste (Adreß)Byte wird die Adressleitung und das Control-Wort definiert, es folgen Adresse High/Low Byte
- Control-Wort Schreiben 1010_0010(b)=A2(h)
- Control-Wort Lesen 1010_0011(b)=A3(h)

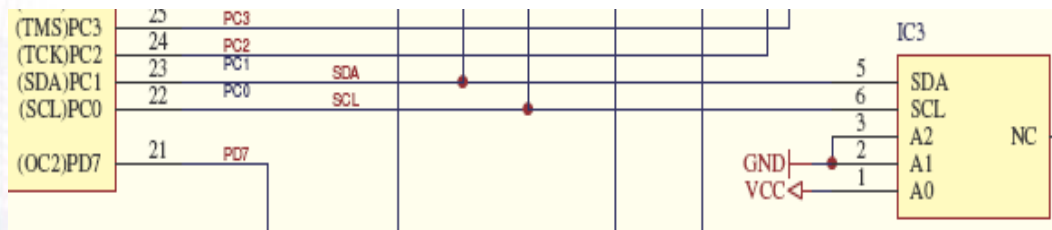
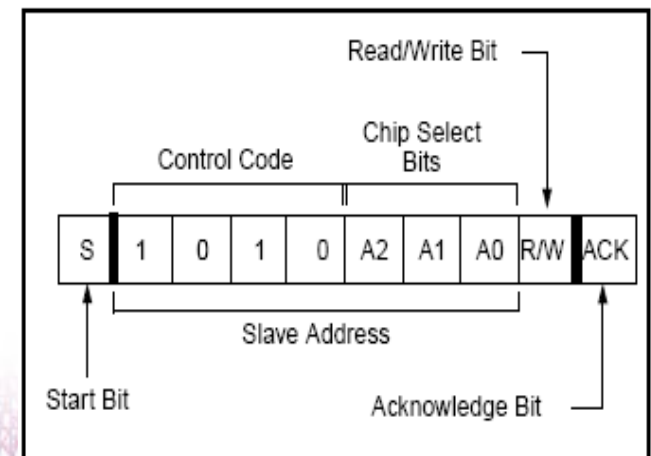


FIGURE 5-1: CONTROL BYTE FORMAT





I²C mit EEPROM

Schreiben (Auszug)

I2cstart	'* Start I2C
I2cwbyte &HA2	'* Sende gewünschte Slave Adresse und Control-Word
I2cwbyte Hb	'* Sende High-Byte der Speicheradresse
I2cwbyte Lb	'* Sende Low-Byte der Speicheradresse
I2cwbyte VAR	'* Sende zu schreibenden Wert
I2cstop	'* Stop I2C
Waitms 10	'* warte 10ms, laut BASCOM Handbuch, Grund?

....

Lesen (Auszug)

I2cstart	'* Start I2C
I2cwbyte &HA2	'* sende gewünschten Slave Adresse und Control-Word
I2cwbyte Hb	'* Sende High-Byte der Speicheradresse
I2cwbyte Lb	'* Sende Low-Byte der Speicheradresse
I2cstart	'* Erneuter Start I2C, ist vom 24C64 so spezifiziert
I2cwbyte &HA3	'* sende Slave Adresse +1 für Lesen, siehe Datenblatt 24C64 S.6
I2crbyte VAR , Nack	'* lese Daten vom EEPROM, mit NACK abschliessen
I2cstop	'* Stop I2C



Servoansteuerung

- Wird nach gereicht.....



Nicht zu vergessen

Möglichkeiten mit dem RMB-AVR Board und BASCOM

- Serielle Kommunikation per RS232, statt „LCD“ „Print“
- Auswertung DCF77 Signale
- Realtime-Clock
- Erzeugung PWM Signale (Servoansteuerung)
- Messung der Eingangsspannung für Batteriegespeiste Applikationen
- Speichern von Daten auf einer SD-Karte, Dateimanipulation
-

**Beachte Doppelbelegungen auf dem RMB Board
beim Anschluss von Sensoren o.ä!**



Projektvorschläge

- Lichtschranke zur Geschwindigkeitsmessung
- Anschluss eines Farbdisplays
- Wetterstation
- Ablage von Messdaten auf einer SD Karte, siehe MCSELEC.COM und DISPLAY3000.DE
- Steuern des RMB Boards mit einem Windows Programm per RS232, siehe http://www.rn-wissen.de/index.php/Windows_Programm_zum_Steuern_des_AVR's



Tipps

Berühmt-berüchtigte Programmierfehler

- Komma statt Punkt oder Semikolon
 - Deklarationsfehler der Variablen
 - Abbruchbedingung einer Schleife wird nie erreicht
 - Zum Erlernen einer Programmiersprache am besten die Befehlsliste lesen
 - Teilziele setzen, z.B. wie funktioniert der A/D-Wandler.
 - Kleine Programme mit Teillösungen entwickeln
 - Buchempfehlung: Stefan Hoffmann „Einfacher Einsteig...“ ISBN 978-3-8391-8430-1
- Interessante Links:
- <http://www.mcselec.com/>
 - <http://www.mikrocontroller.net/>
 - <http://www.rn-wissen.de/index.php/Bascom>
 - <http://www.myavr.de/>
 - <http://www.bascom-forum.de/>