1 Pin-Belegungen der ATmega-Controller

1.1 Controller (Steuergerät)

Port A: ADC und LED-Ansteuerung
PA.0 (ADC0): Vorlaufspannung vom SWR-Koppler
PA.1 (ADC1): Rücklaufspannung vom SWR-Koppler
PA.2 (ADC2): ADC-Eingang für spätere Vollautomatik, z.B. Phase
PA.3 (ADC3): ADC-Eingang für spätere Vollautomatik, z.B. Impedanz

PA.4: Schaltausgang für Lock-LED
PA.5: Schaltausgang für Prog-LED
PA.6: Schaltausgang für Auto-LED
PA.7: Schaltausgang für Man-LED

Port B: LED-Ansteuerung, Counter, ISP und LCD
PB.0: Schaltausgang derzeit unbelegt
PB.1: Eingang für Counter (Timer 1)
PB.2: LCD Pin "RS"
PB.3: LCD Pin "E"
PB.4: LCD Pin "D7"
PB.5: ISP-MOSI und LCD Pin "D6"
PB.6: ISP-MISO und LCD Pin "D5"
PB.7: ISP-SCK und LCD Pin "D4"

An PB1 ist ein Frequenzzähler mit Vorverstärker, Schmitt-Trigger und 1:8 Teiler angeschlossen. Eine Frequenzkalibrierung ist mit dem Trimmer C12 möglich. Die Eingangsempfindlichkeit beträgt ca. 200 mVss. Die Zählerereignisse wertet der 16bit-Timer 1 aus, das Zählergate steuert der 8bit-Timer 0. Zum Counter-Timing siehe Excel-Tabelle.

Port C: Taster-Eingänge mit internem Pullup
PC.0: Derzeit unbelegt
PC.1: Derzeit unbelegt
PC.2: Encoder-Taster
PC.3: Mode-Taster
PC.4: Store-Taster
PC.5: Reset-Taster
PC.6: Tune-Taster
PC.7: Lock-Taster

Der Taster-Port C wird fortlaufend mit jedem Timer 0-Interrupt abgefragt. Die Auswertung, ob ein Taster betätigt wurde, und wenn ja, welcher, erfolgt in der Timer-Interrupt-Routine "Timer0_isr".

Port D: RS485-Kommunikation mit der Remote-Unit, Encoder und RX/TX-Steuerung
PD.0 (RXD): RS485-Empfangskanal
PD.1 (TXD): RS485-Sendekanal
PD.2: (INT0): RS485-Sende-/Empfangsumschaltung (High: TX, Low: RX)
PD.3: Schaltausgang für PTT-Relais
PD.4: Schaltausgang für +12V TX-Mute-Signal, PD.3 und .4 werden gleichzeitig geschaltet
PD.5: Jumper Disable Remote Unit, Jumper in: disabled
PD.6: Encoder B
PD.7: Encoder A

Vorsichtshalber sind an den Ausgängen A und B des MAX485 auf der Platine je ein Pullup-/Pull-down-Widerstand (R11/R13) vorgesehen, die aber im Mustergerät nicht bestückt wurden.

Falls der Encoder falsch herum dreht, Anschlüsse A und B vertauschen. Auf der Platine sind dazu Lötpads vorgesehen.

1.2 Remote Unit

Der in der **Remote Unit** eingesetzte MEGA16 ist eigentlich total unterfordert, bietet aber genug Beinchen, um die gewünschten Funktionen ohne großen Aufwand zu realisieren.

Port A: Steuerung der Relais der C-Bank über einen ULN2803A.

Port B: ISP für die Inline-Programmierung (PB5-7) und ein 16x2-LCD für Testzwecke (PB0-4). Der Jumper J1 zwischen MOSI (PB5) und LCD-D4 kann ständig gesteckt bleiben. Es wurde (damals, noch zu Anfang meiner AVR-Aktivitäten) befürchtet, dass die ISP-Programmierung durch den Anschluss des LCD am gemeinsam genutzten Port PB5 gestört werden könnte. Das war aber nicht so.

Port C: Wie Port A für die L-Bank. In der Variante "Christian-Tuner" mit nur 7 Induktivitäten sind nur die Ausgänge PC0 bis PC6 (7 Bit-Auflösung) aktiv.

Port D: RS485 wie beim Controller, PD3-6 unbenutzt, PD7 für die Relais 1 bis 4 zur Umschaltung Hoch- / Tiefpass.

2 Die Software

Die Software für die beiden ATMega ist in BASCOM AVR geschrieben.

Der Code ist ausführlich dokumentiert, so dass hier nur die wesentlichen Aspekte erläutert werden sollen. Der Quellcode ist Open Source für den privaten Gebrauch. Eine kommerzielle Nutzung ist nicht gestattet.

Die Software (.bas und .hex) steht einschließlich AVRDUDE-Batchfiles und einer Excel-Mappe mit Layoutunterlagen weiter unten zum Download zur Verfügung.

Bei der Wahl des Quarzes für den Taktgenerator des ATmega im Controller sind zwei Aspekte zu betrachten: Eine möglichst genaue Baudratengenerierung für die RS485-Schnittstelle und eine möglichst einfache Berechnung der Counterfrequenz. Das trifft für einen 16 MHz-Quarz zu. Die Berechnung erfolgte mit Excel (im Download).

2.1 ATU-Controller, Programmstart

In der Routine "CheckEEPROM" wird zuerst das EEPROM geprüft. Bei neuen oder frisch gebrannten Controllern sind die EEPROM-Daten mit dezimal 255 vorbesetzt. Findet das Programm diesen Zustand vor, wird das EEPROM mit Daten initialisiert. Diese Daten sind in der Datei "ATU_Contoller_EEPROM_Data.bas" hinterlegt. Die Datei muss zur Kompilierung mit BASCOM im gleichen Verzeichnis wie die Quelle des Hauptprogramms liegen. Bei der Kompilierung wird die Datei per Include eingebunden.

Nachfolgend wird die Kommunikation zur Remote-Unit geprüft. Dazu wird ein Testdatensatz an die Remote-Unit geschickt und auf Antwort gewartet (Routine "SendRemote"). Es erfolgt eine entsprechende Meldung im Display. Um das Steuergerät auch ohne Remote Unit testen zu können, ist auf der Platine der Jumper J1 vorgesehen. Ist er gesteckt, ist die Kommunikation zur Remote Unit außer Betrieb.

2.2 ATU-Controller, Hauptprogramm

Es gibt 4 Betriebsmodi:

(1) **Automatic**: Suche zur anliegenden TX-Frequenz die ATU-Voreinstellungen im EEPROM und stelle den ATU entsprechend ein.

(2) **Manual**: Stelle den ATU manuell mit dem Encoder ein.

(3) **Program**: EEPROM programmieren, stelle den ATU manuell mit dem Encoder ein und speichere die ATU-Daten im EEPROM.

(4) **Setup**: Passe die im EEPROM gespeicherten grundlegenden Betriebsparameter an.

In der Do...Loop werden ständig abgefragt und entsprechend reagiert:

- **Mode**-Taster
  Durchtakten den Betriebsmodi (1) bis (4)
- **Encoder**-Taster
  Aktiviere im Modus (2) oder (3) die manuelle Einstellung wahlweise für L, C oder Hoch-/Tiefpass.
- **Reset**-Taster
  Setze im Modus (2) oder (3) je nach aktueller Aktivierung den Wert für L oder C auf "Null" zurück.
- **Lock**-Taster
  Sperre in Modus (2) die manuelle Encodereinstellung (Lock-LED leuchtet). Im Automatic-Modus (1) kann mit einem manuellen Lock verhindert werden, dass die ATU-Relais fortlaufend geschaltet werden, wenn die TX-Frequenz genau auf einer Bandsegmentgrenze liegt, so dass sich der Controller nicht entscheiden kann. Wiederholtes Tasten hebt die Sperre wieder auf.
- **Tune**-Taster
  In den Betriebsmodi (1) bis (3) wird die TX-Steuerung zur Leistungsreduktion während des Schaltens der ATU-Relais getestet. Dazu werden das PTT-Relais geschaltet und +12V an den TX-Mute-Ausgang gelegt, womit der TX seine Leistung reduzieren sollte. Diese reduzierte Leistung wird gemessen und mit der im Setup festgelegten maximalen Leistung verglichen. Liegt sie darunter, wird "is OK" angezeigt, sonst "notOK". Im ersten Fall erhält die Remote Unit bei den Abstimmvorgängen Steuersignale zum Schalten der ATU-Relais, im zweiten Fall nicht.
  Im Setup-Mode dient der Tune-Taster zum Weiterschalten des Cursors um eine Stelle nach rechts.

2.2.1 Automatic Mode

Die Routine " FindBand" sucht anhand der anliegenden TX-Frequenz die passende ATU-Einstellung im EEPROM. Wurde ein neues Bandsegment gefunden, sind diese EEPROM-Daten die Anfangsdaten auch für den Manual- und den Program-Mode. Die Routine "DisplayFreq" zeigt die TX-Frequenz und die Mittenfrequenz des zugehörigen Frequenzsegments an. Die Remote Unit wird mit den Einstelldaten für L, C und Hoch-/Tiefpass versorgt, die Routine "DisplayLC" zeigt diese an.

2.2.2 Manual Mode

Die Routine "DisplayFreq" zeigt die TX-Frequenz und die Mittenfrequenz des zugehörigen Frequenzsegments an. In Routine "DisplayLC" werden die mit dem Encoder manuell einzustellenden Werte für L, C und Hoch-/Tiefpass angezeigt. Die Einstellungen bleiben erhalten, solange in einem Bandsegment gearbeitet wird.

2.2.3 Program Mode

Manuelle Einstellung wie im Manual Mode. Mit Betätigen des **Store**-Tasters werden die Einstellungen zu dem gerade aktuellen Bandsegment im EEPROM gespeichert.

2.2.4 Setup Mode

Die Setup-Einstellungen 1 ... 15 werden in jeweils 2 Schritten durchlaufen:
(1) Lesen und Anzeigen der jeweiligen EEPROM-Daten
(2) Ändern der angezeigten Daten mit dem Encoder.

Die geänderte Tastenbelegung wird im Display, Zeile 4, angezeigt.

2.3 Kommunikation mit dem Koppler (Remote Unit)

In den Modes "M" (Manual), "A" (Automatic) und "P" (Programm EEPROM) werden alle Änderungen von L, C und H/L an die Remote Unit per RS485 übertragen. In Mode "A" erfolgt eine Übertragung an die Remote Unit nur dann, wenn infolge Änderung der TX-Frequenz auch ein Wechsel des EEPROM-Bandsegments stattfindet. Die Übertragung ist erkennbar, indem die jeweilige Mode-LED zu Beginn der Übertragung erlischt und nach Abschluss wieder leuchtet. Während dieser Zeit wird TX-Mute aktiviert, womit z.B. die PTT-Leitung unterbrochen werden kann. Das TX-Mute-Relais hat Umschaltkontakte, so dass eine Verbindung geöffnet oder geschlossen werden kann. Parallel dazu wird an den Mute-Ausgang +12V für ggf. weitere Sende-/Empfangsumschaltungen gelegt. Die Remote Unit ist nur dann ansprechbar, wenn ein vorher mit dem Tune-Taster erfolgter Test der reduzierten TX-Leistung (s.o. zu 2.2) erfolgreich war.

Die Remote Unit quittiert jede Übertragung. Tritt ein Fehler auf, wird dies mit "Time out" oder "Line Error x", x = 2 bis 5 in der zweiten Displayzeile kurz angezeigt.

Fehler-Nr.	Bedeutung (s.u. RS485-Protokoll in Abschnitt 4)
2	Remote Unit hat kein "ACK" als OK-Quittierung gesendet
3	Antwort der Remote Unit enthält falsche Zeichen
4	Remote Unit wurde mit einer falschen Adresse angesprochen
5	Das erste Zeichen in der Nachricht ist kein STX

Die Bedeutung der Fehlernummern ist auch im Code dokumentiert. "Time out" wird angezeigt, wenn die Remote Unit innerhalb von ca. 1 sec. keine Quittung schickt.

Das Kommunikationsprotokoll ist bewusst einfach gehalten.

Controller-Protokoll zur Übermittlung der Nutzdaten							
Byte	1	2	3	4	5	6	7
	STX	ADR	LEN	B01	B02	B03	CRC

Antwort der Remote Unit				
Byte	1	2	3	4
	STX	ADR	LEN	RES

Bedeutung der Bytes	
STX	Start of Transmission = 2
ADR	Adresse der angesprochenen bzw. antwortenden Remote Unit (hier = 1)
LEN	Anzahl der nachfolgenden Bytes
B01	Encoder-Wert für L (0 ... 255), für "Christian"-Option 0...127
B02	Encoder-Wert für C (0 ... 255)
B03	Hochpass (1) oder Tiefpass (0)
CRC	CRC8-Checksum aus den Bytes 1 bis 6
RES	Bestätigung = ACK (6), OK oder = NACK (21), nicht OK

Die Remote Unit wertet das Protokoll aus, indem sie prüft, ob das Telegramm mit STX beginnt, sie mit der Adresse ADR gemeint ist und das übertragene CRC mit einem selbst berechneten CRC8 übereinstimmt. Sind beide CRC identisch, schaltet sie die Relais entsprechend B01 bis B03 und schickt ein RES=ACK zurück. Der Controller wartet also nach jeder Übertragung auf eine Bestätigung durch die Remote Unit. Auf ein erneutes Übertragen der Nutzdaten im Falle eines RES=NACK wurde verzichtet. Statt dessen wird im LCD des Controllers eine Fehlermeldung kurz angezeigt. Ebenso wird ein Time out (auf ca. 1 sec. eingestellt) im LCD angezeigt. Das Byte ADR ist in dieser Anwendung noch entbehrlich, da es nur eine Remote Unit gibt. Damit ist das Protokoll aber noch offen für Erweiterungen.

2.4 SWR- und Leistungsmessung, der ADC des ATmega1284P

Der SWR-Koppler liefert 2 Gleichspannungssignale für Vor- und Rücklauf, die an den ADC-Eingängen ADC0 und ADC1 anliegen. Daraus werden das SWR und die TX-Leistung berechnet.

Die Werte von ADC0 (Vorlauf) und ADC1 (Rücklauf) werden aus mehreren Einzelmessungen gemittelt, um ein Flackern der Anzeige zu reduzieren, im Code einstellbar mit " Const bytADCRep = 3". Die Mittelwerte (ADC=0...1023) werden mit Hilfe der Vref-Spannung (AVCC, ATmega 1284P, Pin 27, festgelegt im Setup 6) in Volt umgerechnet:

$$V_{ADC} = \frac{ADC * Vref}{1024}\ (V)$$

Die Function "DiodeResponse" berechnet aus den Gleichspannungen V_{ADC} über die Näherungsformel für den Korrekturfaktor CF

$$CF = a * x^b + c$$

(a, b und c im Setup 8, 9, 10 festgelegt) die an den Gleichrichterdioden im Koppler anliegenden HF-Spitzenspannungen für Vor- und Rücklauf:

$$V_{HF} = V_{ADC} * CF\ (V_s)$$

Das SWR ergibt sich daraus zu

$$SWR = \frac{Vvor_{HF} + Vrück_{HF}}{Vvor_{HF} - Vrück_{HF}}$$

In der Routine "BarGraph" wird das SWR als Balken, in der Länge begrenzt mit "Const sngSWRMax = 5", und als numerischer Wert angezeigt. Der numerische Wert ist auf maximal 9,99 begrenzt.

In der Routine "ShowPower" werden vor- und rücklaufende effektive Leistung angezeigt.

$$\text{Mit } V_{eff} = \frac{V_{HF}}{\sqrt{2}},$$

der Windungszahl N der Kopplerspulen (festgelegt im Setup 7, eigentlich das Windungsverhältnis 1 zu N) und der Impedanz Z (im Code festgelegt mit "Const sngImp = 50") wird die effektive HF-Leistung für Vor- bzw. Rücklauf berechnet:

$$P_{eff} = \frac{(V_{eff*N})^2}{Z}\ (W).$$

Nachdem im ersten Anlauf mit den BASCOM-Bordmitteln Config ADC... und GetADC SWR- und Poweranzeige funktionierten, zeigte sich, dass der Encoder etwas schleppender reagierte als vorher ohne den ADC. Die BASCOM-Funktion GETADC hatte ich da in Verdacht, da sie mit jedem Aufruf den Multiplexer auf den jeweiligen ADC-Port einstellt, der Vorteiler einstellt, den ADC startet, wartet, bis die Messung ausführt ist und schließlich den ADC ausliest. Das braucht Zeit. Überhaupt – was versteckt sich hinter solchen Hochsprachenbefehlen? Führen sie tatsächlich das aus, was beabsichtigt ist? Ohne Assemblerkenntnisse steht man da im Dunkeln.

Zeit also, sich mit den ADC-Registern im Datenblatt zu beschäftigen, um den ADC direkt über die Register zu steuern und auszulesen. Da der verwendete ATmega1284P aus einer neueren Baureihe stammt, gibt es einige Unterschiede zu den "alten" Controllern wie dem ATmega8 und ATmega16/32. Hier also die Beschreibung der ADC-Register des **ATmega1284P**.

2.4.1 Register ADMUX – ADC Multiplexer Selection Register

Mit diesem Register werden die Referenzspannung, die Datenablage und der Multiplexer eingestellt, Schreib- und Lesezugriff.

Bit	7	6	5	4	3	2	1	0
Name	**REFS1**	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**
Default	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- **Bit 7-6: REFS0, REFS1 (Reference Selection Bits 0,1)**
  Diese Bits bestimmen, welche Referenzspannung der ADC verwenden soll. Eine Änderung wird erst nach Abschluss der laufenden Wandlung vorgenommen.

REFS1	REFS0	Referenzspannungsquelle
0	0	Extern an AREF anliegend, internes Vref deaktiviert
0	1	Extern an AVCC anliegend mit externen Kondensator am AREF Pin
1	0	Interne 1,1V Referenz mit externen Kondensator am AREF Pin
1	1	Interne 2,56V Referenz mit externen Kondensator am AREF Pin

- **Bit 5: ADLAR (ADC Left Adjust Result)**
  Dieses Bit legt fest, wie der ermittelte 10bit-Messwert im ADC Data Register ADCL und ADCH abgelegt wird. Eine Änderung wird sofort, also auch bei laufender Wandlung vorgenommen.
  ADLAR=0: Ausrichtung rechts, (ADCH_ADCL) = (000000xx_xxxxxxxx)
  ADLAR=1: Ausrichtung links (ADCH_ADCL) = (xxxxxxxx_xx000000).

- **Bit 4-0: MUX4:0 (Analog Channel and Gain Selection Bits)**
  Diese Bits legen fest, welche Analogeingänge über den Multiplexer an den ADC (es gibt nur einen ADC) gelegt werden. Für Differentialeingänge können sie auch die Verstärkung einstellen. Eine Änderung wird erst nach Abschluss der laufenden Wandlung vorgenommen.

  Hier nur die Einfacheingänge, Differentialeingänge im Datenblatt, Seite 256.

MUX4...0	00000	00001	00010	00011	00100	00101	00110	00111
Eingang	ADC0	ADC1	ADC2	ADC3	ADC4	ADC5	ADC6	ADC7

2.4.2 ADCSRA - ADC Control and Status Register A

Dieses Register steuert den ADC und stellt Kontrollsignale zur Verfügung, Schreib- und Lesezugriff.

Bit	7	6	5	4	3	2	1	0
Name	**ADEN**	**ADSC**	**ADATE**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**
Default	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- **Bit 7: ADEN (ADC Enable)**
  ADEN = 0: ADC wird deaktiviert,
  ADEN = 1: ADC wird aktiviert
  Wird der ADC während einer Wandlung deaktiviert, wird die Wandlung abgebrochen.

- **Bit 6: ADSC (ADC Start Conversion)**
  ADSC = 0: Kein Effekt,
  ADSC = 1: Wandlung wird gestartet.
  In der Einfachwandlung (Single Conversion Mode) muss ADSC vor jeder Wandlung auf 1 gesetzt werden.
  Im Freilaufmodus (Free Running Mode) wird die fortlaufende Wandlung mit ADSC=1 erstmalig angestoßen. Die erste Wandlung nach Start des ADC mit ADEN=1 und anschließendem Setzen von ADSC=1 ist eine mit 25 ADC-Takten lange Initialisierungs-wandlung. Anschließende Wandlungen brauchen nur 13 ADC-Takte.

  ADSC behält der Wert 1, solange die Wandlung läuft. Nach deren Ende geht ADSC auf 0. Damit kann also das Ende einer Wandlung abgefragt werden.

- **Bit 5: ADATE (ADC Auto Trigger Enable)**
  ADATE = 0: Kein Auto Trigger
  ADATE = 1: Aktivierung des Auto Trigger Modes. Eine Wandlung wird durch die positive Flanke des ausgewählten Triggersignals ausgelöst. Die Triggersignalquelle wird mit dem Bit ADTS im Register ADCSRB (s. unten) festgelegt.
  Das Bit muss für den Free Running Mode gesetzt werden. Ebenso im Single Conversion Mode, wenn Interrupts zum Anstoßen einer neuen Wandlung verwendet werden.

- **Bit 4: ADIF (ADC Interrupt Flag)**
  Dieses Bit stellt ein Interruptflag zur Verfügung (ADC Complete Interrupt). Während der laufenden Wandlung ist sein Wert 0. Er wechselt auf 1, wenn eine Wandlung erfolgt ist und das Ergebnis in den Data Registern ADCH, ADCL verfügbar ist. Der ADC Complete Interrupt wird ausgelöst, wenn das ADIE-Bit (s.u.) gesetzt ist und die Interrupts global aktiviert sind. Das Bit wird gelöscht, wenn die entsprechende Interrupt Service Routine verlassen wird. Mit Schreiben einer 1 in das ADIF-Bit kann das Flag gelöscht werden.

- **Bit 3: ADIE (ADC Interrupt Enable)**
  ADIE = 1: Der ADC Complete Interrupt (s.o. ADIF) wird aktiviert. Er wird aber nur dann auch ausgelöst, wenn die globalen Interrupts aktiviert sind. Darauf kann mit einer zugehörigen Interrupt Service Routine reagiert werden.
  ADIE = 0: Triggerquelle ist ein anderer Interrupt.

- **Bit 2-0: ADPS2:0 (ADC Prescaler Select Bits)**
  Der ADC wird mit einer geringeren Frequenz als der Controller selbst getaktet. Diese lässt sich mit einem einstellbaren Teilerfaktor aus der XTAL-Frequenz des Controllers herunterteilen. Die resultierende ADC-Taktfrequenz XTAL / Teilerfaktor sollte bei Messungen mit 10 bit Auflösung zwischen 50kHz und 200kHz liegen, z.B. XTAL=16 MHz ⇨

Teilerfaktor=128 ⇨ADC-Takt 125 kHz. Falls nur 8 bit ausgewertet werden, kann der ADC-Takt bis zu 1 MHz betragen. Zu 8/10 bit-Auflösung s.u. zu ADC Data Register.
ADPS2 bis ADPS0 bestimmen den Teilerfaktor zwischen der XTAL-Frequenz und dem Eingangstakt des ADC. Die ersten zwei Teilerfaktoren 2 sind kein Schreibfehler!

ADPS2	ADPS1	ADPS0	Teilerfaktor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

2.4.3 ADCSRB - ADC Control and Status Register B

Dieses Register legt die Auto Trigger-Signalquelle fest, Schreib- und Lesezugriff.
Bei älteren Controllern, z.B. ATmega 16/32, stehen die ADTSx-Bits im SFIOR (Special Function I/O-Register).

Bit	7	6	5	4	3	2	1	0
Name	**--**	**ACME**	**--**	**--**	**--**	**ADTS2**	**ADTS1**	**ADTS0**
Default	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- **Bit 7-3:** Reserviert, mit 0 beim Schreiben in Register ADCSRB vorbesetzten.
- **Bit 2-0: ADTS2:0 (ADC Auto Trigger Source)**
  Falls ADATE im Register ADCSRA auf 1 gesetzt ist, wird hiermit die Triggerquelle für eine ADC-Wandlung festgelegt. Wenn ADATE=0 gesetzt wird, werden ADTS2:0 ignoriert.
  Eine Wandlung wird durch die positive Flanke des ausgewählten Interrupt Flags angestoßen.

ADTS2	ADTS1	ADTS0	Triggerquelle
0	0	0	Free Running Mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Compare Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Compare Overflow
1	1	1	Timer/Counter1 Capture Event

2.4.4 DIDR0 – Digital Input Disable Register 0

Mit diesem Register können die digitalen ADC-Eingangspuffer deaktiviert werden, Schreib- und Lesezugriff.

Bit	7	6	5	4	3	2	1	0
Name	**ADC7D**	**ADC6D**	**ADC5D**	**ADC4D**	**ADC3D**	**ADC2D**	**ADC1D**	**ADC0D**
Default	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Das Setzen einzelner Bits auf 1 deaktiviert den digitalen Eingangspuffer des entsprechenden ADC-Pins. Wenn also einzelne ADC-Eingänge nicht für Messungen verwendet werden sollen, lassen sie sich hiermit deaktivieren. Das spart Strom.

2.4.5 ADC Data Register ADCL / ADCH

In den beiden Registern ADCL und ADCH stehen die Ergebnisse einer ADC-Wandlung, nur Lesezugriff. Die Anordnung der Daten (10 bit) ist abhängig vom Bit ADLAR im Register ADMUX (s.o.).

ADLAR = 0 (rechtsbündig, Right adjust)

	ADCH								ADCL							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert							ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

ADLAR = 1 (linksbündig, Left adjust)

	ADCH								ADCL							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0						

Das ADC Data Register wird erst dann mit Daten aus einer neuen Wandlung beschrieben, wenn ADCH gelesen wurde. Wenn also alle 10 bit aus ADCL / ADCH zu lesen sind, muss zuerst ADCL, dann ADCH gelesen werden. Dies ist bei dem hier angewandten programmierten Neustart des ADC zwar nicht erforderlich, wurde aber aus den Vorversuchen mit dem freilaufenden (free running) Betrieb des ADC beibehalten.

Ist ADLAR = 0 (rechtsbündig), befinden sich die niederwertigsten bits ADC7 bis ADC0 in ADCL, und die höchsten beiden ADC9 und ADC8 in ADCH. Die restlichen bits sind 0. Damit kann unmittelbar eine 16bit-Word-Variable mit dem 10 bit-Maximalwert 2^10-1 = 1023 beschrieben werden.

Ist ADLAR=1 (linksbündig), werden höherwertigsten 8 bits ADC9 bis ADC2 in ADCH abgelegt, und die ersten beiden ADC1 und ADC0 in ADCL. Die restlichen bits sind 0. Damit steht in ADCH das Ergebnis um 2 bits nach rechts verschoben. Eine Verschiebung um 2 binäre Stellen bedeutet aber eine Division durch 2^2=4. Wenn also eine 8 bit-Auflösung ausreichend ist, ist ADLAR=1 (linksbündig) zu setzen und nur das Register ADCH auszulesen. Der Werteumfang ist dann nicht mehr 0...1023 (10 Bit), sondern nur noch 0...255 (8 Bit).

2.4.6 ADC-Einstellungen und ADC-Steuerung

Mit den im vorherigen Abschnitt gezeigten Möglichkeiten wurden zwei Varianten getestet: Free running und Single Mode. Im Free running Mode war ein beträchtliches Rauschen der Messwerte festzustellen, was sich im Single Mode deutlich reduzierte. Also wurde der Single Mode programmiertem Start der einzelnen ADC-Wandlungen gewählt.

Mit der Initialisierung **ADMUX = &B01000000** wird eingestellt:

Bit	Einstellung	
7-6	01	Referenzspannung extern an AVCC anliegend mit externen Kondensator am AREF Pin
5	0	ADLAR=0: ADCL/ADCH rechtsbündig
4-0	0000	ADC0 wird gemessen

Mit der Initialisierung **ADCSRA = &B10001111** wird eingestellt:

Bit	Einstellung	
7	1	ADEN = 1: ADC einschalten
6	0	ADSC = 0: ADC (noch) nicht starten
5	0	ADATE = 0: Kein Auto Trigger (free running mode)
4	0	ADIF wird von außen nicht gesetzt, ist das Conversion completion Interrupt flag
3	1	ADIE = 1: Conversion completion interrupt wird aktiviert
2-0	111	ADPSx = 111: Teilerfaktor 128 (Crystal 16 MHz / 128 = 125 kHz ADC-Takt)

Mit der Initialisierung **ADCSRB = &B00000000** wird der Default 00000000 bestätigt, eigentlich unnötig:

Bit	Einstellung	
7-3	00000	Reserviert, Default
2-0	000	ADTSx = 000: ADC Free running mode, mit ADATE = 0 aber kein Auto Trigger

Mit der Initialisierung **DIDR0 = &B11111100** sollten eigentlich die digitalen Eingangspuffer 0 und 1 (ADC0, ADC1) aktiviert werden. Eine probeweise Deaktivierung der Eingänge 1 und 0 zeigte aber keine Wirkung (??).

Bit	Einstellung	
7-2	111111	Digitale Eingänge 7 bis 2 deaktiviert
1-0	00	Digitale Eingänge 1 und 0 aktiviert

Mit **On ADC ADC_isr** wird die Interrupt Service Routine definiert, in die mit abgeschlossener Wandlung gesprungen wird.

Unmittelbar vor der Do...Loop wird mit
bytADChannel = 0 der erste ADC-Kanal (ADC0) vorbesetzt und mit
ADCSRA.ADSC = 1 der ADC gestartet.

Mit Beendigung einer Wandlung verzweigt das Programm in die Interrupt Service Routine ADC_isr, da ADIE = 1 gesetzt ist. Dort werden die Messwerte ADCL und ADCH ausgelesen und mit den Overlay-Variablen bytADCLo und bytADCHi der Word-Variablen wrdADC zugeordnet. Diese wird in die Variable wrdADCVal(bytADCValIdx), bytADCValIdx = 1, 2, (Vor- und Rücklauf) geschrieben.

Mit Hochzählen von bytADChannel (0, 1)wird der nächste Multiplexerkanal festgelegt mit **Admux = bytADMUX Or bytADChannel**. Anschließend wird der ADC wieder gestartet.

2.5 Weitere Interrupts (Timer)

Im Controller werden alle drei Timer zur Erzeugung entsprechender Interrupts verwendet.

Timer0 (8bit) für die Counter-Torzeit, die Taster, das LED-Blinken und das Kommunikations-Timeout,

Timer1 (16bit) für die Frequenzzählung und

Timer2 (8bit) für den Encoder.

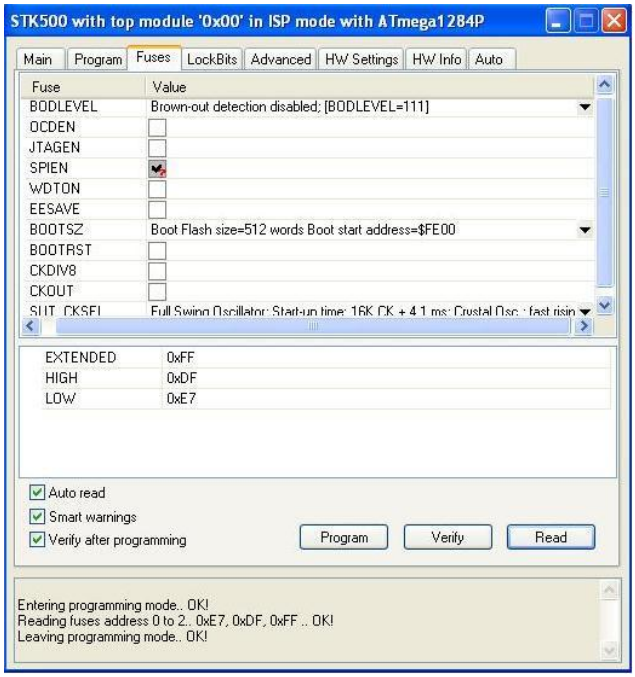
Timer 0 ist mit einem Prescale = 56 verkürzt, so dass in einer Torzeit von 0,064 sec. der achtfache Wert (in kHz) der anliegenden TX-Frequenz gezählt wird (siehe Excel-Berechnung). Die Division durch 8 erfolgt mit einem Speicher sparenden Bit-Shift um 3 Stellen (2^3=8). Die Daten aus den Interrupt-Service-Routinen "TIMERx_isr" werden dem Hauptprogramm in der Endlosschleife "Do ... Loop" zur weiteren Bearbeitung und Anzeige zur Verfügung gestellt.

3 Programmierung der AVR

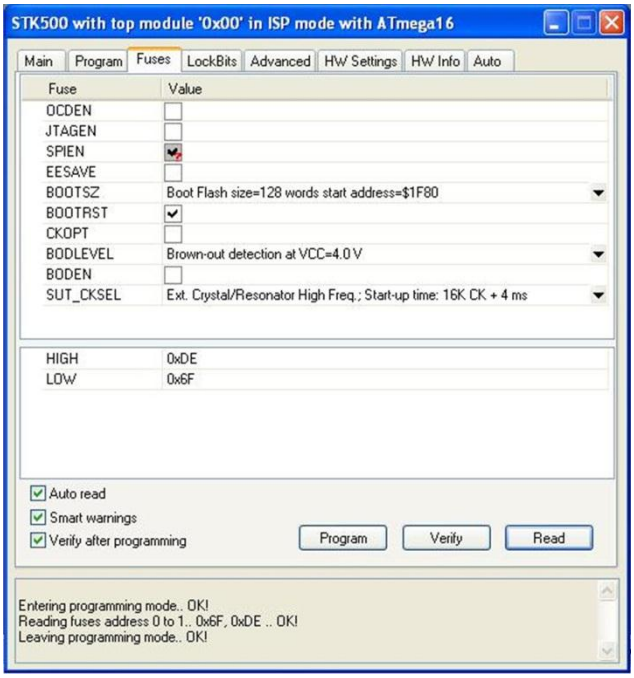
Einzelheiten zur Programmierung von AVR über USB sind unter
http://dl6gl.de/software/avr-programmieren-mit-bascom-und-avr-studio
auf dieser Website erläutert. Zur Einstellung der Fuse-Bits und zum Flashen der .hex-Files mit **AVR Studio** wurden getestet:

- AVR USB ISP Programmer (http://stores.ebay.de/stange-distribution-Shop)
- DIAMEX PROG-S (reichelt.de)

Beide benutzen das STK500-Protokoll.



Fuse Bits des ATmega1248P (Steuergerät)



Fuse Bits des ATmega16 (Remote Unit)

Ist ohne weitere Programmierarbeiten nur das .hex-File zu brennen, kann man sogleich auf den Reiter "Program" in AVR Studio wechseln, das .hex-File lokalisieren und flashen. BASCOM wird hierfür also nicht gebraucht.

Die Benutzung von BASCOM mit den o.g. USB-Programmern oder dem USBasp ist unter
http://dl6gl.de/software/avr-programmieren-mit-bascom-und-avr-studio
auf dieser Website beschrieben.