VERSA1 -
Integriertes Datenerfassungssystem mit 8051er Mikrocontrollerkern

Dr. Claus Kühnel

VERSA1 ist ein vollintegriertes Datenerfassungssystem, welches neben einem 12-Bit AD-Umsetzer einschließlich interner Referenzspannungsquelle zusätzlich eine Reihe eher selten verfügbarer Baugruppen, wie eine programmierbare Stromquelle und eine leistungsfähige Arithmetikeinheit (MAC-Block), auf dem Chip bietet.

Die programmierbare Stromquelle kann Widerstandsbrücken, Thermistoren und anderes treiben. Mit dem MAC-Block lassen sich präzise mathematische Berechnung in hoher Geschwindigkeit ausführen.

Dieser leistungsfähige Baustein kommt im 44-poligen PQFP-Gehäuse einher (Abbildung 1).

Abbildung 1 VERSA1 Baustein

Hersteller des VERSA1 sowie einiger ähnlicher Mikrocontroller ist die kanadische Firma GOAL Semiconductor [www.goalasic.com], die in Deutschland durch SE Spezial Electronic in Bückeburg [www.spezial.de] vertreten wird.

Der folgende Beitrag soll die interessanten Möglichkeiten des VERSA1 für die Messwerterfassung aufzeigen.

Die Programmbeispiele wurden mit BASCOM-8051, einer leistungsfähigen Entwicklungsumgebung mit BASIC-Compiler erstellt, um einer breiten Leserschaft auf einfache Weise den Zugang zu diesem Baustein zu ermöglichen. Eine voll funktionsfähige Demoversion, die auf 2 KB ROM limitiert ist, kann vom Entwickler direkt per Download bezogen werden [www.mcselec.com/download_8051.htm].

Inbetriebnahme und Test der Programmbeispiele erfolgten mit dem ebenfalls von GOAL angebotenen VERSA1 Development Kit.

1. Ressourcen des VERSA1 Mikrocontrollers

Folgende Merkmale zeichnen den VERSA1 aus:

- Erweiterter 8051-kompatibler Kernel (basiert auf Dallas 80C320 Core)
- 64 KByte reprogrammierbarer Flash Memory und 2 KByte OTP-RAM
- 1280 Bytes RAM
- Hardware MAC-Block
- 4-Kanal A/D-Umsetzer
- Interne Spannungsreferenz mit einem Temperaturkoeffizienten von 7 ppm/°C
- Programmierbare Stromquelle
- zwei UARTs
- SPI-Interface (Master/Slave)
- J1708/RS-485 Transceiver
- Clock Control/Power Management Unit
- drei externe Interrupteingänge
- drei Timer/Counter
- zwei General-I/O Pins
- Interne Reset/Brown-Out Schaltung

Abbildung 2 zeigt ein Blockschema dieses komplexen Mikrocontrollers. Grau hinterlegt sind diejenigen Funktionsgruppen, die im folgenden Beitrag behandelt werden. Der Schwerpunkt liegt dabei beim AD-Umsetzer, der für die Messwerterfassung eine zentrale Stellung einnimmt.

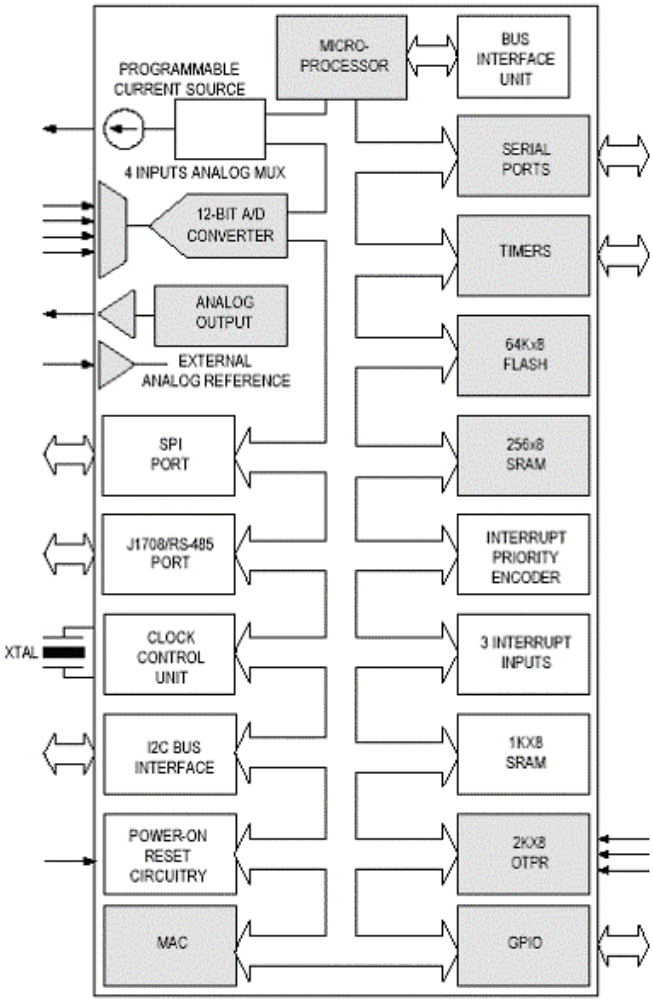

Abbildung 2 VERSA1 Blockschema

Nicht nur die Peripherie wurde komfortabel ausgebaut, sondern auch der vorhandene Speicher. Neben den 256 Byte internen Datenspeicher sind noch 1 KByte RAM und 2 KByte OTP-ROM auf dem Chip integriert. Obwohl dieser Speicher auf dem Chip integriert ist, ist es aus der Sicht des 8051er Prozessors wegen der Adressierung externer Speicher. Abbildung 3 zeigt den im VERSA1 verfügbaren Programm- und Datenspeicher.

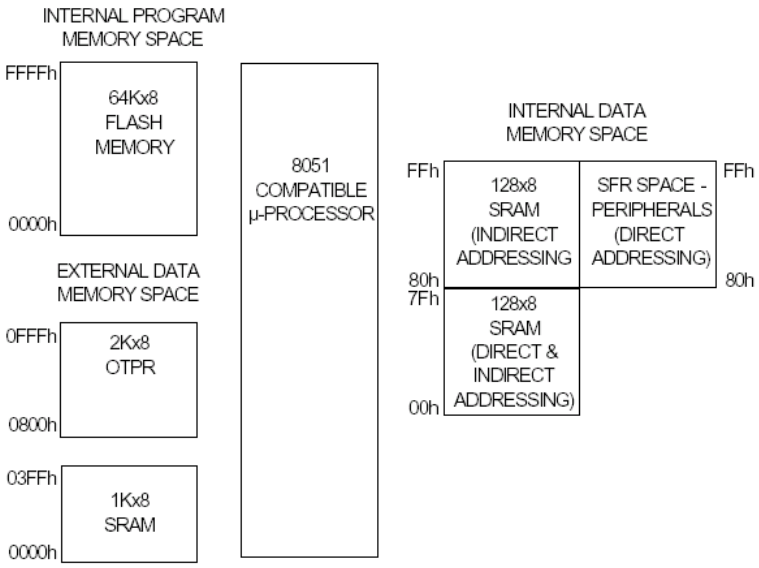

Abbildung 3 Programm- und Datenspeicher im VERSA1

In der Anwendungsschaltung präsentiert sich der VERSA1 Baustein mit den in Abbildung 4 gezeigten Anschlüssen.

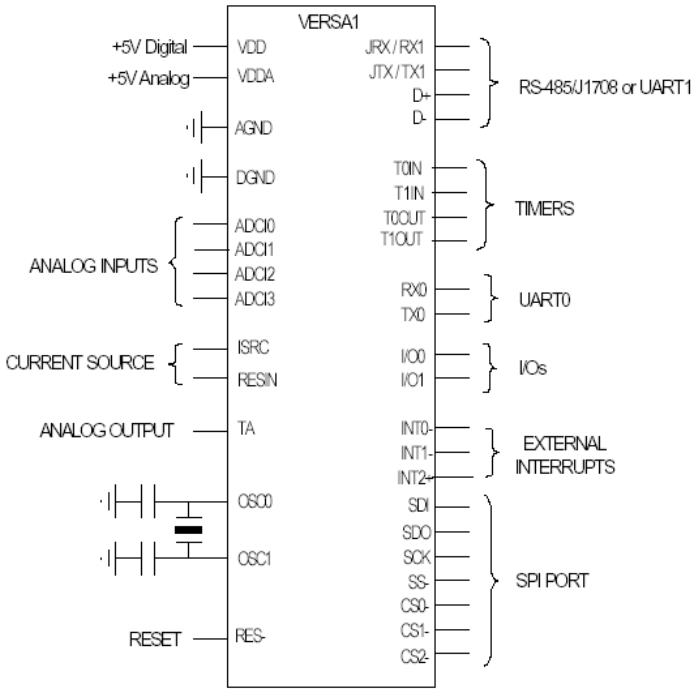

Abbildung 4 Pinbelegung des VERSA1

2. Serielle Schnittstellen

Der VERSA1 Baustein weist zwei unabhängige UARTs auf. Beim Entwicklungsboard arbeitet UART1 als Programmierschnittstelle. UART0 steht für die Applikation zur Verfügung und dient im folgenden zur Ausgabe von Statusinformation an ein angeschlossenes Terminal.

Wie in den Programmbeispielen noch gezeigt wird, soll die Schnittstelle mit 19200 Baud, acht Datenbits, einem Stoppbit und ohne Parität (19200,8,N,1) arbeiten.

Gegenüber dem klassischen 8051er Mikrocontroller gibt es hier keine nennenswerte Besonderheiten, weshalb auf die Programmbeispiele verwiesen wird.

3. Analoge Funktionseinheiten

Der integrierte A/D-Umsetzer stellt vier Kanäle extern zur Verfügung. Über vier weitere Kanäle lassen sich interne Spannungen überwachen. Ein Eingangsmultiplexer schaltet den gewünschten Kanal zum A/D-Umsetzer durch.

Eine interne Bandgap-Referenzspannung mit typisch 1,23 V und einem Temperaturkoeffizienten von 7 ppm/°C kann als Referenz für die A/D-Umsetzung dienen.

Eine programmierbare Stromquelle kann zwischen 33 µA und 133 µA umgeschaltet werden.

Ein Ausgangsmultiplexer kann verschiedene interne Spannungen an einen externen Anschluss führen.

Abbildung 5 zeigt die analogen Funktionsgruppen im VERSA1 in einem Blockschema.

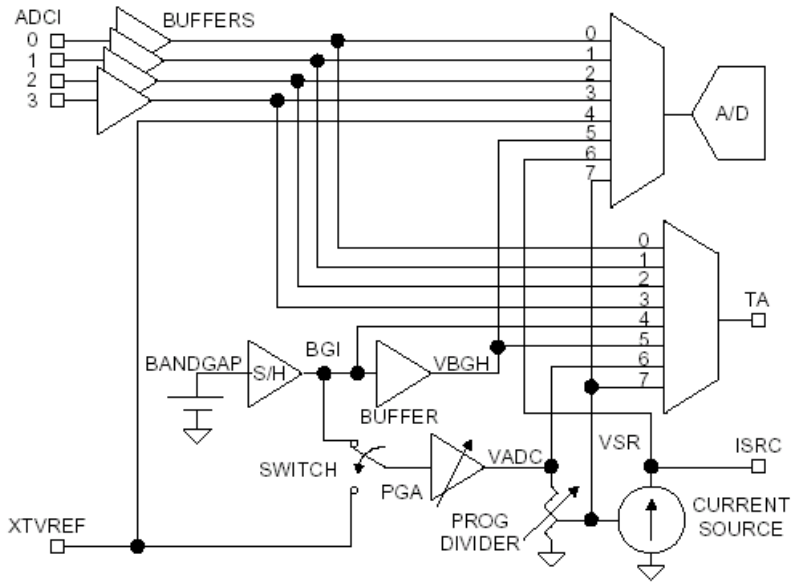

Abbildung 5 Analoge Funktionsgruppen

Neben der internen Bandgap-Referenz kann auch eine externe Referenzspannung am Eingang XTVREF als Referenz VADC für die AD-Umsetzung dienen.

Zur Verbesserung der Störsignalunterdrückung kann die interne Bandgap-Referenzspannung gefiltert werden. Hierzu kann das in Abbildung 6 gezeigt PI-Filter zwischen die Anschlüsse TA und XTVREF geschaltet werden.

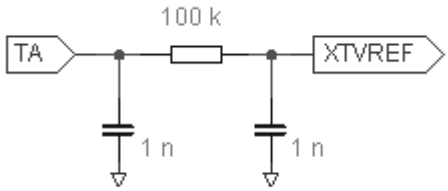

Abbildung 6 Filter für interne Bandgap-Referenz

Bei der Initialisierung der analogen Funktionsgruppen muss dann dafür gesorgt werden, dass Kanal 5 des Ausgangsmultiplexers auf den Anschluss TA geschaltet und XTVREF als Referenzspannung selektiert wird.

Jeder VERSA1 Baustein muss während des Herstellungsprozesses extensive Test- und Kalibierprozesse durchlaufen. Die Kalibrierdaten werden zum Ende der Tests im OTP-ROM abgelegt und können von da bei der Initialisierung in die betreffenden Special-Function-Register (SFR) geladen werden. Den Kalibrierdaten sind folgende Adressen im OTP-ROM zugeordnet.

OTP Adresse	*Kalibrierdaten*
$B01_H$	BGAP CAL DATA
$B05_H$	PGA CAL DATA
$B07_H$	ISRC 200mV Wert
$B09_H$	ISRC 800mV Wert

Der VERSA1 A/D-Umsetzer kennt folgende Betriebsarten

- Single-Shot Umsetzung für einen ausgewählten Kanal
- Single-Shot Umsetzung für alle vier Kanäle nacheinander
- Zyklische Umsetzung für einen ausgewählten Kanal
- Zyklische Umsetzung für alle vier Kanäle nacheinander

Die Auflösung des A/D-Umsetzers beträgt 12 Bit, so dass bei einem FullScale-Wert von 2,7 V die Quantisierungseinheit ca. 0,66 mV beträgt.

Die Conversion Rate Register legen die Rate für die zyklische Umsetzung fest. Es gilt die Beziehung:

Conversion Rate Register Value (24bits) = (External Clock / 2) / (CONV_RATE)

Bei einer Taktfrequenz von 16 MHz kann diese Rate für einkanaligen A/D-Umsetzung zwischen 0,5 Hz und 2 kHz eingestellt werden. Bei vierkanaliger Umsetzung beträgt die maximale Rate etwa 570 Hz.

Für eine präzise A/D-Umsetzung kann der Prozessortakt während der A/D-Umsetzung herabgesetzt oder ganz abgeschaltet werden. Der Interrupt am Ende der A/D-Umsetzung kann dann den Prozessor wecken und die Interrupt-Serviceroutine (ISR) ausführen.

Bevor der A/D-Umsetzer eingesetzt werden kann, muss noch der ADC Clock Divider auf einen Wert von etwa 250 kHz eingestellt werden. Es gilt hier die Beziehung:

ADC CLK REF = (External Clock / 2) / [(2 * ADCCDIV) + 2]

Bei einer Taktfrequenz von 16 MHz sollte das Register ADCCDIV auf den Wert 15 (&H0F) gesetzt werden.

Die folgenden Register bestimmen die Arbeitsweise der analogen Funktionsgruppen. Die Einzelheiten können hier nicht aufgeführt werden. Die Kommentare in den Programmbeispielen verdeutlichen aber die Programmierung. Für die Details muss auf das immerhin 62 Seiten umfassende Datenblatt verwiesen werden.

SFR	*Register*	*Bedeutung*
FE_H	ADCCDIV	ADC Clock Divider Register
FD_H	OUTMUX	Analog Output Mux Control Register
FC_H	INMUX	ADC Input Mux Control Register
FB_H	ISRC2	Current Source Control Register2
FA_H	ISRC1	Current Source Control Register1
$F9_H$	PGACTRL	PGA Control Register
$F7_H$	CONVRHI	Conversion Rate High Register
$F6_H$	CONVRMED	Conversion Rate Med Register
$F5_H$	CONVRLO	Conversion Rate Low Register
AF_H	ADCD3HI	ADC Channel 3 High Register
AE_H	ADCD3LO	ADC Channel 3 Low Register
AD_H	ADCD2HI	ADC Channel 2 High Register
AC_H	ADCD2LO	ADC Channel 2 Low Register
$A7_H$	ADCD1HI	ADC Channel 1 High Register
$A6_H$	ADCD1LO	ADC Channel 1 Low Register
$A5_H$	ADCD0HI	ADC Channel 0 High Register
$A4_H$	ADCD0LO	ADC Channel 0 Low Register
$9C_H$	ADCSTAT	ADC Status Register
97_H	ADCALDAT	ADC Calibrate Data Register
96_H	ADCALADR	ADC Calibrate Address Register
95_H	BGAPCAL	Bandgap Calibrate Data Register
94_H	ADCCTRL	ADC Control Register

4. Arithmetikeinheit MAC

Mit der integrierten Arithmetikeinheit (Multiplying Accumulator – MAC) können arithmetische Operationen stark beschleunigt werden. Die Programmbeispiele vermitteln einen Eindruck von der Geschwindigkeit der MAC-unterstützen Arithmetikoperationen.

Der MAC-Block arbeitet als separate Einheit und ist nur mit den Operanden zu laden, um anschließend das Ergebnis abzuholen. Der MAC-Block berechnet die folgende Beziehung

MACRES = (MACA * MACB) + MACACC

.

Abbildung 7 zeigt die Kopplung des MAC-Blocks an den Prozessor einschließlich der Formate von Operanden und des Resultats.

Das Ergebnis und der Akkumulator sind jeweils 32 Bit breit, während die Multiplikanden jeweils 16 Bit breit sind.

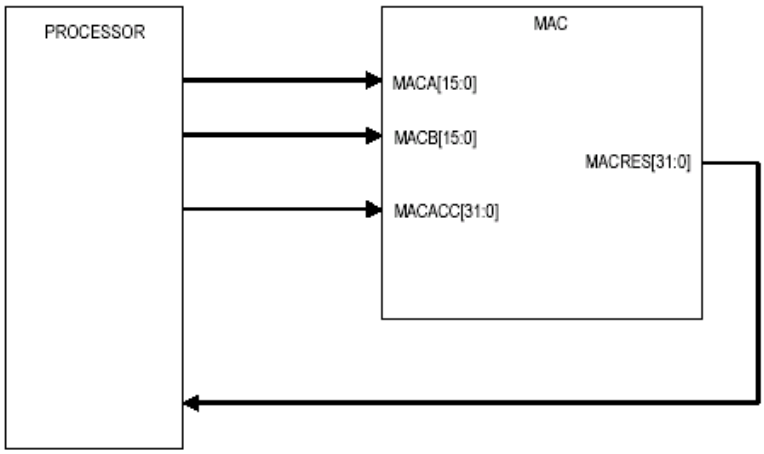

Abbildung 7 Kopplung des MAC-Blocks an den Prozessor

5. VERSA1 Entwicklungsboard

Mit dem VERSA1 Entwicklungsboard bietet GOAL eine einfache Inbetriebnahme- und Testmöglichkeit mit einem großen Prototypingbereich. Die Anschlusse des VERSA1 Mikrocontrollers sind über Pfostenleisten abgreifbar. Periphere Erweiterungen lassen sich so direkt auf dem Entwicklungsboard aufbauen. Unter www.goalasic.com/v1devkit.html kann die dazugehörige Dokumentation (18 Seiten / 1.1 MB) heruntergeladen werden. Abbildung 8 zeigt das VERSA1 Entwicklungsboard. In der Mitte ist der ZIF-Sockel für die Aufnahme des VERSA1 Bausteins zu erkennen, seitlich davon der Prototypingbereich.

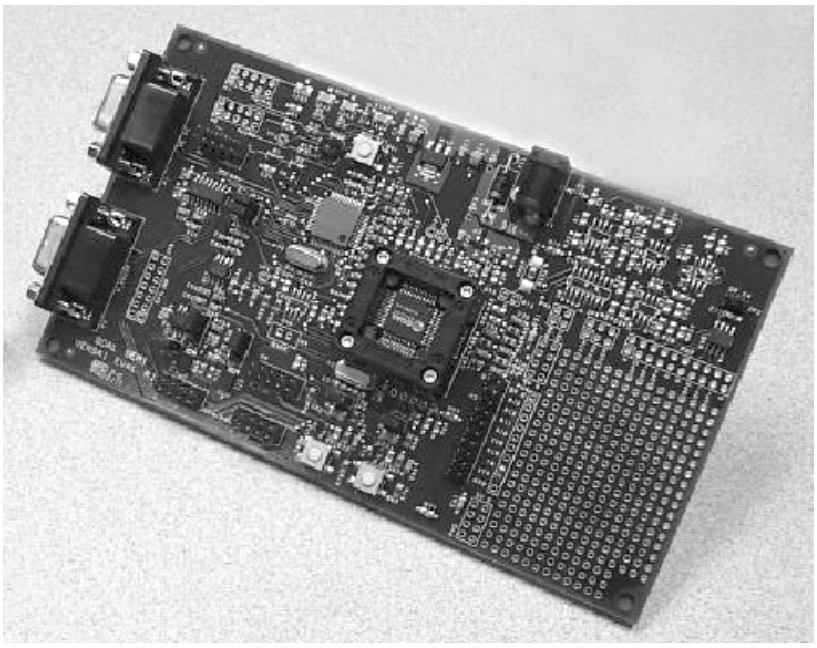

Abbildung 8 VERSA1 Entwicklungsboard

Zur Programmierung des VERSA1 Bausteins wird die Möglichkeit der In-System-Programmierung über die serielle Schnittstelle des VERSA1 Entwicklungsboards genutzt. Der GoalTender VERSA1 DevBoard Programmer ist im Lieferumfang des VERSA1 Entwicklungsboards enthalten.

Wie Abbildung 9 zeigt, ist das erzeugte Hex-File zu laden und kann dann durch Anklicken des Tastenfelds "Erase + Program" in den zu programmierenden VERSA1 Baustein heruntergeladen werden.

Abbildung 9 GoalTender VERSA1 DevBoard Programmer

Der Programmer kann als externer Programmer direkt in BASCOM-8051 eingebunden werden. Hierzu sind im Menü **Options>Programmer** die folgenden Eintragungen vorzunehmen (Abbildung 10). Der unter Program angegebene Pfad ist abhängig vom Ort der Installation der GoalTender Programmer Software.

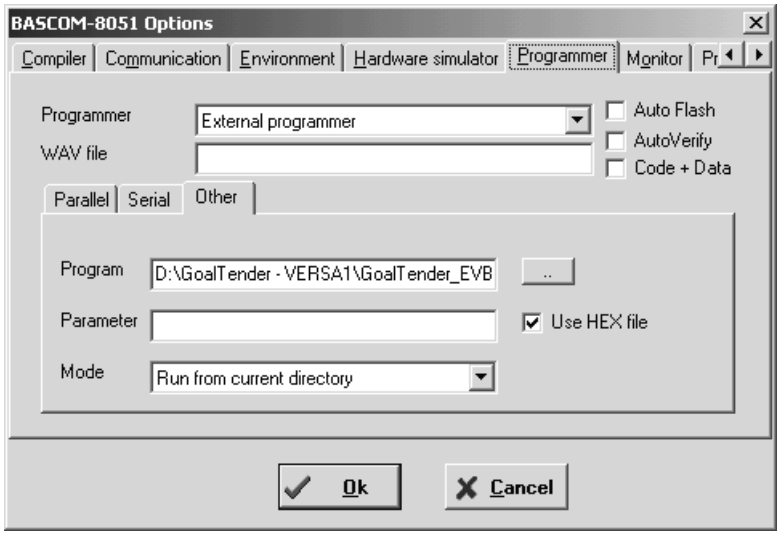

Abbildung 10 Externer Programmer in BASCOM-8051

6. Programmbeispiele

Die folgenden Programmbeispiele verdeutlichen die Programmierung der VERSA1 Bausteine mit BASCOM-8051.

Das Registerfile VERSA1.DAT stellt die Verfügbarkeit der zahlreichen SFRs sicher. Es ist momentan noch nicht im Lieferumfang von BASCOM-8051 enthalten, steht aber neben den Programmbeispielen auf der Begleit-CD bzw. beim Autor zum Download bereit [www.ckuehnel.ch/download.htm]. Die folgenden Programmbeispiele wurden alle mit BASCOM-8051 V.2.0.11.0 programmiert.

6.1. Hello World

Das Programm "Hello World" dient (wie so oft) als Test der Entwicklungsumgebung.

BASCOM-8051 nutzt per default Timer1 als Baudrategenerator. Üblicherweise stellt man mit BASCOM-8051 die Baudrate gemäss Listing 1 ein.

```
$regfile = "VERSA1.dat"
$crystal = 16257000                         ' Clock frequency of VERSA1 Devboard
$baud = 9600                                ' Baud rate UART0 is 4800 due to CLK/2

Do
  Print "Hello"
  Waitms 200
Loop

End
```

Listing 1 Programmbeispiel SERIAL.BAS

Beim Test des Programms muss man dann feststellen, dass die Baudrate nur 4800 Baud beträgt. Der Grund liegt beim VERSA1, dessen Timer mit der halben Clockfrequenz (hier also ca. 8 MHz) getaktet werden.

Wesentlich mehr Flexibilität erhält man, wenn Timer2 als Baudrategenerator eingesetzt wird. Nun muss allerdings die Initialisierung von Timer2 "von Hand" vorgenommen werden. Listing 2 zeigt das Programmbeispiel HW.BAS mit der Subroutine Iniser0t2i zur Initialisierung von Timer2. Diese Subroutine wurde hier in Assembler notiert und ist für den 8051 Standard.

```
$regfile = "VERSA1.dat"
$crystal = 16257000           ' Clock frequency of VERSA1 Devboard

Gosub Iniser0t2i              ' Initialize Serial Port 0

Print "Hello"                 ' Print to serial 0

End

Iniser0t2i:
$asm
   MOV     SCON,#&H5A         ' CONFIG SCON_0 MODE_1,
                              ' Define Reload Value For Timer 2
                              ' Rcap2h , Rcap2l = 65536 - [ Fcrystal /(64 * Fcomm)]
                              ' For Fosc = 8mhz(16mhz Ext)
   MOV     RCAP2H,#&HFF       ' MSB RELOAD FOR 19200bps @ 8MHz =FFh
   MOV     RCAP2L,#&HF3       ' LSB RELOAD FOR 19200bps @ 8MHz =F3h
   MOV     T2CON,#&H34        ' SERIALPORT0, TIMER2 RELOAD INTERNAL START
   CLR     SCON.0             ' CLEAR UART RI FLAG
   SETB    SCON.1             ' CLEAR UART TI FLAG
   RET
```

```
$end Asm
```

Listing 2 Programmbeispiel HW.BAS

6.2. AD-Umsetzer

Beim Einsatz des VERSA1 AD-Umsetzers kann nicht die Funktion GETAD() von BASCOM-8051 verwendet werden, da diese auf die Mikrocontroller 80515, 80535, 80517, 80535 und 80552 zugeschnitten ist. Im folgenden Programmbeispiel übernimmt die Subroutine ADC(Channel) diese Funktion.

In einer Endlosschleife wird Kanal #0 des AD-Umsetzers im Sekundentakt abgefragt und das Ergebnis über die serielle Schnittstelle an ein Terminalprogramm gesendet. Listing 3 zeigt die erforderlichen Schritte.

```
$regfile = "VERSA1.dat"
$crystal = 16257000                    ' Clock frequency of VERSA1 Devboard

Dim Channel As Byte
Dim Temp As Byte
Dim Adc_result As Word
Dim Sresult As String * 10

Declare Sub Adc(channel As Byte)

Gosub Iniser0t2i                       ' Initialize Serial Port 0
Gosub Initadc

Channel = 0                            ' Measure Channel #0
Print "AD Conversion on Channel #" ; Channel
Do
   Call Adc(channel)
   Sresult = Str(adc_result)
   Print "ADC Channel #" ; Channel ; " = " ; Sresult ; " (decimal)"
   Wait 1
Loop

End

Sub Adc(channel As Byte)
   Temp = Channel And &H07
   Shift Temp , Left , 4
   Inmux = Inmux Or Temp
   ORL     ADCCTRL,#&B00100000        'Start Single Shot Conversion
   While Adcstat.0 = 0 : Wend
   Temp = Varptr(adc_result)
   Poke Temp , Adcd0lo
   Incr Temp
   Poke Temp , Adcd0hi
End Sub


Initadc:
$asm
   MOV     ADCALDAT,#&H10  'SKIP AUTOCALIBRATION
   MOV     DPTR,#&H0B01    'RETRIEVE CALIBRATION DATA FROM OTPR
   MOVX    A,@DPTR
   MOV     BGAPCAL,A       'CALIBRATION BANDGAP

   MOV     DPL,#&H03       'RETRIEVE CALIBRATION DATA FROM OTPR
   MOVX    A,@DPTR
   MOV     INMUX,A         '[7] BANDGAP ORDER CORRECTION
```

```
   MOV     DPL,#&H05       'RETRIEVE CALIBRATION DATA FROM OTPR
   MOVX    A,@DPTR
   MOV     PGACTRL,A       'CALIBRATION PGA

   MOV     A,INMUX         '[7] BANDGAP ORDER CORRECTION
   ORL     A,#&H0F         '[6:4] ADC CONVERSION DONE ON 1ST FOUR CHANNELS
   MOV     INMUX,A         '[3:0] ACTIVATE 4 INPUT BUFFERS

   MOV     OUTMUX,#&H19    '[4] POWER DOWN SPARE AMPS
                           '[3 : 1] Bgi Routed To Ta
                           '[0] Outmux Enable

   MOV     ADCCDIV,#&H0F   'SET CLOCK 250kHz FOR ADC
   MOV     ADCCTRL,#&B11010001
   RET
$end Asm

Iniser0t2i:
$asm
   MOV     SCON,#&H5A   'CONFIG SCON_0 MODE_1,
                        ' Define Reload Value For Timer 2
                        ' Rcap2h , Rcap2l = 65536 - [ Fcrystal /(64 * Fcomm)]
                        ' For Fosc = 8mhz(16mhz Ext)
   MOV     RCAP2H,#&HFF 'MSB RELOAD FOR 19200bps @ 8MHz =FFh
   MOV     RCAP2L,#&HF3 'LSB RELOAD FOR 19200bps @ 8MHz =F3h
   MOV     T2CON,#&H34  'SERIALPORT0, TIMER2 RELOAD INTERNAL START
   CLR     SCON.0       'CLEAR UART RI FLAG
   SETB    SCON.1       'CLEAR UART TI FLAG
   RET
$end Asm
```

Listing 3 Programmbeispiel ADC.BAS

In der Subroutine ADC() ist der Eingangsmultiplexer zu setzen bevor der Prozess der AD-Umsetzung gestartet wird. Der Start der Umsetzung erfolgt hier durch die Assembleranweisung ORL ADCCTRL,#&B00100000, die das Startbit (ONESHOT) im Register ADCCTRL setzt. Das Bit ADCSTAT signalisiert das Ende der Umsetzung, wonach das Ergebnis der AD-Umsetzung aus den beiden Ergebnisregistern ADCD0LO und ADCD0HI in die Variable adc_result "gepoket" wird. Das direkte Beschreiben der Variablen adc_result ist wesentlich schneller, als die Wordvariable durch Laden und Schieben der beiden Bytevariablen zusammen zu setzen.

Der wesentlich spannendere Teil des Programmbeispiels ist die Initialisierung des AD-Umsetzers. Im Quelltext sind hierzu bereits zahlreiche Kommentare zu den einzelnen Registern zu finden.

Zu Beginn der Initialisierung des AD-Umsetzers wird die Autokalibrierung verhindert, da die im OTP-ROM abgelegten Kalibrationsdaten verwendet werden sollen.

Dann folgt das Laden der Kalibrationsdaten aus dem OTP-ROM beginnend ab Adresse &H0B01.

Da zur besseren Rauschunterdrückung die Bandgap-Referenzspannung über das externe PI-Filter geführt werden soll, ist der Ausgangsmultiplexer entsprechend zu schalten und freizugeben.

Schliesslich ist der Takt für die AD-Umsetzung auf einen Wert von ca. 250 kHz einzustellen. Daran anschliessend kann der Eingang XTVREF durchgeschaltet, die AD-Umsetzung auf einen vorgegebenen Kanal eingestellt sowie AD-Umsetzer und Bandgap-Referenz freigegeben werden.

Wie bei Initialisierungen üblich entscheidet die fehlerfreie Initialisierung der recht komplexen AD-Umsetzerbaugruppe über das Ergebnis der Umsetzung. Eine sorgfältige

Plausibilitätskontrolle der bei der Inbetriebnahme erhaltenen Werte schützt vor späteren Überraschungen.

Die Initialisierung der seriellen Schnittstelle zeigt nichts neues.

Bevor das Programm in die Endlosschleife eintritt, wird der Kanal des Eingangsmultiplexers zugewiesen. Bei der hier verwendeten Betriebsart des AD-Umsetzers wird das Ergebnis der AD-Umsetzung unabhängig vom selektierten Kanal immer im Registerpaar ADCD0 abgelegt.

Das Ergebnis der AD-Umsetzung wird schliesslich in der Form

ADC Channel #0 = 4095 (decimal)

für eine FullScale-Eingangsspannung über die serielle Schnittstelle ausgegeben.

Da mit dem AD-Umsetzer auch interne Spannungen überwacht werden können, kann durch Selektion von Kanal #5 die Bandgap-Referenz gemessen werden. Listing 4 zeigt das leicht angepasste Programmbeispiel ADC1.BAS. Da die Subroutines unverändert sind, wurden aus Platzgründen die betreffenden Quelltexte durch ... ersetzt.

```
$regfile = "VERSA1.dat"
$crystal = 16257000                  ' Clock frequency of VERSA1 Devboard

Dim Channel As Byte
Dim Temp As Byte
Dim Adc_result As Word
Dim Sresult As String * 10
Dim Voltage As Single

Declare Sub Adc(channel As Byte)

Gosub Iniser0t2i                     ' Initialize Serial Port 0
Gosub Initadc

Channel = 5                          ' Measure BandGap Reference
Print "AD Conversion of Badgap Reference"
Do
   Call Adc(channel)
   Sresult = Str(adc_result)
   Print "ADC Channel #" ; Channel ; " = " ; Sresult ; " (decimal)"
   Voltage = Adc_result * 2.7
   Voltage = Voltage / 4096
   Sresult = Str(voltage)
   Print "Badgap Reference Voltage = " ; Sresult ; " V"
   Wait 1
Loop

End

Sub Adc(channel As Byte)
...
End Sub

Initadc:
$asm
...
$end Asm

Iniser0t2i:
$asm
...
$end Asm
```

Listing 4 Programmbeispiel ADC1.BAS

Nennenswert in Listing 4 ist die veränderte Auswertung des Ergebnisses der AD-Umsetzung.

Hier wird durch die Singlevariable Voltage das Ergebnis skaliert und in der in Abbildung 11 gezeigten Form dargestellt.

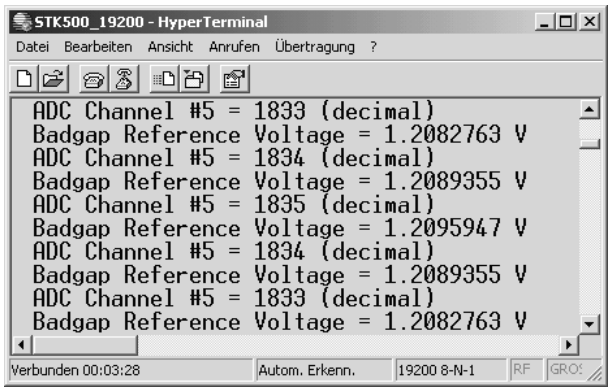

Abbildung 11 AD-Umsetzung der Bandgap-Referenz

Diese beiden Programmbeispiele können als Vorlage für eigene Routinen zur AD-Umsetzung dienen.

Bei der hohen Auflösung ist eine entsprechende Sorgfalt beim Schaltungsaufbau mit zusätzlichen externen Komponenten wie Spannungsteilern und/oder Anpassverstärkern erforderlich. Anderenfalls kann man nur einen Bruchteil der angebotenen Performance nutzen.

6.3. Arithmetikeinheit

Berechnungen mit der Arithmetikeinheit beschränken sich auf das Beschreiben und Lesen der betreffenden Register des MAC-Blocks. Mit den folgenden beiden Programmbeispielen soll deshalb der Performancegewinn untersucht werden.

Das in Listing 5 gezeigte Programm MAC.BAS ermittelt die für eine Multiplikation von zwei Longvariablen benötigte Rechenzeit. Mit Hilfe des Programms LookRS232, welches den von der seriellen Schnittstelle empfangenen Daten Zeitstempel zuordnet, kann die Laufzeit ermittelt werden. Das in Abbildung 12 geöffnete About Window zeigt die Website des Hersteller von LookRS232, wo man eine freie Evaluationversion downloaden kann.

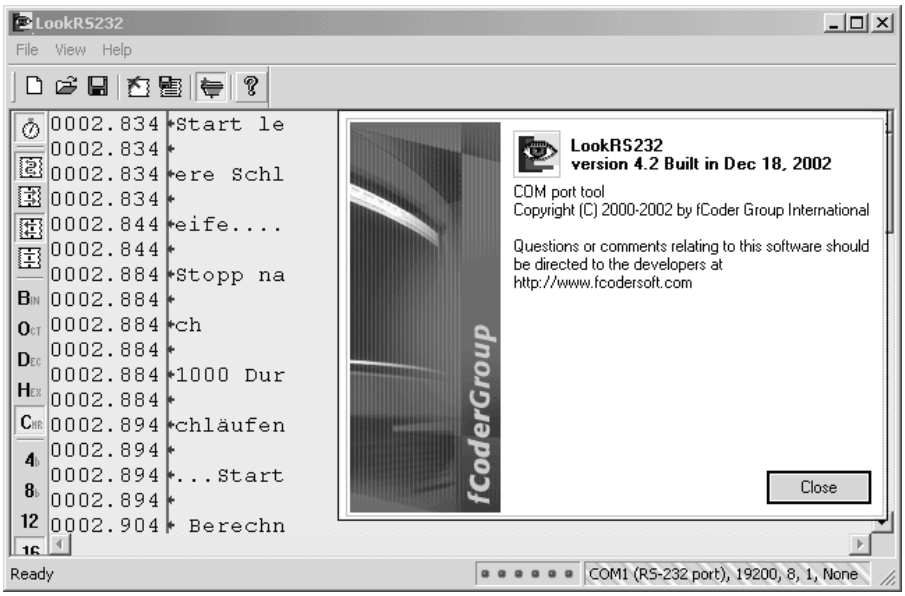

Abbildung 12 Mitschnitt durch LookRS232

Hat man ein solches Programm nicht zur Verfügung, dann erhöht man einfach die Zahl der Schleifendurchläufe und verwendet eine Stoppuhr zur überschlägigen Zeitmessung.

Im Programm MAC.BAS wird nach Deklaration der Variablen und der bekannten Initialisierung der seriellen Schnittstelle eine Subroutine deklariert, die eine Multiplikation von zwei Wordvariablen (16 Bit) zu einem Longergebnis (32 Bit) ausführt. In der Subroutine selbst erkennt man nur die Schreibbefehle der MAC-Register, sowie den Aufruf der Adresse der Ergebnisvariablen für das anschließende Poken der Ergebnisbytes.

Um eine vernünftige Zeitmessung zu ermöglichen, muss die Multiplikation in einer Schleife mehrfach wiederholt werden. Im Programm wird mit 1000 Schleifendurchläufen gearbeitet.

Zuerst wird eine leere Schleife durchlaufen, damit die nicht von der Multiplikation verbrauchte Zeit in den Betrachtungen berücksichtigt werden kann. Dann folgen eine konventionelle Berechnung und die Berechnung mit der Arithmetikeinheit.

```
$regfile = "VERSA1.dat"
$crystal = 16257000

Dim S As String * 10
Dim X As Long
Dim Y As Long
Dim Z As Long
Dim Addr As Byte
Dim I As Word

Gosub Iniser0t2i

Declare Sub Mul(x As Word , Y As Word)

X = 12345
Y = 12345

Print "Start leere Schleife.."
For I = 1 To 1000
Next
S = Str(z) : Decr I
Print "Stopp nach " ; I ; " Durchläufen."

Print "Start Berechnung konventionell.."
```

```
For I = 1 To 1000
   Z = X * Y
Next
S = Str(z) : Decr I
Print "Stopp nach " ; I ; " Berechnungen."
Print X ; " * " ; Y ; " = " ; S

Z = 0

Print "Start Berechnung über MAC.."
For I = 1 To 1000
   Call Mul(x , Y)
Next
S = Str(z) : Decr I
Print "Stopp nach " ; I ; " Berechnungen."
Print X ; " * " ; Y ; " = " ; S

End

Sub Mul(x As Word , Y As Word)
   Maca0 = Low(x)
   Maca1 = High(x)
   Macb0 = Low(y)
   Macb1 = High(y)
   Addr = Varptr(z)
   Poke Addr , Macres0 : Incr Addr
   Poke Addr , Macres1 : Incr Addr
   Poke Addr , Macres2 : Incr Addr
   Poke Addr , Macres3
End Sub

Iniser0t2i:
$asm
...
$end Asm
```

Listing 5 Programmbeispiel MAC.BAS

Die Ergebnisse der Laufzeitmessungen sind in Tabelle 1 zusammengestellt. Unter Einsatz der Arithmetikeinheit zeigt sich eine deutliche Verbesserung des Laufzeitverhalten für die hier untersuchte Multiplikation.

Laufzeit MAC.BAS	*Laufzeit [s]*	*Laufzeit netto [s]*	*Faktor*
Leere Schleife	4.00E-05		
Konventionelle Multiplikation	1.60E-03	1.56E-03	
Multiplikation über MAC	8.00E-05	4.00E-05	39.05

Tabelle 1 Laufzeitmessungen beim Programm MAC.BAS

Filterfunktionen, die bei der Messwerterfassung häufig gefordert werden, lassen sich mit dieser Arithmetikeinheit vorteilhaft implementieren.

Im zweiten Programmbeispiel MAC1.BAS wird untersucht, ob sich der Wert der Operanden auf die Laufzeit auswirkt. Listing 6 zeigt das entsprechend Programmbeispiel.

```
$regfile = "VERSA1.dat"
$crystal = 16257000

Dim S As String * 10
Dim X As Long
Dim Y As Long
```

```
Dim Z As Long
Dim Addr As Byte

Gosub Iniser0t2i

Declare Sub Mul(x As Word , Y As Word)

X = 1 : Y = 2
Z = X * Y
S = Str(z)
Print "Konventionelle Berechnung: ";
Print X ; " * " ; Y ; " = " ; S

Z = 0
Call Mul(x , Y)
S = Str(z)
Print "MAC Berechnung: ";
Print X ; " * " ; Y ; " = " ; S

X = 1000 : Y = 2000
Z = X * Y
S = Str(z)
Print "Konventionelle Berechnung: ";
Print X ; " * " ; Y ; " = " ; S

Z = 0
Call Mul(x , Y)
S = Str(z)
Print "MAC Berechnung: ";
Print X ; " * " ; Y ; " = " ; S

End

Sub Mul(x As Word , Y As Word)
   Maca0 = Low(x)
   Maca1 = High(x)
   Macb0 = Low(y)
   Macb1 = High(y)
   Addr = Varptr(z)
   Poke Addr , Macres0 : Incr Addr
   Poke Addr , Macres1 : Incr Addr
   Poke Addr , Macres2 : Incr Addr
   Poke Addr , Macres3
End Sub

Iniser0t2i:
$asm
...
$end Asm
```

Listing 6 Programmbeispiel MAC1.BAS

Hier wurde im Simulator die jeweilige Anzahl von Maschinenzyklen zur Laufzeitbestimmung verwendet. Wie Tabelle 2 zeigt, spielt die Größe der Operanden für die Rechenzeit der Multiplikation praktisch keine Rolle.

Laufzeit MAC1.BAS	*Zyklen für z=1*2*	*Faktor*	*Zyklen für z=1000*2000*	*Faktor*
Konventionelle Multiplikation	2239		2384	
Multiplikation über MAC	37	60.51	39	61.13

Tabelle 2 Laufzeitmessung beim Programm MAC1.BAS

7. Schlussbemerkung

Mit dem VERSA1 Baustein steht ein sehr leistungsfähiger Mikrocontroller zur Verfügung, der gerade für die Messwerterfassung in hohem Masse interessant sein dürfte.

Durch die bestehenden Interfacemöglichkeiten kann sehr einfach ein Zwei-Prozessorsystem aufgebaut werden, in dem der VERSA1 das analoge Frontend bildet und beispielsweise über eine schnelle SPI-Verbindung mit einem Host-Mikrocontroller mit externen Speichermöglichkeiten und/oder Ethernet-Interface ein Gesamtsystem bildet. Über das RS-485 kann aber ebenso ein sensorseitiges Controller-Netzwerk aufgebaut werden.

Die für ein Single-Controller-System unter Umständen etwas knapp bemessene digitale I/O kann sehr einfach über digitale Schieberegister ergänzt werden.