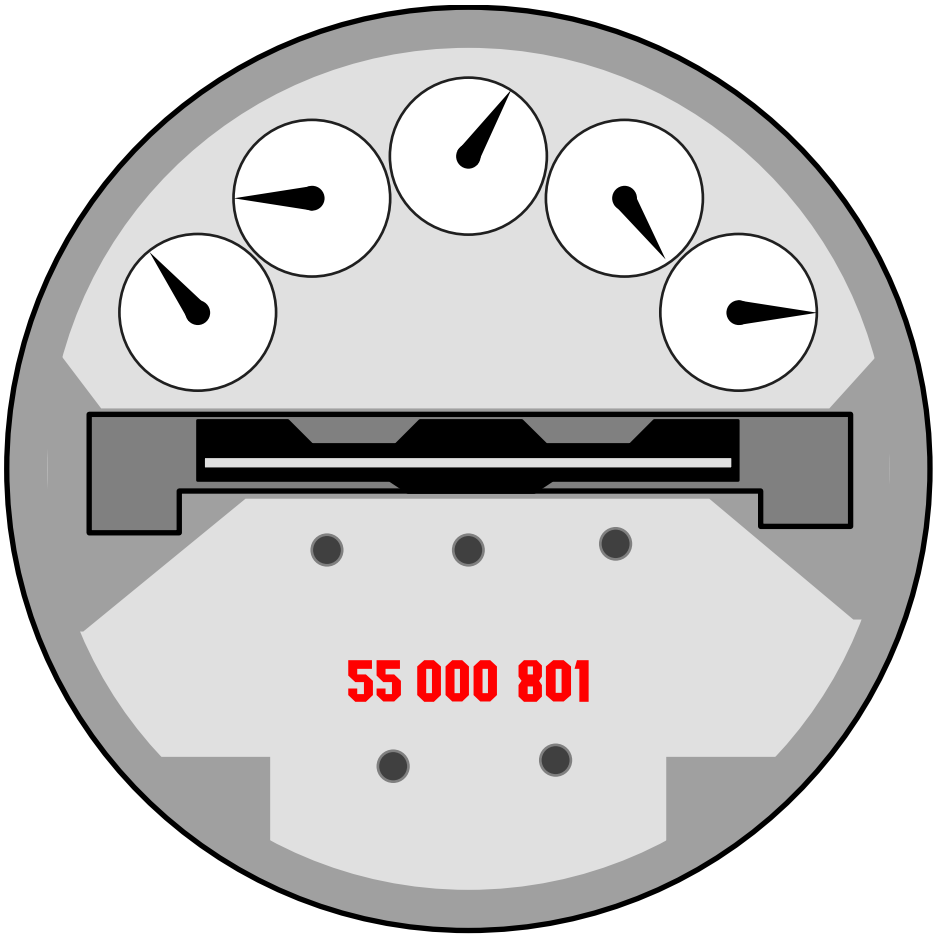


MODUL 4

ANALOG-DIGITAL UMSETZER des 80C517(A)

V1.2 1997 J. Humer

INHALTSVERZEICHNIS
MODUL 4
ANALOG-DIGITAL UMSETZER
des 80C517

Inhalt Seite

1

1. ANALOG - DIGITAL Umsetzer (ADU)

1.1 Allgemeines:

Ein ADU (engl. ADC Analog to Digital Converter) mißt das Verhältnis eines analogen Eingangssignals *a* zu einer Referenzgröße a_r und gibt dieses Verhältnis in Form eines digitalen Worts an.
ADUs finden vor allem bei der Signalverarbeitung Verwendung. Müssen z.B.: Sensorsignale, die ja physikalische Größen wie Druck, Temperatur, Feuchte, Längenänderung etc. messen und in eine proportionale elektrische Größe (Spannung oder Strom) umwandeln eingelesen und mittels Computer weiterverarbeitet werden, so wird der ADU deshalb benötigt, um die analoge Eingangsgröße in ein digitales Wort umzusetzen.

Der mögliche Eingangsspannungsbereich wird dazu in *n* gleich große Teile zerlegt und es wird dann festgestellt, in welchem dieser Intervalle die Eingangsgröße liegt.
Den Intervallen kann ein beliebiger Kode zugeordnet werden. Sinnvollerweise wird zumeist allerdings der duale Kode verwendet.

Die AD Umsetzung besteht also prinzipiell aus zwei grundlegenden Schritten:

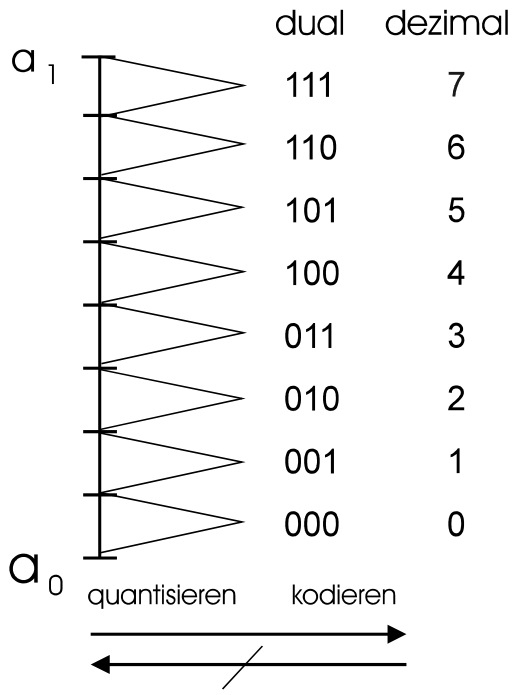
1. **Quantisieren**
2. **Kodieren**

ACHTUNG:

☝ *Durch die **Quantisierung** (Zuordnung zu den Teilintervallen) erfolgt ein Informationsverlust, der nicht mehr rückgängig gemacht werden kann !*

Dieser Verlust an Information kommt dadurch zustande, daß die Unterteilung der Intervalle nicht unendlich klein werden kann und so bei der Zuordnung zu den „am besten passenden Intervallen“ zwangsläufig ein Fehler entsteht.

Beispiel: ADU mit 3 Bit Auflösung ergibt 8 Teilintervalle ($2^3 = 8$)



Die Eingangsgröße **a0 ≤ a ≤ a1** wird durch eines der 8 möglichen Kodewörter abgebildet.

1.2 Einige Kenngrößen, die einen ADU beschreiben

• Auflösung:

Beschreibt die kleinste Änderung der Eingangsspannung, die zu einem Wechsel des niederwertigsten Bit (LSB = Least significant Bit) des Ausgangskodes führt.
Die Auflösung kann nun entweder als Stellenzahl in Bit oder in Prozent des Aussteuerbereichs (Eingangsspannungsbereich) angegeben werden.

Der 3 Bit ADU aus unserem vorigen Beispiel hat also eine Auflösung von 8 Bit; entsprechend hätte ein 10 Bit ADU (z.B. interner ADU des Mikrokontrollers 80C552) eine Auflösung von 2^{10} (=1024 Intervalle) bzw. 0,097% des Eingangsspannungsbereichs.

Die kleinste Spannungsauflösung beträgt bei 10 Bit Auflösung und 5V Eingangsspannungsbereich also **5V / 2 ^{10} = 5V / 1024 ≈ 5mV**.

Bezogen auf unser Beispiel des 3 Bit ADU ergibt sich bei einem Eingangsspannungsbereich von 5V die Spannungsauflösung von **5V / 2^3 = 625 mV.**
Das bedeutet, daß sich die Eingangsspannung um 625 mV ändern muß, um eine Änderung des LSB zu erreichen.

Achtung: Die Auflösung entspricht nicht automatisch der Genauigkeit, denn in die Genauigkeit des ADU gehen weitere Kenngrößen ein, wie :

- **Quantisierungsfehler:**
  entsteht durch die treppenförmige Umsetzfunktion zwischen analogem Signal und digitalem Wort.

- **Quantisierungsgeräusch** (auch Quantisierungsrauschen)
  entsteht durch die sägezahnförmige Fehlerspannung bei der Umsetzung.

- **Linearitätsfehler:**
  Maximale Abweichung der realen Kennlinie von der idealen Kennlinie.

- **Nullpunktsfehler (und Meßbereichsendwertfehler):**
  Kennlinie verläuft nicht genau durch den Nullpunkt bzw. Aussteuerbereichsendwert.

- **Umsetzzeiten:**
  Die für die Umsetzung der analogen Eingangsgröße in ein digitales Wort benötigte Zeit.

2 Analog - Digital Umsetzer des 80C517

2.1 Interne Merkmale des ADU 80C517 (Siemens)

- *8 Bit Sampling ADU nach dem Prinzip der Succzessiven Approximation*
- *12 gemultiplexte Analogeingänge die auch als digitale Eingänge benutzt werden können.*
- *programmierbare Referenzspannungsquellen (in 16 Stufen)*
- *13µs Umsetzzeit bei 12Mhz Taktfrequenz*
- *der Start der Umsetzung kann intern oder extern ausgelöst werden*
- *Generierung eines Interrupt nach jeder beendeten Umsetzung*

2.2 Erklärungen

- *Sampling* das Eingangssignal wird abgetastet, zwischengespeichert und dem ADU zugeführt.

- *gemultiplext* Es steht für die 12 Eingänge nur ein Umsetzer zur Verfügung, der Multiplexer übernimmt die Zuschaltung des jeweils angewählten Eingangspins (Kanals) auf den ADU.

- *Successiven Approximation*

  auch *Wägeverfahren* genannt, ist eines der 3 Verfahren der AD Umsetzung. (Paralellverfahren, Wägeverfahren, Zählverfahren)
  Die (gesamplete) abgetastete Eingangsspannung wird mittels DA-Umsetzer und Komparator mit dem vorigen Wandlungsergebnis verglichen, und entsprechend dem Ergebnis, (größer oder kleiner als der vorige Wert) wird ein Bit im *Succzessiven Approximation Register* gesetzt bzw. rückgesetzt. Das wird vom höchstwertigen Bit (MSB) bis zum niederwertigsten Bit in gleicher Weise durchgeführt. Am Ende dieser Vergleiche steht im Register eine Zahl, die DA-Umgesetzt (innerhalb der Auflösungsgrenzen) der Eingangsspannung entspricht.

2.3 Blockschaltbild ADU 80C517

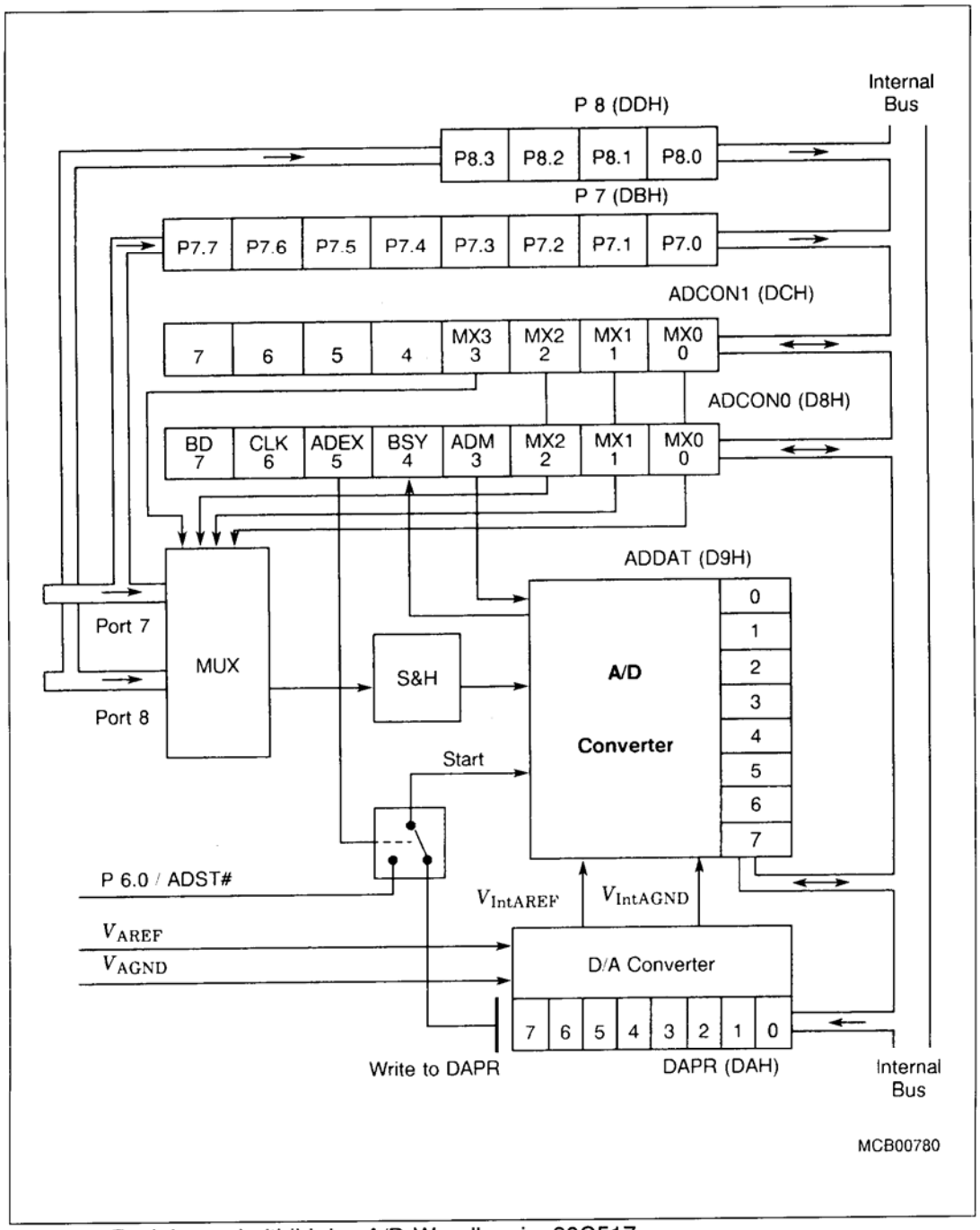


Bild 10-1: Funktionsschaltbild des A/D-Wandlers im 80C517

2.4 Initialisierung und Wahl des Eingangskanals

Es existieren beim 80C517 zwei *Special Function Register (SFR)* **ADCON0** und **ADCON1**, durch deren Beschreiben es möglich ist, zwei Arbeitsmodi auszuwählen, den Status zu lesen und den gewünschten Eingangspin (Portpin) auf den ADU zu schalten.

ADCON0:

BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0

- mit den 3 BIT *MX0 MX1 MX2* kann nun einer der 8 Eingangskanäle (Kanal 0 - 7) ausgewählt werden.

- ***ADM*** (ADU Mode) ist dieses Bit gesetzt (= 1) so befindet sich der ADU in einem Modus in dem er kontinuierlich umsetzt.
  Ist dieses Bit 0, so stoppt der ADU nach jeder Umsetzung.

- ***BSY*** (Busy Flag) dieses Bit zeigt an ob sich der ADU in Ruhe befindet oder gerade eine Konversion durchführt.
  BSY = 1 Umsetzung wird gerade durchgeführt
  Ist die Umsetzung beendet und liegt das Ergebnis vor, so wird das Bit von der Hardware zurückgesetzt.

- ***ADEX*** *(ADU extern)* Interner bzw. externer Start einer AD Umsetzung
  Wenn das ADEX Bit gesetzt ist, kann die Konversion am P6.0 /ADST# gestartet werden.

ADCON1 :

Die verbleibenden 4 Kanäle beziehungsweise alle 12 Kanäle können durch die 4 Bit *MX0 - MX3* als aktueller Eingang im *SFR **ADCON1*** gewählt werden.

Es kann also entweder jeder der 12 Kanäle in ADCON1 gewählt werden, oder die unteren 8 Kanäle in ADCON 0.

2.5 Auswahl der Eingangskanäle

MX3	MX2	MX1	MX0	KANAL	PIN
0	*0*	*0*	*0*	*0*	**P7.0**
0	*0*	*0*	*1*	*1*	**P7.1**
0	*0*	*1*	*0*	*2*	**P7.2**
0	*0*	*1*	*1*	*3*	**P7.3**
0	*1*	*0*	*0*	*4*	**P7.4**
0	*1*	*0*	*1*	*5*	**P7.5**
0	*1*	*1*	*0*	*6*	**P7.6**
0	*1*	*1*	*1*	*7*	**P7.7**
0	*X*	*0*	*0*	*8*	**P8.0**
0	*X*	*0*	*1*	*9*	**P8.1**
1	*X*	*1*	*0*	*10*	**P8.2**
1	*X*	*1*	*1*	*11*	**P8.3**

Liegt nun ein Ergebnis einer Umsetzung vor, so wird dieses Ergebnis im Register ***ADDAT*** gespeichert.
Dieses Ergebnis ist nun eine Umsetzung des 0 - 5 Volt Eingangssignals mit einer Auflösung von 8 Bit (Spannungsauflösung 5 mV).

Durch einen Trick kann beim 80C517 die Auflösung von 8 Bit auf maximal 10 Bit erhöht werden. Der Mikrokontroller 80C517A hat bereits einen 10 Bit ADU mit 12 gemultiplexten Eingangskanälen integriert.

Ein weiterer µC mit 10 Bit ADU ist der 80C552 der neben einem 8 Kanal ADU auch ein I²C Bus Interface besitzt.

2.6 Erhöhung der Umsetzerauflösung von 8 Bit auf 10 Bit:

1. man bestimmt zuerst den Wert der Eingangsspannung mit Referenzspannungen von 0 und 5V.

2. man verschiebt je nach Größe der Eingangsspannung die Referenzspannungen und reduziert somit den Umsetzungsbereich .

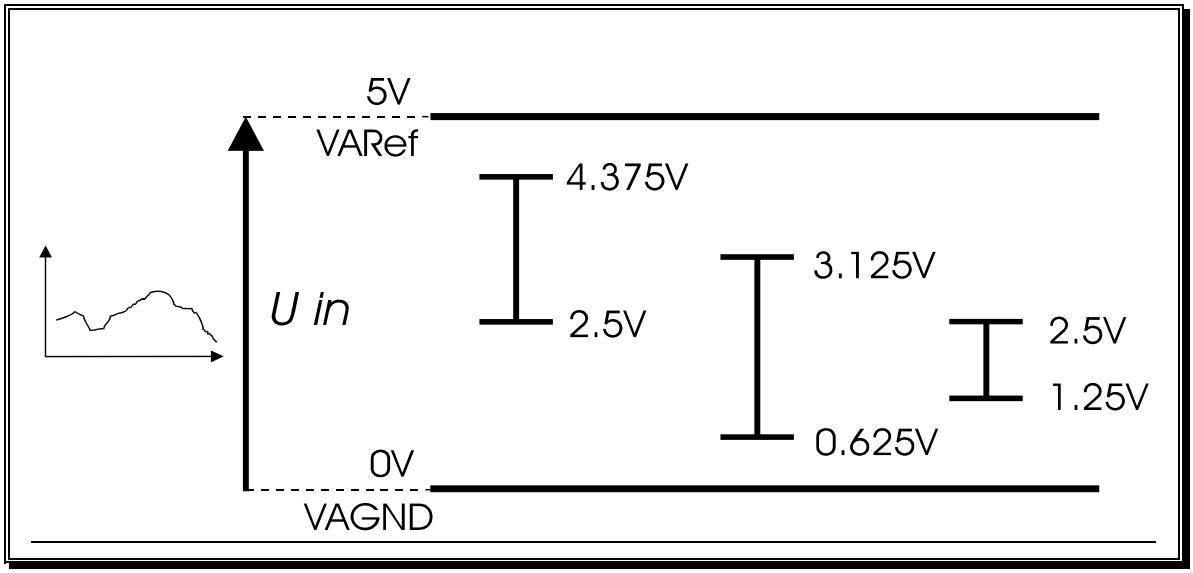
3. man führt eine zweite Umsetzung in diesem eingegrenzten Bereich durch und erhält in diesem eingegrenzten Bereich wieder 8 Bit Auflösung.

☞ *Die kleinste Intervallgröße ist 1/4 des Referenzspannungsbereichs also 1,25 V.*

- Bei 8 Bit und 5V Referenzspannung beträgt die Auflösung (engl. resolution) demnach $5 / 2^8 = 19{,}5$ mV.

- Bei 8 Bit und 1.25V Referenzspannung beträgt die Auflösung $1.25V / 2^8 = 4{,}88$ mV. Das entspricht 10 Bit Auflösung bei 5V Referenzspannung, denn $5V/ 2^{10} = 4{,}88$ mV.

2.7 Anpassen der Referenzspannungen an die Eingangsspannung:



2.7.1 Interne einstellbare Referenzspannungen:

Die internen Referenzspannungen **VintAGND** und **VintAREF** können in 1/16 Schritten im Bereich der äußeren Referenzspannungen VAREF - VAGND programmiert werden. In unserem Fall wäre dies 5V - 0V. 1/16 Schritt entspricht also wie aus der nachfolgenden Tabelle zu entnehmen 0,3125V.
Der minimale Abstand der beiden Referenzspannungen muß allerdings mindestens 1V betragen um eine einwandfreie Funktion des AD Umsetzers zu gewährleisten.
Das entspricht also bei 5V mindestens 4 (1/16) Schritte Unterschied.
Durch Schreiben ins Register *DAPR* werden diese Referenzspannungen programmiert.

Bits 4 - 7	Bits 0 -3
VintAREF	**VintAVGND**

Schritt	DAPR (.3 - .0) DAPR (.7 - .4)	Vint AGND	Vint AREF
0	*0000*	*0.0*	*5.0*
1	*0001*	*0.3125*	*-*
2	*0010*	*0.625*	*-*
3	*0011*	*0.9375*	*-*
4	*0100*	*1.25*	*1.25*
5	*0101*	*1.5625*	*1.5625*
6	*0110*	*1.875*	*1.875*
7	*0111*	*2.1875*	*2.1875*
8	*1000*	*2.5*	*2.5*
9	*1001*	*2.8125*	*2.8125*
10	*1010*	*3.125*	*3.125*
11	*1011*	*3.4375*	*3.4375*
12	*1100*	*3.75*	*3.75*
13	*1101*	*-*	*4.0625*
14	*1110*	*-*	*4.375*
15	*1111*	*-*	*4.68754*

3 Analog - Digital Umsetzer 517A

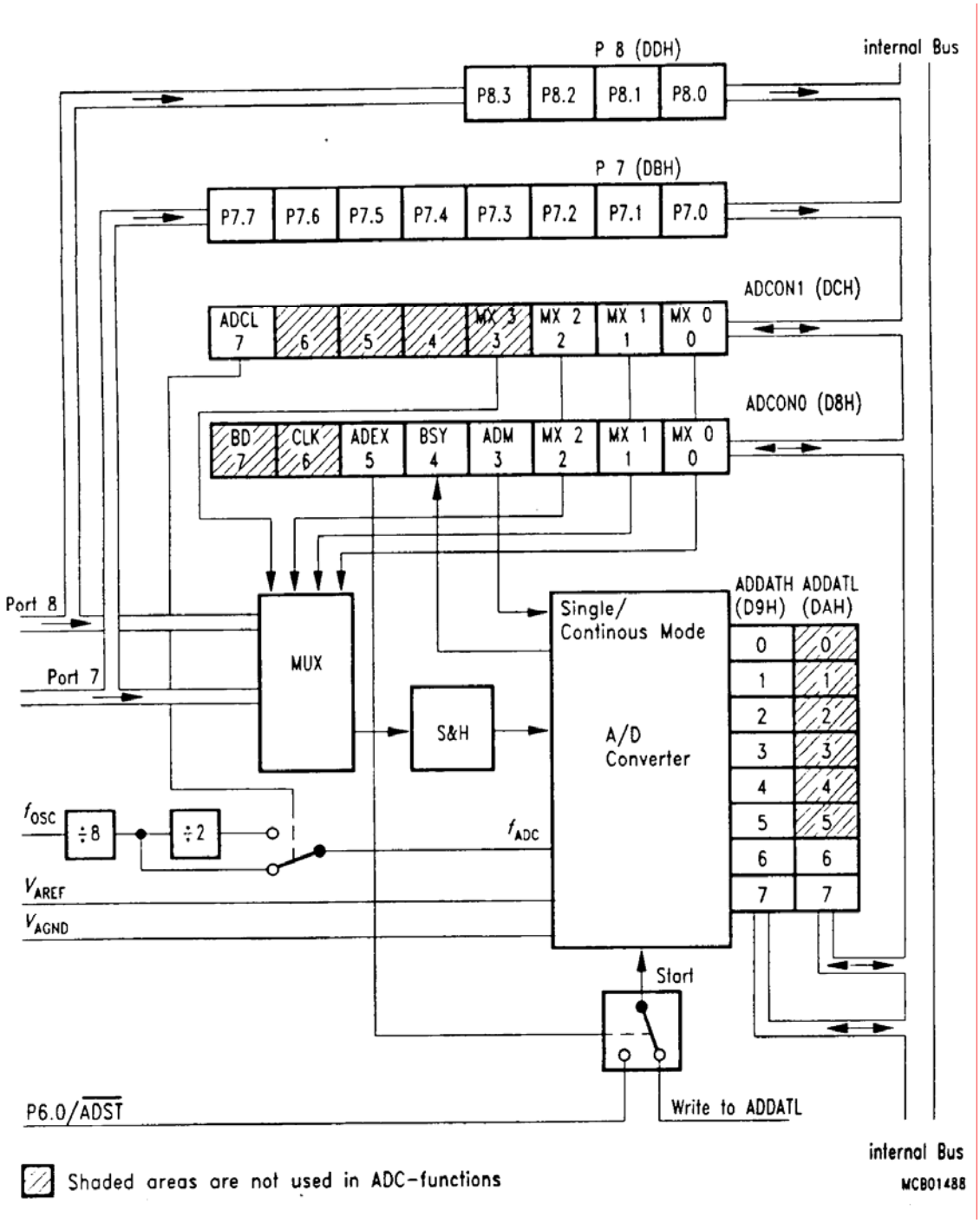
3.1 Allgemeines

Der Analog-Digital-Umsetzer im 517A ist wesentlich verbessert worden. Intern wird nach jeder Konversion eine Kalibrierung durchgeführt. Dieser Vorgang führt natürlich zu einem besseren Ergebnis als bei der Type 517. Obwohl beide Typen eine Auflösung von 10 bit erreichen hat der A-Type eine bessere Genauigkeit sowie höhere Umsetzgeschwindigkeit aufzuwarten.

Eckdaten:

1) 12 ANALOGE Eingangskanäle, Programmkompatibel zu 517
2) Alle analogen Eingänge können auch als Digital In verwendet werden.
   (siehe Tastatur)
3) 10 bit Ergebnis, kein Einstellen der Referenzspannung mehr.
4) Wandlungsgeschwindigkeit ist einstellbar, dies erlaubt eine Anpassung an die benutzte Quarzfrequenz, um immer kürzeste Wandlungszeiten (minimal 7us) zu erreichen.

3.2 Blockschaltbild



Blockschaltbild des ADU im 80C517A

3.2.1 Steuerregister ADCON0

ADCON0 (D8H), bitadressierbar

MSB							LSB
BD	CLK	**BSY**	**ADM**	**ADEX**	**MX2**	**MX1**	**MX0**
DFH	DEH	DDH	DCH	DBH	DAH	D9H	D8H

A/D-Wandler-Steuerregister 0 (A/D Converter Control Register 0). Es enthält Steuerbits für den A/D-Wandler und die serielle Schnittstelle 0.
Reset-Wert: 00H

Bitsymbol	Funktion
ADEX	Selektionsbit für externen Start der A/D-Wandlung. Wenn es gesetzt ist, wird eine A/D-Wandlung durch eine fallende Flanke am Portpin P6.0/ADST# ausgelöst.
BSY	Busy-Flag. Es ist während der Wandlung gesetzt und wird von der Hardware gesteuert.
ADM	Betriebsart des A/D-Wandlers (A/D Converter Mode). Es legt fest, ob eine einmalige (ADM = 0) oder Dauerwandlung (ADM = 1) durchgeführt wird.
MX0 MX1 MX2	Auswahl des Analogkanals (siehe Bild 10-4)

3.2.2 Steuerregister ADCON1

ADCON1 (DCH), nicht bitadressierbar

MSB							LSB
ADCL	-	-	-	**MX3**	**MX2**	**MX1**	**MX0**

A/D-Wandler-Steuerregister 1 (A/D Converter Control Register 1). Es enthält die Kanalauswahlbits MX0 bis MX3. Die Bits MX0 bis MX2 können wahlweise sowohl in ADCON1 als auch in ADCON0 geschrieben oder gelesen werden. Intern sind diese Bits jeweils miteinander verbunden.
Reset-Wert: 0xxx 0000B

Bitsymbol	Funktion
ADCL .	Takt für den A/D-Wandler (A/D Converter Clock) ADCL = 0: $f_{ADC} = f_{osz} / 8$ ADCL = 1: $f_{ADC} = f_{osz} / 16$ ADCL muß gesetzt sein, wenn $f_{osz} > 16MHz$
MX0 MX1 MX2 MX3	Auswahl des Analogkanals (siehe Bild 10-4)

3.2.3 Datenregister ADDAT

ADDATH (D9H), nicht bitadressierbar
ADDATL (DAH), nicht bitadressierbar

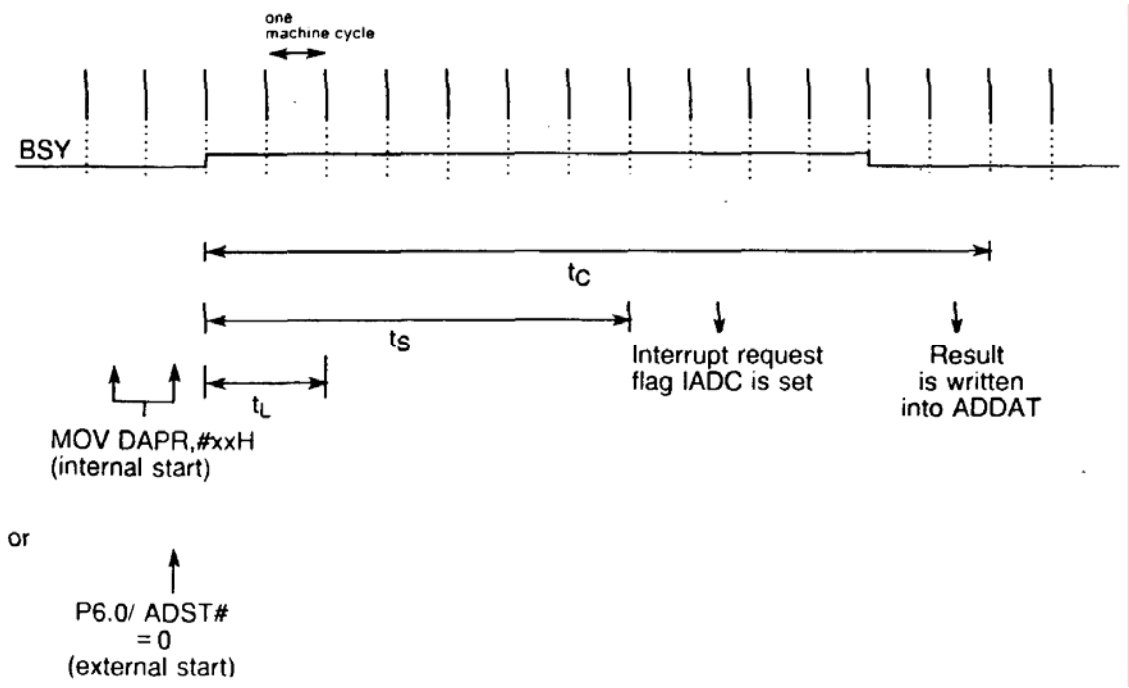
MSB							LSB
ADDATH.7	ADDATH.6	ADDATH.5	ADDATH.4	ADDATH.3	ADDATH.2	ADDATH.1	ADDATH.0
ADDATL.7	ADDATL.6	-	-	-	-	-	-

Ergebnisregister für den A/D-Wandler des 80C517A. Die Register ADDATH und ADDATL enthalten das 10-bit-Ergebnis der A/D-Wandlung, wobei das höherwertigste Bit in ADDATH.7, das niederwertigste Bit in ADDATL.6 enthalten ist. Ein Schreibzugriff auf ADDATL startet die A/D-Wandlung, wenn nicht externer Start mit ADEX = 1 eingestellt ist. Reset-Wert: 00H (ADDATH), 00xx xxxxB (ADDATL).

3.2.4 Wandlungszeiten

f_{osz} [MHz]	ADCL	f_{ADC} [MHz]	T_S [µs]	T_C [µs]
12	0	1,50	2,67	9,33
12	1	0,75	5,33	18,66
16	0	2,00	2,00	7,00
16	1	1,00	4,00	14,00
18	0	-	-	-
18	1	1,13	3,55	12,40

3.2.5 Timingdiagramm



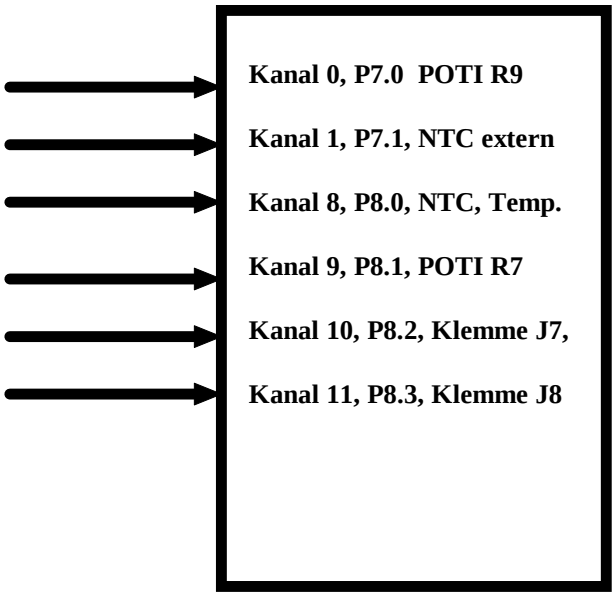
4 ÜBUNG 12 (517):

Schreiben sie eine Funktion mit Namen ad517, der man die gewünschte Kanalnummer übergeben kann, und die als Rückgabewert das 8 Bit Ergebnis des ADU bei 0 - 5V Referenzspannung liefert.

Hinweis:

nötiger zweiter Parameter ist der Referenzspannungsbereich bei 0 - 5V --> 0x00

5 ÜBUNG 13: Spannungsmessung



Schreiben Sie ein Programm, das fortlaufend die Spannung am Potentiometer R7 Port 8.1 = Kanal 9 aber auch am Kanal 0 mißt und an der LCD entsprechend visualisiert.

6 ÜBUNG 14 (517):

Schreiben sie eine Funktion mit Namen ad10Bit, die durch Verschieben der Referenzspannungsbereiche eine erhöhte Auflösung erzielt.

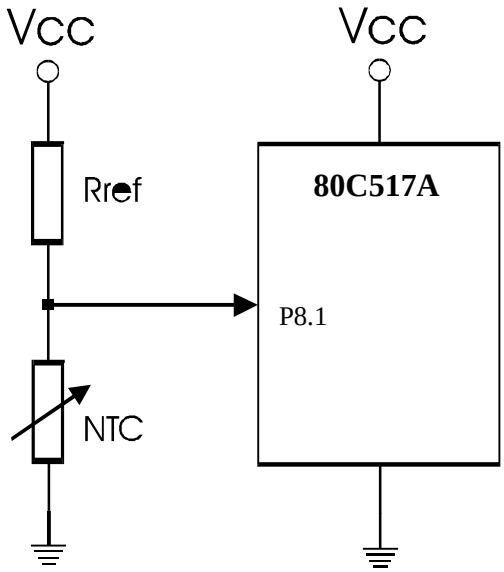
Hinweis: *Beziehen sie die Funktion ad517(Parameter 1,Parameter 2) mit ein.*

7 Temperaturmessung mittels NTC:

Daten des Heißleiters :

Hersteller : Siemens

Typenbezeichnung : S863 5,0 KΩ (Nennwiderstand bei 25 ℃)
-40°C - 70°C , ±1% Toleranz
Anwendung : Temperaturkompensation, -messung, -regelung



Die temperaturabhängige Spannung des NTC soll mittels internem ADU des Mikrokontrollers 80C517(A) ausgewertet , und die Kennlinie durch die Software mit der sogenannten **Steinhart-Hart-Gleichung** linearisiert werden.
Eine Linearisierung ist deshalb nötig weil die Widerstandsänderung nicht linear mit der Temperatur verläuft. Bei guten Heißleitern werden allerdings Wertetabellen angegeben, aus denen sich die Koeffizienten für die Steinhart Hart Gleichung errechnen lassen.

Die Temperatur läßt sich dann mittels dieser Gleichung folgendermaßen errechnen :

Steinhart Hart Gleichung :

- ***u*** = *Spannung am NTC* = $U * \left(\frac{k}{1024}\right) = U * \frac{r\_temp}{r\_temp + r\_ref}$

- ***r_temp*** = $k * \frac{r\_ref}{1024 - k}$

- ***Temperatur*** = $\frac{1}{a + b * \ln(r\_temp) + c * (\ln r\_temp)^3}$

Da es sich hier um fixe Fließkomma (float) Werte handelt, lautet die Konstantenvereinbarung im Programm folgendermaßen :

```
code float  a = 0.0012814814,
            b = 0.000236678791,
            c = 0.0000000908736406,
            r_ref = 4990;
```

a,*b*,c Koeffizienten des NTC

☞ Diese Formeln finden sie bereits in ihrem Beispielprogramm.

Da mathematische Funktionen wie der Logarithmus verwendet werden, muß das Headerfile ***Math.h*** mittels ***# include*** Anweisung eingebunden werden.

Die Anweisung **CODE** veranlaßt den Linker die Konstanten in den Adressraum des EPROMS zu schreiben dadurch werden sie als Programmkode behandelt, und sind folglich nicht veränderbar.

8 ÜBUNG 15 :

Schreiben Sie unter Verwendung der zuvor programmierten Funktionen ein Programm, welches die Temperatur berechnet und am LC - Display anzeigt.

9 ÜBUNG 16 :

Erweitern Sie das Programm derart, daß sowohl Temperatur, als auch die veränderliche Potentiometerspannung ständig am LC Display folgendermaßen anzeigt werden :

1. Zeile : **Port 8.1 : 4.56 Volt**
2. Zeile : **Port 8.0 : 20.34 °c**

Vorschlag:

Verwenden Sie den definierten °c *Charakter (Nr.7) aus der LCD Funktion "spezial_ graf()" zur Temperaturanzeige.*

10 Musterlösung Übung 12 (517):

```
/*****************************************************************
*                    AD517.C
* liest Spannung über internen ADC ein  ONLY 517
* Variable kanal  :  Kanalnummer des einzulesenden Analogeingangs
* Variable v_intref :Einstellung der internen Referenzspgn. im
*                    Register DAPR
* Rueckgabewert:     8bit Wort aus Ergebnisregister ADDAT
*****************************************************************/



unsigned char ad517(unsigned char kanal, unsigned char v_intref)
{

while(BSY) /* warte auf Ende einer möglichen Konvertierung  */
  {    ;   }
  ADCON1=kanal;    /* Kanalauswahl                */
  DAPR = v_intref; /* Einstellung der Aufloesung    */
                   /* sowie Start der Konvertierung */

  while(BSY)       /* Wartet auf das Ende der Konvertierung*/
  {    ;   }

  return(ADDAT);   /* Gibt Ergebnis zurueck         */
```

11 Musterlösung Übung 13

```
/*********************************************************/
/*                      UEB13.C                          */
/*              Ausgeben an die LCD Anzeige              */
/*********************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma mod517 code debug pl(61)

/****   Angabe der Include Dateien              ****/
#include <reg517a.h>
#include <stdio.h>
#include "lcd.h"

/* globale Variable */
unsigned char buf[21];


int aduwert(char kanal)
{
  ADCON1=kanal;
  ADDATL = 0;      /* ADU starten */
  while(BSY);
  return((ADDATL>>6)+ADDATH*4);
}
main()
{
init_lcd();
blank_lcd();
while(1)
{
  while(BSY);
  sprintf(buf,"Kanal 9 Wert= %4d",(int)aduwert(9));
  print_lcd(2,1,buf);
  sprintf(buf,"Kanal 0 Wert= %4d",(int)aduwert(0));
  print_lcd(1,1,buf);


}
}
```

12 Musterlösung Übung 14 (517):

```
/*****************************************************************/
/*   AD10BIT.C
/*   Einlesen des AD Wandlers und Erhöhung der Genauigkeit durch
/*   feinere Wahl der Aufloesung
/*****************************************************************/

float ad10bit(unsigned char kanal)
{
     code float              aufloes1 = 0.01953125;
     xdata float             u, res_float;
     xdata unsigned char     result;

     result = ad517(kanal, 0x00);
     res_float = (float) result;

     if (res_float <= 63)
     {  result = ad517(kanal,0x40);
        u = ((float) result * aufloes1 / 4);

     } else if (res_float > 63 && res_float <= 127)
     {   result = ad517(kanal, 0x84);
        u = ((float) result * aufloes1 / 4) + 1.25;

     } else if (res_float > 127 && res_float <= 191)
     {   result = ad517 (kanal, 0xB8);
        u = ((float) result * aufloes1 / 4) + 2.5;

     } else {
        result = ad517 (kanal, 0x0B);
        u = ((float) result * aufloes1 / 4) + 3.75;
     }
return u;
}
```

13 Musterlösung Übung 15 :

```
/*********************************************************/
/*                      ueb15.C                          */
/*              Ausgeben an die LCD Anzeige              */
/*********************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma mod517 code debug pl(61)

/****   Angabe der Include Dateien              ****/
#include <reg517a.h>
#include <stdio.h>
#include <math.h>
#include "lcd.h"

/* globale Variable */
unsigned char buf[21];
int adu;
float r_temp,lnr,temp;
code float a=0.0012814814,
      b=0.000236678791,
      c=0.0000000908736406,
      r_ref = 4990;

int aduwert(char kanal)
{
  ADCON1=kanal;
  ADDATL = 0;
  while(BSY);
  return((ADDATL>>6)+ADDATH*4);
}

main()
{
init_lcd();
blank_lcd();

while(1)
{
  adu = aduwert(8);
  r_temp = r_ref*adu/(1023-adu);
  lnr = log(r_temp);
  temp = (1.0/(a+b*lnr+c*lnr*lnr*lnr));
  temp-=273.2;
  sprintf(buf,"R=%7.1f T=%6.2f",r_temp,temp);
  print_lcd(1,1,buf);
}
}
```

14 Musterlösung Übung 16:

```
/*******************************************************/
/*                      UEB16.C                        */
/*              Ausgeben an die LCD Anzeige            */
/*******************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma mod517 code debug pl(61)

/****   Angabe der Include Dateien              ****/
#include <reg517a.h>
#include <stdio.h>
#include <math.h>
#include "lcd.h"

/* globale Variable */
unsigned char buf[21];
int adu;
float r_temp,lnr,temp;
code float  a=0.0012814814,
            b=0.000236678791,
            c=0.0000000908736406,
            r_ref = 4990;

int aduwert(char kanal)
{
  ADCON1=kanal;
  ADDATL = 0;
  while(BSY);
  return((ADDATL>>6)+ADDATH*4);
}

main()
{
init_lcd();
blank_lcd();
spezial_graf();

while(1)
{
  adu = aduwert(8);
  r_temp = r_ref*adu/(1023-adu);
  lnr = log(r_temp);
  temp = (1.0/(a+b*lnr+c*lnr*lnr*lnr));
  temp-=273.2;
  sprintf(buf,"Port 8.0: %5.2f",temp);
  print_lcd(2,1,buf);
  sprintf(buf,"Port 8.1: %6.3f V",5.0*aduwert(9)/1024.0);
  print_lcd(1,1,buf);
  char_lcd(2,17,7);
}
}
```