16C62 A-D Wandler und Interruptfunktionen

Ziel

Benutzung des A/D-Wandlers zur Messung eines analoges Eingangssignales.

Umfeld

Der M16C62 verfügt über 8 gemultiplexte A/D Kanäle, welche mit einem SAR-Wandler digitalisiert werden. Der Wert steht nach dem Wandlungszyklus im dem Eingang zugehörigen Register AD0..AD7 bereit.
Normalerweise wird die Wandlung einzelner Werte durch Auslösen über eines AD-Interrupts angezeigt. In der Interruptroutine wird der Wert gelesen und für das Programm in einer globalen Variablen bereit gestellt und signalisiert.

Rahmenbedingungen

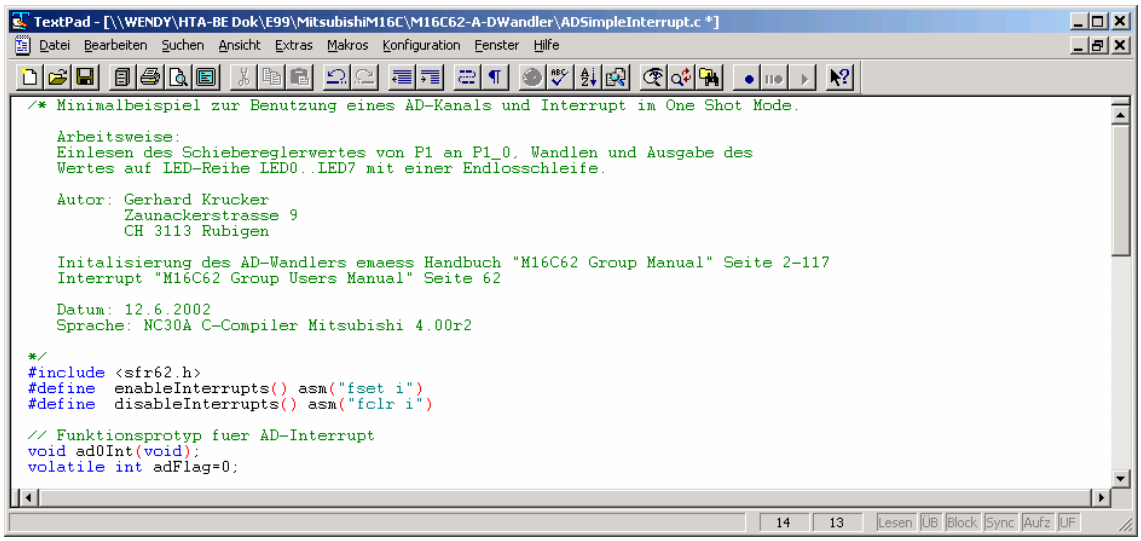
Die Adresse der Interruptfunktion muss in SECT30.INC eingetragen werden und die Interruptquellen sind gemäss Vorschrift zu initialisieren. Vor einer Initialisierung immer Interrupts sperren (mit Makro disableInterrupts). Nach der Initialisierung Interrupts freigeben.

Beispiel: Analogwert von Potentiometer einlesen und Ausgabe auf LED-Reihe

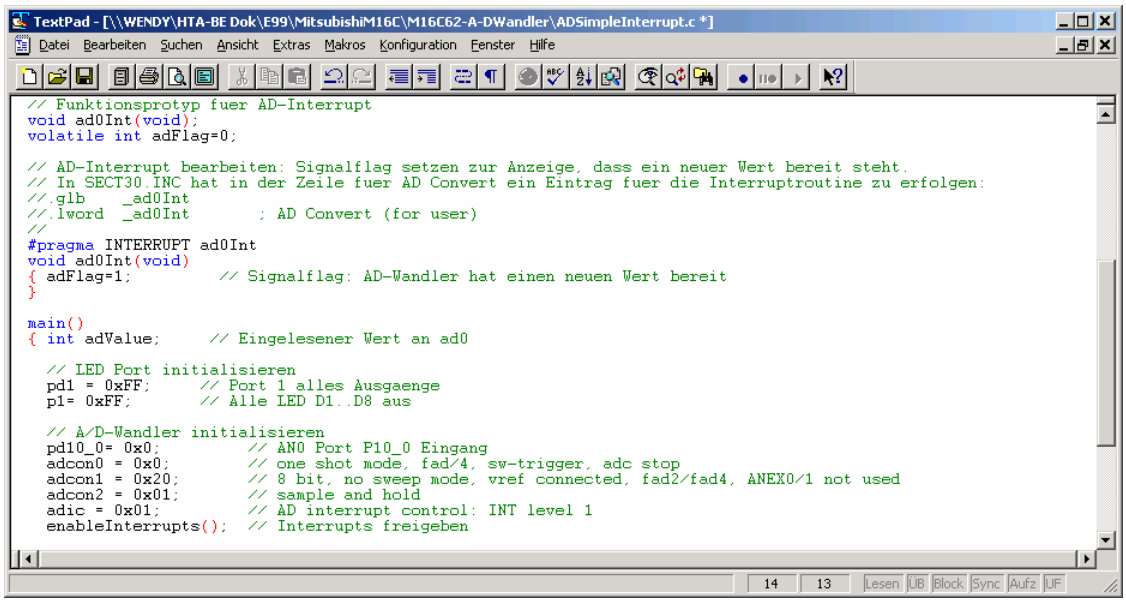
1. Neues Projekt analog Übung 2 erzeugen. (Verwenden Sie wegen des angepassten SECT30.INC kein altes Projekt.)

2. Neues C-Sourcefile mit dem Editor erzeugen, benennen und dem Projekt zufügen, z.B. ADSimple.C.

3. Im Sourcefile zuerst die Makros und Funktionsprototypen für die AD-Wandlerfunktionen definieren:

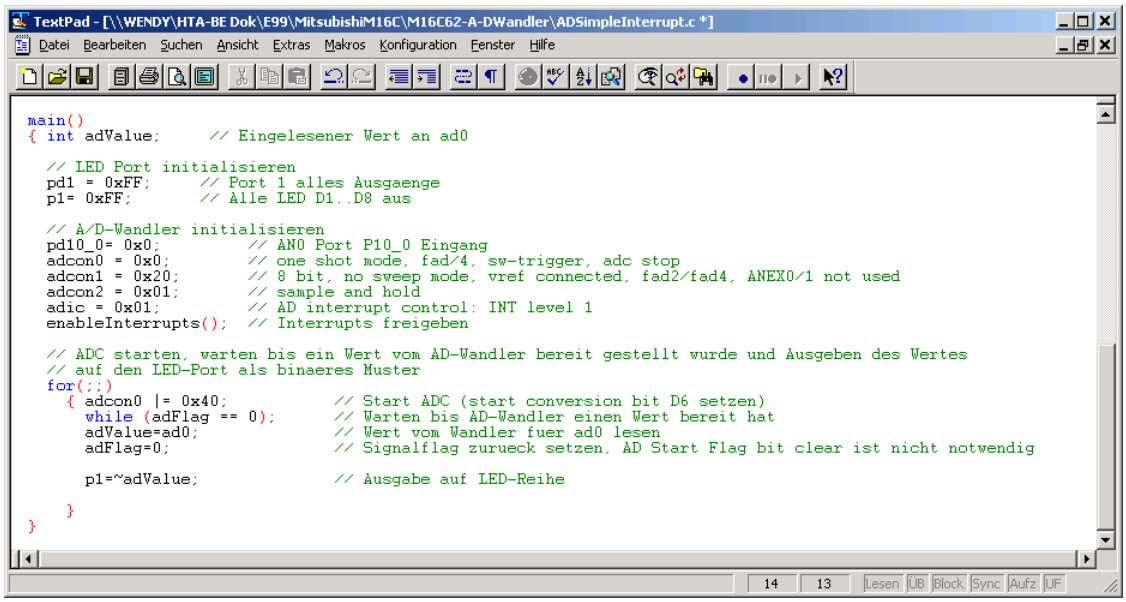


Dann die AD-Wandler Initialisierung, Interruptfunktion zur Signalisierung und Portinitialisierung codieren:



Die globale Variable `adFlag` ist mit dem Prefix `volatile` zu definieren. Damit wird eine Optimierung für diese Variable unterdrückt und der Wert wird bei jedem Zugriff neu aus dem Speicher geladen. Dies ist notwendig, weil ein Schreibzugriff aus dem separaten Interruptprozess erfolgt und die Wertänderung in der Warteschleife bei einer Registervariablen nicht erkannt würde.

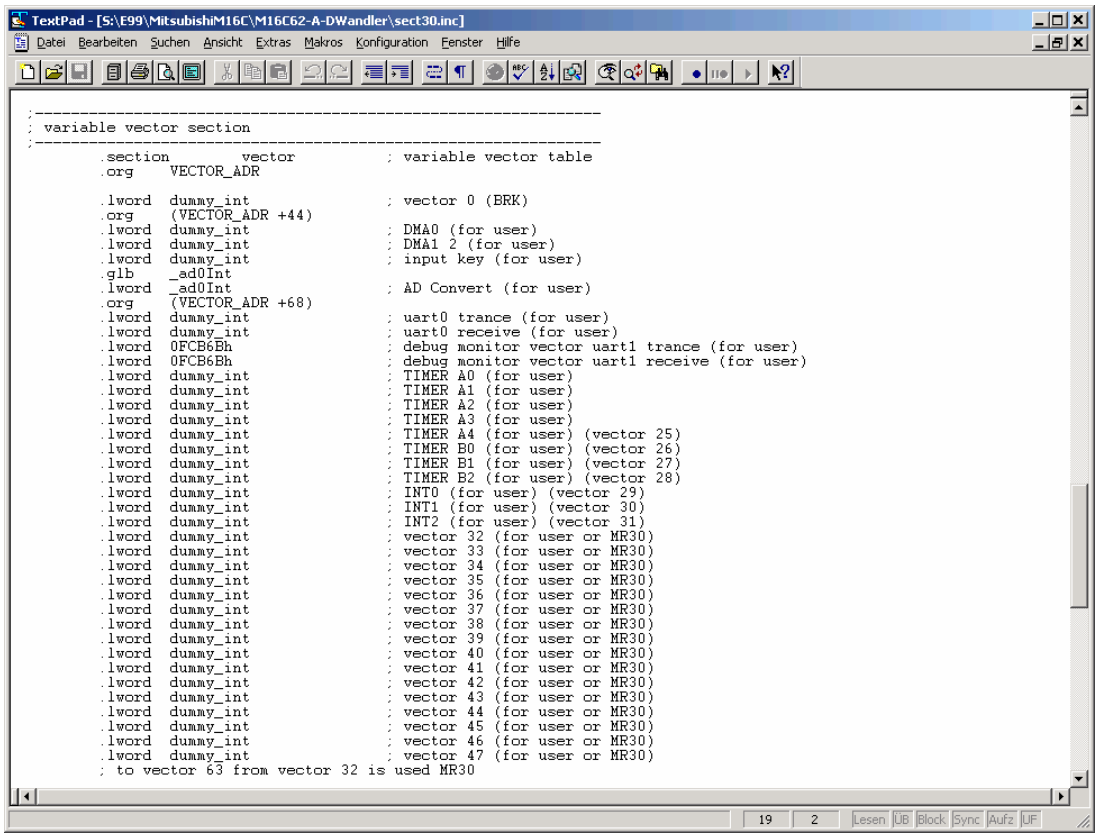
Anschliessend den Rest des Hauptprogrammes auscodieren:



4. Nun ist der Eintrag der Adresse für die Interruptfunktion für den AD-Interruptvektor in `SECT30.INC` vorzunehmen.
   Dazu `SECT30.INC` im Projektmanager öffnen. In der Vektortabelle an der Stelle für AD Convert den Eintrag `.lword dummy_int` löschen und gegen

```
.glb _ad0Int
.lword  _ad0Int          ; AD Convert (for user)
```

   ersetzen.



Bemerkungen:
`.glb _ad0Int` definiert das C-Symbol ad0Int für den Linker als extern.
`.lword _ad0Int` setzt den Interruptvektor für den AD-Wandler auf die Adresse Funktion `ad0Int`.

Durch den C-Compiler wird allen exportierten Symbolen ein _-Zeichen vorgesetzt.

Werden Änderungen in SECT30.INC vorgenommen ist das gesamte Projekt neu zu builden (Rebuild-Knopf).

5. Programm compilieren, download und testen.

Aufgaben

1. Auscodieren des Beispiels und Test.

2. Änderung des Beispiels so, dass die Potentiometerstellung mit genau der darunter liegenden liegenden LED angezeigt wird, d.h. Poti P1 im Linksanschlag: LED J8 leuchtet, Poti P1 im Rechtsanschlag: LED J1 leuchtet, Port P1 in der Mitte: LED J4 oder J5 leuchtet. (Das Poti is ev. nicht linear.)

3. Änderung des Beispiels so, dass die Potentiometerstellung mit einem Leuchtband angezeigt wird, d.h. Poti P1 im Linksanschlag: Alle LED J1..J8 leuchteten, Poti P1 im Rechtsanschlag: Keine LED leuchtet, Pot P1 in der Mitte: LED J1..J4 leuchten.

4. Kombination mit Übung 2: Einlesen des Potentiometerwertes. Dieser wird zur Steuerung der Blinkfrequenz benutzt. Poti im linkem Anschlag: 10ms Blinkdauer an LED J1..J8. Poti P1 im rechten Anschlag: Blinkintervall 10s für LED J1..J8.

5. Untersuchen Sie den Wertebereich, den der Wandler in beiden Potentiometerstellungen digitalisiert. Entspricht er Ihren Erwartungen? Wenn nein finden Sie eine Erklärung!

Gesamtes Beispiel:

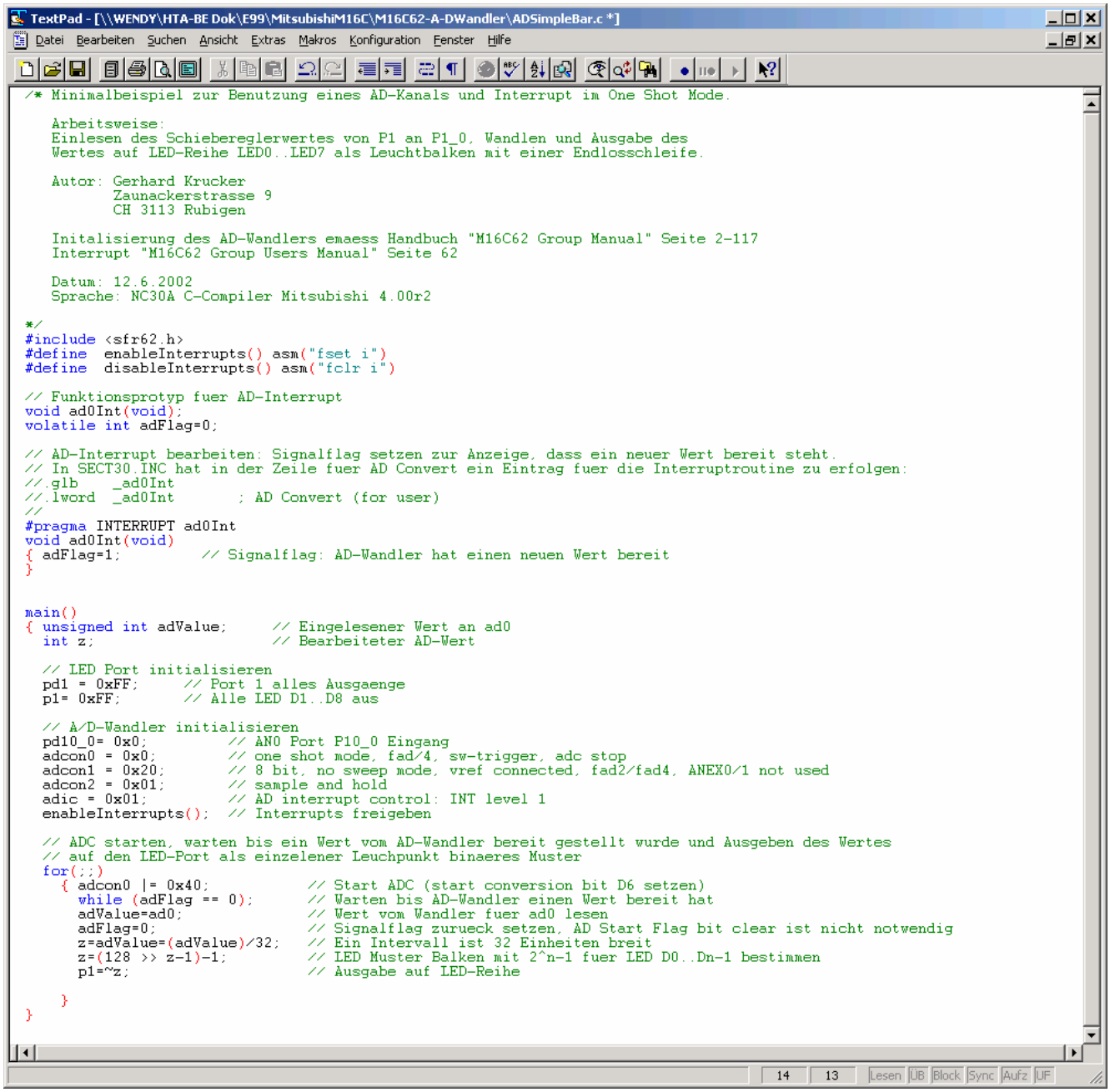
TextPad - [\\WENDY\HTA-BE Dok\E99\MitsubishiM16C\M16C62-A-DWandler\ADSimpleInterrupt.c *]

Datei Bearbeiten Suchen Ansicht Extras Makros Konfiguration Fenster Hilfe

```
/* Minimalbeispiel zur Benutzung eines AD-Kanals und Interrupt im One Shot Mode.

   Arbeitsweise:
   Einlesen des Schiebereglerwertes von P1 an P1_0, Wandlen und Ausgabe des
   Wertes auf LED-Reihe LED0..LED7 mit einer Endlosschleife.

   Autor: Gerhard Krucker
          Zaunackerstrasse 9
          CH 3113 Rubigen

   Initalisierung des AD-Wandlers emaess Handbuch "M16C62 Group Manual" Seite 2-117
   Interrupt "M16C62 Group Users Manual" Seite 62

   Datum: 12.6.2002
   Sprache: NC30A C-Compiler Mitsubishi 4.00r2

*/
#include <sfr62.h>
#define  enableInterrupts() asm("fset i")
#define  disableInterrupts() asm("fclr i")

// Funktionsprotyp fuer AD-Interrupt
void ad0Int(void);
volatile int adFlag=0;

// AD-Interrupt bearbeiten: Signalflag setzen zur Anzeige, dass ein neuer Wert bereit steht.
// In SECT30.INC hat in der Zeile fuer AD Convert ein Eintrag fuer die Interruptroutine zu erfolgen:
//.glb    _ad0Int
//.lword  _ad0Int       ; AD Convert (for user)
//
#pragma INTERRUPT ad0Int
void ad0Int(void)
{ adFlag=1;          // Signalflag: AD-Wandler hat einen neuen Wert bereit
}

main()
{ int adValue;     // Eingelesener Wert an ad0

  // LED Port initialisieren
  pd1 = 0xFF;      // Port 1 alles Ausgaenge
  p1= 0xFF;        // Alle LED D1..D8 aus

  // A/D-Wandler initialisieren
  pd10_0= 0x0;          // AN0 Port P10_0 Eingang
  adcon0 = 0x0;         // one shot mode, fad/4, sw-trigger, adc stop
  adcon1 = 0x20;        // 8 bit, no sweep mode, vref connected, fad2/fad4, ANEX0/1 not used
  adcon2 = 0x01;        // sample and hold
  adic = 0x01;          // AD interrupt control: INT level 1
  enableInterrupts();  // Interrupts freigeben

  // ADC starten, warten bis ein Wert vom AD-Wandler bereit gestellt wurde und Ausgeben des Wertes
  // auf den LED-Port als binaeres Muster
  for(;;)
    { adcon0 |= 0x40;           // Start ADC (start conversion bit D6 setzen)
      while (adFlag == 0);      // Warten bis AD-Wandler einen Wert bereit hat
      adValue=ad0;              // Wert vom Wandler fuer ad0 lesen
      adFlag=0;                 // Signalflag zurueck setzen, AD Start Flag bit clear ist nicht notwendig

      p1=~adValue;              // Ausgabe auf LED-Reihe

    }
}
```

14 13 Lesen ÜB Block Sync Aufz UF

Lösung LED Balken:



Lösung LED-Punkt:

TextPad - [\\WENDY\HTA-BE Dok\E99\MitsubishiM16C\M16C62-A-DWandler\ADSimplePoint.c *]

Datei Bearbeiten Suchen Ansicht Extras Makros Konfiguration Fenster Hilfe

```
/* Minimalbeispiel zur Benutzung eines AD-Kanals und Interrupt im One Shot Mode.

   Arbeitsweise:
   Einlesen des Schiebereglerwertes von P1 an P1_0, Wandeln und Ausgabe der Potentiometerstellung
   als Punkt auf die LED-Reihe LED0..LED7 mit einer Endlosschleife.

   Autor: Gerhard Krucker
          Zaunackerstrasse 9
          CH 3113 Rubigen

   Initalisierung des AD-Wandlers emaess Handbuch "M16C62 Group Manual" Seite 2-117
   Interrupt "M16C62 Group Users Manual" Seite 62

   Datum: 12.6.2002
   Sprache: NC30A C-Compiler Mitsubishi 4.00r2

*/
#include <sfr62.h>
#define  enableInterrupts() asm("fset i")
#define  disableInterrupts() asm("fclr i")

// Funktionsprotyp fuer AD-Interrupt
void ad0Int(void);
volatile int adFlag=0;

// AD-Interrupt bearbeiten: Signalflag setzen zur Anzeige, dass ein neuer Wert bereit steht.
// In SECT30.INC hat in der Zeile fuer AD Convert ein Eintrag fuer die Interruptroutine zu erfolgen:
//.glb    _ad0Int
//.lword  _ad0Int       ; AD Convert (for user)
//
#pragma INTERRUPT ad0Int
void ad0Int(void)
{ adFlag=1;          // Signalflag: AD-Wandler hat einen neuen Wert bereit
}

main()
{ unsigned int adValue;     // Eingelesener Wert an ad0
  int z;                    // Bearbeiteter AD-Wert

  // LED Port initialisieren
  pd1 = 0xFF;      // Port 1 alles Ausgaenge
  p1= 0xFF;        // Alle LED D1..D8 aus

  // A/D-Wandler initialisieren
  pd10_0= 0x0;          // AN0 Port P10_0 Eingang
  adcon0 = 0x0;         // one shot mode, fad/4, sw-trigger, adc stop
  adcon1 = 0x20;        // 8 bit, no sweep mode, vref connected, fad2/fad4, ANEX0/1 not used
  adcon2 = 0x01;        // sample and hold
  adic = 0x01;          // AD interrupt control: INT level 1
  enableInterrupts();   // Interrupts freigeben

  // ADC starten, warten bis ein Wert vom AD-Wandler bereit gestellt wurde und Ausgeben des Wertes
  // auf den LED-Port als einzelener Leuchpunkt binaeres Muster
  for(;;)
    { adcon0 |= 0x40;           // Start ADC (start conversion bit D6 setzen)
      while (adFlag == 0);      // Warten bis AD-Wandler einen Wert bereit hat
      adValue=ad0;              // Wert vom Wandler fuer ad0 lesen
      adFlag=0;                 // Signalflag zurueck setzen, AD Start Flag bit clear ist nicht notwendig
      z=adValue=(adValue)/32;   // Intervallbreite ist 32 Einheiten
      z=(128 >> z);             // LED Muster bestimmen
      p1=~z;                    // Ausgabe auf LED-Reihe

    }
}
```

14 13 Lesen UB Block Sync Aufz UF