Messdatenverarbeitung

PR 0430 342

Praktikumsaufgaben A1

Prof. Dr.-Ing. Clemens Gühmann
José-Luis Bote-Garcia

22. März 2017

Technische Universität Berlin
Fakultät IV Elektrotechnik und Informatik
Institut für Energie- und Automatisierungstechnik
Fachgebiet Elektronische Mess- und Diagnosetechnik

Messkette I - ADU Auslesen

Qualifikationsziele

Nach der Praktiumsaufgabe "ADU Auslesen"...

- können Sie den Analog-Digital-Wandler eines 8 Bit-Mikrocontrollers auslesen.
- können Sie Routinen zur Interrupt-Programmierung in C schreiben.
- sind Sie mit den Strukturierungsmöglichkeiten für komplexe C-Programme vertraut.

Problemstellung

Die Messdatenerfassung im Rahmen des Praktikums soll mit Hilfe eines Mikrocontrollers aus der AVR-Familie der Firma Atmel erfolgen. Dieser verfügt über einen integrierten Analog-Digital-Wandler. Um mit Hilfe des Mikrocontrollers Messdaten aufnehmen zu können, muss der Analog-Digital-Wandler zunächst initialisiert werden. Anschließend wird nach dem Abschluss jeder einzelnen Analog-Digital-Wandlung ein Interrupt ausgelöst. In der dazugehörigen Interrupt-Routine, wird dann das Ergebnis der Umsetzung in den Sende-Puffer der seriellen Schnittstelle geschrieben.

Benötigte Hardware

Für diese Praktikumsaufgabe werden die folgenden Hardware-Komponenten benötigt:

- Sensorknoten (virtenio, RTnode)
- Anschlussplatine (TU-MDT-Eigenbau)
- Spannungsquelle (Labornetzteil/Funktionsgenerator)

Vorbereitung

Durch die Vorbereitungsaufgaben soll das Auslesen des ADUs an Hand eines einfachen Beispiels geübt werden.

1. Lesen Sie das Kapitel über den Analog-Digital-Wandler im Datenblatt des Prozessors durch und machen Sie sich mit dessen Funktionsweise vertraut.

2. Legen Sie ein neues Projekt an und schreiben Sie ein Programm, das periodisch Umsetzungen startet und das Umsetzungsergebnis ausliest. In Abhängigkeit von dem ausgelesenen Wert sollen die LEDs gesetzt werden (siehe Tabelle 1.1).
   Anforderungen:
   - Es soll die interne 2,56 V Referenz als Spannungsreferenz genutzt werden.
   - Als Eingang soll der Kanal `ADC0` im Single-Ended-Modus genutzt werden.
   - Der ADU-Takt soll ein $\frac{1}{32}$ des CPU-Takts betragen $F_{ADU} = F_{CLK}/32$
   - Der ADU soll im "free-running" Modus betrieben werden.
   - Nach jeder Umsetzung soll ein ADC-Interrupt ausgelöst werden (Vektor: `ADC_vect`). Das Auslesen des Umsetzungsergebnis und das setzen der LEDs soll in der Behandlungs-Routine zu dem Interrupt geschehen.
3. Betrachten Sie Ihr geschriebenes Programm. Mit welcher Abtastrate arbeitet der ADU? Welche Abtastraten ließen sich durch Anpassung des Teilers zur Erzeugung des ADU-Takts einstellen?
4. Laden Sie sich das AVR Studio-Projekt MDV_PR von der Homepage des Fachbereichs herunter und machen Sie sich mit der Struktur des Codes vertraut.

ADU-Wert	Rot	Grün	Orange
0...127	aus	aus	aus
128...511	**an**	aus	aus
512...768	**an**	**an**	aus
769...1023	**an**	**an**	**an**

Tabelle 1.1: LED-Muster für Umsetzungsergebnisse

Praktikum

Im Labor soll ein praxistaugliches Messdatenerfassungssystem aufgebaut werden.

1. Testen Sie ihr in den Vorbereitungsaufgaben erstelltes Programm zum Auslesen des ADUs.
2. Implementieren Sie in der Datei `adc.c` die Funktion `adcInit`. Sie soll die nötige Initialisierung des ADUs durchführen damit anschließend Messungen gestartet werden können. Verwenden Sie hierfür den folgenden Funktionskopf:

```
/**
 * \brief initializes the ADC
 * \author your_name
 * \date day_of_implementation
 */
void adcInit();
```

   Anforderungen:
   - Es soll die interne 2,56 V Referenz als Spannungsreferenz genutzt werden.
   - Als Eingang soll der Kanal `ADC0` im Single-Ended-Modus genutzt werden.
   - Der ADU-Takt soll ein $\frac{1}{32}$ des CPU-Takts betragen $F_{ADU} = F_{CLK}/32$
   - Es soll Auto-Triggering als Methode zum Starten einer Umsetzung verwendet werden. Eine Umsetzung soll durch einen Compare-Match von Timer1 gestartet werden

3. Implementieren Sie in der Datei `adc.c` die Funktion `adcStart`. Sie soll die Aufnahme einer einstellbaren Anzahl von Samples bei einer wählbaren Sampling-Rate starten, indem Sie Timer1 geeignet initialisiert und startet. Verwenden Sie hierfür den folgenden Funktionskopf.

```
/**
 * \brief initializes the adc
 * \author your_name
 * \date day_of_implementation
 * \param sampleRateCode compare match value
 *            that determines the sample rate
 *            SAMPLE_RATE = F_CPU / (RATE_CODE + 1)
 * \param sampleCount number of sample to be acquired
 * \param triggerMode trigger mode:
 *            NONE - trigger is disabled
 *            RISING - trigger at rising edge
 *            FALLING - trigger at falling edge
 * \param triggerLevel trigger level in quantization steps
 *            (-512...511)
 */
void adcStart(uint16_t sampleRateCode, uint32_t sampleCount,
              trigger_t triggerMode, int16_t triggerLevel);
```

Hinweise:

- Timer1 soll im CTC-Modus betrieben werden.
- Es hat sich als günstig erwiesen Timer1 hierbei direkt mit dem CPU-Takt zu takten.
- Eine Trigger-Funktion **muss nicht** realisiert werden. Die Parameter `triggerMode` und `triggerLevel` können ignoriert werden.

4. Implementieren Sie in der Datei `adc.c` eine Interrupt-Routine, die nach dem Abschluss einer Analog-Digital-Wandlung aufgerufen wird. Die Interrupt-Routine soll das Ergebnis der Umsetzung auslesen und eine erste Offset-Korrektur durchführen. Hierzu soll von dem Ergebnis der Umsetzung, das Werte im Intervall $0 \ldots 1023$ annehmen kann der Wert 512 abgezogen werden, sodass sich Werte im Intervall $-512 \ldots 511$ ergeben. Der so errechnete Wert soll mit der Funktion `filterWrite2Buf` in einen Puffer für die weitere Verarbeitung geschrieben werden. Außerdem soll in der Interrupt-Routine dafür gesorgt werden, dass die Aufnahme von Messwerten nach der Aufnahme aller gewünschten Samples gestoppt wird. Der Kopf der Interrupt-Routine sollte wie folgt aussehen:

```
/**
 * \fn ISR(ADC_vect)
 * \author your_name
 * \date day_of_implementation
 * \brief Interrupt-Routine for the ADC-Interrupt.
 *          Gets called when an analog-to-digital
 *          conversion is complete
 */
ISR(ADC_vect) {
        ...
}
```

Die Funktion `filterWrite2Buf` ist bereits in der Datei `filter.c` implementiert. Sie hat den folgenden Funktionskopf:

```
/**
 * \brief writes a value into the input buffer
 * \author Jürgen Funck
 * \date 2010-03-23
 * \param val value to be written
 */
uint8_t filterWrite2Buf(int16_t val);
```

5. Implementieren Sie in der Datei `adc.c` die Funktion `adcIsRunning`. Sie soll eine 1 zurückgeben, wenn der ADU weitere Samples aufnehmen muss. Sind alle gewünschten Samples aufgenommen soll sie eine 0 zurückgeben. Der Funktionskopf sollte wie folgt aussehen.:

```
/**
 * \brief signifies whether the ADC is currently acquiring data
 * \author your_name
 * \date day_of_implementation
 * \return > 0 if the ADC is acquiring data, 0 otherwise
 */
uint8_t adcIsRunning();
```

6. (Zusatzaufgabe) Realisieren Sie eine Triggerfunktion, die die Messung erst bei einer steigenden bzw. fallenden Flanke in Abhängigkeit der eingestellten Trigger-Schwelle startet. Testen Sie die Triggerfunktion und beheben Sie ggf. Fehler in Ihrem Programm.

Abgabe

1. Welches Messverfahren verwendet der AVR-Mikrocontroller? Erläutern Sie dieses in Kürze.
2. Über welche Register lässt sich der ADC des AVR-Mikrocontrollers eintstellen?
3. Welche Abtastraten sind einstellbar? Was ist die maximale Abtastfrequenz des AVR-Mikrocontrollers?
4. Welche Spannungsquellen sind als Refrenzspannung für den ADC einstellbar?
5. Kommentieren Sie ihren Code in Ihrem Projekt.
   a) Geben Sie dabei nicht wieder was der Befehlt macht.

   ```
   DDRC |= (1<<PC1); // Setze Bit 1 in Register DDRC
   ```

   b) Erklären Sie schwer verständliche Zeilen.

   ```
   TCCR1B = (1<<WGM12) | (1<<CS12) | (1<<CS10); // kein ...
   Input/Capture, CTC-Mode, Timer-Clock = F_CPU/1024
   ```

   c) Gliedern Sie ihren Code.

   ```
   // Port initialisieren
   DDRC |= (1<<PC1); PORTC |= (1<<PC1);
   ```

Die Abgabe findet über den enstprechenden ISIS-Kurs statt. Bitte laden Sie ihren Bericht sowie das AVR-Projekt als Archiv fristgerecht hoch.

Literaturverzeichnis

[1] Atmel Corporation: 8-bit Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash: ATmega640/V ATmega1280/V ATmega1281/V ATmega2560/V ATmega2561/V