Embedded Systems (EBS)

– Veranstaltung SS 2012 –

Aufgaben zur praktischen Selbstbetätigung

Stand: 15. 5. 12

Ausrüstung: PC mit Hyperterminal. Starterkits Atmel STK500. Programmierung in C. Es ist genügend Ausrüstung vorhanden, so daß alle Aufgaben selbständig bearbeitet werden können.

1. Kommandosatz für Atmel AVR. Bits ein- und ausschalten. Über Terminal einzugeben (zeichenorientiert). Zunächst ist herauszufinden, wie man mit GNU C Unterbrechungsbehandungsroutinen (ISRs) schreiben kann. Ggf. nachsehen (Internet).

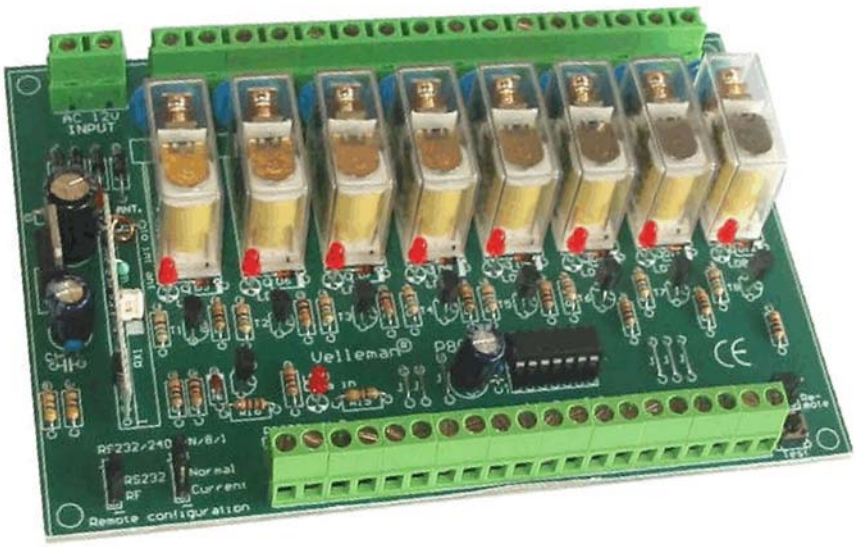
2. Kommandosteuerung der Velleman-Relaisplatine mittels Atmel nachbilden. LEDs des Starterkits anstelle der Relais.

3. Die Velleman-Relaisplatine (oder deren Nachbildung) von einem anderen AVR (Starterkit) aus ansteuern. Zunächst die Tasten des Starterkits zur Auslösung verwenden. Dann wird die Bedientafel 10b eingesetzt. Es können auch SPS-mäßige Funktionen implementiert werden (Sequencer, Zeitrelais, Jalousiensteuerung usw.).

4. XY-Forschung. Mittels XY-Adapter 09a eine Figur (anfänglich Quadrat oder Rhombus, später PacMan o. dergl.) auf dem Oszilloskop darstellen (Starterkit). Vom Hyperterminal aus über den Bildschrim bewegen (Tasten u (up), d (down), l (left), r (right) usw.). Wie könnte eine Zeichendarstellung (Ziffern usw.) ausssehen? (Zunächst an Beispielen ausprobieren, um Aufwand und Laufzeit abschätzen zu können. Die Frage ist, ob in C die Wiederholrate ausreicht. In Assembler geht es jedenfalls.)

Velleman 8056 Relaisplatine mit seriellem Interface

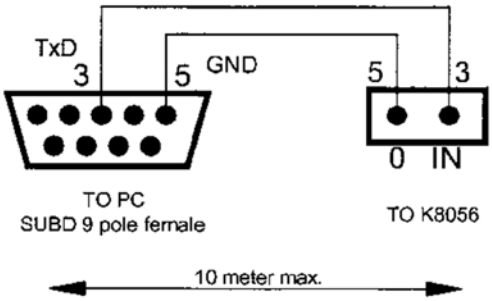
Stand: 15. 5. 2012
Bildquellen: Velleman



19. Control options

1) RS232 cable :

9 pole
3 : TxD
5 : GND
25 pole
2 : TxD
7 : GND

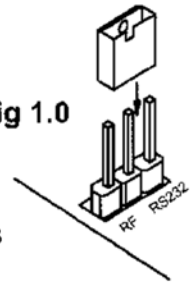


Do not use a null-modem cable !!

Jumper setting :

- Place JP1 in OFF position
- Place JP2 in RS232 mode, see fig 1.0.
- Choose the impedance, high or low with JP3. Define the impedance on experimental use. The impedance depends on the length of RS232 cable you're using.

Fig 1.0



Normally the impedance is 10Kohm => select normal.
If you use current loop (long cable), select current. The impedance will be 1Kohm.

Normal	Current
10K	1K

20. Addressing the K8056 card through RS232 commands.

Note: A Test program can be downloaded from our web site.
The source code (VB) is also available. (English only)

1) Instructions for software design :

As the instructions consist of a string of ASCII characters, it is easy to design software which transmits the instructions via the serial port of the PC.

Port settings are: 2400/8/N/1

- For wired operation, the instruction sequence needs to be sent at least twice.
- For wireless operation, we recommend to send the instruction string at least 5 times in a row to ensure proper reception under all conditions. For improved reliability, add a pause of a least 300ms between two different instructions.

☝ Please note that environmental conditions can interfere with the signal. Therefore, the unit is not suited for use as or as part of any kind of equipment which might cause harm to people or property if a malfunction should occur.

2) Instruction sequence :

To execute a command, the correct sequence needs to be sent to the K8056.
Basically, a command sequence looks like this :

1. CHR$(13)
2. card address (1..255)
3. instruction
4. address (1..255) or relay# ('1'..'9' ASCII)
5. checksum (2-complement of the sum of the 4 previous bytes + 1)

all

3) Instructions :

'E' : Emergency stop all cards, regardless of address. (Carefull, relays turned on by open collector inputs will not be turned off by this command).

'D' : Display address. All cards show their current address in a binary fashion. (LD1 : MSB , LD8 : LSB)

'S' : Set a relay. 'S'-instruction should be followed by relay # '1' to '8'. ('9' sets all relays at once).

'C' : Clear a relay. 'C'-instruction should be followed by a relay # '1' to '8'. ('9' clears all relays at once.)

'T' : Toggle a relay. 'T'-instruction should be followed by a relay # '1' to '8'.

'A' : Change the current address of a card. 'A'-instruction should be followed by the new address (1..255)

'F' : Force all cards to address 1 (default)

'B' : Send a byte. Allows to control the status of all relays in one instruction, by sending a byte containing the relay status for each relay. (MSB : relay1, LSB : relay8)

Die Beschreibung zur Prüfsumme (Checksum) ist inkorrekt.

1. **Alle** vorhergehenden Bytes sind als Binärzahlen aufzuaddieren.

2. Das Prüfbyte ist das Zweierkomplement dieser Summen, NICHT das Zweierkomplement + 1.

3. Wenn es keine Sprachanweisung oder keinen Befehl zur Zweierkomplementbildung gibt, genügt eine bitweise Invertierung (Negation) und das Addieren einer Eins.

Aus einem Beispielprogramm der Fa. Velleman (Visual Basic):

Relais Nr. 7 ist zu setzen.

1. Erzeugen der Prüfsumme:

```
checksum = (255 - ((((13 + address + Asc("S") + Asc("7")) / 256)
- Int((13 + address + Asc("S") + Asc("7")) / 256)) * 256)) + 1
```

2. Senden des Kommandos;

```
messagestring = Chr$(13) & Chr$(address) & "S7" & Chr$(checksum)
```

Alternative mit Power Basic:

1. Erzeugen der Prüfsumme:

```
chk =( ((NOT (13+1+ ASC("S") + ASC("7"))) + 1) AND &hff)
```

2. Senden des Kommandos:

```
COMM PRINT nComm, CHR$(13)+ CHR$(1)+ "S7" +CHR$(chk);
```