April 2009

Henne's Sites

AN012: DMX-Empfang mit AVRs

Inhalt

Einleitung

In dieser Application Note wird der Empfang eines DMX-Signals mit Hilfe von AVR-Mikrocontrollern beschrieben. Die state machine steht sowohl in Assembler als auch in einer C-Version zum Download bereit. Der Code wurde für den DMX-Transceiver von Henne's Sites geschrieben, sollte sich aber auf die meisten AVRs portieren lassen.

Danken möchte ich Martin Schneebacher und Jörgen Dierking für Ihre Unterstützung.

Nutzungsbedingungen

DMX 512
DMX 512 ist ein unidirektionales differentielles serielles Protokoll mit einer Übertragungsrate von 250kBaud. Es sind ein Master und bis zu 32 Slaves an einem Bus vorgesehen. Die Zahl der Slaves kann durch Einsatz von Splittern oder Boostern erhöht werden. Ein Universe kann bis zu 512 Kanäle haben. Die Übertragung wird durch einen Break (LO-Pegel) von mind. 88µs Länge, einem Mark after Break (MAB, HI-Pegel) von mind. 8µs Länge und ein Startbyte (=0) eingeleitet. Ein Byte besteht aus 1Startbit, 8Datenbits und 2Stoppbits.

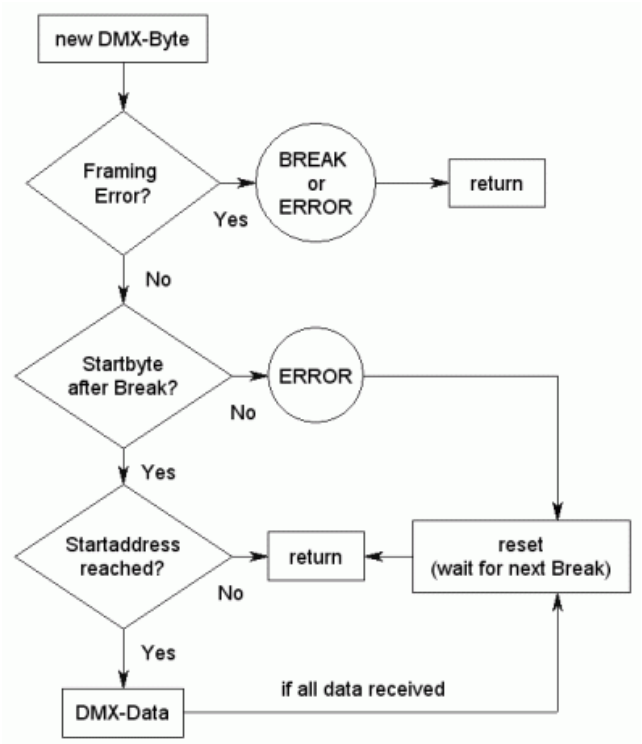
Beschreibung des Codes
Wenn kein Fehler auftritt, läuft der Empfang folgendermaßen ab:

Nach dem Einlesen des Datenbytes und des Statusregisters wird ein "Framing-Error" festgestellt, der ein Break sein könnte. Es wird zunächst davon ausgegangen, dass es sich wirklich um eine BREAK-Condition handelt. Dies wird im DMX Status vermerkt und zum Hauptprogramm zurückgekehrt.

Falls der vorige FrameError wirklich ein Break war, dürfte das nächste Byte ein Startbyte (also 0) sein. Ist dies so, wird der Status inkrementiert und das X-Register als Byte-Counter mit der Startadresse geladen. Falls das Startbyte ungleich 0 ist, gab es einen echten FramingError, der Empfang wird abgebrochen und bis zum nächsten BREAK gewartet.

Im Folgenden wird so lange mit jedem eintreffenden Byte der Byte-Counter dekrementiert, bis die Startadresse erreicht ist. In diesem Fall wird wieder der Status inkrementiert.

Bei den folgenden Bytes handelt es sich um die gewünschten Daten. Sind alle Daten empfangen, wird der DMX Status auf 0 gesetzt und auf den nächsten BREAK gewartet.



Assembler-Version
Assembler ist zwar etwas schwerer zu verstehen und aufwändiger zu schreiben als C, dennoch möchte hier aus Gründen der Performance dazu raten.

Der Code wurde geschrieben mit AVR Studio 4.13.

```
#define DMX_FIELD       0x60    //base address of DMX array
#define DMX_CHANNELS    2       //no. of channels
#define DMXstate        R19     //status reg for DMX-ISR (r16-r31)

#define F_OSC           8000    //oscillator freq. in kHz (typical 8MHz or 16MHz)
#define USE_DIP                 //should get start address from DIPs?
```

DMX_FIELD gibt den Beginn des DMX-Empfangspuffers im SRAM an. Das SRAM beginnt mit 0x60.
DMX_CHANNELS gibt die Anzahl der zu empfangenen DMX-Kanäle an.
DMXstate wird als Statusregister in der state machine benötigt.
F_OSC ist die Quarzfrequenz in kHz.
Wenn USE_DIP definiert ist, wird die Startadresse von den DIP-Schaltern des Transceivers bezogen. Andernfalls wird „get_address“ zur Ermittlung der Startadresse gecalled.

Mit „init_dmx“ wird der USART für den DMX-Empfang initialisiert und der Empfangspuffer auf 0 gesetzt.

Mit „get_byte“ als USART receive complete ISR werden die DMX-Daten empfangen.

Auf die Daten im DMX_Field kann jederzeit zugegriffen werden.

Am Ende des Entry-Files (aber noch vor evtl. Tabellen) sollte die Library eingebunden werden.

C-Version

Der Code wurde geschrieben mit AVR Studio 4.13 und WinAVR-20060125.

In der Headerdatei müssen folgende Variablen und Konstanten definiert sein:

```
#define  F_OSC   8000                    //oscillator freq. in kHz (typical 8MHz or 16MHz)
#define  USE_DIP                         //get start address from DIPs

volatile  uint8_t    DmxRxField[8];      //array of DMX vals (raw)
volatile  uint16_t   DmxAddress;         //start address

extern void          init_DMX_RX(void);
extern void          get_dips(void);
```

F_OSC ist die Quarzfrequenz in kHz.

Wenn USE_DIP definiert ist, wird die Startadresse von den DIP-Schaltern des Transceivers bezogen. Andernfalls kann die Startadresse auch direkt in DmxAddress abgelegt werden.

Die DMX-Daten werden in DmxRxField[] gespeichert.

Mit „init_DMX_RX()“ wird der USART für den DMX-Empfang initialisiert und der Empfangspuffer auf 0 gesetzt.

Mit „get_dips()“ wird aus dem Status der DIP-Schalter des Transceivers die Startadresse bestimmt.