

AN013: DMX-Transmission mit AVR

Inhalt

- Einleitung
- Nutzungsbedingungen
- DMX 512
- Beschreibung des Codes
- Assembler-Version
- C-Version

Einleitung

In dieser Application Note wird die Erzeugung eines DMX-Signals mit Hilfe von AVR-Mikrocontrollern beschrieben. Die state machines stehen 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 AVR portieren lassen.

Nutzungsbedingungen

Die state machines können gemäß der gnu general public license (GPL) genutzt werden.

Falls eine GPL-konforme Nutzung nicht möglich ist, wenden Sie sich bitte an den Urheber.

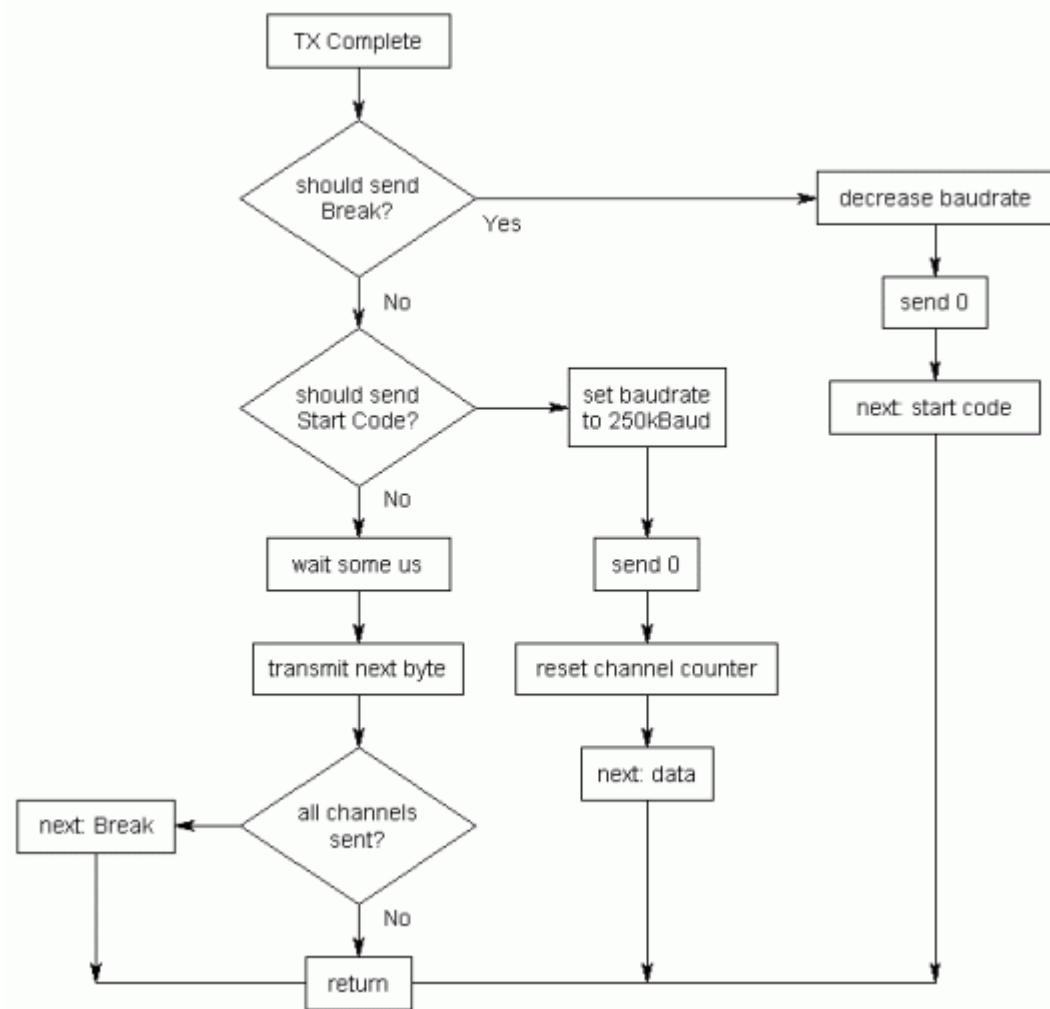
DMX 512

DMX 512 ist ein unidirektionales differentielles serielles Protokoll mit 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 Frames (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 Frame besteht aus 1Startbit, 8Datenbits und 2Stopbits.

Beschreibung des Codes

Der Code besteht neben der Initialisierung aus einer ISR:

In der „USART Transmission Complete“-Routine wird zunächst geprüft, ob ein Break gesendet werden soll. Falls ja, wird mit einer sehr viel kleineren Baudrate ein Nullbyte gesendet, das die Empfänger als einen Break mit über 100µs und einen Mark After Break von über 8µs interpretieren.



Um nach dem Break das Startbyte senden zu können, wird die Baudrate wieder auf 250kBit/s gesetzt und erneut ein Nullbyte gesendet. Abschließend wird der Kanalzähler zurückgesetzt.

Nach Senden des Startbytes können die einzelnen Kanäle übertragen werden. Ist das Universe komplett gesendet, wird wieder mit einem Break begonnen.

Eine Besonderheit dieses Codes ist die variable Interbyte-Gap: Je größer sie ist, desto geringer ist die DMX-Refreshrate und desto länger blockiert die ISR den Controller. Allerdings benötigen zahlreiche Empfänger diese Zeit (der Transceiver nicht!), um das gerade empfangende Byte zu puffern. Die Gap sollte deshalb so klein wie nötig gewählt werden.

Zudem sollten mindestens 50 Kanäle gesendet werden, um die Empfänger nicht mit einem zu hohen Refresh zu überfordern. Sollten Sie weniger Kanäle senden wollen, füllen Sie die restlichen einfach mit Nullen.

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  50            //no. of channels
#define F_OSC          8000         //oscillator freq. in kHz (typical 8MHz or 16MHz)
```

DMX_FIELD gibt den Beginn des DMX-Sendearrays im SRAM an. Das SRAM beginnt mit 0x60.

DMX_CHANNELS gibt die Anzahl der zu sendenden DMX-Kanäle an.

F_OSC ist die Quarzfrequenz in kHz.

Mit „init_dmxx“ wird der USART für das Senden von DMX-Daten initialisiert und das Sendearray auf 0 gesetzt.

Mit „send_byte“ als USART transmission complete ISR werden die DMX-Kanäle gesendet.

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.

```
#define F_OSC          8000          //oscillator freq. in kHz (typical 8MHz or 16MHz)
volatile uint8_t DmxField[50];     //array of DMX vals
```

F_OSC ist die Quarzfrequenz in kHz.

DmxField ist das Sendearray mit den zu sendenden Kanälen.

Mit „init_dmxx“ wird der USART für das Senden von DMX-Daten initialisiert und das Sendearray auf 0 gesetzt.

© Hendrik Hölscher
all rights reserved

Das ungenehmigte Kopieren von Inhalten sowie Mirroring dieser AN ist untersagt.
Die Autoren übernehmen keine Haftung oder Gewährleistung.