April 2021

Henne's Sites

AN112: DMX reception with AVR

Content

- Introduction
- Terms of Use
- DMX 512
- Code description

Introduction

This application note describes the reception of a DMX signal using AVR microcontrollers. The state machine was written in C for the DMX Transceiver from Henne's Sites, but should also be portable to other circuits.

Terms of Use

The state machines can be used in accordance with the gnu general public licence (GPL).
If a GPL-compliant use is not possible, please contact the author.

DMX 512

DMX 512 is a unidirectional differential serial protocol according to the RS485 standard with a transmission rate of 250kBaud. One byte consists of one start bit [4], eight data bits [14, 15] and two stop bits [7, 8]. There is no parity bit (8n2). Inter byte gaps [9] can be inserted between the bytes to allow slower receivers a reliable reception.

A bus (or universe) consists of one master and up to 32 slaves. The number of slaves can be increased by using splitters or boosters. A universe can have up to 512 channels. The transmission is initiated by a break [1] of at least 88µs followed by a mark-after-break [2] of at least 8µs. Afterwards, the start byte [14] is transmitted, which always has a zero value for DMX.

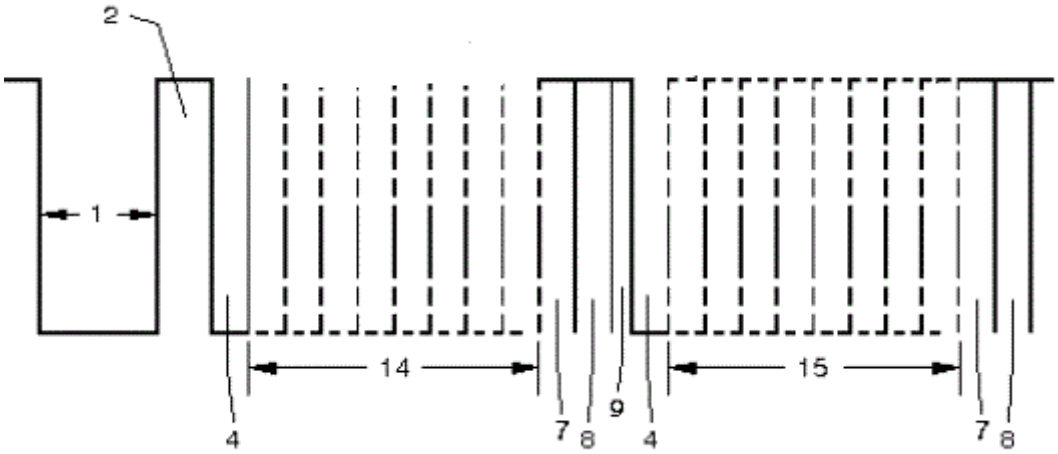

Figure 1: Sketch of the DMX protocol acc. to ANSI E1.11

Code description

After reading the data byte and the status register, a "framing error" is detected, which could also be a break. It is initially assumed to be a BREAK. This is noted in the DMX status and then returned.

If the previous framing error was really a break, the next byte should be a start byte with a value of 0. In this case, the DMX status is incremented. If the start byte is not 0, a real transmission error occurred and the reception is aborted. Now we are waiting for the next BREAK again.

In the following, the byte counter is decremented with each incoming byte until the start address is reached. In this case, the DMX status is incremented again

The following bytes are the desired channels. When all data has been received, the DMX status is reset and we wait for the next BREAK.

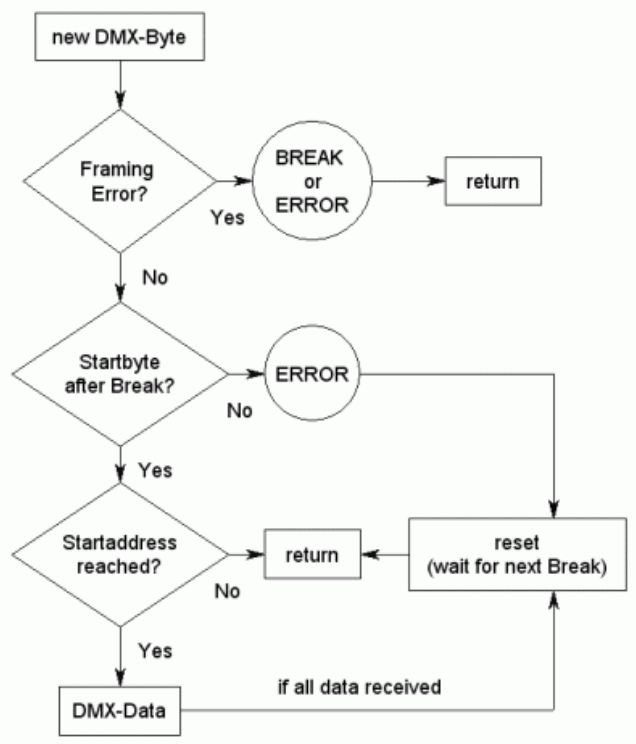

Figure 2: flow sheet of the state machine for DMX reception

The code was written with the ATMEL Studio 7.
The following variables and constants must be defined in the header file:

```
#define  F_CPU    (8000000UL)                  //oscillator freq.(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_CPU is the crystal frequency in Hz.

If USE_DIP is defined, the start address is obtained from the DIP switches of the transceiver. Otherwise, the start address can also be stored directly in DmxAddress.

The DMX data is stored in DmxRxField[].

With "init_DMX_RX()", the USART is initialised for DMX reception and the receive buffer is flushed.

With "get_dips()" the start address is read from the DIP switches of the Transceiver.