Januar 2015

Henne's Sites

AN018: Remote Device Management [Controller]

Inhalt

Einleitung

Bei RDM handelt es sich um eine Erweiterung des DMX512-Protokolls, die 2006 in der ANSI E1.20 durch die ESTA spezifiziert wurde.

Mittels RDM konnte der DMX-Bus um einen Rückkanal erweitert werden, so dass Empfänger wie Dimmerpacks oder Movinglights dem Pult ihren Status und andere Betriebsdaten mitteilen können. Auch lassen sich so einfach zentral vom Pult aus die Empfänger parametrieren.
Typische Aufgaben von RDM sind:

- Patchen von DMX-Empfängern
- Parametrierung von DMX-Empfängern
- Überwachung von Temperaturen, Spannungen und Strömen
- Leuchtmittelüberwachung
- Fehlerdiagnose
- Firmware-Updates

RDM-fähige und konventionelle DMX-Geräte können gemeinsam in einem Bus betrieben werden.

Trotz dieser viel versprechenden Möglichkeiten sollte man sich die Implementation eines RDM-Protokollhandlers aber genau überlegen:

- Nur wenige hochklassige Pulte arbeiten auch als RDM-Controller.
- Viele DMX-Splitter arbeiten nur unidirektional und sind deshalb **nicht** RDM-fähig.
- Jedes RDM-Gerät benötigt eine GUID. (vergleichbar mit einer MAC-Adresse)
- Zur Übermittlung von Betriebsdaten müssen auch Sensoren zu deren Ermittlung vorhanden sein.
- Der Download von Firmwares ist (noch) nicht normiert und somit nicht universell einsetzbar.
- Das Pult kennt zwar den Namen und den Typ von angeschlossenen DMX-Empfängern – nicht aber deren Anordnung, sodass vor einem Adress-Patch die Positionen der Geräte zunächst einmal identifiziert werden müssen.

Nutzungsbedingungen

Grundlagen
Anders als bei DMX werden bei RDM die Daten paketweise übertragen. Die folgende Tabelle zeigt den Aufbau eines RDM-Datenpaketes:

Slot Nr.	Bedeutung
0	alternate start code (0xCC)
1	sub start code (0x01)
2	Paketlänge in Bytes
3 - 8	Zieladresse (UID)
9 - 14	Quelladresse (UID)
15	Transaktionsnummer
16	Port ID / Typ der Antwort
17	Anzahl der wartenden Nachrichten
18 - 19	Sub-Device
20	Befehlsart
21 - 22	Parameter ID
23	Länge der Nutzdaten in Bytes
24 – (n-2)	Nutzdaten
(n-1) - n	Prüfsumme

Zahlen werden in Big-Endian (Motorola-Format) übertragen.
Die Paketlänge entspricht der Slot Nr. des ersten Prüfsummen-Bytes.
Die UIDs setzen sich aus einer 16bit Manufacturer-ID, die von der ESTA vergeben wird, und einer 32bit Device-ID zusammen.
Die Prüfsumme ergibt sich durch Summation aller Bytes des Paketes.

Zu Beginn einer Session werden alle RDM-Geräte (Responder) im Bus vom RDM-Pult (Controller) über eine Discovery-Funktion ermittelt:
Dazu verschickt der Controller eine Discovery-Nachricht an alle Busteilnehmer (Broadcast) mit der Unter- und Obergrenze des UID-Adressbereiches als Nutzdaten. Liegt die UID eines Responders innerhalb dieses Bereiches, so antwortet er mit seiner verschlüsselten UID.
Jeder gefundene Responder wird vom Controller anschließend gemuted.

Als Algorithmus wird von der ESTA eine Baumsuche vorgeschlagen.

Für detaillierte Informationen soll an dieser Stelle auf die E1.20 verwiesen werden.

Controller Library
Der Code wurde für den DMX-Transceiver Rev. 3.01 geschrieben, sollte aber auf die meisten AVRs der mega-Serie portierbar sein. Es wurde auf eine möglichst geringe Codesize Wert gelegt.

Als IDE wurde das AVR-Studio in Verbindung mit WinAVR genutzt.

- „init_RDM_TX ()“ initialisiert den RDM-Controller.
- „Discover(0)“ führt eine vollständige Device-Discovery durch.
- „Discover(1)“ führt eine inkrementelle Discovery durch, die nur nach zusätzlichen Geräten sucht.
- „Identify(n)“ fordert das n-te Gerät aus der Liste auf, sich zu identifizieren.
- „SetAddress(n, adr)“ sendet adr als Dmx-Startadresse an das n-te Gerät.
- „SetPersonality(n, pers)“ sendet pers als Betriebsart an das n-te Gerät.
- „GetDevInfo(n)“ fordert die Gerätedaten vom n-ten Gerät an. Die Verarbeitung der Antwort erfolgt an anderer Stelle. Der Aufbau der Antwort ist im Code erläutert.

Die Lib wird im Hauptprogramm ausgeführt und kommt ohne Interrupts aus. Ein Watchdog kann verwendet werden. Timer0 wird für die Erkennung von Timeouts verwendet. Bei einer abweichenden Taktfrequenz müssen ggf. Zeiten und Prescaler angepasst werden. Bevor ein Paket gesendet wird, muss der USART freigegeben werden (UCSRB=0). Auf diese Weise ist ein unterbrechungsfreier DMX-Datenverkehr gewährleistet.

F_OSC gibt die Quarzfrequenz in kHz an.

Funktionsweise
Zunächst wird über eine vollständige Discovery eine Liste mit allen RDM-fähigen Geräten im Bus angelegt.

Hierzu wird zunächst ein UnMute an sämtliche Geräte geschickt und anschließend Discovery-Aufforderungen versendet: Antworten mehrere Geräte gleichzeitig, stellen wir eine Datenkollision fest und halbieren das Suchintervall. Antwortet kein Gerät wechseln wir zur anderen Hälfte des Suchintervalles. Ist das Intervall ausreichend klein, dass nur noch ein Gerät antwortet, nehmen wir es in die Liste auf, muten es und beginnen die Suche über das gesamte Intervall erneut bis keine Geräte mehr antworten oder die Liste voll ist. (Die Suche ließe sich über Rekursionen weiter beschleunigen. Dies kann allerdings bei Leitungsproblemen oder fehlerhaft implementierten Respondern zum Absturz führen.)

Nach erfolgreicher Discovery können diverse Anfragen gezielt an einzelne Geräte verschickt werden.

Anmerkungen
Falls Sie die Library weiter optimieren oder gefundene Fehler fixen konnten, würde ich mich über eine Mail freuen. Auch Erweiterungen sind jederzeit willkommen ;-)