April 2008

Henne's Sites

AN017: Remote Device Management [Responder]

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.
- Trotz diverser Griffe in die Trickkiste werden für die RDM-Unterstützung gut 2kB Flash und 55Byte SRAM benötigt.

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

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.

Responder 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.

Mit „init_RDM()“ wird der RDM- und DMX-Empfang initialisiert.
„check_rdm()“ verarbeitet empfangende RDM-Pakte und sollte mit möglichst hoher Frequenz vom Hauptprogramm aufgerufen werden.

UID_0 und UID_1 bilden die ESTA Manufacturer-ID.
UID_2 bis UID_5 ergeben die Device-ID.

F_OSC gibt die Quarzfrequenz in kHz an.

Treffen neue DMX-Daten ein, wird das Flag „EVAL_DMX“ in der Variable Flags gesetzt. Die DMX-Daten befinden sich im DmxField[].

Ist DEBUG definiert, haben an PortA angeschlossene LEDs folgende Funktionen:
- PA0 wechselt den Status, wenn ein RDM-Paket mit gültigem Anfang eintrifft.
- PA1 wechselt den Staus, wenn das RDM-Paket in den Puffer passt.
- PA2 wechselt den Status, falls die Prüfsumme stimmt.
- PA3 wechselt den Status, falls auf einen Discovery-Request geantwortet wird.
- PA4 ist high, wenn das Device gemuted ist.

Die darauf folgenden Definitionen sind ein Auszug aus der E1.20.

Bislang noch nicht implementierte Funktionen:
- Sub-Devices
- Queued Messages
- Acknowledge Timer
- Einige Parameter

Funktionsweise
Lautet das Startbyte nach einem Break 0xCC und das darauf folgende Byte 0x01, so handelt es sich um ein RDM-Paket. Ist der Puffer leer, wird das Paket weiter empfangen.
Bei dem dritten Byte handelt es sich um die Paketlänge. Passt die Nachricht in den Puffer, wird die angegebene Zahl an Bytes empfangen und gepuffert. Parallel dazu wird die Prüfsumme in „RxCount_16“ berechnet.
Sind alle Datenbytes empfangen, wird die berechnete Prüfsumme mit der übertragenen verglichen und bei Identität das Flag „EVAL_RDM“ gesetzt.

Beim nächsten Aufruf von „check_rdm()“ wird nun festgestellt, dass eine neue Nachricht eingetroffen ist.
Sind wir der Empfänger oder die Nachricht wurde gebroadcasted (Dest-ID= 0xFFFFFFFFFFFF), schauen wir uns den Typ der Nachricht an:
Wurde auf die Nachricht eine Antwort implementiert, so antworten wir damit – andernfalls antworten wir, dass wir keine Antwort auf die Nachricht haben ;-)

Um Platz zu sparen, werden vordefinierte Antworten aus Tabellen im Flash gelesen.

Zum Schluss wird mit „respondMsg()“ das Antwortpaket gebildet und verschickt.

Bei empfangenen Paketen für uns wechselt die grüne LED ihren Status.
Bei gesendeten Paketen wechselt die rote LED ihren Status.

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