Kommunikationsprotokoll LC444

4x4x4 LED-Cube

Version 1.0

Autor:	eQ-3 Entwicklung GmbH
Erstelldatum:	3. Februar 2011
Letzte Aktualisierung:	10.Februar.2011, 10:20
Dateiname:	Kommunikationsprotokoll LC444.doc

Tel: +49 (0)491 6008 700
Fax: +49 (0)491 6008 99 700
Mail: info@
entwicklung.eq-3.de

eQ-3
Entwicklung GmbH
Maiburger Straße 36
26789 Leer

Geschäftsführer
Prof. H.-G. Redeker

Registergericht:
Amtsgericht Aurich
HRB 110388

Zentrale:
Telefon: +49 (0)491 6008 700
Telefax: +49 (0)491 6008 99 700

Internet:
www.eQ-3.de

1 Inhaltsverzeichnis

2 Änderungsverzeichnis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor
Nr.	Datum	Version			
1	03.02.11	1.0			Wie

Tabelle 1: Änderungsverzeichnis

3 Ziel

Mit diesem Dokument wird die Kommunikation zwischen dem 4x4x4-LED-Cube und einem angeschlossenen Computer erläutert.

4 Das PC-Interface (Schnittstelle)

Der LC444 verfügt auf der Platine über einen USB-zu-UART-Schnittstellenwandler vom Typ CP2102 der Firma Silicon Laboratories. Durch diesen Baustein kann ein Computer mit der entsprechenden PC-Software direkt über die USB-Schnittstelle mit dem LC444 kommunizieren, dazu ist nur der beigefügte Treiber zu installieren. Es wird keine zusätzliche Hardware benötigt.
Für die Kommunikation zwischen der PC-Software und dem LC444, kann der USB-zu-UART-Schnittstellenwandler als Direct-USB-Device oder als Virtual-COM-Port betrieben werden. Näheres zu diesem Thema ist dem Dokument
„Anwendung des CP2102 ID Changer.pdf“
zu entnehmen.

5 Schnittstellenparameter

Wird der LC444 mittels eines Virtuellen COM-Ports (VCP) betrieben, sind folgende Einstellungen für die Kommunikation vorzunehmen.

Baudrate: 115200
Datenbits: 8
Parität: keine
Stoppbits: 1

6 Aufbau des Datenrahmens

Jeder gesendete Befehl und die dazugehörige Antwort wird in einem definierten Datenrahmen übertragen. Innerhalb dieses Rahmens werden die eigentlichen Nutzdaten übertragen.

Die folgende Tabelle zeigt den grundlegenden Aufbau des Rahmens.

Startzeichen	Paketnummer	Nutzdatenlänge	Nutzdaten	Checksumme
1 Byte	1 Byte	2 Byte	n Bytes	2 Byte

Tabelle 2: Datenrahmen

Startzeichen	Zeigt den Anfang eines Datenpakets an
Paketnummer	Indizierung der Datenpakete (ist immer NULL)
Nutzdatenlänge	Länge der nicht codierten Nutzdaten in Bytes
Nutzdaten	Beinhaltet ein Kommando und eventuelle zusätzliche Daten
Checksumme	16-Bit Checksumme der gesendeten Daten

6.1 Startzeichen

Startzeichen	Beschreibung
0x02	Eindeutiger Indikator für den Anfang eines Datenrahmens

Tabelle 3: Startzeichen

6.2 Paketnummer

Paketnummer	Beschreibung
0x00	Die Paketnummer wird hier immer mit NULL gesendet

Tabelle 4: Paketnummer

6.3 Nutzdatenlänge

Nutzdatenlänge	Beschreibung
0x??	Low-Byte der Nutzdatenlänge
0x??	High-Byte der Nutzdatenlänge

Tabelle 5: Nutzdatenlänge

6.4 Nutzdaten

Nutzdaten		Beschreibung
Befehl	0x??	Auszuführender Befehl
Parameter 1	0x??	Erster übermittelter Parameter
•	0x??	•
•	0x??	•
•	0x??	•
Parameter n	0x??	Letzter übermittelter Parameter

Tabelle 6: Nutzdaten

6.5 Checksumme

Checksumme	Beschreibung
0x??	High-Byte der 16-Bit Checksumme
0x??	Low-Byte der 16-Bit Checksumme

Tabelle 7: Checksumme

6.5.1 Checksummenbildung

Nach den Nutzdaten folgt die 2 Byte lange CRC16-Checksumme. Sie wird mit dem Polynom 0x8005 nach dem allgemein bekannten Berechnungsverfahren bestimmt. Die Checksumme wird über alle übertragenen Zeichen gebildet, für die beiden Checksummenbytes selbst wird jeweils eine NULL eingesetzt.

Folgender Quellcode zur CRC-Berechnung kann genutzt werden:

```
//Declaration of functions
void crc16_init(void);
void crc16_checksum(unsigned char);

// CRC Polynom for CRC16 calculation
#define CRC16_POLYNOM 0x8005;

// global variable for CRC16 calculation
unsigned short crc16_register;

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
// Initialize the CRC16 checksum variable
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
void crc16_init(void)
{
      crc16_register=0xffff;  // Start value of checksum
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
// Calculate the CRC16 checksum
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
void crc16_checksum(unsigned char data)
{
      unsigned char q;
      unsigned char flag;
      data&=0xff;
      for(q=0;q<8;q++)
      {
            flag=(crc16_register & 0x8000)!=0;
            crc16_register<<=1;
            if(data&0x80)
            {
                  crc16_register|=1;
            }
            data<<=1;
            if(flag)crc16_register ^= CRC16_POLYNOM;
      }
}
```

7 Datencodierung

Um einen eindeutigen Anfang im Datenrahmen zu ermöglichen, ist das erste Byte eines neuen Datenpakets immer das aus dem ASCII-Zeichensatz stammende Startzeichen „STX“ (0x02). Dieses Startzeichen, darf innerhalb des Datenrahmens kein zweites mal vorkommen. Aus diesem Grund müssen entsprechende Zeichen umkodiert werden. Dazu wird das ASCII-Steuerzeichen „ENQ“ (0x05) verwendet und das oberste Bit des zu kodierenden Zeichens gesetzt.

< STX >	→	< ENQ ><0x82>
< ENQ >	→	< ENQ ><0x85>

8 Befehlsübersicht

Beschreibung	Befehl	Parameter (Anzahl an Bytes)	Parameterinformationen	Antwort
PC-Verbindung aktivieren	x	Status (1 Byte)	0x00 PC-Verbindung deaktivieren	ACK
			0x01 PC-Verbindung aktivieren	NAK (Parameter falsch oder bereits de-/aktiviert)
Geräteversion auslesen	V	Kein Parameter notwendig		Version (2 Bytes); 100 bedeutet v1.00
Header-Informationen der LED-Sequenzen empfangen	e	Kein Parameter notwendig		Header-Informationen der LED-Sequenzen aller 20 Speicherplätze (440 Byte)
Header-Informationen der LED-Sequenzen senden	E	Header-Informationen aller LED-Sequenzen (440 Byte)		ACK
				NAK (Parameter falsch oder bereits de-/aktiviert)
Sequenzdaten im Flash speichern	S	Zielpage (1 Byte)	Wertebereich der Zielpage 0 - 4095	ACK
		LED-Bilder (528 Byte)		NAK (Parameter falsch oder außerhalb des Wertes)
Löschen einer vorhandenen Sequenz	K	Sequenznummer (1 Byte)	Wertebereich der Sequenznummer 0 - 19	ACK
				NAK (Parameter falsch oder außerhalb des Wertes)
Werkseinstellung herstellen	R	Kein Parameter notwendig		ACK
				NAK (Parameter falsch oder außerhalb des Wertes)
Zum Bootloader springen	!	Kein Parameter notwendig		ACK
				NAK (Parameter falsch oder außerhalb des Wertes)

Tabelle 8: Befehlsübersicht

8.1 PC-Verbindung aufbauen

Um einen Datenaustausch mit dem LC444 und einem PC-Programm zu ermöglichen, muss der LC444 zunächst in den Transfermodus gebracht werden. In diesem Modus werden keine Visualisierungen dargestellt, um den Transfer von Daten zu beschleunigen.

Um den Transfermodus zu aktivieren, wird der entsprechende Befehl mit einem weiteren Statusbyte an den LC444 gesendet.
Mittels des Statusbytes wird entweder der Transfermodus aktiviert oder deaktiviert.

<Startzeichen>	<Paketnummer>	<Nutzdatenlänge>	<Befehl>	<Status>	<Checksumme>
1 Byte	1 Byte	2 Byte	1 Bytes	1 Bytes	2 Byte

Tabelle 9: Verbindungsaufbau

8.2 Firmwareversion auslesen

Mittels des Befehls „V“ kann die aktuelle Firmwareversion des LC444 ausgelesen werden. Die Antwort beinhaltet die Versionsnummer als 16-Bit Wert.

<Startzeichen>	<Paketnummer>	<Nutzdatenlänge>	<Befehl>	<Checksumme>
1 Byte	1 Byte	2 Byte	1 Bytes	2 Byte

Tabelle 10: Firmware auslesen

8.3 Auslesen der Header-Informationen

Mit Hilfe der im EEPROM des LC444 gespeicherten Header-Informationen, werden die Speicherplätze der LED-Sequenzen klar definiert.

Eine Anfrage liefert immer die Informationen über alle 20 Speicherplätze (440 Byte).

Die Informationen werden als Array aus 20 Elementen der folgenden Struktur vom LC444 gesendet:

```
typedef struct _Sequence_EEPROMData
{
      unsigned short          uiStartPage;          // Startpage der Sequence im Datenflash
      unsigned short          uiStartAdress;        // Startadresse der Sequence
      unsigned short          uiNumberofImages;     // Anzahl der Images
      unsigned char           ucNameLength;         // Laenge des Namens
      unsigned char           ucName[15];           // Name der Sequenzy
}Sequence_EEPROMData;
```

8.4 Senden neuer Header-Informationen

Wie beim Auslesen werden auch beim Senden immer alle 20 Header-Informationen gesendet.

Der LC444 erwartet die Header-Informationen in der gleichen Weise, wie sie empfangen wurden.

<Startzeichen>	<Paketnummer>	<Nutzdatenlänge>	<Befehl>	<Sequence_EEPROMData[0]> <...> <Sequence_EEPROMData[19]>	<Checksumme>
1 Byte	1 Byte	2 Byte	1 Bytes	22 Bytes	2 Byte

Tabelle 11: Neue Header-Informationen senden

8.5 Senden der neuen Sequenzdaten

Wie bekannt, bestehen die Sequenzen aus einer Abfolge von einzelnen LED-Bildern. Die wiederum bestehen aus $4^3 = 64$ Helligkeitsinformationen zu je 4 Bit und einem Byte mit einer Zeitinformation. Es werden also 33 Byte je LED-Bild benötigt.

Die Helligkeitsinformationen sind zu 16 Bit pro LED-Reihe , also immer vier übereinanderliegende LEDs, zusammengefasst.

```
typedef struct _Image
{
      unsigned short LED_Image[16]; // Beinhaltet die Helligkeitswerte der einzelnen LEDs
      unsigned char Dauer;          // Wert zwischen 1 und 255 --> 10 ms bis 2550 ms
}Image;
```

Diese Daten werden in dem externen Flashspeicher des LC444 gespeichert. Da der Flashspeicher mit einer Speicherseitengröße (Page) von 528 Bytes arbeitet, passen genau 16 Einzelbilder auf eine Page.

Eine Sequenz besteht immer aus Stücken zu je 528Bytes. Das bedeutet, es werden immer ganze Pages des Flashspeichers verwendet. Falls eine Page nicht komplett mit Bilddaten gefüllt wird, kann der restliche Bereich nicht für eine weitere Sequenz genutzt werden.
Dieses Verfahren wird deshalb genutzt, da so die Defragmentierung des Flashspeicher einfacher durchzuführen ist.

Um also eine erstellte Sequenz an den LC444 zu senden, müssen die Helligkeitswerte und Zeitinformation in Pakete zu je 528 Byte gepackt werden. Anschießend kann dieses Paket dann mit der Information auf welche Page die Daten gespeichert werden sollen an den LC444 gesendet werden. Dieser Vorgang wird solange wiederholt, bis alle Sequenzdaten an den LC444 gesendet wurden, falls eine Page nicht komplett genutzt wird, kann der Rest frei bleiben.

<Startzeichen>	<Paketnummer>	<Nutzdatenlänge>	<Befehl>	<Zielpage>	<LED Bild 1>	<...>	<LED-Bild n>	<Checksumme>
1 Byte	1 Byte	2 Byte	1 Bytes	1 Bytes	33 Bytes		33 Bytes	2 Byte

Tabelle 12: Neue Sequenzdaten senden

Anschließend müssen die aktuellen Header-Informationen (EEPROM-Daten) an den LC444 gesendet werden, in denen nun auch die neue Sequenz enthalten ist.

Neue Sequenzen werden immer an das Ende angehängt. Die durch das Löschen einer vorhandenen Sequenz entstehende Lücke im Flashspeicher, wird automatisch vom LC444 durch eine Defragmentierung behoben.

8.6 Löschen einer vorhandenen Sequenz

Um eine Sequenz aus dem LC444 zu löschen, wird die entsprechende Sequenznummer mit dem Befehl zum Löschen an den LC444 gesendet. Dabei besitzt die erste Sequenz die Nummer NULL. Alle nachfolgenden Sequenzen sind entsprechend ihrer Position höher nummeriert.

<Startzeichen>	<Paketnummer>	<Nutzdatenlänge>	<Befehl>	<Sequenznummer>	<Checksumme>
1 Byte	1 Byte	2 Byte	1 Bytes	1 Bytes	2 Byte

Tabelle 13: Sequenz löschen

Anschließend müssen die Header-Informationen (EEPROM-Daten) vom LC444 neu ausgelesen werden, aus denen nun die gelöschte Sequenz entfernt wurde.

9 Tabellenverzeichnis