Kommunikationsprotokoll
Signalgenerator ⇔ Computer

Hendrik Lüth, LabConnect

4. Mai 2015

Inhaltsverzeichnis

1 Allgemeines

In diesem Dokument wird die Datenübertragung zwischen dem Mikrocontroller des Signalgenerators und eines Computers definiert. Die Daten werden über den USB-Bus übertragen. Die USB-Spezifikationen[1] enthalten alle nötigen Informationen, welche für Kommunikationen über den Bus nötig sind.
Der Signalgenerator wird als HID (Human Interface Device) am Computer angemeldet, wodurch keine Installation von zusätzlichen Treibern nötig ist. Die Übertragung der Daten erfolgt über HID-Reports. Zum aktuellen Zeitpunk benutzt LabConnect für den Signalgenerator die VID 0x1209 und die PID 0x2222, welche unter Linux als GenericHID-Gerät von InterBiometrics zu finden ist. Da es sich bei der VID um eine VID handelt, welche für OpenSource Projekte gedacht ist, ist es fraglich ob der Signalgenerator je richig angezeigt wird. Von dem Kauf einer eigenen VID für LabConnect wird derzeit abgesehen.

2 Aufbau einer Kommunikationseinheit

Eine Kommunikationseinheit, im folgenden als Paket"bezeichnet, besteht aus 13 Byte. Jedes Paket hat einen 1 Byte großen Header an seinem Anfang und einen 1 Byte großen Tail an seinem Ende. Der Header enthält die Paket-ID, an welcher sich Flussrichtung der Daten und Art der Daten erkennen lassen. Ist das 5. Bit des Headers gleich 0, so ist die Flussrichtung der Daten vom Computer zum Mikrocontroller, ist es gleich 1 vom Mikrocontroller zum Computer. An den unteren 4 Bit lässt sich der Typ des Paketes erkennen. In der folgenden Tabelle sind alle Befehle nach Paket-ID sortiert aufgelistet:

Paket-ID	Flussrichtung	Bezeichnung	Größe der Daten
0x00	PC→ μC	Config-Request	1 Byte
0x01	PC→ μC	Set-Command	12 Byte
0x02	PC→ μC	Data-Request	0 Byte
0x03	PC→ μC	Error/Status-Request	0 Byte
0x10	μC →PC	Config-Response	10 Byte
0x12	μC →PC	Data-Response	12 Byte
0x13	μC →PC	Error/Status-Response	5 Byte

3 Aufbau einzelner Befehle

In diesem Abschnitt wird der Aufbau einzelner Befehle erläutert. Ob ein Befehl vom Computer zum Signalgenerator geht ist an der Paket-ID zu erkennen. Dies ist im Abschnitt „Aufbau einer Kommunikationseinheit“beschrieben.

[1] http://www.usb.org/developers/docs/usb20_docs/usb_20_031815.zip

3.1 Computer → Signalgenerator

3.1.1 Config-Request

Der Config-Request steht am Anfang jeglicher Kommunikation zwischen Signalgenerator und Computer nach dem anstecken des Signalgenerators. Der Config-Request fragt beim Signalgenerator diverse Kalibrierungsdaten wie die Frequenz des Refferenztaktes oder die Boot-Daten an.

Byte	Wert	Beschreibung
0	0x00	Paket-ID
1	0x55	Prüfdaten, damit der Inhalt des Paketes nicht null ist. Der Wert ist auf 0x55 festgesetzt.

3.1.2 Set-Command

Mit dem Set-Command werden alle nötigen Informationen wie Frequenz, Registerwerte für die digitalen Potentiometer und Bootdaten übergeben. Die Folgende Tabelle zeigt den Aufbau:

Byte	Wert	Beschreibung
0	0x01	Paket-ID
1	*	Diese beiden Bytes enthalten die Daten für das
2	*	Kontrollregister des AD9833.
3	*	Diese vier Byte enthalten die Daten für das
4	*	Frequenzregister des AD9833. Die Berechnung
5	*	Dieser Werte ist im Verlauf dieses Dokumentes
6	*	erklärt.
7	*	In diesen beiden Bytes sind die Registerwerte
8	*	des Digi-Poti für die Ausgangsspannung enthalten.
9	*	In diesen beiden Bytes sind die Registerwerte
10	*	des Digi-Poti für die Offset-Spg. enthalten.
11	*	Multiplexer
12	*	Bootdaten

3.1.3 Data-Request

Nach einem Config-Request werden die Daten ausgewertet. Sollten die Bootdaten anzeigen, dass bereits beim einschalten des Signalgenerators die gespeicherte Konfiguration geladen wurde, so wird ein Data-Request gesendet, um herauszufinden wie die Konfiguration ist um sie später in der graphischen Oberfläche anzuzeigen. Dieses Paket hat keine Nutzdaten.

Byte	Wert	Beschreibung
0	0x02	Paket-ID

3.1.4 Error/Status-Request

Ein Error/Status-Request kann zu jedem Zeitpunk, z.B. nach einer Datenübertragung gestellt werden um den Status des Systems zu prüfen. Dieses Paket enthält keine Nutzdaten.

Byte	Wert	Beschreibung
0	0x03	Paket-ID

3.2 Signalgenerator → Computer

3.2.1 Config-Response

Byte	Wert	Beschreibung
0	0x10	Paket-ID
1	*	Seriennummer
2	*	Bootdaten
3	*	Kalibrierungs-Daten des DDS-IC
4	*	Frequenz des MCLK in Hz
5	*	
6	*	
7	*	Kalibrierungs-Daten für das Digi-Poti
8	*	Multiplikatoren für Berechnung
9	*	Wert der Ausgangsspannung in mV_{ss}
10	*	

3.2.2 Data-Response

Dieses Paket ist die Antwort auf einen Data-Request. Es enthält die selben Daten wie ein Set-Command. Die Daten müssen vom Host dann in Frequenzen und Spannungen umgerechnet werden.

Byte	Wert	Beschreibung
0	0x01	Paket-ID
1	*	Diese beiden Bytes enthalten die Daten für das
2	*	Kontrollregister des AD9833.
3	*	Diese vier Byte enthalten die Daten für das
4	*	Frequenzregister des AD9833. Die Berechnung
5	*	Dieser Werte ist im Verlauf dieses Dokumentes
6	*	erklärt.
7	*	In diesen beiden Bytes sind die Registerwerte
8	*	des Digi-Poti für die Ausgangsspannung enthalten.
9	*	In diesen beiden Bytes sind die Registerwerte
10	*	des Digi-Poti für die Offset-Spg. enthalten.
11	*	Multiplexer
12	*	Bootdaten

3.2.3 Error/Status-Response

Der Error/Status-Response enthält alle Error/Status-Meldungen die angefallen sind.

Byte	Wert	Beschreibung
0	0x13	Paket-ID
1	*	
2	*	
3	*	Error-Codes, bis zu 5 Stück.
4	*	
5	*	

4 Berechnung der Registerwerte

In dieser Sektion ist aufgelistet, wie die Registerwerte für den Signalgenerator berechnet werde. Es ist sich zwingend an die Formeln zu halten, da der Signalgenerator sonst nicht die gewünschten Ausgangssignale liefert.

4.1 Frequenz

Die Frequenzregister sind die Register, welche die Frequenz des Ausgangssignals kontrollieren. Mit einer Formel muss in Abhängigkeit vom Referenztakt und der gewünschten Frequenz des Ausgangssignals der Wert für dieses Register errechnet werden. Hier die allgemeine Formel:

$$Registerwert = F_{out} \div \frac{F_{MCL}}{2^{28}}$$

Und hier ein Beispiel für die Werte $F_{MCLK} = 25MHz$ und $F_{out} = 7,325MHz$:

$$Registerwert = F_{out} \div \frac{F_{MCL}}{2^{28}} = 7,325MHz \div \frac{25MHz}{2^{28}}$$
$$Registerwert = 78651588,61 \approx 78651589$$

In diesem Fall ist es möglich zu runden, da ein Bit nur c.a. 0,093Hz entsprechen. Nun muss der Wert noch in Binär umgerechnet werden:

Dec"78651589"=Bin"100 1011 0000 0010 0000 1100 0101"

Um den Wert in das Frequenzregister zu schreiben wird der binäre Wert in LSBs und MSBs aufgeteilt und hängen die Adressierung des Registers "01ünd die fehlenden Nullen, um auf 28Bit zu kommen, davor:

MSBs: 0101 0010 1100 0000
LSBs: 0110 0000 1100 0101

Dies ist ein Beispielcode in C++, in dem die entsprechenden Register berechnet werden:

```
float mclk = 25000000, register_size = 268435456;
float teiler = mclk / register_size;
int f_regwert = frequenz / teiler;
//block1=lsb Block4=msb
unsigned char block1, block2, block3, block4;

block1 = f_regwert;
f_regwert = f_regwert >> 8;
block2 = f_regwert;
block2 = (block2 | 0x40) & (~0x80);
f_regwert = f_regwert >> 6;
block3 = f_regwert;
f_regwert = f_regwert >> 8;
block4 = f_regwert;
block4 = (block4 | 0x40) & (~0x80);
```

4.2 Signalform

Für das Register der Signalform gibt es nicht viel zu berechnen, da es nur drei Signalformen gibt. Die 2 Byte, mit denen die Signalform gesteuert wird können folgende Werte annehmen:

Wert	Signalform
0x2000	Sinus
0x2002	Dreieck
0x0000	Rechteck

Es ist hierbei darauf zu achten, dass auch der Zustand des Multiplexers angepasst wird, da es ansonsten zu unerwünschten Ausgangsspannungen kommen kann.

4.3 Spitzenspannung

Die Amplitude des Ausgangsignals lässt sich über den Multiplexer und das Digitalpotentiometer einstellen. Hierzu wird die Ausgangsspannung des DDS-IC als Berechnungsgrundlage hinzu gezogen. Mit dem Multiplexer kann man auswählen, ob das Signal direkt auf den Verstärker gegeben wird oder ob eine Teilung von 5:1 bzw eine Verstärkung von ungefähr 3 stattfinden soll, bevor das Signal auf den Verstärker gegeben wird. Welchen Wert das entsprechende Byte annehmen muss ist unter SSonstige Registerïm Unterabschnitt Multiplexernachzulesen. Die Registerwerte des Digitalpotentiometers werden wie folgt berechnet:

$$Registerwert_{gesamt} = \left(\frac{100k\Omega}{\frac{U_{Ausgang}}{U_{Eingang}} - 1} - 2,2k\Omega\right) \div \frac{200k\Omega}{512}$$

Hier ein Beispiel für die Werte $U_{Ausgang} = 7,5V_{ss}$ und $U_{Eingang} = 1V_{ss}$:

$$Registerwert_{gesamt} = (\frac{100k\Omega}{\frac{7,5V_{ss}}{1V_{ss}} - 1} - 2,2k\Omega) \div \frac{200k\Omega}{512}$$

$$Registerwert_{gesamt} \approx 13184,61 \div 390,625$$

$$Registerwert_{gesamt} \approx 34$$

Da der Registerwert für zwei in Reihe geschaltete Widerstände gilt muss dieser Wert noch auf beide Register aufgeteilt werden. Sollte das Ergebnis eine ungerade Bit-Zahl annehmen so erhält eines der Register einfach ein Bit mehr. Daraus ergibt sich, dass beide Register den dezimalen Werte "17"haben bzw 0x11 in Hexadezimal.
Der Folgende Beispielcode ist in C# geschrieben und berechnet den Gesamtwert beider Register.

```
int umax = 12000, bits = 512, register1, register2;
int ergebnis = u_amplitude_mv / (umax / bits);

if (510 < ergebnis)
{
  ergebnis = 510;
}

if (ergebnis%2==0)
{
  register1 = 255 - ergebnis / 2;
  register2 = 255 - ergebnis / 2;
}
else
{
  ergebnis = ergebnis- 1;
  register1 = 255-((ergebnis / 2) + 1);
  register2 = 255 - ergebnis / 2;
}
```

4.4 Offsetspannung

4.5 Sonstige Register

4.5.1 Bootdaten

Bootdaten	Beim Boot laden	Werte Speichern
0x00	Nein	Nein
0x01	Nein	Ja
0x10	Ja	Nein
0x11	Ja	Ja

4.5.2 Multiplexer

5 Errorcodes

Fehlercode	Beschreibung des Fehlers
0x00	Kein Fehler
0x01	Keine gültige Package-ID
0x02	Transportdaten des Config-Requests sind falsch
0x03	Digitalpotentiometer ist nicht erreichbar
0x04	
0x05	
0x06	
0x07	