Das Software-Interface des Bootloaders

Autor: sprut (www.sprut.de)
Stand: 24.04.2007

-

1 Inhaltsverzeichnis

-

2 NUTZUNGSBEDINGUNGEN:

DIE SOFTWARE DARF OHNE ENTRICHTUNG EINER LIZENZGEBÜHR BENUTZT WERDEN. DAS GILT FÜR DIE PRIVATE UND GEWERBLICHE NUTZUNG.

DIE PUBLIKATION DER SOFTWARE ERFOLGT "AS IS". FÜR DIE EINHALTUNG ZUGESICHERTER EIGENSCHAFTEN ODER FÜR SCHÄDEN, DIE DURCH DEN EINSATZ ENTSTANDEN SEIN KÖNNTEN, ÜBERNIMMT DER AUTOR KEINERLEI HAFTUNG. SIE NUTZEN DIE SOFTWARE AUF EIGENE GEFAHR!

3 Einleitung

Dieses Dokument beschreibt das Softwareinterface des Bootloaders.

4 Das USB-Interface

Der Bootloader ist ein bus-powered USB-2.0 Gerät.

Interface	USB2.0 full speed
Stromversorgung	bus powered
Stromaufnahme	< 100 mA
VID	0x04D8
PID	0xFF0B
Anzahl der USB-Konfigurationen	1
Anzahl der Interfaces	1
Anzahl der Endpunkte	2

Endpunkt1:

Datenrichtung	OUT
Betriebsart	BULK
Puffergröße	64 Byte

Endpunkt2:

Datenrichtung	IN
Betriebsart	BULK
Puffergröße	64 Byte

USB-Timeouts:

Schreibzugriffe	100 ms
Lsezugriffe	1000 ms

5 Bootloader und Firmware

Der Bootblock des Steuer-PIC (Adressen 0x0000 ... 0x07FF) wird von der Bootloader Software eingenommen. Der restliche Adressbereich des Steuer-PIC-Programmspeichers steht für die Firmware zur Verfügung.

-

Bootloader und Firmware sind unabhängige und eigenständig lauffähige Programme.

Der Bootloader startet nach einem Reset des Steuer-PIC wenn mindestens eine der beiden folgenden Bedingungen erfüllt ist:

1. Das Pin 1 des Steuer-PIC liegt auf low-Pegel
2. Der Inhalt der EEPROM-Zelle ist der Adresse 0xFE ist 0xFF.
   (nicht beim Bootloader-2)

Ist keine der beiden Bedingungen erfüllt, startet beim Reset die Firmware des USB-Geräetes.

6 Konfiguration des PIC

Der Bootloader legr die Konfiguration des PIC fest.

Jeden Bootloader gibt es jeweils für drei unterschiedliche Quarzfrequenzen des PIC (4 MHz, 8 MHz und 20 MHz). Die durch den Bootloader mit USBoot nachgeladene Firmware hat keinen Einfluß auf die Konfiguration des PIC. Die ist wie folgt festgelegt:

- Externer Quarz/Resonator-Takt: 4, 8, 20 MHz (ja nach Bootloderversion)
- Tosc: 48 MHz
- Tcycl: 12MHz
- Tusb: 48MHz
- Power-Up-Timer: aktiv
- Brown-out-Reset: aus
- USB-Voltage-Regulator:aktiv
- Watchdog-Timer:aus
- MCLRE-Pin:digital-IO RE3
- PBADEN: aus
- Low-Voltage-Programming: aus
- Codeprotection: aus
- Write-Protect für Boot-block: aktiv

Das Softwareinterface des Bootloaders erlaubt zwar prinzipiell auch die Manipulation der PIC-Konfiguration, die Windowssoftware USBoot (V 3.0) unterstützt das aber aus Sicherheitsgründen noch nicht.

Spätere USBoot-Versionen werden möglicherweise die Veränderung unkritischer Konfigurationsparameter erlauben, die Takt- und USB-relevanten Parameter werden aber weiterhin geschützt bleiben.

-

7 Bootloader

7.1 Grundlagen

Der folgende Abschnitt ist nur von Interesse, wenn man selbst eine Software schreiben will, die mit Hilfe des Bootloaders den PIC brennen kann.
Wer unter Windows mit dem Bootloader ein HEX-File in den Flash des PIC brennen will, der greift am Besten zur Fertiglösung: dem Programm USBoot.

Der Bootloader basiert auf den Microchip-Bootloader zum „PICDEM USB FS DEMOBOARD“.

Die Kommunikation erfolgt über beide Endpunkte und wird immer vom PC initiiert. Sie läuft immer nach folgendem Schema ab:

1. Der PC schreibt einen Datenblock in den out-Endpunkt.
2. Der Bootloader arbeitet daraufhin eine Aufgabe ab.
3. Der Bootloader schreibt Daten in den in-Endpunkt
4. Der PC list den Datenblock aus dem in-Endpunkt

Auch bei Aufgaben, die eigentlich keinen Datentransport zum PC erfordern, wird wenigstens ein 1 Byte-langer Datenblock als Quittung zum PC zurückgegeben.

Die nachfolgende Tabelle zeigt die Struktur der Datenblöcke, die zwischen PC und Bootloader ausgetauscht werden. Die maximale Länge eines Datenblocks ist auf 64 Byte begrenzt.
Die konkrete Länge hängt vom Kommando ab. Wenn ein Kommando weder eine Adresse noch Nutzdaten benötigt, dann kann ein Datenblock auch aus nur einem Byte (dem Kommando selbst) bestehen.

Wird vom PC ein Datenblock gesendet, der für das konkrete Kommando zu lang ist, dann werden die überflüssigen Bytes am Ende des Datenblocks ignoriert.
War der Datenblock zu kurz, und es fehlen essenzielle Daten am Ende des Datenblocks, dann werden die Daten benutzt, die sich zufällig im USB-Pufferspeicher befinden.

Generelle Datenstruktur :

	Adresse	**Bedeutung**
Kommando	0x00	Beschreibt die Aufgabe
Datenlänge	0x01	Anzahl der Datenbytes
Adresse low	0x02	Bits 0..7 des Adresse
Adresse high	0x03	Bits 8..15 des Adresse
Adresse upper	0x04	Bits 16..23 des Adresse
Datenbyte 1	0x05	
Datenbyte 1	0x06	
...	...	
Datenbyte n	0xFF	

Folgende Kommandos werden vom Bootloader unterstützt:

Name des Kommandos	Code im Datenblock
READ_VERSION	0x00
READ_FLASH	0x01
WRITE_FLASH	0x02
ERASE_FLASH	0x03
READ_EEDATA	0x04
WRITE_EEDATA	0x05
READ_CONFIG	0x06
WRITE_CONFIG	0x07
RESET	0xFF

Wird ein Datenblock mit einem anderen (nicht definiertem) Kommando empfangen, so wird der Datenblock vom Bootloader ignoriert. Es erfolgt auch keine Rücksendung eines Antwort-Datenblocks.

7.2 Befehle für den Bootloader

7.2.1 READ_VERSION

Da ich nur eine feste VID_PID für meine USB-Geräte habe, melden sich alle meine Geräte beim PC mit dieser VID_PID an. Mit dem READ_VERSION-Kommando kann man aus dem Gerät zwei Kennbytes auslesen. Eines identifiziert das Gerät, ein zweites kennzeichnet die Softwareversion/Firmwareversion des Gerätes.

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x00

Der Bootloader sendet den Gerätecode 0x01.

Bootloader -> PC

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x00
0x01	keine	-
0x02	Version	-
0x03	Gerät	0x01

7.2.2 READ_FLASH

Mit diesem Befehl kann der Inhalt des Programmspeichers des Steuer-PIC ausgelesen werden. Es können maximal 59 aufeinanderfolgende Bytes auf ein Mal ausgelesen werden. Die Adresse gibt an, ab welcher Adresse im Flash-Programmspeicher mit dem Auslesen begonnen werden soll

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x01

-

0x01	Datenlänge	1 .. 59
0x02	Adresse low	
0x03	Adresse high	
0x04	Adresse upper	

Der Bootloader sendet den gleichen Datenblock zurück, an den aber die gesuchten Bytes angehangen wurden.

PC -> Bootloader

Adresse	**Bedeutung**	**Inhalt**
0x00	Kommando	0x01
0x01	Datenlänge	1 .. 59
0x02	Adresse low	
0x03	Adresse high	
0x04	Adresse upper	
0x05	Datenbyte 1	
...	...	...
0xXX	Datenbyte n	

7.2.3 WRITE_FLASH

Mit diesem Befehl kann der Programmspeichers des Steuer-PIC beschrieben werden.
Es ist zwingend nötig, vor dem Schreiben den Flash-Speicherbereich mit dem ERASE_FLASH-Kommando zu löschen. Ansonsten werden die neuen Daten mit den alten Speicherinhalten UND-Verknüpft.

Es werden genau 16 aufeinanderfolgende Bytes auf ein Mal geschrieben. Die Adresse gibt an, ab welcher Adresse im Flash-Programmspeicher mit dem Schreiben begonnen werden soll. Die Startadresse muss am Beginn eines 16-Byte-Blocks liegen. Folglich sind die unteren 4 Bits der Adresse „0000“.

Wird eine andere Adresse angegeben, dann schreibt der Bootloader die Daten trotzdem ab dem Block-Anfang in den Flash. Ist die Datenlänge kleiner als 16 Byte, dann wird nichts geschrieben.

PC -> Bootloader

Adresse	**Bedeutung**	**Inhalt**
0x00	Kommando	0x02
0x01	Datenlänge	0x10
0x02	Adresse low	0xX0
0x03	Adresse high	
0x04	Adresse upper	
0x05	Datenbyte 1	
...	...	...
0x14	Datenbyte 16	

Der Bootloader sendet als Quittung nur das Kommando zurück.

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x02

7.2.4 ERASE_FLASH

Mit diesem Kommando kann Flash-Speicher gelöscht werden. Das Löschen erfolgt immer in 64-Byte großen Speicherblöcken. Es können mit einem Mal mehrere aufeinanderfolgende 64-Byte-Blöcke gelöscht werden.
Der erste gelöschte Block ist derjenige auf den die Adresse verweist. Die niederwertigen 6 Bit der Adresse werden dabei ignoriert.

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x03
0x01	Anzahl der Blöcke	
0x02	Adresse low	
0x03	Adresse high	
0x04	Adresse upper	

Der Bootloader sendet als Quittung nur das Kommando zurück.

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0x03

7.2.5 RESET

Mit diesem Kommando wird ein Reset des Steuer-PIC ausgelöst. Vorher meldet sich der Bootloader korrekt beim USB-Controller im PC ab.

PC -> Bootloader

Adresse	Bedeutung	Inhalt
0x00	Kommando	0xFF

Der Bootloader sendet KEINE Quittung an den PC zurück.

-

7.3 Ein Beispiel für die Nutzung des Bootloaders

Folgender Delphi-Code zeigt die Nutzung des Bootloaders. Sie benutzt die Prozedur **Sende_Empfange(NrS, NrE).**
Diese Routine sendet **NrS** Bytes aus dem Byte-Array **send_buf** zum USB-Device und empfängt anschließend **NrE** Bytes aus dem USB-Device in das Byte-Array **receive_buf**.
Beide Byte-Arrays sind je 64 Byte lang.

Der Bootloader wird mit READ_VERSION erkannt.
Anschließend wird der Programmspeicher des Steuer-PIC im Bereich vom 0x0800 bis 0x7FFF mit neuem Code beschrieben. Der neue Code stammt aus dem Array **Hexin.Flash**.
Danach wird der neu beschriebene Bereich des Steuer-PIC wieder ausgelesen, und mit **Hexin.Flash** verglichen, um eventuelle Fehler zu erkennen.
War alles fehlerfrei, dann wird die EEPROM-Zelle 0xFE des Steuer-PIC mit 0 beschrieben, und der Steuer-PIC neu „gebootet“.

```
  // ist denn da ein Bootloader ?
  send_buf[0]:=READ_VERSION;    //0
  Sende_Empfange(1, 4);
  if (receive_buf[0] <>  READ_VERSION) or (receive_buf[3]<>1)
then exit;

  //brennen
  Adresse    := $800;
  Endadresse :=$7FFF;
  while Adresse<Endadresse do begin
    //64 byte löschen
    send_buf[0]:= ERASE_FLASH;
    send_buf[1]:= 1;                              // 1 x 64 byte
    send_buf[2]:=  Adresse and $0000FF;           // low
    send_buf[3]:= (Adresse and $00FF00) shr  8;   // high
    send_buf[4]:= (Adresse and $FF0000) shr 16;   // upper
    Sende_Empfange(5,1);
    //4 x 16 byte schreiben
    for k:=0 to 3 do begin
      send_buf[0]:= WRITE_FLASH;
      send_buf[1]:= 16;                             // länge 16
Byte
      send_buf[2]:=  Adresse and $0000FF;           // low
      send_buf[3]:= (Adresse and $00FF00) shr  8;   // high
      send_buf[4]:= (Adresse and $FF0000) shr 16;   // upper
      for L:=0 to 15 do  send_buf[5+L]:=Hexin.Flash[Adresse+L]
and $FF;
      Sende_Empfange(21,1);
      Adresse:=Adresse+16
    end;
  end;
```


-

```
      //prüfen
      Fehler:=0;
      Adresse    := $800;
      Endadresse :=$7FFF;
      while Adresse<Endadresse do begin
        send_buf[0]:= READ_FLASH;
        send_buf[1]:= 16;                              // länge
        send_buf[2]:=  Adresse and $0000FF;            // low
        send_buf[3]:= (Adresse and $00FF00) shr  8;    // high
        send_buf[4]:= (Adresse and $FF0000) shr 16;    // upper
        Sende_Empfange(5,send_buf[1]+5);

        for k:=0 to receive_buf[1]-1 do begin
          if (receive_buf[k+5] and $FF) <> (Hexin.Flash[Adresse+k]
    and $FF)
          then begin
            inc(Fehler);
          end;
        end;
        Adresse:=Adresse+16
      end;

      if Fehler=0 then begin
        // EEPROM-Zellen 0xFE und 0xFF mit 0 beschreiben
        send_buf[0]:= WRITE_EEDATA;
        send_buf[1]:= 2;            // länge 1 Byte
        send_buf[2]:= $FE;          // low
        send_buf[3]:= 0;            // high
        send_buf[4]:= 0;            // upper
        send_buf[5]:= 0;
        send_buf[6]:= 0;
        Sende_Empfange(7,1);

        // neu booten
        send_buf[0]:= RESET;
        Sende_Empfange(1,0);

      end else Memo.lines.add('Flash-Error');
```


-

8 EEPROM des Steuer-PIC

Im EEPROM des Steuer-PIC wird in der Zelle 0xFE das Bootloaderkennzeichen gespeichert.

von	bis	Zahl der Bytes	Datentyp	Wert	Standardwerer
0xFE	0xFE	1	Byte	Bootloaderkennzeichen	0

-

9 USBoot

Unter Windows kann man die Software USBoot benutzen, um eine Firmware unkompliziert in den Flash-Speicher des Pic zu laden.

USBoot beschreibt ausschließlich den Flash-Speicher ab der Adresse 0x800. Dadurch kann USBoot den Bootloader nicht beschädigen.

-