ebStartUp

Benutzerhandbuch

Version 1.2

Peer Georgi

5. Mai 2005

Inhaltsverzeichnis

Kapitel 1

Allgemeines

Das Programm „*ebStartUp*" erlaubt den Transfer von Daten, welche in Form von Dateien vorliegen, zu einem eingebetteten Zielsystem. Als Kommunikationsschnittstelle kommt USB zum Einsatz. Die Art der unterstützten Protokolle orientiert sich dabei an dem, was zum Zeitpunkt des Systemstarts von einem Zielsystem angeboten wird. Ausgangspunkt ist, dass zu diesem Zeitpunkt noch kein Betriebssystem auf dem Zielsystem installiert ist. Derzeit werden folgende Protokolle unterstützt:

- USB-DFU (**D**evice **F**irmware **U**pgrade)
- USB-LDBA (**L**oa**d** **B**ig **A**pplications)

Das USB-DFU Protokoll ist in der USB-Spezifikation beschrieben und kommt bei vielen eingebetteten Computersystemen zum Einsatz. Das USB-LDBA-Protokoll wurde entwickelt, um grosse Datenmengen, an verschiedene Bereiche des Zielsystem-Arbeitsspeichers, zu übertragen.

„*ebStartUp*" kann mehrere Übertragungen durchführen, deren Protokolle unterschiedlich sind. Aufgrund dieser Eigenschaft lassen sich komplexe Betriebssystem-Starts bequem vom PC aus durchführen.

1.1 Funktion und Einsatz

Damit „*ebStartUp*" in der Lage ist Datenübertragungen zum Zielsystem durchzuführen, bedarf es entweder expliziter Unterstützung durch die verwendete MCU des Zielsystems oder es werden Programme auf der MCU ausgeführt, welche den Kommunikationsweg zur Verfügung stellen. Im Falle eines ARM9-basierenden Zielsystems, wird das USB-DFU-Protokoll bereits von der MCU-Firmware unterstützt, sodass dieser Übertragungsweg unmittelbar nach dem Systemreset zur Verfügung steht. Das LDBA-Protokoll muss hingegen mit einem Programm zur Verfügung gestellt werden, dass vor dessen Verwendung auf der MCU zur Ausführung gebracht werden muss. Die folgende Abbildung visualisiert eine mögliche Vorgehensweise bei Verwendung des AT91RM9200.

Die folgende Abbildung zeigt einen beispielhaften Ablauf.

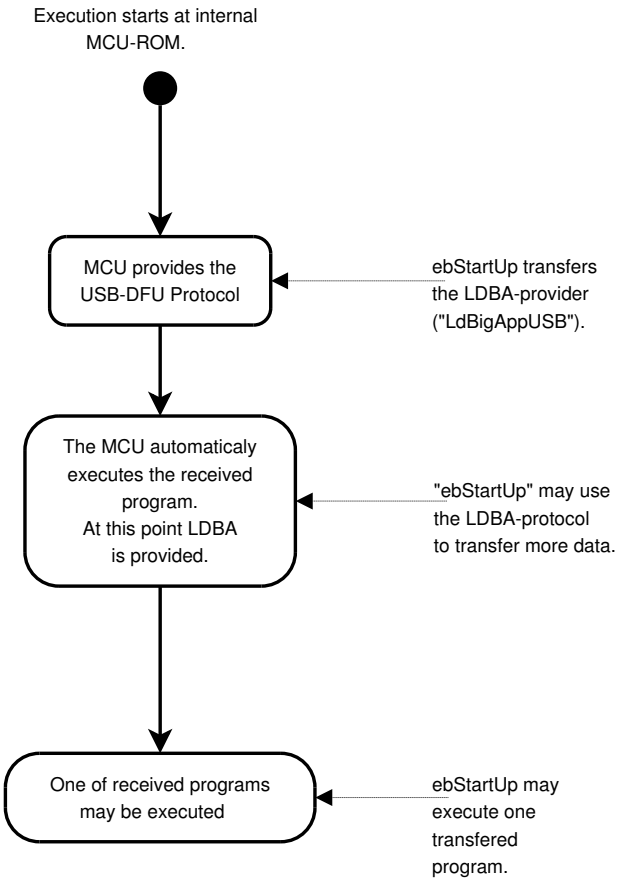

Abbildung 1.1: Möglicher Systemstart mit „ebStartUp".

1.2 Verwendung

Das Program „ebStartUp" liegt als Kommandozeilenprogramm vor, das durch eine Konfigurationsdatei konfiguriert wird. Es ist durchaus möglich und angedacht, dass „ebStartUp" innerhalb einer Entwicklungsumgebung automatisch gestartet wird. Weiterhin wird das Verhalten des Programms durch die Angabe von Kommandozeilenparametern beeinflusst.

1.2.1 Kommandozeilenparameter

Kommandozeilenparameter werden dem Programm bei der Ausführung übergeben. Einige Kommandozeilenparameter sind optional und müssen nicht unbedingt angegeben werden. In diesem Fall werden Voreinstellungen verwendet, die jedoch bei expliziter Angabe eines entsprechenden Kommandozeilenparameters überschrieben werden können. Alle Kommandozeilenparameter haben eine Kurz- und eine Langform. Die Kurzform (z.B. „-p"), spart Schreibarbeit, während die Langform („–project"), einen gewissen Dokumentationswert hat.

1.2.1.1 Voreinstellungen

Name der Konfigurationsdatei
Nach dem Start des Programms wird im aktuellen Verzeichnis nach einer Konfigurationsdatei gesucht, welche den Namen „**setup.ebs**" trägt. Diese Voreinstellung kann mit Hilfe der Kommandozeilenoption „-p Dateiname" oder „–project Dateiname" geändert werden.

Ausgaben
Per Voreinstellungen werden nur Fehler, Warnungen und wichtige Informationen auf der Standartausgabe ausgegeben. Dies kann mit den Parametern „-q" bzw. „–quiet" unterbunden werden. In diesem Fall werden nur noch Fehler ausgegeben.
Der Parameter „–verbose" hingegen sorgt für die Ausgabe aller möglichen Informationen auf der Standartausgabe.

1.2.1.2 Hilfe

Der Parameter „–help" gibt eine Liste möglicher Kommandozeilenparameter aus. Dabei wird jeder Eintrag kurz beschrieben. Im Zweifelsfalle sei auf diese Möglichkeit verwiesen, da sie unter Umständen aktueller ist, als dieses Handbuch.

1.3 Unterstützte Zielsystem-Platformen

Das Programm wurde ursprünglich für die Datenaufbereitung und Konfiguration eine ARM9-basierenden, eingebetteten Computersystems entwickelt. Es ist jedoch für alle eingebetteten Computersystem geeignet, die einen entsprechenden Boot-Loader und Übertragungsprogramme zur Verfügung haben.

1.4 Unterstützte Platformen

- Linux unter Verwendung der Kernelversion 2.6.x.

Derzeit wird nur Linux unterstützt.

Kapitel 2

Die Konfigurationsdatei

Die Projektdatei ist eine Textdatei, welche mit einem gewöhnlichen Texteditor editiert werden kann. Dabei werden die derzeit üblichen Zeilenendungen der Betriebssysteme „MacOS", „Windows" und Unix unterstützt. Sie ist streng hirarisch aufgebaut und verwendet Schlüsselworte und Blöcke zur Strukturierung.

2.1 Syntax

Die Syntax erinnert an C bzw. C++. Das Ende einer Zuweisung wird durch ein Semikolon besiegelt. Kommentare sind mit „// Kommentar" möglich und gelten bis zum Zeilenende. Ein Block wird mit „{" geöffnet und mit „}" geschlossen.
Gross und Kleinschreibung wird **unterschieden**!

2.1.1 Einlesen der Konfigurationsdatei

Die Konfigurationsdatei wird ähnlich, wie ein Skript eingelesen und die Reihenfolge der Anweisungsblöcke wird beachtet. Sie besteht aus einem oder mehreren Anweisungsblöcken. Die Reihenfolge der Konfigurationsvariablen, innerhalb eines Anweisungsblocks, ist hingegen nicht relevant.

2.1.2 Die Anweisungsblöcke

Jeder Anweisungsblock steht für eine oder mehrere Übertragungen. Dabei enthält jeder Anweisungsblock Informationen über den Übertragungsweg und das verwendete Übertragungsprotokoll. Je nach verwendetem Protokoll müssen Protokollparameter spezifiziert werden. Dies geschieht über Zuweisungen von Konfigurationsvariablen. Je nach Protokoll kann ein Anweisungsblock mehrere Übertragungen aufnehmen. Dies ist derzeit nur bei dem LDBA-Protokoll der Fall. Es ist jedoch genauso möglich mehrere Anweisungsblöcke zu spezifizieren.

2.1.2.1 Der USB-DFU Anweisungsblock

```
setup "USB-DFU"
  {
    image
```

```
        {
          Filename = "LdBigAppUSB.bin";
        }
    }
```

Es handelt sich um einen Anweisungsblock, der das Programm „*LdBigAppUSB.bin*" unter Verwendung des USB-DFU-Protokolls zum Zielsystem überträgt. Das DFU-Protokoll hat keine zusätzlichen Parameter. Das angegebene Programm wird unmittelbar nach der Übertragung ausgeführt. Der Übertragungsschritt gilt als beendet, wenn sich das USB-DFU Gerät abgemeldet hat.

2.1.2.2 Der USB-LDBA Anweisungsblock

```
    setup "USB-LDBA"
      {
        image
          {
            LoadAddress = 0x20007fc0;
            Filename =  "uImage";
            Execute = no;
            Comment = "Uebertrage den Linux Kernel\\n";
          }
        image
          {
             // Weitere Übertragungsschritte können folgen...
          }
    }
```

Dies zeigt einen USB-LDBA Anweisungsblock. Jeder Übertragungsschritt betrifft eine Datei und wird mit dem Schlüsselwort „Image" eingeleitet. Für jeden Übertragungsschritt muss die Zieladresse im Arbeitsspeicher des Zielsystems angegeben werden. Diese wird mit der Konfigurationsvariable „*LoadAddress*" festgelegt. Die Angabe der Adresse muss hexadezimal mit führendem „0x" erfolgen. Weiterhin muss der Dateiname angegeben werden, der festlegt, welche Datei zum Zielsystem übertragen wird. Die Variable „*Execute*" trägt als Parameter „yes" oder „no". Wird „yes" angegeben wird das Programm unmittelbar nach des Übertragung ausgeführt.
„Comment" gibt eine Nachricht auf der Standartausgabe des Zielsystems aus. Diese Angabe ist sinnvoll aber nicht notwendig. Sollte sie nicht angegeben werden, so wird stattdessen der Dateiname ausgegeben, damit der Ladevorgang beobachtet werden kann.

Es können mehrere Übertragungsschritte spezifiziert werden. In diesem Fall muss beachtet werden, dass nur eines davon ausgeführt werden sollte.

Kapitel 3

Beispiele

Die Beispiele sollen als Anhaltspunkte gelten und können als Ausgangspunkt für eigene Konfigurationsdateien dienen.

3.1 Start eines entwickelten Programms

Das Zielsystem soll das MCU-Modul der Firma Conitec Datensysteme GmbH, sein. Es handelt sich um ARM9-basierendes, eingebettetes Computersystem, welches den AT91RM9200 Mikrokontroller einsetzt. Das Computersystem verfügt über 128MByte Arbeitsspeicher und einen 8MByte grossen Festwertspeicher.

Mit Hilfe von ebStartUp soll das Programm, ohne vorherige Einrichtung des Zielsystems, gestartet werden. Es wird davon ausgegangen, dass im Festwerspeicher kein gültiges Programm vorliegt, sodass die MCU auf seriellem Wege versucht ein Programm zu empfangen. Für die genauen technischen Hintergründe sei auf das Datenblatt der MCU verwiesen.

Es liegen folgende Programme vor:

- LdBigAppUSB
  Das Übertragungsprogramm, welches das LDBA-Protokoll auf der MCU zur Verfügung stellt.
- Das Testprogramm, dass bei dem Systemstart ausgeführt werden soll
  Nennen wir es „**pong.bin**".

3.1.1 Vorbereitungen

Zunächst müssen Informationen zusammengetragen werden, an welcher Startadresse das Programm im Arbeitsspeicher zur Ausführung kommen soll. Diese Adresse wird beim Linkvorgang des Programms angegeben. Eine Ausnahme ist der Fall, dass das Programm keine absoluten Sprungbefehle verwendet und nur relative Speicherzugriffe macht. Dies kann bei einigen Compilern erzwungen werden. Im Falle, dass das Programm in Assembler entwickelt wurde, hat der Entwickler natürlich direkten Einfluss auf dieses Merkmal.

Im Folgenden wird davon ausgegangen, dass das Programm an der Startadresse **0x20000000** starten soll.

3.1.2 Erstellen der Projektdatei

```
setup "USB-DFU"
  {
    image
      {
       // damit wird das LDBA-Protokoll zur
       // Verfügung gestellt.
       Filename = "LdBigAppUSB.bin";
      }
  }
setup "USB-LDBA"
  {
    image
      {
        LoadAddress = 0x20000000;
        Filename =  "pong.bin";
        Execute = yes;
        Comment = "Uebertrage das Testprogramm...\\n";
      }
  }
```

Diese Projektdatei befindet sich mit dem Namen „**start-pong.ebs**" im Verzeichnis examples.

3.1.3 Starten der Übertragung

Das Programm „ebStartUp" muss mit Root-Rechten ausgeführt werden, da es direkt auf das USB-Dateisystem zugreift.

```
ebStartUp -p start-pong.ebs
```

Es gibt zwei Möglichkeiten die Übertragung zu starten. Zum Einen wird die Übertragung bei Herstellung der USB-Verbindung gestartet[1]. Sollte die Verbindung bereits bestehen, ist das Zielsystem manuell zu resetten.Die andere Möglichkeit ist, das Zielsystem manuell zu resetten.
Nach Ablauf der Übertragung sollte das Programm auf dem Zielsystem gestartet sein.

[1] Dies ist bei MCUs der Fall, die USB-DFU per Firmware unterstützen. Bei dem AT91RM9200 ist das der Fall.

3.2 Start des Programms „u-boot”

„u-boot” ist ein Linux-Ladeprogramm, welches eine Vielzahl unterstützter Platformen und Zielsysteme unterstützt. Es handelt sich um ein freies Programm, das in der Lage ist den Linuxkern zu starten und Übertragungen per Ethernet durchzuführen.

```
setup "USB-DFU"
  {
    image
      {
        Filename = "LdBigAppUSB";
      }
  }
setup "USB-LDBA"
  {
    image
      {
        Filename = "u-boot.bin";
        LoadAddress = 0x21f00000;
        Execute = yes;
      }
  }
```

Es muss beachtet werden, dass das Program „*u-boot*” richtig konfiguriert und an das Zielsystem angepasst ist. Die Zieladresse von u-boot ist die, welche beim Linkvorgang angegeben wurde. Im Falle von u-boot kann diese Adresse aus der Datei „***board/at91rm9200dk*/config.mk**” aus dem u-boot Quellverzeichnis ermittelt werden.

ebStartUp -p start-uboot.ebs

Nach dem Start der Übertragung erscheint auf der Standartausgabe des Zielsystems (RS232-Terminal) folgende Ausgabe:

```
USB Big Application Loader v0.9a (c) 2004 Conitec Datensystem GmbH.
Setting up USB...
UDP-Warning: - write timeout stage 1
loading LdLinux++ to address 0x20780000...
  ********** - ok
loading u-boot.bin to address 0x21f00000...
  ************************ - ok
Shutting down...
Executing application at 0x21f00000
U-Boot 1.0.0 (Apr 25 2005 - 17:57:17)
U-Boot code: 21F00000 -> 21F14CE0  BSS: -> 21F186B8
DRAM Configuration:
Bank #0: 20000000 32 MB
Atmel: AT49BV1614 (16Mbit)
Flash:  2 MB
DataFlash:AT45DB642
Nb pages:   8192
Page Size:   1056
Size= 8650752 bytes
Logical address: 0xC0000000
Area 0: C0000000 to C0007FFF (RO)
Area 1: C0008000 to C001FFFF (RO)
Area 2: C0020000 to C0027FFF
Area 3: C0028000 to C083FFFF
In:    serial
Out:   serial
Err:   serial
Uboot>
```

Die Ausgaben im oberen Teil wurden von dem Übertragungsprogramm der MCU ausgegeben.

3.3 Start von Linux

Es ist mit „ebStartUp" möglich, den vollständigen Linux-Systemstart zu initiieren und dabei auf das Programm „u-boot" zu verzichten. Dies ist dann von Vorteil, wenn der Systemstart noch andere Aufgaben wahrnehmen soll, die u-boot nicht leistet oder nicht leisten soll. Dabei muss ein Programm erstellt werden, dass u-boot ersetzt und den Linux-Kernel startet. Diese Aufgabe kann, für den AT91RM9200, beispielweise von LdLinux++ übernommen werden. Die genauen Zusammenhänge sollen hier nicht erläutert werden.

Es sei in diesem Zusammenhang ein etwas komplexeres ebStartUp-Skript vorgestellt, dass das Ziel erfüllt den Linux-Kern auf einem AT91RM9200 basierenden Computersystem, zu starten.

```
setup "USB-DFU"
  {
    image
      {
        Filename = "LdBigAppUSB.bin";
      }
  }
setup "USB-LDBA"
  {
    image
      {
        LoadAddress = 0x20007fc0;
        Filename =  "uImage";
        Execute = no;
        Comment = "Uebertrage den Linux Kernel\\n";
      }
    image
      {
        LoadAddress = 0x20000100;
        Filename =  "cmd.line";
        Execute = no;
        Comment = "Die Linux-Kommandozeile...\\n\\n  ";
      }

    image
      {
        LoadAddress = 0x20780000;
        Filename = "LdLinux++";
        Execute = yes;
        Comment = "Dies ist der kleine Linux-Loader\\n  ";
      }
  }
```

Kapitel 4

Das LDBA-Protokoll

Das LDBA Protokoll wird in einem separaten Dokument abgehandelt. Die Dokumentation ist derzeit in Arbeit.