

Update im Feld

Einfach mit USB-Bootloader

In der heutigen Zeit wird der Ruf nach einfach zu aktualisierenden Applikationen im Feld immer lauter. Der folgende Artikel beschreibt, wie man einen einfachen Bootloader mit einem USB-Host-Treiber, einem FAT16/32-Filesystem und einem Mass-Storage-Class-Treiber aufsetzen kann. Alle drei Komponenten können im Web frei heruntergeladen und auf einer 16FX-Mikrocontroller MB96F338USA von Fujitsu implementiert werden.

Schaut man in diesen Tagen auf die Anwenderfreundlichkeit von Applikationen im Feld, so kann davon ausgegangen werden, dass Updates mittels eines einfachen, handelsüblichen USB-Sticks die wohl komfortabelste Methode darstellen, da kein Laptop oder ein spezieller Programmer gebraucht wird. Voraussetzung hierfür ist natürlich, dass die verwendete MCU systembedingt einen USB-1.1-Host-Anschluß haben muß. Fujitsu bietet hierzu die 16FX-Prozessoren MB96F33xU inklusive Applikationsbeispiel sowie 16LX MB90F33x oder die FR-MCUs MB91F66x. Was der Benutzer darüber hinaus noch braucht, ist ein PC, die Fujitsu-Entwicklungsumgebung Softune, den Fujitsu-Flash-Programmer und einen USB-Stick.

Die USB-Host- bzw. Function-Bibliothek hat Fujitsu von der Firma Thesycon entwickeln lassen, damit der Anwender auf einfache Weise Zugriff auf die USB-API hat. Diese Bibliotheken beinhalten alle in der USB-Spezifikation aufgeführten API-Funktionen, die jedoch zunächst nur für die Anwendung des Bootloaders gebraucht werden.

Mittlerweile hat Fujitsu beide Bibliotheken mit dem Versuchsboard SK-16FX-144PMC-USB beim „USB Implementers Forum“ erfolgreich zertifizieren lassen. Bei der Host-Zertifizierung kam hierbei die USB-Bibliothek selbst, nebst der Mass-Storage-Class, dem FAT-Filesystem und einer kleinen Haupt-Applikation zum Einsatz, die per USART ein Mini-Menü aus-

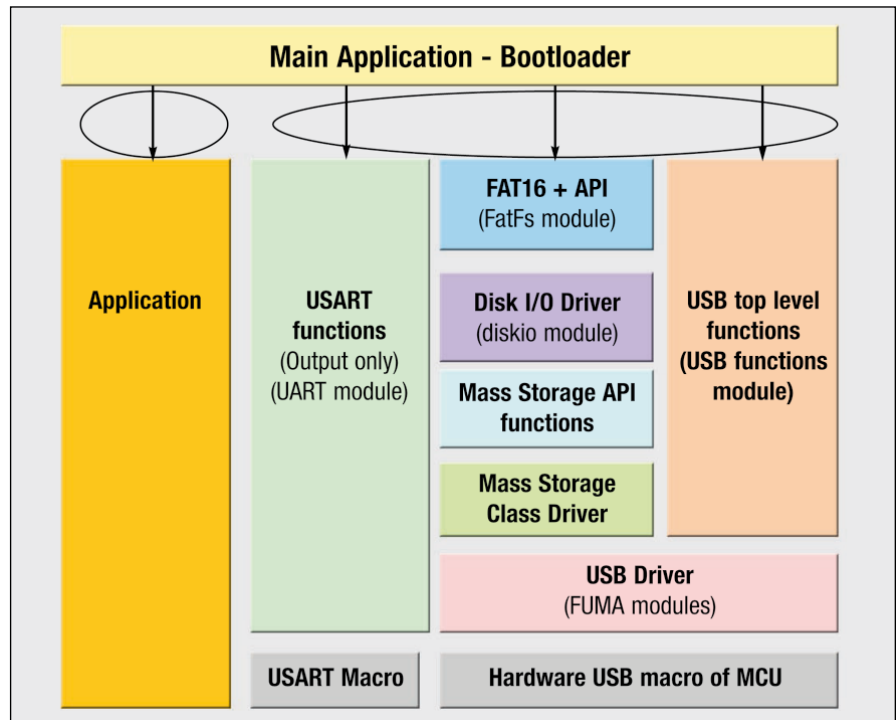


Bild 1: Verwendete Software-Komponenten des USB-Bootloaders.

(Alle Bilder: Fujitsu)

gibt, um auf einen angeschlossenen USB-Stick zuzugreifen.

Funktionskonzept

Bei dem hier vorgestellten Bootloader-Konzept muß ein anderer Weg eingeschlagen werden. Kennzeichnend für einen Bootloader ist zunächst die Auswahl, ob die eigentliche Applikation beziehungsweise der Bootloader selbst gestartet werden soll. Dies geschieht üblicherweise direkt nach einem Power-on Initialisierungszyklus, was bei Bedarf vom Anwender natürlich leicht geändert werden kann. Hierbei schaut der Bootloader, ob in einer gewissen Zeit eine erfolgreiche Enumeration erfolgt, was einem angeschlossenen USB-Stick entspricht. Anschließend wird nach einer bestimmten Datei im Stammverzeichnis gesucht, die bei Vorhandensein dann in den Flash-Speicher der MCU

geladen wird. Der hierbei anfallende zeitintensive Sector-Erase verhindert zunächst die USB-Konformität. Der Anwender kann jedoch nach jedem Flash-Erase die Enumeration neu einleiten, so dass man stets USB-konform bleibt. Die Praxis hat jedoch gezeigt, daß es den verwendeten USB-Sticks und anderen Mass-Storage-Devices nichts ausmacht, wenn für eine gewisse Zeit keine USB-Kommunikation vorliegt.

Nach dem Flash-Vorgang wird die neue Applikation erstmalig gestartet. Wenn keine Applikations-Datei beziehungsweise kein USB-Stick angeschlossen ist, startet die alte Applikation innerhalb weniger Millisekunden (siehe Abb. 2).

Verwendete Software-Komponenten

Bild 1 zeigt die Software-Komponenten. Der verwendete USB-Host-Treiber mitsamt

AUTOR



Marc Willam, Applications Engineer bei Fujitsu Microelectronics Europe.



Manuel Schreiner, Student und USB Bootloader „Spezialist“, praktiziert bei Fujitsu.

der Bibliothek wird kostenfrei von der Firma Thesycon angeboten, wobei hierbei das FUMA-Paket zum Einsatz kommt. Die verwendeten Bibliotheken liegen dort als C-Quellcode vor, können aber auch wie alle anderen Komponenten als C-Library kompiliert werden.

Die Mass Storage Class wurde von Fujitsu Microelectronics Europe selbst entwickelt. Sie beinhaltet die gängigsten SCSI-Befehle, die für den Zugriff auf MSC-Devices notwendig sind. Im Bootloader-Konzept kommt jedoch eine abgespeckte MSC zum Einsatz, da nicht alle Befehle verwendet werden müssen. Die MSC wandelt die SCSI-Befehle in USB-konforme Wrapper-Befehle um und sendet diese dann an das angeschlossene Mass-Storage-Device.

Das FAT16/32-Filesystem wird von einem privaten Programmierer im Internet angeboten. Fujitsu hat dieses Filesystem bereits seit mehr als einem Jahr erfolgreich im Einsatz. Wer ein zertifiziertes Filesystem benötigt, der sei auf die gängigen Software-Hersteller verwiesen. Das verwendete Filesystem selbst besteht aus zwei Unterkomponenten: Das Disk-I/O-System und die FAT16/32-Komponente selbst und arbeitet Hand-in-Hand mit dem MSC von Fujitsu. Für den Bootloader wurde das Filesystem auf die wesentlichen Bestandteile abgespeckt: Funktionen wie Daten schreiben, Verzeichnisse erstellen, ein Block Device zu formatieren oder genauere Informationen aus dem Filesystem lesen wurden herausgelassen.

Eine weitere wichtige Komponente ist der Bootloader, der sich um die Koordination aller anderen Software-Bestandteile kümmert. Der Bootloader kommt immer nach einem Reset zum Einsatz, wobei jener dann sofort im Reset-Cause-Register nachprüft, ob gerade ein Power-Cycle stattgefunden hat. Im positiven Falle wird dann oben Beschriebenes ausgelöst.

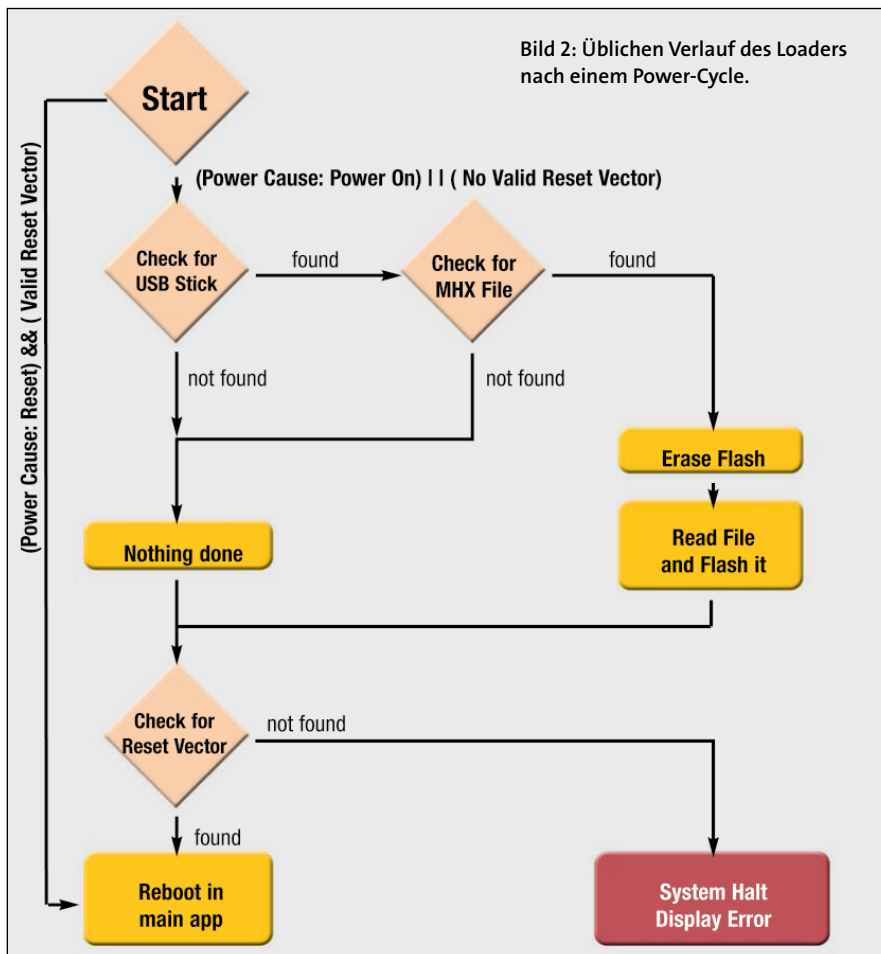
Der Bootloader

Der Bootloader selbst koordiniert alle Vorgänge nach einem Power-Cycle. Wenn die Datei der neuen Applikation gefunden wurde, prüft der Bootloader, ob es sich um eine valide Datei handelt. Seine Aufgabe ist es außerdem, darauf zu achten, daß keinerlei Adressen enthalten sind und die Adressen in den Sektoren, in denen der Bootloader steht, vorhanden sind.

Außerdem enthält der Bootloader einen Motorola-Hex-Format-Dekoder, so daß

übliche MHX-Dateien als neue Applikation verwendet werden können. Dabei ist es unerheblich, ob die einzelnen S-Record-Zeilen mit einer geraden oder einer ungeraden Adresse beginnen oder gerade oder ungerade Anzahl von Bytes besitzen, obwohl beim Programmieren immer ein ganzes 16-Bit-Wort geschrieben wird.

Nach dem Programmieren oder bei gleicher Applikation wird dann immer in den Reset-Vektor gesprungen. Zu beachten ist, daß immer ein kleiner RAM-Bereich ►



wegen des Bootloaders überschrieben wird, der dann der neuen Applikation nicht mehr zur Verfügung steht. Somit sind Datenvererbungen applikationsübergreifend nicht möglich.

Bild 2 zeigt den üblichen Verlauf des Bootloaders nach einem Power-Cycle. Zu erwähnen sei, daß die Implementierung der Sektion „System Halt/Display Error“ dem Benutzer selbst obliegt. Im Fujitsu-Beispiel wird dazu ein Error-Ton an einem Port ge-

most (CRC/EMPTY) beziehungsweise je-ner an einer Debug-USART ausgegeben.

Speicheraufteilung

Der Bootloader befindet sich in den Sektoren SA0 bis SA2, benutzt also 24K Bytes von 0xDF0000 bis 0xDF5FFF. Hierbei sind alle oben beschriebenen Komponenten enthalten. Alle anderen Sektoren S32 bis S39 (im Falle von MB96F338USA 0xF8000 bis 9xFFFFF) stehen der eigenen Applikation

uneingeschränkt zur Verfügung. Für eine genauere Beschreibung der Sektoren, ist das Datenblatt zur MCU zu konsultieren. Erwähnenswert sei noch, daß der Bootloader zwei entscheidende Sektionen hat, auf die er zugreifen kann: Die alternative Interrupt-Vector-Table für die USB-Host-Bibliothek und den RAM-Code, der beim Flash-Programmieren ins RAM heruntergeladen werden muß. Bei den „klassischen“ Sektorenanordnungen bei den FR- und 16LX-MCUs kann leicht eine ähnliche Stelle für den Bootloader gefunden werden.

Praxis und Ausblick

In der Praxis braucht der Systemingenieur nicht mehr genau zu verstehen, wie der Application-Update im Einzelnen funktioniert. Er muß sich nur darauf verlassen können, daß die Schritte, die nötig sind, um die Anwendung zu aktualisieren, einfach und fehler-tolerant sind. Das einzige, was er zu tun hat, ist einen USB-Stick oder ein anderes USB-MSC-Device mit sich zu führen, es am Gerät anzuschließen und es kurz aus- und dann wieder einzuschalten. Dies kann in der Regel auch durch einen Service-Techniker durchgeführt werden. In der derzeitigen Version kann der Bootloader nicht durch sich selbst aktualisiert werden. Dazu ist ein sogenannter Boot-Manager notwendig, der diese Aufgabe übernehmen kann. Hierfür gibt es bereits Lösungsansätze, die nach Anfrage bei Fujitsu Microelectronics Europe geliefert werden können. (sb)

	423ei0609
▶ Link zu Fujitsu Microelectronics	
www.elektronik-industrie.de	