USB-UHPI-Programmer

User Guide

v1.08 – 2007-08-06

Authors: Alexander Stohr

Siegbert Baude

This page intentionally left blank

Content

Figures

Tables

Changes

Pos.	Date	Description	Executor
v1.0	2006-06-09	Start of Document as V1.0 Draft	A. Stohr
v1.2a	2006-09-11	PCB, Installation, Tools, Samples, Info-Updates	A. Stohr
v1.3	2006-11-07	Release-version, spell-checked, added hint to USB hub problem, added contact information	S. Baude
V1.4	2006-12-06	Added hint for DIP-switch setting to UHPI mode	S. Baude
V1.5	2006-12-08	Improved screen shots and CLI description	S. Baude
V1.6	2007-02-28	Added errata for “OnBtnReadTek failed” error	S. Baude
V1.7	2007-06-22	EMV DIP switch hint, problem helper list, FBTC infos	A. Stohr
V1.8	2007-08-06	Updated screen shots and description for tool version 1.7	A. Stohr

Table 1 Changes

Used Documents and References

You will find the referenced documents in the “Documentation” folder on the CD.

Pos.	Description
1	TI sprs268d.pdf *TMS320C6727 [...] Floating-Point Digital Signal Processors* (Rev. D, February 2006)
2	TI spraa69c.pdf and sprc203.zip *Using the TMS320C672x Boot loader* (Rev. C, March 2006)
3	TI spru719.pdf *TMS320C672x DSP Universal Host Port Interface (UHPI)* (December 2005)
4	TI spra999a.pdf and spra999a.zip *Application Report: Creating a Second-Level Boot loader for FLASH Boot loading on TMS320C6000 Platform With Code Composer Studio* (Rev. A1, May 2006) **Attention:** This only covers C620x/670x and C621x/671x/64x devices, but not yet the C6727
5	TI spru186p.pdf *TMS320C6000 Assembly Language Tools v 6.0 Beta User’s Guide* (Rev. P, July 2005)
6	TI spru187n.pdf *TMS320C6000 Optimizing Compiler v 6.0 Beta* (Rev. N, July 2005)
7	http://focus.ti.com/docs/prod/folders/print/tms320c6727.html *TMS320C6727, Floating-Point Digital Signal Processor*
8	**DSP-Weuffen** EVM-board_Technical_Reference_TMS320C6727_V1_11.pdf *C6727 EVM, Technical Reference*

Table 2 Used documents and references

1 Safety Instructions

Keep the unit away from heat sources.
Do not use the unit near water.
Unplug the USB cable if the unit will not be used for a long period.
Remove power source when attaching or removing module!

Mains safety:

WARNING
The power supply unit of the C6727 EVM has no internal fuse. Do not open power supply unit.
Risk of Electric Shock. You can come in contact with hazardous voltage.

Caution

The EEPROM and the USB controller units are sensitive to electro static discharges. Use proper precautions against static electricity damage at all times.
The device is intended for evaluation and experimental purposes in a microelectronics lab and similar environments, so it comes without any casing. For EMI/EMC sensitive environments, for environments with aggressive environmental conditions, for food or medical environments, the board possibly needs suitable covering, shielding, casing, encapsulation, or whatever applies.

2 Copyright

No claim to the receiver is made for the provided software examples concerning the patents situation, copyright issues or similar legal responsibilities.
The user is free to use and modify the code for his own purpose.

3 Contact

Address:	DSP-Weuffen GmbH Bernhard-Müller-Str. 11 D-88239 Wangen Germany
Phone:	+49-(0)7528-975531
Fax:	+49-(0)7528-975532

4 Introduction

The USB-UHPI-Programmer device is an add-on component for the “C6727 EVM” board made by DSP-Weuffen GmbH. It is a low-cost programmer device that enables the user to evaluate the platform and to develop applications for the TI C6727 DSP family. The hardware is bundled with a set of ready-to-run Win32 software for Windows 2000 and Windows XP.

4.1 Onboard Features

The provided hardware interface unit has the following features:

- Support for the TMS320C6727 EVM board through its UHPI connector (X4 and X5)
- +5V powered through the USB connection and/or the EVM board
- No buttons or configuration switches for interface, all functionality is controlled by the host. The attached EVM only needs a one time adjustment for the boot mode DIP switches.
- Comes bare without casing allowing the user to integrate this into his applications
- Boone led indicator for monitoring the power state
- Full UHPI control including all UHPI registers/latches and interrupt signaling
- Reset control wiring for the connected target
- Ready to run interface firmware in an EEPROM mounted in an 8-pin DIP socket

4.2 Software Features

The provided software package includes these components and features:

- User land DLL interface with support files and related documentation
- Kernel mode driver and an inf-file based installer
- Ready-to-run GUI application for program uploading and flash programming control
- Straight forward command line application for flexible integration into the user’s tool chain
- Control applications allowing to reset the EVM board through a hardware reset
- Allows the complete flashing operation to be done in only a few mouse clicks
- Allows the complete exchange of your application on any download
- Allows quick testing of your applications in RAM before flashing takes place
- Allows fast communication of your PC application with your running system on the DSP
- The system exposes flexibility, so the user can go far beyond the scope of the provided demos

4.3 Scope of Delivery

- USB-UHPI-Programmer
- CD with software examples, data sheets and documentation in PDF format
- Printed manual
- USB cable

4.4 Board images

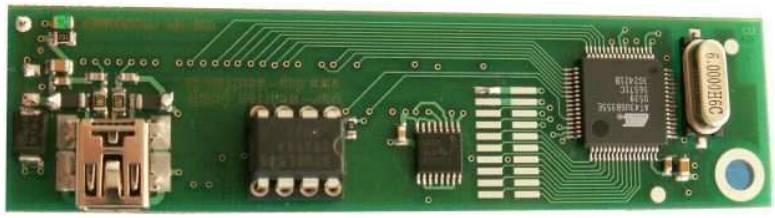

Figure 1 Interface top view

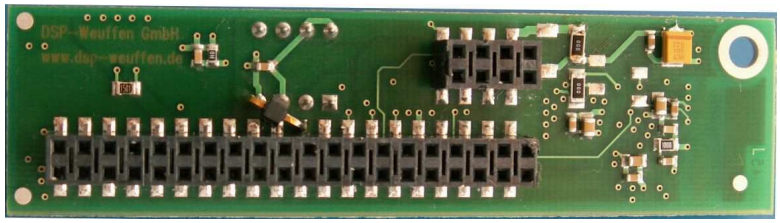

Figure 2 Interface bottom view

5 Overview

The programmer hardware is an electrical interface adapter that allows a standard PC to communicate with an EVM board through a USB port whilst the EVM board is accessed through the UHPI interface. In this setup, the PC operates as master to the interface and to the target. The exposed UHPI interface of the DSP is a shared memory interface by its nature and offers some extra features like configuration, interrupt signaling, or address auto increment. In advance of the UHPI functionality, the interface further provides reset control of the EVM.

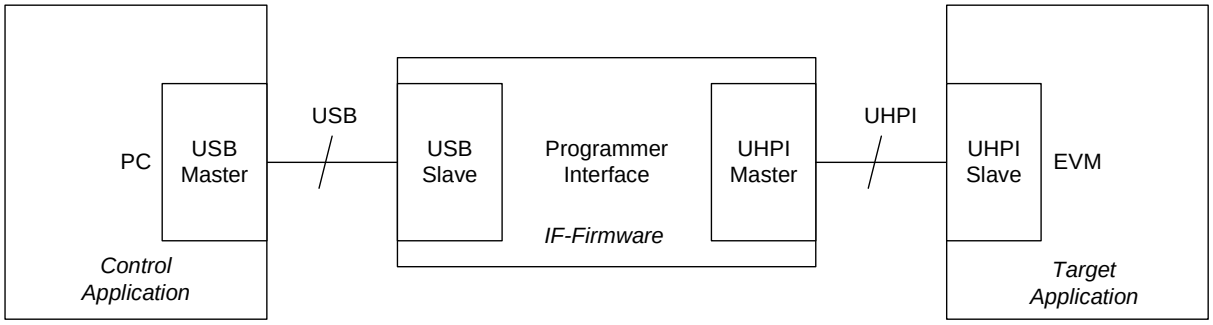

Figure 3 Concept of the interface interconnection

This system allows

- remote control,
- monitoring and diagnostics,
- memory access,
- uploading of data,
- remote flashing,
- buffered communications

and lots more.

The provided package comes with a basic set of sample applications whilst it is up to the user to explore the full range of possibilities for his individual applications.

6 Usage

6.1 Hardware Setup

The USB interface is uncritical. Any USB 2.0 compliant A(normal)-B(mini) cable and cable length (2 m) should work. As the the device was designed in order to comply with the USB “Full Speed” standard, it is expected to be physically and logically compatible with any other USB compliant master system.
The interface should get connected to the “C6727 EVM” whilst no EVM power, no USB (also providing power) and no other power source are connected. The UHPI connection should stay short to minimize any EMI and EMC effects. This also helps in maintaining good signal quality. In ideal case the interface board gets directly plugged on top of the EVM board. If in need, a few centimeters of ribbon cable in between the two devices should still work.

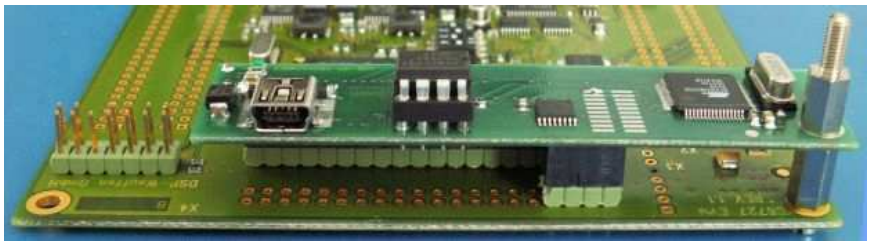

Figure 4 Module stacking with added bolts

Plug and unplug the two devices with only moderate force. While combining both units, make sure all pins of the EVM will slide into the correct holes of the two connectors on the interface. You might want to fix the boards together by some standard electronics screws or bolts. If ever in need for dismounting this again, grabbing the short sides and light alternating pulls are the recommended method for loosening the connection. Other methods might raise the danger of damaging both boards.
Make sure that the DIP switches on the “C6727 EVM” are set to “UHPI” boot mode, i.e. all switches set to “on”, see section 4.8 “Boot Mode” of “Technical reference C6727 EVM” located on the CD at: *\Documentation\DSP-Weuffen\C6727_EVM_Technical_Reference_v1.11.pdf*

6.2 Software Setup

The interface comes along with a set of software and documentation on CD. Copy that software to a local hard disk. If you received an update through e-mail or our website www.dsp-weuffen.de, then you might possibly need to extract the provided files first.

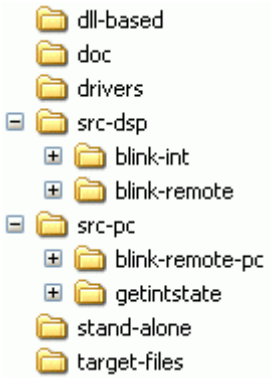

Figure 5 Software package directories

When you plug in the USB connector of the board, your operating system should notice the plugging and attempt to install drivers and applications. Any sort of automatic install (search the web, local hard disk, local driver DB) is likely to fail and should not be used. Instead of this, specify the driver location on your own and select the "*drivers*" directory from your previous installation.

If your system was not asking you for a driver, select the "device manager" application of your system. Right-click on the unrecognized or falsely (as USB-HID device) recognized device and then select the driver install/update menu option. Be aware that your system might need an individual driver install for any USB port that you want to use. I.e., if you plug the interface to a new port, make sure that the correct drivers get installed again.

6.3 Provided Software

The software stack consists of a set of monolithic ready-to-run RAM/flash programmer applications, a few DSP application binaries, a set of sample application pairs (Win32 and DSP) that are making use of the DLL in source and binary form, the interface DLL with needed support files and additional documentation. There are no Win32 workspaces, because they would depend on the used tools. Please build these files on your own.

When writing your own projects, you should tune your projects so that they will match the characteristics of the programmer interface. You should therefore build, link and post process your own DSP and Win32 applications like in the provided samples.

Aside to the provided software, you will find the “DLL programmer manual” that will help you in setting up your own PC based applications. If needed, additional information about the UHPI can be found in related TI data sheets and application notes.

6.4 Operating the Control Applications

6.4.1 GUI Control Application

Start the provided GUI control application "*stand-alone\uhpi-dlg.exe*". It should appear on your screen similar to this:

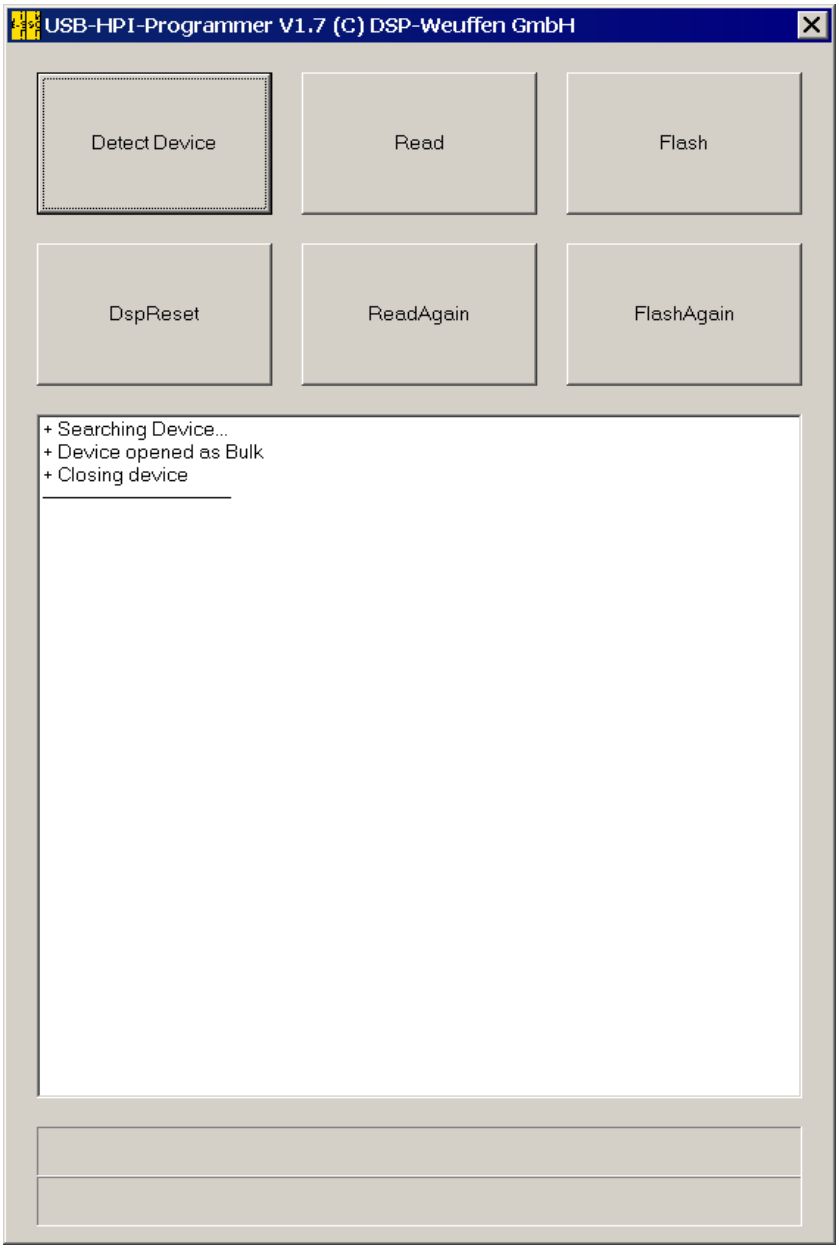

Figure 6 GUI control application

On top there is a set of control buttons, below you find an output window presenting some status information.
The exposed buttons operate in the following way:

- **Detect Device** repeats the initial programmer interface detection cycle. This button is for diagnostics and device error management, it is not needed for normal operation.
- **DspReset** performs a hardware reset of the attached DSP device on the EVM board. This button is useful if you want to stop your application remotely.
- **Read** opens a windows file dialog that asks for a file in Tektronix file format (**.tek;*.tex*) or a Code Composer generated Coff file (*.out) and then loads the contents with the control application. (For the Coff file a dialog pops up that will ask for symbolic and numeric details of the read operation, a *.tek/tex helper file will be written to your file system – please see below.) After that the file gets format checked, section analyzed and a connection gets established with the programmer interface and the target. If this is successful, the control application sends the image data to the EVM board using a two-stage process and verifies the written memory. The application will inform the user about the image contents, the progress and any error in the output window. This option is intended for single turn programming of the EVM board.

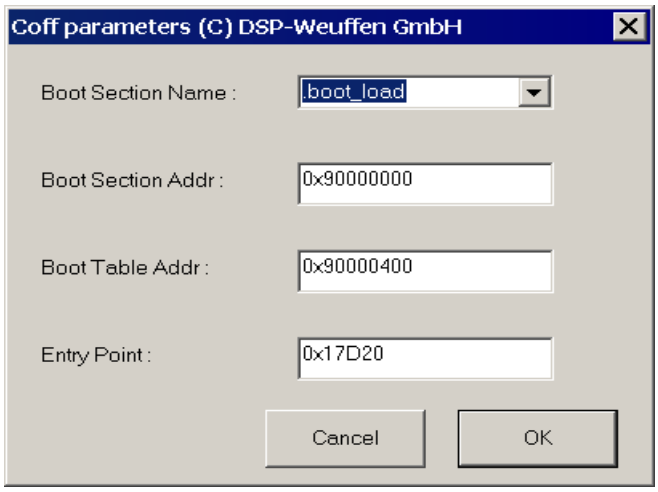

Figure 7 Coff parameters dialog

The Coff dialog offers several rather basic options that are comparable to the hex6x.exe from the standard TI tool chain - this is described a bit more in detail in a later chapter. The first section found in the just read file gets the default for the of the **Boot Section Name**. For this you should give it a sounding name in your project. The **Boot Section Address** refers to the FLASH ROM base address where your boot loader should be stored. The default is tuned to fit the C6727 EVM. With the **Boot Table Address** you will be defaulted to the regular offset for the C6727 EVM samples. The application **Entry Point** address got just read from the out file. Generally you should not need to change anything.

- **ReadAgain** will open the previously used file, if there was any, and then perform the very same process as for *Read* despite asking for a filename with a dialog. This function is intended for repeated programming of the same source file. The file is read again any time the button is pressed, so any change in the input file will immediately be promoted to the EVM board. Operation from slow media, e.g. a floppy disk, should be avoided.
- **Flash** will ask for the DSP based flash programming application in its first file dialog. Choose "*target-files\FBTC_C6727EVM.tex*" and click "OK". After that a second file dialog will appear and ask for the application that has to be sent to the flash. Choose e.g. "*target-files\Blink.tex*" or your own application. Both images need to be in Tektronix file format.
- **FlashAgain** will open the previously used set of files from the most recent *Flash* operation and will then perform downloading and programming as before. As for *ReadAgain* the file is read again each time the button is pressed.

Depending on the current operation mode, the application will show distinct colored progress bars. The first bar always indicates the progress in writing, the second will indicate the verification progress.

- **Green** means that application data gets written to internal and externa RAM areas.
- **Yellow** means that the application is writing flash data into a buffer area in external RAM.
- **Red** means the EVM is programming the on-board FLASH using the buffered data.

The erasing phase of the FLASH memory is performed as an atomic operation right before the flash write cycle. It therefore cannot be reflected by the progress bar indicator. This erasing will last for about 10 seconds whilst a short notice gets displayed at the message area of the flash application.

6.4.2 Command Line Control Application

There is also a command line application "*stand-alone\uhpi-cli.exe*" that might be helpful for automated flashing or for integration into your favorite development platform. If launched with the provided batch "*target-files\test-cli.bat*", it will produce the following output:

```
USB-HPI-Programmer CLI control application V1.7
=======================================
  (C) 2006 by DSP-Weuffen GmbH, Neuravensburg, GERMANY

Commandline:
  <this-program> <options> <parameters>
Options:
  -nologo      supresses version and copyright notice of this tool
  -program     loads the application image into the RAM of the EVM (default)
  -flash       loads the application and programs the flash_image to the FLASH
  -reset       performs a reset of the target device, this does not load anything
Options that will only apply to coff encoded files:
  -cs<name>    name of the section with the flash boot loader code (default: .boot_load)
  -cf<hexaddr> target base address for the flash boot loader       (default: 90000000)
  -ct<hexaddr> target base address for the flash boot loader table (default: 90000400)
The options program/flash/reset are exclusive to each other.
Parameters for programming:
  application  filename of user application to upload into RAM
    (the application can be either a Tektronix encoded flash image
    or a coff encoded TI *.out file, this will also build a *.tek file)
Parameters for flashing:
  application  filename of flash routines to upload into RAM
  flash_image  filename of user application to program into FLASH
    (the flash_image can be either a Tektronix encoded flash image
    or a coff encoded TI *.out file, this will also build a *.tek file)
Parameters for resetting:
```

Figure 8 Help text of the CLI application

The above figure shows the possible operations of the CLI application and their descriptions. All applicable functions of the GUI version are represented. There are no dialogs, so all filenames have to be entered on the command line at call time. An error result value is generated, depending on the result of the operation.

For example, you could use the following command line to flash the application "*blink.tex*":

```
D:\DSP-WEUFFEN>uhpi-cli.exe -flash FBTC_C6727EVM.tex blink.tex
```

6.5 Provided Samples

The device comes further with a set of samples. The sources are in the "*src-dsp*" and "*src-pc*" folders, the binaries in the "*dll-based*" folder. They are meant to be run from any computer that has the drivers installed and should demonstrate the more advanced functionality of the DLL and furthermore show how to build a working communication whilst an application got already loaded and is running on the target platform. As these are meant to be educational pieces of software they are provided as source and in binary form.

In the "*target-files*" subfolder you can find some more binaries:

- *Blink.tex* - is just a simple LED play animation that is not much interactive
- *CODEC-gstq.tex* - is an audio demonstration that reacts on key presses from the user
- *FBTC_C6727EVM.tex* - is the flash downloading target software, use it for flashing the EVM

The files "*Blink.tex*" and "*CODEC-gstq.tex*" do represent just some demo application that can be flashed to the target or alternatively started from RAM. They are provided in source form only with the C6727 EVM board.

The first true sample application is "*blink-int*" with its PC counterpart "*getintstate*". It shows how to trigger an interrupt from inside the DSP and detecting plus handshaking it on the PC. Every time the blink counter on the DSP crosses some limit, the interrupt line gets altered. The PC cyclically requests the interrupt state of the EVM on its own by sending a query to the interface. When an interrupt gets detected, the PC will confirm this signal by sending a handshake through the interface to the target EVM. For this sample to work, the target software should get uploaded manually to the target.

The second sample application is "*blink-remote*" with its PC counterpart "*blink-remote-pc*". It demonstrates how bi-directional data transmission between the two systems could be set up. As the basic element, a FIFO design in the form of software implemented ring buffers in the memory of the target system gets used. The concept only provides the simple case, where any data unit has the very same size and meaning for each of the data directions. It should not be that difficult to adapt this sample, so that e.g. command value based messaging could be done. In the current form, the DSP on the EVM cyclically reads in its button state and, if the button state has changed since the last time, sends the determined value through its data channel to the PC system, where the PC prints a hexadecimal value for any received data. On the other side, the PC is using a counter that is cyclically updated and sent through the data channel to the target system, where this value is used for updating the state of the LEDs on the EVM. As a demonstrational feature, the PC part does upload the DSP

target application on its own, waiting for the DSP program being started up and then resolves the location, where the communication buffers and control structures are located.

The binary "*FBTC_C6727EVM.tex*" is the FlashBoot target component. FlashBoot is a third add on for the Code Composer IDE and is capable of programming several sorts of non volatile memorys. For a specific target a so called FlashBoot target component is required. This is a program that runs on the destination system and that communicate with the controlling PC program. The provided binary does exactly fill in this role for the target of the C6727 EVM. It will cope as well with the UHPI PC software.

7 Technical Reference

7.1 Boot Loader Design

The software specific boot process of the DSP is three-staged. See also the figure below.

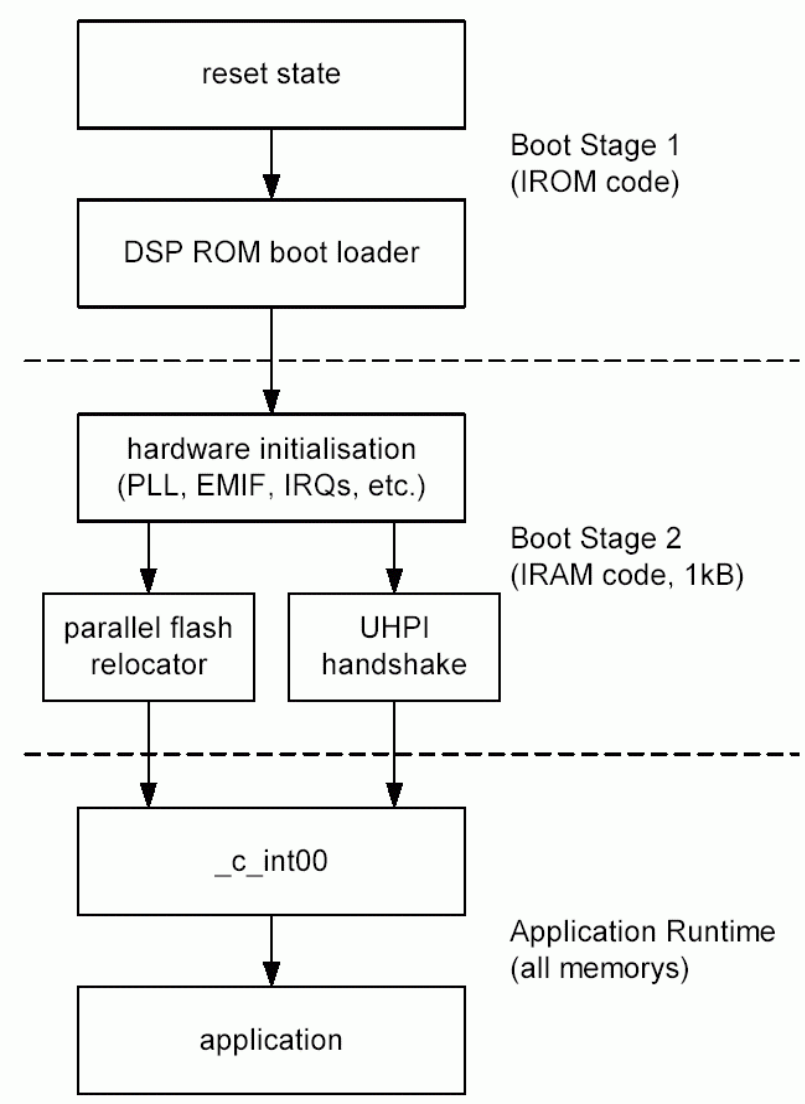

Figure 9 Boot loader software modules

The **first stage** is performed out of the ROM code of the DSP in combination with the register sampled configuration pins. It is out of reach of any form of user programming. At high order volumes only, the chip vendor might be able to allow custom mask programming for this area.

A user-defined module that has two major tasks makes up the **middle stage**:

- The first task is initializing the clock system (PLL) plus setting up the matchmaking parameters for the externally attached SDRAM. Prior to finishing this, no access to the SDRAM should happen, since neither operation of the device is guaranteed nor the data might be safe.
- The second task is to load the sections for the application program. If booting from a true parallel flash device, the second stage loader reads the flash, analyzes the stored section table and transfers the contained date to its destination “load” addresses. If booting through UHPI, the second stage loader signals to the UHPI master that hardware initialization is done and the sections transfer is allowed to start now. When done, the master signals completion of this task back to the second stage loader.

After finishing this, the second stage code will call the **application** entry point, that it got either from the table in the flash memory or was read from a special location in the IRAM to which it was written by the UHPI master program on the PC. The application code is the final stage. It fully belongs to the system developer so that it can freely run and perform miscellaneous duties on its own.

7.2 Tools and Boot Process

The *hex6x* command line utility from TI converts the executable file of your project into a flash image. For this, a few options have to be passed on, either on the command line or through some extra command file. In the next figure only the most important switches for properly controlling the creation process of such a flash image are shown. There are most likely several other switches to set up for a full build that will not be discussed here. Refer to the TI tools documentation for more details. You might want to start your works by using one of the provided sample scripts from the demonstration DSP applications.

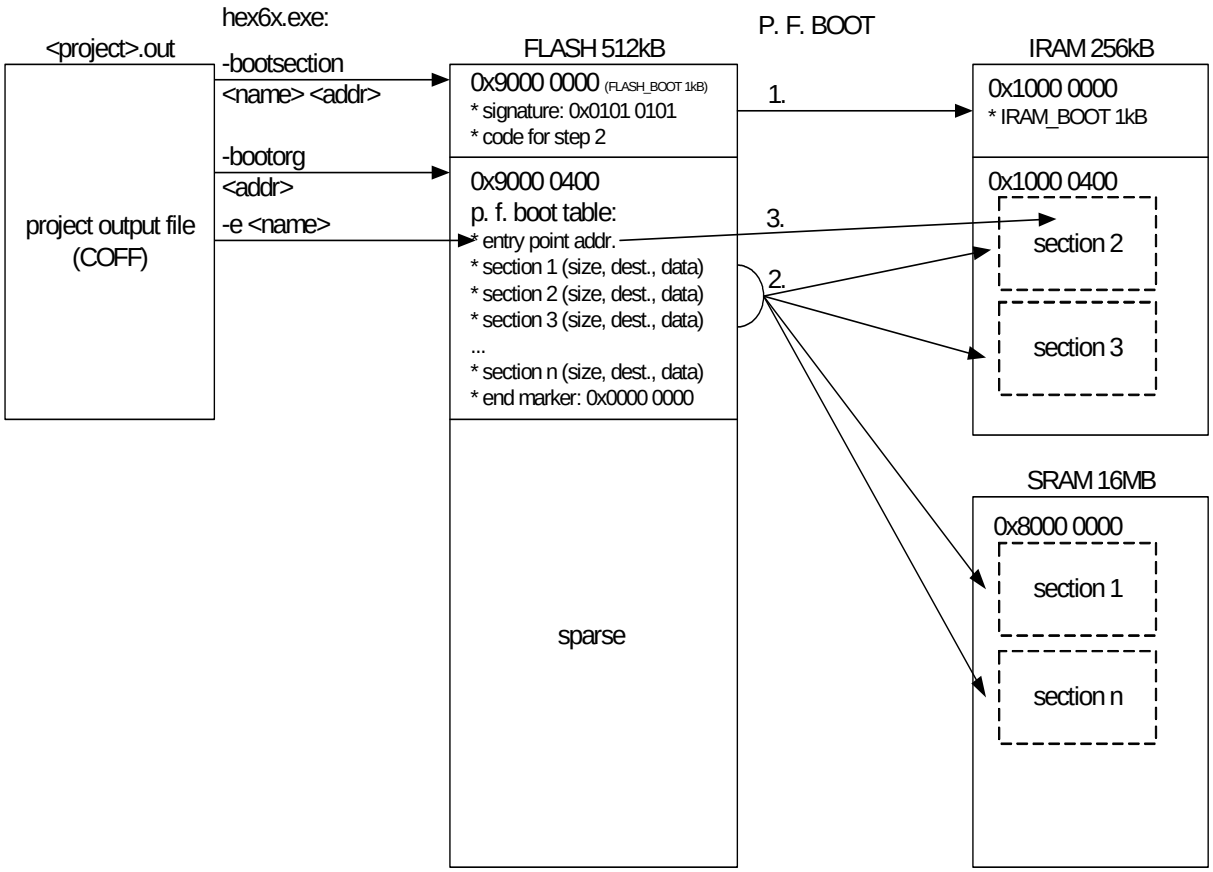

Figure 10 Flash image creation and boot process

With the flash image you have two options. You can load it directly into the RAM by means of the USB UHPI programmer interface. Or you can program this image into the flash device of the EVM by using the provided toolset and booting it later on directly from the parallel flash, even as a stand alone device. Regardless of which variant you choose, it is always a three-step loading process where in step 1 the second stage boot loader will get transferred to the start of the IRAM of the DSP.

In the second step the loaded code gets started and some init (PLL, EMIF) takes place. At the end of the second stage code, the application sections will get loaded in their destination RAM areas and the entry point address for the application will get determined. The application will finally get started by a simple branch command.

7.3 Pinout

The pinout of the mounted USB connector follows the rules of the USB standard. The pair of female grid connectors for the EVM do use the assignment as documented by the side-packaged schematics, part list and placement plan of the provided interface (available in PDF format). They are designed to be compatible with the C6727 EVM from DSP-Weuffen GmbH.

7.4 Limitations

The following limitations might affect your system in combination with UHPI programming:

- Communication between PC and the USB-UHPI programmer can fail, if the programmer is attached to a USB hub. It is recommended to attach the programmer directly to a PC USB port.
- UHPI must be enabled and configured to allow access through the interfaces. This happens either through application software on the EVM board or through the IROM code and special switch settings at EVM boot time. The hardware provides no way (except JTAG interaction) to access the EVM module when a program is already running and the program itself is not cooperative. (JTAG interaction would mean to significantly increase solution costs and would mean to use an alternate way.) Sometimes it could be beneficial to use both solutions in parallel, e.g. debugging code through JTAG whilst monitoring some data through UHPI.When the EVM is halted through a JTAG interface, then there might be no valid responses to any sort of UHPI request or to solely UHPI data exchange requests especially to reset requests.
- The maximum size for the second stage boot loader is limited to 1 kB. This limitation is caused by the IROM code in the DSP and is therefore not changeable. This will mainly affect programs that shall later run from FLASH, but for clarity this is also the case for UHPI downloads. Working around this limitation would probably require using a three-stage boot loader.
- The memory where the code for the second stage boot loader is running from cannot receive any flash image sections. Only after the application stage got reached this memory can be reused, e.g. by loading some program overlay into it or similar actions. If the boot loader is replaced later on by something else, a “warm” boot using this routine will not be possible any longer.
- As the pins driving the UHPI on the EVM are multiplexed with a few other purposes, it might be that not all functions are operable when the UHPI interface remains active in the application. Whilst the onboard flash is accessed, the data pins will become blocked and need release after that. Regardless of this, the interrupt signal will continue to operate in such a phase. After download, it depends partially on the stage 2 boot loader cleanup code to disable the UHPI.

7.5 Extensibility

The interface has an integrated 2-port USB hub, which is supported by its firmware. It is just not routed to anywhere on the PCB, so these ports are not usable by default. If the customer decides to modify the board in that way or any other, all warranty gets lost and there will be no right for refund or exchange of the product anymore. Parts that are sent back with modifications will, if ever, get only replaced with comparable devices that do not bear the modifications.

It should be possible to use alternative firmware versions for the interface by adding a small piggy back PCB that plugs into the firmware socket and has a switch on it for individually allowing the CS signal to select alternative firmware versions to load from devices that are plugged on top of the PCB. This sort of extension will not be part of the vendor's service or support unless individually negotiated.

On special request, the vendor is able to provide alternative or updated firmware for the programmer device. This might be in the form of a semiconductor that the user can interchange with the default firmware or by means of new binary provided firmware data.

The programmer hardware might of course be capable of much more features, like transfers of application specific data towards and back from the EVM device (e.g. in form of a DLL), exchange of diagnostics and a wide variety of debugging information (e.g. as a stand alone application or a CCS module) if there has been put some intelligent software on both ends of the device. Neither is this currently integrated into the provided client software nor is such customer specific EVM code included in any of the already published software samples. On customer demand DSP-Weuffen GmbH is able to offer additional services that fulfill a much wider range of features with the provided interface hardware.

8 Appendix

8.1 Abbreviations

DSP	Digital Signal Processor
TI	Texas Instruments
ROM	Read Only Memory, non volatile
FLASH	A ROM, but programmable only in larger blocks after an explicit erase
RAM	Random Access Memory
SDRAM	Synchronous Dynamic RAM
UHPI	Universal Host Port Interface
EMIF	Extended Memory Interface
PLL	Phase Locked Loop
IRQ	Interrupt Request
USB	Universal Serial Bus
MCP	Microprocessor
EVM	Evaluation Module
IDE	Integrated Development Environment
CCS	Code Composer Studio (TI IDE)
GUI	Graphical User Interface
CLI	Command Line Interface
IDE	Integrated Drive Electronics
PCB	Printed Circuit Board
DLL	Dynamically Linkable Library

8.2 Common problems and solution approaches

- If the interface is not found by the application then check the USB connections, make sure its reported in the windows system application called Device Manager and try re-installing it's drivers for the USB port in question. If behavior does not change try replugging the interface or even perform a full reboot on your PC system.
- In case the programming application reports connection problems with the target then please make sure that the dip switches on the EVM are set correctly. Check for correct positioning of the interface on the EVM connector. Try if the problem vanishes if you use a different USB connector on your system.

8.3 Errata

2006-10-26: Communication between PC and the USB-UHPI programmer can fail if the programmer is attached to an USB hub. It is recommended to attach the programmer directly to a PC USB port.

2007-02-28: Even if attached directly to a PC USB port both *Read(Tek)* and *Flash(Tek)* from the GUI control application "*stand-alone\uhpi-dlg.exe*" can fail with the error message:

"Timeout: Access to UHPI not reenabled – invalid EVM state OnBtnReadTek failed"

The only known workaround is to press *Read(Tek)Again* or *Flash(Tek)Again* until success. There can be several consecutive failures before the action finally succeeds, so don't give up too early.