



Evaluation Module C6727 EVM

Technical Reference
V1.13 – 2007-07-27

Authors: Siegbert Baude

Dieter Weuffen

Thomas Dorn

Alexander Stohr

This page intentionally left blank

Contents

1	Safety Instructions	7
2	Contact	8
3	Introduction	9
3.1	Onboard features	9
3.2	Scope of Delivery	9
4	Functions of the TMS320C6727 EVM-Board	11
4.1	Block Diagram	11
4.2	Overview	12
4.3	DSP Configuration for using Peripheral Bus Units	13
4.3.1	EMIF Address Ranges	13
4.3.2	Flash Banks	15
4.4	EMIF Address Mapping	16
4.4.1	Onboard SDRAM (128Mbit, 16Mbyte)	16
4.4.2	Onboard Flash (512kByte)	17
4.4.3	External Memory Expansion via Connectors CE2, CE3, CE4	17
4.4.4	CPLD Register	19
4.5	Hints for Practical Use	20
4.6	Boot Mode	20
5	Getting Started	23
5.1	General Remarks	23
5.2	Recommended Tools	23
5.3	Installation Procedure	23
5.4	Configure Your Project:	24
5.5	Loading a Project in CCS	24
5.6	Running an Application from Internal RAM	25
5.7	Running an Application from External RAM	25
5.8	Running an Application from Flash	25
5.9	Boot process	26
5.10	Tools	28
5.11	Codec Example	29
5.12	UHPI Usage	29
5.13	Power Consumption	29
6	Appendix	30
6.1	Abbreviations	30

6.2	Part Placement	31
6.3	Connectors	33
6.3.1	Audio In/Out	33
6.3.2	Power Supply Socket	34
6.3.3	JTAG	34
6.3.4	External Connectors	35
6.4	Errata	41

Figures

Figure 1	Board layout of the EVM-board	10
Figure 2	C6727 EVM Block Diagram	11
Figure 3	Bus Interface (EMIF) memory ranges	14
Figure 4	Sub division of Flash memory ranges into banks	15
Figure 5	Memory mapping	16
Figure 6	Boot mode DIP switches	20
Figure 7	Application startup program flow	26
Figure 8	Parallel flash boot overview	27
Figure 9	Part placing top view	31
Figure 10	Part placing Bottom view	32
Figure 11	Jack 3.5mm	33
Figure 12	Power supply socket	34
Figure 13	Standard JTAG interface	34
Figure 14	Pin numbering for 40 pin (2x20) Headers	35

Tables

Table 1	Control signals for CPLD	13
Table 2	CPLD register	19
Table 3	Available boot modes and needed pin levels	21
Table 4	Switch settings for selecting boot mode with DIP switches at device RESET	21
Table 5	UHPI boot mode details	22
Table 6	Connectors	33
Table 7	Microphone	34
Table 8	Headphone, line-in, line-out	34
Table 9	Power supply connector	34

Table 10 JTAG port pins.....	34
Table 11 Header X1 – Bus Interface Connctor (EMIF)	35
Table 12 Header X2 – Bus Interface Connctor (EMIF)	36
Table 13 Header X4 – Host Port / GPIO Connector	37
Table 14 Header X5 – Host Port / GPIO Connector	38
Table 15 Header X11 – Audio/Serial Peripheral Connector.....	39
Table 16 Header X12 – Audio/Serial Peripheral Connector.....	40

Changes

Ver.	Date	Description	Executor
--	13.02.2006	Start of Document	Thomas Dorn
--	16.02.2006	First draft version	Thomas Dorn
v1.01	20.03.2006	First release version	Dieter Weuffen
v1.02	08.05.2006	Minor corrections	Siegbert Baude
v1.03	16.06.2006	Reflecting hardware changes, extended description of use	Siegbert Baude
v1.05	27.06.2006	Added description of software examples	Siegbert Baude
--	01.07.2006	Added external connectors, revisited external memory	Siegbert Baude
v1.08	06.07.2006	Added description of boot mechanism	Dieter Weuffen
v1.09	21.07.2006	List doc versions, minor layout adjustments, part list errata.	Alexander Stohr
v1.10	24.07.2006	Description of FlashBurn refined	Dieter Weuffen
v1.11	07.11.2006	Updated pinout and descriptions of header X5, added contact information	Siegbert Baude Alexander Stohr
V1.12	06.12.2006	Improved DIP-switch figure, corrected phone numbers	Siegbert Baude
V1.13	27.07.2007	Unified connector naming, graphics, EMIF address mapping	Alexander Stohr

Used Documents and References

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

Pos.	Description
1	TMS320C6727 Floating-Point Digital Signal Processors Rev. C (Datasheet\sprs268c - tms320c6727.pdf)
2	AMD Flash AM29LV400B (Datasheet\am29lv400b - Flash.pdf)
3	Micron SDRAM (Datasheet\128MbSDRAMx32.pdf)
4	TLV320AIC23 Audio Codec (Datasheet\tlv320aic23 - Codec.pdf)
5	CCS IDE Quick Start (TI\sprue40 - CCS IDE Quick Start.pdf)
6	CPU and Instruction Set (TI\spru733 - CPU.pdf)
7	Optimizing Compiler v 6.0 Beta (TI\spru187n - C-Compiler.pdf)
8	Assembly Language Tools (TI\spru186p - Assembly Language Tools.pdf)
9	Peripherals Overview (TI\spru723a - peripheral overview.pdf)
10	External Memory Interface (TI\spru711b - EMIF.pdf)
11	Multichannel Audio Serial Port (TI\spru878a - McASP.pdf)
12	ROM (TI\sprs277 - on chip ROM.pdf)
13	Software-Programmable Phase-Locked Loop Controller (TI\spru879a - PLL.pdf)
14	Real-Time Interrupt (TI\spru717 - Real-Time Interrupt.pdf)
15	Serial Peripheral Interface (TI\spru718 - SPI.pdf)
16	Universal Host Port Interface (TI\spru719 - UHPI.pdf)
17	Dual Data Movement Accelerator (TI\spru795c - dMAX.pdf)
18	Inter-Integrated Circuit (I ² C) Module (TI\spru877c - I2C.pdf)
19	Lattice CPLD ispLSI® 2032VE (Lattice\2032ve.pdf)
20	EVM C6727 Technical Reference, this document in high resolution. Always check www.dsp-weuffen.de for the newest version. (DSP-Weuffen\Technical reference.pdf)
21	Schematic Diagram, silk screen, copper layers, bill of materials (DSP-Weuffen\Schematics.pdf)
22	Silicon Errata TMS320C6727 (Datasheet\ sprz232b - Silicon errata.pdf)

1 Safety Instructions


Keep the unit away from heat sources.

Do not use the unit near water.

Unplug the power lead if the unit will not be used for a long period.

Remove power source when attaching or removing modules or JTAG-connector!

Mains safety:

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

Caution

The flash memory and DSP are static sensitive. Use proper precautions against static electricity damage at all times.

Use caution when designing external hardware, which is to be connected to the EMIF. The signals present on the interface connector are extremely high speed and failure to handle them appropriately can cause functional problems with the FIFO Port. We do not recommend driving cables directly as capacitive load. Ringing issues can cause corruption of the transmitted data. Whenever you use the board, plug in the power supply unit last.

Due to capacitive-filters inside the AC-supply unit, there is a voltage of about half the AC-supply voltage with a high impedance to be found on the 5V DC-supply plug. Therefore care has to be taken that the AC-supply plug is the last plug when connecting the board. This capacitive coupled voltage can destroy your application circuit or the emulators connected to the board, if you connect the AC-plug first. This behavior is common to most EVM-board AC-supplies and should be taken into account anytime.

2 Contact

Address:	DSP-Weuffen GmbH Bernhard-Müller-Str. 11 D-88239 Wangen Germany
Phone:	+49-(0)7528-9755-31
Fax:	+49-(0)7528-9755-32
Mail:	info@dsp-weuffen.de
Web:	www.dsp-weuffen.de

3 Introduction

This TMS320C6727 EVM-board is a low-cost development platform, which enables users to evaluate and develop applications for the TI C6727 DSP family. Schematics, logic equations and application notes are available to speed up hardware development and reduce time to market.

3.1 Onboard features

Many onboard features enable the user to connect a variety of application environments. The following ones are included (see Figure 1)

- TI TMS320C6727 DSP at 300 MHz (silicon version 1.0).
- AIC23 stereo codec
- SDRAM 128Megabit
- FLASH 4 Megabit 256k x 16-Bit
- 4 x pushbuttons and 4 x LEDs for user applications
- High and low voltage input and output headphone plugs (microphone in, speaker out, line in, line out)
- JTAG connector
- Connectors for UHPI or expansion for low cost USB-connector-card via UHPI
- Connectors for memory interface expansion
- Boot options selectable via DIP switches (parallel flash on EM_CS[2], SPI0 or I²C1 master mode from serial EEPROM, SPI0 or I²C1 slave mode from external MCU, UHPI from an external MCU)
- +5V single voltage power supply

3.2 Scope of Delivery

- EVM-board C6727
- Power supply unit PSM11R-050 with international connectors *or* Power supply unit PSS-2500SV with euro plug and six secondary adapter plugs
- CD with software examples, data sheets and documentation in PDF format
- Printed manual
- Headers for external connectors X1, 2, 4, 5, 10, 11, 12
The headers X1, 2, 4, 5, 11, 12 are not soldered to provide more flexibility for the user.

C6727 EVM

Technical Reference v1.13 – 2007-07-27

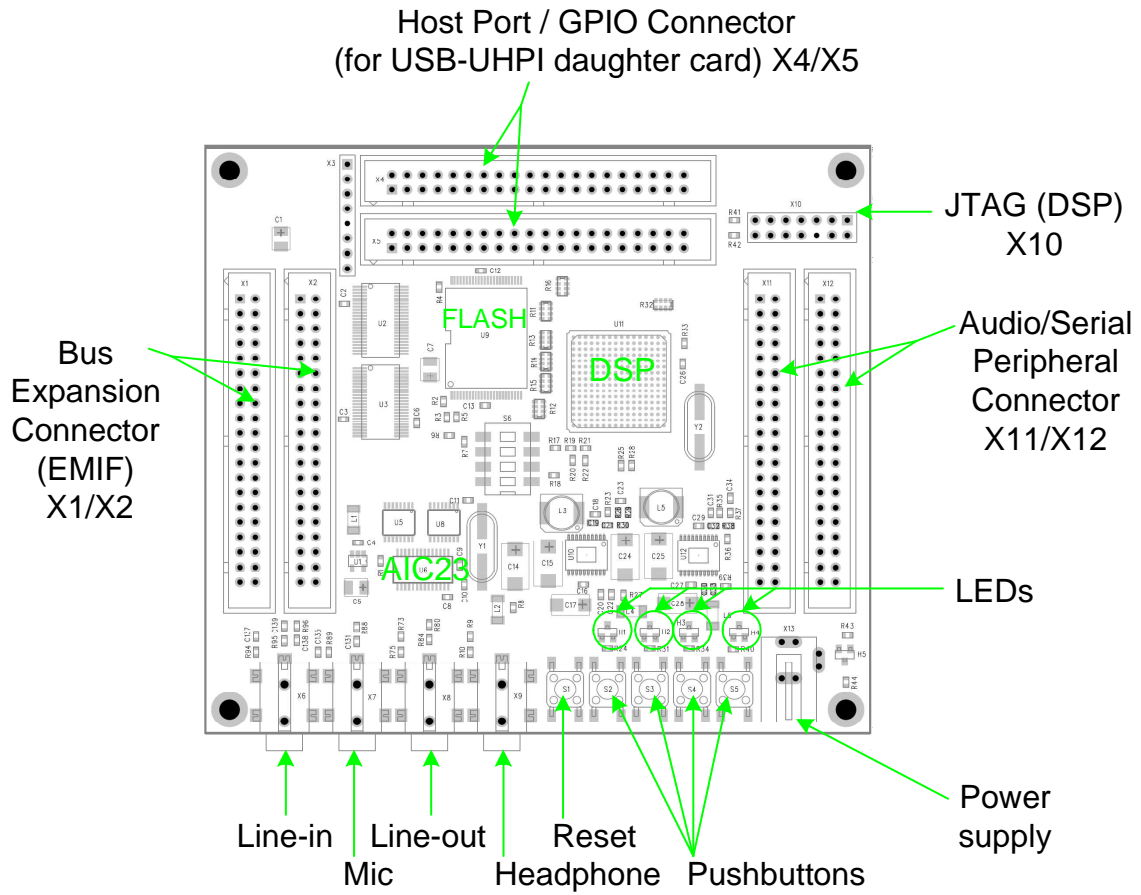


Figure 1 Board layout of the EVM-board



4 Functions of the TMS320C6727 EVM-Board

4.1 Block Diagram

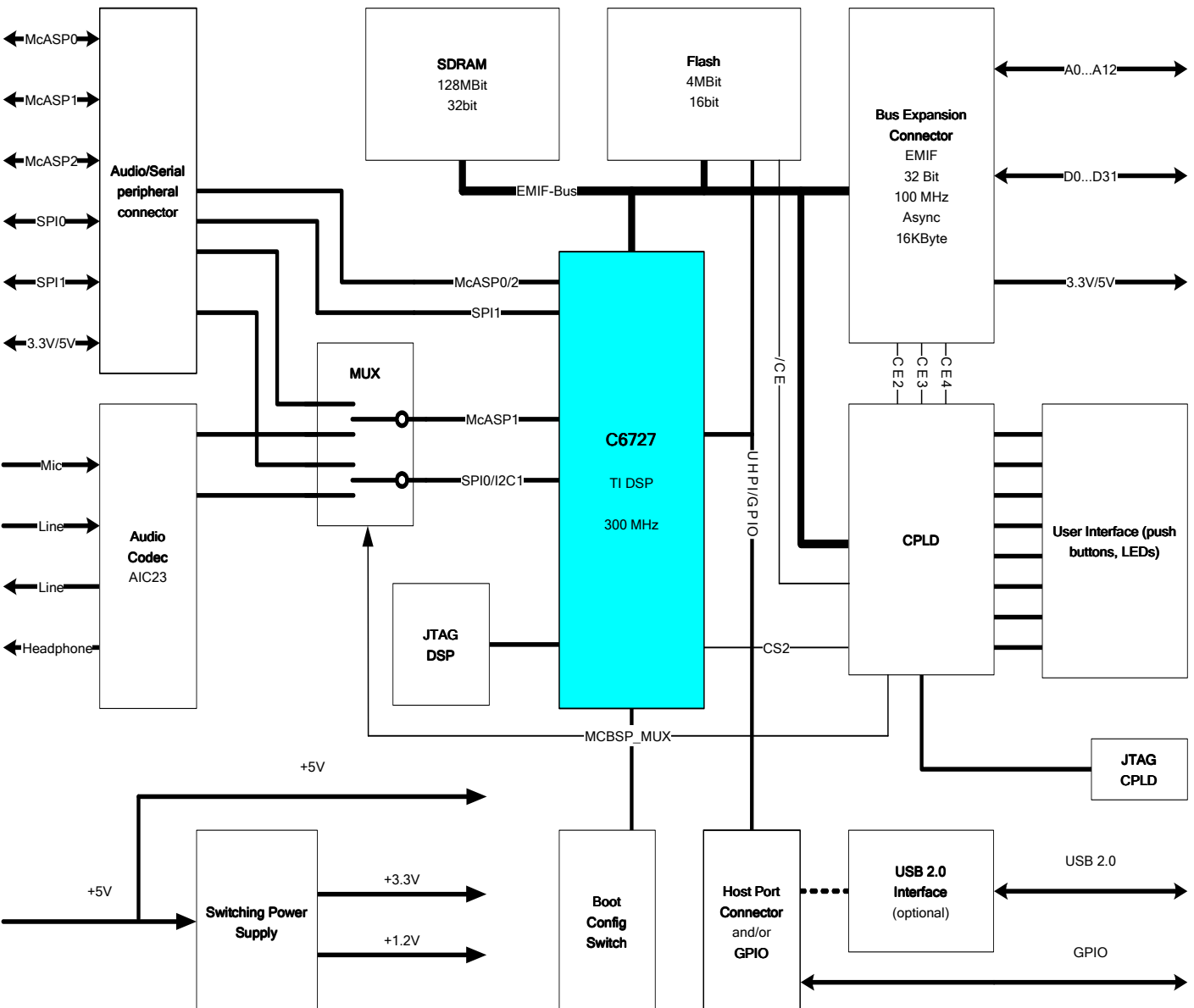


Figure 2 C6727 EVM Block Diagram

4.2 Overview

The TMS320C6767 EVM-board includes a 13-bit wide address bus and a 32 bit wide data EMIF. The SDRAM and flash are connected onboard to these busses. The EMIF is also connected to the daughter card expansion connectors that are used for external applications.

The 16-bit wide data bus of the flash (TD0 to TD15) is interfaced to the lower 16-bit of the DSP's 32-bit wide data bus (DQ0 to DQ15). The flash is set to word-width (16-bit-wide) data transfer by not using the BA0 byte-address pin of the DSP. Only for EVM internal use (flash, SDRAM), the address range of the 13-bit wide address bus is expanded by a CPLD with the signals UHPI_HD16 to UHPI_HD22, which for this purpose are also called G_A15 to G_A21 accordingly. With this setting, the signals UHPI_HD0 to UHPI_HD15 and UHPI_HD23 to UHPI_HD31 are always available for GPIO purposes. If you want to use UHPI_HD16 to UHPI_HD22 for GPIO, you have to change the configuration. The three control signals for the CPLD (GA_19, GA_20, GA_21) manage the switching between using flash, the daughter card interface or the CPLD registers. These signals must be set as needed by the DSP GPIOs. Note: This use of flash addressing using GPIO may conflict with your application.

The CPLD includes firmware that handles the expanded address management. Although it is possible, do *not* attempt to change the CPLD firmware.

Four pushbuttons and four LEDs enable the user to check or set states of the application. The CPLD onboard registers control this user interface. These registers are set and read by the DSP (see [Table 2](#)).

McASP0 and McASP2 are connected to the external headers X11 and X12. The McBSP_MUX routes McASP1, I²C and SPI either to the headers X11 and X12, too, or to the onboard AIC23 stereo codec. The DSP transmits and receives analog audio over this codec, which is also attached to four 3.5mm audio jacks (microphone input, headphone output, line input, line output).

Code Composer Studio normally uses the JTAG interface to load and evaluate firmware applications, which requires an emulator. Another possibility is to use a low-cost USB emulator expansion card, which has to be plugged into the UHPI expansion connector. In this case, the UHPI may not be used for the application. The USB card is sold separately. The user can download applications into the flash over the USB connection. In addition, a library for the PC as well as for the DSP is included which gives the user the ability to exchange data with the host via the USB.

A 5V external power supply is used to power the board for stand-alone applications. Onboard regulators provide both a 1.2V supply for the DSP core and a 3.3V supply for I/O demand.

4.3 DSP Configuration for using Peripheral Bus Units

The DSP has to be configured before you can access the flash memory, the daughter card, the user interface via the CPLD or any other expansion bus (EMIF) based unit or. For this the data direction of the GPIOs for GA[21:19] must be set to output. For accessing the on board flash further the data direction of the GPIOs for GA[18:15] must be set to output as well.

4.3.1 EMIF Address Ranges

For addressing an EMIF unit the state of these GPIOs must be set like shown in [Table 1](#) or [Figure 3](#).

GA_21 / UHPI_HD22	GA_20 / UHPI_HD21	GA_19 / UHPI_HD20	Selected Range / CPLD functionality
0	0/1 ¹	0/1 ¹	Flash addressing (/CE) for flash ranges 0 to 3
1	0	0	Expansion memory addressing (CE_2)
1	0	1	Expansion memory addressing (CE_3)
1	1	0	Expansion memory addressing (CE_4)
1	1	1	Read or write the CPLD registers

Table 1 Control signals for CPLD

¹ For the 512kB Flash only state 0-0-0 is valid. The maximum addressable amount of memory is 2MB.

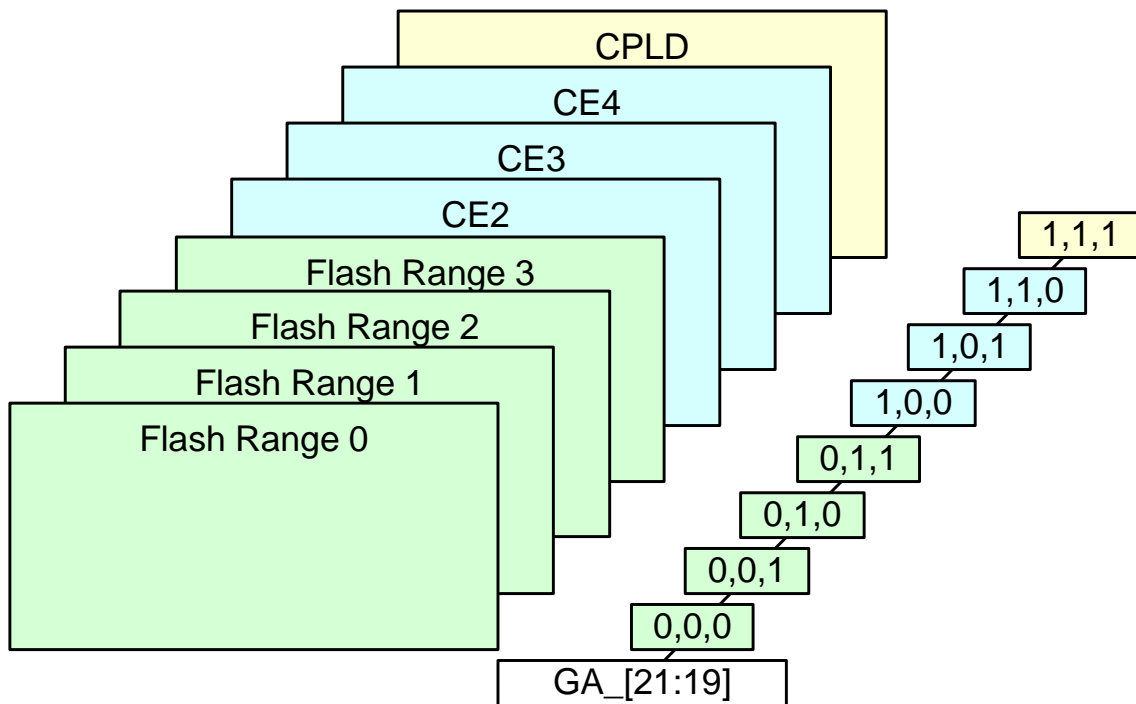


Figure 3 Bus Interface (EMIF) memory ranges

Use much caution if you are going to attach some own hardware that will be able drive exactly those signals. When booting from the default on-board flash then please make sure that range 0 is selectable.

Access to external memorys is only possible when the correct range got setup. This means that with GA_[21:19] you are selecting your desired range (flash[3:0], CE[4:2], or CPLD). It is the responsibility of the user to set all needed GPIOs during program execution and debugging.

4.3.2 Flash Banks

Inside a flash range you further have to use GA_[18:15] to address each of the sixteen 32kB banks in those range. With the 512kB flash chip you will receive a set of a single range with a total of sixteen banks. With a 2MB flash chip you would get a total of 64 banks à 32kB. See also Figure 4.

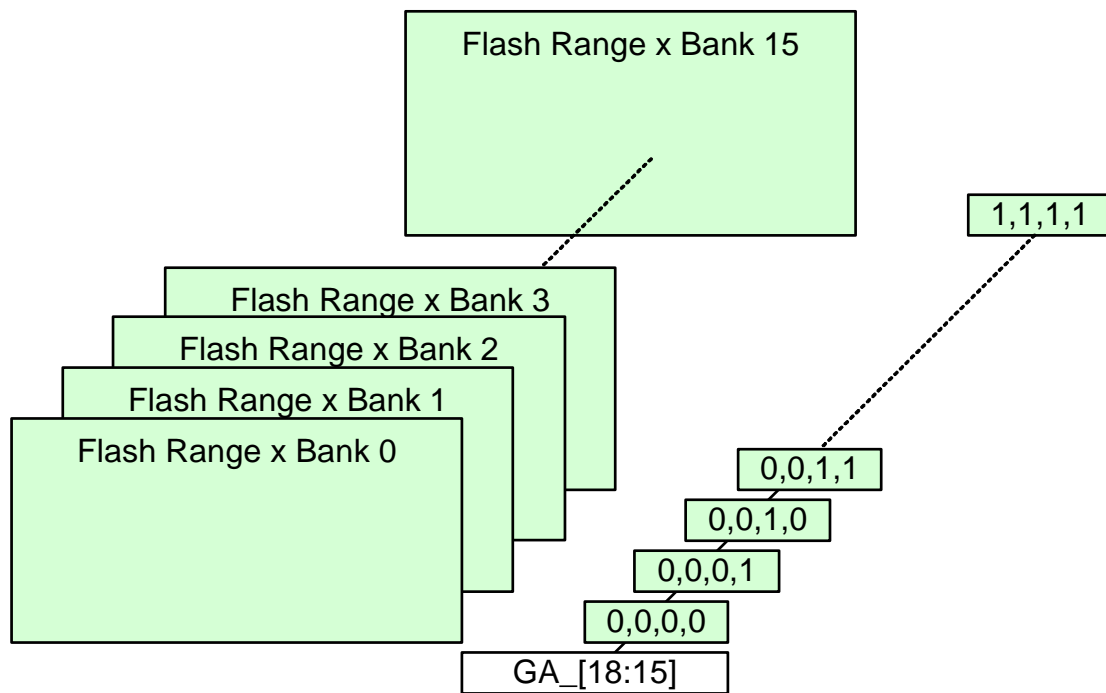


Figure 4 Sub division of Flash memory ranges into banks

For more details on the address decoding of the flash unit please see chapter 4.4.2 and further see the provided schematics.

4.4 EMIF Address Mapping

The EMIF address mapping of the DSP is shown in Figure 5.

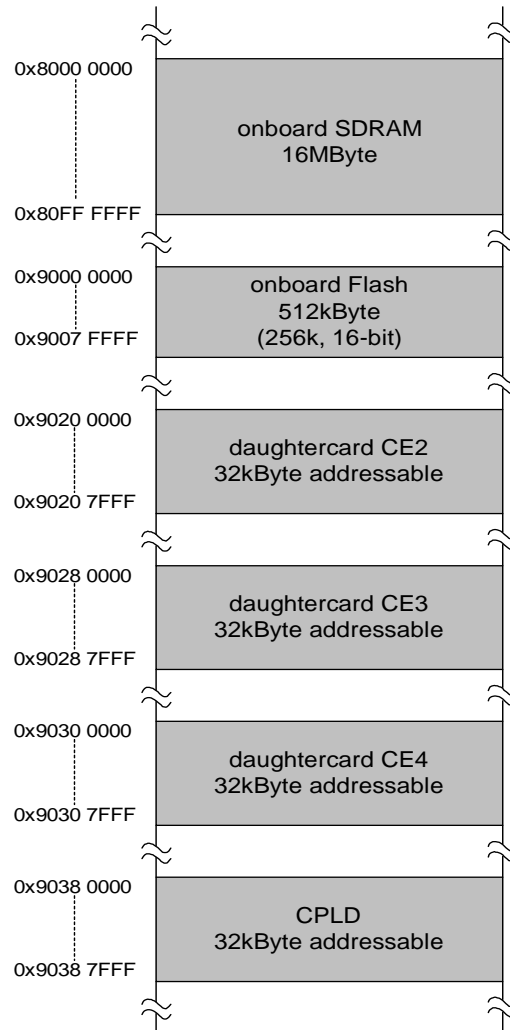


Figure 5 Memory mapping

4.4.1 Onboard SDRAM (128Mbit, 16Mbyte)

Address range: 0x8000 0000 ... 0x80FF FFFF

Data width: 32-bit

The onboard SDRAM is addressed by the EM_CS0* pin that is set by the DSP when addressing 0x8000 000 to 0x8FFF FFFF. The DQM0 ... DQM3 wires are used to determine the chosen byte in the 32-bit wide address.

The access of the SDRAM is handled by the DSP. However, you have to initialize the SDRAM either in boot code (or when attaching by a debugger and CCS with a GEL file), see 5.8.

4.4.2 Onboard Flash (512kByte)

Address range: 0x9000 0000 ... 0x9007 FFFF

Data width: 16-bit

Data address ability: word (16-bit)

Address signals: BA1, TA0 ...TA12 (→ 14 address wires)

The CE addresses the onboard flash and is managed by the CPLD (see [Table 1](#)), which uses the EM_CS2* pin (set by addressing from 0x9000 0000 to 0x9FFF FFFF) and the CPLD control signals.

Address bit	-	-	-	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Signal	G_A...							EM_A...													EM_BA...		
Value	0 ¹	0 ¹	0 ¹	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	- ²

With the 19 pins EM_BA[1:0], EM_A[12:0], and G_A[18:15] you can address the range 0x0 0000 to 0x7 FFFF equivalent to 512kB memory.

¹ These three pins address the flash via the CPLD that sets CE*, because it gets EM_CS2 if an address between 0x9000 0000 to 0x9FFF FFFF is addressed, see [Table 1](#).

² EM_BA0 is not physically attached, because the flash has 16-bit data width. Nevertheless it is part of the logical addressing.

4.4.3 External Memory Expansion via Connectors CE2, CE3, CE4

Address range: 32kByte per connector

Address range for CE2: 0x9020 0000 ... 0x9020 7FFF

Address range for CE3: 0x9028 0000 ... 0x9028 7FFF

Address range for CE4: 0x9030 0000 ... 0x9038 7FFF

Data address ability: byte / 2-byte / 4-byte (8 bit / 16 bit / 32 bit)

Address signals: BA0, BA1, TA0 ...TA12 (→ 15 address wires)

Data width: 32-bit

The expansion memory is addressed by the CE2* managed by the CPLD (see [Table 1](#)), which uses the EM_CS2* pin (set by addressing from 0x9000 0000 to 0x9FFF FFFF) and the CPLD control signals.

CE2:

Address bit	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Signal	G_A...							EM_A...														EM_BA...	
	21	20	19	18	17	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	
Value	1 ³	0 ³	0 ³	- ⁴	- ⁴	- ⁴	- ⁴	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

With the 22 pins EM_BA[1:0], EM_A[12:0], and G_A[21:15] you can address the range 0x20 0000 to 0x20 7FFF equivalent to 32kB memory.

³ These three pins together with EM_CS2 address the CE2 external memory via the CPLD that sets CE2*.

⁴ unused.

CE3:

Address bit	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Signal	G_A...							EM_A...														EM_BA...	
	21	20	19	18	17	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	
Value	1	0	1	-	-	-	-	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	

You can address the range 0x28 0000 to 0x28 7FFF.

CE4:

Address bits	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Signal	G_A...							EM_A...														EM_BA...	
	21	20	19	18	17	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	
Value	1	1	0	-	-	-	-	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	

You can address the range 0x30 0000 to 0x30 7FFF.

4.4.4 CPLD Register

Address: 0x9038 0000

Data width: 8-bit

CPLD:

Address bits	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Signal	G_A...							EM_A...														EM_BA...	
	21	20	19	18	17	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	
Value	1 ⁵	1 ⁵	1 ⁵	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	- ⁴	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	

With the 22 pins EM_BA[1:0], EM_A[12:0], and G_A[21:15] you can address byte 0x38 0000.

Independent of the setting of the unused pins the CPLD always sets or returns the content of its eight registers.

⁴ unused.

⁵ These three pins together with EM_CS2 address the CPLD.

The CPLD has a register that can be read / written by the DSP. By either reading or setting the CPLD register, the pushbuttons are read or the LEDs are set accordingly. [Table 2](#) shows the CPLD register.

Bit	7	6	5	4	3	2	1	0
Read	Reserved	Reserved	Reserved	Reserved	Button 3	Button 2	Button 1	Button 0
Write	Reserved	Reserved	Reserved	McBSP_MUX	LED 3	LED 2	LED 1	LED 0

Table 2 CPLD register

The McBSP_MUX signal in the circuit resembles bit 4 of the CPLD register. It is responsible for the routing of McASP1, I²C and SPI. These can be either attached to the AIC23 codec or to the external headers X11 and X12. If McBSP_MUX is low then the signals are routed to the external connectors, if it is high they are routed to the internal codec. Note that you have to set UHPI_HD22, UHPI_HD21, and UHPI_HD20 according to [Table 1](#) before you can modify the CPLD register. Have a look at the code example "Codec" for details.

4.5 Hints for Practical Use

As you can only access one of the three extension modules (flash, daughter card interface, CPLD) at the very same time, you have to make sure that the three control signals for the CPLD (GA_19, GA_20, GA_21) are always set as needed. This is also necessary during debugging.

The CPLD register bits are mutually independent for reading and setting. This means, you cannot read back the settings that you have just written. Therefore it is good practice to keep the written values in a static variable. As the external bus is operating much slower than the DSP core you might be able to change the state of the GPIO pins much faster than such a transfer would need to complete. In order to wait for the external bus it is recommended to let the DSP core perform a single read access to the external bus after you have performed your last write access before reprogramming the GPIOs.

4.6 Boot Mode

The boot mode DIP switches can select the boot mode. The C672x DSP supports only one hardware boot mode option, this is to boot from the internal ROM starting at address 0x0000 0000. A software boot loader stored in ROM implements other boot mode options. The software boot loader uses the CFGPIN0 and CFGPIN1 registers, which capture the state of various device pins during an externally applied RESET, to determine which mode to enter. In practice, the software uses only a few pins. Note, that using the flash boot mode without having any application inside the flash will cause the DSP executing undefined code.

boot mode DIP switch

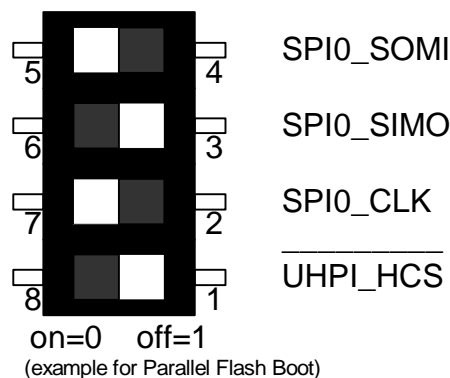


Figure 6 Boot mode DIP switches

The ROM boot modes include the following options:

- UHPI from an external MCU
- Parallel flash on EM_CS[2]
- SPI0 or I²C1 master mode from serial EEPROM
- SPI0 or I²C1 slave mode from external MCU

These are the boot modes that can be selected by driving the DSP pins at start up time:

BOOT MODE	UHPI_HCS	SPI0_CLK	SPI0_SIMO	SPI0_SOMI
UHPI (host port boot)	0	NMUX ¹	FULL ¹	BYTEAD ¹
USB-UHPI with DSP-Weuffen daughter card	0	0	0	0
Parallel flash ²	1	0	1	0
SPI0 master	1	1	0	0
SPI0 slave	1	1	1	0
I ² C1 master	1	1	0	1
I ² C1 slave	1	1	1	1

Table 3 Available boot modes and needed pin levels

¹ Refer to the C9230C100 TMS320C672x Floating-Point Digital Signal Processor ROM Data Manual (literature number SPRS277) for details on supported boot modes and their implementation. See also the details in Table 5.

² Boot mode for booting from onboard flash.

These are the boot modes as they are selectable by the 4 element on board DIP switches:

BOOT MODE	SW1	SW2	SW3	SW4
UHPI (host port boot)	ON	NMUX ¹	FULL ¹	BYTEAD ¹
USB-UHPI with DSP-Weuffen daughter card	ON	ON	ON	ON
Parallel flash ²	OFF	ON	OFF	ON
SPI0 master	OFF	OFF	ON	ON
SPI0 slave	OFF	OFF	OFF	ON
I ² C1 master	OFF	OFF	ON	OFF
I ² C1 slave	OFF	OFF	OFF	OFF

Table 4 Switch settings for selecting boot mode with DIP switches at device RESET

BIT NO.	NAME	RESET VALUE	READ WRITE	DESCRIPTION
31:5	Reserved	N/A	N/A	Reads are indeterminate. Only 0s should be written to these bits.
4	BYTEAD	0	R/W	UHPI Host Address Type 0 = Host Address is a word address 1 = Host Address is a byte address
3	FULL	0	R/W	UHPI Multiplexing Mode (when NMUX = 0) 0 = Half-Word (16-bit data) Multiplexed Address and Data Mode 1 = Fullword (32-bit data) Multiplexed Address and Data Mode
2	NMUX	0	R/W	UHPI Non-Multiplexed Mode Enable 0 = Multiplexed Address and Data Mode 1 = Non-Multiplexed Address and Data Mode (utilizes optional UHPI_HA[15:0] pins). Host data bus is 32 bits in Non-Multiplexed mode. Setting this bit prevents the EMIF from driving data out or 'parking' the shared EM_D[31:16]/UHPI_HA[15:0] pins.
1	PAGEM	0	R/W	UHPI Page Mode Enable (Only for Multiplexed Address and Data Mode). 0 = Full 32-bit DSP address specified through host port. 1 = Only lower 16 bits of DSP address are specified through host port. Upper 16 bits are restricted to the page selected by CFGHPIAMSB and CFGHPIAUMB registers.
0	ENA	0	R/W	UHPI Enable 0 = UHPI is disabled 1 = UHPI is enabled. Set this bit to '1' only after configuring the other bits in this register.

Table 5 UHPI boot mode details

Since the lines from the UHPI are multiplexed with the data bus and the flash address lines, care must be taken to select the proper UHPI mode. I.e., set FULL and NMUX to zero to avoid conflicts.

5 Getting Started

5.1 General Remarks

The ROM boot loader of the C6727 initializes some peripherals, because it tries to boot from them. Therefore these peripherals may be in a different state than described in the TI documentation. For example, before using the I²C connected to the codec, you have to disable the SPI.

5.2 Recommended Tools

- Emulator XDS510PP plus from Spectrum Digital
<http://support.spectrumdigital.com/ccs31/>
- Code Composer Studio IDE 3.1
At <http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html> you can purchase CCS IDE v3.1 or get a 120-day Free Evaluation version. The “Quick Start Reference Guide” (sprue40.pdf, 389 KB) that you will find there is also included on our CD in the “documentation” folder.
- At <http://focus.ti.com/docs/toolsw/folders/print/sprc223.html> download chip support library CSL 3.00.08.04 or newer (sprc223.zip).

Via TI’s “update advisor” you will also get:

- CCS-Patch for C6727
- DSP/BIOS 4.90 or newer
- Code generation tools 5.30

5.3 Installation Procedure

Installing Code Composer Studio IDE:

Note: Before starting the Code Composer Studio IDE installation, be sure that any antivirus software is disabled. You must install and run Code Composer Studio IDE using administrator privileges.

1. Insert the installation CD into the CDROM drive. An install screen should appear after a few seconds; if not, go to Windows Explorer and run setup.exe from your CD-ROM.
2. Respond to the dialog boxes as the installation program runs. You can customize the components that are installed. The path of your CCS installation will be referred as <CCS_INSTALL_PATH> from then on.
3. After the installation, you will be directed to the TI home page, where you can register to get updates.
4. Use the “update advisor” to download all suggested updates.
5. Install the CCS3.1 Service Pack 1.

6. Install the C672x chip support package.
7. Unzip the sprc223.zip-file. You will find three included zip files called “csl_C672x_v3...”, “csl_C672x_intc_v3...”, and “csl_C672x_src_v3...”. Unzip the “csl_C672x_v3...” and move the complete csl_C672x-folder to <CCS_INSTALL_PATH>\C6000\ so that you end up with <CCS_INSTALL_PATH>\C6000\csl_C672x.
8. To work around an inconsistency between “csl_types.h” and “std.h” change the following lines in <CCS_INSTALL_PATH>\C6000\csl_C672x\dsp\inc\csl_types.h:

From:	To:
<pre>#define TRUE ((Bool) 1)</pre>	<pre>#ifndef TRUE #define TRUE ((Bool) 1) #endif</pre>
<pre>#define FALSE ((Bool) 0)</pre>	<pre>#ifndef FALSE #define FALSE ((Bool) 0) #endif</pre>

9. Install Spectrum Digital’s SD CCS3.1 Emulation Drivers.
10. Copy C6727EVM.gel file to <CCS_INSTALL_PATH>\cc\gel\C6727EVM.gel.
11. Start “Setup CCStudio v3.1”.
12. Install the board of your preferred emulator.
13. Right-click on the board’s c672x icon, choose properties and select the gel file C6727EVM.gel that you copied in step 8.
14. Click “Save&Quit”.
15. Confirm “Start Code Composer Studio on exit” with “Yes”.
16. Inside CCS connect to the board with ALT-C.

5.4 Configure Your Project:

To use the C6727 chip support library, you have to add a search path to your project configuration.

Right-click your project and choose “Build Options”. Click on the “Compiler” tab and choose

Category -> Preprocessor. In the “Include Search Path” field add

<CCS_INSTALL_PATH>\C6000\csl_C672x\dsp\inc\

Then choose the “Linker” tab and add in the “Library Search Path”

<CCS_INSTALL_PATH>\C6000\csl_C672x\dsp\lib\

5.5 Loading a Project in CCS

To use the provided code examples, copy them from the CD-folder “Examples” to your hard disk. For

each example you will find a project file, e.g. in the folder “Examples\Blink” you will find “Blink.pjt”.

Inside CCS click on Project->Open and select the project file. You can compile it with “F7”, connect to the EVM board with “ALT-C”, load the produced object file “Examples\Blink\Debug\Blink.out” with “CTRL-L” and start the application with “F5”. To stop the application you can use “SHIFT-F5” and to disconnect from the board again “ALT-C”.

5.6 Running an Application from Internal RAM

Use CCS 3.x and JTAG emulator → boot mode DIP switches don’t matter.

The included “Memtest” program is an example for loading an application into the internal RAM. It can be used to test the internal and external RAM of the board.

5.7 Running an Application from External RAM

Use CCS 3.x and JTAG emulator.

Important: When using the UHPI-Boot mode, the HPI must be set into HPI-Mode (C6713-compatible) before accessing the SDRAM. See 4.5.

The GEL file “C6727EVM.gel” provided with the board initializes the EMIF and the PLL before loading the program. Note: The EMIF has to be initialized before you can use the SDRAM. This is done by the GEL file functions.

The included “Blink” program is an example for loading an application into external RAM. It can be used to test the LEDs and pushbuttons of the board. You can also burn this application into flash and start it from there with the appropriate DIP switch setting (see 4.5 and 5.8).

5.8 Running an Application from Flash

Use CCS 3.x and JTAG emulator → boot mode DIP switches must be set to “parallel flash”

The GEL file initializes the EMIF for usage of the SDRAM and the flash as well. Using the flash burn utility, which comes with the board, can overwrite the application program. See the utilities readme for further explanations. The current version of the flash burn utility can be downloaded from:

www.softwaredesignsolutions.com

The source code of the flash burn utility part that runs on the DSP side (“FBTC C6727 EVM”) is not included, but can be bought from software design solutions as part of the flash burn porting kit package. Please ask us for a general flash burn routine. With the USB daughter board, we will ship the source code of the specific flash routine that works with that tool.

For your convenience you will find a variant of our “Blink” example already burnt into flash. If you set the DIP switches to “parallel flash” mode and reset the board, you will see some blinking LEDs.

5.9 Boot process

Figure 7 shows the DSP-startup in this case. First the on-chip boot code is executed. This boot code is device dependent. There are differences in the on-chip boot code between the DA710 and C6727. Please refer to the device documentation.

Second the on-chip boot code initializes the peripherals from which the chip tries to boot. Then it samples some dedicated pins (see 4.6) to decide from which peripheral it has to boot from. If it is the flash, then it reads an 8-bit value from CE2-address 0x90000000. This magic number is used to decide if there is an 8-bit or a 16-bit device attached.

Therefore the boot loader must contain this magic number set to 0x01010101 because the C6727 EVM-Board has a 16-bit flash.

After that, the on-chip boot loader copies the first 1k of code into the DSP's internal RAM located at 0x10000000. This is the so called "2nd stage bootloader" or "application loader", which is part of the "Blink" example. The source code of this loader can be found in the file boot_c6727.asm and may be used to adapt the bootloader to customer needs.

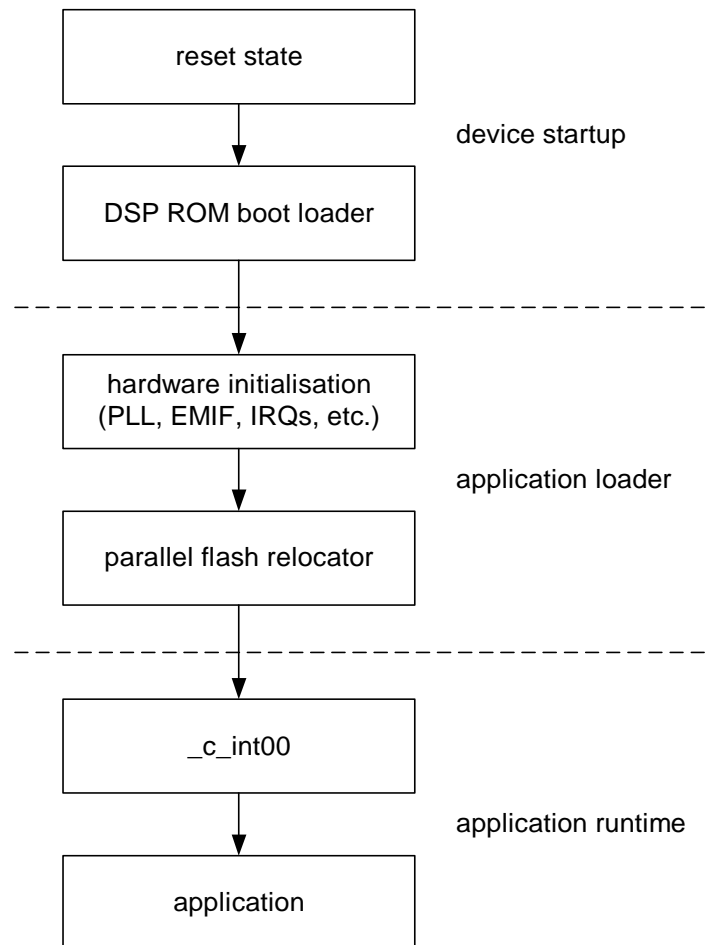


Figure 7 Application startup program flow

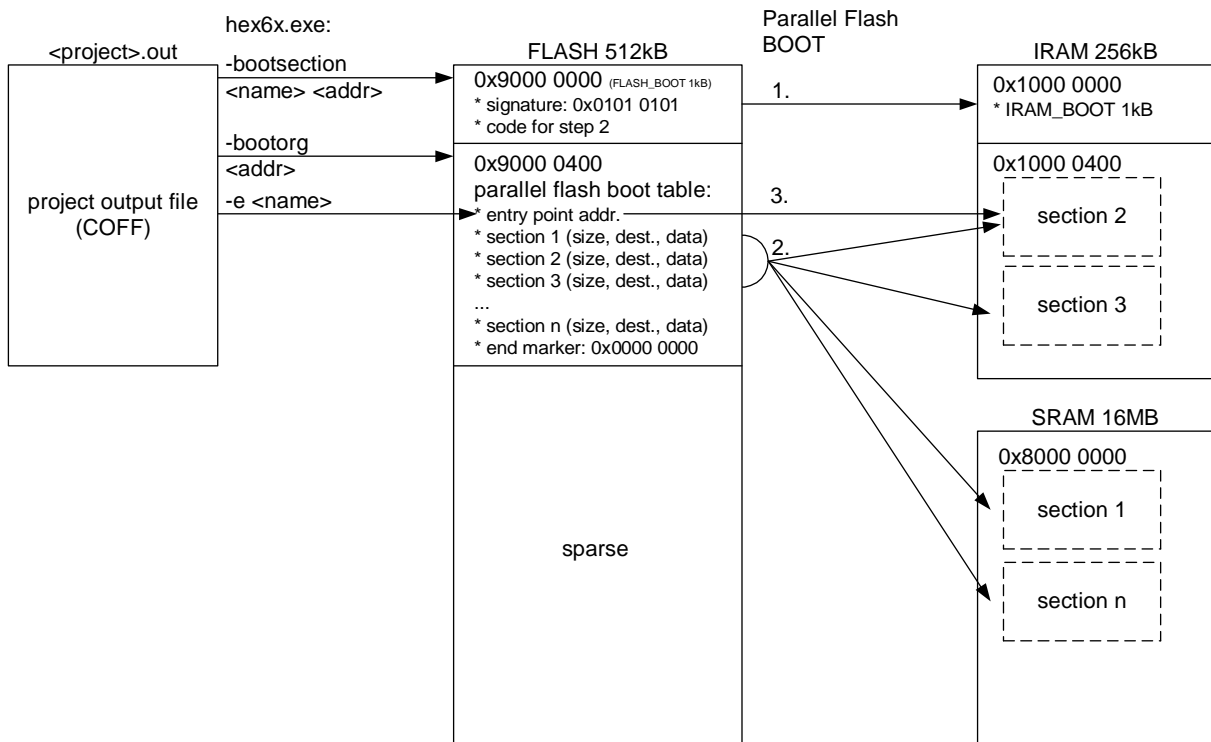


Figure 8 Parallel flash boot overview

The boot loader processes the following steps:

1. Initialize the EMIF and PLL

Since the boot loader is responsible for initializing the RAM with the application code, it has to initialize the used hardware (the external memory interface) first. The external RAM is an SDRAM type which needs a certain timing. Because this timing depends on the EMIF frequency, the boot loader has to initialize the PLL to the frequency of the final application as well. By default, it initializes the DSP to 300MHz core frequency and 100MHz EMIF clock. If you need different speeds, you will have to change the initialization values for SDRAM and PLL as well.

2. Test for USB download tool

Since the same boot loader code can be used in conjunction with the USB download tool, there is a decision made inside the boot loader, if this is a boot from UHPI (mode used when the USB tool is attached) or from flash. If the boot loader boots from UHPI it will simply wait for a handshake byte signaling that the USB download tool loaded all sections into the board RAM. Then it branches to the address given by the “-bootorg” command in the hex6x_pp_pf.cmd-command file. This file is used for starting the hex6x-utility from TI. The given address is usually

the “_c_int00”-symbol. For further description have a look at the documentation “USB-UHPI-Programmer User’s Guide” from DSP-Weuffen GmbH available at www.dsp-weuffen.de.

3. Section copy from flash to RAM

Figure 8 shows which sections are copied by the boot loader’s “parallel flash relocater”. The sections are organized according to the format the hex6x-utility generates. You find the format documented in “spru186n, chapter 11.9”.

Because the flash is banked (upper address lines are controlled by GPIOs), each access to Flash requires a write to the GPIOs first.

4. Jump to application start address

After all sections are initialized, the boot loader jumps to the address given by the option “-bootorg” in the hex6x_pp_pf.cmd command file. This address is usually the “_c_int00”-symbol.

5.10 Tools

When using the examples in the Code Composer environment without booting from flash, the boot_c6727.asm file can be omitted. Then the GEL file functions initialize the EMIF. But if an application needs booting from flash, you have to add the boot_c6727.asm file to your project. When using Code Composer you have to be aware that, if you set the boot mode DIP switches to “Boot from Flash” and you select “Debug\Reset CPU” in CCS, then the code from flash will be booted and executed instead of your application loaded via CCS “File\Load Program..”-menu. To avoid this, you can either omit the boot_c6727.asm-file or you can select “Debug\Restart” after a “Debug\Reset CPU”. This will set the program counter to the symbol _c_int00, which is the program startup entry point.

Creating a flash image is a task that can best be passed to the hex6x command line utility from TI by passing it a few options either on the command line or through some extra command file. In Figure 8 the most important switches for controlling the special flash image process are shown. There are most likely others to set, but they are not discussed here. Please read the tool’s documentation if you need to change any of the provided default setups. This configuration is already done in the examples provided with the C6727 EVM board. To create an image file use the helper command file “hex6x_pp_fb.cmd” for the flash-burn utility and “hex6x_pp_pf.cmd” for the USB programming utility.

You can either program the flash image into the flash device of the EVM using the provided toolset or you can load it directly into the RAM by means of the USB UHPI programmer interface. The former option allows booting the image later on directly from the parallel flash even as a stand alone device. Regardless which variant you will choose, it is always a three-step loading process where in step 1 the 2nd stage boot loader will get transferred to the start of the IRAM of the DSP. In step 2 the loaded code

gets started and some init (PLL, EMIF) takes place. At the end of the 2nd stage code, the application sections will get loaded in their destination RAM areas and the entry point address for the application will be determined. Finally the application will get started by a simple branch command.

5.11 Codec Example

Included you find a CCS project called “Codec” that demonstrates the use of audio loop-back from microphone or line-in to line-out and headphones. The program configures the codec via I²C and uses the pushbuttons to switch between different functions.

The functions are:

- Analog loop-through codec without data transfer to the DSP
- Digital loop-through using the McASP in polling mode
- Headphone mute functions of left and right codec channels (line-out is not affected)
- Headphone volume control (line-out is not affected)

5.12 UHPI Usage

There is an additional board that uses UHPI for program and data transfer via USB.

Features are:

- Download tool
- Communication tool in form of a DLL on the PC-side and C-source code on the DSP-side.

Check our web site www.dsp-weuffen.de for further information about this UHPI USB board.

5.13 Power Consumption

The AC-Adapter that is included can deliver a maximum power of 10W at 5V. This power is available for the board itself and for extension boards.

One TPS54310PWP is responsible for the 3.3V supply with a maximum power of 5W, the second TPS54310PWP is responsible for the 1.2V supply with a maximum power of 1.8W. The DSP uses 1.2W of the 1.2V supply, the RAM 0.5W of the 3.3V supply. For your own extensions you should therefore consider to use the 3.3V or the 5V supply.

6 Appendix

6.1 Abbreviations

CE	C hip E nable, used to select a chip e.g. flash
CPLD	C omplex P rogrammable L ogic D evice
CPU	C entral P rocessing U nit
dMAX	Dual Data Movement Accelerator
DSP	D igital S ignal P rocessor
EMIF	E xternal M emory I nterface
FIFO	F irst I n, F irst O ut
I ² C / I2C	Inter I C-Bus, serial bi-directional bus
IRAM	Internal R AM
IDE	Integrated D evelopment E nvironment
LED	Light E mitting D iode
McASP	M ultichannel A udio S erial P ort
PLL	P hase- L ocked L oop
ROM	R ead O nly M emory
RAM	R andom A ccess M emory
SDRAM	S ynchronous D ynamic R AM, data memory that has to be refreshed regularly
SPI	S erial P eripheral I nterface, SDI / SDO / SCLK / CE – master-slave use
TI	T exas I nstruments (DSP manufacturer)
(U)HPI	(U niversal) H ost P ort I nterface, fast interface for communication between processors

6.2 Part Placement

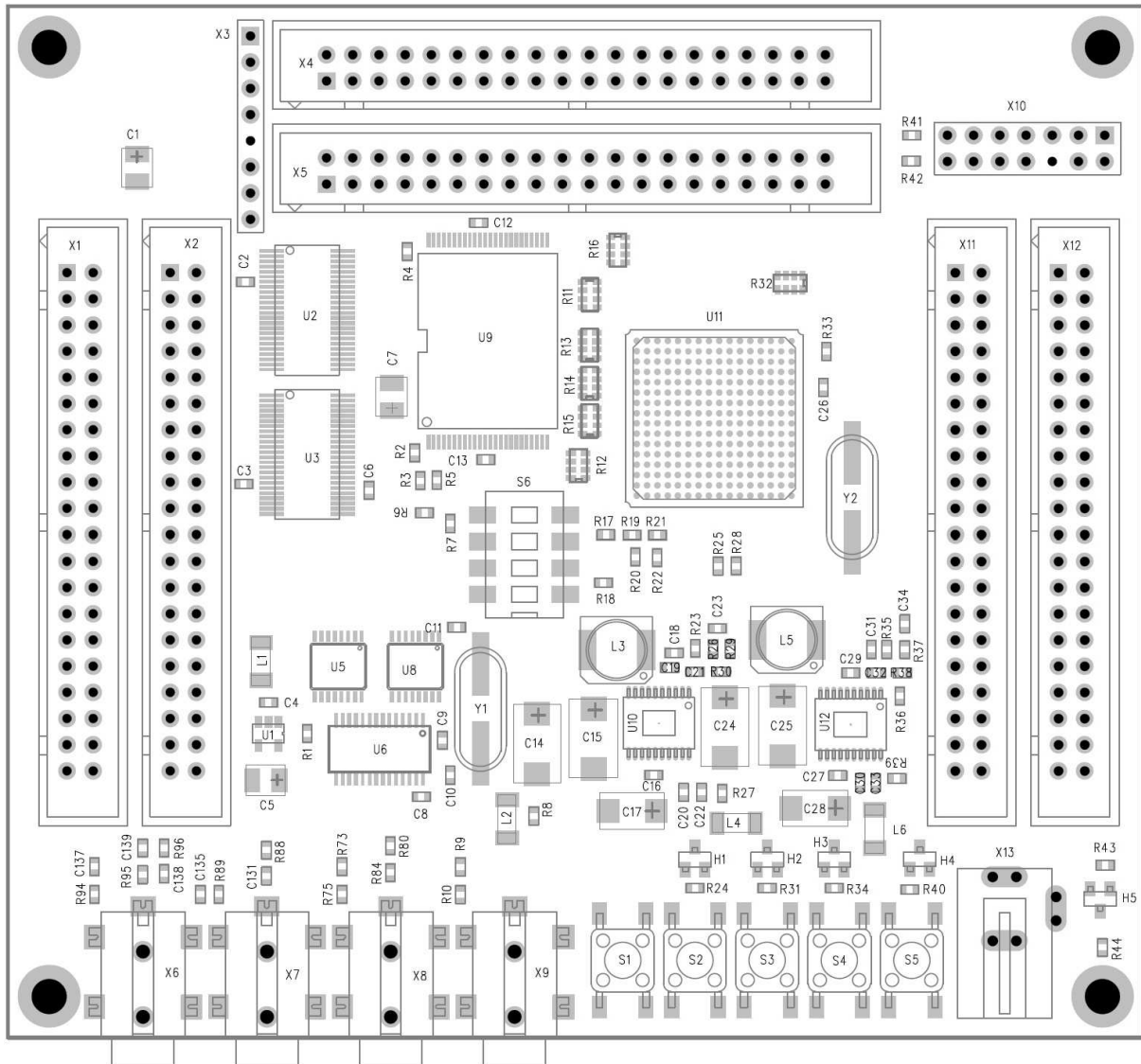


Figure 9 Part placing top view

C6727 EVM

Technical Reference v1.13 – 2007-07-27

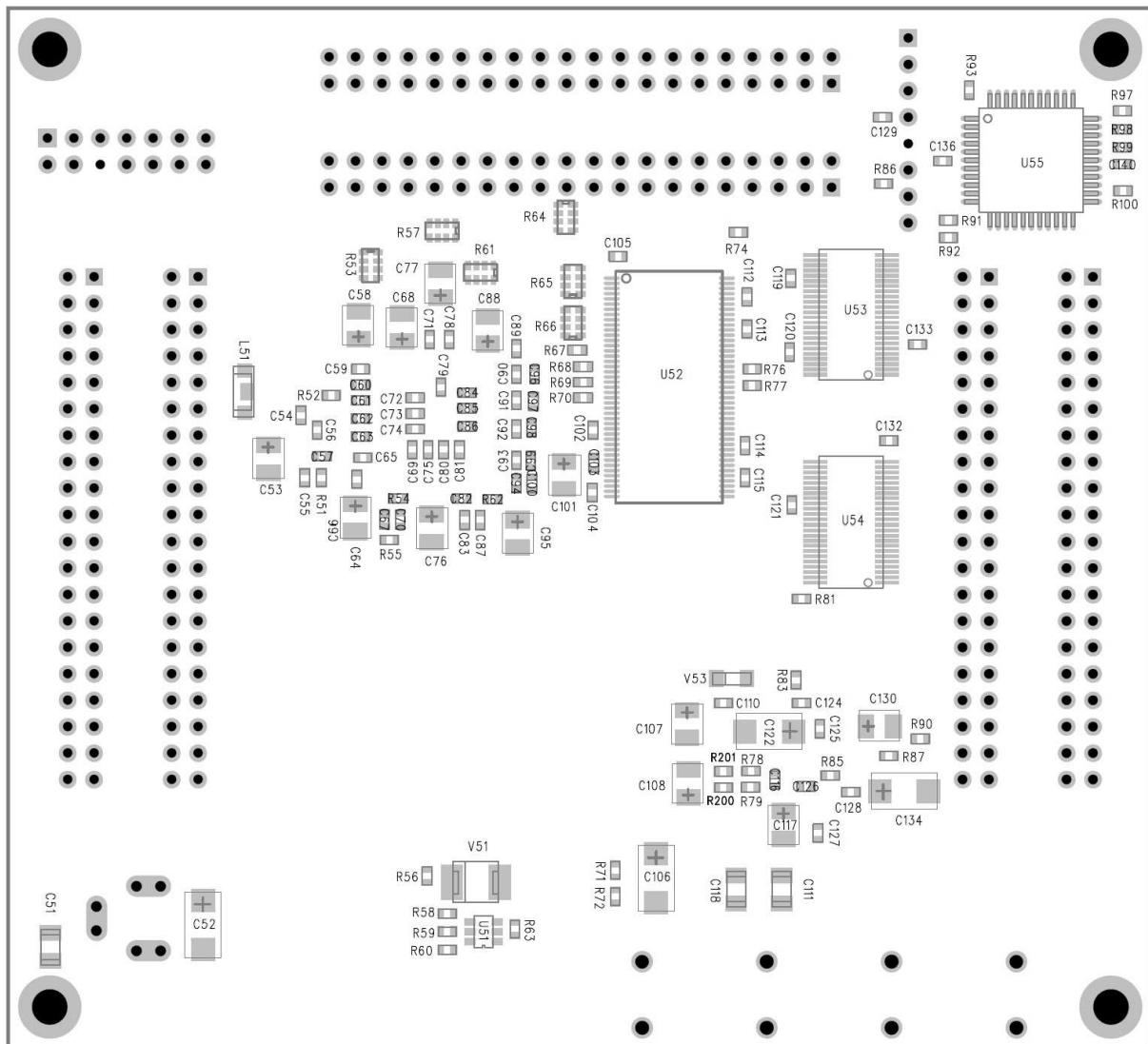


Figure 10 Part placing Bottom view

6.3 Connectors

The EVM C6727 has 13 connectors that provide the user access to the various signals on the board.

Connector	Function	# Pins	Form
Line-in	Stereo analog audio input	3	Jack socket, 3.5mm
Line-out	Stereo analog audio output	3	Jack socket, 3.5mm
Microphone	Monaural analog audio input	3	Jack socket, 3.5mm
Headphone	Stereo analog audio output	3	Jack socket, 3.5mm
X1	Bus Expansion Connector (EMIF)	40 (2x20)	Header, 2.54 mm grid
X2	Bus Expansion Connector (EMIF)	40 (2x20)	Header, 2.54 mm grid
X3	CPLD service interface (not intended for end user)	7	Header, 2.54 mm grid
X4	Host Port / GPIO Connector (for USB-UHPI daughter card)	40 (2x20)	Header, 2.54 mm grid
X5	Host Port / GPIO Connector (for USB-UHPI daughter card)	40 (2x20)	Header, 2.54 mm grid
X10	JTAG	13 + key (2x7)	Header, , 2.54 mm grid, already soldered
X11	Audio/Serial Peripheral Connector	40 (2x20)	Header, 2.54 mm grid
X12	Audio/Serial Peripheral Connector	40 (2x20)	Header, 2.54 mm grid
X13	Power supply	2	Type DC-10A, 6.3/2.1mm

Table 6 Connectors

6.3.1 Audio In/Out

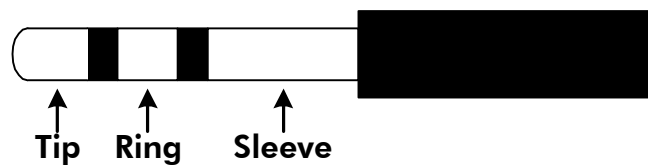


Figure 11 Jack 3.5mm

Tip	Microphone Bias
Ring	Microphone In
Sleeve	Ground

Table 7 Microphone

Tip	Left Channel
Ring	Right Channel
Sleeve	Ground

Table 8 Headphone, line-in, line-out

6.3.2 Power Supply Socket

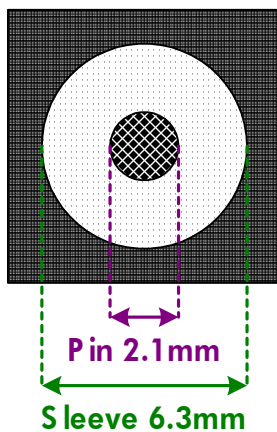


Figure 12 Power supply socket

Pin	+5V
Sleeve	Ground

Table 9 Power supply connector

6.3.3 JTAG

TMS	1	2	TRST*
TDI	3	4	GND
PD (+3.3V)	5	6	no pin
TDO	7	8	GND
TCKRET	9	10	GND
TCK	11	12	GND
EMU0*	13	14	EMU1*

Figure 13 Standard JTAG interface

11	TCK	Test Clock
1	TMS	Test Mode Select
3	TDI	Test Data In
7	TDO	Test Data Out
2	TRST*	Test Reset
13	EMU0*	Emulation Pin 0
14	EMU1*	Emulation Pin 1
9	TCKRET	Test Clock Return
5	PD (+3,3V)	Power Detect
4,8,10,12	GND	Signal Ground
6	no pin	orientation encoding

Table 10 JTAG port pins

6.3.4 External Connectors

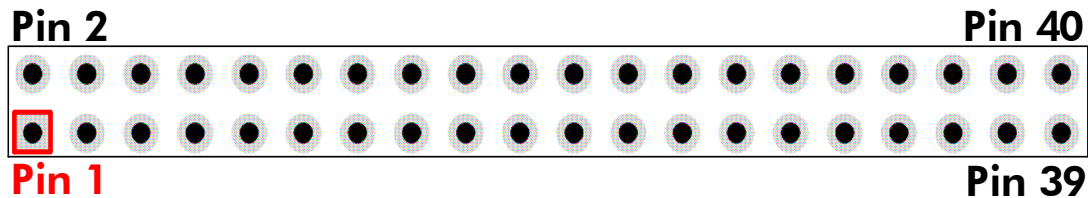


Figure 14 Pin numbering for 40 pin (2x20) Headers

Pin	Processor Signal	Circuitry Labeling	Description	Pin	Processor Signal	Circuitry Labeling	Description
1	+3.3V	-	+3.3 Volts	2	+3.3V	-	+3.3 Volts
3	EM_D22	B_D22	EMIF data bus pin 22	4	EM_D23	B_D23	EMIF data bus pin 23
5	EM_D20	B_D20	EMIF data bus pin 20	6	EM_D21	B_D21	EMIF data bus pin 21
7	EM_D18	B_D18	EMIF data bus pin 18	8	EM_D19	B_D19	EMIF data bus pin 19
9	EM_D16	B_D16	EMIF data bus pin 16	10	EM_D17	B_D17	EMIF data bus pin 17
11	GND	-	Ground	12	GND	-	Ground
13	EM_D14	B_D14	EMIF data bus pin 14	14	EM_D15	B_D15	EMIF data bus pin 15
15	EM_D12	B_D12	EMIF data bus pin 12	16	EM_D13	B_D13	EMIF data bus pin 13
17	EM_D10	B_D10	EMIF data bus pin 10	18	EM_D11	B_D11	EMIF data bus pin 11
19	EM_D8	B_D8	EMIF data bus pin 8	20	EM_D9	B_D9	EMIF data bus pin 9
21	GND	-	Ground	22	GND	-	Ground
23	EM_D6	B_D6	EMIF data bus pin 6	24	EM_D7	B_D7	EMIF data bus pin 7
25	EM_D4	B_D4	EMIF data bus pin 4	26	EM_D5	B_D5	EMIF data bus pin 5
27	EM_D2	B_D2	EMIF data bus pin 2	28	EM_D3	B_D3	EMIF data bus pin 3
29	EM_D0	B_D0	EMIF data bus pin 0	30	EM_D1	B_D1	EMIF data bus pin 1
31	GND	-	Ground	32	GND	-	Ground
33	EM_WE*	B_WE*	Active-low write enable	34	EM_RW*	B_RW*	Read/Write select pin
35	EM_WAIT	EM_WAIT	Wait input	36	EM_OE*	B_OE*	Active-low pin enable
37	CE2* ¹	B_CE2*	Chip enable 2	38	NC	-	Not connected
39	GND	-	Ground	40	GND	-	Ground

Table 11 Header X1 – Bus Interface Connector (EMIF)

¹ CE2* is set using UHPI_HD20, UHPI_HD21, UHPI_HD22, see “Table 1 Control signals for CPLD”

Pin	Process or Signal	Circuitry Labeling	Description	Pin	Process or Signal	Circuitry Labeling	Description
1	+5V	-	+5 Volts	2	+5V	-	+5 Volts
3	CE3* ²	B_CE3*	Chip enable 3	4	NC	-	Not connected
5	CE4* ²	B_CE4*	Chip enable 4	6	NC	-	Not connected
7	EM_A12	B_A12	EMIF address bus pin 12	8	NC	-	Not connected
9	EM_A10	B_A10	EMIF address bus pin 10	10	EM_A11	B_A11	EMIF address bus pin 11
11	GND	-	Ground	12	GND	-	Ground
13	EM_A8	B_A8	EMIF address bus pin 8	14	EM_A9	B_A9	EMIF address bus pin 9
15	EM_A6	B_A6	EMIF address bus pin 6	16	EM_A7	B_A7	EMIF address bus pin 7
17	EM_A4	B_A4	EMIF address bus pin 4	18	EM_A5	B_A5	EMIF address bus pin 5
19	EM_A2	B_A2	EMIF address bus pin 2	20	EM_A3	B_A3	EMIF address bus pin 3
21	+5V	-	+5 Volts	22	+5V	-	+5 Volts
23	EM_A0	B_A0	EMIF address bus pin 0	24	EM_A1	B_A1	EMIF address bus pin 1
25	EM_BA0	B_BA0	SDRAM Bank Address	26	EM_BA1	B_BA1	SDRAM Bank Address
27	EM_WE*_DQM2	B_DQM2	Write Enable or Byte Enable for EM_D[23:16]	28	EM_WE*_DQM3	B_DQM3	Write Enable or Byte Enable for EM_D[31:24]
29	EM_WE*_DQM0	B_DQM0	Write Enable or Byte Enable for EM_D[7:0]	30	EM_WE*_DQM1	B_DQM1	Write Enable or Byte Enable for EM_D[15:8]
31	GND	-	Ground	32	GND	-	Ground
33	EM_D30	B_D30	EMIF data bus pin 30	34	EM_D31	B_D31	EMIF data bus pin 31
35	EM_D28	B_D28	EMIF data bus pin 28	36	EM_D29	B_D29	EMIF data bus pin 29
37	EM_D26	B_D26	EMIF data bus pin 26	38	EM_D27	B_D27	EMIF data bus pin 27
39	EM_D24	B_D24	EMIF data bus pin 24	40	EM_D25	B_D25	EMIF data bus pin 25

Table 12 Header X2 – Bus Interface Connector (EMIF)

² CE3*/CE4* are set the same way than CE2*, see page 13 and footnote ¹

Pin	Processor Signal	Circuitry Labeling	Description	Pin	Processor Signal	Circuitry Labeling	Description
1	+5V	-	+5 Volts	2	+5V	-	+5 Volts
3	GND	-	Ground	4	NC	-	Not connected
5	NC	-	Not connected	6	GND	-	Ground
7	GND	-	Ground	8	UHPI_HD16	GA_15	UHPI Data Bus pin 16
9	NC	-	Not connected	10	UHPI_HD18	GA_17	UHPI Data Bus pin 18
11	UHPI_HD17	GA_16	UHPI Data Bus pin 17	12	UHPI_HD20	GA_19	UHPI Data Bus pin 20
13	UHPI_HD19	GA_18	UHPI Data Bus pin 19	14	UHPI_HD22	GA_21	UHPI Data Bus pin 22
15	UHPI_HD21	GA_20	UHPI Data Bus pin 21	16	GND	-	Ground
17	UHPI_HD23	HD23	UHPI Data Bus pin 23	18	NC	-	Not connected
19	NC	-	Not connected	20	UHPI_HD24	HD24	UHPI Data Bus pin 24
21	GND	-	Ground	22	UHPI_HD26	HD26	UHPI Data Bus pin 26
23	UHPI_HD25	HD25	UHPI Data Bus pin 25	24	UHPI_HD28	HD28	UHPI Data Bus pin 28
25	UHPI_HD27	HD27	UHPI Data Bus pin 27	26	UHPI_HD30	HD30	UHPI Data Bus pin 30
27	UHPI_HD29	HD29	UHPI Data Bus pin 29	28	NC	-	Not connected
29	UHPI_HD31	HD31	UHPI Data Bus pin 31	30	GND	-	Ground
31	GND	-	Ground	32	NC	-	Not connected
33	NC	-	Not connected	34	GND	-	Ground
35	GND	-	Ground	36	+3.3V	-	+3.3 Volts
37	NC	-	Not connected	38	GND	-	Ground
39	GND	-	Ground	40	+3.3V	-	+3.3 Volts

Table 13 Header X4 – Host Port / GPIO Connector

Pin	Processor Signal	Circuitry Labeling	Description	Pin	Processor Signal	Circuitry Labeling	Description
1	UHPI_HBE3*	HBE3n	UHPI byte enable 3	2	UHPI_HBE2*	HBE2n	UHPI byte enable 2
3	UHPI_HRDY*	HRDYn	UHPI ready output	4	-	RES_HPI_IN*	External reset in
5	UHPI_HBE1*	HBE1n	UHPI byte enable 1	6	UHPI_HBE0*	HBE0n	UHPI byte enable 0
7	+3.3V	-	+3.3 Volts	8	+5V	-	+5 Volts
9	UHPI_HD1	HD1	UHPI Data Bus pin 1	10	AMUTE2/ HINT*	AMUTE2_ _HOST INTn	Audio mute 2/ UHPI host interrupt
11	UHPI_HD3	HD3	UHPI Data Bus pin 3	12	UHPI_HD0	HD0	UHPI Data Bus pin 0
13	UHPI_HD5	HD5	UHPI Data Bus pin 5	14	UHPI_HD2	HD2	UHPI Data Bus pin 2
15	UHPI_HD7	HD7	UHPI Data Bus pin 7	16	UHPI_HD4	HD4	UHPI Data Bus pin 4
17	UHPI_HRW*	HRWn	UHPI read/write	18	UHPI_HD6	HD6	UHPI Data Bus pin 6
19	UHPI_HD8	HD8	UHPI Data Bus pin 8	20	GND	-	Ground
21	UHPI_HD10	HD10	UHPI Data Bus pin 10	22	UHPI_HD9	HD9	UHPI Data Bus pin 9
23	UHPI_HD12	HD12	UHPI Data Bus pin 12	24	UHPI_HD11	HD11	UHPI Data Bus pin 11
25	UHPI_HD14	HD14	UHPI Data Bus pin 14	26	UHPI_HD13	HD13	UHPI Data Bus pin 13
27	GND	-	Ground	28	UHPI_HD15	HD15	UHPI Data Bus pin 15
29	UHPI_HDS2*	HDS2n	UHPI data strobe 2	30	GND	-	Ground
31	GND	-	Ground	32	UHPI_HAS*	HASn	UHPI address strobe
33	UHPI_HDS1*	HDS1n	UHPI data strobe 1	34	GND	-	Ground
35	GND	-	Ground	36	UHPI_HCNTL0	HCNTL0	UHPI control 0
37	UHPI_HCS*	HCSn	UHPI chip select	38	GND	-	Ground
39	GND	-	Ground	40	UHPI_ HCNTL1	HCNTL1	UHPI control 1

Table 14 Header X5 – Host Port / GPIO Connector

Pin	Processor Signal	Circuitry Labeling	Description	Pin	Processor Signal	Circuitry Labeling	Description
1	GND	-	Ground	2	GND	-	Ground
3	+5V	-	+5 Volts	4	+5V	-	+5 Volts
5	GND	-	Ground	6	GND	-	Ground
7	+5V	-	+5 Volts	8	+5V	-	+5 Volts
9	+5V	-	+5 Volts	10	+5V	-	+5 Volts
11	+5V	-	+5 Volts	12	+5V	-	+5 Volts
13	+3.3V	-	+3.3 Volts	14	+3.3V	-	+3.3 Volts
15	+3.3V	-	+3.3 Volts	16	+3.3V	-	+3.3 Volts
17	+3.3V	-	+3.3 Volts	18	+3.3V	-	+3.3 Volts
19	AHCLKR0 AHCLKR1	AHCLKR0A HCLKR1	McASP0 and McASP1 Receive Master Clock	20	ACLKR0	ACLKR0	McASP0 Receive Bit Clock
21	AHCLKX0A HCLKX2	AHCLKX0A HCLKX2	McASP0 and McASP2 Transmit Master Clock	22	AFSR0	AFSR0	McASP0 Receive Frame Sync
23	GND	-	Ground	24	GND	-	Ground
25	ACLKX0	ACLKX0	McASP0 Transmit Bit Clock	26	AFSX0	AFSX0	McASP0 Transmit Frame Sync
27	AMUTE0	AMUTE0_M	McASP0 MUTE Output	28	AXR00	AXR00	McASP0 Serial Data 0
29	GND	-	Ground	30	GND	-	Ground
31	AXR01	AXR01	McASP0 Serial Data 1	32	AXR02	AXR02	McASP0 Serial Data 2
33	AXR03	AXR03	McASP0 Serial Data 3	34	AXR04	AXR04	McASP0 Serial Data 4
35	GND	-	Ground	36	AXR06 SPI1_ENA*	AXR06 SPI1_E NA*	McASP0 Serial Data 6 or SPI1 Enable
37	AXR05 SPI1_SCS*	AXR05 SPI1_SCS*	McASP0 Serial Data 5 or SPI1 Slave Chip Select	38	AXR08 AXR15 SPI1_SOMI	AXR08 AXR15 SPI1_S OMI	McASP0 Serial Data 8 or McASP1 Serial Data 5 or SPI1 Data Pin Slave Out Master In
39	AXR07 SPI1_CLK	AXR07 SPI1_CLK	McASP0 Serial Data 7 or SPI1 Serial Clock	40	AXR09 AXR14 SPI1_SIMO	AXR09 AXR14 SPI1_SI MO	McASP0 Serial Data 9 or McASP1 Serial Data 4 or SPI1 Data Pin Slave In Master Out

Table 15 Header X11 – Audio/Serial Peripheral Connector

Pin	Processor Signal	Circuitry Labeling	Description	Pin	Processor Signal	Circuitry Labeling	Description
1	GND	-	Ground	2	GND	-	Ground
3	NC	-	Not connected	4	AXR010 AXR13	AXR010 AXR13	McASP0 Serial Data 10 or McASP1 Serial Data 3
5	AXR011 AXR12	AXR011 AXR12	McASP0 Serial Data 11 or McASP1 Serial Data 2	6	AXR012 AXR11	AXR012 AXR11_M	McASP0 Serial Data 12 or McASP1 Serial Data 1
7	AXR013 AXR10	AXR013 AXR10_M	McASP0 Serial Data 13 or McASP1 Serial Data	8	AXR014 AXR21	AXR014 AXR21	McASP0 Serial Data 14 or McASP2 Serial Data 1
9	GND	-	Ground	10	GND	-	Ground
11	AXR015 AXR20	AXR015 AXR20	McASP0 Serial Data 15 or McASP2 Serial Data 0	12	SPI0_SOM I2C0_SDA	SPI0_SOM I2C0_SDA	SPI0 Data Pin Slave Out Master In or I2C0 Serial Data
13	SPI0_SIM O	SPI0_SIMO	SPI0 Data Pin Slave In Master Out	14	SPI0_SCS * I2C1_SCL	SPI0_SCS* I2C1_SCL_M	SPI0 Slave Chip Select or I2C1 Serial Clock
15	SPI0_ENA * I2C1_SDA	SPI0_ENA* I2C1_SDA_M	SPI0 Enable (Ready) or I2C1 Serial Data	16	SPI0_CLK I2C0_SCL	SPI0_CLK I2C0_SCL	SPI0 Serial Clock or I2C0 Serial Clock
17	RESET*	RES*		18	NC	-	Not connected
19	GND	-	Ground	20	GND	-	Ground
21	ACLKR1	ACLKR1_M	McASP1 Receive Bit Clock	22	AFSR1	AFSR1_M	McASP1 Receive Frame Sync
23	AHCLKX1	AHCLKX1	McASP1 Transmit Master Clock	24	ACLKX1	ACLKX1_M	McASP1 Transmit Bit Clock
25	AFSX1	AFSX1_M	McASP1 Transmit Frame S	26	AMUTE1	AMUTE1_M	McASP1 MUTE Output
27	AHCLKR2	AHCLKR2_M	McASP2 Receive Bit Clock	28	ACLKR2	ACLKR2_M	McASP2 Receive Bit Clock
29	AFSR2	AFSR2_M	McASP2 Receive Frame Sync	30	ACLKX2	ACLKX2_M	McASP2 Transmit Bit Clock
31	AFSX2	AFSX2_M	McASP2 Transmit Frame Sync	32	AMUTE2 HOSTINTn	AMUTE2 HOSTINTn	McASP2 MUTE Output or UHPI Host Interrupt
33	NC	-	Not connected	34	NC	-	Not connected
35	GND	-	Ground	36	GND	-	Ground
37	GND	-	Ground	38	GND	-	Ground
39	NC	-	Not connected	40	NC	-	Not connected

Table 16 Header X12 – Audio/Serial Peripheral Connector

6.4 Errata

Known since document v1.08:

- Inverted signals in the schematics are named inconsistently, e.g. HDS1 is called HDS1n instead of HDS1*.
- Schematics qualifier AMUTE2_HOSTINTn should be called AMUTE2/HOSTINT*.

Known since document v1.09:

- Schematics part list mentions *Kingbright* LED KM-23ID which should in fact be KM-23ID-F.
- The samples “CODEC” and “MEMTEST” failed in release build due to missing defines and paths. This is fixed in latest customer support updates from the website.

Known since document v1.11:

- Pinout of header X5 described a pre-production state. It now reflects board revision 1.1.

***** End of document *****