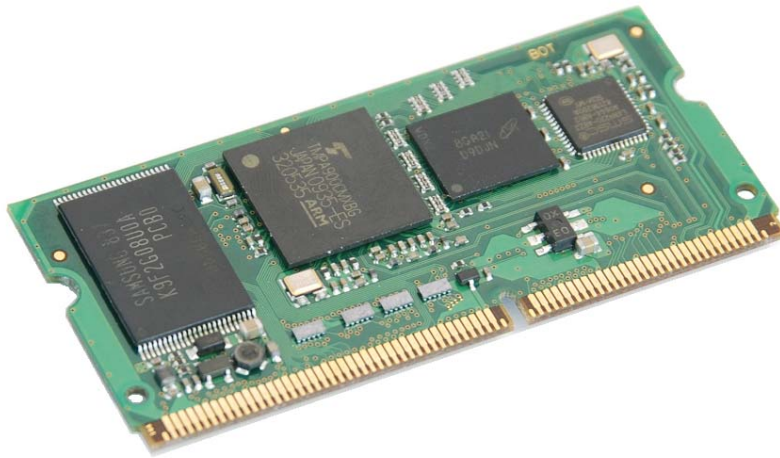




Developer Manual TMPA900 CPU Board



2010 GLYN GmbH & Co. KG

All rights reserved. No part of this documentation may be reproduced or, with the use of electronic systems, edited copied or transmitted, in any form (print, photocopy, microfilm or another procedure) without the express authority of the GLYN GmbH & Co. KG, D-65510 Idstein.

The GLYN GmbH & Co. KG, D-65510 Idstein does not accept liability or provide any guarantee with respect to the contents of this documentation. The GLYN GmbH & Co. KG, D-65510 Idstein retains the right of revising this work. All programs and descriptions have been created to the best of our knowledge and tested with great care. However, errors cannot be entirely excluded. For this reason, the GLYN GmbH & Co. KG does not accept liability for possible errors and consequential damage resulting from the provision, performance or use of this material.

Table of Contents

1.0 Revision List	4
1.1 Contacts	4
2.0 Summary	5
2.1 Product Information TMPA900-CPU BOARD	5
2.2 Energy Consumption at 3.3V (preliminary values).....	6
2.3 Block Diagram	6
2.4 TMPA900 CPU Board Reference Circuitry	7
2.5 Glyn’s Graphic Base Board	8
2.5.1 Glyn’s Graphic Base Board Connectors.....	9
2.6 Instructions for use	11
3.0 Operational Description	12
3.1 CPU Core	12
3.2 Memory – Architecture.....	13
3.2.1 Nand Flash Memory – Unique Characteristics	14
3.3 TMPA900CMXBG Block Diagram with Multilayer AHB	16
3.4 Power Supply	17
3.5 RESET.....	19
3.6 External power source control output	19
3.7 Ethernet - Network Controller	20
3.8 UARTs.....	21
3.9 USB 2.0 - Device	22
3.10 USB 2.0 - HOST	23
3.11 I2C	24
3.12 SPI (SSP)	25
3.13 I2S (Inter-IC Sound).....	26
3.14 PWM (Pulse Width Modulation) / 16bit-Timers	27
3.15 JTAG	28
3.16 Keys / Keyboard	29
3.17 Analog/Digital Converter.....	29
3.18 Touch Screen Interface (TSI)	30
3.19 LCD Controller (LCDC).....	31
3.20 Glyn Graphic Base Board & Glyn TFT Family Concept.....	33
3.21 SD Host Controller.....	35
3.22 CMOS Camera Interface	36
3.23 Melody/Alarm Generator.....	37
3.24 Low Frequency Clock Output	37
4.0 Pin Allocation SODIMM 144 Connector.....	38
5.0 Software Components	44
5.1 Basics - Data Transfer to TMPA900 CPU Board	44
5.1.1 ELDIO Download Wizard.....	44
5.1.2 Basics – Installing J-Link Lite.....	46
5.1.3 Basics - Installing a TFTP Server	46

5.1.4 Basics - Working completely under Linux	46
5.2 u-boot	47
5.2.1 The Boot Process	48
5.2.2 Flashing the u-boot	49
5.2.2.1 Flashing the u-boot over JTAG	49
5.2.2.2 Update the u-boot via network (handle with care)	50
5.2.2.3 U-boot - Environment Setup	51
5.2.2.4 IP and MAC Address Setup	51
5.2.2.5 Configuration of the Display Parameters	52
5.2.2.6 Configuration of the File System Type	52
5.2.2.7 Splash Screen Support	52
5.2.2.8 Erase u-boot Environment	53
5.2.2.9 u-boot - NFS Server Setup	54
5.2.2.10 More u-boot commands	54
5.2.2.11 What to do if the boot loader has been flashed incorrectly	56
5.3 Standard Application (IAR Compiler)	58
5.3.1 Debugging the Application (IAR Compiler)	58
5.3.2 Make a Release for Flash (IAR Compiler)	58
5.3.3 Flashing the Application (No Linux)	58
5.3.4 Getting Started with SEGGER Evaluation Software and IAR	59
6.0 Linux for TMPA900 CPU board	61
6.1 Major Components of a Linux System	62
6.2 Flashing the Linux Application	63
6.3 Flash Layout TMPA900-CPU-BOARD	63
6.4 Installation Linux Tool chain TMPA900 CPU board	64
6.5 Linux Kernel Build	64
6.5.1 Linux Kernel Source Tree	64
6.5.2 Linux Kernel Configuration	66
6.5.3 Compiling the Linux Kernel	68
6.5.4 Installing the Linux Kernel	68
6.6 Linux File System	69
6.7 Small C-Examples under Linux	71
6.7.1 Linux "Hello World" Example	71
6.7.2 IO-Toggle – Example for an easy accesses to the peripherals	72
6.8 µCross – Linux Tool Package	74
7.0 Installing the Display with the Glyn Graphic Base Board	77
7.1 Other Resolutions/Other Timings – Calculation of the Display Settings	78
8.0 Mechanical Specifications (Formating)	79
8.1 Soldering the TMPA900-CPU-Board – No Connector	80
Appendix A: Available u-boot Commands	81
Appendix B: Ordering Information	83
Appendix C: KC Labs Public Git Server	84
Appendix D: Literature and References	87
Appendix E: CD file directory tree	89
Appendix F: Contact Information	90

1.0 Revision List

V0.1	30.12.2009	CTE/OLE	Document compilation
V0.2	13.01.2010	CTE/OLE	Revision Pin – Allocation
V0.3	18.02.2010	OLE	Revision NAND FLASH
V0.4	07.04.2010	OLE	Manual for flashing the u-boot
V0.5	08.04.2010	OLE	Manual for flashing the application
V1.0	09.04.2010	CTE / HFR	Proof-reading
V1.1	14.04.2010	OLE	Bootloader Correction
V1.2	21.04.2010	OLE	Correction Installation Linux Image
V1.3	18.05.2010	OLE	Corrections
V1.4	21.06.2010	CTE	New chapters 5.0, 6.5 and 7.1
V1.5	05.08.2010	CTE	Completely revised manual
V1.6	18.05.2010	CTE	New ELDIO Download Wizard
V1.7	26.01.2011	CTE	Some Corrections, new chap. 6.7.2

1.1 Contacts

Christoph Tenbergen
Dominik Peuker

christoph.tenbergen@glyn.de
dominik.peuker@glyn.de

+49 2157 127-227
+49 6126 590-270

2.0 Summary

This report contains all important technical information regarding the TMPA900-CPU-board SODIMM module. Detailed information about the implemented parts can be found on the appropriate data sheets and a list with references is at the end of the document.

2.1 Product Information TMPA900-CPU BOARD

Glyn's TMPA900-CPU board is a CPU module by Glyn GmbH & Co KG with Toshiba's TMPA900CMXBG ARM9 microcontroller. The integrated TFT controller enables RGB display driving with a resolution of 800x480 with integrated hardware acceleration. Dispensing with this acceleration enables up to 1024x1024. The module is a highly efficient and easy-to-integrate processor platform with graphic and video capability. The board is designed for mounting on a SODIMM socket which is also available from us – it is a SODIMM 144-pin socket used in the PC field. Additionally, it is also possible to dispense with the socket and solder the module for larger series sizes (further information available).

System On-Module

- Processor TMPA900CMXBG, 200 MHz
- RAM 64 MB DDRRAM
- ROM 256 MB NAND Flash
- Power supply single 3.0V to 3.6V
- Size SO-DIMM 144
- Temp.-range -20°C..85°C

Key Features

- 10/100Mbps Ethernet (MAC+PHY)
- High-speed USB 2.0 Device (480Mbps)
- Full-speed USB Host 2.0 (12Mbps)
- LCD controller
- CMOS camera interface
- Interfaces: UART, SD-CARD, I2C, PWM, Keypad, Digital Audio (I2S), 4/5 wire touch screen

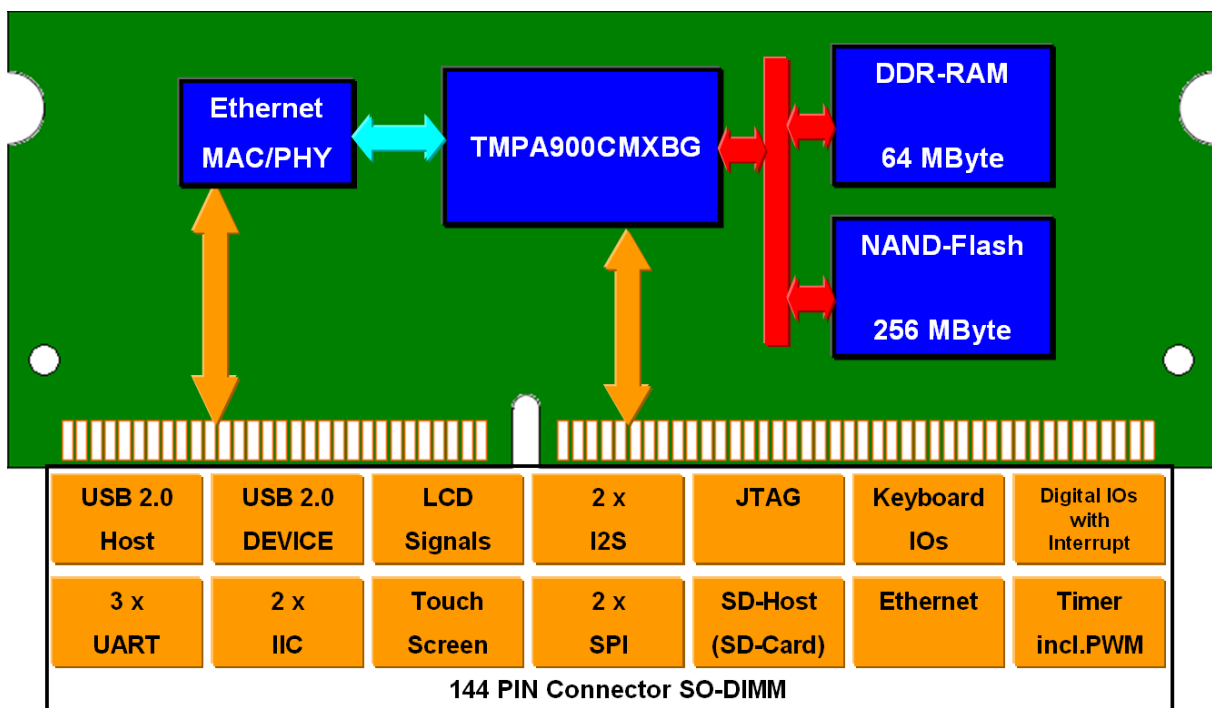
The TMPA900-CPU-Board what comes with the Starterkit is pre-programmed with the UBoot and a Splashscreen.

There is no Linux-Kernel or other application programmed!

2.2 Energy Consumption at 3.3V (preliminary values)

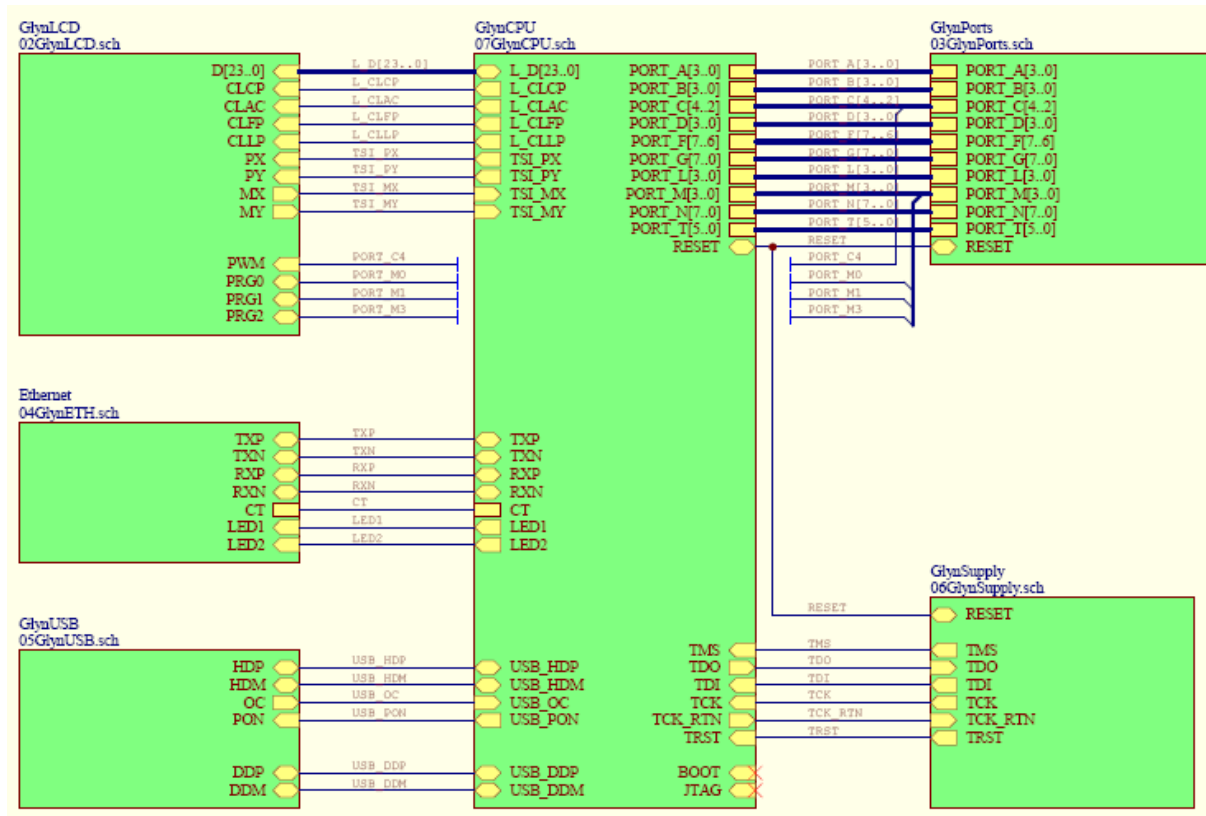
f (Fc = 200 Mhz)	mit ethernet		ohne Ethernet (Initialisierung nicht vollständig)	
	I/mA	P/mW	I/mA	P/mW
fc	328	1082	174	574
fc/2	267	881	137	452
fc/4	239	789	119	393
fc/8	225	743	110	363
nach reset (halt)	190	627	75	248

2.3 Block Diagram



2.4 TMPA900 CPU Board Reference Circuitry

The complete reference circuitry can be found on the CD.

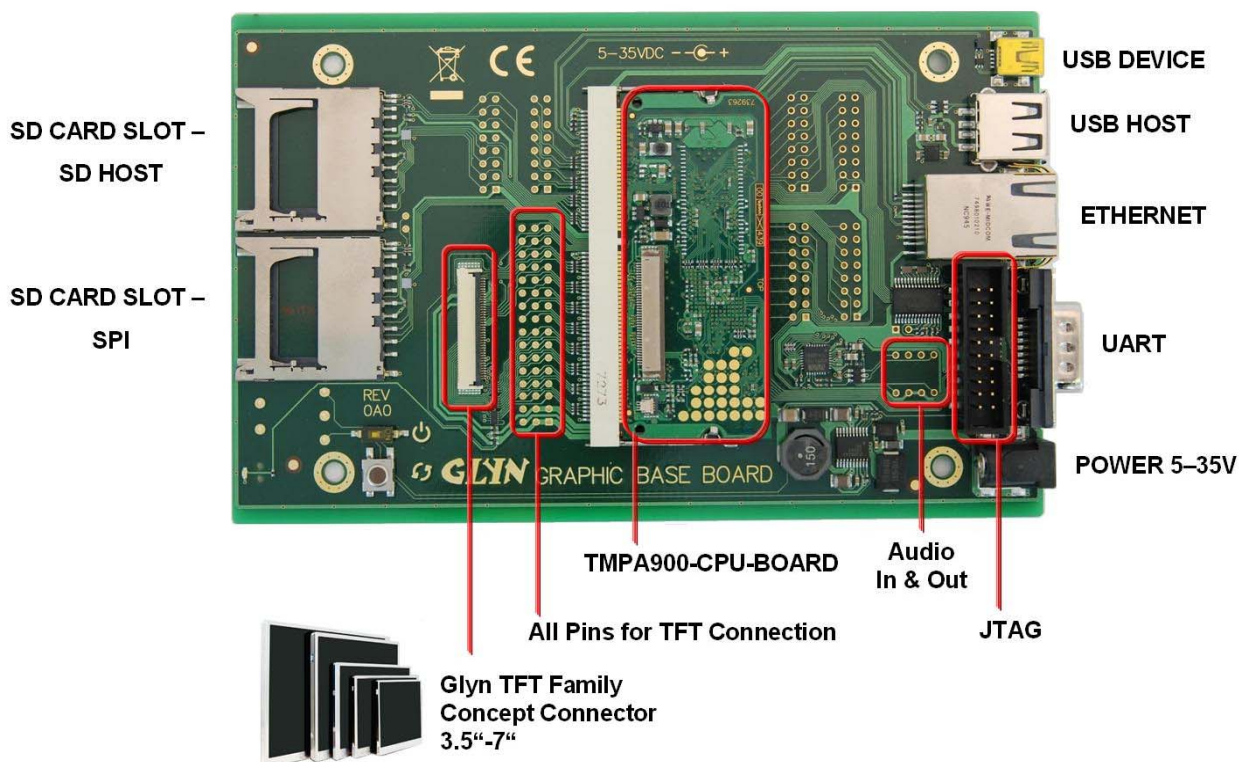


2.5 Glyn´s Graphic Base Board

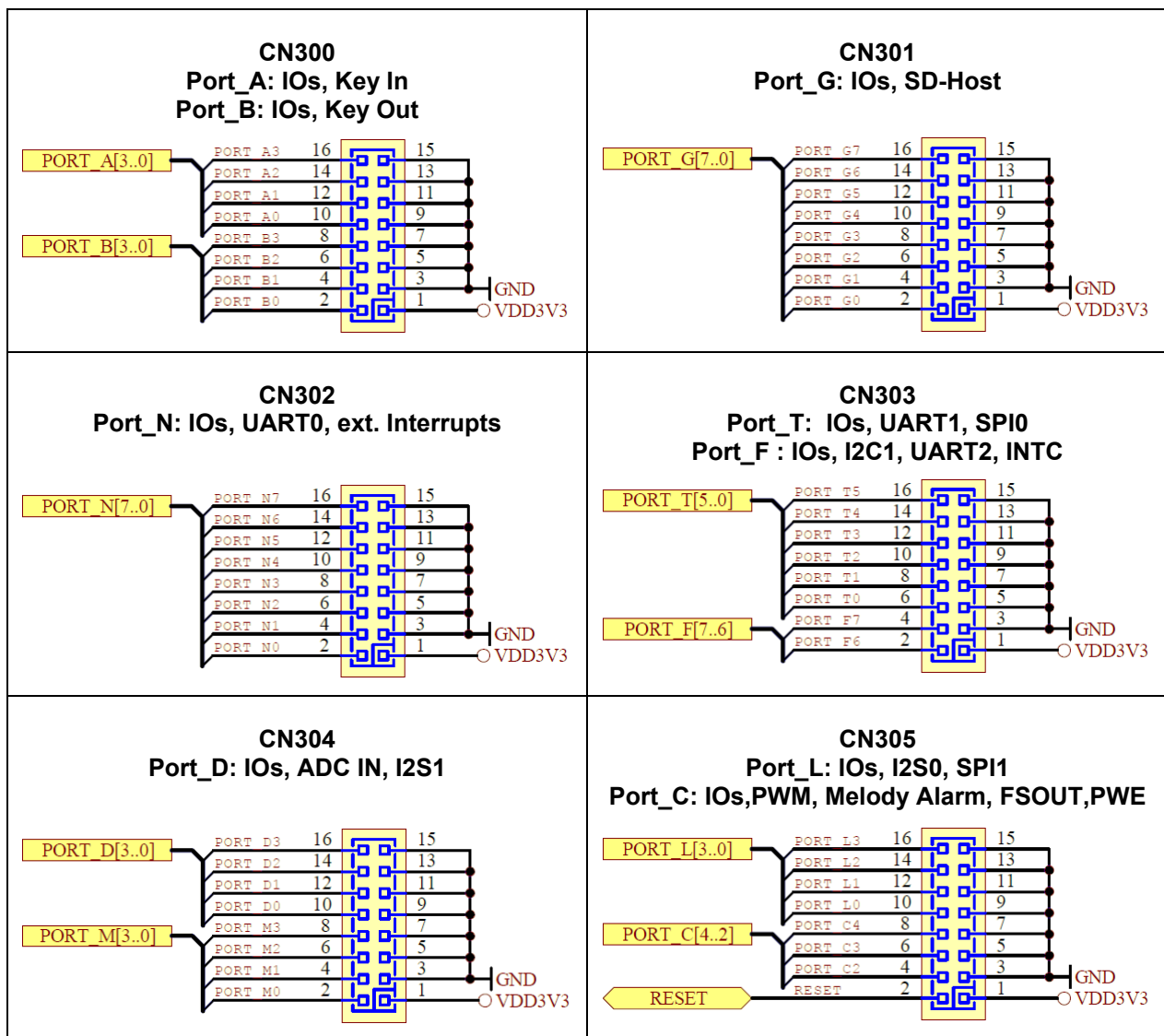
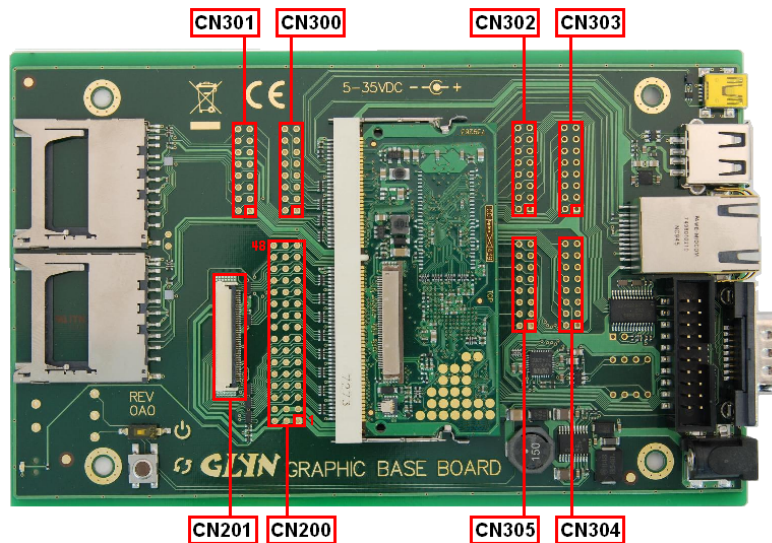
Glyn´s Graphic Base Board is a platform in order to use the TMPA900-CPU board for a development. At the same time, it is also the reference design for including a TMPA900-CPU board. The plans and BOM list can be found on the CD in the file „Circuit_Diagramm“. Should layout data be required, this can also be provided against an NDA.

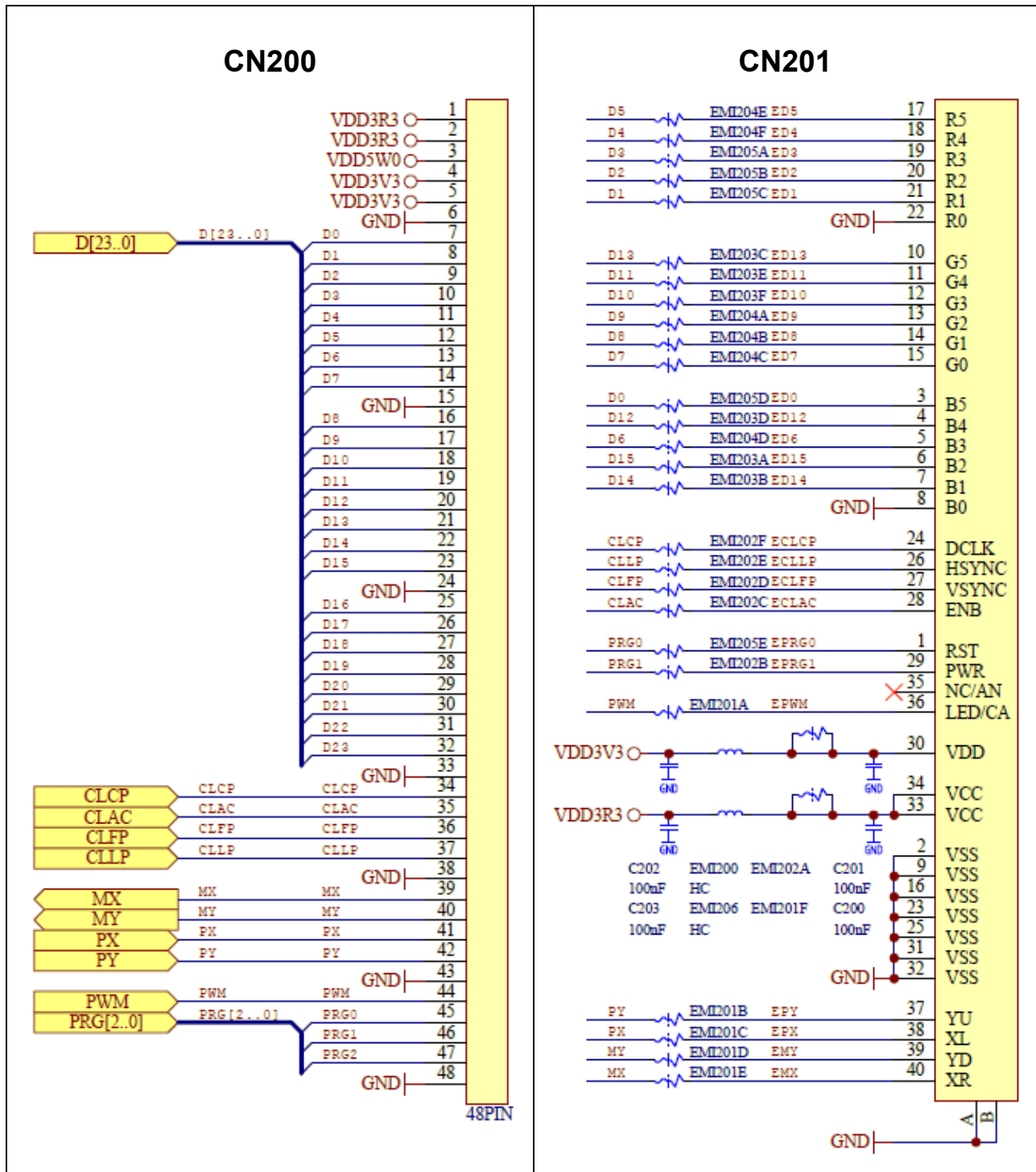
Features

- 144 PIN SODIMM connector
- Ethernet connector
- USB host connector
- USB device connector
- RS232
- WM8983 Audio Codec by Wolfson
- SD card sockets (SD-Host controller and via SPI)
- Glyn TFT concept connector für 3.5" – 7" TFTs
- JTAG interface
- 100mm x 160mm
- Single power supply 5-35V



2.5.1 Glyn's Graphic Base Board Connectors





2.6 Instructions for use

The standard measures of precaution regarding touching and operating circuitry in low voltage ranges apply. Electrostatic discharging which may damage parts is to be avoided.

The board may be plugged in or out only when the supply voltage is switched off.

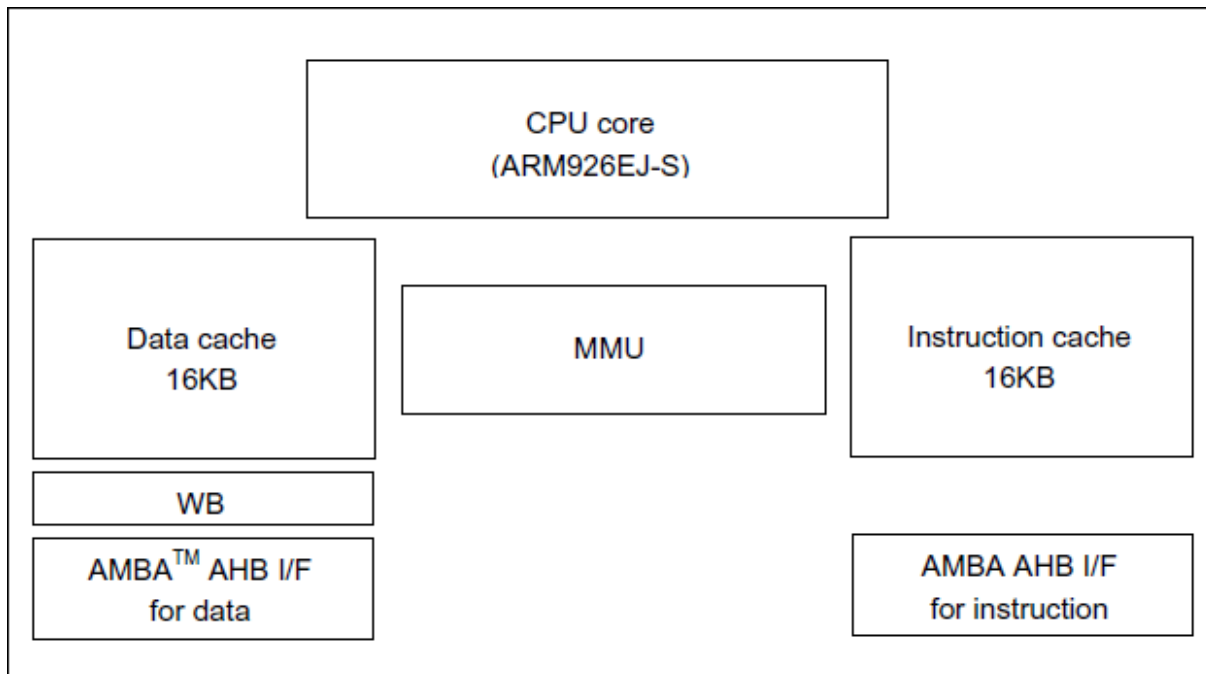
The parts used on the TMPA900 CPU board are specified for use between -20 and +85 Grad Celsius. The TMPA900 CPU board requires a DC voltage of $3.3V \pm 0.2V$.

3.0 Operational Description

This chapter provides a brief description of the module and its interfaces.

3.1 CPU Core

Note that this document provides only an overview of the CPU block. Please contact ARM Holdings for operation details and refer to the TMPA900CMXBG manual. The TMPA900CM has a built-in 32-bit RISC processor ARM926EJ-S manufactured by ARM. The schematic diagram of the ARM926EJ-S core is shown below.



The TMPA900CM does not feature the functions shown below.

1. Coprocessor I/F
2. Embedded ICE RT
3. TCM I/F
4. ETM9TM I/F

3.2 Memory – Architecture

The TMPA900 is characterised by a multilayer AHB bus. The advantage over conventional architectures is the higher internal data throughput. To expand on this concept, there are two memory controllers for the external flash and RAM. The first is responsible for communication with NORFLASH, SRAM or SDRAM and the second is responsible for communication with NORFLASH, SRAM und DDR SDRAM. The external memory chips are each directly connected to one of these controllers.

The CSs of the external NAND flash are connected to pin D7/D8 on the TMPA900.
The CSs of the external SRAM/DDRRAM are connected to pin K12 on the TMPA900

The TMPA900 has two operating modes – the external memory mode and the internal boot ROM mode which are specified by the external mode pins AM0 and AM1. Pin AM0 is set to high. Pin AM1 is connected to the expansion connector and marked BOOT (PIN132).

AM1	HIGH	BOOT (start from internal Boot ROM)
AM1	LOW	Start from external bus/memory (16-bit Bus)

Due to the internal structure of NAND memories, it is not possible to run a program directly from this memory. In fact, the user program code has to be copied from the NAND memory to the external RAM before starting the program. Access to the NAND flash takes place in a multiplexed 8-Bit mode.

The process has not been disclosed in detail. Should the customer require more information, we can request personalised documentation at Toshiba. This is free of charge.

3.2.1 Nand Flash Memory – Unique Characteristics

- When it is erased, all bits are set to '1' (you will see 0xff on all bytes in a hexdump)
- You can change as many bits as you want to '0'
- You cannot set a bit back to '1' by using a regular write.
- You have to erase a whole erase block to do so
- The number of erase cycles per block is limited. Once you have reached the limit, some bits will not get back to 0xff. In the case of the in Samsung Flash K9F2G08 this is 100.000 guaranteed per-block erase cycles.

NAND page

A NAND page consists of a number of data bytes plus a number of out-of-band (OOB) bytes.

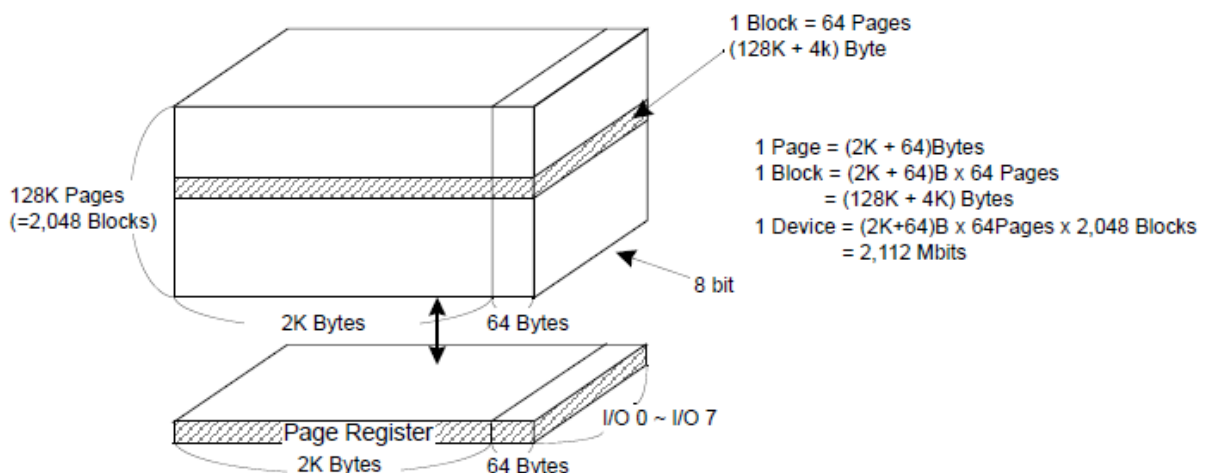
Only the data bytes are used for application data. The OOB bytes are used for

- Marking an erase block as bad (first or second page of erase block)
- Storing ECC (error correction codes)
- Storing file system specific information (JFFS2)

NAND erase block

An erase block consists of multiple pages. In K9F2G08 every erase block has 64 pages.

K9F2G08X0A Array Organization



Problem: Bad Blocks

NAND memory apparently gets shipped with blocks that are already bad. The vendor just marks those blocks as bad, thus resulting in higher yield and lower per-unit cost.

The flash contains four kinds of blocks (16kBytes):

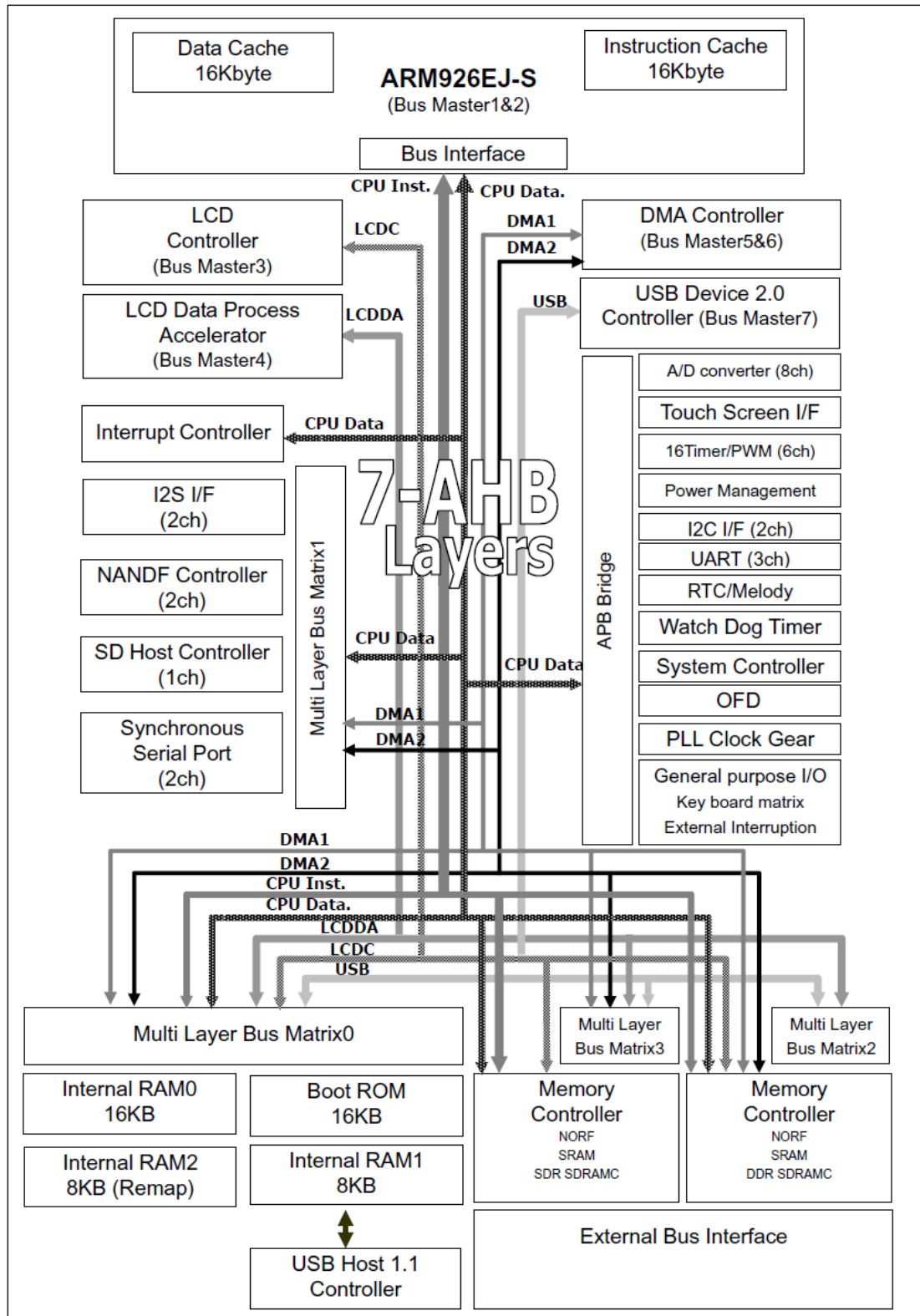
- Factory default bad blocks
 - Samsung marks the 6th OOB byte as non 0xFF in the first and/or second page in blocks that are bad
- Worn-out bad blocks
- Good blocks
- The first block. This block is guaranteed to not require error correction up to 1000 writes. This is needed as the initial boot code can't do ECC.

It is also guaranteed that a minimum of 2008 blocks (out of the total 2048) are good. This means up to 40 blocks (5.1MBytes) can be dead.

More about bad block handling in chapter 5.0 u-boot!

3.3 TMPA900CMXBG Block Diagram with Multilayer AHB

The TMP900CM uses a multilayer AHB bus system with 7 layers.

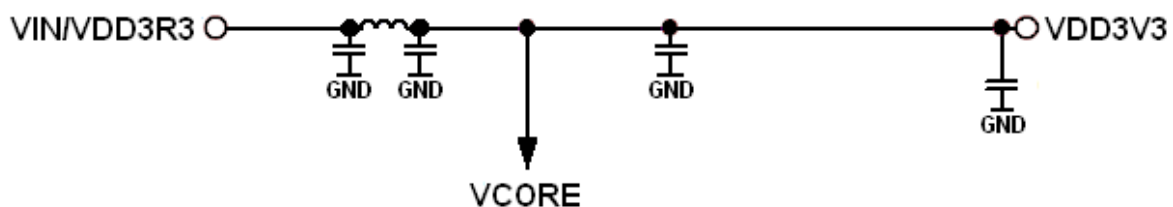


3.4 Power Supply

The parts used on the TMPA900 CPU board are specified for use between -20 and +85 Grad Celsius. The TMPA900 CPU board requires a DC voltage of $3.3V \pm 0.2V$.

The power is fed over the VDD3R3 pins. This power runs through some filters and is outputted over VDD3V3. External components which aren't supplied by the module should be supplied over VDD3V3.

The filters are switched between VIN/VDD3R3 and VDD3V3, so the voltage peak has the same effect on external components as on the module. The EMV critical part should be connected via the filters.



Power Supply Pins VDD3R3:

Signal	PIN	Input/Output	Function Remarks
VDD3R3	136	Input	DC-IN supply: $3.3V \pm 0.2V$
VDD3R3	138	Input	
VDD3R3	140	Input	
VDD3R3	142	Input	

Power Output Pins VDD3V3:

Signal	PIN	Input/Output	Function Remarks
VDD3V3	3	Output	DC-OUT VDD3R3 filtered
VDD3V3	4	Output	
VDD3V3	61	Output	
VDD3V3	62	Output	
VDD5W0	5	-	Not connected!

Ground Pins:

Signal	PIN	Input/Output	Function Remarks
GND	1		Ground
GND	2		
GND	11		
GND	12		
GND	21		
GND	22		
GND	31		
GND	32		
GND	41		
GND	42		
GND	51		
GND	52		
GND	63		
GND	64		
GND	73		
GND	74		
GND	83		
GND	84		
GND	93		
GND	94		
GND	103		
GND	104		
GND	113		
GND	114		
GND	123		
GND	124		
GND	133		
GND	134		
GND	143		
GND	144		

3.5 RESET

Before resetting the TMPA900CM, make sure that the power supply voltage is within the operating range, oscillation from the internal oscillator is stable at at least 20 system clock cycles (0.8 μ s @ X1 = 25 MHz), and the RESET input pin is pulled Low. When the TMPA900CM is reset, the PLL stops, the PLL output is unselected, and the clock gear is set to TOP (1/1).

The system clock therefore operates at 25 MHz (X1 = 25 MHz).

Signal	PIN	Input/Output	Function Remarks
RESET	112	Input	RESET

3.6 External power source control output

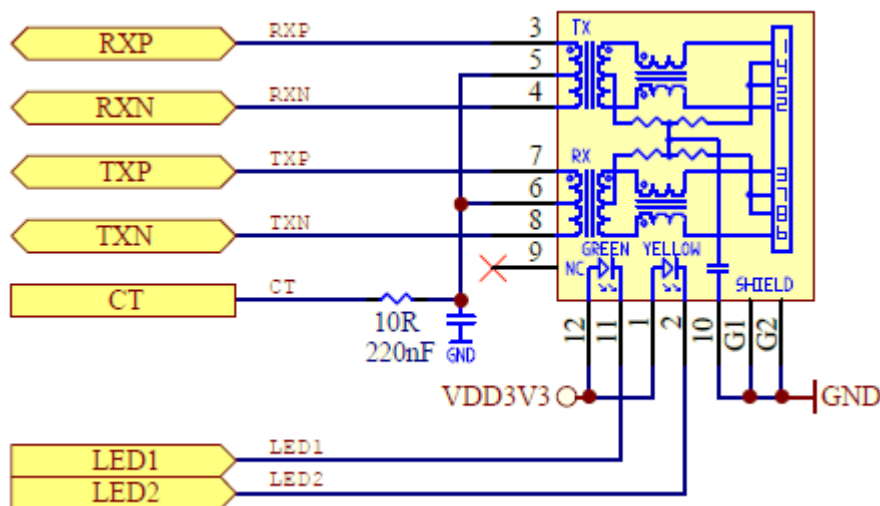
Signal	PIN	Input/Output	Function Remarks
PWE (PORT_C2)	110	Output	This pin controls ON/OFF of the external power source. The "H" level is output during regular operations, and the "L" level is output during standby mode.

3.7 Ethernet - Network Controller

There is an LAN9221i (Industrial Temperature Range) – Ethernet controller by SMC integrated on the TMPA900-CPU board. This supports 10/100 MBit data communication. The Fifo – Select connection (Pin13) is connected to address A12 of the TMPA900. A 330Ohm series resistor has already been integrated for the network LEDs. Furthermore, the TXP, TXN, RXP and RXN are connected by a 490Ohm pull-up resistor. A MAC address has not been set; the firmware takes care of the assignment.

Signal	PIN	Input/Output	Function Remarks
LED1	6	Output	LED1
LED2	8	Output	LED2
CT	10	Output	AVDD Output to Ethernet Magnetics
RXP	14	Input	Receive Positive
RXN	16	Input	Receive Negative
TXP	18	Output	Transmit Positive
TXN	20	Output	Transmit Negative

Connection Example: RJ45 Female Connector on TMPA900-CPU-Board



3.8 UARTs

The TMPA900-CPU-Board contains three UART channels. The feature of each channel is shown below.

	UART 0	UART 1	UART 2
Transmit FIFO	8-bit width / 16 location deep		
Receive FIFO	12-bit width /16location deep		
Transmit/Receive data format	DATA bits : 5,6,7,8bits can be selected PARITY: use / no use STOP bit:1bit / 2bits		
FIFO ON/OFF	ON (FIFO mode)/ OFF (characters mode)		
Interrupt	(1) Combined interrupt factors are output to interrupt controller. (2) The permission of each interrupt factor is programmable.		
baud rate generator	Generates a common transmit and receive internal clock from the UART internal reference clock input. Supports baud rates of up to 6.15Mbps at fPCLK = 100MHz.		
DMA	Support	NO support	support
IrDA 1.0 Function	(1) Max data rate: 115.2kbps(half-duplex) (2) support low power mode	N/A	N/A
Control pins	U0RXD U0TXD U0CTS _n U0CTS _n (Clear To Send) U0DCD _n (Data Carrier Detect) U0DSR _n (Data Set Ready) U0RI _n (Ring Indicator) U0RTS _n (Request To Send) U0DTR _n (Data Terminal Ready)	U1RXD U1TXD	U2RXD U2TXD
Hardware flow control	RTS support CTS support	NC	N/A

UART0:

Signal	PIN	Input/Output	Function Remarks
U0RTS _n (PORT_N7)	24	Output	Output modem control line RTD(Request To Send)
U0DTR _n (PORT_N6)	26	Output	Output modem control line DTR (Data Terminal Ready)
U0RI _n (PORT_N5)	28	Input	Modem status signal RI (Ring Indicator)
U0DSR _n (PORT_N4)	30	Input	Modem status signal DSR (Data Set Ready)
U0DCD _n (PORT_N3)	34	Input	Modem status signal DCD (Data Carrier Detect)
U0CTS _n (PORT_N2)	36	Input	UART0 data can be transmitted (Clear to send)
U0RXD (PORT_N1)	38	Input	UART0 receive data
U0TXD (PORT_N0)	40	Output	UART0 transmission data

UART1:

Signal	PIN	Input/Output	Function Remarks
U1RXD (PORT_T5)	44	Input	UART1 receive data
U1TXD (PORT_T4)	46	Output	UART1 transmission data

UART2:

Signal	PIN	Input/Output	Function Remarks
U2RXD (PORT_F7)	58	Input	UART2 receive data
U2TXD (PORT_F6)	60	Output	UART2 transmission data

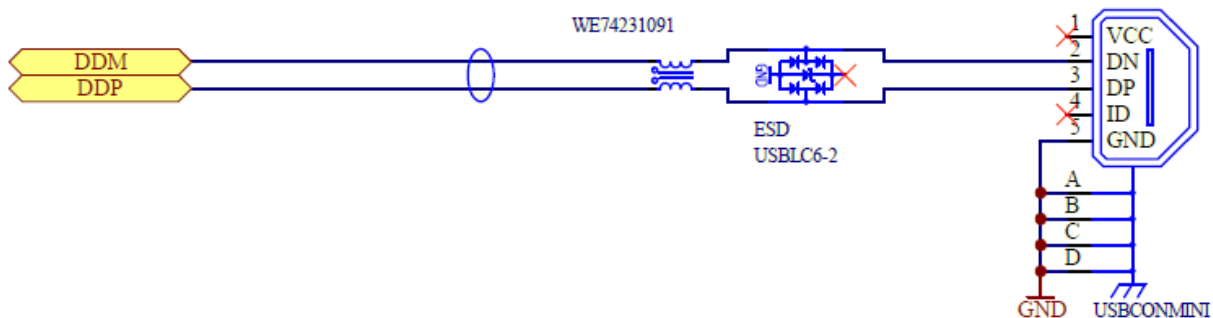
3.9 USB 2.0 - Device

Features:

- Conforms with universal serial bus specification Rev. 2.0
- Supports both high-speed and full-speed (low-speed is not supported).
- Supports Chirp.
- USB protocol processing
- Detects SOF/USB_RESET/SUSPEND/RESUME.
- Generates and checks packet IDs.
- Generates and checks data synchronization bits (DATA0/DATA1/DATA2/MDATA).
- Checks CRC5, generates and checks CRC16.
- Supports PING.
- Supports 4 transfer modes (Control/Interrupt/Bulk/Isochronous).
- Supports 4 endpoints:
 - Endpoint 0: Control 64 bytes × 1 FIFO
 - Endpoint 1: Bulk (IN) 512 bytes × 2 FIFOs
 - Endpoint 2: Bulk (OUT) 512 bytes × 2 FIFOs
 - Endpoint 3: Interrupt (IN) 64 bytes × 1 FIFO
- Supports dual packet mode (except for Endpoint 0).
- Interrupt source signal to interrupt controller: INTS[21]

Signal	PIN	Input/Output	Function Remarks
USB_DDP	17	Input/Output	USB Device pin (D+)
USB_DDM	19	Input/Output	USB Device pin (D-)

Connection Example: MINI USB on the TMPA900-CPU-Board



3.10 USB 2.0 - HOST

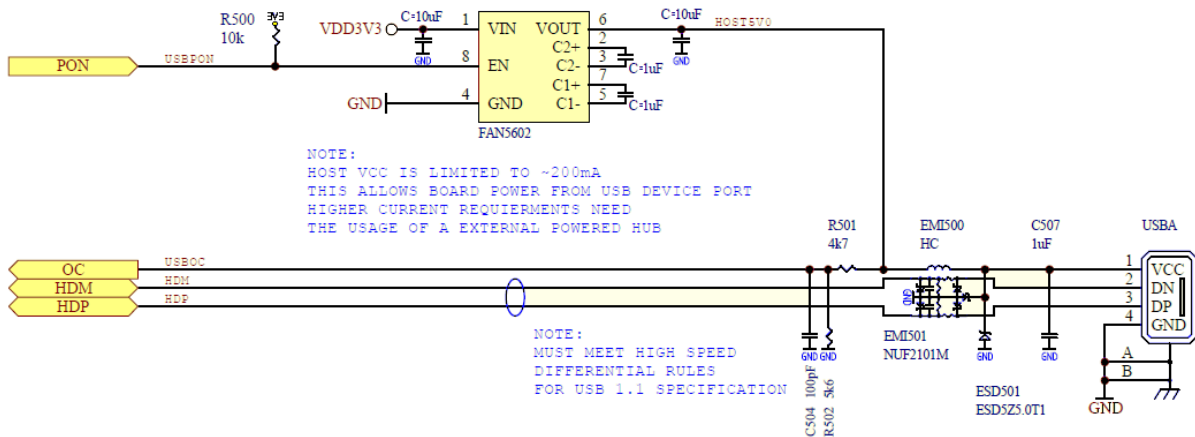
The USB host controller (USBHC) is compliant with USB specification revision 2.0 and the open HCI specification release 1.0a and supports USB transfers at 12 Mbps (full-speed). The USBHC is connected to the multi-layer bus system via on-chip SRAM.

Features:

- Supports full-speed (12 Mbps) USB devices. But doesn't supports low-speed (1.5Mbps)
- Supports control, bulk, interrupt and isochronous transfers.
- Contains two 16-byte FIFO buffers (IN and OUT) in the bus bridge logic for connecting with the CPU, allowing a maximum of 16-byte burst transfers.
- Supports data transfers between the FIFO buffers in the bus bridge logic and the on-chip SRAM.

Signal	PIN	Input/Output	Function Remarks
USB_OC	7	Input	Over current detect for USB host
USB_PON	9	Output	Power on enable for USB host
USB_HDP	13	Input/Output	USB host data (D+)
USB_HDM	15	Input/Output	USB host data (D-)

Connection Example: USB host type A on the TMPA900 CPU board



3.11 I2C

This module operates in I2C bus mode and is compliant with the typical I2C bus standard (Philips specifications).

Features:

- Contains two channels (ch0 and ch1).
- Allows selection between master and slave.
- Allows selection between transmission and reception.
- Supports multiple masters (arbitration, clock synchronization recognition).
- Supports standard mode and fast mode (fastest baud rate in master mode: 89.91 kHz and 357.14 kHz, respectively, at fPCLK = 100 MHz)
- Supports the addressing format of 7 bits only.
- Supports transfer data sizes of 1 to 8 bits.
- Provides one transfer (transmission or reception) complete interrupt (level-sensitive).
- Can enable or disable interrupts. (Interrupt source for I2C ch0: INTS[6], Interrupt source for I2C ch1: INTS[7])

I2C Channel 0:

Normally used for USB host

Signal	PIN	Input/Output	Function Remarks
I2C0CL (USB_OC)	7	Input/Output	I2C clock I/O
I2C0DA (USB_PON)	9	Input/Output	I2C data I/O

I2C Channel 1:

Signal	PIN	Input/Output	Function Remarks
I2C1CL (Port_F6)	60	Input/Output	I2C clock I/O
I2C1DA (Port_F7)	58	Input/Output	I2C data I/O

3.12 SPI (SSP)

Features:

- Contains two channels (ch0 and ch1).
- Communication protocol includes SPI : 3 types
- Master/ slave mode support
- Transmit FIFOs 16-bit wide, 8 locations deep
- Receive FIFOs 16-bit wide, 8 locations deep
- Transmit/Receive data size 4 to 16 bits
- Interrupt type:
 - Transmits interrupt
 - Receives interrupt
 - Receives overrun interrupt
 - Timeout interrupt
- Baud rate master mode: fPCLK/2 (Max 20 Mbps)
- Slave mode: fPCLK/12 (Max 8.33 Mbps)
- DMA Support
- Internal loop back test mode available

SPI (SSP) Channel 0:

Signal	PIN	Input/Output	Function Remarks
SP0D (Port_T3)	48	Input	Data input pin for SSP0
SP0DO (Port_T2)	50	Output	Data output pin for SSP0
SP0CLK (Port_T1)	54	Input/Output	Clock pin for SSP0
SP0FSS (Port_T0)	56	Input/Output	FSS pin for SSP0

SPI (SSP) Channel 1:

Signal	PIN	Input/Output	Function Remarks
SP1DI (Port_L3)	96	Input	Data input pin for SSP1
SP1DO (Port_L2)	98	Output	Data output pin for SSP1
SP1CLK (Port_L1)	100	Input/Output	Clock pin for SSP1
SP1FSS (Port_L0)	102	Input/Output	FSS pin for SSP1

3.13 I2S (Inter-IC Sound)

The TMPA900CM contains a serial input/output circuit compliant with the I2S format. By connecting an external audio LSI, such as an AD converter or DA converter, the I2S interface can support the implementation of a digital audio system.

Features:

	Channel 0	Channel 1
Transmit/Receive	Receive only	Transmit only
Modes	Receive master mode Receive slave mode	Transmit master mode Transmit slave mode
	Full-duplex master mode Full-duplex slave mode Clock through mode	
Data formats	(1) I2S format-compliant (2) Stereo/monaural (3) MSB first/LSB first selectable (4) Left-justified supported (synchronous to WS, no delay)	
FIFO buffer	2 × 8 words	2 × 8 words
Data length	16 bits only	16 bits only
Interrupts	FIFO overflow interrupt FIFO underflow interrupt	FIFO overflow interrupt FIFO underflow interrupt

I2S Channel 0:

Signal	PIN	Input/Output	Function Remarks
I2S0MCLK (Port_L3)	96	Output	I2S0 master clock output for receive circuit
I2S0DATI (Port_L2)	98	Input	I2S0 receive serial data input
I2S0CLK (Port_L1)	100	Input/Output	I2S0 serial clock Input/output
I2S0WS (Port_L0)	102	Input/Output	I2S0 word select Input/output

I2S Channel 1:

Signal	PIN	Input/Output	Function Remarks
I2S1MCLK (Port_M3)	86	Output	I2S1 master clock output for receive circuit
I2S2DATO (Port_M2)	88	Output	I2S1 transmission serial data output
I2S3CLK (Port_M1)	90	Input/Output	I2S1 serial clock Input/output
I2S4WS (Port_M0)	92	Input/Output	I2S1 word select Input/output

3.14 PWM (Pulse Width Modulation) / 16bit-Timers

The TMPA900 CPU board contains six channels of 16-bit timers. Two of them, timer 0 and timer 2 support PWM (Pulse Width Modulation) output.

- 1) Free-running mode
- 2) Periodic timer mode
- 3) PWM function support

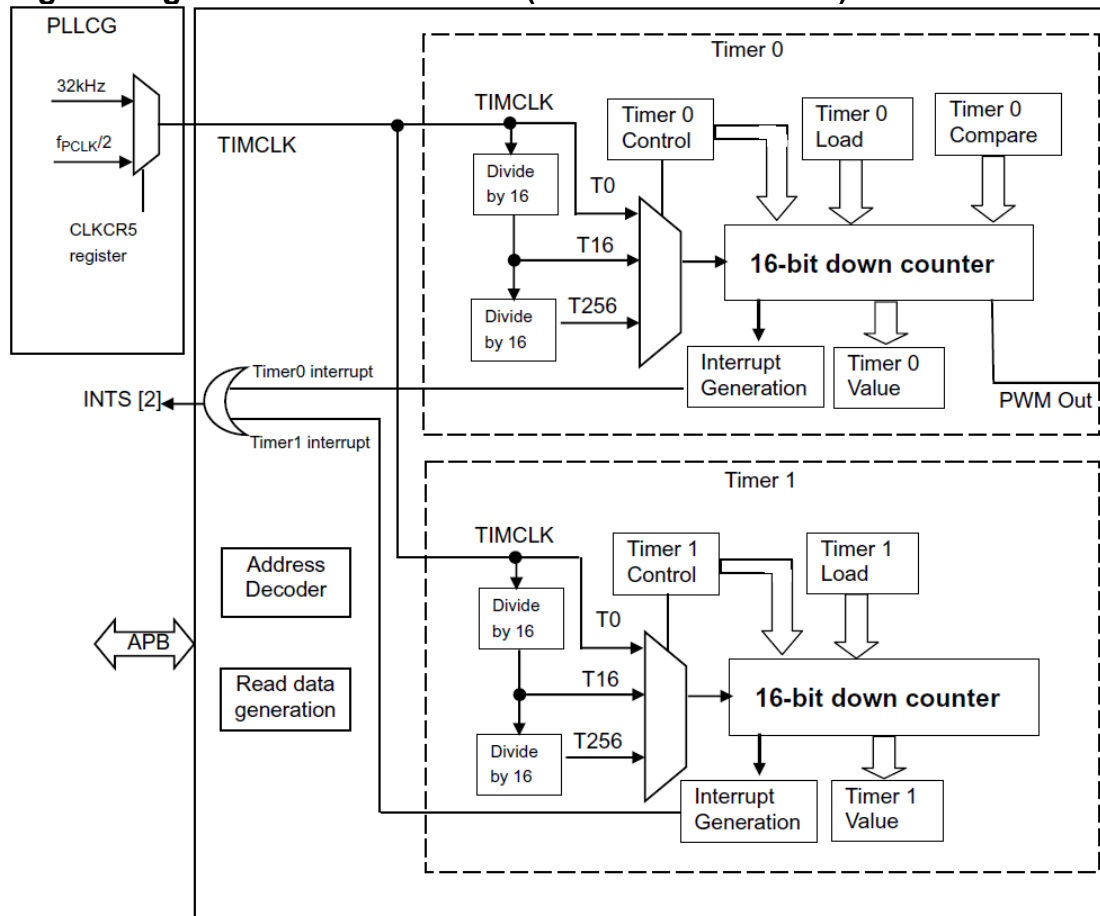
The circuit consists of three blocks, each associated with two channels. Of the three blocks, block 1 and block 2 support PWM (Pulse Width Modulation) output.

	Block 1		Block 2		Block 3	
	Timer0	Timer1	Timer2	Timer3	Timer4	Timer5
Free-Running	●	●	●	●	●	●
Periodic Timer	●	●	●	●	●	●
PWM	●	N/A	●	N/A	N/A	
	PWM0OUT (Port_C3)	x	PWM2OUT (Port_C4)	x	x	x
Interrupt Source Signal	INTS[2]		INTS[3]		INTS[4]	

Signal	PIN	Input/Output	Function Remarks
PWM0OUT (Port_C3)	108	Output	Timer 0 PWM output port
PWM2OUT (Port_C4)	106	Output	Timer 2 PWM output port

Each timer block, containing two channels of timer circuits, comprises two programmable, 16-bit free-running decrement counters. The TIMCLK input is used for counter operation. This clock can be selected from the internal system clock divided by two (fPCLK/2) and fs (32.768 kHz).

Figure diagram of the timer block (Timer 0 and Timer 1).



3.15 JTAG

The TMPA900CMXBG provides a boundary scan interface that is compatible with Joint Test Action Group (JTAG) specifications and uses the industrial standard JTAG protocol (IEEE Standard 1149.1•1990 <Includes IEEE Standard 1449.1a•1993>).

Signal	PIN	Input/Output	Function Remarks
TDO	116	Output	Data output pin for JTAG
TCK RTN	118	Output	JTAG test feedback serial clock output
TCK	120	Input	Clock pin for JTAG
TMS	122	Input	Data for JTAG
TDI	126	Input	Data input pin for JTAG
TRST	128	Input	RESET pin for JTAG
JTAG	130	Input	Boundary scan switching pin, ICE/JTAG test select input (compatible with the Enable signal) -> 0: ICE 1: JTAG
BOOT	132	Input	AM1 Pin

3.16 Keys / Keyboard

Port A can be used not only as a general-purpose input pin with pull up, but also as a key input pin. By enabling interrupts, Port A is used as key input pins (KI3-KI0). Port A can be used without pull up.

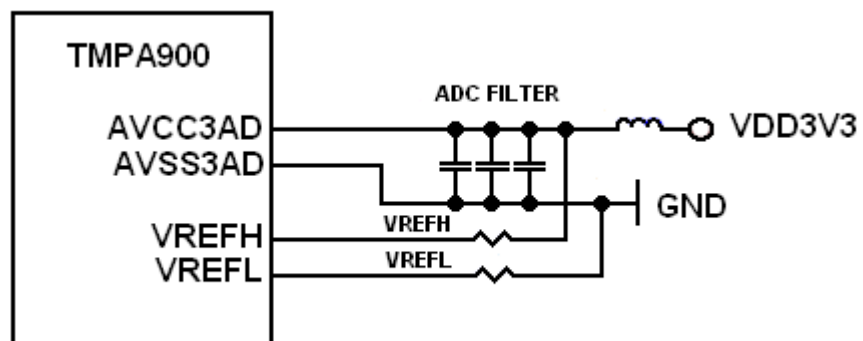
Port B can be used not only as general-purpose output pins, but also as key output pins. By enabling open-drain output, Port B is used as key output (KO3-KO0).

Signal	PIN	Input/Output	Function Remarks
Port A0 (KI0)	29	Input	Port A0 to A3: Input ports Key input KI0 to KI3: Pins for key-on wake up 0 to 3 (with Schmitt input and pull-up resistor)
Port A1 (KI1)	27	Input	
Port A2 (KI2)	25	Input	
Port A3 (KI3)	23	Input	
Port B0 (KO0)	33	Output	Port B0 to B3: Output ports Key output KO0 zp KO3: Key out pins 0 to 3 (open-drain can be set)
Port B1 (KO1)	35	Output	
Port B2 (KO2)	37	Output	
Port B3 (KO3)	39	Output	

3.17 Analog/Digital Converter

A 10-bit serial conversion analog/digital converter (AD converter) with eight channels of analog input is built-in. Four channels (AN4-AN7) are normally used for touch screen interface.

Power Supply of ADC:



Signal	PIN	Input/Output	Function Remarks
AN0 (Port_D0)	82	Input	Port D0 to D3: Input ports Analog input AN0 to AN3: AD Converter Input Pins
AN1 (Port_D1)	80	Input	
AN2 (Port_D2)	78	Input	
AN3 (Port_D3)	76	Input	

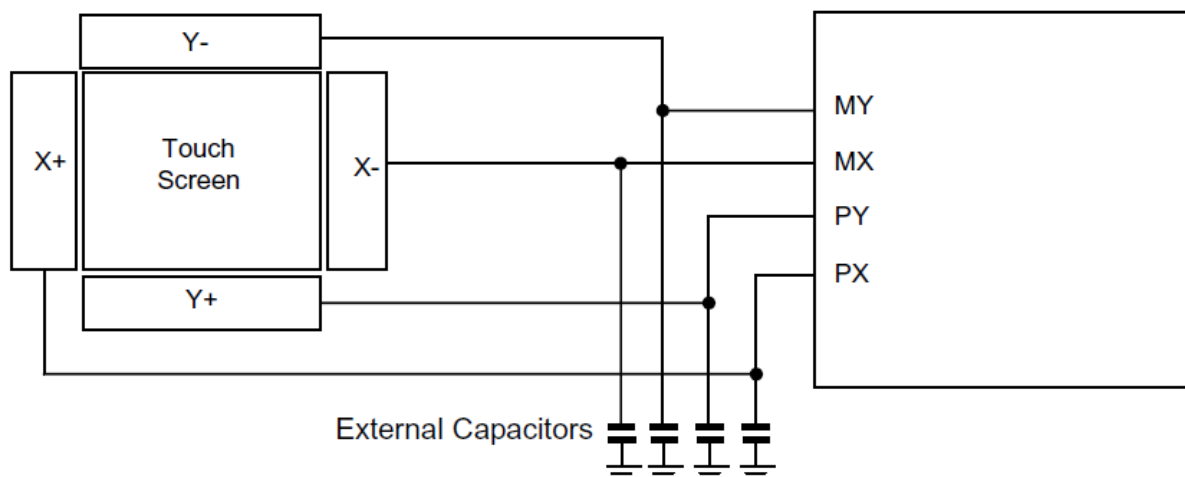
Pins normally used for Touch Screen Interface

Signal	PIN	Input/Output	Function Remarks
AN7 (TSI_PY)	65	Output	Port D4 to D7: Input ports Analog input AN4 to AN7: AD Converter Input Pins
AN6 (TSI_PX)	67	Output	
AN5 (TSI_MY)	69	Output	
AN4 (TSI_MX)	71	Output	

3.18 Touch Screen Interface (TSI)

An interface for a 4-terminal resistor network touch screen is built-in. The TSI easily supports two procedures: touch detection and X/Y position measurement. Each procedure is performed.

TMPA900_CPU_BOARD



Signal	PIN	Input/Output	Function Remarks
TSI_PY (Port_D7/AN7)	65	Output	Y-plus: Y-connecting pin for touch panel
TSI_PX (Port_D6/AN6)	67	Output	X-plus: X-connecting pin for touch panel
TSI_MY (Port_D5/AN5)	69	Output	Y-minus: Y-connecting pin for touch panel
TSI_MX (Port_D4/AN4)	71	Output	X-minus: X-connecting pin for touch panel

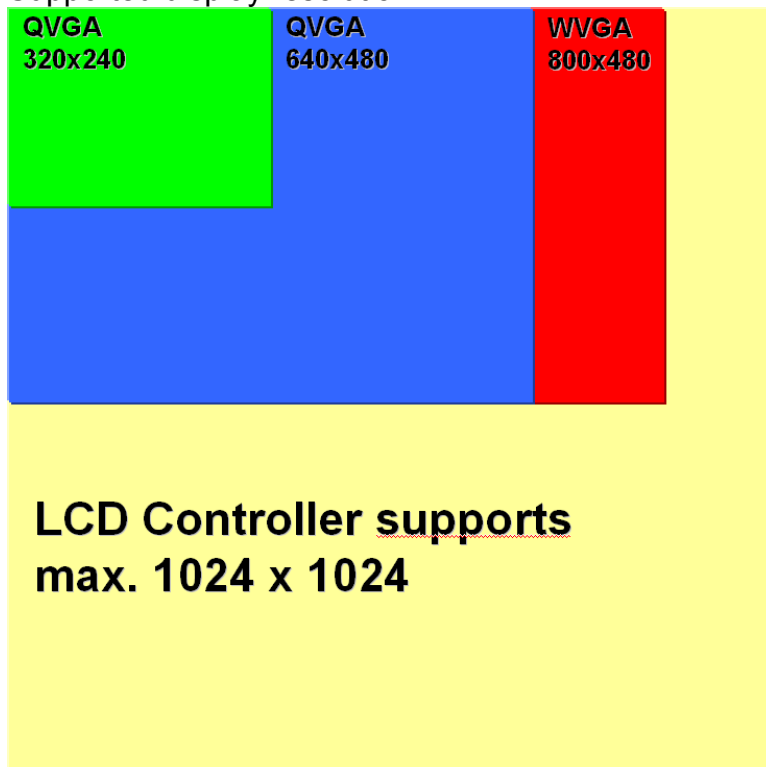
3.19 LCD Controller (LCDC)

SUPPORTED DISPLAY COLORS & DISPLAY RESOLUTION

Display colours: from 256 to 16 million:

Data bus width	RGB	Display colours
24 bit	R8:G8:B8	16 million
18 bit	R6:G6:B6	262 144
16 bit	R5:G6:B5	65 536
12 bit	R4:G4:B4	4 096
8 bit	R3:G3:B2	256

Supported display resolution:



Up to WVGA with LCD data process accelerator:

- Scaling function (expansion/reduction)
- Filter function (bi-cubic convolution)
- Image blending function (font blending)

Up to 1024 x 1024 without accelerator and limited colour depth.

LCD Interface Signals:

Signal	PIN	Input/Output	Function Remarks
L_CLLP	116	Output	Data output pin for JTAG
L_CLFP	118	Output	JTAG test feedback serial clock output
L_CLAC	120	Input	Clock pin for JTAG
L_CLCP	122	Input	Data for JTAG

Signal	PIN	Input/Output	Function Remarks
L_D0	141	Output	LCD data driver line
L_D1	139	Output	LCD data driver line
L_D2	137	Output	LCD data driver line
L_D3	135	Output	LCD data driver line
L_D4	131	Output	LCD data driver line
L_D5	129	Output	LCD data driver line
L_D6	127	Output	LCD data driver line
L_D7	125	Output	LCD data driver line
L_D8	121	Output	LCD data driver line
L_D9	119	Output	LCD data driver line
L_D10	117	Output	LCD data driver line
L_D11	115	Output	LCD data driver line
L_D12	111	Output	LCD data driver line
L_D13	109	Output	LCD data driver line
L_D14	107	Output	LCD data driver line
L_D15	105	Output	LCD data driver line
L_D16	101	Output	LCD data driver line
L_D17	99	Output	LCD data driver line
L_D18	97	Output	LCD data driver line
L_D19	95	Output	LCD data driver line
L_D20	91	Output	LCD data driver line
L_D21	89	Output	LCD data driver line
L_D22	87	Output	LCD data driver line
L_D23	85	Output	LCD data driver line

3.20 Glyn Graphic Base Board & Glyn TFT Family Concept

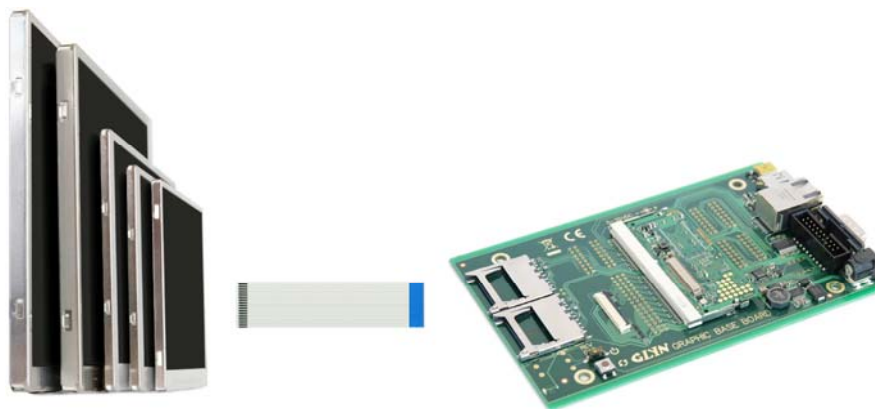
The Concept

Our objective was to offer a number of TFTs that . . .

- 1) are compatible with each other,
- 2) provide a modern interface and
- 3) have long-term availability.

In close collaboration with our partner EDT (Emerging Display Technologies), we have selected a number of TFTs that have been modified according to your needs. A PCB on the back of the display provides important functionalities as well as a common interface.

The family concept displays can be connected to the graphic base board (starter kit) directly. All display signals are lead through the SODIMM slot.



7.0" - 5.7" - 5.0" - 4.3" - 3.5"

Part number	Size	Resolution	Dimensions
G-ET0350G0DM6 (DH6)	3.5"	320 x 240	76.8 x 63.8 mm
G-ET0430G0DM6 (DH6)	4.3"	480 x 272	105.5 x 67.2 mm
G-ET0500G0DM6 (DH6)	5.0"	800 x 480	118.5 x 77.6 mm
G-ETQ570G0DM6 (DH6)	5.7"	320 x 240	124.7 x 100 mm
G-ETQ570G2DM6 (DH6)	5.7"	320 x 240	142.1 x 100 mm *
G-ETV570G0DMU (DHU)	5.7"	640 x 480	124.7 x 100 mm
G-ETV570G2DMU (DHU)	5.7"	640 x 480	142.1 x 100 mm *
G-ET0700G0DM6 (DH6)	7.0"	800 x 480	166 x 105.4 mm

DH6/DHU = with touch panel

* with mounting lugs

Pin Assignment of TFT Family Concept

1	NC or /RESET	Hardware Reset (no Reset for 4.3" and 7.0")
2	Vss	Ground (connected to metal housing)
3	B5	Blue Data Bit 5
4	B4	Blue Data Bit 4
5	B3	Blue Data Bit 3
6	B2	Blue Data Bit 2
7	B1	Blue Data Bit 1
8	B0	Blue Data Bit 0
9	Vss	Ground (connected to metal housing)
10	G5	Green Data Bit 5
11	G4	Green Data Bit 4
12	G3	Green Data Bit 3
13	G2	Green Data Bit 2
14	G1	Green Data Bit 1
15	G0	Green Data Bit 0
16	Vss	Ground (connected to metal housing)
17	R5	Red Data Bit 5
18	R4	Red Data Bit 4
19	R3	Red Data Bit 3
20	R2	Red Data Bit 2
21	R1	Red Data Bit 1
22	R0	Red Data Bit 0
23	Vss	Ground (connected to metal housing)
24	DCLK	Dot Clock
25	Vss	Ground (connected to metal housing)
26	HSYNC	Horizontal Sync Input
27	VSYNC	Vertical Sync Input
28	ENB	Data Enable Input
29	PWCTRL	LED driver shut down (on 5.7" it is a complete shut down of internal circuit)
30	VDD	Power supply for digital circuit
31	Vss	Ground (connected to metal housing)
32	Vss	Ground (connected to metal housing)
33	Vcc	Power supply for Vcom driver circuit (internal voltages)
34	Vcc	Power supply for Vcom driver circuit (internal voltages)
35	NC	Not connected / This becomes LED anode, when jumper setting is changed
36	LEDCTRL	Brightness control (or LED cathode, when jumper setting is changed)
37	YU	Touch (Top)
38	XL	Touch (Left)
39	YD	Touch (Bottom)
40	XR	Touch (Right)

3.21 SD Host Controller

Features:

- Data transmission/reception in frame units
- Error check: CRC7 (for commands), CRC16 (for Data)
- Synchronous method: bit synchronous by SDCLK
- SD memory/IO card interface: COMMAND (1bit), Data (4 bits), INT (1bit)
- Multiple port support: 1 card
- 512byte ×2 data buffer: 256words×16bits×2
- Card detect support (SDCxCD or SDCxDAT3)
- Data write protect support
- Detected below Status error
 - SDbuffer underflow /overflow
 - timeout (response, other)
 - END error, CRC error, CMD error
- Recognizes the various response frame formats through the register settings
- The SD_CLK cycle division ratio can be set from fPCLK/2 to fPCLK/512
- The transfer data length can be set from either 1byte to 512byte
- Sector counter for multiple read/write operation (read: single read only)
- Buffer status mode transfer support

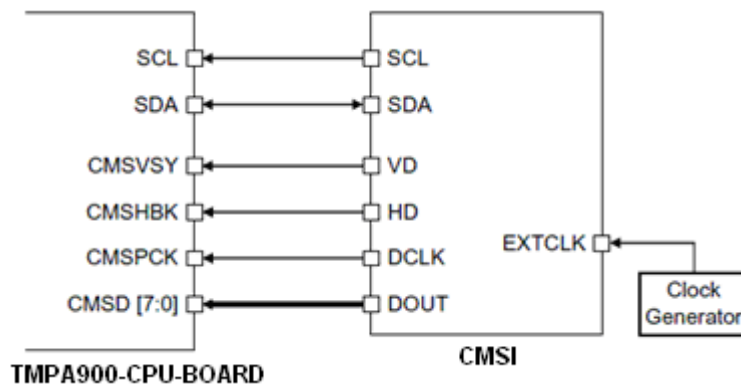
This product contains an SD host controller for controlling SD cards. To use the SD host controller, you need to join the SD Association. Please also note that a non-disclosure agreement must be signed with us before the detailed specifications of the SD host controller can be disclosed. For details, please contact us.

Signal	PIN	Input/Output	Function Remarks
SDC0DAT0 (Port_G0)	59	Input/Output	SDC0DAT0 to SDC0DAT3: Data I/O pin for SD card
SDC0DAT1 (Port_G1)	57	Input/Output	
SDC0DAT2 (Port_G2)	55	Input/Output	
SDC0DAT3 (Port_G3)	53	Input/Output	
SDC0CMD (Port_G4)	49	Input/Output	Command I/O pin for SD card
SDC0WP (Port_G5)	47	Input	Write-protect input pin for SD card
SDC0CD (Port_G6)	45	Input	Card detection input pin for SD card
SDC0CLK (Port_G7)	43	Input	Clock output pin for SD card

3.22 CMOS Camera Interface

The CMSI has the following features:

- SXGA(1280×1024), 4VGA(1280×960), VGA(640×480), QVGA(320×240), Special(320×180), QQVGA(160×120), CIF(352×288), QCIF(176×144)
- Input data format CRGB
- Input data sampling ratio 8-bit YUV4:2:2 or RGB888 if no color space conversion)
- Downscaling function
- 4VGA → VGA, QVGA, QQVGA
- VGA → QVGA, QQVGA
- QVGA → QQVGA
- Trimming function: Data can be trimmed to a desired size.



Signal	PIN	Input/Output	Function Remarks
L_D12(CMSPCK)	111	Input	Clock input for CMOS sensor
L_D13(CMSHSY)	109	Input	Horizontal synchronization input for CMOS sensor
L_D14(CMSHBK)	107	Input	Valid data detect input for CMOS sensor
L_D15(CMSVY)	105	Input	Vertical synchronization input for CMOS sensor
L_D16(CMSD0)	101	Input	CMOS data driver line
L_D17(CMSD1)	99	Input	CMOS data driver line
L_D18(CMSD2)	97	Input	CMOS data driver line
L_D19(CMSD3)	95	Input	CMOS data driver line
L_D20(CMSD4)	91	Input	CMOS data driver line
L_D21(CMSD5)	89	Input	CMOS data driver line
L_D22(CMSD6)	87	Input	CMOS data driver line
L_D23(CMSD7)	85	Input	CMOS data driver line

Attention: CMOS camera Interface uses some of the same pins as the LCDC data bus – so only 12bit colour (4096 display colours) is possible.

3.23 Melody/Alarm Generator

Melody Generator

Based on the low-speed clock (32.768 kHz), clock wave forms can be generated at any frequency from 4 Hz to 5461 Hz and outputted from the MLDALM pin. By connecting a buzzer etc outside, melody sounds can easily be played.

Alarm Generator

Can generate eight patterns of alarm output.
 Can generate five types of fixed-interval interrupts (1 Hz, 2 Hz, 64 Hz, 512 Hz and 8192 Hz).
 By connecting a buzzer etc outside, alarm sounds can easily be played.

Signal	PIN	Input/Output	Function Remarks
MLDALM (PORT_C3)	108	Output	Melody/alarm output pin

3.24 Low Frequency Clock Output

Signal	PIN	Input/Output	Function Remarks
FSOUT (PORT_C4)	106	Output	Low frequency output clock pin

4.0 Pin Allocation SODIMM 144 Connector

IS10 J19154-144-XXA110X		
1	GND01	GND02
3	VDD3V3	VDD3V3
5	FUTURE_USE	ETHLED1
7	PC7/I2C0DA/INT9/USBOCN	ETHLED2
9	PC6/I2C0CL/USBPON	ETHCT
11	GND03	GND04
13	SN6/USBHDP	ETHRX
15	SN7/USBHDM	ETHRXN
17	SR0/USBDDP	ETHTX
19	SR1/USBDDM	ETHTXN
21	GND05	GND06
23	PA3/KI3	PN7/UORTSN/INTG
25	PA2/KI2	PN6/UODTRN/INTF
27	PA1/KI1	PN5/UORIN/INTE
29	PA0/KI0	PN4/UODSRN/INTD
31	GND07	GND08
33	PB3/KO3	PN3/UODCDN
35	PB2/KO2	PN2/UOCTS
37	PB1/KO1/LCLAC	PN1/UORXD/SIROIN
39	PB0/KO0	PN0/UOTXD/SIROOUT
41	GND09	GND10
43	PG7/SDC0CLK	PT5/U1RXD
45	PG6/SDC0CD	PT4/U1TXD
47	PG5/SDC0WP	PT3/SP0D1
49	PG4/SDC0CMD	PT2/SP0D
51	GND11	GND12
53	PG3/SDC0DAT3	PT1/SP0CLK
55	PG2/SDC0DAT2	PT0/SP0FSS
57	PG1/SDC0DAT1	PF7/I2C1DA/INTC/U2RXD
59	PG0/SDC0DAT0	PF6/I2C1CL/U2TXD
61	VDD3V3	VDD3V3
63	GND13	GND14
65	PD7/INTB/AN7/PY	FUTURE_USE
67	PD6/INTA(TS1)/AN6/PX	FUTURE_USE
69	PD5/AN5/MY	FUTURE_USE
71	PD4/AN4/MX	FUTURE_USE
73	GND15	GND16
75	SU4/LCLLP	PD3/AN3
77	SU3/LCLFP	PD2/AN2
79	SU1/LCLAC	PD1/AN1
81	SU0/LCLCP	PD0/AN0
83	GND17	GND18
85	LC_D23	PM3/I2S1MCLK
87	LC_D22	PM2/I2S1DAT0
89	LC_D21	PM1/I2S1CLK
91	LC_D20	PM0/I2S1WS
93	GND19	GND20
95	LC_D19	PL3/I2S0CLK/SP1D1
97	LC_D18	PL2/I2S0DAT1/SP1D0
99	LC_D17	PL1/I2S0CLK/SP1CLK
101	LC_D16	PL0/I2S0WS/SP1FSS
103	GND21	GND22
105	LC_D15	PC4/FSOUT/PWM3OUT
107	LC_D14	PC3/MLDALM/PWM1OUT
109	LC_D13	PC2/PWE
111	LC_D12	RESET
113	GND23	GND24
115	LC_D11	SP5/TDO_JTAG
117	LC_D10	SP4/RTCK_JTAG
119	LC_D9	SP0/TCK_JTAG
121	LC_D8	SP1/TMS_JTAG
123	GND25	GND26
125	LC_D7	SP2/TDI_JTAG
127	LC_D6	SP3/TRSTN_JTAG
129	LC_D5	SN2/SELJTAG
131	LC_D4	SM7/AM1_JTAG
133	GND27	GND28
135	LC_D3	VIN3V3
137	LC_D2	VIN3V3
139	LC_D1	VIN3V3
141	LC_D0	VIN3V3
143	GND29	GND30

PINOUT (1/5)

Number of Pin	Pin Name	Input/Output	Function Remarks
1-2	GND	-	Ground
3-4	VDD3V3	Output	Power
5	nc	Output	leave open
6	LED1	Output	Ethernet LED1: Speed LED
7	USB_OC	Input/Output Input/Output Input	PC7: I/O port I2C0DA: I2C data I/O INT9: Interrupt request pin9: an interrupt request pin that can program the rising/falling edge USB0Cn: Over Current detect for USB Host
8	LED2	Output	Ethernet LED2: Link/Active LED
9	USB_PON	Input/Output Input/Output Output	PC6: Port C6: I/O port I2C0CL: I2C clock I/O USBPON: Power On Enable for USB Host
10	CT		Ether
11-12	GND	-	Ground
13	USB_HDP		USB Host pin (D+)
14	RXP		Ethernet
15	USB_HDM		USB Host pin (D-)
16	RXN		Ethernet
17	USB_DDP		USB Device pin (D+)
18	TXP		Ethernet
19	USB_DDM		USB Device pin (D-)
20	TXN		Ethernet
21-22	GND	-	Ground
23	Port A3	Input	PA3: Input port Key input KI3: Pin for key-on wake up (with Schmitt input and pull-up resistor)
24	Port N7	Input/Output Output Input	PN7: I/O port U0RTSn: Output modem control line RTD(Request To Send) INTG: Interrupt request pin G: an interrupt request pin that can program the rising/falling edge
25	Port A2	Input	Port A2: Input port Key input KI2: Pin for key-on wake up (with Schmitt input and pull-up resistor)
26	Port N6	Input/Output Output Input	PN6: I/O port U0DTRn: Output modem control line DTR (Data Terminal Ready) INTF: Interrupt request pin F: an interrupt request pin that can program the rising/falling edge
27	Port A1	Input	Port A1: Input port Key input KI1: Pin for key-on wake up (with Schmitt input and pull-up resistor)
28	Port N5	Input/Output Input Input	PN5: I/O port U0RIn: Modem status signal RI (Ring Indicator) INTE: Interrupt request pin E: an interrupt request pin that can program the rising/falling edge
29	Port A0	Input	Port A0: Input port Key input KI0: Pin for key-on wake up (with Schmitt input and pull-up resistor)
30	Port N4	Input/Output Input Input	PN4: I/O port U0DSRIn: Modem status signal DSR (Data Set Ready) INTD: Interrupt request pin D: an interrupt request pin that can program the rising/falling edge
31-32	GND	-	Ground

PINOUT (2/5)

Number of Pin	Pin Name	Input/Output	Function Remarks
33	Port B3	Output Output	PB3: Output port KO3: Key out pins (open-drain can be set)
34	Port N3	Input/Output Input	PN3: I/O port U0DCDn: Modem status signal DCD (Data Carrier Detect)
35	Port B2	Output Output	PB2: Output port KO2: Key out pins (open-drain can be set)
36	Port N2	Input/Output Input	Port N2: I/O port U0CTS _n : UART function 0 data can be transmitted (Clear to send)
37	Port B1	Output Output	PB1: Output port KO1: Key out pins (open-drain can be set)
38	Port N1	Input/Output Input Input	Port N1: I/O port U0RXD: UART function 0 receive data SIR0IN: Data input pin for IrDA1.0
38	Port B0	Output Output	PB0: Output port KO0: Key out pins (open-drain can be set)
40	Port N0	Input/Output Output Output	Port N0: I/O port UART function 0 transmission data Data output pin for IrDA1.0
41-42	GND		Ground
43	Port G7	Input/Output Input/Output	Port G7: I/O port SDC0CLK: Clock output pin for SD card
44	Port T5	Input/Output Input	Port T5: I/O port U1RXD: UART function 1 receive data
45	Port G6	Input/Output Input	Port G6: I/O port SDC0CD: Card detection input pin for SD card
46	Port T4	Input/Output Output	Port T4: I/O port U1TXD: UART function 1 transmission data
47	Port G5	Input/Output Input	Port G5: I/O port SDC0WP: Write-protect input pin for SD card
48	Port T3	Input/Output Input	PT3: I/O port SP0DI: Data input pin for SSP0
49	Port G4	Input/Output Input/Output	Port G4: I/O port SDC0CMD: Command I/O pin for SD card
50	Port T2	Input/Output Output	Port T2: I/O port SP0DO: Data output pin for SSP0
51-52	GND	-	Ground
53	Port G3	Input/Output Input/Output	PG3: I/O port SDC0DAT3: Data I/O pin for SD card
54	Port T1	Input/Output Input/Output	Port T1: I/O port SP0CLK: Clock pin for SSP0
55	Port G2	Input/Output Input/Output	PG2: I/O port SDC0DAT2: Data I/O pin for SD card
56	Port T0	Input/Output Input/Output	PT0: I/O port SP0FSS: FSS pin for SSP0
57	Port G1	Input/Output Input/Output	PG1: I/O port SDC0DAT1: Data I/O pin for SD card
58	Port F7	Input/Output Input/Output Input	PF7: I/O port I2C1DA: I2C data I/O INTC: Interrupt request pin C: an interrupt request pin that can program the rising/falling edge U2RXD: UART function 2 receive data
59	Port G0	Input/Output Input/Output	PG0: I/O port SDC0DAT0: Data I/O pin for SD card
60	Port F6	Input/Output Input/Output Output	PF6: I/O port I2C1CL: I2C clock I/O U2TXD: UART function 2 transmission Data
61-62	VDD3V3	Output	
63-64	GND	-	Ground

PINOUT (3/5)

Number of Pin	Pin Name	Input/Output	Function Remarks
65	TSI_PY	Input Input Output Input	PD7: Input port AN7: Analog input 7 converter input pin PY: Y-plus: Y-connecting pin for touch panel INTB: interrupt request pin B - an interrupt request pin that can program the rising/falling edge
66	NC	-	Not connected.
67	TSI_PX	Input Input Output Input	PD6: Input port AN6: Analog input 6 converter input pin PX: X-plus: X-connecting pin for touch panel INTA: Interrupt request pin A - an interrupt request pin that can program the rising/falling edge
68	NC	-	Not connected.
69	TSI_MY	Input Input Output	PD5: Input port AN5: Analog input 5 -> AD converter input pin MY: Y-minus -> Y-connecting pin for touch panel
70	NC	-	Not connected.
71	TSI_MX	Input Input Output	PD4: Input port AN4: Analog input 4 -> AD converter input pin MX: X-minus -> X-connecting pin for touch panel
72	NC	-	Not connected.
73-74	GND	-	Ground
75	L_CLLP	Output	LCLLP: LCD driver output pin
76	Port D3	Input Input	PD3: Input port AN3: Analog input pin
77	L_CLFP	Output	LCLFP: LCD driver output pin
78	Port D2	Input Input	PD2: Input port AN2: Analog input pin
79	L_CLAC	Output	LCLAC: LCD driver output pin
80	Port D1	Input Input	PD1: Input port AN1: Analog input pin
81	L_CLCP	Output	LCLCP: LCD driver output pin
82	Port D0	Input Input	PD0: Input port AN0: Analog input pin
83-84	GND	-	Ground
85	L_D23	Output Output Output	PK7: Output ports LD23: Data bus for LCD driver CMSD7: Data bus for CMOS Sensor
86	Port M3	Input/Output Output	PM3: I/O port I2S1MCLK: I2S1 master clock output for transmission circuit
87	L_D22	Output Output Output	PK6: Output ports LD22: Data bus for LCD driver CMSD6: Data bus for CMOS Sensor
88	Port M2	Input/Output Output	PM2: I/O port I2S1DATO: I2S1 transmission serial data output
89	L_D21	Output Output Output	PK5: Output ports LD21: Data bus for LCD driver CMSD5: Data bus for CMOS Sensor
90	Port M1	Input/Output Input/Output	PM1: I/O port I2S1CLK: I2S1 serial clock input/output
91	L_D20	Output Output Output	PK4: Output ports LD20: Data bus for LCD driver CMSD4: Data bus for CMOS Sensor
92	Port M0	Input/Output Input/Output	PM0: I/O port I2S1WS: I2S1 word select input/output
93-94	GND	-	Ground

PINOUT (4/5)

Number of Pin	Pin Name	Input/Output	Function Remarks
95	L_D19	Output Output Output	PK3: Output ports LD19: Data bus for LCD driver CMSD3: Data bus for CMOS Sensor
96	Port L3	Input/Output Output Output	PL3: I/O port I2S0MCLK: I2S0 master clock output for receive circuit SP1DI: Data input pin for SSP1
97	L_D18	Output Output Output	PK2: Output ports LD18: Data bus for LCD driver CMSD2: Data bus for CMOS Sensor
98	Port L2	Input/Output Input Output	PL2: I/O port I2S0DATI: I2S0 receive serial data input SP1DO: Data output pin for SSP1
99	L_D17	Output Output Output	PK1: Output ports LD17: Data bus for LCD driver CMSD1: Data bus for CMOS Sensor
100	Port L1	Input/Output Input/Output Input/Output	PL1: I/O port I2S0CLK: I2S0 serial clock input/output SP1CLK: Clock output pin for SSP1
101	L_D16	Output Output Output	PK0: Output ports LD16: Data bus for LCD driver CMSD0: Data bus for CMOS Sensor
102	Port L0	Input/Output Input/Output Input/Output	PL0: I/O port I2S0WS: I2S0 word select input/output SP1FSS: FSS pin for SSP1
103-104	GND	-	Ground
105	L_D15	Input/Output Output Input	PJ7: Input/Output port LD15: Data bus for LCD driver CMSHBK: Vertical synchronization Input for CMOS Sensor
106	Port C4	Output Output Output	PC4: Output port FSOUT: Low-frequency output clock pin PWM2OUT: Timer PWM out port
107	L_D14	Input/Output Output Input	PJ6: Input/Output port LD14: Data bus for LCD driver CMSHBK: Valid Data detect input for CMOS Sensor
108	Port C3	Output Output Output	PC3: Output port MLDALM: Melody alarm output pin PWM0OUTPUT: Timer PWM out port
109	L_D13	Input/Output Output Input	PJ5: Input/Output port LD13: Data bus for LCD driver CMSHSY: Horizontal synchronization Input for CMOS Sensor
110	Port C2	Output Output	PC2: Output port PWE: External power source control output.
111	L_D12	Input/Output Output Input	PJ4: Input/Output port LD12: Data bus for LCD driver CMSPCK: Clock input for CMOS Sensor
112	RESET	Input	Reset: Initializes TMPA910CRA (with Schmitt input and pull-up resistor)
113-114	GND	-	Ground

PINOUT (5/5)

Number of Pin	Pin Name	Input/Output	Function Remarks
115	L_D11	IOutput Output	PJ3: Output ports LD11: Data bus for LCD driver
116	TDO	Output	TDO: Data output pin for JTAG
117	L_D10	Output Output	PJ2: Output ports LD10: Data bus for LCD driver
118	TCK_RTN	Output	RTCK: Clock output pin for JTAG
119	L_D9	Output Output	PJ1: Output ports LD9: Data bus for LCD driver
120	TCK	Input	Clock pin for JTAG
121	L_D8	Output Output	PJ0: Output ports LD8: Data bus for LCD driver
122	TMS	Input	TMS: Pin for JTAG
123-124	GND	-	Ground
125	L_D7	Output	LD7: Data bus for LCD driver
126	TDI	Input	TDI: Data input pin for JTAG
127	L_D6	Output	LD6: Data bus for LCD driver
128	TRST	Input	TRSTn: Reset pin for JTAG
129	L_D5	Output	LD5: Data bus for LCD driver
130	JTAG		
131	L_D4	Output	LD4: Data bus for LCD driver
132	BOOT		
133-134	GND		
135	L_D3	Output	LD3: Data bus for LCD driver
136	VIN	Input	Power Supply
137	L_D2	Output	LD2: Data bus for LCD driver
138	VIN	Input	Power Supply
139	L_D1	Output	LD1: Data bus for LCD driver
140	VIN	Input	Power Supply
141	L_D0	Output	LD0: Data bus for LCD driver
142	VIN	Input	Power Supply
143-144	GND	-	Ground

The module's power supply must run through pins VIN: 136, 138, 140, 142, GND: 133, 134, 143, 144 in order to ensure good disturbance reaction results. It is recommended to filter and buffer the power supply.

Signal currents of external appliances may be connected via the module's GND connections, e.g. in order to facilitate the layout. In order to avoid failures, however, higher currents and current peaks are not permissible.

5.0 Software Components

5.1 Basics - Data Transfer to TMPA900 CPU Board

There are three methods for the data transfer available:

- **ELDIO Download Wizard**
Programming Software over USB-Device with a small Windows program - This method works **only** if the memory (Flash) is empty (first time programming or cleared before).
- **JTAG (Joint Test Action Group)**
This method works even if the memory (Flash) is empty (first time programming).
- **Network**
Using an appropriate network boot ROM or a boot loader, it is also possible to download your application over a network using for example TFTP, FTP or NFS. The target will download the data from a server residing on the host and then executed. During the development phase, this method allows you to quickly test your application without having to burn the flash. **We use the free boot manager u-boot and a TFTP server for uploading the software.**

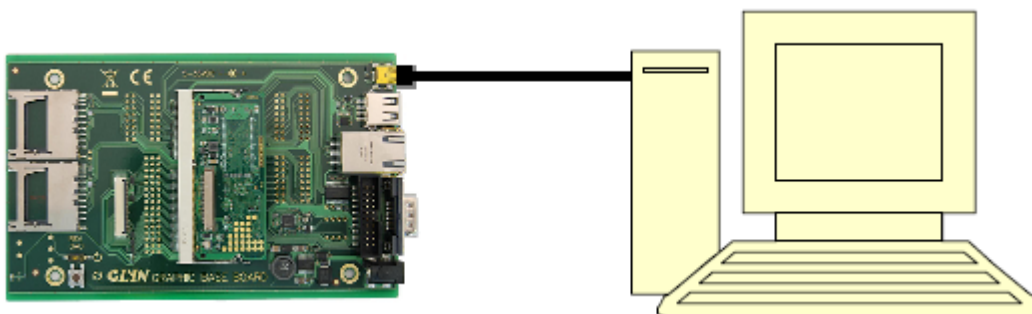
5.1.1 ELDIO Download Wizard

ATTENTION:

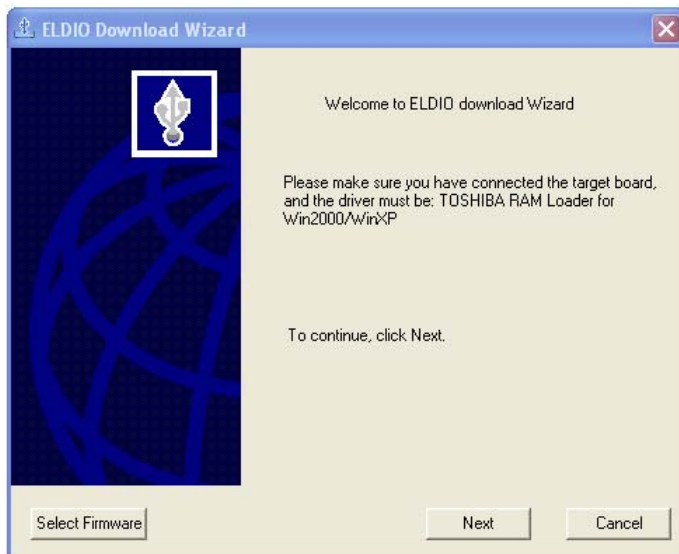
- Works only with Empty NAND Flash (at least page 0)
- Board has to be connected to PC without USB Hub

First, before using ELDIO Updater install the driver. Rightclick on: trl_drv_2k.inf (For Windows 2000) or trl_drv_xp.inf (Windows XP) and select install.

Connect the Base Board (USB-Device) to the PC. If you have problems with connection to the board when you attach the board via USB Hub, please connect the board directly to the PC.

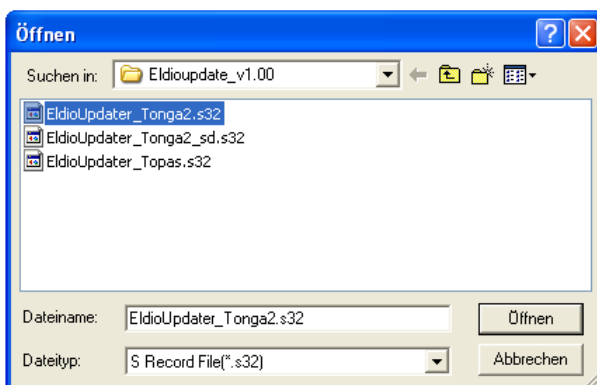


Start up ELDIOUpdater.exe



Lower left side - Click on Select Firmware - Choose Firmware for our board:

EldioUpdater_Tonga2.s32



Then Click on Next- Wait until connection established appears. Click on Next
Select Destination (Ram, NOR, NAND) – RAM or NAND.
Select start page (normally 0 if you want to flash u-boot, or bigimage)

Then select Target File to Flash. Click on next
Wait until flash process has finished.

There is also a small script “**gen_flash_image**” to make one “big” image out of the components – u-boot, splashscreen, kernel and root-filesystem.

Run this script under Linux in a terminal in the following way:

```
./gen_flash_image <u-boot> <kernel> <splash> <rootfs> <output>
```

Example:

```
./gen_flash_image ./u-boot_nand_tonga2.bin ./ulmage ./splash.bin ./rootfs.jffs2  
./output.bin
```

The result is one big output.bin file. Now you can program the complete software in one step!

You find it with a demo on the CD: Eldiouupdate_v1.00\bigimage

5.1.2 Basics – Installing J-Link Lite

The first software is a MS Windows software, so it runs only under windows. Before you plug your J-Link Lite into your computer's USB port, you have to extract the setup tool **Setup_JLinkARM_V<VersionNumber>.zip**. You will find it on the CD in the folder Segger_JLINK_Lite. The setup wizard will install the software and documentation pack that also includes the certified JLink USB driver. Start the setup by double clicking Setup_JLinkARM_V<Version-Number>.exe. The license agreement dialog box will open. Accept the terms with the "Yes" button.

After installing the software, connect the Jlink on your host to an USB port and on the dedicated JTAG connector on the Glyn graphic base board.

5.1.3 Basics - Installing a TFTP Server

To transfer data to the board, you have to install a TFTP server. Nowadays, there are various free servers available. The documentation of these servers is usually so detailed that the installation is usually not directly addressed. One of the most important features is the fast uploading software via TFTP.

It's the best way during development phase.

5.1.4 Basics - Working completely under Linux

The delivered J-Link is also working under Linux using openOCD Please refer to <http://openocd.berlios.de/web/> for installing and usage.

It's also possible to have a tpft server running under Linux - please refer to your Linux Distribution Documentation how to install

5.2 u-boot

Without a good boot loader, the TMPA900-CPU-Board is just a complicated piece of silicon with nothing to do. That's where u-boot, a free universal boot loader software, steps in (www.denx.de).

A boot loader, sometimes referred to as a boot monitor, is a small piece of software that executes soon after powering up the module. In an embedded system, the role of the boot loader is more complicated since these systems do not have a BIOS to perform the initial system configuration. The low level initialization of microprocessors, memory controllers, and other board specific hardware varies from board to board and CPU to CPU. These initializations must be performed before a Linux kernel or a normal C-program image can execute.

The minimum an embedded loader has to offer are the following features:

- Initializing the hardware, especially the memory controller.
- Providing boot parameters for the Linux kernel.
- Starting the Linux kernel or program/application

Additionally, most boot loaders also provide "convenience" features that simplify development:

- Set up a UART for terminal
- Reading and writing to the memory
- Uploading new binary images to the board's RAM via a serial line or Ethernet
- Copying binary images from RAM to FLASH memory
- First handling of the memory devices Nand-Flash & DDR-RAM.

U-boot is a boot loader which is very common in the embedded Linux world. The u-boot supports different architectures - in our case, ARM. The boot loader has been published under the GNU licence which means that it is absolutely free of charge.

Detailed documentation can be found at:

<http://www.denx.de/wiki/u-bootdoc/Presentation>
<http://www.denx.de/wiki/u-bootdoc/WebHome>

GIT server for the u-boot (u-boot sources patched to work with our board):

<http://git.labs.kernelconcepts.de/?p=u-boot-tmpa9xx.git;a=summary>

Demo file systems, Linux kernel binary, u-boot binary:

<http://www.mucross.com/downloads/tonga-linux/>

5.2.1 The Boot Process

How does the u-boot work on the TMPA900 CPU board?

There is a DDR-RAM and NAND flash integrated on the TMPA900 CPU board. It is not possible to execute the program code directly from the NAND flash. The program code has to be copied in segments by a boot manager (u-boot) from the NAND flash to the DDR RAM and then executed from there.

First, the CPU executes the boot code in the boot ROM. This boot code loads a special boot block from the NAND flash into the SRAM of the CPU. This special boot block initialises the RAM and loads the u-boot from the NAND to the RAM and jumps into it.

There are many additional features on the TMPA900-CPU-Board, such as installation of software via RS232 and TFTP. The u-boot needs approx. 200kb. Including the boot block, it needs approx. 330kBytes.

Being an interactive console (11500,8N1), the u-boot enables hardware initialisation during the development process. The u-boot allows hardware tests (e.g. RAM, network). One of the most important features is the uploading of software, e.g. via TFTP. U-boot also offers debug features.

An overview of some important u-commands:

- **help** prints online help
- **help <command>** prints online help for a command
- **printenv** prints environment variables

Configuration commands:

- **setenv ipaddr <ip>** Sets the IP-Address of the board
- **setenv serverip <ip>** Sets the TFTP server (host)IP address
- **saveenv** Saves environment variables

Flash memory commands:

- **nand write <Source-Address> <Destination-Address> <Size>**
- **nand read <Source-Address> <Size>**
- **nand erase <Start-Address> <Size>**

5.2.2 Flashing the u-boot

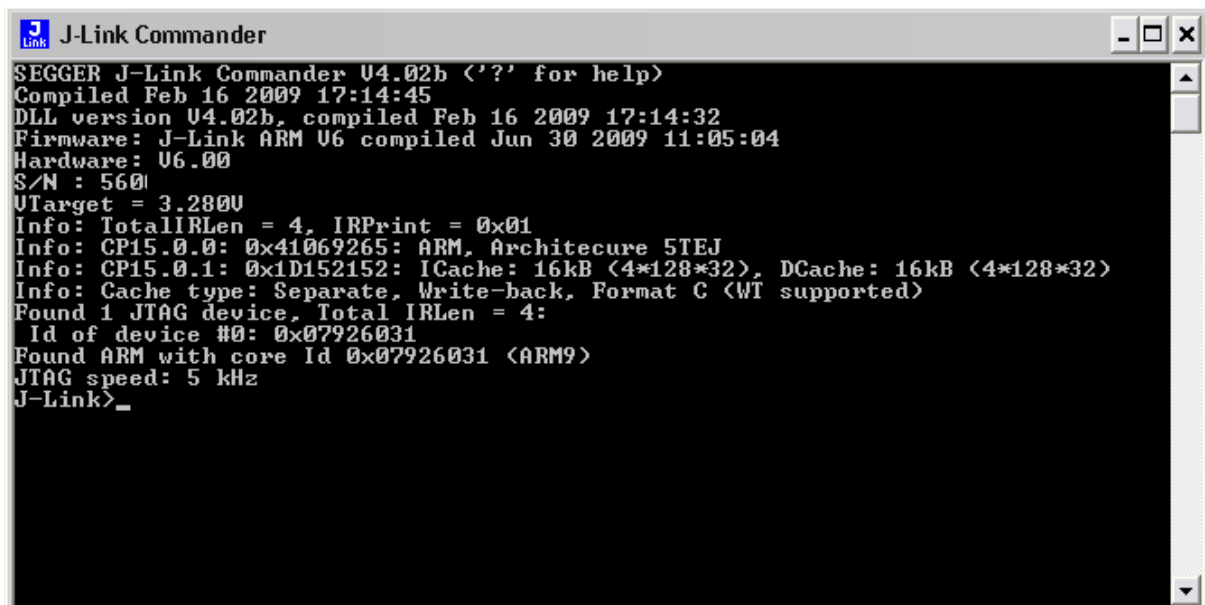
There are two ways to flash the u-boot on the board.

5.2.2.1 Flashing the u-boot over JTAG

This method works even if the memory (flash) is empty. For this, check also chapter: 5.2.2.9 What to do if the boot loader has been flashed incorrectly

In order to flash the precompiled version of the boot manager on the board, you have to use the J-Link Lite **with the u-boot RAM version (u-boot.bin)**. To make sure that the J-Link works, you have to run the installation software from the Segger folder. This allows all required drivers, as well as the J-Link – commander, to be installed on the PC. Now connect the J-Link to the GLYN-BASE-BOARD and your host PC. Run the program Terraterm with the setting 115200/8/no/1/no flow. (alternatively Hypertherm).

The boot manager is constantly updated. So you please download the current version from our web page – www.toshiba-mikrocontroller.de.



```
J-Link Commander
SEGGER J-Link Commander V4.02b ('?' for help)
Compiled Feb 16 2009 17:14:45
DLL version V4.02b, compiled Feb 16 2009 17:14:32
Firmware: J-Link ARM V6 compiled Jun 30 2009 11:05:04
Hardware: V6.00
S/N : 5601
UTarget = 3.280V
Info: TotalIRLen = 4, IRPrint = 0x01
Info: CP15.0.0: 0x41069265: ARM, Architecture 5TEJ
Info: CP15.0.1: 0x1D152152: ICache: 16kB (4*128*32), DCache: 16kB (4*128*32)
Info: Cache type: Separate, Write-back, Format C (WT supported)
Found 1 JTAG device, Total IRLen = 4:
  Id of device #0: 0x07926031
Found ARM with core Id 0x07926031 (ARM9)
JTAG speed: 5 kHz
J-Link>_
```

Please enter the following commands and confirm by pressing return:

Speed adaptive

r

g

halt

loadbin u-boot.bin,0x43F00000

setpc 0x43F00000

g

From now on, the boot manager should login to the terminal and report its status. In order to flash the boot manager, go back to the J-link commander and enter the following commands:

halt

loadbin u-boot_nand_tonga2.bin,0x40600000

g

Now go back to the terminal program and enter the following commands:

nand erase 0 0x60000

nand write 0x40600000 0 0x60000

If you now push the RESET button, the u-boot in the terminal program will answer and can be configured. The u-boot can be used to load both a Linux kernel and a normal application. The bad block handling, which is necessary for NAND flashing, is also done by the u-boot.

Note: The start address or size depends on which u-boot is used and the matching kernel. These values refer to the versions of 6/2/2010.

5.2.2.2 Update the u-boot via network (handle with care)

If you want to change the existing u-boot on the board, you can do this via TFTP server.

```
>tftp u-boot_nand_tonga2.bin // Loads u-boot into RAM.
```

```
>nand erase u-boot // Erases old u-boot
```

```
>nand write ${fileaddr} u-boot // Writes new u-boot into flash
```

Important: After installing the uboot you have to reset your system (restart u-boot).

If you have some problems with the update – for example nothing works anymore - please check chapter: **5.2.2.9 What to do if the boot loader has been flashed incorrectly**

If you want to change the existing -old- u-boot (Before August 2010), you can do this also via TFTP server.

```
> tftp 0x40600000 u-boot_nand_tonga2.bin // Loads u-boot into RAM.  
> nand erase 0 0x60000 // Erases old u-boot  
> nand write 0x40600000 0 0x60000 // Writes new u-boot into flash
```

5.2.2.3 U-boot - Environment Setup

First, the serial terminal program runs and auto boot is aborted by pressing the SPACE bar.

```
U-Boot 2010.06 (Sep 01 2010 - 11:08:49)
```

```
DRAM: 64 MiB  
NAND: 256 MiB  
Found Environment offset in OOB..  
Net: smc911x-0
```

```
NAND read: device 0 offset 0x80000, size 0x300000  
3145728 bytes read: OK  
Hit any key to stop autoboot: 0  
Tonga2>
```

The u-boot settings can now be viewed by entering “printenv”.

Important: After installing the uboot you have to reset your system (restart u-boot).

5.2.2.4 IP and MAC Address Setup

Configuration of the IP address of the development board and your host PC.

The board should be allocated the address 192.168.0.122 which is set by entering „set ipaddr 192.168.0.122”. You have to adapt the IP addresses according to your environment. These values can be saved by entering “saveenv”. You can test the correct configuration with a ping on your host PC.

```
>setenv ipaddr 192.168.100.121  
>setenv serverip 192.168.100.120  
>saveenv
```

If required, you have to set the MAC address:

```
>setenv ethaddr DE:AD:DE:AD:DE:AD  
  
>saveenv
```

5.2.2.5 Configuration of the Display Parameters

To ensure that bigger displays with other timings also work with our board, the kernel must be informed of the timing-parameters, such as resolution.

```
>setenv videoparams video=tmpa9xxfb:19211e4c:10040cef:013f380d
>saveenv
```

Check also chapter 8.1 Other Resolutions/Other Timings – Calculation of the Display Settings. There you can find how this parameter is calculated.

LCD Control Register

The fourth parameter of the LCD control register (LCDControl) can be included in the video params call. For the EDT Family Concept this is not necessary.

```
>setenv videoparams video=tmpa9xxfb:19211e4c:10040cef:013f380d:nnnnnnnn
```

nnnnnnnn stands for the LCD Control Register value.

5.2.2.6 Configuration of the File System Type

You can choose between JFFS2 and UBIFS.

JFFS2 is a log-structured file system for use in flash memory devices.

UBIFS - Unsorted Block Image File System is a successor to JFFS2.

Setting for UBIFS format:

```
>setenv rootfs_base 'setenv rootfs ${rootfs_ubifs}'
>saveenv
```

Setting for JFFS2 format (Default):

```
>setenv rootfs_base 'setenv rootfs ${rootfs_jffs2}'
>saveenv
```

5.2.2.7 Splash Screen Support

A splash screen is an image that appears after a very short time to notify the user that the program is processing while the system is loading the kernel etc.

The u-boot for the TMPA9XX boards is also capable of doing splash screen before booting the Linux kernel.

There are some limitations given:

- There is only support for TFT panels at the moment.

- The splash screen has to be exactly the size and the bits per pixel as the display.
- The only format supported is BMP or compressed BMP with gzip.
- The splash screen has to be A8R8G8B8 in the case of 24bit TFTs
- The splash screen has to be R5G6B5 in the case of 16bit TFTs

Both formats can be easily generated with gimp (when saving BMP open up Advanced Options and select Format)

If you would like to use the splash, you have to use the pre-boot environment variable.

Example:

```
>ftp splash.bmp  
>nand write ${fileaddr} splash  
>setenv preboot 'nand read 0x43000000 splash;bmp display 0x43000000'  
>saveenv
```

Erase splash screen:

```
>nand erase splash
```



5.2.2.8 Erase u-boot Environment.

```
>nand erase u-boot_env
```

This erases the whole environment variable area in the memory. After this, you have the default setting.

5.2.2.9 u-boot - NFS Server Setup

It is possible to download your application via a network. Then the target will download the data from a server (e.g. your PC) into RAM. This method allows you to test your application quickly without having to burn the flash.

In order to test your program quickly and easily, first you have to copy your root file system (tar.gz) to the nfsroot-directory of your Linux host system.

Doing this, you can archive very short turn-around cycles for debugging your application or testing your kernel as all changes are executed directly from the host and the board uses the new testing versions directly without the need of flashing everything for a new test.

Example setup NFS boot in your u-boot:

```
>setenv bootargs_base 'setenv bootargs console=ttyS0,115200n8 ${mtdparts}
root=/dev/nfs nfsroot=192.168.100.120:/target ip=192.168.100.121
${videoparams} ethaddr=${ethaddr}'
```

```
>setenv bootcmd 'run bootargs_base;tftp ulmage;bootm'
```

```
>saveenv
```

When you now boot your target, the complete file system will come from the NFS server.

Using the setenv bootargs_base your filesystem will come from your host system (NFS-Server).

Using the setenv bootcmd your kernel will come from your host system (tftp-Server).

When using both, your whole system (except u-boot) will be provided by your host system.

5.2.2.10 More u-boot commands

By the first start of the u-boot the command PREBOOT is put and appears setup. Set-up looks whether mtdparts is put and then explains the following commands:

```
nand bad
dynpart
nand env.oob set u-boot_env
setenv preeboot
setenv setup
saveenv
```

This is done automatically when u-boot starts the first time.

Explanation of the commands

>nand bad

The answer in the terminal could look like this:

```
Device 0 bad blocks:  
04380000  
0d940000
```

The u-boot command "nand bad" lists the offsets of the bad blocks. More about "bad blocks" in chapter **3.2.1 Nand Flash Memory – Unique Characteristics**.

>dynpart

There is a new 'dynpart' command which, when executed, uses the compile time board partition sizes combined with the bad block table to generate the device-specific 'dynamic' partition table. Among other things, this table contains a partition with the name u-boot_env. This is the partition where the environment is saved! The result is stored in the mtdparts environment variable. Everything else is standard u-boot/kernel behaviour.

If you now look in the environment (printenv), you will find the new device-specific 'dynamic' partition table as below:

```
mtdparts=mtdparts=tm9a900-nand:0x00060000(u-boot),0x00020000(u-  
boot_env),0x00300000(splash),0x00300000(kernel),0x0f980000(rootfs)
```

>nand env.oob set u-boot_env

This command stores the position of the environment partition in the out-of-band (OOB) bytes of the first page which is always fine.

Why do we do this? The u-boot environment is traditionally stored at a fixed location within the NAND flash. This is not acceptable, since it could be a factory-set bad block. The solution is to put the in-flash address of the environment into the out-of-band (OOB) area of the first block (the one which is guaranteed to be good).

5.2.2.11 What to do if the boot loader has been flashed incorrectly

By setting the magic word, booting is initiated in the first sector of the NAND flash and the internal boot loader is activated. The set boot option prevents initialisation of the DDR RAM and, therefore, prevents the described flash process. Should the u-boot not start due to an incorrect update, it is not possible to flash the application using the normal tool.

The update script in the u-boot allows the initialisation of the DDR RAM and loading of the application in the RAM. To do this, please open the Segger J-Link commander. Copy the contents of the whole script by pressing “STRG+C” and paste the contents by pressing “STRG+V”.

```
Speed adaptive
r
g
halt
w4 0xf005000c, 0x00000007
w4 0xf0050010, 0x00000065
w4 0xf005000c, 0x00000087
w4 0xf0050008, 0x00000003
w4 0xf0050004, 0x00000000
w4 0xf0050054, 0x00000040
w4 0xf080c424, 0x000000fd
w4 0xf080c428, 0x00000002
w4 0xF0020260, 0x3
w4 0xf4310014, 0x4
w4 0xf4310018, 0x1
w4 0xf431001c, 0x2
w4 0xf4310020, 0xa
w4 0xf4310024, 0xa
w4 0xf4310028, 0x13
w4 0xf431002c, 0x10a
w4 0xf4310030, 0x13
w4 0xf4310034, 0x2
w4 0xf4310038, 0x2
w4 0xf431003c, 0x1
w4 0xf4310040, 0xa
w4 0xf4310044, 0xc
w4 0xf4310048, 0x14
w4 0xf431000c, 0x10012
w4 0xf4310304, 0x58
w4 0xf4310010, 0xa60
w4 0xf4310200, 0x000140FC
w4 0xf4310204, 0x000180FF
w4 0xf4310208, 0x000180FF
w4 0xf431020c, 0x000180FF
w4 0xf4310008, 0xc0000
w4 0xf4310008, 0x00000
w4 0xf4310008, 0x40000
w4 0xf4310008, 0x40000
w4 0xf4310008, 0x80032
w4 0xf4310008, 0xc0000
w4 0xf4310008, 0xa0000
w4 0xf4310100, 0x5
w4 0xf4310104, 0x5
w4 0xf4310108, 0xb
w4 0xf431010c, 0x5
w4 0xf4310110, 0x5
w4 0xf4310114, 0x5
w4 0xf4310004, 0x0
w4 0xf00a0050, 0x1
w4 0xf4311014, 0x4afaa
w4 0xf4311018, 0x1
w4 0xf4311010, 0xc00000
loadbin D:\u-boot.bin,0x40300000
loadbin D:\u-boot_nand_tonga2.bin,0x40600000
setpc 0x40300000
g
```

Afterwards, the flash has to be programmed. To do this, switch to the terminal program and enter the following commands:

```
nand erase 0x0 0x60000  
nand write 0x40600000 0x0 0x60000
```

5.3 Standard Application (IAR Compiler)

In order to recompile the projects, you need the IAR embedded workbench. You will find a trial version on the web at www.IAR.com.

5.3.1 Debugging the Application (IAR Compiler)

An IAR project for direct use without an operating system can be found on the CD which is included in the delivery. The Segger J-Link lite has to be connected to the TMPA900 CPU module to debug. The work space is set to "DDR_Debug" for debugging. The program's normal functions can now be tested.

5.3.2 Make a Release for Flash (IAR Compiler)

Stop the current debugging session in the IAR embedded workbench. Select the release NAND configuration and rebuild the sample application. You will find a release file in the project folder NAND_Release\Exe.

5.3.3 Flashing the Application (No Linux)

There is a **download tool by Segger** on the CD in the folder Segger_Download_tool or you can find the current version on the Segger homepage www.segger.com.

This tool set restores your GLYN graphic base board with the TMPA900 CPU module to a defined state. It restores u-boot to loading after start up and downloads an application to be run by u-boot after a short start up delay.

The tool set can be used together with IAR EWARM to automatically download an output into NAND with u-boot. u-boot and the application starts after a short delay.

To use the tool set independently to download a binary together with u-boot, please make sure that you rename your binary to "application.bin" and put it into the same folder as the toolset. Please be aware that the tool set expects the vectors to be located at addr. 0x40600000.

To program your image independently, simply start the batch file "Download.bat".

Flashing without Download Tool:

The application must be copied from the NAND flash to a free RAM address behind the boot manager before starting. To do this, it is copied to the RAM of the TMPA900 CPU module via the TFTP server via Ethernet. Enter the following commands:

tftp 0x40600000 GettingStarted.bin

The program is copied to the external DDR RAM.

nand erase 0x80000 0x200000

The NAND flash is deleted with this command. The first value is the start address; the second value states the size of the area.

nand write 0x40600000 0x80000 0x200000

This flashes the program which is shown in the DDR RAM from address 0x40600000 onwards to the NAND address 0x80000. The size of the program is 0x200000.

Auto copy und start application

bootcmd nand read 0x40600000 0x80000 0x200000 \;go 0x40700000

During start up, the program is copied from the NAND address 0x80000 to the RAM address 0x40600000 (File size: 0x200000). The application is started from address 0x40700000 (reset vector).

saveenv

All changes are saved and the application starts automatically after resetting.

5.3.4 Getting Started with SEGGER Evaluation Software and IAR

You can find a complete evaluation version of the Segger software in the folder Segger_Demo on the CD.

This evaluation package has been designed to provide a complete, easy-to-use software package for the TMPA900 CPU Board and IAR.s embedded workbench for ARM (target compiler). It allows you to easily check the target hardware, the target compiler and Segger software components. This evaluation process typically does not take a long time since the software can be easily recompiled and downloaded to the target.

Software Components in the Package

- **emFile**
emFile is SEGGER's embedded file system that can be used on any media for which you can provide basic hardware access functions. emFile is a high-

performance software that has been optimised for speed, versatility and memory footprint. emFile documentation can be found at "Doc\UM02001_emFile.pdf".

- **emWin**
emWin is SEGGER's embedded Graphical User Interface (GUI) using a feature-rich API and providing an efficient, processor and LCD controller-independent GUI for any application that operates with a display. emWin documentation can be found at "Doc\UM03001_emWinUser.pdf".
- **emboss**
embOS is SEGGER's embedded priority-controlled multitasking system. It is designed to be used as an embedded operating system for the development of real-time applications and has been optimised for minimum memory consumption in both RAM and ROM, as well as high speed and versatility. embOS documentation can be found at "Doc\UM01001_embOS_Generic.pdf" and "Doc\UM01002_embOS_ARM_IAR.pdf".
- **embOS/IP**
embOS/IP is SEGGER's embedded TCP/IP stack. It is a CPU independent, high-performance TCP/IP stack that has been optimised for speed, versatility and small footprint. embOS/IP documentation can be found at "Doc\UM07001_embOSIP.pdf".
- **emUSB**
emUSB is SEGGER's embedded USB stack. It is written in ANSI C and features bulk communication as well as device classes such as MSD, CDC or HID. emUSB documentation can be found at "Doc\UM09001_USBStack.pdf".

Eval Limitations

The included eval versions of the different components of the eval package have the following limitations:

Component	Description
emFile	The eval version of the emFile libraries can only handle one open file at any given time.
embOS	The eval version of the embOS libraries run without a time limit with a maximum of three tasks. If your application creates more than three tasks, embOS stops after a time limit of 15 minutes.
emOS/IP	IP The eval version of the embOS/IP libraries have a time limit of 15 minutes on the connection.
emUSB	The eval version of the emUSB libraries have a time limit of 15 minutes on the connection.
emWin	The eval version of the emWin library shows an evaluation notification before the actual application starts.

Your use of the eval package or of any part included in the project indicates your

acknowledgement of and agreement to the SEGGER eval software license .
License.txt is located in the root directory of the eval package.

For details look also the Segger applicaton note in the folder *Segger_Demo - AN00002_GettingStartedWithSeggerEvalSoftwareAndIAR.pdf* .

6.0 Linux for TMPA900 CPU board

Important Preliminary Remark:

A precompiled Linux kernel comes with the board. If you have your own base board, the right way is to make your own (custom) kernel.

Example: You do not need "SD over SPI Driver" because no SD card is used in your design. In this case, our kernel would constantly show an error message in the terminal window. So, you should take the "SD over SPI Driver" out of the kernel.

To optimise the efficiency of the TMPA900 CPU boards, we always advise a custom kernel:

- For performance reasons:
- More free memory
- Quicker boot
- Less basic load (e.g., MMC over SPI is polled)

First steps in chapter: 6.5 Linux Kernel Build

Where to find:

Linux kernel sources:

<http://git.labs.kernelconcepts.de/?p=topas.git;a=summary>

First steps with the git server in **Appendix C: KC Labs Public Git Server**

Demo file systems, kernel binary, u-boot binary - configured for the Glyn graphics base board:

<http://www.mucross.com/downloads/tonga-linux/>

Our Linux Cross Compiler and Root File System is based on μ Cross, a modern Linux based software distribution for embedded and mobile devices. The μ Cross package is a product from kernel concepts (www.kernelconcepts.de).

More in chapter **6.9 μ Cross – Linux Tool Package.**

6.1 Major Components of a Linux System

A Linux system, be it on a main frame or an appliance, consists of three major parts:

- Boot loader (on a PC the BIOS)
- Linux kernel
- Root file system

There are plenty of ways to combine and create all this and each part has its quite specific features and effect on the behaviour of the system. As a general rule, you can assume that a Linux system starts up in the following way:

1. CPU power-on reset loads and executes boot loader.
2. Boot loader initialises some required hardware and then loads Linux kernel binary image and jumps to load address.
3. Linux kernel initialises its drivers and thus the hardware, then mounts the root file-system and finally executes the first user space program, which is either **/sbin/init** or **/bin/sh**.

So the **init** process is always the first process started. This becomes the root of the process tree and triggers all other processes.

6.2 Flashing the Linux Application

Note: The start address or size depends on the u-boot used and the matching kernel. These values refer to the versions of August 2010.

It is assumed that the current version of the boot manager as well as the TFTP – server have already been installed. You will find a consistently updated kernel on our web page (<http://www.toshiba-mikrocontroller.de/>) and the file system on the CD in the subdirectory KernelConcepts.

Press RESET on the base board. The following commands are re-entered via a serial terminal (115200/8/no/1/no Flow).

Install kernel:

```
>tftp ulmage  
>nand erase kernel  
>nand write ${fileaddr} kernel
```

Install Rootfs (max size ~60MB):

```
>tftp mucross-1.0-x11-gtk-qt4-image-tonga2-summary.jffs2  
>nand erase rootfs  
>nand write.jffs2 ${fileaddr} rootfs ${filesize}
```

6.3 Flash Layout TMPA900-CPU-BOARD

Partition	Start	Length (Hex)	Size
Bootblock + u-boot	0x00000000	0x60000	384 kBytes
Environment	0x00060000	0x20000	128 kBytes
Splash Screen Partition	0x00080000	0x300000	3 MBytes
Linux Kernel	0x00380000	0x300000	3 MBytes
File System	0x00680000	0xFC80000	249.5 MBytes

Note: The start address or size depends on the u-boot used and the matching kernel. In the current version (August 2010) these sizes are dynamic (see u-boot command dynpart).

6.4 Installation Linux Tool chain TMPA900 CPU board

The cross tool chain is shipped as a compressed Unix tar archive and is suitable as-is for the most current Linux x86 32-bit hosts. Tested host distributions include recent Debian and Ubuntu releases.

Unpack the archive to the root (/) directory of the host work station. Alternatively, a virtual machine such as VMware or VirtualBox with a Linux installation can be used.

You can find the current tool chain at:

<http://www.mucross.com/downloads/tonga-linux/mucross-1.0-i686-linux-armv5te-linux-gnueabi-toolchain-gtk-qt4.tar.bz2>

The SDK files can be found in the directory /opt/mucross/<arch>.

6.5 Linux Kernel Build

First, there is a ready-to-use kernel on the CD – Ulmage. But sometimes you have to rebuild the kernel according to your needs. The main reason is to optimise the kernel to your environment (hardware and usage patterns). E.g. I2C1 and UART2 have the same port pins – for this you have to choose what peripheral you want to use.

First you have to download the kernel source:

<http://git.labs.kernelconcepts.de/?p=topas.git;a=summary>

6.5.1 Linux Kernel Source Tree

The Linux kernel source code is organized as a tree. The following list shows the root of our kernel tree as with version 2.6.x.

```
/
.gitignore      .mailmap      COPYING      CREDITS
Documentation   Kbuild        MAINTAINERS  Makefile
README         README.kc    REPORTING-BUGS
arch          block        crypto        drivers
firmware       fs          include      init
ipc            kernel       lib           mm
net            samples     scripts      security
sound         tools       usr          virt
```

Below you will find a description of some important folders.

/arch

The arch/ drawer contains the support for the different platforms supported by Linux. Any source file having a platform dependency can be found here. Especially for our board you find the **tonga.c** ARM architecture in the **mach-tmpa910 folder for our board** .

/drivers

The drivers/ directory host the device drivers of the Linux kernel. The various subfolders are sorted according to the different device categories. When writing a new device driver, start here by copying a working one that is similar to the function of your new driver.

E.g. In this folder you can find the video folder with the frame buffer driver for our board -> **/ drivers / video / tmpa910_fb.c**

/fs

This drawer includes all the file system related functionality of the Linux kernel. It contains the core functions and the specific support files for the different file systems.

/include

This is the header file drawer of Linux. All header files needed for kernel compilation can be found here. A special case is the asm/ subfolder, which gets replaced by a symbolic link to asm-arm/ if using ARM as the target platform.

/init

This directory contains the main initialisation (init) code of the Linux kernel. This also includes the mounting of the root file system and the start of the init process.

/kernel

At kernel/ all parts of the kernel core functions are stored. These parts normally use plain 'C' code. Any platform specific code, sometimes written in assembly language, can be found at arch/.

/net

This drawer hosts the networking core support of the Linux kernel. The various subfolders include TCP/IP support and other common network functions.

6.5.2 Linux Kernel Configuration

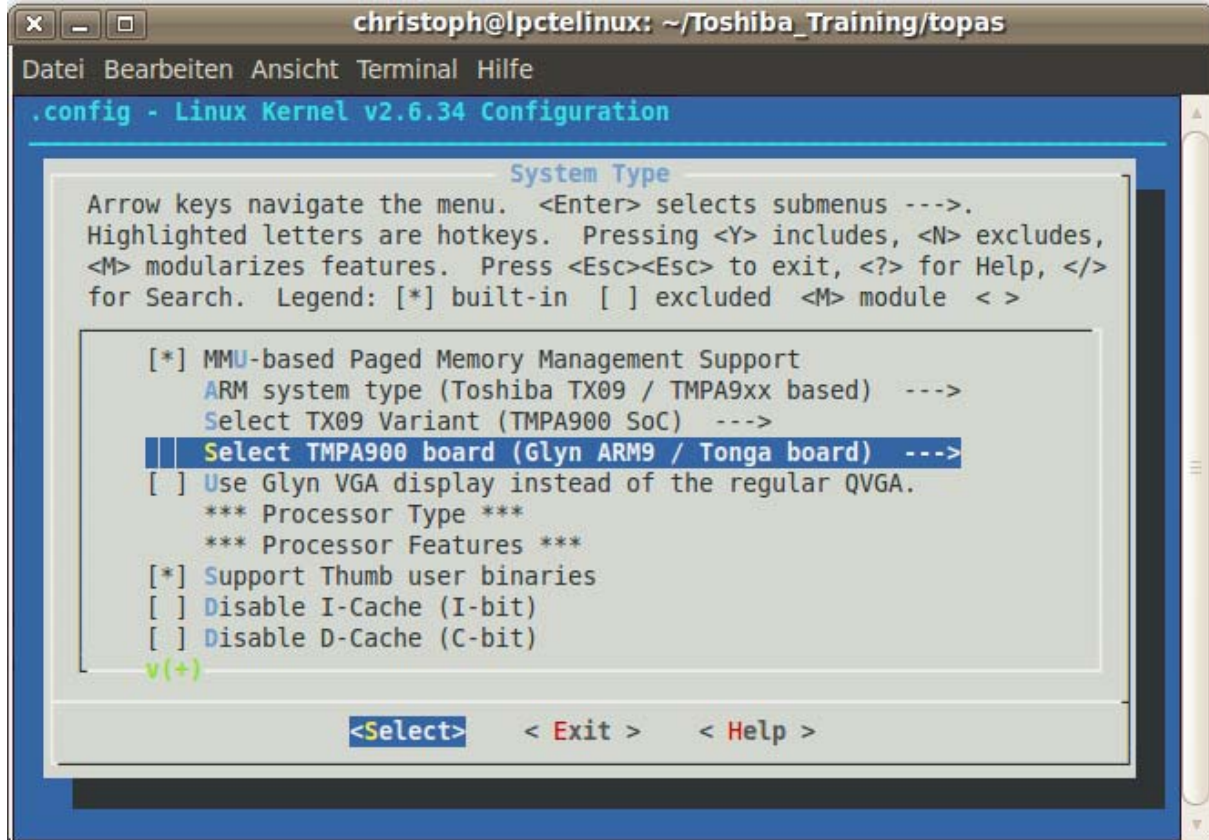
Configuration of the kernel means enabling or selecting certain features from the feature pool of the kernel. The kernel build uses a huge set of conditional compile switches to include or exclude specific features. Some of the selections are mandatory, like choosing the target platform. Others are optional, like including a certain set of device drivers. Finally, some of the selections create dependencies which must be resolved prior to compiling the kernel which has just been configured. To ease the setting of hundreds of different compile switches, a menu driven configuration system is provided.

Console Configuration Method

The menuconfig way of configuring a kernel is a console-based program that offers a way to move around the kernel configuration using the arrow keys on the keyboard.

To start this configuration, *first* you have to *copy the defconfig from arch/arm/configs/ tonga_defconfig to .config*

```
make ARCH=arm CROSS_COMPILE=/opt/mucross/arm/bin/arm-mucross-linux-gnueabi- menuconfig
```



Kernel features can either be compiled into the kernel or alternatively provided by a code module that can be loaded at kernel run-time. The method of providing kernel features as modules is commonly found with mainstream platforms. To avoid the need of compiling features into the kernel, making the kernel much bigger, a module approach is implemented here. The basic kernel just uses a minimum set of features and loads others on a dynamic basis.

Embedded systems mostly have well-defined interfaces – such as on-board or built-in devices, allowing the selection of these features at compile time.

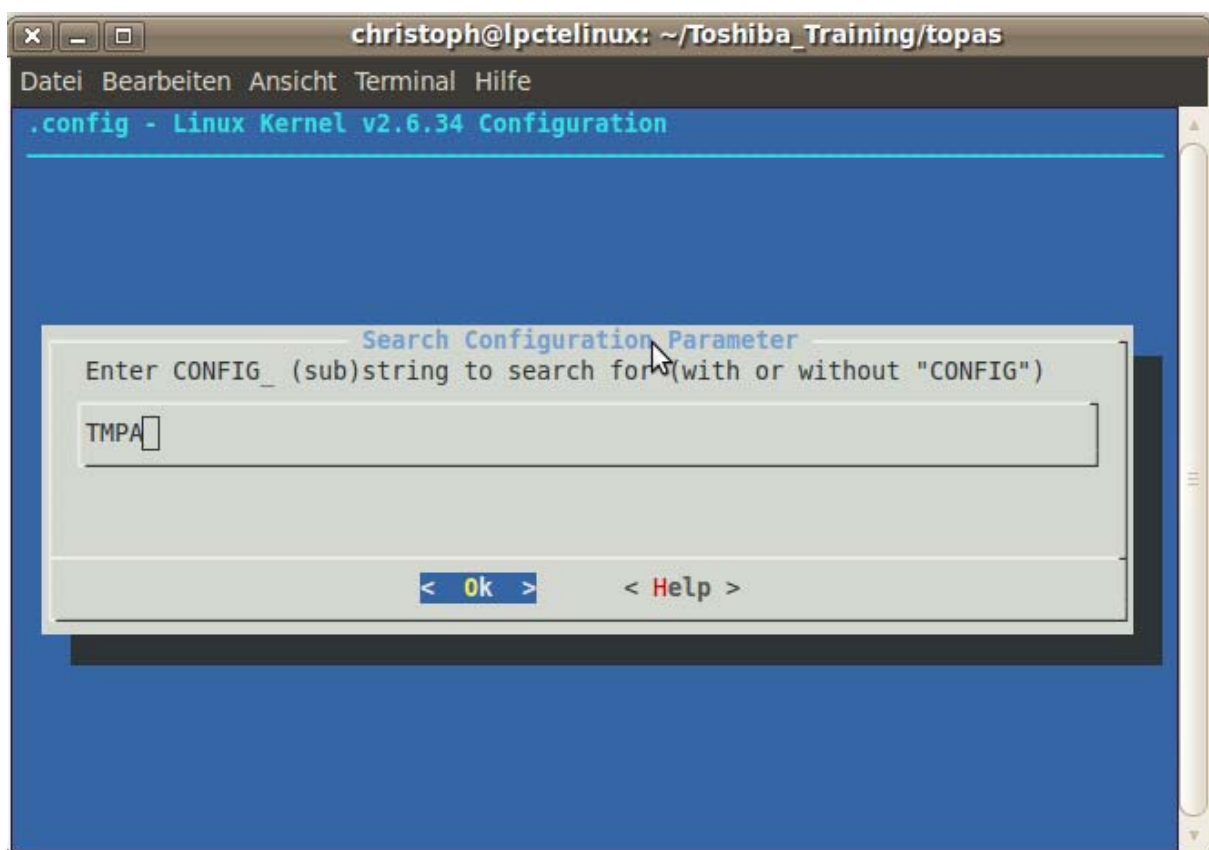
The menu configuration of the kernel reflects these differences as follows:

[] empty bracket at the feature location disables the feature

[*] the asterisk selects the feature as a built-in feature

[M] the letter 'M' selects the feature to be compiled as module

You can also search in the kernel configuration. Press “Shift” and “7”.



6.5.3 Compiling the Linux Kernel

After successfully configuring the kernel features, the kernel needs to be compiled. This is accomplished by simply typing 'make' on the command line. Using 'make' as the command will compile all parts of the kernel.

In order to compile a kernel for another platform, the desired architecture tool chain has to be given in the arguments.

- ARCH specifies the architecture. Each supported one is located in each entry of the directory arch/. **For ARM, we will use "arm"**
- CROSS_COMPILE specifies the tool chain to be used. For example: "arm-mucross-linux-gnueabi-"

Make examples:

- make ARCH=arm CROSS_COMPILE=/opt/mucross/arm/bin/arm-mucross-linux-gnueabi eabi- modules

Only selects the modules for compiling. You have to copy the build modules to your root file system.

- make ARCH=arm CROSS_COMPILE=/opt/mucross/arm/bin/arm-mucross-linux-gnueabi- ulmage

A kernel, adapted for u-boot of the current configuration

Important: To start compiling, *first* you have to *copy the **defconfig** from arch/arm/configs/ tonga_defconfig to .config*

6.5.4 Installing the Linux Kernel

After compiling the Linux kernel, it must be installed onto the target system. Therefore, we use our u-boot.

First copy the ulmage into your TFTP folder.

Press RESET on the base board. The following commands are re-entered via a serial terminal (115200/8/no/1/no Flow).

Install Kernel:

```
>tftp ulmage  
>nand erase kernel  
>nand write ${fileaddr} kernel
```


6.6 Linux File System

The following chapter is a short introduction to Linux for users who are using this operating system for the first time. There is a large amount of literature on this subject one can fall back on.

LINUX systems use a unified file system. Other than using different drive letters to identify drives and partitions, LINUX systems use a single starting point for the file system. This single point is marked with the '/' (slash). It is the starting point or 'root' of the file system. In this tree one can navigate with the command `cd`. It is always possible to type the path to a list or a file as absolute or relative to navigate and select a program.

There are no disk drive letters as known from other OS. Thus, the kernel (the real operating system) and the programs always "know" where certain lists (and with them the required files, like configuration files, libraries, program modules...) and certain resources are to be found.

After successful installation of the Linux kernels and a file system, you can connect to a serial console with the Linux board, e.g. the console answers after installation of the example

mucross-1.0-x11-gtk-qt4-image-tonga2-summary.jffs2

as follows:

Mucross Linux by kernel concepts
<http://www.mucross.com>
mucross@kernelconcepts.de
Mucross 1.0 tonga2 ttyS0

Created with Imagetool v1.0 tonga2
tonga2 login:

Type **root** followed by twice **cd..** . Now you are in the root file system.
With the command **ls** the structure of the system is shown.

The Root File System

The following files (or symbolic links to files) can be found in the root file system:

/						
bin/	dev/	home/	media/	proc/	sys/	usr/
boot/	etc/	lib/	mnt/	sbin/	tmp/	var/

/bin contains user programs that are essential to the system. This includes system shells - the command line processor of the system - as well as standard tools like:

- ls list directory content
- cp copy files
- mv move files
- rm delete (remove) files

/boot contains the LINUX kernel and boot loader

/dev device driver

/etc contains the LINUX system configuration files.

/home contains the LINUX system user home directories

/lib directory contains the LINUX system library files

/media Mounting point for temporary media.

/mnt optional, represents the LINUX system generic mount point.

/proc represents another special drawer of a LINUX system. Typical files and directories within this drawer are dynamically created by the kernel and its device drivers. These files are used to communicate internal system information.

/sbin is the system administrator's equivalent to the /bin drawer for normal users. Programs found at /sbin are used for system administration.

/sys mounting point for temporary media

/tmp directory contains temporary files of the current system.

/usr contains the user binaries - programs - of the LINUX System.

/var variable Data contains 'living' system data like LINUX System log files (/var/log/messages) or printer spooler data.

It makes sense to keep the root directory as small as possible. However, in general the principle is that application programs should not put directories in the root directory, but should revert to the given file structure

6.7 Small C-Examples under Linux

6.7.1 Linux “Hello World” Example

This section shows how to compile, download and run a simple “Hello, World” Linux application on the TMPA900-CPU-Board. First open an editor and write the following small program:

```
#include <stdio.h>  
  
int main (int argc, char **argv)  
{  
    printf("Hello World!\n");  
  
    return 0;  
}
```

Now that you have created a C "source" file - the human readable source for your program - it needs to be "compiled", i.e. turned into machine language that your CPU can actually use. There are two basic ways you can do that: use "gcc" (or "cc", which is usually the same thing) or "make".

First we want compile this for your „Host“ PC – means compiling for x86.

```
$ gcc -O2 -Wall -o hello-host hello.c
```

Now look in the folder with

```
$ ls -l
```

```
->
```

```
-rwxr-xr-x 1 user user 7149 2010-08-16 16:32 hello-host
```

With the command file we can see what kind of file hello-host is. You see it is an executable one.

```
$ file hello-host
```

```
hello-host: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically  
linked (uses shared libs), for GNU/Linux 2.6.15, not stripped
```

file is a standard program for determining the type of data contained in a computer file.

Now let's start the program:

```
$ ./hello-host  
Hello World!
```

Now we want to compile it for the ARM architecture:

```
$ ./opt/mucross/arm/environment-setup
```

```
$ arm-mucross-linux-gnueabi-gcc -O2 -Wall -o hello-arm hello.c
```

Now we have made a program for the target

```
$ file hello-arm  
hello-arm: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked  
(uses shared libs), for GNU/Linux 2.6.16, not stripped
```

Copy to target:

```
$ scp hello-arm root@"YOUR_TARGET_IP":
```

Run on target:

```
root@tonga2:~# ./hello-arm  
Hello World!
```

Working with Network Disk:

Copy the program you wish to test to the `/nfs_exchange/` directory. For example, the previously compiled helloworld.

```
> cp ~/src/simple/helloworld/helloworld /nfs_exchange/
```

6.7.2 IO-Toggle – Example for an easy accesses to the peripherals

First a short explanation of some needed commands:

Command system()

To begin another program from an executable program, the function **system ()** is available to you. Syntax:

```
#include <stdlib.h>  
int system(const char *kommandozeile);
```

Explanation: System () hand over a command line as a string. If the call could be explained successfully, the function returns a value incomparably to 0, otherwise 1. For the string you can give everything what is also permitted in the command line.

Echo

The echo command displays a message on the screen and is primarily useful for programmers writing shell scripts. But anyone can use echo to show the value of environment variables.

Here is the Listing of the small program:

```
#include <stdio.h>
#include <unistd.h>

void main(void)
{
    system("echo 8 >/sys/class/gpio/export");
    system("echo out >/sys/class/gpio/gpio8/direction")

    while (1)
    {
printf("Switching PB0 on\n");
system("echo 1 >/sys/class/gpio/gpio8/value");
    sleep(1);
printf("Switching PB0 off\n");
system("echo 0 >/sys/class/gpio/gpio8/value");
    sleep(1);
    }
}
```

Compile it for the ARM architecture:

```
$ ./opt/mucross/arm/environment-setup
```

```
$ arm-mucross-linux-gnueabi-gcc -O2 -Wall -o gpio_switch_arm gpio_switch.c
```

Copy to target or Network Disk -> Chapter 6.7.1.:

```
Run on target: root@tonga2:~# ./ gpio_switch_arm
```

Now PIN PB0 should toggle.

6.8 μ Cross – Linux Tool Package

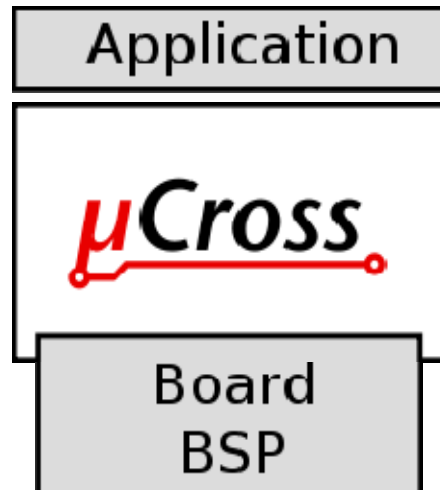
The file system and tool chain come from the μ Cross package. You can buy this package from kernel concepts - www.kernelconcepts.de

More information about μ Cross at: www.mucross.com

Why μ Cross?

μ Cross is a complete package to enable rapid project development. It contains a wide selection of pre-compiled packages that suit almost any requirements and is complemented by a matching cross development tool chain and SDK. A distinguishing feature of μ Cross is the support for graphical user interface (GUI) development using e.g. GTK+ or Qt - QT/embedded being supported as well. With tools known from desktop Linux, such as IDEs, user interface builders and debugging tools, an experienced GUI developer can start to develop embedded GUI applications within the shortest possible time.

μ Cross can be seen as the glue layer between the BSP and the customers specific application:



μ Cross complements the BSP that comes with the hardware board. Only the hardware specific parts of the BSP are needed, i.e. boot loader, kernel and possibly specific drivers. Everything else is supplied by μ Cross.

Components

µCross is based on a set of well-tuned and tailored components which form a complete and stable solution:

- Cross tool chain (GCC)
- SDK including GUI development, GTK+, Qt, Qt/embedded and DirectFB
- Root file system with package management
- Package feeds – the source for the pre-compiled binary packages which are used to build the root file system. Feeds can also be used as a source for post-deploy installations, more than 6000 packages are available
- Updates – a regular release schedule with new releases every six months
- Broad support offers - please enquire

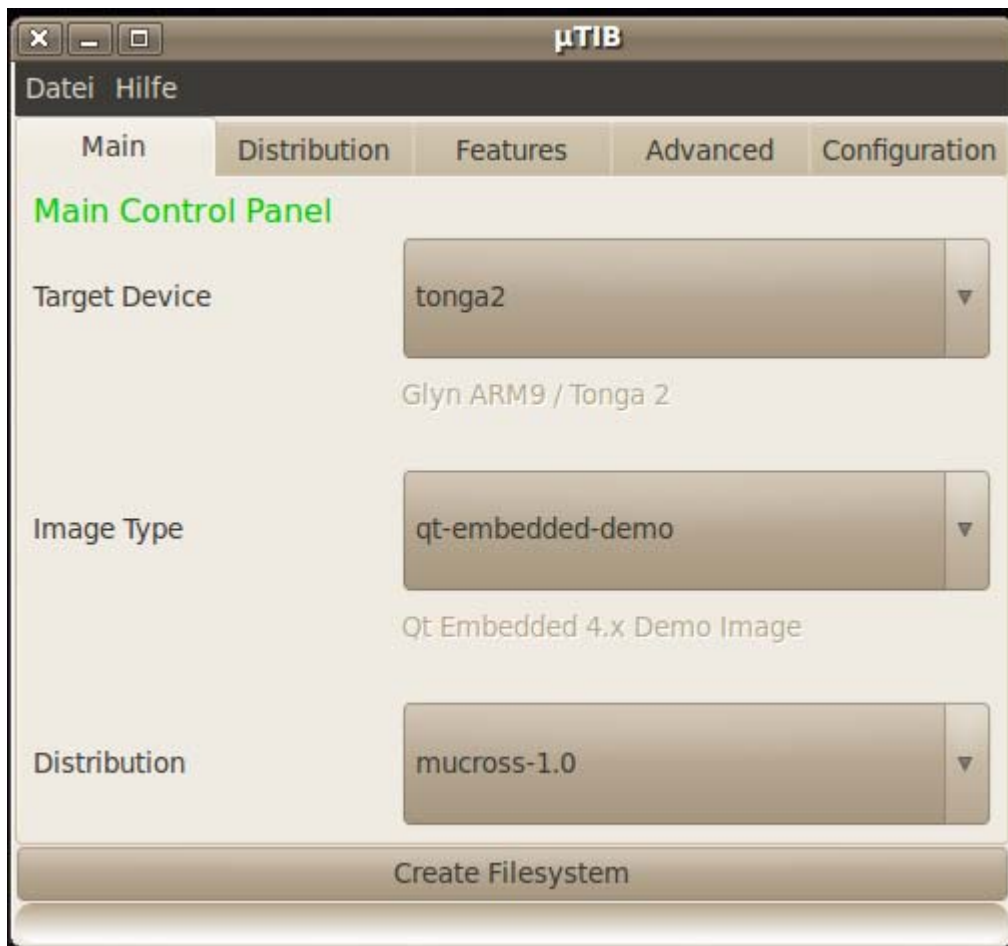
Services

- Complete development and runtime environment for embedded and mobile Linux devices
 - µCross tool chain: development tools for the target platform, consisting of compilers (C / C++), assembler and linker
 - µCross-SDK: complements the tool chain with libraries for application and GUI development
 - Documentation: **how to integrate the µCross-SDK into standard IDEs** such as Eclipse, Qt-Creator, Anjuta/Glade
 - µCross-Runtime: all packages are available both in development and runtime versions
- Wide range of packages
 - Building blocks: choose what is needed, leave out the rest
- Stable versions:
 - Once deployed, every version of µCross stays reproducible
 - So does the source
- µCross target image builder - creates the firmware flash image from the building blocks - on your development host
- Continuously improved - a new version every six months
- We support customer versions of µCross:
 - Complete customized version along with every main release
 - Or, with the customers approval, inclusion of a (hardware specific) subset into the main releases

µCross Target Image Builder - µTIB

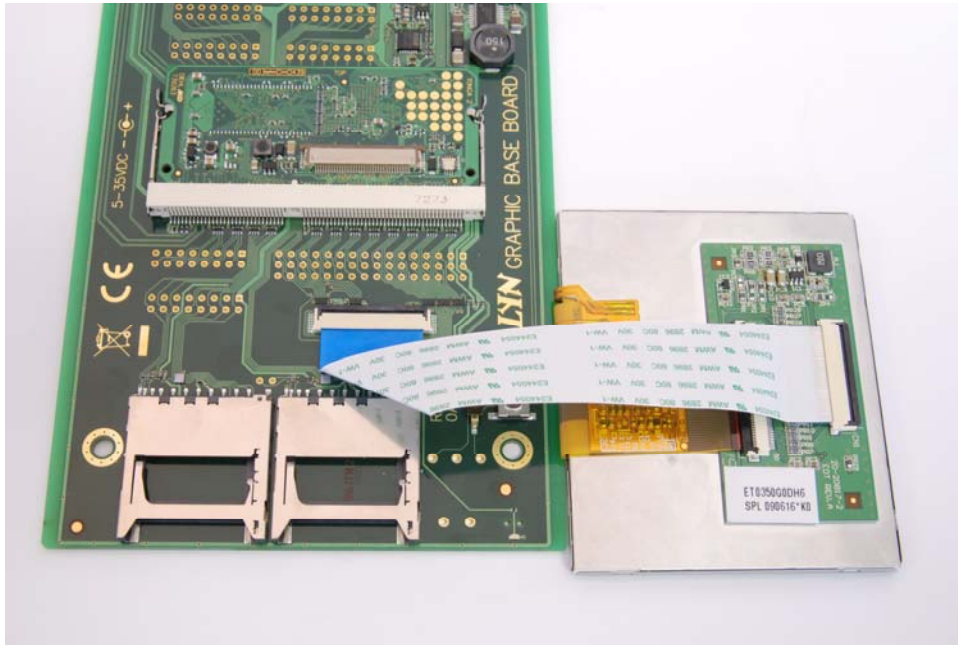
Features

- GUI
- Easy to use
- Creates file systems with a minimum of effort
- File system definition independent from target device (define once - deploy on multiple devices)
- Fast operation 1-2 minutes to build a file system
- Human readable and easy to modify device and file system descriptions
- Access to multiple configuration parameters
- Support for file system variants (e.g. debug and release)
- Output formats: TAR archives, JFFS2 images and UBIFS images
- Includes arbitrary files
- Runs arbitrary commands on boot
- Local operation possible (no network connection required)



7.0 Installing the Display with the Glyn Graphic Base Board

3,5", 4,3", 5.0", 7.0"



5,7"



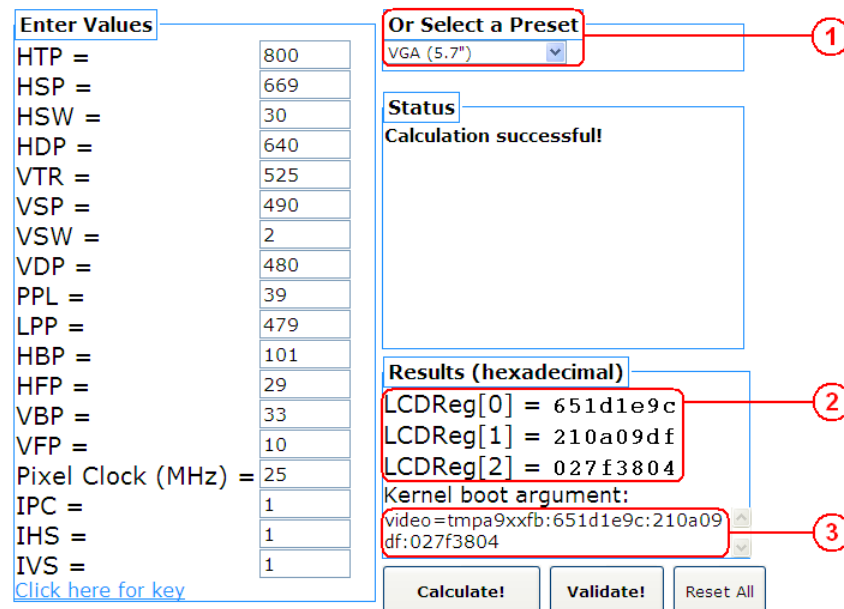
7.1 Other Resolutions/Other Timings – Calculation of the Display Settings

To ensure that bigger displays with other timings also work with our board, the kernel must be informed of the parameters, such as resolution. For this, we have a convenient tool available on our project homepage.

<http://www.mucross.com/downloads/tonga-demo/display-settings/>

Register-Value-Generator

NOTE You don't need to know all values!
Once you have provided enough input, the missing values will automatically be calculated.



Enter Values	
HTP =	800
HSP =	669
HSW =	30
HDP =	640
VTR =	525
VSP =	490
VSW =	2
VDP =	480
PPL =	39
LPP =	479
HBP =	101
HFP =	29
VBP =	33
VFP =	10
Pixel Clock (MHz) =	25
IPC =	1
IHS =	1
IVS =	1

[Click here for key](#)

Or Select a Preset: VGA (5.7")

Status: Calculation successful!

Results (hexadecimal):
 LCDReg[0] = 651d1e9c
 LCDReg[1] = 210a09df
 LCDReg[2] = 027f3804
 Kernel boot argument:
 video=tmpa9xxfb:651d1e9c:210a09df:027f3804

Buttons: Calculate! Validate! Reset All



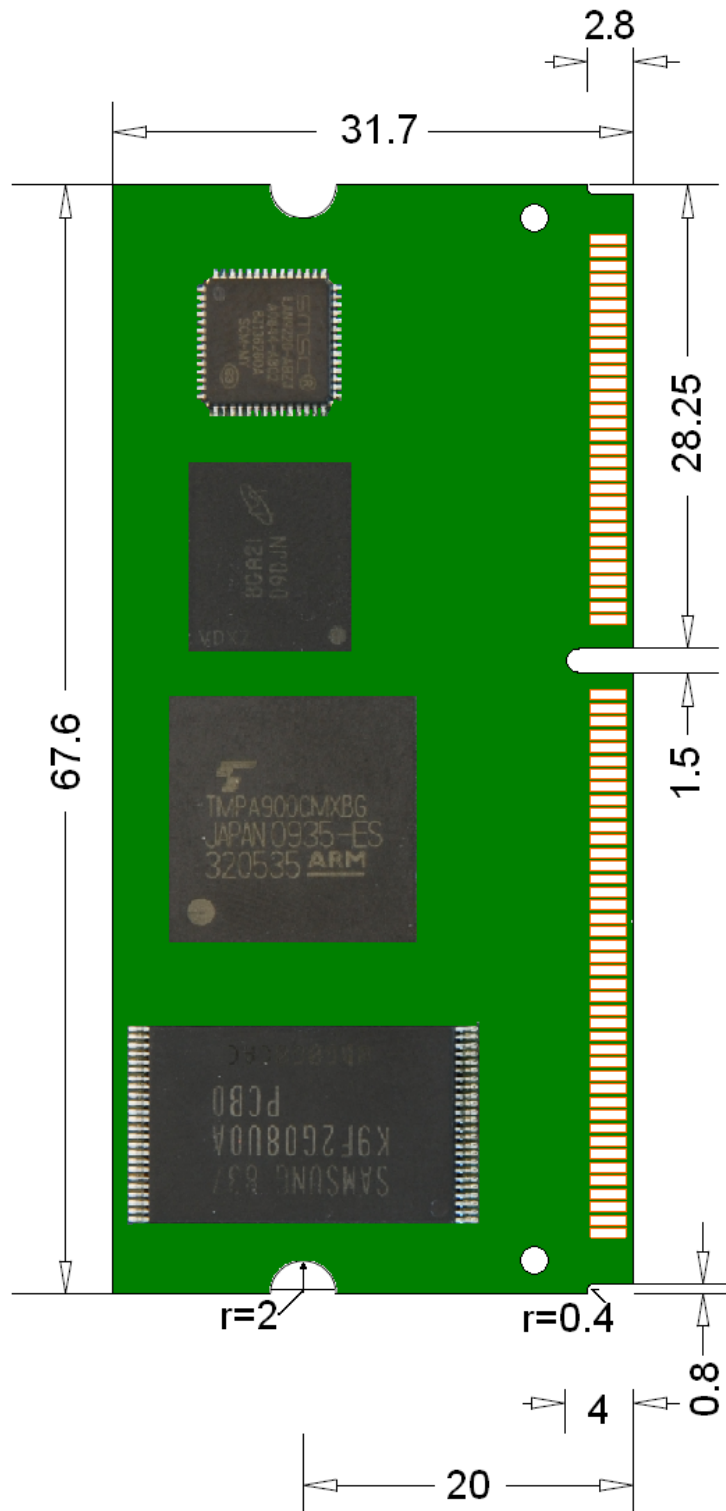
Please type in the timings of your display. For our displays of the EDT – Family Concept all parameters are given. You can find them at „Or Select a Preset“ ¹. After all timings have been entered, press the button “Calculate” and then all required register values will be calculated. The calculated values can be found under ².

Details of these registers can be found in the controller manual TMPA900CMXBG. Transfer the “kernel boot argument” ³ to the u-boot environment:

```
>setenv videoparams 'video=tmpa9xxfb:19211e4c:10040cef:013f380d'
```

End with **saveenv** !

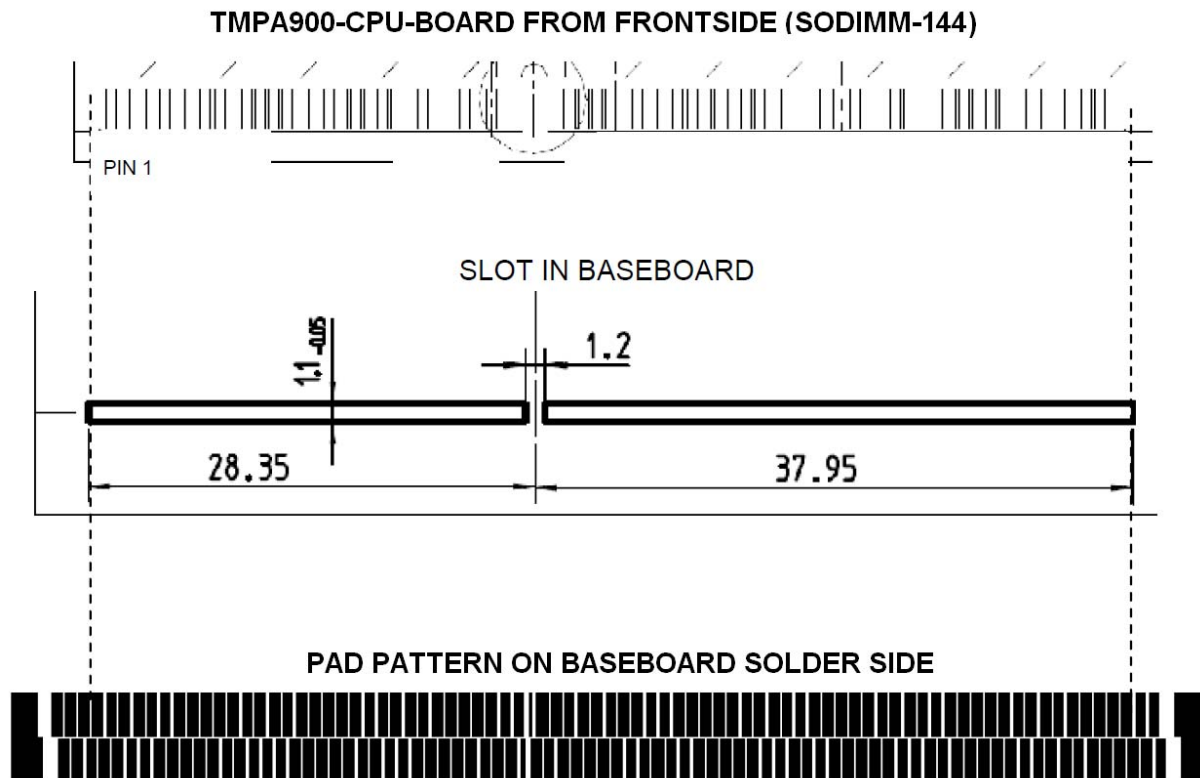
8.0 Mechanical Specifications (Formating)



8.1 Soldering the TMPA900-CPU-Board – No Connector

Mechanical Specifications (Formating)

Recommendations for Baseboard Slot for TMPA900-CPU-Board.



Dimension of PAD-Pattern: copper 0.6mm, space between copper 0.2mm

- Pads are intentionally made larger than the slot allows and will be cut by the PCB manufacturer during milling to prevent large gaps
- Additional PADS on both sides are needed to get a controlled, continuous solder flow.
- The radius for the milling tool of the slot is to be optimized. Two subsequent millings are recommended: one with 1.2mm and one with 0.8mm to get an accurate fit of the board without play.

These are recommendations only and must be optimized for individual solder stations.

Appendix A: Available u-boot Commands

?	- alias for 'help'
base	- print or set address offset
boot	- boot default, i.e., run 'bootcmd'
bootd	- boot default, i.e., run 'bootcmd'
bootm	- boot application image from memory
bootp	- boot image via network using BOOTP/TFTP protocol
chpart	- change active partition
cmp	- memory compare
coninfo	- print console devices and information
cp	- memory copy
crc32	- checksum calculation
dhcp	- boot image via network using DHCP/TFTP protocol
dynpart	- dynpart - dynamically calculate partition table based on BBT
echo	- echo args to console
editenv	- edit environment variable
exit	- exit script
false	- do nothing, unsuccessfully
fsinfo	- print information about filesystems
fsload	- load binary file from a filesystem image
go	- start application at address 'addr'
help	- print command description/usage
iminfo	- print header information for application image
imxtract	- extract a part of a multi-image
itest	- return true/false on integer compare
loadb	- load binary file over serial line (kermit mode)
loads	- load S-Record file over serial line
loady	- load binary file over serial line (ymodem mode)
loop	- infinite loop on address range
ls	- list files in a directory (default /)
md	- memory display

mdc	- memory display cyclic
mm	- memory modify (auto-incrementing address)
mtdparts	- define flash/nand partitions
mtest	- simple RAM read/write test
mw	- memory write (fill)
mwc	- memory write cyclic
nand	- NAND sub-system
nboot	- boot from NAND device
nfs	- boot image via network using NFS protocol
nm	- memory modify (constant address)
printenv	- print environment variables
rarpboot	- boot image via network using RARP/TFTP protocol
reset	- Perform RESET of the CPU
run	- run commands in an environment variable
saveenv	- save environment variables to persistent storage
saves	- save S-Record file over serial line
setenv	- set environment variables
showvar	- print local hushshell variables
sleep	- delay execution for some time
source	- run script from memory
test	- minimal test like /bin/sh
tftpboot	- boot image via network using TFTP protocol
true	- do nothing, successfully
ubi	- ubi commands
version	- print monitor version

Appendix B: Ordering Information

Starterkit: **TMPA900-CPU-BOARD-Starter**

The baseboard of the starter kits comes in a format 100 x 160 mm so it can be inserted into standard cases. Connection possibilities on the board are Ethernet, USB device and host, SD card (SPI & SD host) and UART. Another component of the starter kits is a QVGA display with touch screen.

- 1xTMPA900 CPU board
- 1xGlyn graphic base board
- 1xSegger - Jlink ARM Lite
- 1xQVGA-TFT EDT with touch
- 1xEthernet cable
- 2xUSB cable
- 1xserial cable
- 1xpower supply
- Software (partial eval - versions)

:

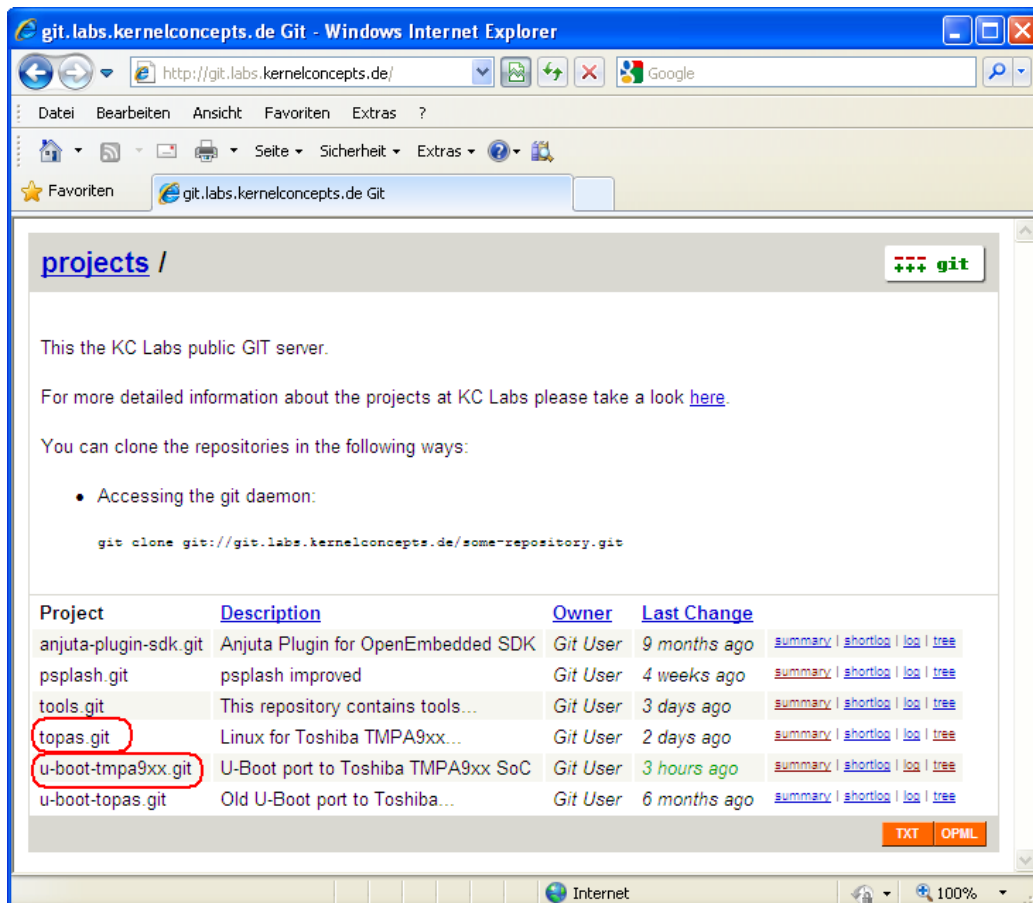
CPU Board: **TMPA900-CPU-BOARD**

- Processor TMPA900CMXBG, 200 MHz
- RAM 64 MB DDRRAM
- ROM 256 MB NAND Flash
- Power supply single 3.0V to 3.6V
- Size SO-DIMM 144
- Temp. range -20°C..85°C
- 10/100Mbps Ethernet (MAC+PHY)
- High speed USB 2.0 device (480Mbps)
- Full speed USB host 2.0 (12Mbps)
- LCD controller
- Interfaces: e.g. UART, SD-CARD, I2C, PWM, keypad, digital audio (I2S), 4/5 wire touch screen

Appendix C: KC Labs Public Git Server

You can find the sources of our Linux package and the u-boot on the KC Labs public GIT server:

<http://git.labs.kernelconcepts.de/>



Installing Git on Linux:

If you want to install Git on Linux via a binary installer, you can generally do it through the basic package management tool that comes with your distribution. E.g. on a Debian-based distribution like Ubuntu, try apt-get:

```
$ apt-get install git-core
```

Important: For the KC Labs public GIT server you have to open **TCP port 9418**.

If the port is not open, you will get an error message:
fatal unable to look up git.kernelconcepts.de (port9418) (Name or service not known)

Accessing the GIT daemon:

You can clone the repositories in the following ways:

Kernel:

```
git clone git://git.labs.kernelconcepts.de/topas.git
```

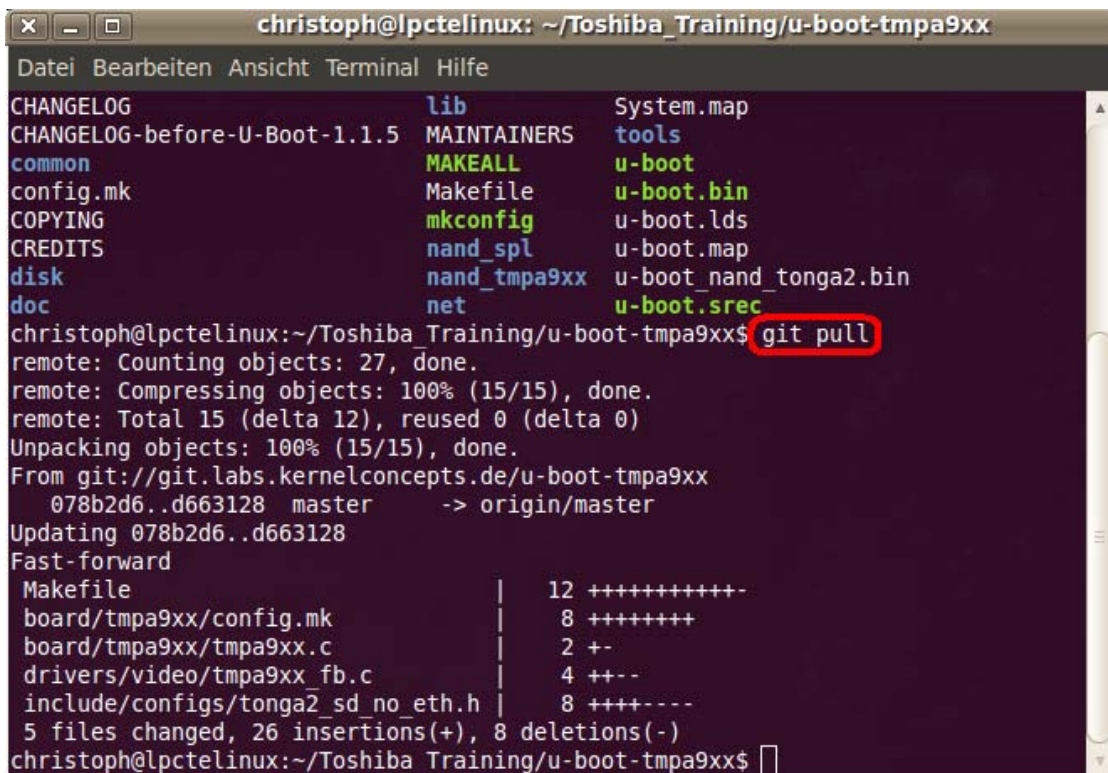
U-Boot:

```
git clone git://git.labs.kernelconcepts.de/u-boot-tpma9xx.git
```

Updates from the GIT server:

Once you have downloaded (clone), boot loader and/or kernel – you can update your sources easily -> go into the source folder and type in:

git pull



```

christoph@lpctelinux: ~/Toshiba_Training/u-boot-tpma9xx
Datei Bearbeiten Ansicht Terminal Hilfe
CHANGELOG          lib                System.map
CHANGELOG-before-U-Boot-1.1.5 MAINTAINERS       tools
common             MAKEALL           u-boot
config.mk          Makefile          u-boot.bin
COPYING            mkconfig          u-boot.lds
CREDITS            nand_spl          u-boot.map
disk               nand_tpa9xx      u-boot_nand_tonga2.bin
doc               net               u-boot.srec
christoph@lpctelinux:~/Toshiba_Training/u-boot-tpma9xx$ git pull
remote: Counting objects: 27, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 15 (delta 12), reused 0 (delta 0)
Unpacking objects: 100% (15/15), done.
From git://git.labs.kernelconcepts.de/u-boot-tpma9xx
 078b2d6..d663128 master -> origin/master
Updating 078b2d6..d663128
Fast-forward
 Makefile                | 12 ++++++++
 board/tpma9xx/config.mk |  8 +++++
 board/tpma9xx/tpma9xx.c |  2 +-
 drivers/video/tpma9xx_fb.c |  4 +++
 include/configs/tonga2_sd_no_eth.h |  8 +++++
 5 files changed, 26 insertions(+), 8 deletions(-)
christoph@lpctelinux:~/Toshiba_Training/u-boot-tpma9xx$

```

If necessary:

```
git reset --hard
```

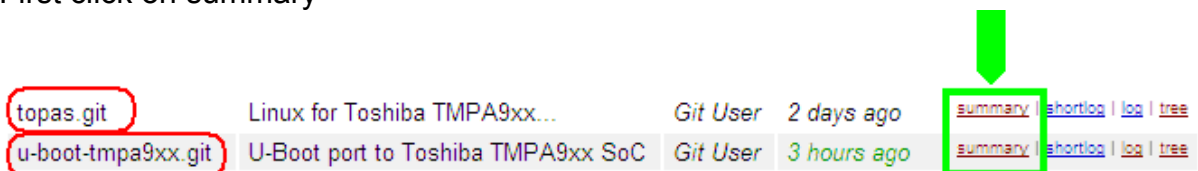
Snapshots

The major difference between Git and any other VCS (subversion and friends included) is the way Git thinks about its data. Conceptually, most other systems store information as a list of file-based changes.

Git doesn't think of or store its data in this way. Instead, Git thinks of its data more like a set of snapshots of a mini file system. Every time a new project state (u-boot / Kernel) is stored, it basically takes a picture of what all the files look like at that moment. To be efficient, if files have not changed, Git doesn't store the file again—just a link to the previous identical file it has already stored.

So you can also download a snapshot – for this you don't have to install GIT. But without GIT you have to download the whole project again every time something changes.

First click on summary



If you now click on snapshot, you can download the snapshot as a tar archive.



Documentation:

Details about Pro Git in the book from Scott Chacon on our CD (folder Pro Git) or at:

<http://labs.kernelconcepts.de/downloads/books/Pro%20Git%20-%20Scott%20Chacon.pdf>

Appendix D: Literature and References

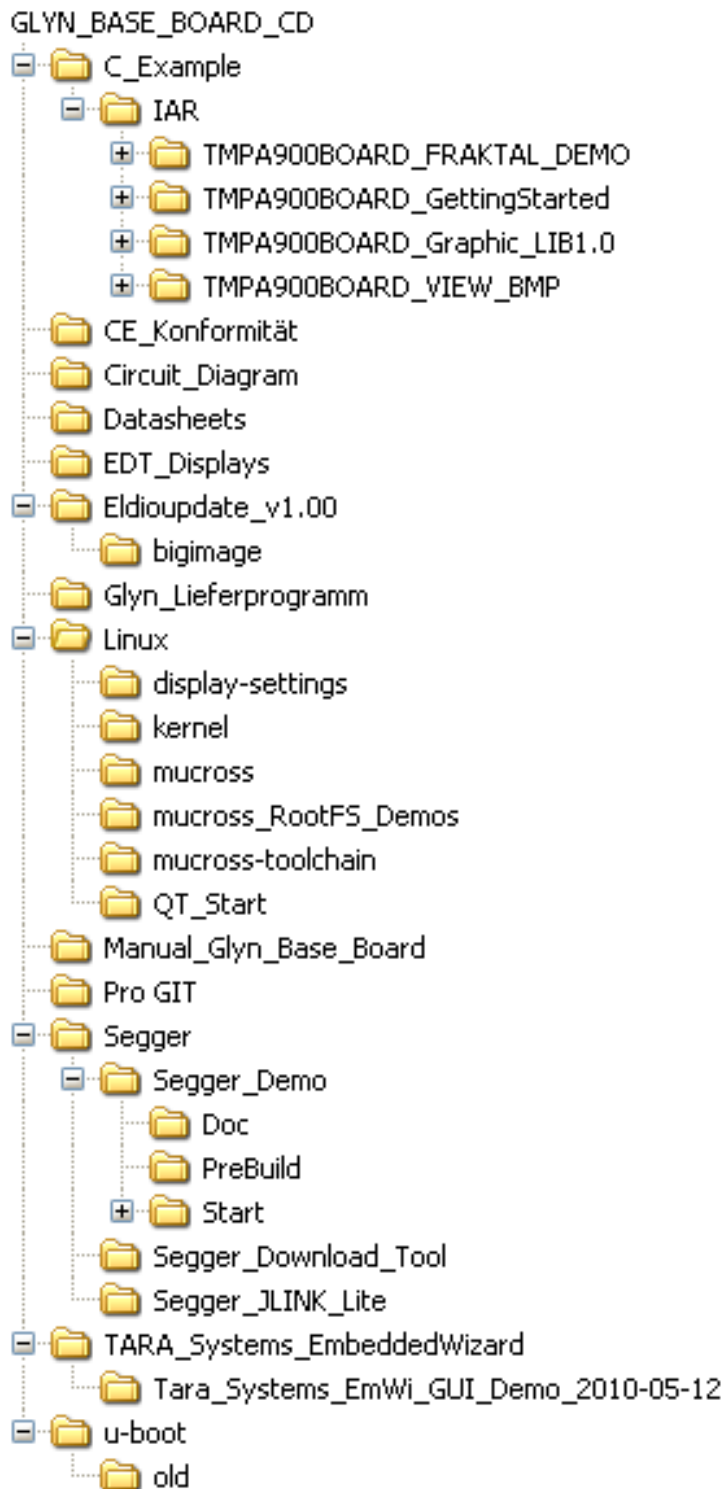
This Appendix lists documents and links, which we think may be useful to gain deeper understanding of technical details.

Field	Title	Comments
Hardware	User Manual TOSHIBA Original RISC 32-Bit Microprocessor ARM Core Family TMPA900CMXBG	Detail information about the TMPA900CMXBG and his peripherals. It is publicly available from Toshiba (www.toshiba-components.com)
Hardware	User Manual LAN9221/LAN9221i High Performance 16-bit NON-PCI 10/100 Ethernet Controller with Variable Voltage I/O	Detail information about the LAN9221/LAN9221i. It is publicly available from SMSC (www.smisc.com)
Hardware	User Manual WM8983 Mobile Multimedia CODEC with 1W Speaker Driver	Detail information about the Soundchip on the Starterki. It is publicly available from Wolfson (www.wolfsonmicro.com)
Hardware	Manual Samsung Flash Memory K9F2G08UXA	Detail information about the Nand-Flash on the CPU Board. It is publicly available from Samsung (www.samsung.com)
Hardware	Manual Samsung DDR RAM 32Mx16 Mobile DDR SDRAM	Detail information about the DDR RAM on the CPU Board. It is publicly available from Samsung (www.samsung.com)
Hardware	Sonitexx J19154-144	Drawing of Sonitexx J19154-144 SODIMM-144 Socket
Hardware	FPC Series ZIF for FFC / FPC Connector 0.5mm Pitch 90° SMT	Drawing of Displayconnector on the Starterkit from Yamaichi Electronics
Hardware	TFT Family Concept Compatible and Flexible - A cooperation between Glyn and EDT	Information about TFT Familyconcept. It is publicly available from Glyn (www.glyn.com)
Tools	SEGGER J-Link / J-Trace User's Guide.	This document gives information about using the SEGGER J-Link / JTrace ARM. It is publicly available from SEGGER (www.segger.com).
Software	embOS for ARM and IAR Embedded Workbench	This document gives information about using embOS for IAR EWARM. It is publicly available from SEGGER (www.segger.com).
Software	embOS/IP User Guide	This document gives information about using the SEGGER IP stack. It is publicly available from SEGGER (www.segger.com).
Software	User's and reference manual for emUSB	This document gives information about using the SEGGER USB stack. It is publicly available from SEGGER (www.segger.com).
Software	emFile User's Guide	This document gives information about using the SEGGER embedded filesystem. It is publicly available from SEGGER (www.segger.com).
Software GUI	User's and reference manual for emWin	This document gives information about using the SEGGER GUI software. It is publicly available from SEGGER (www.segger.com).

Note: Components on the board can change without notice!

Field	Title	Comments
Software Linux	µCross – The Innovative Distribution	This document gives information About µCross Linux based software-distribution. It is publicly available from kernelconcepts (www.mucross.com)
Software GUI	Creating Fantastic Graphical User Interfaces with Embedded Wizard	This document gives information about GUI development & prototyping suite “Embedded Wizard”. It is publicly available from Tara-Systems. (www.tara-systems.de)
Software GUI	Qt – cross platform application and UI framework	Homepage gives information, downloads for Qt (www.qt.nokia.com)
Software GUI	DirectFB is a thin library that provides hardware graphics acceleration. DirectFB adds graphical power to embedded systems and sets a new standard for graphics under Linux.	Homepage gives information, downloads for DirectFB (www.directfb.org)
Software GUI	GTK+ is a highly usable, feature rich toolkit for creating graphical user interfaces which boasts cross platform compatibility and an easy to use API.	Homepage gives information, downloads for GTK+ (www.gtk.org)
Software Linux	LINUX DEVICE DRIVERS by Author: Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman.	Book about writing device drivers for the Linux system. Third Edition Februar 2005, ISBN 978-0-596-00590-0, 636 Seiten (http://labs.kernelconcepts.de/Bookshelf/)
Software Management	Pro GIT Author: Scott Chacon	Book with information about GIT Server. (http://progit.org/) (http://labs.kernelconcepts.de/Bookshelf/)
Software Linux	LXR (formerly "the Linux Cross Referencer") is a software toolset for indexing and presenting source code repositories	Homepage with Linux Cross Reference (http://lxr.linux.no/+trees)
Linux	Linux-Kompendium	Online Source: http://de.wikibooks.org/wiki/Linux-Kompendium
Linux	Linux-Kompendium. Ubuntu / Arbeiten mit dem Terminal	Online Source: http://de.wikibooks.org/wiki/Linux-Kompendium:_Ubuntu/_Terminal
Linux	Running Linux under VMware Author: Bill Giannikos	Online Source: http://www.linwik.com/wiki/running+linux+under+vmware+workstation

Appendix E: CD file directory tree



Appendix F: Contact Information



GLYN GmbH & Co. KG
Head Office
www.glyn.de
sales@glyn.de

GLYN GmbH & Co. KG
Office Nettetal
www.glyn.de
nettetal@glyn.de

GLYN GmbH & Co. KG
Office Norderstedt
www.glyn.de
norderstedt@glyn.de

GLYN GmbH & Co. KG
Office Pforzheim
www.glyn.de
pforzheim@glyn.de

GLYN GmbH & Co. KG
Office Unterhaching
www.glyn.de
unterhaching@glyn.de

GLYN GmbH & Co. KG
Office Zirndorf
www.glyn.de
zirndorf@glyn.de

GLYN Austria
GLYN GmbH & Co. KG (Germany)
www.glyn.at
sales@glyn.at

GLYN Switzerland
GLYN GmbH & Co. KG (Germany)
www.glyn.ch
sales@glyn.ch

GLYN Benelux
GLYN GmbH & Co. KG (Germany)
www.glyn.nl
sales@glyn.nl

GLYN Poland
GLYN GmbH & Co. KG (Germany)
www.glyn.pl
sales@glyn.pl

GLYN Czech Republic
GLYN GmbH & Co. KG (Germany)
www.glyn.cz
sales@glyn.cz

GLYN Hungary
GLYN GmbH & Co. KG (Germany)
www.glyn.hu
sales@glyn.hu

GLYN Finland
GLYN GmbH & Co. KG (Germany)
www.glyn.fi
sales@glyn.fi

GLYN Sweden
GLYN GmbH & Co. KG (Germany)
www.glyn.se
sales@glyn.se

GLYN Denmark
GLYN GmbH & Co. KG (Germany)
www.glyn-nordic.dk
sales@glyn-nordic.dk

GLYN Norway
Link Electronics AS
www.linknordic.com
sales@linknordic.com

GLYN Bulgaria
Cooperations Partner
Universal 98 Ltd.
www.uni98-bg.com
sales@uni98-bg.com

GLYN U.K.
Cooperations Partner
First Byte Micro Ltd.
U.K. Head Office
www.firstbytemicro.com
sales@firstbytemicro.com

GLYN Ltd. Australia
www.glyn.com.au
sales@glyn.com.au

GLYN Ltd. New Zealand
www.glyn.co.nz
sales@glyn.co.nz