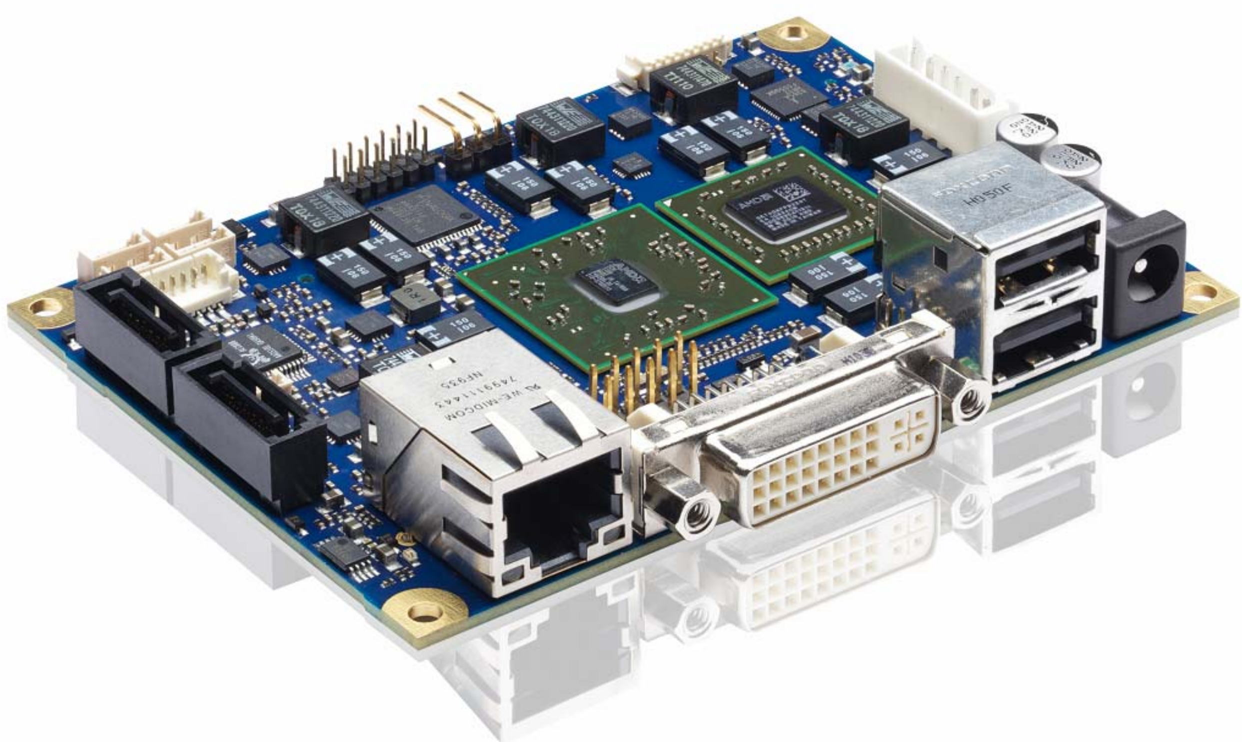
» Kontron Software Guide «



KTA55/pITX

KTD-S0043-A

» Table of Contents «

1 User Information

1.1 About This Document

This document provides information about products from KONTRON Technology A/S and/or its subsidiaries. No warranty of suitability, purpose or fitness is implied. While every attempt has been made to ensure that the information in this document is accurate the information contained within is supplied “as-is” - no liability is taken for any inaccuracies. Manual is subject to change without prior notice.
KONTRON assumes no responsibility for the circuits, descriptions and tables indicated as far as patents or other rights of third parties are concerned.

1.2 Copyright Notice

1.3 Trademarks

Brand and product names are trademarks or registered trademarks of their respective owners.

1.4 Standards

KONTRON Technology A/S is certified to ISO 9000 standards.

1.5 Warranty

This product is warranted against defects in material and workmanship for the warranty period from the date of shipment. During the warranty period KONTRON Technology A/S will at its discretion decide to repair or replace defective products.
Within the warranty period the repair of products is free of charge as long as warranty conditions are observed.
The warranty does not apply to defects resulting from improper or inadequate maintenance or handling by the buyer, unauthorized modification or misuse, operation outside of the products environmental specifications or improper installation or maintenance.
KONTRON Technology A/S will not be responsible for any defects or damages to third party products that are caused by a faulty KONTRON Technology A/S product.

1.6 Life Support Policy

KONTRON Technology's products are not for use as critical components in life support devices or systems without express written approval of the general manager of KONTRON Technology A/S.
As used herein:
Life support devices or systems are devices or systems which
a) are intended for surgical implant into body or
b) support or sustain life and whose failure to perform, when properly used in accordance with instructions for use provided in the labelling, can be reasonably expected to result in significant injury to the user.
A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

1.7 Technical Support

Please consult our web site at **http://www.kontron.com/support** for the latest product documentation, utilities, drivers and support contacts or use the special e-mail address **sbc-support@kontron.com** for a technical problem. In any case you can always contact your board supplier for technical support.
Before contacting support please be prepared to provide as much information as possible:
Board identification:

- Type
- Part number (find PN on label)
- Serial number (find SN on label)

Board configuration:

- DRAM type and size
- BIOS revision (find in the BIOS Setup)
- BIOS settings different than default settings (refer to the BIOS Setup section)

System environment:

- O/S type and version
- Driver origin and version
- Attached hardware (drives, USB devices, LCD panels ...)

2 BIOS Update

The update tools are available for three environments: DOS, EFI-Shell and Windows® (Windows® XP or Windows® 7 32/64 bit).

Note: If you update the Bios with the AFUDOS update tool please use the tool from the KTA55/pITX product website. To verify that the correct version is used check that the displayed version info matches the following terms:

AMI Firmware Update Utility(APTIO) v?.??
Copyright (C)2011 ... [or later]

Software conditions:

Operating System	Tool Name	Required Revision
DOS	AFUDOS.EXE (special version for EFI Bios)	APTIO 2.37 or greater
	BF.EXE	7.59 or greater
Windows® (Console)	BF.EXE	7.59 or greater
EFI-Shell	KTA55UPD.EFI	1.0 or greater

2.1 AFUDOS (AMI)

Only the following combination of command line arguments has been tested and should be used for the update process. The syntax of the DOS update tool is:

AFUDOS <Bios filename> **/X /P /B /N**

/X = do not check ROM ID
/P = program main Bios image
/B = program boot block
/N = overwrite NVRAM (setup settings)

2.2 KTA55UPD (Kontron)

This program is a EFI tool and works only in a EFI environment. You can use a FAT16/FAT32 formatted device (e.g. a USB key). Copy the update tool and the Bios to the root directory. Then boot the KTA55 board and press the **F7** key during the boot process (calls the boot menu). Choose the entry '**UEFI: Built-in EFI Shell**' and press Enter. After the Shell is active type '**fs0:**' and press Enter if you use a USB key - for other boot devices this identifier can be different. Now you can use KTA55UPD.EFI with the following syntax:

KTA55UPD <Bios filename>

2.3 BFLASH (Kontron)

In this manual the abbreviation **BF** is used for BFlash. This also matches with the actual name of the tool (BF.EXE). BF can be used to read and write data to and from BIOS flash. With this tool it is possible to update the BIOS, change DMI codes, setup vendor codes and save copies of all data. The copies can be used as master data for mass production.
Type **BF** <ret> from DOS prompt to see the BFlash version number and the board version. Only the following combination of command line arguments has been tested and should be used for the update process.

BF **read** <BIOS filename> **0** **400000**
BF **write** <BIOS filename> **0**

3 Custom Logo

With a EFI tool you can create your own custom logo. Use a FAT16/FAT32 formatted device (e.g. a USB key) and copy the logo tool and the image to the root directory. Then boot the KTA55 board and press the **F7** key during the boot process (calls the boot menu). Choose the entry '**UEFI: Built-in EFI Shell**' and press Enter. After the Shell is active type '**fs0:**' and press Enter if you use a USB key - for other boot devices this identifier can be different. You have two options:
change the logo in the ROM-file and update after that the Bios or
change the logo in the flash device.

First option:

KTA55LOGO **-i**<Bios filename> <Image filename>

-i = input. Change the logo in the ROM-file (input identical with output file)
Note: do not leave any space between '-i' and the Bios filename

Second option:

KTA55LOGO <Image filename>

3.1 Image Format

The following color depths are tested: 4 bit, 8 bit, 8 bit gray scale and 24 bit. As file format you can use the JPEG (file extension: JPG) or the Windows® Bitmap (file extension: BMP) format. The JPEG format does not support gray scale images. Only two resolutions are possible: VGA (640 x 480 pixel) and SVGA (800 x 600 pixel). For higher resolutions the image will be stretched to the display size. Summary:

- File format: JPEG or Windows® Bitmap
- Color depth: 4 to 24 bit
- Resolution: 640 x 480 or 800 x 600 pixel

To activate the logo function in the Bios Setup the entry '**Boot Menu/Enhanced Boot Settings/Dark Boot**' must be '**Enabled**'.

4 Graphics Interface

4.1 LCD/LVDS Technology Overview

4.1.1 Detailed Timing Descriptor (EDID or DisplayID™)

The input fields Pixel Clock, Horizontal Active, Horizontal Blank, Horizontal Sync Offset, Horizontal Sync Width, Vertical Active, Vertical Blank, Vertical Sync Offset and Vertical Sync Width must be filled in with the correct values according to the panels data sheet. In many cases the value for Horizontal/Vertical Blank cannot be read directly from the data sheet. Instead terms such as Display Period (active pixels/lines) or Horizontal/Vertical Total appear.
In this case the following calculation can be made:

⇒ **Blank Value = Total Value – Active Value.**

Sometimes the datasheet does not specify Sync Offset and/or Sync Width. In this case the permissible values can only be determined though testing. However the rule is:

⇒ **The sum of Sync Offset and Sync Width must not exceed the value for Horizontal/Vertical Blank.**

Also datasheets are often different for displays with double pixel clock. If Pixel Clock and Horizontal Values seem to be halved this must be corrected for input:

⇒ **The values must always be entered as though it were a panel with single pixel clock.**

Example 1:

PRIMEVIEW PM070WL4 (single pixel clock)
Data sheet specifications:

Clock Frequency [typ.]	32 MHz	
HSync Period [typ.]	1056 Clocks	(equivalent to Horizontal Total)
HSync Display Period [typ.]	800 Clocks	(equivalent to Horizontal Active)
HSync Pulse Width [typ.]	128 Clocks	
HSync Front Porch [typ.]	42 Clocks	
HSync Back Porch [typ.]	86 Clocks	
VSync Period [typ.]	525 Lines	(equivalent to Vertical Total)
VSync Display Period	480 Lines	(equivalent to Vertical Active)
VSync Pulse Width [typ.]	2 Lines	
VSync Front Porch [typ.]	10 Lines	
VSync Back Porch [typ.]	33 Lines	

Result:

Pixel Clock	32	
Horizontal Active	800	
Horizontal Blank	256	((128 + 42 + 86) → H. Pulse Width + H. Front Porch + H. Back Porch)
Horizontal Sync Offset	42	(H. Front Porch)
Horizontal Sync Width	128	(H. Pulse Width)
Vertical Active	480	
Vertical Blank	45	((2 + 10 + 33) → V. Pulse Width + V. Front Porch + V. Back Porch)
Vertical Sync Offset	10	(V. Front Porch)
Vertical Sync Width	3	(V. Pulse Width)

Example 2:

SHARP LQ190E1LW01 (double pixel clock)
Data sheet specifications (no definition of Sync Offset and Sync Width):

Clock Frequency [typ.]	54 MHz	
Horizontal Period (1) [typ.]	844 Clocks	(equivalent to Horizontal Total)
Horizontal Display Period	640 Clocks	(equivalent to Horizontal Active)
Vertical Period [typ.]	1066 Lines	(equivalent to Vertical Total)
Vertical Display Period	1024 Lines	(equivalent to Vertical Active)

Result:

Pixel Clock	108	(2 x 54 MHz)
Horizontal Active	1280	(2 x 640 Clocks)
Horizontal Blank	408	((844 – 640) x 2 Clocks)
Horizontal Sync Offset	45	(normally approx. 10 – 15 % of Horizontal Blank)
Horizontal Sync Width	140	(normally approx. 30 – 70 % of Horizontal Blank)
Vertical Active	1024	
Vertical Blank	42	(1066 – 1024 Lines)
Vertical Sync Offset	1	(normally approx. 1 – 3 Lines)
Vertical Sync Width	3	(normally approx. 1 – 15 Lines)

Example 3:

LG-PHILIPS LM170E01-TLA1 (double pixel clock)
Data sheet specifications:

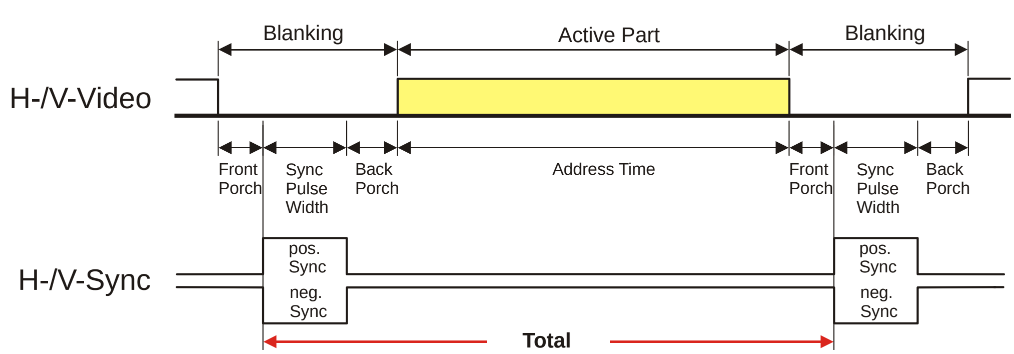
Clock Frequency [typ.]	54 MHz
Hsync Period [typ.]	844 Clocks
Horiz. Valid [typ.]	640 Clocks
Horiz. Back Porch [typ.]	124 Clocks
Horiz. Front Porch [typ.]	24 Clocks
Vsync Period [typ.]	1066 Lines
Vert. Valid [typ.]	1024 Lines
Vert. Back Porch [typ.]	38 Lines
Vert. Front Porch [typ.]	1 Line

Result:

Pixel Clock	108	(2 x 54 MHz)
Horizontal Active	1280	(2 x 640 Clocks → Horizontal Addr. Time)
Horizontal Blank	408	((844 – 640) x 2 Clocks)
Horizontal Sync Offset	48	(2 x 24 Clocks → Horizontal Front Porch)
Horizontal Sync Width	112	(((408/2 – 124 – 24) x 2) → H. Blank – H. Back Porch – H. Front Porch)
Vertical Active	1024	(Vertical Addr. Time)
Vertical Blank	42	(1066 – 1024 Lines)
Vertical Sync Offset	1	(Vertical Front Porch)
Vertical Sync Width	3	(Vertical Blank – Vertical Back Porch – Vertical Front Porch)

The following picture shows the typical video timing.

Timing Parameter Definitions



4.1.2 24 Bit Color Mapping Tips

The double pixel clock or 24-bit color depth can generally be taken from the datasheet. There are two interface modes existing at 24-bit color depth: **FPDI** (Flat Panel Display Interface) or **LDI** (LVDS Display Interface). Some panels use the line SELL LVDS (SELect Lvds data order). The LVDS data assignment in the datasheet can give you an indication by the last channel (e.g. RX3/TX3 – SELL LVDS = low) wether it is a LDI panel (contains the lowest bits). Most panels have a FPDI interface.

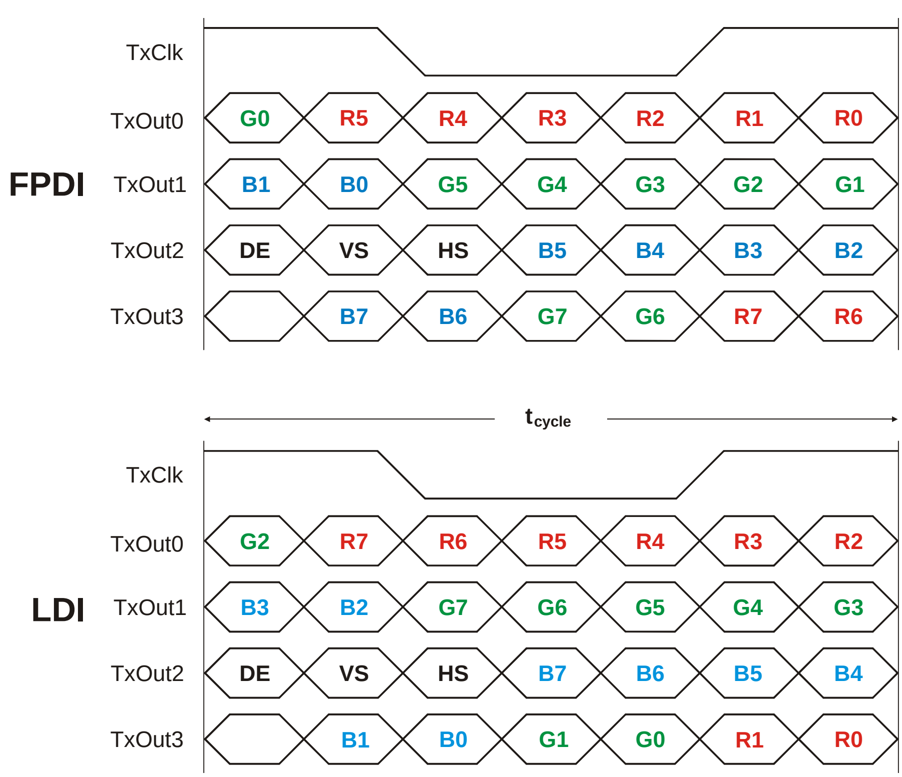
Example:

FPDI data assignment (LVDS channel 3 even or odd):

Tx/Rx27	Red 6 (e.g. even: RE6 or ER6)
Tx/Rx5	Red 7
Tx/Rx10	Green 6 (e.g. even: GE6 or EG6)
Tx/Rx11	Green 7
Tx/Rx16	Blue 6 (e.g. even: BE6 or EB6)
Tx/Rx17	Blue 7
Tx/Rx23	not used

LDI data assignment (LVDS channel 3 even or odd):

Tx/Rx27	Red 0 (e.g. even: RE0 or ER0)
Tx/Rx5	Red 1
Tx/Rx10	Green 0 (e.g. even: GE0 or EG0)
Tx/Rx11	Green 1
Tx/Rx16	Blue 0 (e.g. even: BE0 or EB0)
Tx/Rx17	Blue 1
Tx/Rx23	not used



4.2 EDID 1.3 Specification (VESA)

The EDID (Extended Display Identification Data) record has a fixed structure. The first 8 bytes contain the distinctive identification 00h, FFh, FFh, FFh, FFh, FFh, FFh, 00h. The end of the record is marked by the checksum (1 byte). The result of the addition of all bytes including the checksum has to be zero.
For a comprehensive support of the majority of available panels you do not need all fields of the EDID record. The **Detailed Timing Descriptor** (18 bytes) is the most important field. No 24bit panels (FPDI/LDI) are supported though. This means EDID should only be used for 18bit panels.
For further information please consult the official EDID specification from the VESA comitee which has to be payed.

4.3 DisplayID™ Specification (VESA)

Intended as a replacement for all previous EDID versions DisplayID™ contains many new features. It is a structure with several well defined elements (tags). Not every element that is listed in the specification has to be part of the resulting data set (basic section).
KONTRON has decided to use this selection of tags (mandatory presence).

Tag	Description
00h	Product Identification Data Block (Vendor ID, Product Code, Manufacturing Date ...)
03h	Type I Detailed Timing Data Block (Pixel Clock, Horizontal/Vertical Data ...)
0Ch	Display Device Data Block (Device Technology, Operating Mode, Color Depth ...)
0Dh	Interface Power Sequencing Data Block (Power On/Off Timing)
0Fh	Display Interface Data Block (Interface Type, Interface Attribute ...)

4.3.1 DisplayID™ Parameter Summary

Only a part of the parameters used in the DisplayID™ Windows® tool are interpreted by a specific board. The following table shows a summary of the used parameters (valid for KTA55/pITX).

Group	Parameter	Comment
Type I Timing	**Pixel Clock**	
Type I Timing	**Horizontal Active**	
Type I Timing	**Horizontal Blank**	
Type I Timing	**Horizontal Sync Offset**	Front porch
Type I Timing	**Horizontal Sync Width**	
Type I Timing	**Vertical Active**	
Type I Timing	**Vertical Blank**	
Type I Timing	**Vertical Sync Offset**	Front porch
Type I Timing	**Vertical Sync Width**	
Display Interface 1	**Bits per Pixel**	Color depth (18 or 24bit)
Display Interface 1	**24 Bit Color Mapping**	
Display Interface 2	**Signal Polarity**	Only H-Sync and V-Sync
Power Sequencing 1	**T2**	
Power Sequencing 1	**T3**	

Power Sequencing 1	**T5**	
Power Sequencing 1	**T6**	
Power Sequencing 2	**T4**	

4.3.2 LCD Panel Selection

The choice of an LCD display is basically defined by two parameters.

Parameter	Value
Pixel per Clock (Channels)	**1 or 2**
DisplayPort Interface	

Currently this leads to a maximum resolution of

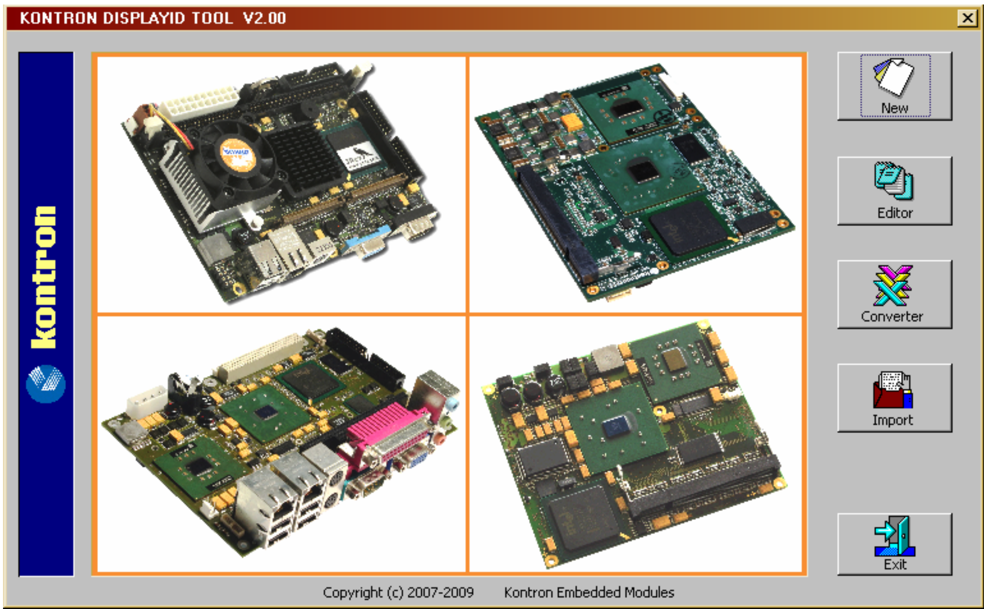
1920 x 1200 Pixel

With the AMD® graphic driver it is not guaranteed that every resolution can be achieved. KONTRON does not guarantee the correct function of the board for untypical resolution. In principal the use of DisplayID™ allows realizing every special display resolution. For this a valid DisplayID™ dataset must be written to the EEPROM of the DP/LVDS Monitor Controller.

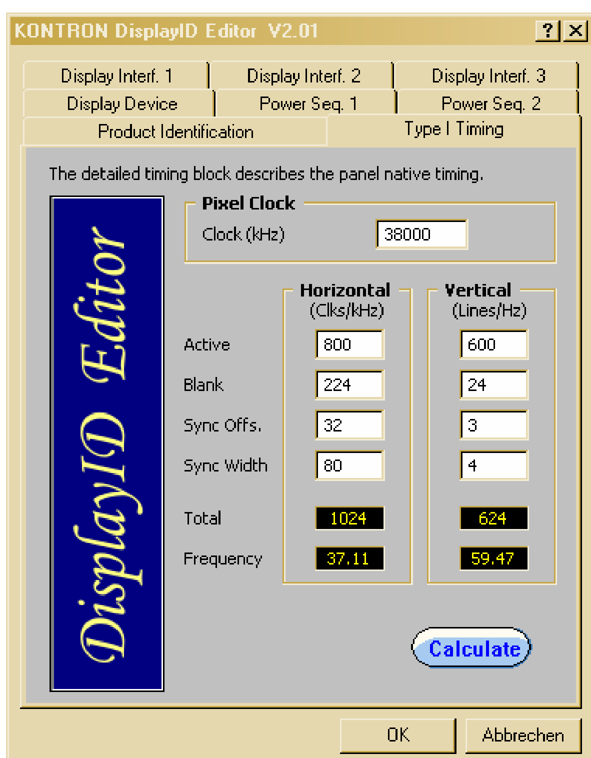
Many displays with a resolution up to XGA (1024 x 768) have a digital (TTL) interface. KONTRON offers a special adapter to connect these displays to the LVDS interface (KAB-ADAPT-LVDStoTTL with part number 61029).

4.3.3 DisplayID™ Windows® Tool

The DisplayID™ parameter can be modified with the DisplayID™ Windows® tool.



For an example the following picture shows the input fields for the **Detailed Timing** parameters.



For more information see the documentation of the DisplayID™ tool (software can be downloaded from kontron.com).

The DisplayID™ Editor saves the parameters in a intermediate file format. The file extension is 'KDD' (Kontron DisplayID™ Data). This file format cannot be used to program the EEPROM.
For transfering this file format into the binary file format for the EEPROM apply the Converter.

4.3.4 Building DisplayID™ File

❶ Start the Windows® tool **DisplayID.exe**.
❷ Use the **Editor** if you want to modify an existing DisplayID™ file or select **New** to create a complete new record.
❸ Change respectively enter new parameters.
❹ Save the parameters in a file with the extension 'KDD'.
❺ Open the saved 'KDD'-file using the **Converter**.
❻ Save the binary file with the extension 'KDB' (Kontron DisplayID™ Binary).
❼ Program the EEPROM using the board specific update tool.

4.3.5 EEPROM Panel File Structure

The KTA55/pITX can manage up to 16 separate panel entries. The EEPROM update tool makes it possible to overwrite each entry. In the Bios Setup choose the menu '**Advanced/Display Configuration/Digital Display Port 0**' and set it to '**LVDS**'. Now you can see two additional menus: '**LVDS Flat Panel Type**' and '**Brightness Level**'. The first menu allows the selection of the required panel.

Attention: *If you see only the labels 'Panel 0' to 'Panel 15' the EEPROM is not programmed.*

4.3.6 EEPROM Update Tool

You need a EFI tool to update the DisplayID™ record. Use a FAT16/FAT32 formatted device (e.g. a USB key) and copy the update tool and the record to the root directory. Then boot the KTA55 board and press the **F7** key during the boot process (calls the boot menu). Choose the entry '**UEFI: Built-in EFI Shell**' and press Enter. After the Shell is active type '**fs0:**' and press Enter if you use a USB key - for other boot devices this label can be changed.
The syntax of the EFI EEPROM update tool is:

KTA55DISP <Option> <Filename>

no option = read a file (KDB-format) and write the content to the EEPROM on panel position 0
-p<n> = read a file (KDB-format) and write the content to the EEPROM on panel position n
-b<val> = write a backlight brightness value (0 ... 255) to the EEPROM
-backup = read the content of the EEPROM and save it to a file (firmware included)
-update = read the previous backup file and write the content to the EEPROM
Note: do not leave any space between '-p' or '-b' and the parameter

Some examples:

KTA55DISP filename.kdb

Writes the DisplayID record to panel position 0 (default).

KTA55DISP -p3 filename.kdb

Writes the DisplayID record to panel position 3.

KTA55DISP -b64

Writes the backlight brightness value 64 (valid for all panel entries).

KTA55DISP -backup kta55.bin

Reads all panel records and the firmware of the DP/LVDS Monitor Controller. You can use this file to update a lot of boards with the same configuration.

KTA55DISP -update kta55.bin

Writes all panel records and the firmware to the DP/LVDS Monitor Controller (always in context with the backup function).

5 SDIO/microSD Card Interface

The following operating systems allow booting from a microSD card: DOS, Linux and Windows® XP Embedded Standard 7 (WES7).

5.1 DOS Boot

All standard DOS programs for partitioning and formatting can be used (e.g. FDISK.EXE and FORMAT.COM). However it can not be guaranteed that all functions of INT13h respectively INT21h work correctly.
Condition: the microSD card is formatted as a harddisk.

The table shows a list of the tested DOS versions.

Operating System	Version	Result
MS-DOS	**6.22**	✔
MS-DOS	**WIN 98 (7.10)**	✔
DR-DOS	**7.03**	✔

6 CPLD Interface

Various functions are implemented in the CPLD: e.g. the watchdog. Access to the CPLD register is via an index-data register pair using only two I/O byte locations (fixed addresses).

Index Register	Data Register
0xA80	0xA81

Register overview:

Index	Type	Reset	Function
0x00	RO	---	**CPLD version register**
0x01	RW	---	**System control register** Bit 0 Reserved Bit 1 Reserved
0x02	RO	---	**Reserved**
0x03	RO	---	**System status register** Bit 0 Recovery jumper 0 = not active 1 = active Bit 1 Autostart jumper 0 = active 1 = not active Bit 2 Reserved Bit 3 Reserved Bit 4 Reserved Bit 5 Last reset caused by watchdog 0 = no 1 = yes
0x04	RO	---	**Reserved**
0xA2	RW	0x00 1)	**Watchdog control register** Bit 0 - 1 Watchdog time value Bit 2 - 3 Reserved Bit 4 Watchdog time base 0 = seconds 1 = minutes Bit 5 Reserved Bit 6 Watchdog control 0 = disabled 1 = enabled Bit 7 Trigger control 0 = disabled 1 = enabled
0xA3	RO	---	**Watchdog timer register**

Note: *1) Default Setup settings.*

6.1 Reserved Bits

Every bit which is marked as **Reserved** may not be changed. Not observing this hint can in the worst case lead to system crashes.

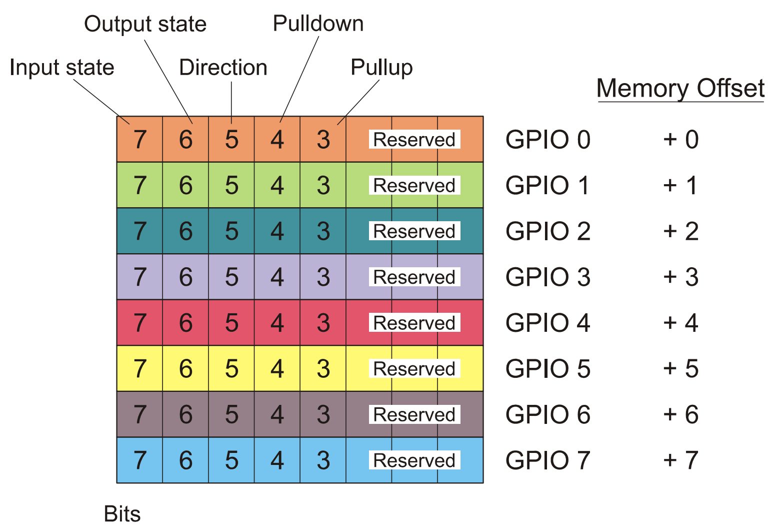
6.2 Watchdog Example

The following example (DOS program) show the access to the CPLD features (C compiler: BORLAND C++).
Note: These programs cannot be run on Linux and Windows®.

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

#define CPLD_BASE_ADDR          0xA80
#define WDT_INDEX               0xA2
#define WDT_TIMER               0xA3
#define WDT_ENABLE              0x40
#define WDT_TRIGGER             0x80

#define WDT_1SEC                0x00
#define WDT_5SEC                0x01
#define WDT_10SEC               0x02
#define WDT_30SEC               0x03
#define WDT_1MIN                0x10
#define WDT_5MIN                0x11
#define WDT_10MIN               0x12

void ActivateWatchdog (void)
{
   outp (CPLD_BASE_ADDR, WDT_INDEX);
   outp (CPLD_BASE_ADDR+1, WDT_10SEC);
   delay (1);                   // wait one millisecond
   outp (CPLD_BASE_ADDR+1, WDT_ENABLE | WDT_10SEC);
}

void TriggerWatchdog (void)
{
   outp (CPLD_BASE_ADDR, WDT_INDEX);
   outp (CPLD_BASE_ADDR+1, WDT_ENABLE | WDT_TRIGGER | WDT_10SEC);
   delay (1);                   // wait one millisecond
   outp (CPLD_BASE_ADDR, WDT_TIMER);
   inp (CPLD_BASE_ADDR+1);
}

void main (void)
{
   int i;

   ActivateWatchdog ();
   for (i = 0; i < 5; i++)      // wait half of expiry time (= 5 seconds)
      delay (1000);             // wait 1 second
   TriggerWatchdog ();          // trigger the watchdog - total expiry time now 15 seconds
}
```

7 GPIO Interface

The GPIO part is a component of the AMD® Hudson-E1 Controller Hub. Each GPIO pin has its own control register (memory-mapped, memory base address 0xFED80100). For customer usage only GPIO0 to GPIO7 are available - access to other GPIO pins is forbidden and can damage the board. Under a 16 bit environment register access is possible but very complex except you use the free Open Watcom C/C++ compiler (example compiled with version 1.9).
The following picture shows the register structure (8 bit wide):



<u>Open Watcom C/C++ compiler settings:</u>

Target Environment:	DOS - 32-bit
Image Type:	PMODE/W Executable [.exe]

7.1 DOS Example

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

#define u8                    unsigned char

#define GPIO_BASE_ADDR        0xFED80100
#define MAX_GPIO              8
#define PU_SHIFT              3
#define PD_SHIFT              4
#define DIR_SHIFT             5
#define OUT_SHIFT             6
#define IN_SHIFT              7
```

```
void WriteDigitalIO (u8 port, u8 value)
{
    u8 content, *ptr = (u8 *) GPIO_BASE_ADDR;

    ptr += port;
    content = *ptr;
    content &= ~(1 << OUT_SHIFT);
    *ptr = content | ((value & 1) << OUT_SHIFT);
}

u8 ReadDigitalIO (u8 port)
{
    u8 value, *ptr = (u8 *) GPIO_BASE_ADDR;

    ptr += port;
    value = *ptr;
    return ((value >> IN_SHIFT) & 1);
}

// Variable 'dir':
//    0 = Output
//    1 = Input
void SetDirectionIO (u8 port, u8 dir)
{
    u8 content, *ptr = (u8 *) GPIO_BASE_ADDR;

    ptr += port;
    content = *ptr;
    content &= ~(1 << DIR_SHIFT);
    *ptr = content | ((dir & 1) << DIR_SHIFT);
}

// Variable 'pu':
//    0 = Pullup enable
//    1 = Pullup disable
void SetPullupIO (u8 port, u8 pu)
{
    u8 content, *ptr = (u8 *) GPIO_BASE_ADDR;

    ptr += port;
    content = *ptr;
    content &= ~(1 << PU_SHIFT);
    *ptr = content | ((pu & 1) << PU_SHIFT);
}

// Variable 'pd':
//    0 = Pulldown disable
//    1 = Pulldown enable
void SetPulldownIO (u8 port, u8 pd)
{
    u8 content, *ptr = (u8 *) GPIO_BASE_ADDR;

    ptr += port;
    content = *ptr;
    content &= ~(1 << PD_SHIFT);
    *ptr = content | ((pd & 1) << PD_SHIFT);
}
```

```
void WriteGPIO (u8 value)
{
    u8 i;

    for (i = 0; i < MAX_GPIO; i++)
        WriteDigitalIO (i, (value >> i));
}

u8 ReadGPIO (void)
{
    u8 i, value = 0;

    for (i = 0; i < MAX_GPIO; i++)
        value |= ReadDigitalIO (i) << i;
    return value;
}

void InitializeGPIO (u8 dir, u8 pu, u8 pd)
{
    u8 i;

    for (i = 0; i < MAX_GPIO; i++)
        SetDirectionIO (i, (dir >> i));
    for (i = 0; i < MAX_GPIO; i++)
        SetPullupIO (i, (pu >> i));
    for (i = 0; i < MAX_GPIO; i++)
        SetPulldownIO (i, (pd >> i));
}

void main (void)
{
    InitializeGPIO (0x55, 0x00, 0x00);          // Pullup enable, Pulldown disable
    WriteGPIO (0x0A);
    getch ();
    WriteGPIO (0xA0);
    printf ("\nInput value = 0x%02X\n", ReadGPIO ());
}
```

7.2 Linux Example

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/mman.h>


int main (void)
{
    int i;
    int fd = open ("/dev/mem", O_RDWR);                // open device "mem"
    volatile unsigned char *ptr;

    if (fd < 0)                                        // is mem device present ?
    {
        printf ("\nCould not open memory device\n");
        return -1;
    }
    // map 4k page to address 0xFED80000
    ptr = (unsigned char*) mmap (NULL, 4096, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0xFED80000);
    if (ptr == MAP_FAILED)                             // did mapping failed ?
    {
        printf ("\nMapping failed\n");
        close (fd);
        return -1;
    }
    for (i = 0; i < 8; i++)
        printf ("GPIO%d pin status: %0x02X\n", i, ((ptr[0x0100+i] & 0x80) >> 7));    // read and mask GPIO register
    close (fd);
    return 0;
}
```

Note: You must have administrator rights to execute this example. You can also use 'sudo' if you have no administrator rights.

8 Thermal Management

The thermal management supports one mode: active cooling (no performance loss, but noise production).

8.1 Active Cooling

Only the AMD® Hudson-E1 Controller Hub controls the fan. The ACPI interface has no influence on the fan control - no 'Thermal Zone' part is implemented.
An Internal Micro-Controller (IMC) based on a 8051 core manages the temperature measurement and fan control. The Bios Setup makes it possible to define different fan control algorithm, see the submenu '**Advanced/Miscellaneous/Active Cooling Configuration**'. You can choose between the LINEAR MODE and the STEP MODE (entry '**Fan Speed Mode**'). The STEP MODE holds the fan speed constant within a temperature range. See the submenu '**Enhanced Fan Control**' where you can define maximal 8 steps.

9 ACPI Wakeup

One wake event is possible: Wake On LAN (WOL).

9.1 Wake On LAN

9.1.1 Linux (Debian distributions)

Linux normally disables the wake functionality. This can be verified within the terminal with

```
cat /proc/acpi/wakeup
```

The following command allows the activation of a wake function (example **USB0**):

```
echo USB0 > /proc/acpi/wakeup
```

10 UEFI Interface

The Unified Extensible Firmware Interface (UEFI) defines a software interface between an operating system and a specific hardware related firmware (Bios). UEFI encloses some data tables that contain platform information as well as boot and runtime services for the Operating System (OS) loader respectively the proper OS. From older Bios implementations the ACPI and the System Management Bios (SMBios) parts are overtaken.
The boot services are only available while the Bios owns the board (before invoking 'ExitBootServices'). This services part involves for an example text and graphical consoles as well as block and file operations. Runtime services are still accessible while the operating system is running, e.g. NVRAM access.
The KTA55/pITX Bios has only implemented the 64 bit UEFI version. The Bios and the OS must be size-matched, that means the board can only boot an original 64 bit UEFI-OS (needs a GUID Partition Table = GPT). Other operating systems are bootable via the Compatibility Support Module (CSM), e.g. 16 bit MS-DOS, 32 bit Windows® or 32 bit Linux.
Linux (64 bit) offers some kernel configurations for a UEFI system:

CONFIG_EFI=y
CONFIG_RELOCATABLE=y
CONFIG_FB_EFI=y
CONFIG_FRAMEBUFFER_CONSOLE=y
CONFIG_EFI_PARTITION=y
CONFIG_EFI_VARS=m (required to manipulate UEFI runtime variables using tools like efibootmgr)

10.1 EFI Developer Kit (EDK)

You can enhance the Bios functionality with your own UEFI applications. This is possible with Intels® EFI Developer Kit (EDK). The normal installation realizes only the 32 bit environment - you must add manually 64 bit C/C++ compiler and linker. Applications with 32 bit are not executable.
Likely you must change or add some entries in system files, e.g. 'LocalTools.env' or 'CommonTools.env', to enlarge the environment with 64 bit.

10.2 UEFI Boot

Boot the KTA55/pITX board and press the **F7** key during the boot process (calls the boot menu). Choose the entry '**UEFI: Built-in EFI Shell**' and press Enter. The **help** command lists all available UEFI applications.
For further information see the web page

http://software.intel.com/en-us/articles/efi-shells-and-scripting/

If you want a page-oriented display output use the option '**-b**' but not all commands/applications known this option. Another interesting option is '**-q**' to quit an application.
Kontron Technology A/S cannot guarantee that all UEFI commands/applications work properly.

Appendix A: Reference Documents

KONTRON Technology A/S cannot guarantee the availability of internet addresses.

Document	Internet Address
Advanced Configuration and Power Interface (ACPI)	http://www.acpi.info/spec.htm
AT Attachment Storage Interface Specification (ATA)	http://t13.org
Digital Visual Interface (DVI)	http://www.ddwg.org
High Definition Audio Specification (HD Audio)	http://www.intel.com/standards/hdaudio
High Speed Serialized AT Attachment (S-ATA)	http://www.sata-io.org/developers
IEEE 802.3 Specification (Ethernet)	http://standards.ieee.org/getieee802
Low Pin Count Interface Specification (LPC-Bus)	http://developer.intel.com/design/chipsets/industry/lpc.htm
Open LVDS Display Interface Spec. (Open LDI)	http://www.national.com/analog/displays/open_ldi
PCI Express Base Specification (PCI Express)	http://www.pcisig.com/specifications
SD Specification (SD Card)	http://www.sdcard.org/developers/tech/sdio/sdio_spec
System Management Bus Specification (SMBus)	http://www.smbus.org/specs
Universal Serial Bus Specification (USB)	http://www.usb.org/developers/docs
Unified Extensible Firmware Interface (UEFI)	http://www.uefi.org/specs
EFI Shells and Scripting	http://software.intel.com/en-us/articles/efi-shells-and-scripting

Appendix B: Document Revision History

Revision	Date	Author	Changes
S0043-A	05/24/12	MHU	Remove the designator 'Fusion'
S0043-0	03/08/12	IVO, MHU	Created preliminary manual

Corporate Offices

Europe, Middle East & Africa

Oskar-von-Miller-Str. 1
85386 Eching/Munich
Germany
Tel.: +49 (0)8165/ 77 777
Fax: +49 (0)8165/ 77 219
info@kontron.com

North America

14118 Stowe Drive
Poway, CA 92064-7147
USA
Tel.: +1 888 294 4558
Fax: +1 858 677 0898
info@us.kontron.com

Asia Pacific

17 Building,Block #1,ABP
188 Southern West 4th Ring Road
Beijing 100070, P.R.China
Tel.: + 86 10 63751188
Fax: + 86 10 83682438
info@kontron.cn