

MB86297A 'Carminé' Chip Initialization

© Fujitsu Microelectronics Europe GmbH

History

Date	Author	Version	Comment
03.06.2008	Anders Ramdahl	1.00	First release

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, **Application Notes**, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

0 Abstract

This document describes how to initialize the MB86297A graphics controller at system start-up.

1 References

Referenced documents:

CHWRM Fujitsu Microelectronics
MB86297A 'Carmine' Hardware Reference Manual
rev. 1.27, 27 May 2008

Additional information:

JESD79E JEDEC Standard, Double Data Rate (DDR) SDRAM Specification
Release E, May 2005
<http://www.jedec.org>

PCI3.0 Peripheral Component Interconnect Special Interest Group (PCI-SIG),
Conventional PCI 3.0
<http://www.pcisig.com>

K4H561638F Samsung Electronics
256Mb F-die DDR SDRAM Specification
revision 1.2, October 2004
<http://www.samsung.com/global/business/semiconductor/>

All Fujitsu Microelectronics documents listed above are available on the Fujitsu Graphics Solutions website:

<http://www.fujitsu.com/emea/services/microelectronics/displaycontrollers/>

2 Definitions

2.1 Numbers

Hexadecimal numbers Hexadecimal (base 16) numbers are denoted with prefix 0x, e.g. 0x2A is the hexadecimal representation of 42.

Optionally, underscores may be added for improved readability, e.g. 0xFEED_ABBA.

Binary numbers Binary (base 2) numbers are denoted with prefix 0b, e.g. 0b101010 is the binary representation of 42.

Optionally, underscores may be added for improved readability, e.g. 0b0010_1010.

All numbers are decimal (base 10) unless explicitly denoted otherwise.

2.2 Trademarks

Windows XP Windows and Windows XP are registered trademarks of Microsoft Corporation.
<http://www.microsoft.com>

3 MB86297A Initialization

The following steps are needed to get the MB86297A up and running:

1. PCI Initialization
2. Clock Initialization
3. DDR Controller Initialization
4. Loading firmware

These steps are described in detail in the following sub sections.

3.1 PCI Initialization

The first step of the MB86297A initialization is to configure the PCI interface. This initialization has to be done before the OS and the GDC driver is loaded.

The following registers of the PCI configuration space have to be initialized:

Address	Register	Comment
IDSEL+0x04	Command	Set bit 1 to enable access to memory space. Optionally, bits 6 and 8 can be set to enable PERR and SERR signaling when parity errors are detected.
IDSEL+0x10	BAR0	Internal registers base address. These registers are for debugging purposes only. See section 2.3.5 of the CHWRM for more details.
IDSEL+0x14	BAR1	Internal registers base address. These registers are for debugging purposes only.
IDSEL+0x18	BAR2	Video RAM base address. The DRAM Controller has to be initialized before this address is accessible. See section 3.3 for more details.
IDSEL+0x1C	BAR3	MB86297A registers base address. The clocks have to be initialized before this address space is fully accessible. See section 3.2 for more details.

In addition to the above, it may also be necessary to initialize the Interrupt Line register (address IDSEL+0x3C) for proper interrupt handling.

On a PC, the PCI initialization is normally handled by the BIOS.

For more information on PCI initialization, please refer to the PCI specification (PCI3.0).

3.2 Clock Initialization

Before accessing the different units of the MB86297A, the corresponding clocks have to be enabled. The clocks are individually enabled with the Clock Enable Register (address BAR3+0x0040_000C).

All the clocks are disabled after reset. Only the MB86297A control registers (addresses BAR3+0x0040_0000 through BAR3+0x004F_FFFF) are accessible at this point. The clocks of the other units have to be enabled before the unit is accessed.

A software reset is required after enabling the clocks.



Do not access registers in units whose clocks have not been enabled. Doing so could lead to system hang-up.

3.3 DDR Controller Initialization

The DDR Controller of the MB86297A has to be initialized before the video DRAM can be accessed. The following steps are required for proper initialization:

1. Set the I/O mode and drive strength of the DDR interface pads. This is controlled with the DRAM CTRL IO CTRL registers (address BAR3+0x30_0024).

Normally, drive strength 2 is recommended. However, using drive strength 1 may significantly improve the EMI characteristics of the DDR interface. Proper timing analysis is needed to check that the timing requirements of the DDR interface is fulfilled for the drive strength used.

2. Configure the DRAM controller parameters to suit the DDR SDRAM being used. The affected registers are listed in the following table:

Address	Register	Description
BAR3+0x30_0000	DRAM CTRL ADD	Controls how the linear address is coded into row, column and bank addresses. This has to be properly set up for 32 or 64 bit DDR SDRAM interface.
BAR3+0x30_0002	DRAM CTRL MODE	Controls timing parameters of the SDRAM interface. The OPM field must be set to 0b00_0010 in order to reset the DLL of the DDR SDRAM.
BAR3+0x30_0004	DRAM CTRL EMODE	Controls the extended mode of the DDR SDRAM memory. See the data sheet of the memory for more information.
BAR3+0x30_0006	DRAM CTRL TIME1	Controls timing parameters of the SDRAM interface.
BAR3+0x30_0008	DRAM CTRL TIME2	Controls timing parameters of the SDRAM interface.
BAR3+0x30_000A	DRAM CTRL REFRESH	Controls timing parameters of the SDRAM interface.
BAR3+0x30_0014	DRAM CTRL DDRIF1	Controls the output enables of the SDRAM interface. In 32-bit mode, the second clock output can be disabled in order to save power.
BAR3+0x30_0016	DRAM CTRL DDRIF2	Controls internal handling of CAS latency.

3. Issue an EMRS command to the DDR SDRAM. This is done by writing the value 0x0003 to the DRAM CTRL STATE register (address BAR3+0x30_000C).
This will also reset the DLL of the DDR SDRAM, since the OPM field of the DDR CTRL MODE register is set to 0b00_0010.
4. Wait until the DRAM controller returns to the START state, i.e. bits 3:0 of the DRAM CTRL STATE register becomes 0.
5. Set the OPM field of the DDR CTRL MODE register to 0b00_0000 to disable DLL reset.
6. Issue a MRS command to the DDR SDRAM. This is done by writing the value 0x0002 to the DRAM CTRL STATE register (address BAR3+0x30_000C).
Since the DLL reset has been disabled, the DRAM Controller will automatically transit to the HOME state once the MRS command is complete.
7. Wait until the DRAM controller enters its HOME state, i.e. bits 3:0 of the DRAM CTRL STATE register become 0b0100.
Once the DRAM controller has entered its HOME state, the DRAM controller is ready for use and the video DRAM can be accessed.



Do not access the video DRAM unless the DRAM Controller has been properly initialized. Doing so could lead to system hang-up.

This applies to PCI accesses as well as memory accesses from any unit of the MB86297A.



Do not modify the value of any reserved register bits. Doing so could lead to system hang-up and could even damage the SDRAM interface or the attached SDRAM devices.

3.4 Loading Firmware

For complete operation of the MB86297A, two separate pieces of firmware have to be loaded.

3.4.1 Loading Geometry Firmware

The geometry firmware is copied into the video DRAM while the geometry processor is being halted. Once the copy is complete, the

1. Execute a software reset. This resets and halts the geometry processor.
This step can usually be left out, since a software reset has already been done after the clock initialization.
2. Copy the geometry firmware into Video DRAM at address BAR2+0x000C_0000.
3. Start the geometry processor. This is done by clearing the FRHALT bit of the FRHALT register (address BAR3+0x0001_0038).
4. Wait until the geometry processor becomes idle. This is done by polling the ST bit of the FR_ST register (address BAR3+0x0001_8520).
A timeout should be implemented to detect if the geometry processor does not initialize properly.

3.4.2 Loading Pixel Firmware

The pixel firmware is loaded into internal memory of the MB86297A using the LoadFirm display list command.

The pixel processing firmware is a display list containing the required display list commands. This display list is sent to the display list processor by pushing it onto the display list command FIFO.

Appendix A Initializing the MB86297A PCI Evaluation Board

This appendix contains example code demonstrating how to initialize the MB86297A.

The code has been extracted from the Windows XP driver for the MB86297A PCI Evaluation Board.

The PCI initialization is not included in this appendix, since it is handled by the BIOS.

```

/*****
 * MB86297A initialization
 *****/

#include "gdcPixelFirm.h"
#include "gdcGeoFirmSC.h"

#define MB86290_GEO_FIRM_ADDR      (0x000C0000UL)

GDC_LONG init_mb86297()
{
    GDC_LONG ret;

    /* Initialize clocks */
    ret = init_clocks();
    if (ret != ENV_SUCCESS)
        return ret;

    /* Initialize DRAM controller */
    ret = init_dram_ctrl();
    if (ret != ENV_SUCCESS)
        return ret;

    /* Load geometry firmware */
    ret = load_geo_firm(gdcGeoFirmSC, sizeof(gdcGeoFirmSC),
                      MB86290_GEO_FIRM_ADDR);
    if (ret != ENV_SUCCESS)
        return ret;

    /* Load pixel firmware */
    ret = load_pixel_firm(gdcPixelFirm, sizeof(gdcPixelFirm));
    if (ret != ENV_SUCCESS)
        return ret;

    /* Success */
    return ENV_SUCCESS;
}

```

The routines used above are explained in the following sub sections.

A.1 Clock Initialization

The following routine enables all clocks of the MB86297A.

```
/* *****  
 * Clock initialization  
 * ***** */  
  
#define MB86290_CLOCK_ENABLE          (0x03FFUL)  
  
GDC_LONG init_clocks()  
{  
    /* Sets internal clock */  
    MB86290_WRITE_CTL_REGISTER(GDC_CTL_REG_CLOCK_ENABLE,  
                               MB86290_CLOCK_ENABLE);  
  
    /* Software reset */  
    MB86290_WRITE_CTL_REGISTER(GDC_CTL_REG_SOFTWARE_RESET, 1UL);  
    MB86290_WRITE_CTL_REGISTER(GDC_CTL_REG_SOFTWARE_RESET, 0UL);  
  
    /* Success */  
    return ENV_SUCCESS;  
}
```

A.2 DRAM Controller Initialization

The MB86297A PCI Evaluation Board is equipped with 128 MB of video DRAM, organized as 16Mx64 bits. The DDR SDRAM used is Samsung K4H561638F-UCCC, which has the following characteristics:

Organization: 16Mx16 (four parallel devices are used)
Clock frequency: 133 MHz
CAS latency: 2 (at 133 MHz)
Row address: A12 through A0
Column address: A9 through A0

The following routine initializes the DRAM controller accordingly.

```
/* *****  
 * DRAM Controller initialization  
 * ***** */  
  
#define MB86290_DCTL_ADD                (0x05C3UL)  
#define MB86290_DCTL_MODE                (0x0121UL)  
#define MB86290_DCTL_EMODE              (0x8000UL)  
#define MB86290_DCTL_SET_TIME1          (0x4749UL)  
#define MB86290_DCTL_SET_TIME2          (0x2A22UL)  
#define MB86290_DCTL_REFRESH            (0x0042UL)  
#define MB86290_DCTL_STATES              (0x0003UL)  
#define MB86290_DCTL_RESERVE0           (0x0020UL)  
#define MB86290_DCTL_FIFO_DEPTH         (0x000FUL)  
#define MB86290_DCTL_RESERVE2           (0x0000UL)  
#define MB86290_DCTL_DDRIF1             (0x6646UL)
```

```

#define MB86290_DCTL_DDRIF2          (0x0055UL)
#define MB86290_DCTL_MODE_AFT_RST   (0x0021UL)
#define MB86290_DCTL_STATES_AFT_RST (0x0002UL)
#define MB86290_DCTL_IO_CONT0       (0x0555UL)
#define MB86290_DCTL_IO_CONT1       (0x0555UL)

GDC_LONG init_dram_ctrl()
{
    GDC_LONG ret;

    /* Set drive strengt of SDRAM interface pads */
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_IOCONT1_IOCONT0,
                                MB86290_PACK(MB86290_DCTL_IO_CONT0,
                                                MB86290_DCTL_IO_CONT1));

    /* Set up memory parameters */
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_MODE_ADD,
                                MB86290_PACK(MB86290_DCTL_MODE,
                                                MB86290_DCTL_ADD));
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_SETTIME1_EMODE,
                                MB86290_PACK(MB86290_DCTL_SET_TIME1,
                                                MB86290_DCTL_EMODE));
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_REFRESH_SETTIME2,
                                MB86290_PACK(MB86290_DCTL_REFRESH,
                                                MB86290_DCTL_SET_TIME2));
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_RSV2_RSV1,
                                MB86290_PACK(MB86290_DCTL_RESERVE2,
                                                MB86290_DCTL_FIFO_DEPTH));
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_DDRIF2_DDRIF1,
                                MB86290_PACK(MB86290_DCTL_DDRIF2,
                                                MB86290_DCTL_DDRIF1));

    /* Issue EMRS command to SDRAM
    *
    * Since the operating mode was previously set to 0b000010, this
    * command will also reset the DLL of the SDRAM.
    */
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_RSV0_STATES,
                                MB86290_PACK(MB86290_DCTL_RESERVE0,
                                                MB86290_DCTL_STATES));

    /* Wait until DRAM Controller returns to the START state */
    ret = MB86290_WAIT_DCTL_STATE(0x0);
    if (ret != ENV_SUCCESS)
        return ret;

    /* Disable DLL reset */
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_MODE_ADD,
                                MB86290_PACK(MB86290_DCTL_MODE_AFT_RST,
                                                MB86290_DCTL_ADD));

    /* Issue RMS command to SDRAM
    *
    * Since the operating mode has been set to 0b000000, the DRAM
    * controller will switch to HOME (active) state when the
    * command has finished.
    */
    MB86290_WRITE_DCTL_REGISTER(MB86290_DCTL_REG_RSV0_STATES,
                                MB86290_PACK(MB86290_DCTL_RESERVE0,
                                                MB86290_DCTL_STATES_AFT_RST));
}

```

```

    /* Wait until DRAM Controller enters the HOME state */
    ret = MB86290_WAIT_DCTL_STATE(0x4);
    if (ret != ENV_SUCCESS)
        return ret;

    /* Success */
    return ENV_SUCCESS;
}

```

A.3 Loading Geometry Firmware

The firmware is copied into the video DRAM while the geometry processor is halted. Once the firmware has been copied, the geometry processor is activated.

This routine relies on the geometry processor already being reset and halted by the software reset in the clock initialization routine, see section A.1.

```

/*****
 * Load geometry firmware
 *****/

GDC_LONG load_geo_firm(const GDC_ULONG * const firm,
                      const GDC_ULONG size,
                      const GDC_ULONG offset)
{
    GDC_ULONG i;
    GDC_ULONG state;
    GDC_ULONG timeout;

    /* Copy firmware to VRAM */
    for (i = 0UL; i < size/sizeof(GDC_ULONG); i++)
        MB86290_WRITE_VRAM(offset + i*sizeof(GDC_ULONG), firm[i]);

    /* FR80 start */
    MB86290_WRITE_GRAPH_REGISTER(GDC_GRAPH_REG_FRHALT, 0UL);

    /* Wait until status is idle */
    timeout = 1000;

    do
    {
        MB86290_READ_GRAPH_REGISTER(GDC_GRAPH_REG_FR80ST, &state);
        if ( (state != 0) && (timeout > 0) )
        {
            WAIT_MS(1);      /* Wait for 1 ms */
            timeout--;
        }
    } while ( (state != 0) && (timeout > 0) );

    /* Check if timeout */
    if ( (state != 0) && (timeout == 0) )
        return ENVERR_DEVICE_NOT_READY;

    /* Success */
    return ENV_SUCCESS;
}

```

A.4 Loading Pixel Firmware

The pixel firmware is loaded via the LoadFirm display list command. The firmware is stored as a display list that is pushed into the display list command FIFO.

```

/*****
 * Load pixel firmware
 *****/

GDC_LONG load_pixel_firm(const GDC_ULONG * const firm,
                        const GDC_ULONG size)
{
    GDC_ULONG    i;
    GDC_LONG     ret;

    /* Push the firmware onto the DL FIFO */
    for (i = 0UL; i < size/sizeof(GDC_ULONG); i++)
    {
        ret = MB86290_PUSH_DL_FIFO(firm[i]);
        if (ret != ENV_SUCCESS)
            return ret;
    }

    /* Success */
    return ENV_SUCCESS;
}

```