CREMSON

MB86290A Graphics Display Controller

Application Note
January 7 ‘00 (Rev.1.0)

FUJITSU

Overview

This manual describes about the application examples utilizing the Cremson graphics display controller and its driver software. Schematic diagram examples and software program examples put in this application note are for the reference of actual end product designs.

All these examples describes in this application note are just for the purpose of reference example and actual design evaluation must be done at each application design prior to the real system development.

Index

1 External Interface Connection Examples

1.1 Host CPU Interface

Cremson is capable to interface to Hitachi SH3/SH4 or NEC V832 CPU without external glue logic by the MODE pin settings

1.1.1 Connection to SH3, SH4

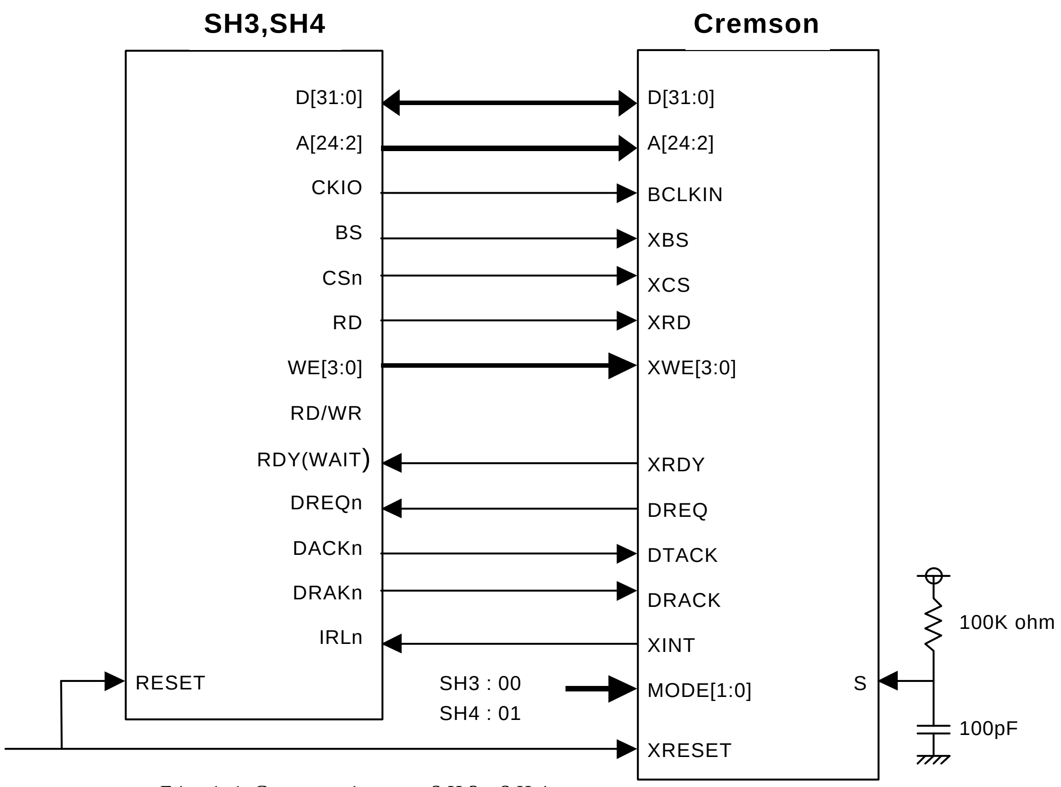

Fig.1.1 Connection to SH3, SH4

- The external RC shown above is to input a 500nsec or longer low pulse to S pin. XRESET input must be held in "L" for more than 300usec after the S pin input goes to "H". The same system RESET signal is recommended to input to both CPU RESET pin and Cremson XRESET pin. (SH3 and SH4 CPUs require a minimum 10msec of "L" level input to the RESET pin at power on reset sequence.)
- DREQ signal needs to be set as low active input. DRACK and DACK signals need to be set as high active output.
- When CS is not asserted, XRDY outputs Hi-Z level. So XRDY is recommended to cramp to low level (pull-down) in SH4 connection and high level (pull-up) in SH3 connection by an external resistor.
- To activate a wait state insertion by XRDY signal, for the Cremson mapping address area in the WCR2 register, insert wait state is set to 1.

1.1.2 Connection to V832

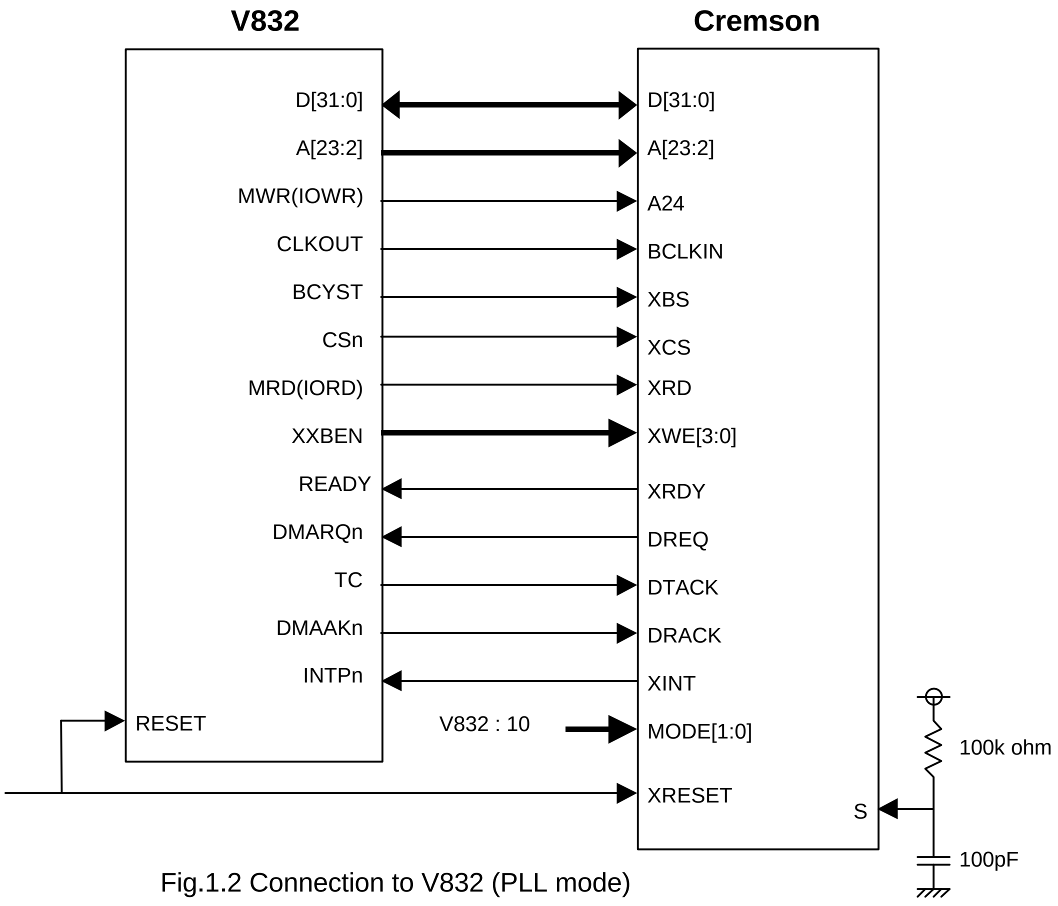

Fig.1.2 Connection to V832 (PLL mode)

- The external RC shown above is to input a 500nsec or longer low pulse to S pin. XRESET input must be held in "L" for more than 300usec after the S pin input goes to "H". The same system RESET signal is recommended to input to both CPU RESET pin and Cremson XRESET pin. (In PLL mode V832 CPU requires a minimum 10msec of "L" level input to the RESET pin at power on reset sequence.)
- DMARQn signal needs to be set as low active input. DMAAKn signal needs to be set as high active output. TC/STOPAK signal needs to be set to TC output. At the end of DMA transaction, "L" level should be output from TC pin.
- When CS is not asserted, XRDY outputs Hi-Z level. So XRDY is recommended to cramp to low level (pull-down) by an external resistor.
- When Fujitsu graphics driver is used, the Cremson resource can not be mapped to the V832 I/O field.

1.2 Video Output Interface

1.2.1 Connection to MB3516A NTSC video encoder

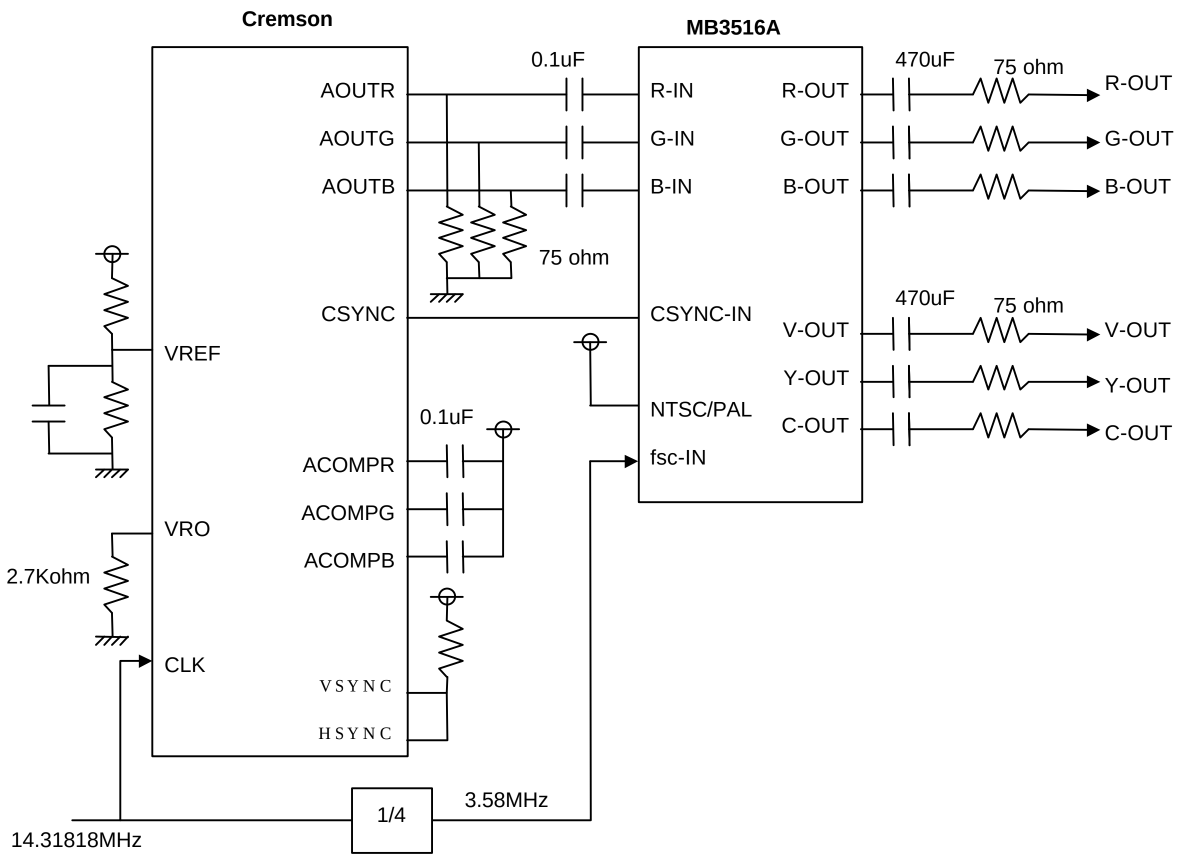

Fig.1.3 Connection to MB3516A NTSC encoder

Note1: When Interlace and sync mode is selected, vertical resolution is relatively higher, but occasionally flicker may occur. To reject such a flicker, Non-interlace mode should be selected.

Note2: The HSYNC of Cremson and color carrier of MB3516A may work asynchronously. However, there could be a case that the leakage of the color carrier signal is emphasized on some display devices. Also, depends on the electrical characteristics of the receiver circuit, there could be a case that color decode is not appropriately done. To avoid these sorts of problems, as shown above, the clock source for the Cremson and MB3516A should be common, and the respective display control registers of Cremson ought to be set to realize frequency ratio between HSYNC and color carrier must be set to 1:227.5. Actually, the following formula should be guaranteed:

(SC+1)*(HTP+1) = 910 * 14

SC: SC field value in the DCM (Display Control Mode) register

HTP: HTP(Horizontal Total Pixel) register value

1.2.2 Circuit example to apply external sync mode

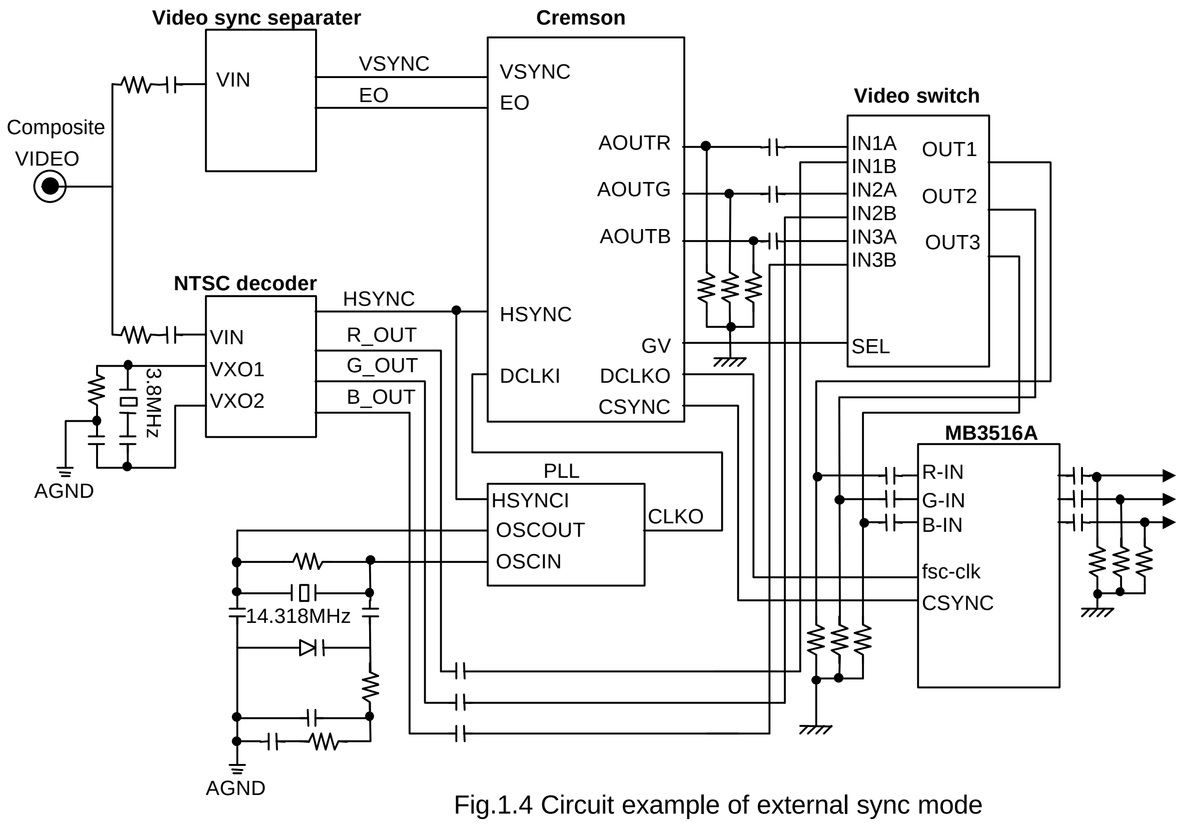

Fig.1.4 Circuit example of external sync mode

The circuit example shown above is an example to apply external sync mode, in which graphics contents rendered and displayed by Cremson is superimposed to the external video screen synchronously to the external sync signal. More in detail of this circuit example would be referred to the schematic diagram of MB86290EB01 (Cremson evaluation board). To apply an external sync signal, the ESY bit of the DCM register (in display control register field) needs to be set to “1”. The base clock for display can be selected from either internal PLL or external DCLKI by the CKS bit setting of the same register. Also, by the setting of KEYC (KEY Color) register, chromakey operation and that key color are selected. That key color can be selected from either display color or C layer color by the setting of CKM (Chroma Key Mode) register.

When external sync mode is applied, the following settings are recommended:

(1) Set HSYNC/VSYNC interval relatively shorter
Internal HSYNC/VSYNC interval should be set little shorter than the external input sync to avoid synchronizing to a double frequency.

(2) Set HSYNC pulse width to max (255)
In external sync mode, HSYNC pulse period is to wai the input of external Hsync signal. To make the synchronization easier, this waiting period ought to be as long as possible.

An example of register setting in external sync mode is shown below:

- Screen resolution: 640 x 480
- Superimpose of NTSC signal
- PLL output is applied for sampling clock

DCE register: 0x8001h,	DCM register: 0x0e37h,
HTP register: 0x02cfh,	HDB register: 0x027fh,
HDP register: 0x027fh,	HSW register: 0xffh,
HSP register: 0x0283h,	VSW register: 0x02h,
VTP register: 0x00ffh,	VDP register: 0x00efh,
VSP register: 0x00f2h	

1.3 SDRAM Interface

1.3.1 Connection to SDRAM devices

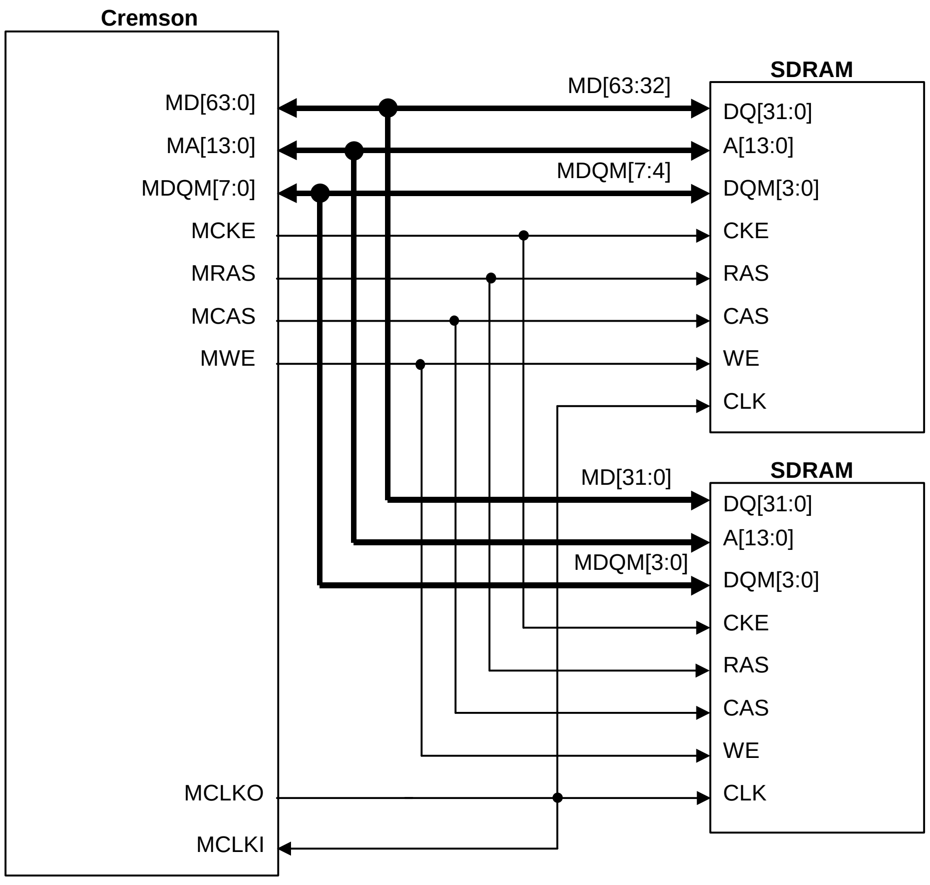

Fig.1.5 Connection to SDRAM devices

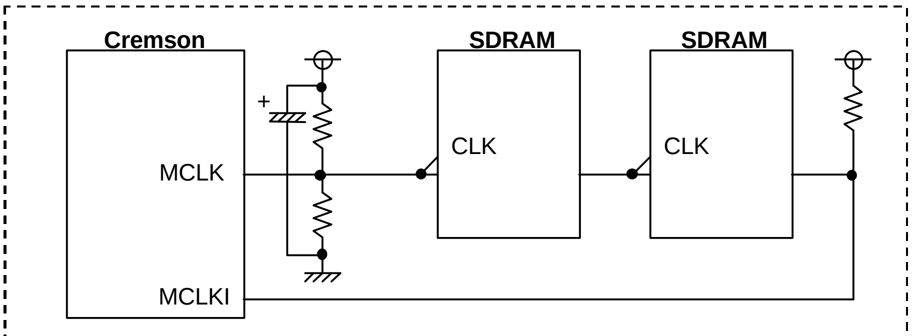

Fig.1.6 SDRAM interface clock circuit example

The circuit examples shown above are the case when two 64Mbit SDRAM devices (512K x 32bit x 4bakns) are connected to the SDRAM interface of Cremson. As described in Cremson hardware specifications, Cremson supports to interface to external SDRAM/SGRAM devices without glue logic according to the MMR (Memory Read Register) setting. To optimize the memory access operations, the following bits in the MMR ought to be set appropriately according to the specifications of the external memory devices:

Bit3	Data bus width
Bit4-5	Row address bus width

2 Start up of Cremson

2.1 Initialization of Cremson

At initialization of Cremson, MMR(Memory Mode Register) and display control register field need to be set according to the system configuration. The following table shows examples of MMR setting parameters. For the setting to display control register field, section 3 “Display Functions” would be referred.

Type	Data bus width	# of devices	Operation condition	MMR setting
SGRAM 16MBit (x32Bit)	32Bit		100MHz, Refresh @1024CLK	0x014fba93h
SGRAM 16MBit (x32Bit)	64Bit	2	<100MHz, Refresh @256CLK*	0x014fba1bh
SDRAM 64MBit (x32Bit)	32Bit	1	100MHz, Refresh @1024CLK	0x014fbab3h
SDRAM 64MBit (x32Bit)	64Bit	2	<100MHz, Refresh @256CLK*	0x014fba3bh
SDRAM 64MBit (x16Bit)	64Bit	4	100MHz, Refresh @1024CLK	0x014fba8bh

* An example to apply low speed temperature expansion version SDRAM

For optimized memory access operations, MMR register ought to be set appropriately according to the applied memory device specifications.

2.2 Execution of graphics operation

Cremson executes all the drawing and display operations by feeding respective display lists (stream of commands and their parameters) to its display list FIFO. For the display list transfer the following three methods are available:

- DMA transfer (Main memory to display list FIFO by the DMAC contained in the host CPU)
- Master transfer by Cremson (Graphics memory to display list FIFO by Cremson itself)
- CPU transfer (write operation of CPU program execution to display list FIFO)

2.2.1 Display list execution of Fujitsu Graphics Driver

Fujitsu Graphics Driver is a set of commands written in C language to assist graphics application programs utilizing Cremson. Fujitsu Graphics Driver is to interface between application programs (or graphical libraries) and hardware. For all the graphics command executions, Fujitsu Graphics Driver provides two operation modes, sync mode and async mode. In sync mode, a new display list will not be transferred to Cremson till current display list execution will complete. In async mode, new display list transfer is started without waiting for the completion of current display list execution. Detail of this scheme would be referred to the Cremson Graphics Driver Users Manual. The following lists indicate program examples for these scheme:

```
 /*Sync Mode*/
GdcExecMode ( GDC_ENABLE );
    /*Set sync mode*/
             :
GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );   /*Display list transfer for register set (1)   */
GdcSetLinePattern ( 0x0000FFFF );                     /*Display list transfer for register set (2)   */

GdcPrimType ( GDC_LINES_FAST );                       /*                                             */
GdcDrawVertex2D ( 0x01000000, 0x00100000 );           /*Display list transfer                        */
GdcDrawVertex2D ( 0x01000000, 0x00100000 );           /* of drawing command                          */
GdcPrimEnd ( );                                       /*                                             */


 /*Async Mode*/
GdcExecMode ( GDC_DISABLE );
    /*Set async mode*/
             :
GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );
GdcSetLinePattern ( 0x0000FFFF );
GdcPrimType ( GDC_LINES_FAST );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcPrimEnd ( );
GdcFlush ( );                                                    /*Transfer    display    lists    en
block*/

 /*Polling for the completion of drawing commands*/
GdcExecMode (GDC_DISABLE);
    /*Set async mode*/

GdcSetAttrLine ( GDC_LINE_WIDTH,GDC_LINE_WIDTH_5 );
GdcSetLinePattern ( 0x0000FFFF );
GdcPrimType ( GDC_LINES_FAST );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcDrawVertex2D ( 0x01000000, 0x00100000 );
GdcPrimEnd ( );
GdcSync ( );                                          /*Transfer display list en block*/
```

```
                                                  /*And wait for completion*/
```

Display list transfer procedure utilizing Fujitsu Graphics Driver is set by GdCFluchDisplay List (ULONG*src, ULONG count) command. Detail of this command would be referred to Cremson Graphics Driver Users Manual.

2.2.2 Execution of display list

Program examples of display list transfer in DMA or CPU write operation are shown in the following sections. Each command in these program examples mean as follows:

```
void wrb(unsigned long addr, unsigned char data) {        /*Byte write*/
    unsigned char *paddr = ((unsigned char *) addr);
    *paddr = data;
}
void wrb_IO(unsigned long ioadr, unsigned char data){    /*Byte wrote to I/O field*/
    asm("out.b %0,0[%1]"::"r"(data),"r"(ioadr));
}
void wrw(unsigned long addr, unsigned short data) {       /*Word write*/
    unsigned short *paddr = ((unsigned short *) addr);
    *paddr = data;
}
void wrw_IO(unsigned long ioadr, unsigned short data){  /*Word write to I/O field*/
    asm("out.h %0,0[%1]"::"r"(data),"r"(ioadr));
}
void wrl(unsigned long addr, unsigned long data) {        /*Long Word write*/
    unsigned long *paddr = ((unsigned long *) addr);
    *paddr = data;
}
void wrl_IO( unsigned long ioadr, unsigned long data){   /*Long Word write to I/O field*/
    asm("out.w %0,0[%1]"::"r"(data),"r"(ioadr));
}
```

2.2.3 DMA transfer (Main memory to display list FIFO)

(1) SH3

Assumption

- P2 logical field is assigned to Cremson.
- CS4 physical field is mapped to Cremson.
- Display list is already set in the CS3 physical field
- DMA channel 1 is assigned for DMA transfer of Cremson.

```
#define SAR1        0xA4000030    /*DMA Source Address Register 1*/
#define DAR1        0xA4000034    /*DMA Destination Address Register 1*/
#define TCR1        0xA4000038    /*DMA Transfer Count Register 1*/
#define CHCR1       0xA400003C    /*DMA Channel Control Register 1*/
#define DMAOR       0xA4000060    /*DMA Operation Register*/

#define CRM_DSU     0xB1FC0004    /*Cremson DMA Set up Register*/
#define CRM_DTC     0xB1FC0000    /*Cremson DMA Transfer Count Register*/
#define CRM_DRQ     0xB1FC0018    /*Cremson DMA Request Register*/

/*-----------------------------------------------------------------------------------------------------------
Setting and trigger of SH3 DMAC
-----------------------------------------------------------------------------------------------------------*/
            wrb ( CRM_DSU,0x02 );       /* Set DMA transfer mode*/
            wrl ( CHCR1, 0x00051011 ); /* DREQ, DACK: High active*/
                                       /*Source address increment*/
                                       /*Destination address fixed*/
```

```
                                   /*Dual address mode*/
           wrw ( DMAOR, 0x0001 );      /* DMA Start*/
           wrl ( SAR1, 0xAC000000 );   /* DMA Source address 0xAC000000*/
           wrl ( DAR1, 0xB1FF04A0 );   /*DMA Destination address DrawBase + 4A0 (DFIFO)*/
           wrw ( TCR1, 0x1000 );       /* DMA Transfer count 0x1000*/

           wrw ( CRM_DTC, 0x1000 );   /* Set transfer count 0x1000 to Cremson*/
           wrb ( CRM_DRQ, 0x01 );     /* Trigger DMA operation to Cremson*/
```

(2) SH4

Assumption

- P2 logical field is assigned to Cremson.
- CS4 physical field is mapped to Cremson.
- Display list is already set in the CS3 physical field
- DMA channel 1 is assigned for DMA transfer of Cremson.

```
#define SAR1         0xFFA00010    /*DMA Source Address Register 1*/
#define DAR1         0xFFA00014    /*DMA Destination Address Register 1*/
#define TCR1         0xFFA00018    /*DMA Transfer Count Register 1*/
#define CHCR1        0xFFA0001C    /*DMA Channel Control Register 1*/
#define DMAOR        0xFFA00040    /*DMA  Operation Register*/

#define CRM_DSU      0xB1FC0004    /*Cremson DMA Set up Register*/
#define CRM_DTC      0xB1FC0000    /*Cremson DMA Transfer Count Register*/
#define CRM_DRQ      0xB1FC0018    /*Cremson DMA Request Register*/

/*----------------------------------------------------------------------------------------------------------
Setting and trigger of SH4 DMAC
----------------------------------------------------------------------------------------------------------*/
           wrb ( CRM_DSU,0x02 );       /* Set DMA transfer mode*/
           wrl ( CHCR1, 0x00001241 );/* DREQ, DACK: High active*/
                                   /*Source address increment*/
                                   /*Destination address fixed*/
                                   /*Single address mode, 32Byte unit*/
           wrl ( DMAOR, 0x00000001 );                 /* DMA Start*/
           wrl ( SAR1, 0xAC000000 );   /* DMA Source address 0xAC000000*/
           wrl ( DAR1, 0xB1FF04A0 );   /*DMA Destination address DrawBase + 4A0 (DFIFO)*/
           wrw ( TCR1, 0x1000 );       /*DMA Transfer count 0x1000*/

           wrw ( CRM_DTC, 0x1000 );   /* Set transfer count 0x1000 to Cremson*/
           wrb ( CRM_DRQ, 0x01 );     /* Trigger DMA operation to Cremson*/
```

(3) V832

Assumption

- Block 3 (CS3) of memory field is assigned to Cremson
- Display list is already set to memory field specified by "glbDmaSAR1".
- DMA channel 0 is assigned for DMA transfer of Cremson.

```
#define V_DSA0H     0xC0000030    /*DMA Source Address Register 0 High*/
#define V_DSA0L     0xC0000032    /*DMA Source Address Register 0 Low*/
#define V_DDA0H     0xC0000034    /*DMA Destination Address Register 0 High*/
#define V_DDA0L     0xC0000036    /*DMA Destination Address Register 0 Low*/
#define V_DBC0H     0xC0000038    /*DMA Byte Count Register 0 High*/
#define V_DBC0L     0xC000003A    /*DMA Byte Count Register 0 Low*/
#define V_DCH0      0xC000003C    /*DMA Channel Control Register 0*/

#define CRM_DRQ     0x03fc0018    /*Cremson DMA Transfer Count Register*/
#define CRM_DSU     0x03fc0004    /*Cremson DMA Request Register*/
/*----------------------------------------------------------------------------------------------------------
```

```
Setting and trigger of V832 DMAC
-----------------------------------------------------------------------------------------------------------*/

    wrw_IO(V_DSA0H, (short)(glbDmaSAR1 >> 16));        /*Set source address*/
    wrw_IO(V_DSA0L, (short)(glbDmaSAR1));

    wrw_IO(V_DDA0H, (short)(glbDmaDAR1 >> 16));        /*Set destination address*/
    wrw_IO(V_DDA0L, (short)(glbDmaDAR1));

    wrw_IO(V_DBC0H, (short)((glbDmaTCR1-1) >> 14));   /*Set transfer count*/
    wrw_IO(V_DBC0L, (short)((glbDmaTCR1-1) << 2));

    wrw_IO (V_DCHC0, 0x00a5);                                       /*Set channel 0 DMA mode*/
    wrw_IO(V_DC, 0x0001);                                           /*DMA mode control*/

    wrb(CRM_HOST_DSU, 0x02 );         /* Set transfer count 0x02 to Cremson*/
    wrb(CRM_HOST_DRQ, 0x01 );         /* Trigger DMA operation to Cremson*/
```

If CPU mode is set to apply V832 (MODE[1:0]=0x2), DTC register register of host interface register field don't need to be set.

2.2.4 Master transfer by Cremson (Graphics memory to display list FIFO)

Cremson is capable to transfer display lists from its local Graphics memory to display list FIFO.

Assumption

- Base address of display list buffer in the Graphics memory is 0x00000000h.
- Display list is already set in the display list buffer
- Allocation of memory field is the same condition as specified in the DMA mode.

```
    #define CRM_LSA     0xB1FC0040    /*Cremson Display list source address register*/
    #define CRM_LCO     0xB1FC0044    /*Cremson Display list transfer count register*/
    #define CRM_LREQ  0xB1FC0048    /*Cremson Display list transfer request register*/
  /*-----------------------------------------------------------------------------------------------------
  Display list read from the Graphics memory
  ------------------------------------------------------------------------------------------------------*/
            wrl ( CRM_LSA, 0x00000000 );     /*Set the base address of display list buffer*/
            wrl ( CRM_LCO, 0x00001000 );     /*Set transfer word count of display list*/
            wrb ( CRM_LREQ, 0x01 );          /*Trigger display list transfer*/
```

Although the above example is a case of SH3 CPU, the sequence is the same for SH4 and V832.

3 Display Control

Cremson supports 4 layers of display frames (C layer, W layer, M layer and B layer). The bottom two layers, M layer and B layer, can be split to two separated windows (L frame and R frame at any position). All these 6 frames are assigned as logically separated field in the Graphics Memory. Besides these display frames, two hardware cursors are also supported.

3.1 Setting of Display Parameter

A display parameter setting program example is shown as follows:

Assumption

- Overlay C, ML, MR, BL and BR layers in a standard VGA screen (640x480) MR and BR can be displayed only when window separation mode is applied. If a single frame is displayed for M and B layers, ML and BL frames must be used.
- Apply 8bit indirect color mode for C layer. Define ULONG pointer “ucolorC” and set color palette data.

3.1.1 HSYNC/VSYNC parameter setting

HTP	Horizontal Total Pixels	800 pixel
HSP	Horizontal Synchronize pulse Position	656 pixel
HSW	Horizontal Synchronize pulse Width	96 pixel
HDP	Horizontal Display Period	640 pixel (L: 320 pixel, R: 320 pixel)
HDB	Horizontal Display Boundary	640 pixel (L: 320 pixel, R: 320 pixel)
VTR	Vertical Total Raster	525 raster
VSP	Vertical Synchronize pulse Position	490 raster
VSW	Vertical Synchronize pulse Width	2 raster
VDP	Vertical Display Period	480 raster

```
    /*-------- An example to apply Fujitsu Graphics Driver --------*/
    /*Display parameter setting*/
    GdcDispTiming ( 800, 656, 96, 640, 525, 490, 2, 480 );        /*Window size and display
timing set*/
    GdcDispDividePos ( 320 );                                    /*Left frame width set*/

    /*-------- Register setting --------*/
    HTP Reg = 0x31fh,  HDP Reg = 0x27fh,  HDB Reg = 0x13fh,
    HSP Reg = 0x28fh,  HSW Reg = 0x5fh,  VTR Reg = 0x20ch,
    VDP Reg = 0x1dfh,  VSP Reg = 0x1e9h,  VSW Reg = 0x1h
```

Note-1: Actual setting value to each register is “applied size - 1”.
Note 2: When single frame display mode is applied to M and B layers (ML and BL frame only), set HDP register = HDB register = 0x27fh.

3.1.2 Display frame setting

Each display frame is set respectively. Setting examples are shown as follows.

(1) C-layer : Draw frame size is 640x480

CH	C-layer Height	480 raster
CW	C-layer memory Width	10 (640 pixel)
CC	C-layer Color mode	8bit indirect color
COA	C-layer Original Address	0x000000h
CDA	C-layer Display Address	0x000000h
CDX	C-layer Display position X	0x0000
CDY	C-layer Display position Y	0x0000
CTC	C-layer Transparent Color	Palette #0 color

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
/*Display frame setting*/
GdcDispDimension ( GDC_DISP_LAYER_C, GDC_EMABLE, GDC_DISP_8_BPP,
        GDC_FLIPMODE_0, 0, 0, 640, 480 );          /*C-layer frame attribute setting*/
/*Display start position setting*/
GdcDispPos (GDC_DISP_LAYER_C, 0, 0, 0 );           /*C-layer display start position setting*/
GdcColorTransparent (GDC_DISP_LAYER_C, 0x0 );   /*C-layer transparent color setting*/
GdcColorZeroMode (GDC_DISP_LAYER_C, GDC_COLOR_TRANSPARENT);
                                                   /*Color zero mode setting*/
GdcColorPalette (GDC_C_LAYER_PALETTE, 0, 256, ucolorC );
                                                   /*C-layer palette setting*/

/*-------- Register setting --------*/
CM Reg {CC, CW, CH} = 0x000a01dfh,  COA Reg = 0x000000h,  CDA Reg = 0x000000h,
CDX Reg = 0x0000h,  CDY Reg = 0x0000h,  CTC Reg = 0x8000h

Note:   Actual setting value to C-layer Height is “applied height - 1”.
/*-------- Palette register setting --------*/
For ( I=0 ; I<256; I++ ){
        CPAL(I) Reg = *( ucolorC );
        UcolorC ++ ;
}
```

(2) ML-layer : Draw frame size is 640x480

MLH	ML-layer Height	480 raster
MLW	ML-layer memory Width	20 (640 pixel)
MLFLP	ML-layer Flip mode	Display frame 0
MLC	ML-layer Color mode	16bit direct color
MLOA0	ML-layer Original Address 0	0x096000h
MLDA0	ML-layer Display Address 0	0x096000h
MLDX	ML-layer Display position X	0x0000
MLDY	ML-layer Display Position Y	0x0000
MLTC	ML-layer Transparent Color	Color code 0

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
/*Display frame setting*/
GdcDispDimension ( GDC_DISP_LAYER_ML, GDC_EMABLE, GDC_DISP_16_BPP,
        GDC_FLIPMODE_0, 0x096000, 0x096000, 640, 480 );
```

```
                                                        /*ML-layer frame attribute setting*/
/*Display start position setting*/
GdcDispPos (GDC_DISP_LAYER_ML, 0, 0, 0 );               /*ML-layer display start position setting*/
GdcColorTransparent (GDC_DISP_LAYER_ML, 0x0 );          /*ML-layer transparent color setting*/
GdcColorZeroMode (GDC_DISP_LAYER_ML, GDC_COLOR_TRANSPARENT);
                                                        /*Color zero mode setting*/

/*-------- Register setting --------*/
MLM Reg {MLC, MLFLP, MLW, MLH} = 0x801401dfh,  MLOA0 Reg = 0x096000h,
MLDA0 Reg = 0x096000h,  MLDX Reg = 0x0000h,  MLDY Reg = 0x0000h,
MLTC Reg = 0x8000h
```

Note: Actual setting value to ML-layer Height is “applied height - 1”.

(3) MR-layer : Draw frame size is 640x480

MRH	MR-layer Height	480 raster
MRW	MR-layer memory Width	20 (640 pixel)
MRFLP	MR-layer Flip mode	Display frame 0
MRC	MR-layer Color mode	16bit direct color
MROA0	MR-layer Original Address 0	0x096000h
MRDA0	MR-layer Display Address 0	0x096000h
MRDX	MR-layer Display position X	0x0000
MRDY	MR-layer Display Position Y	0x0000
MRTC	MR-layer Transparent Color	Color code 0

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
/*Display frame setting*/
GdcDispDimension ( GDC_DISP_LAYER_MR, GDC_EMABLE, GDC_DISP_16_BPP,
        GDC_FLIPMODE_0, 0x096000, 0x096000, 640, 480 );
                                          /*MR-layer frame attribute setting*/
/*Display start position setting*/
GdcDispPos (GDC_DISP_LAYER_MR, 0, 0, 0 );   /*MR-layer display start position setting*/
GdcColorTransparent (GDC_DISP_LAYER_MR, 0x0 );/*MR-layer transparent color setting*/
GdcColorZeroMode (GDC_DISP_LAYER_MR, GDC_COLOR_TRANSPARENT);
                                          /*Color zero mode setting*/

/*-------- Register setting --------*/
MRM Reg {MRC, MRFLP, MRW, MRH} = 0x801401dfh,  MROA0 Reg = 0x096000h,
MRDA0 Reg = 0x096000h,  MRDX Reg = 0x0000h,  MRDY Reg = 0x0000h,
MRTC Reg = 0x8000h
```

Note: Actual setting value to MR-layer Height is “applied height - 1”.

(4) BL-layer : Draw frame size is 1280x960

BLH	BL-layer Height	960 raster
BLW	BL-layer memory Width	40 (1280 pixel)
BLFLP	BL-layer Flip mode	Display frame 0
BLC	BL-layer Color mode	16bit direct color
BLOA0	BL-layer Original Address 0	0x1c2000h
BLDA0	BL-layer Display Address 0	0x1c2000h
BLDX	BL-layer Display position X	0x0000
MLDY	BL-layer Display Position Y	0x0000

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
/*Display frame setting*/
GdcDispDimension ( GDC_DISP_LAYER_BL, GDC_EMABLE, GDC_DISP_16_BPP,
        GDC_FLIPMODE_0, 0x1c2000, 0x1c2000, 1280, 960 );
                                              /*BL-layer frame attribute setting*/
/*Display start position setting*/
GdcDispPos (GDC_DISP_LAYER_BL, 0, 0, 0 );     /*BL-layer display start position setting*/

/*-------- Register setting --------*/
BLM Reg {BLC, BLFLP, BLW, BLH} = 0x802803bfh,  BLOA0 Reg = 0x1c2000h,
BLDA0 Reg = 0x1c2000h,  BLDX Reg = 0x0000h,  BLDY Reg = 0x0000h
```

Note: Actual setting value to BL-layer Height is “applied height - 1”.

(5) BR-layer : Draw frame size is 640x480

BRH	BR-layer Height	480 raster
BRW	BR-layer memory Width	20 (1280 pixel)
BRFLP	BR-layer Flip mode	Display frame 0
BRC	BR-layer Color mode	16bit direct color
BROA0	BR-layer Original Address 0	0x384000h
BRDA0	BR-layer Display Address 0	0x384000h
BRDX	BR-layer Display position X	0x0000
MRDY	BR-layer Display Position Y	0x0000

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
/*Display frame setting*/
GdcDispDimension ( GDC_DISP_LAYER_BR, GDC_EMABLE, GDC_DISP_16_BPP,
        GDC_FLIPMODE_0, 0x384000, 0x384000, 1280, 960 );
                                              /*BR-layer frame attribute setting*/
/*Display start position setting*/
GdcDispPos (GDC_DISP_LAYER_BR, 0, 0, 0 );     /*BR-layer display start position setting*/

/*-------- Register setting --------*/
BRM Reg {BRC, BRFLP, BRW, BRH} = 0x802803bfh,  BROA0 Reg = 0x384000h,
BRDA0 Reg = 0x384000h,  BRDX Reg = 0x0000h,  BRDY Reg = 0x0000h
```

Note: Actual setting value to BR-layer Height is “applied height - 1”.

3.1.3 Overlay mode setting

C-layer is overlay onto all the rest layers (B, M, W-layers). The following example shows the mode setting for this overlay display.

Assumption

- Blend mode is applied
- Blending ratio is 50% (blending parameter = 0x80h)

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
OverlayPriorityMode ( GDC_OVERLAY_C_BLEND );   /*Overlay display mode setting*/
GdcOverlayBlend ( GDC_ENABLE, 0x80h );         /*Blending parameter setting*/

/*-------- Register setting --------*/
BMODE Reg = 0x0001h,  BRATIO Reg = 0x0080h
```

Note: Overlay blend is applicable only when the alpha bit of the C-layer pixel is “1”.

3.1.4 Display control mode register setting

SYNC	Synchronize	Non interlace mode
ESY	External Synchronize	Disable
SF	Synchronize signal output Format	Low active
EO	Even/Odd signal mode	Low in even frame, High in odd frame
SC	Scaling	1/8 (25MHz)
CKS	Clock Source	Internal PLL output clock
CE	C-layer Enable	Enable
WE	W-layer Enable	Disable
MRE	MR-layer Enable	Enable
MLE	ML-layer Enable	Enable
BRE	BR-layer Enable	Enable
BLE	BL-layer Enable	Enable
DEN	Display Enable	Enable

```
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcDispClock ( 0x00000700h );

/*-------- Register setting --------*/
DCM Reg { CKS, SC, EO, SF, ESY, SYNC } = 0x0700h,
DCE Reg { DEN, BE, ME, WE, CE } = 0x800dh
```

3.1.5 Window scroll

Window scroll is performed by changing Display position X, Y and Display Address 0 of the respective frame. The following list indicates a program example to perform windows scroll.

Assumption

- The same condition of 3.1.2 Display frame setting, is applied to MR-layer.
- Scroll MR-layer from the right to the left for 1 pixel at every VSYNC interrupt.

```
/*-------- Window scroll program example --------*/
#define          MRda          0x
#define          Rdx           0
```

```
  #define             Rdy           0

main ( ) {
  -        -        -        -        -        -        -        -
  - - - - - - - -
}
  /*-------- An example to apply Fujitsu Graphics Driver --------*/
void VSyncINT ( void )                  /*VSYNC interrupt */
{
  - - - - - - - -
  GdcDispPos ( GDC_DISP_LAYER_MR, GDC_BANK_0, Rdx, Rdy );
  Rdx = Rdx + 1 ;
  Rdy = Rdy + 0 ;
  if ( Rdx > 639)  Rdx = 0;  if ( Rdy > 479 ) Rdy = 0 ;
  - - - - - - - -
  - - - - - - - -
  GdcClearInterruptStatus ( 0xFB );   /*Interrupt status clear */
}


  /*-------- An example NOT to apply Fujitsu Graphics Driver --------*/
void VSyncINT ( void )                  /*VSYNC interrupt */
{
  - - - - - - - -
  MRDA0 Reg = Rda ;  MRDX Reg = Rdx ;  MRDY Reg = Rdy ;
  Rdx = Rdx + 1 ;
  Rdy = Rdy + 0 ;
  if ( Rdx > 639)  Rdx = 0;  if ( Rdy > 479 ) Rdy = 0 ;
  Rda = 2 x Rdx + 1280 x Rdy + Rda ;
  - - - - - - - -
  - - - - - - - -
  IST Reg = 0x00                        /*Interrupt status clear */
}
```

Note: The following formula is applied to calculate the Display Address in 16bit direct color mode.

$$DA = 2*DX + 64*W*DY + OA$$

DA : Display Address,	DX : Display position X
W : memory Width,	DY : Display position Y
OA : Original Address	

3.1.6 Flipping

Cremson supports two kinds of flipping modes, automatic flipping (controlled by Cremson automatically) and manual flipping (controlled by application program). The following lists indicate program examples of these flipping operations.

(1) Automatic flipping by Cremson

Display frame 0 and 1 of any layer are flipped to each other and displayed automatically. By this mode, a smooth animation is performed.

Assumption
- Flip Display frame 0 and 1 of BL-layer alternatively
- Draw frame size of BL-layer is 640x480
- Set BL-layer Original Address 0 (Displat frame 0) to 0x00000h
- Set BL-layer Original Address 1 (Display frame 1) to 0x96000h
- Flip Display frame 0 and 1 alternatively at every VSYNC interrupt

```
  /*-------- Automatic flipping program example --------*/
```

```
    /*-------- An example to apply Fujitsu Graphics Driver --------*/
   void DISP_INIT ( void ) {                               /*Display         control         parameter
initialization*/
                :
     GdcDispDimension ( GDC_DISP_LAYER_BL, GDC_ENABLE, 16BPP_FORMAT,
                         GDC_FLIPMODE_AUTO, 0x00000000, 0x96000, 640, 480 );
                                                        /*Set BL-layer to automatic flip mode*/
                :
    }
    void VSyncINT ( void ) {                            /*VSYNC interrupt */
                :
    if ( flag ) GdcDrawDimension ( GDC_16BPP_FORMAT, 0x00000, 640, 480 );
                                                        /*Set Display frame 0 to draw frame*/
    else GdcDrawDimension ( GDC_16BPP_FORMAT, 0x96000, 640, 480 );
                                                        /*Set Display frame 1 to daw frame*/
                :
     flag = ~flag
    }

    /*-------- An example NOT to apply Fujitsu Graphics Driver --------*/
   void DISP_INIT ( void ) {                               /*      Display      control      parameter
initialization*./

    BLM Reg {BLC, BLFLP, BLW, BLH} = 0xC02803BF,  BLOA0 Reg = 0x000000,
    BLOA1 Reg = 0x96000
                :
    }
   void VSyncINT ( void ) {                             /*VSYNC interrupt*/
                :
    if ( flag )      FBR Reg = 0x00000,  XRES Reg = 0x00000280
     else            FBR Reg = 0x96000,  XRES Reg = 0x00000280
                :
   flag = ~flag
    }
```

(2) Manual flipping by user program

Display frame 0 and 1 of any layer are flipped to each other and displayed at any timing set by the user program. By this mode, flipping is performed without showing the process of rendering.

Assumption

- Draw frame size of BL-layer is 640x480
- Set BL-layer Original Address 0 (Displat frame 0) to 0x00000h
- Set BL-layer Original Address 1 (Display frame 1) to 0x96000h
- Draw graphics to frame 0 while frame 1 is displayed, and after drawing to frame 0 completes, flip the frame and start displaying frame 0.

```
 /*-------- Automatic flipping program example --------*/
 /*-------- An example to apply Fujitsu Graphics Driver --------*/
 GdcDispDimension ( GDC_DISP_LAYER_BL, GDC_ENABLE, 16BPP_FORMAT,
                    GDC_FLIPMODE_1, 0x00000000, 0x96000, 640, 480);
                                        /* Set frame 1 to display*/
GdcDrawDimension ( GDC_16BPP_FORMAT, 0x000000, 640, 480 ) ;
                                        /*Set frame 0 to draw frame*/
  GdcSync ( ) ;                         /*Execute display list and wait for completion*/
GdcDispDoFlip ( GDC_DISP_LAYER_BL GDC_BANK_0 );      /*Set frame 0 to display*/


 /*-------- An example NOT to apply Fujitsu Graphics Driver --------*/
BLM Reg { BLC, BLFLP, BLW, BLH } = 0xA02803BF, BLOA0 Reg = 0x000000,
 BLOA1 Reg = 0x96000, FBR Reg = 0x000000, XRES Reg = 0x0280

while ( CTR Reg   PS   01) { }          /*Wait for completion of pixel engine operation*/
BLM Reg { BLC, BLFLP, BLW, BLH } = 0x802803BF
```

3.2 Setting of Hardware Cursors

Cremson can display two hardware cursors at the same time. Each cursor is specified in 64x64 pixel size. Only 8bit indirect color mode is applicable and C-layer palette is applied for the color look-up. The display priority to C-layer for each hardware cursor is independently programmable. Each cursor can be displayed either top or underneath of C-layer pixels. If a part of hardware cursor crosses over the border of display frame, the part exceed the border is not displayed. The following list indicates a program example to set hardware cursors.

Assumption

- Pattern data of each cursors is set in the host memory address pointed by curpat or curpat2 respectively. These pattern data are loaded to Graphics memory area, which address starts from 0x384000h or 0x385000h.

```
  /*-------- Cursor setting program example --------*/
  /*-------- An example to apply Fujitsu Graphics Driver --------*/
#define   CURSOE0                0x384000
#define   CURSOE1                0x385000

GdcCursorAddres ( 0, CURSOE0 );                   /*Set cursor 0 patter store address*/
GdcCursorAddres ( 1, CURSOE1 );                   /*Set cursor 1 pattern store address*/
GdcCursorPos ( 0, 128, 256 );                     /*Set cursor 0 display position*/
GdcCursorPos ( 1, 384, 256 );                     /*Set cursor 1 display position*/
GdcCursorPriority ( 0, GDC_PRIORITY_CURSOR );     /*Set cursor 0 priority order to C-layer*/
GdcCursorPriority ( 1, GDC_PRIORITY_CURSOR );     /*Set cursor 1 priority order to C-layer*/
GdcCursorColorTransparent ( 0x0 );                /*Set cursor transparent color*/
GdcCursorColorZeroMode ( GDC_COLOR_TRANSPARENT );
                                                  /*Set color zero mode*/
GdcCursorPattern ( 0, curpat );                   /*Load cursor 0 pattern*/
GdcCursorPattern ( 1, curpat2 );                  /*Load cursor 1 pattern*/
GdcCursorDisplay ( 0, GDC_ENABLE );               /*Enable cursor 0 display*/
GdcCursorDisplay ( 1, GDC_ENABLE );               /*Enable cursor 1 display*/
```

4 Rendering Function

Cremson can deliver various kinds of rendering operations performed by its dedicated 2D/3D graphics engines. For all the drawing operations performed by Cremson, such drawing attribute information, as draw frame setting, color mode and so on, ought to be set appropriately. Normally Cremson performs drawing operation according to the attribute setting done prior to that drawing operation. Current setting of attribute information is applied until it will be changed. This means the attribute information does not need to be set at every drawing operation. Considering the consistency of the attribute to be commonly used for multiple drawing operations, optimized scheme of programming can be made.

4.1 Setting of Draw Frame

Although display frame can be defined for each layer independently, multiple draw frames can not be defined at the same time (only current frame is applicable). At each time when drawing operation for any layer is performed, draw frame must be defined.

The following list indicates a program example of draw frame setting. No program example shown later than this section has specific draw frame definition for it, so every drawing example shown in the sample program is assumed to be put to the draw frame, which is defined just prior to that drawing program.

MDR0 (Mode Register for miscellaneous) register setting

BSH	Bitmap Scan Horizontal	x1
BSV	Bitmap Scan Vertical	x1
CX	Clip X enable	Disable
CY	Clip Y enable	Disable
CF	Color Format	Direct color mode (16bit/pixel)
EM	Endian Mode	Bit 0 is the right

FBR (Frame buffr Base) register setting

FBASE	Frame Buffer BASE	0x00000000

XRES (X Resolution) register setting

XRES	X Resolution	640 pixel

```
/*-------- Draw frame setting program example --------*/
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcDrawDimention ( GDC_16BPP_FORMAT, 0x00000000, 0x280, 0x01e0 )     /*Draw frame setting*/

/*-------- Display list example --------*/
F1010108          /*Type : Set Register, # of data : 1, Register address : 108 (=420h)*/
00008000          /*00008000h -> MDR0*/
F1010110          /*Type : Set Register, #of data : 1, Register address : 110 (=440h)*/
00000000          /*00000000h -> FBR*/
F1010111          /*Type : Set Register, # of data : 1, Register address : 111 (=444h)*/
00000280          /*00000280h -> XRES*/
```

4.2 Clear of Draw Frame

Cremson does not have a dedicated command to clear a draw frame. To clear a draw frame, the entire frame needs to be filled with a certain color by “rectangle fill” function. Rectangle fill is referred to the section 4.6.1 Block fill in single color or tiling pattern. If C or M-layer is filled with a transparent color, the underneath layer(s) become visible.

4.3 Issue of Flash Command

At the end of display list to be transferred to Cremson, the following flash commands must be put. By executing these flash commands, result of drawing operation will be wiped out at synchronization timing.

```
/*-------- Flash command isuing example --------*/
    :                    /*              */
    :                    /*Display list  */
    :                    /*              */
F0C10000                 /*Frame buffer flash command*/
F0C20000                 /*Z buffer flash command*/
```

Note: If Fujitsu Graphic Driver is applied, these flash commands are automatically inserted.

4.4 Clipping Operation

Cremson support a clipping border. When this border is set, all the commands to draw anything outside of the border are automatically rejected. The following list indicates a program sample to set the clip border.

MDR0 (Mode Register for miscellaneous) register setting

BSH	Bitmap Scan Horizontal	x1
BSV	Bitmap Scan Vertical	x1
CX	Clip X enable	Enable
CY	Clip Y enable	Enable
CF	Color Format	Direct color mode (16bit/pixel)
EM	Endian Mode	Bit 0 is the right

CXMIN (Clip X minimum) register setting

CLIPXMIN	CLIP border X Minimum	0 pixel

CXMAX (Clip X maximum) register setting

CLIPXMAX	CLIP border X Maximum	640 pixel

CYMIN (Clip Y minimum) register setting

CLIPYMIN	CLIP border Y Minimum	50 raster

CYMAX (Clip Y maximum) register setting

CLIPYMAX	CLIP border Y Maximum	400 raster

```
/*-------- Clip border setting program example --------*/
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcDrawClipFrame ( 0x0000, 0x0032, 0x0280, 0x0190 );        /*Clip border setting*/
```

```
        GdcSetAttrMisc ( GDC_CLIP, (GDC_CLIP_X_ON | GDC_CLIP_Y_ON ));
                                                        /*Activate          clipping
    operation*/

        /*-------- Display list example --------*/
        F1040115        /*Type : Set Register, # of data : 4, Register address : 115 (=454h)*/
        00000000        /*00000000 -> CXMIN*/
        00000280        /*00000280 -> CXMAX*/
        00000032        /*00000032 -> CYMIN*/
        00000190        /*00000190 -> CYMAX*/
        F1010108        /*Type : Set Register, # of data : 1, Register address : 108 (=420h)*/
        00008300        /*00008300 -> MDR0*/
```

4.5 Drawing of Binary Pattern

For the drawing of binary pattern, 32bit unsigned integer data format is adopted. In this format, each bit corresponds to a pixel. When the bit is “1”, the corresponding pixel is drawn in the foreground color, and when “0”, it is drawn in the background color. (Transparent color can be applied for the background color.)

The applicable drawing attributes in the binary pattern draw are, clipping and enlarge/shrink. The following list indicates a program sample to perform binary pattern drawing.

MDR0 (Mode Register for miscellaneous) register setting

BSH	Bitmap Scan Horizontal	x1
BSV	Bitmap Scan Vertical	x1
CX	Clip X enable	Disable
CY	Clip Y enable	Disable
CF	Color Format	Direct color mode (16bit/pixel)
EM	Endian Mode	Bit 0 is the right

FC (Foreground Color) register setting

FGC	Foreground Color	A = 1, R= 31, G = 31, B = 31 : White

BC (Background Color) register setting

BGC	Background Color	R = 0, G = 0, B = 0 : Black
BT	Background Transparency	Transparent effect ON

Assumption
- Binary pattern : Foreground color = White, Background color = Transparent
- Pattern data is already set in the address indicated by the unsigned Long pointer pat_ptr.
- Binary pattern size is 56 pixels x 16 rasters

```
/*-------- Binary pattern drawing program example --------*/
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcBitPatternMode ( GDC_BPSCALE_H_EQUIV | GDC_BPSCALE_V_EQUIV );
                                              /*Enlarge/shrink mode setting*/
GdcColor ( GDC_WHITE );                       /*Foreground color setting*/
GdcBackColor ( 0x8000 );                      /*Background color setting*/
GdcBitPatternDraw ( 320, 240, 56, 16, pat_ptr );  /*Binary pattern drawing*/
```

```
/*-------- Display list example --------*/
F1010108     /*Type : Set Register, # of data : 1, Register address : 108 (=420h)*/
00008000     /*00008000 -> MDR0*/
F1020120     /*Type : Set Register, # of data : 4, Register address : 120 (=480h)*/
00007FFF     /*00007FFF -> FC*/
00008000     /*00008000 -> BC*/
0B430022     /*Type : DrawBitmapP, Cmd : Bitmap, # of data : 34*/
00F00140     /*RYs, RXs (X,Y) coordinate of binary pattern drawing start point*/
00100038     /*RsizeY : 0010h, RsizeX : 0038h, height and width of pattern data*/
00000000     /*Pattern data                              */
   :         /*   :   Total 32 words of pattern data     */
   :         /*   :   2 words/line x 16 rasters = 32 words*/
00000000     /*Pattern data                              */
```

Note 1: Previously set draw attributes are continuously adopted, unless otherwise changed.

Note 2: 1 word of pattern data (32bit) contains binary pattern for 32pixels (1bit per 1 pixel).
Since horizontal width of the binary pattern in the above example is 56pixels, to specify the
binary pattern of each line, two words of data is needed. Whole 32bit data of the 1st word and
MSB side 3 bytes in the 2nd word are applied to describe the binary pattern for 56pixels
as follows. Also, the vertical height of the above example is 16 rasters. Therefore,
2 words/line x 16 rasters = 32words of pattern data is required.

pat_ptr[0] = 0xXXXXXXXX, pat_ptr[1] = 0xXXXXXX -- : 2 words for 56pixels of pattern data
Least Significant Byte in the 2nd
Word is not valid.

The following data shows an example of 56 x 16 binary pattern to describe a word “Fujitsu”.

```
static unsigned long pat_ptr[] = {
        0x00000000,   0x00000000,   0x00000000,   0x00000000,
        0x00000000,   0x00000000,   0xfe000810,   0x00000000,
        0x80000810,   0x00000000,   0x80000000,   0x10000000,
        0x80000000,   0x10000000,   0x80420810,   0x7e3c4200,
        0xfc420810,   0x10424200,   0x80420810,   0x10404200,
        0x80420810,   0x103c4200,   0x80420810,   0x10024200,
        0x80460810,   0x10424600,   0x803a0810,   0x0e3c3a00,
        0x00000800,   0x00000000,   0x00007000,   0x00000000 };
```

4.6 BLT

Cremson supports the following 5 types of BLT (Block Transfer) operation by a dedicated hardware.

- Block fill in single color or tiling pattern
- Block copy between two physical address area in the graphics memory
- Block copy from host memory to the graphics memory
- Block copy from host memory to internal texture buffer
- Block copy from the graphics memory to internal texture buffer

4.6.1 Block fill in single color or tiling pattern

The applicable drawing attributes in the block fill are, clipping blend mode (exclude alpha blending) and tiling mode. The following list indicates a program sample to perform block fill in single color or tiling pattern. Normally frame buffer clear is done by this operation.

(1) Block fill in single color
MDR4 (Mode Register for BLT) register setting

TI	Tiling enable	Disable
BM	Blend Mode	Normal (source copy)
LOG	Logical operation	CLEAR

FC (Foreground Color) register setting

FGC	Foreground Color	A = 1, R= 31, G = 31, B = 31 : White

Assumption

- Filling color is White.
- Block fill area is 60 pixels wide and 40 rasters tall. The (X, Y) coordinate of the left top pixel is (320, 420).

```
/*-------- Block fill program example --------*/
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcSetAttrBlt ( GDC_BLT_TILE, GDC_DISABLE );     /*Drawing attribute setting for block fill*/
GdcSetAttrBlt ( GDC_BLEND_MODE, GDC_BLEND_COPY );
                                                 /*Drawing attribute setting for block fill*/
GdcColor ( GDC_WHITE );                          /*Block fill color setting*/
GdcBltFill ( 320, 240, 60, 40 );                 /*Block fill*/
```

```
/*-------- Display list example --------*/
F101010C      /*Type : Set Register, # of data : 1, Register address : 10C (=430h)*/
00000000      /*00000000 -> MDR4*/
F1020120      /*Type : Set Register, # of data : 1, Register address : 120 (=480h)*/
00007FFF      /*00007FFF -> FC*/
09410000      /*Type : DrawRectP, Cmd : BltFill*/
00F00140      /*RYs, RXs (X,Y) left top point coordinate of the block*/
0028003C      /*RsizeY : 0028h, RsizeX : 003Ch, height and width of the block*/
```

Note: Previously set draw attributes are continuously adopted, unless otherwise changed.

(2) Block tiling
MDR4 (Mode Register for BLT) register setting

TI	Tiling enable	Enable
BM	Blend Mode	Normal (source copy)

LOG	Logical operation	CLEAR

TOA (Texture buffer Offset Address) register setting

XBO	Texture Buffer Offset	0x00000000

TIS (Tile Size) register setting

TISM	Texture Size M (Horizontal)	M = 32
TISN	Texture Size N (Vertical)	N = 32

MDR2 (Mode Register for Polygon) register setting [1st]

SM	Shading Mode	Flat shading
ZC	Z Compare mode	Disable
ZCL	Z Compare logic	NEVER
ZW	Z Write mask	Compare Z value and overwrite the result
BM	Blend Mode	Normal (source copy)
LOG	Logical operation	Clear
TT	Texture Tile select	Enable tiling operation

MDR2 (Mode Register for Polygon) register setting [2nd]

SM	Shading Mode	Flat shading
ZC	Z Compare mode	Disable
ZCL	Z Compare logic	NEVER
ZW	Z Write mask	Compare Z value and overwrite the result
BM	Blend Mode	Normal (source copy)
LOG	Logical operation	Copy
TT	Texture Tile select	Not used

Assumption

- Block tiling area is 100 pixels wide and 130 rasters tall. The (X, Y) coordinate of the left top pixel is (50, 80).
- Tile pattern data is already set in the address indicated by the COL16 pointer pTile64A. Use that tile pattern data after loading it to internal texture buffer which address area starts from 0x000h.
- Tile pattern size is 32 x 32.

```
/*-------- Block tiling program example --------*/
/*-------- An example to apply Fujitsu Graphics Driver --------*/
GdcSetAttrBlt ( GDC_BLT_TILE, GDC_ENABLE);      /*Drawing attribute setting for block tiling*/
GdcSetAttrBlt ( GDC_BLEND_MODE, GDC_BLEND_COPY );
                                                /*Drawing attribute setting for block tiling*/
GdcTextureLoadInt ( (32*32), pTile64A, 0 );     /*Tile pattern loading into internal texture buffer*/
GdcTextureDimmension ( 0, 32, 32 );                       /*Tile pattern set*/
GdcBltFill ( 50, 80, 100, 130 );                /*Block tiling*/
```

Note: Previously set draw attributes are continuously adopted, unless otherwise changed.

```
/*-------- Display list example --------*/
F101010C        /*Type : Set Register, # of data : 1, Register address : 10C (=430h)*/
00000001        /*00000000 -> MDR4*/
F101011B        /*Type : Set Register, # of data : 1, Register address : 11B (=46Ch)*/
00000000        /*00000000 -> TOA*/
11490200        /*Type : LoadTexture, Cmd : LoadTile, # of data : 512*/
7C007C00        /*Tile pattern data                                              */
   :            /*   :   Total 512 words of tile pattern data (1024pixels)       */
   :            /*   :   1 word = 2 pixels (16bit/pixel)                         */
7C007C00        /*Tile pattern data                                              */
F101011A        /*Type : SetRegister, # of data : 1, Register address : 11A (=468h)*/
00200020        /*00200020 -> TIS : Tile size (M, N) = (32, 32)*/
F101010A        /*Type : SetRegister, # of data : 1, Register address : 10A (=428h)*/
10000000        /*10000000 -> MDR2 : Set drawing mode*/
                /*In order to draw a triangle with DrawTrap command, MDR2 register */
                /*needs to be set appropriately prior to issue the command         */
05600000        /*Type : DrawTrap, Cmd : TrapRight*/
00500000        /*Ys : Y coordinate start position of long side
00320000        /*Xs : X coordinate start position of long side
00000000        /*dXdy : X DDA value of long side
00960000        /*XUs : X coordinate start position of upper side
00000000        /*dXUdy : X DDA value of upper side
00000000        /*XLs : X coordinate start position of lower side
00000000        /*dXLdy : X DDA value of lower side
00820000        /*USN : Number of spans (rasters) of top triangle
00000000        /*LSN : Number of spans (rasters) of bottom triangle
F101010A        /*Type : SetRegister, # of data : 1, Register address : 10A (=428h)*/
00000600        /*00000600    ->    MDR2    :    Set    drawing    mode*/
                /*Set previous drawing attribute back to the MDR2 register*/
```

Note 1: Block tiling is performed by executing DrawTrap commands to draw 2 triangles next to each other. A rectangle is split to two triangles. These two triangles are drawn using DrawTrap command with tiling option ON.
Note 2: Previously set draw attributes are continuously adopted, unless otherwise changed.