APPLICATION NOTE

SH7785 Group

SH7785 Display Unit (DU) Initial Settings Sample Program

R01AN0244EJ0101
Rev.1.01
Mar 17, 2011

Introduction

This application note presents a sample program for setting the display unit (DU) initialization items required at SH7785 startup.

Target Device

SH7785

Contents

1. Introduction

1.1 Specifications

In section 2, this application note provides supplementary documentation to the PCIC hardware manual and in section 3 it presents sample program that displays four types of image on an LCD display and turns display of each of those images on and off.

1.2 Functions Used

- Serial communication interface (SCIF1)
- Interrupt controller (INTC)
- Display unit (DU)
- DMA controller (DMAC)
- Internal memory (U memory)

1.3 Applicable Conditions

Evaluation board: Alpha Project AP-SH4A-3A, an SH-4A board

External memory (area 0): 16 MB of NOR type flash memory
Spansion S29GL128P90TFIR20

(area 3): DDR2 SDRAM: 64 MB
Micron MT47H16M16BG-3

Display output: DVI-I connector
RGB to DVI conversion IC: Texas Instruments TFP410

Microcontroller: SH7785 (R8A7785)

Operating frequencies: CPU clock: 600 MHz
SuperHyway clock: 300 MHz
DDR2 clock: 300 MHz
Bus clock: 100 MHz
Peripheral clock: 50 MHz

Area 0 bus width: 16 bits fixed (MD5 pin = low, MD6 pin = high)

Clock operating mode: Clock mode 16 (MD0 = low, MD1 = low, MD2 = low, MD3 = low, MD4 = high)

Endian mode: Big endian (MD8 = low)

Addressing mode: 29-bit addressing (MD13 = low)

Tool chain: Super-H RISC engine Standard Toolchain Version 9.3.2.0

Compiler options: Other than the options specified in the include file in the High-Performance Embedded Workshop, the default options are used.

```
-cpu=sh4a -include="$(PROJDIR)\inc","$(PROJDIR)\inc\drv"
-object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath
-global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0
-struct_alloc=1 –nologo
```

Assembler options:

```
-cpu=sh4a -endian=big -round=zero -denormalize=off -include="$(PROJDIR)\inc"
-debug -object="$(CONFIGDIR)\$(FILELEAF).obj" -literal=pool,branch,jump,return
-nolist -nologo -chgincpath -errorpath
```

1.4 Technical Terms Used in this Application Note

- Frame:
  Since pixel information directly corresponding to the image to be displayed is stored in memory, a buffer that holds this frame information is called a frame buffer. The DU module reads pixel information from a frame buffer and displays the image.

- Refresh rate (Hz):
  The refresh rate indicates how many times the image can be rewritten in one second by the LCD display connected to the DU. This will be the same as the frequency of the vertical sync signal (VSYNC) output by the DU.

- Frame rate (fps):
  The number of images updated in one second by a display system. The upper limit for the number of images the DU can update is equal to the refresh rate (in Hz).
  If the frame rate is lower than the refresh rate, for example if the frame rate is 30 fps for a DU with a 60 Hz VSYNC, then the DU must display 30 updated images for every 60 VSYNC cycles.

1.5 Applicability of this Application Note

This application note presents code that operates without an operating system installed and displays frame buffer images on a display with an RGB interface. This document describes the basic operation of the DU module. The following functions, however, are not described in this application note.

- Display data format (8 bits/pixel, ARGB 1555, YC)
- Scrolling
- Wraparound
- Blinking
- TV synchronization modes (external synchronization mode)
- Sync method switching mode
- Interlace sync mode
- Interlace sync & video mode
- YC to RGB color space conversion functions
- Color palettes
- Composite output

1.6 Related Application Notes

The sample program in this application note verifies operation under setting conditions modified from those presented in the "SH7785 Initial Settings Sample Program" SH7785 Group application note (R01AN0242EJ0101). Please refer to that application note while using this one.

2. DU Operation

2.1 DU Overview

The DU is an image system module that can read image data from a frame buffer in external memory and display that image by outputting it to an LCD display. This application note presents a sample program that reads 16-bit, RGB 565 BMP format RGB images and displays them on an RGB interface LCD display.

Figure 1 shows an overview of DU operation.

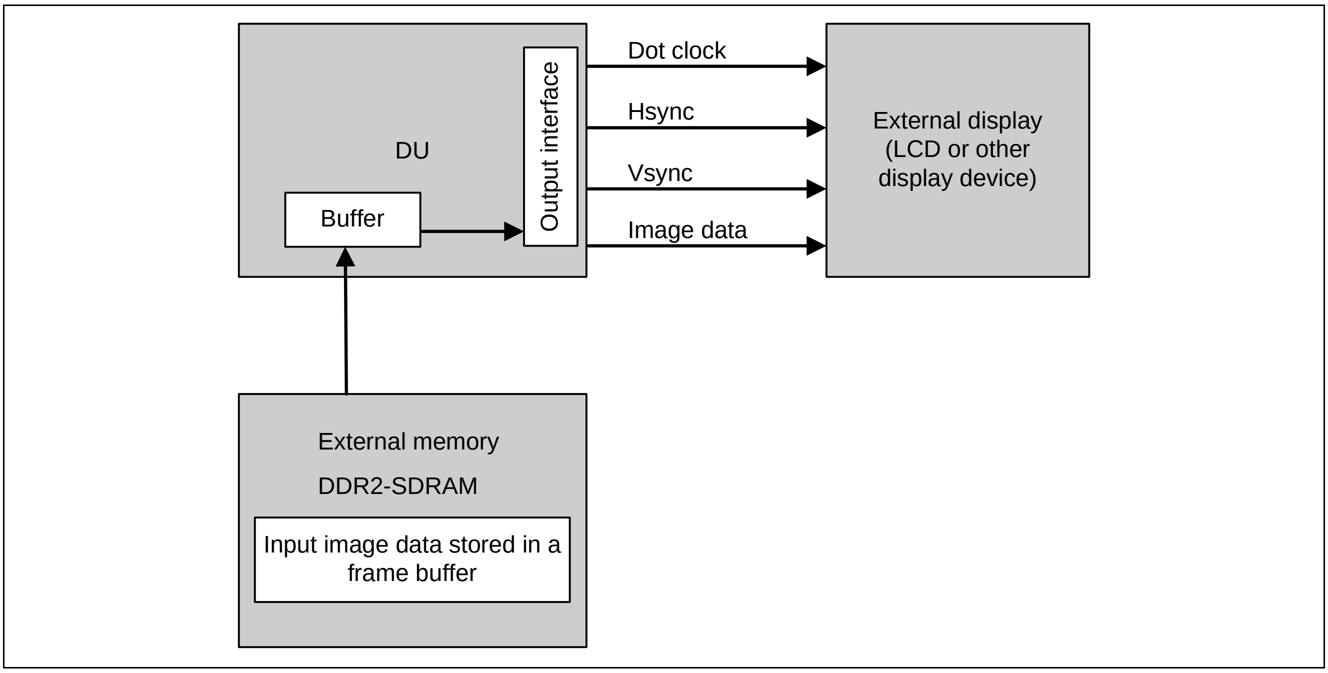


Figure 1 DU Operation Overview

2.1.1 Output Image Structure

The DU refers to an image to be displayed as a plane. There can be up to 6 planes and the number of planes that can be used may be limited by the image size. (For an image size of 480×234, up to 6 planes may be used; for the WVGA size (854×480), up to 4; and for the SVGA size (800×600), up to 3.) Planes can be superimposed and the order in which they are superimposed can be specified.

Furthermore, each plane has a double buffer structure and various functions, such as display on/off, the display data format, and blending, can be specified independently. This application note presents a sample program that uses 4 planes with a WVGA image size and sets the transparent color to superimpose the planes.

Figure 2 shows an overview of the plane structure and the superimposing.

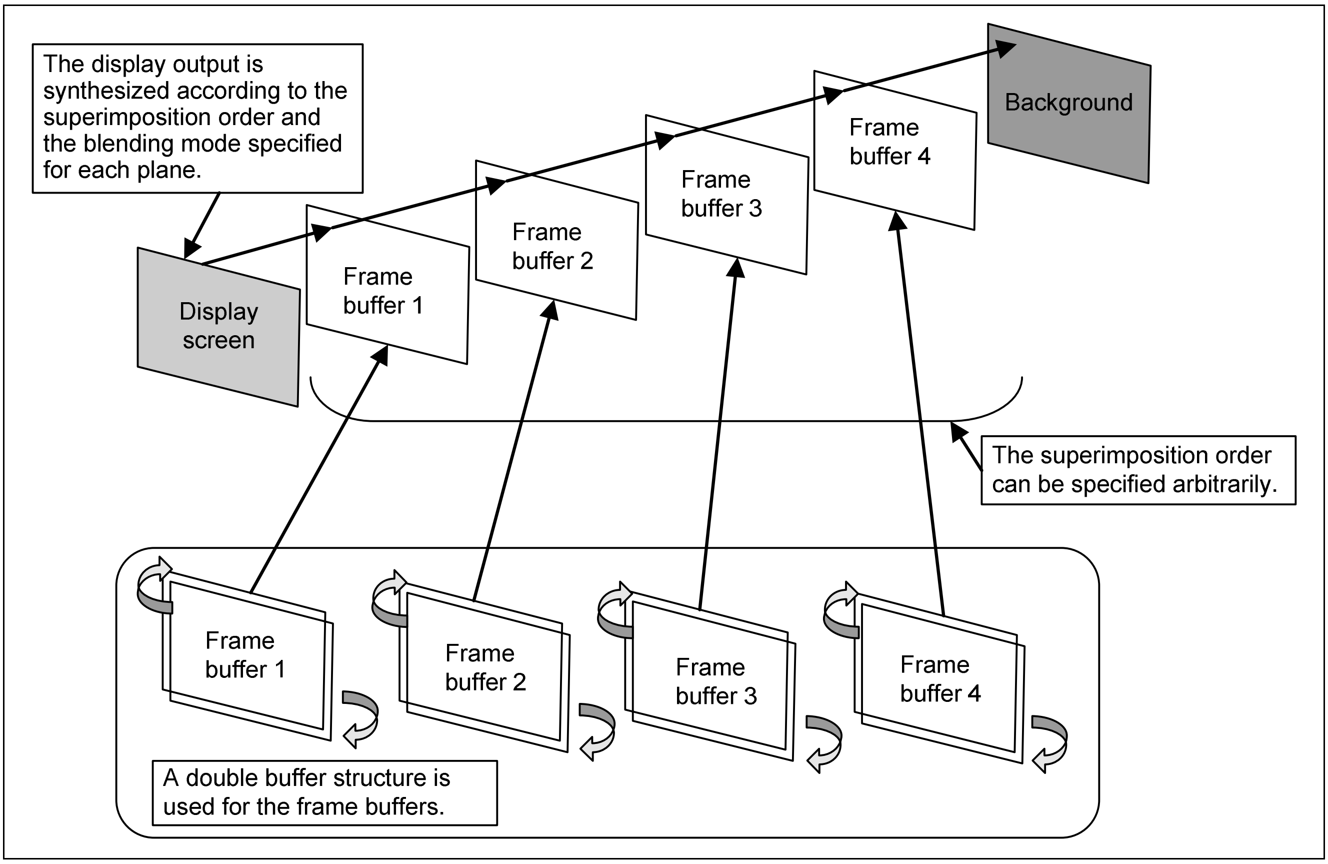

The display output is synthesized according to the superimposition order and the blending mode specified for each plane.
Display screen
Frame buffer 1
Frame buffer 2
Frame buffer 3
Frame buffer 4
Background
The superimposition order can be specified arbitrarily.
Frame buffer 1
Frame buffer 2
Frame buffer 3
Frame buffer 4
A double buffer structure is used for the frame buffers.

Figure 2 Plane Structure and Superimposition Overview

2.1.2 Image Data Readout

The DU module reads out data from the frame buffer starting at the origin at the upper left of the screen and proceeding to the right. (The start position for readout from the origin at the upper left of the frame buffer can be set with the plane n start position X register (PnSPXR) and the plane n start position Y register (PnSPYR).)

Figure 3 shows the image data input.

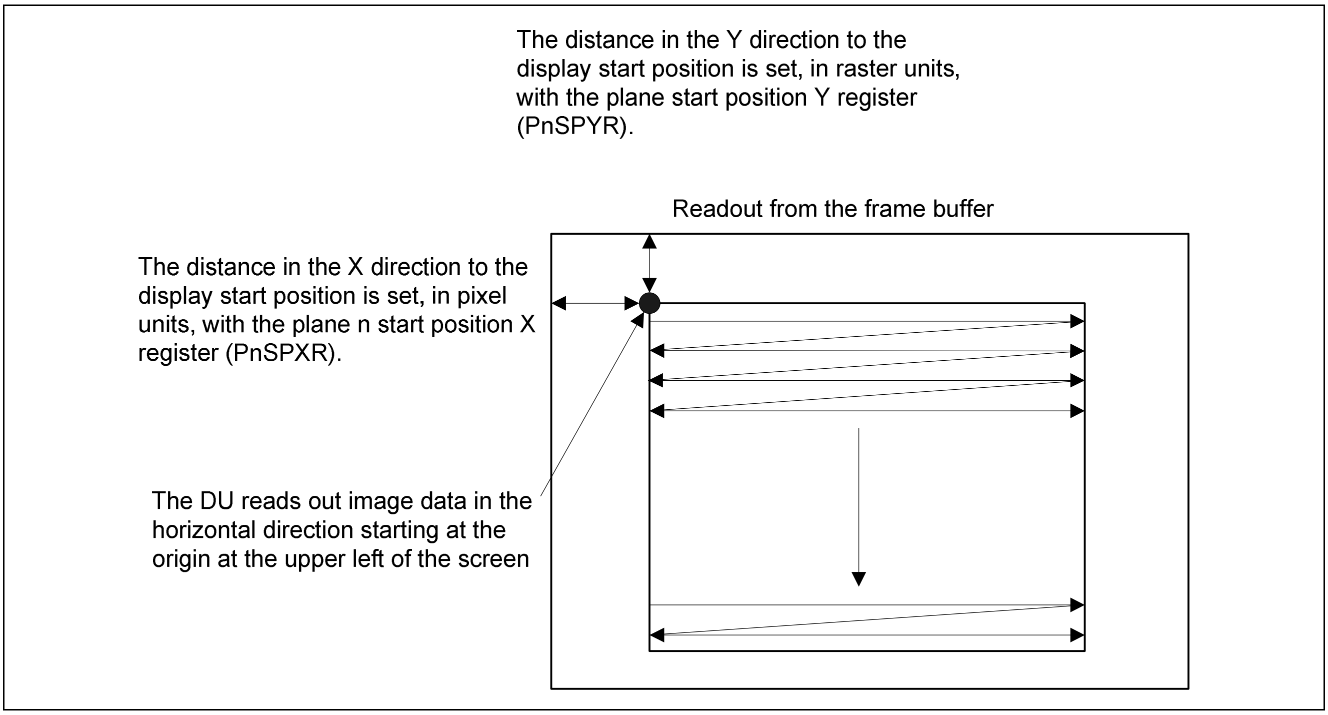

The distance in the Y direction to the display start position is set, in raster units, with the plane start position Y register (PnSPYR).
Readout from the frame buffer
The distance in the X direction to the display start position is set, in pixel units, with the plane n start position X register (PnSPXR).
The DU reads out image data in the horizontal direction starting at the origin at the upper left of the screen

Figure 3 Image Data Input

2.1.3 Image Data Output

The DU outputs image data in synchronization with the following three sync signals.

- Dot clock (DCLKOUT):
  Each pixel unit of information is output in synchronization with the dot clock.
- Horizontal sync signal (Hsync):
  Each horizontal line of image information is output in synchronization with the horizontal sync signal. The periods before and after the sync signal during which pixel information is not output are called the horizontal front porch and horizontal back porch, respectively.
- Vertical sync signal (Vsync):
  Each frame of image information is output in synchronization with the vertical sync signal. The periods before and after the sync signal during which pixel information is not output are called the vertical front porch and vertical back porch, respectively.

Figure 4 shows the pixel data and sync signal output.

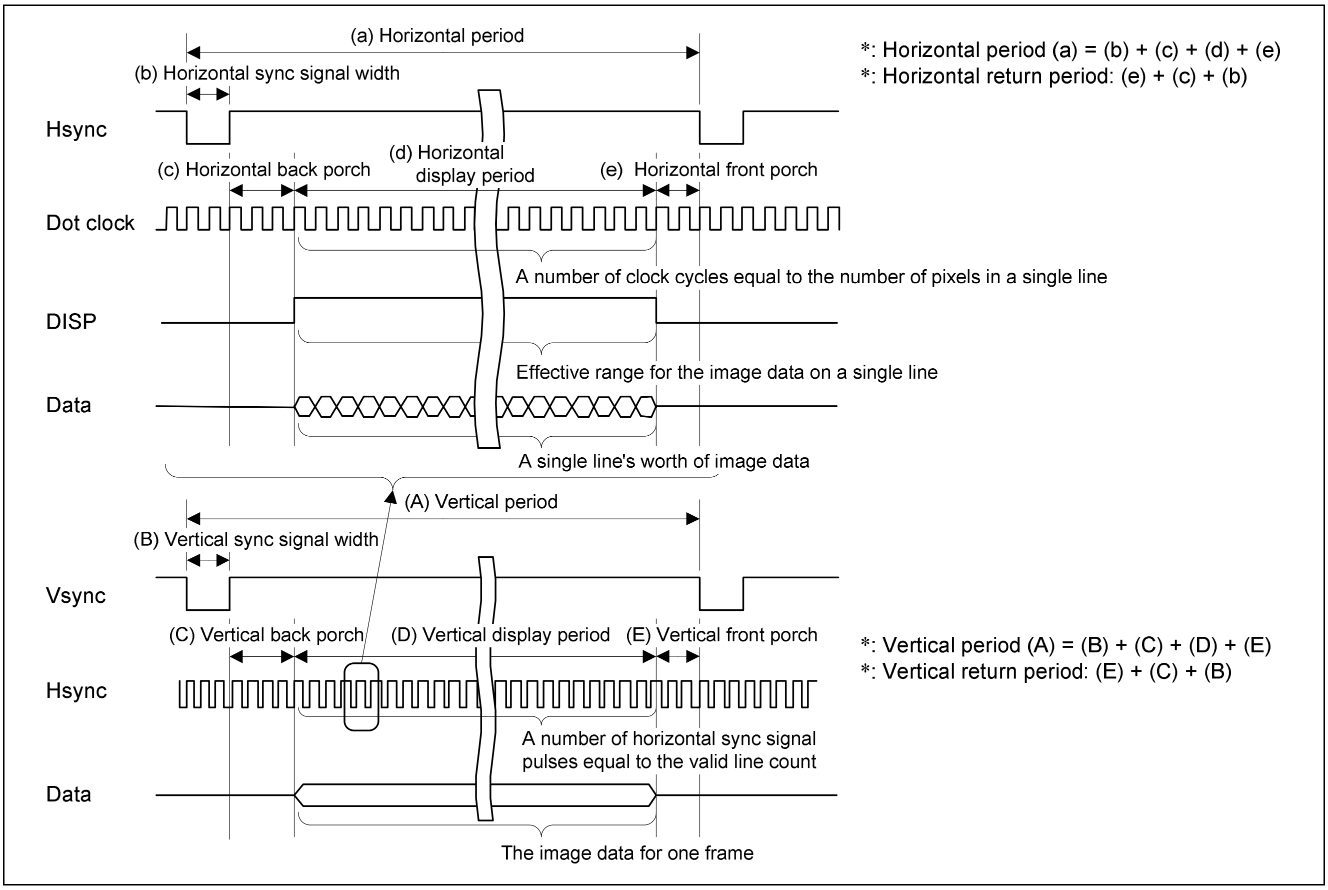

(a) Horizontal period
(b) Horizontal sync signal width
*: Horizontal period (a) = (b) + (c) + (d) + (e)
*: Horizontal return period: (e) + (c) + (b)
Hsync
(c) Horizontal back porch
(d) Horizontal display period
(e) Horizontal front porch
Dot clock
A number of clock cycles equal to the number of pixels in a single line
DISP
Effective range for the image data on a single line
Data
A single line's worth of image data
(A) Vertical period
(B) Vertical sync signal width
Vsync
(C) Vertical back porch
(D) Vertical display period
(E) Vertical front porch
*: Vertical period (A) = (B) + (C) + (D) + (E)
*: Vertical return period: (E) + (C) + (B)
Hsync
A number of horizontal sync signal pulses equal to the valid line count
Data
The image data for one frame

Figure 4 Image Data and Sync Signal Output

2.1.4 Endian Conversion

The DU can perform big endian/little endian conversion according to the setting the DSEC bit in the display system control register (DSYSR).

DU internal operation always uses little endian order, but display data stored in big endian order in memory can be read out and converted to little endian by setting the DSEC bit in the display system control register (DSYSR) to 1.

Table 1 lists the endian conversion units.

Table 1 Endian Conversion Units

PnMR/PnDDF	Data Format	Endian Conversion Units
00	8 bits/pixel	Byte units
01	16 bits/pixel	Word units
10	ARGB	Word units
11	YC	Byte units

Figure 5 shows the endian conversion for each of these units.

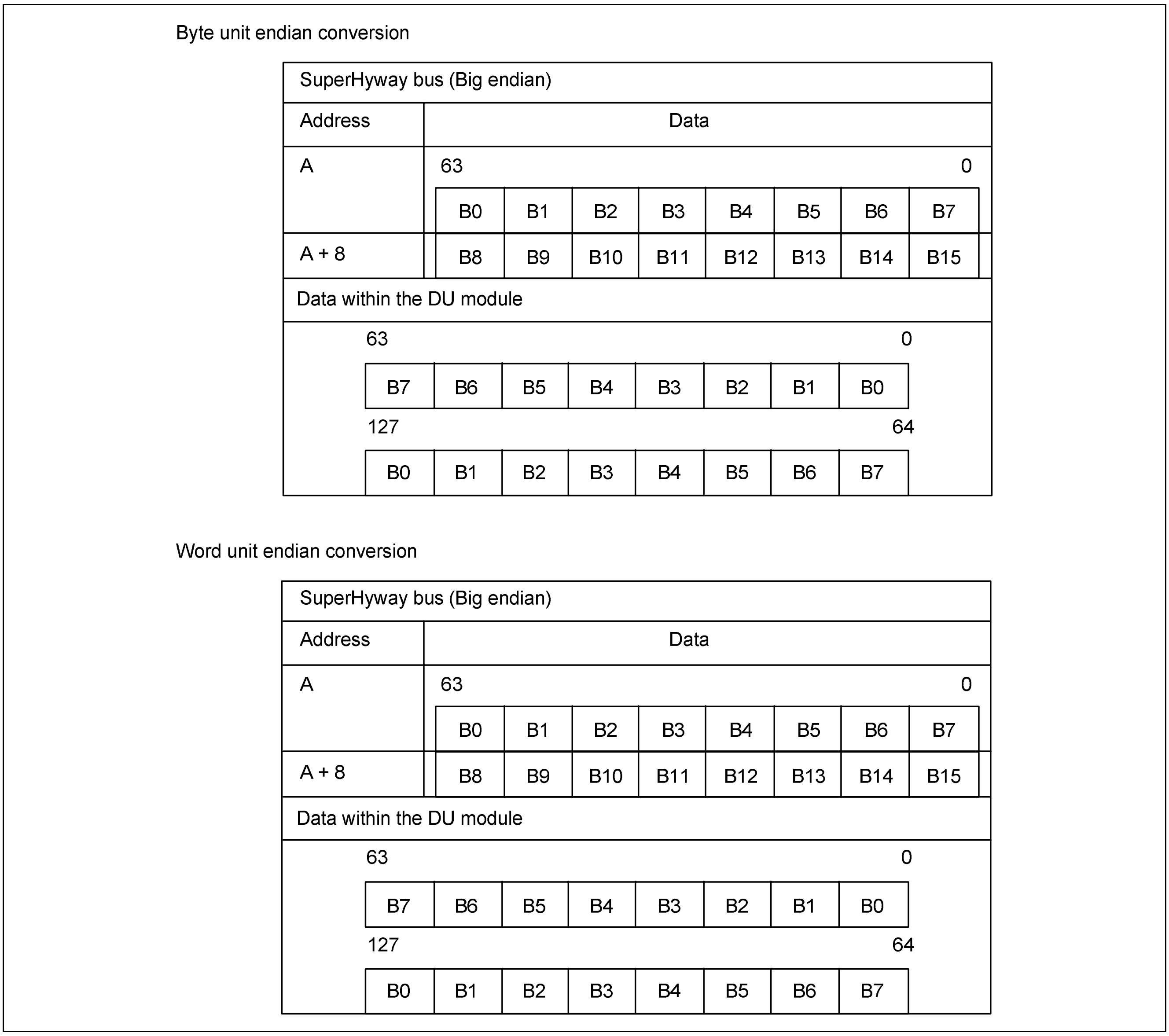

Byte unit endian conversion
SuperHyway bus (Big endian)
Address
Data
A
63
0
B0 B1 B2 B3 B4 B5 B6 B7
A + 8
B8 B9 B10 B11 B12 B13 B14 B15
Data within the DU module
63
0
B7 B6 B5 B4 B3 B2 B1 B0
127
64
B0 B1 B2 B3 B4 B5 B6 B7
Word unit endian conversion
SuperHyway bus (Big endian)
Address
Data
A
63
0
B0 B1 B2 B3 B4 B5 B6 B7
A + 8
B8 B9 B10 B11 B12 B13 B14 B15
Data within the DU module
63
0
B7 B6 B5 B4 B3 B2 B1 B0
127
64
B0 B1 B2 B3 B4 B5 B6 B7

Figure 5 Endian Conversion

2.1.5 Scrolling

The DU module can perform smooth scrolling by setting the display area, display size, and start position for each plane in plane units.

To perform scrolling display, an application cyclically sets the plane n display start position X and Y coordinates (specified with the plane n start position X register (PnSPXR) and the plane start position Y register (PnSPYR)) with the start of memory specified by the display area start address registers 0 and 1 (PnDSA0R and PnDSA1R) for each plane as the origin.

Figure 6 presents an overview of scrolling display. Scrolling display is implemented by moving the display start position from A to B.

Note: The display size and other settings for each plane must be made so that data outside the memory structure area is not displayed.

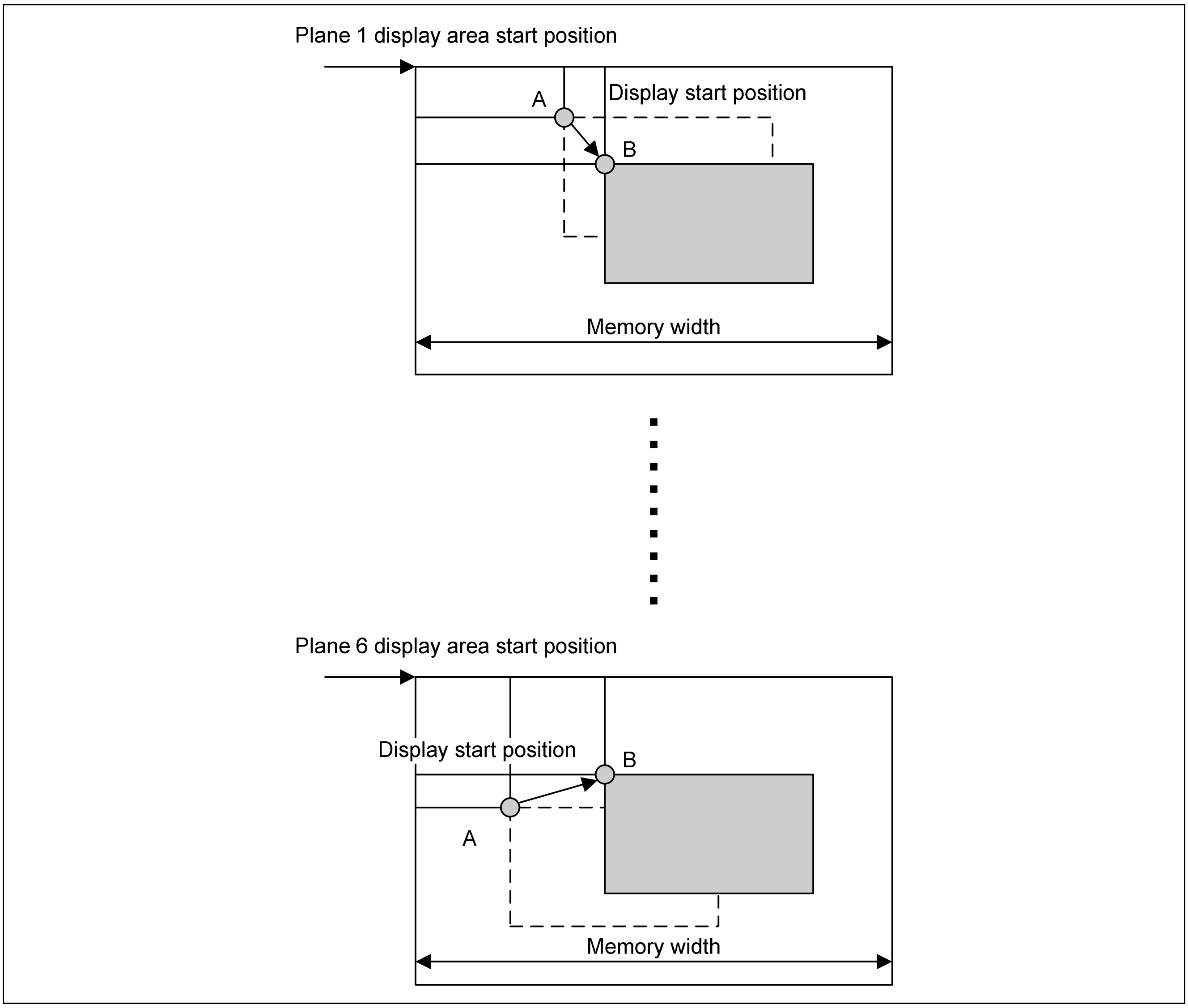


Figure 6 Scrolling Function Overview

2.2 DU Specifications

Table 2 lists the main DU module specifications.

Table 2 DU Specifications

Item	Specification
Maximum size displayed (maximum numbers of planes)	Image size 480 × 234: Maximum number of planes: 6 Image size 854 × 480: Maximum number of planes: 4 Image size 800 × 600: Maximum number of planes: 3
Input data format	8bit/pixel 16bit/pixel:RGB 16bit/pixel:ARGB YC: UYVY format or YUYV format
Output data format	8bit/pixel 16bit/pixel:RGB 16bit/pixel:ARGB YC→RGB
DU output interface	Interface using horizontal and vertical sync signals The signal polarity can be set. The pixel signal output width and output position can be set. A composite sync signal can be output*[1]. Note * : Not used in this application note.
Dot clock	One of two source clocks and a divisor can be selected. External input clock (DCLKIN) DU clock (DUck) Image output can operate at up to 50 MHz. The divisor can be set to be an integer from 1 to 32.
Plane	Up to 6 planes. The on/off state of display for each plane can be set. The superimposition priority can be set. The display size can be set.

2.3 DU Settings

This section describes the DU function and register settings.

2.3.1 Display Output Settings

In addition to Hsync, Vsync, and the digital RGB outputs, the DU can also output a composite sync signal.

The composite sync signal output is not used in this application note.

Table 3 lists the DU registers used for display output settings.

Table 3 Display Output Settings

Function	Register
Display output mode setting	Display unit system control register (DSYSR)
Display output signal settings	Display mode register (DSMR)

2.3.2 Plane Parameter Settings

The frame buffer information required when the DU output image data from memory is shown in figure 7 in conjunction with the plane settings.

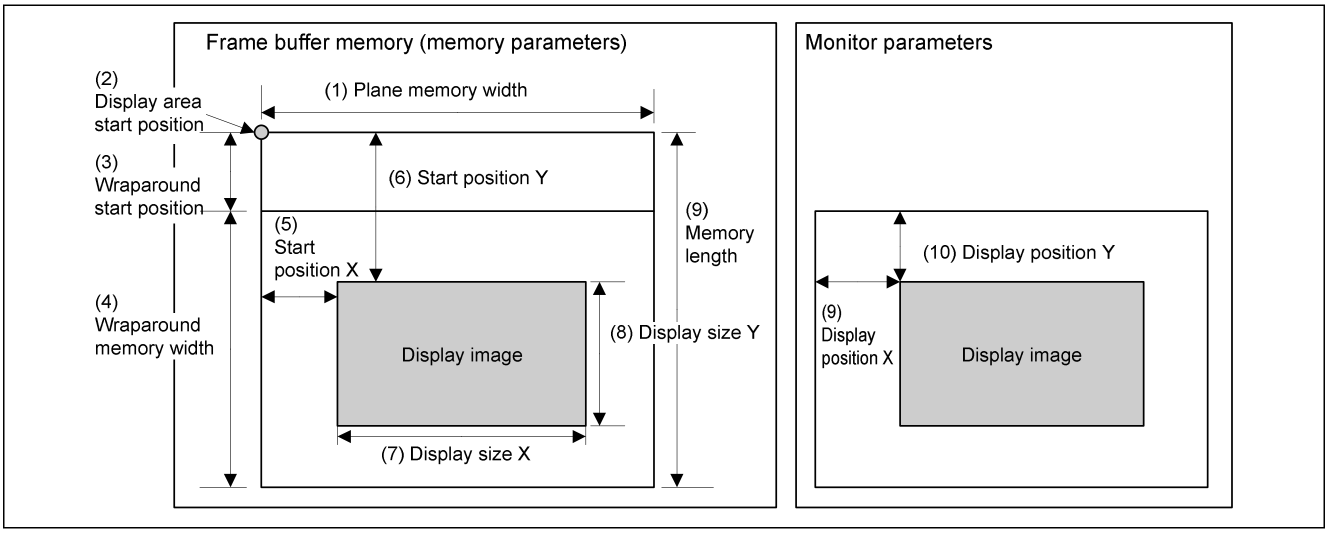


Figure 7 Plane Structure

Table 4 lists the registers used for the DU plane parameter settings.

Table 4 Plane Setting Registers

No.	Function	Register	Description
1	Plane memory width (MWX)	PnMWXR	Sets the plane X direction memory width from 16 to 4096 pixels in 16-pixel units.
2	Display area start position (DSA)	PnDSA0, PnDSA1	See section 2.3.3, Memory Allocation.
3	Wraparound start position (WASPY)	PnWASPR	Sets the wraparound area Y direction start position for each plane in line units referenced to the address set with DSA.
4	Wraparound memory width (WAMWY)	PnWAMWR	Sets the wraparound Y direction memory width to an arbitrary range from 240 to 4095 lines.
5	Start position X (SPX)	PnSPXR	Sets the distance from the display start position in the X direction in pixel units with the address set with DSA as the origin.
6	Start position Y (SPY)	PnSPYR	Sets the distance from the display start position in the Y direction in raster units with the address set with DSA as the origin.
7	Display size X (DSX)	PnDSXR	Sets the display size for each plane in the X direction in pixel units.
8	Display size Y (DSY)	PnDSYR	Sets the display size for each plane in the Y direction in raster units.
9	Display position X (DPX)	PnDPXR	Sets the distance from the display position in the X direction in pixel units with the monitor's upper left as the origin.
10	Display position Y (DPY)	PnDPYR	Sets the distance from the display position in the Y direction in raster line units with the monitor's upper left as the origin.

Note: n = 1 to 6

2.3.3 Memory Allocation

The display plane display area start address can be set individually for each DU plane. Each of the display area start address registers is set to the start address for the memory area used as a 29 or 32-bit value. The DSAE bit in the display extended function enable register (DFER) must be set to 1 if 32-bit addresses are used.

Furthermore, each plane is displayed using double buffer control using display area start address registers 0 and 1 for each plane.

Table 5 list the DU memory allocation setting registers.

Table 5 Memory Allocation Setting Registers

Display Screen	Setting Register	
Plane 1	Plane 1 display area start address register 0	P1DSA0
	Display area start address register 1	P1DSA1
Plane 2	Plane 2 display area start address register 0	P2DSA0
	Display area start address register 1	P2DSA1
Plane 3	Plane 3 display area start address register 0	P3DSA0
	Display area start address register 1	P3DSA1
Plane 4	Plane 4 display area start address register 0	P4DSA0
	Display area start address register 1	P4DSA1
Plane 5	Plane 5 display area start address register 0	P5DSA0
	Display area start address register 1	P5DSA1
Plane 6	Plane 6 display area start address register 0	P6DSA0
	Display area start address register 1	P6DSA1

Table 6 lists the DU set address 32-bit extension enable register.

Table 6 Set Address 32-Bit Extension

Function	Register	Bit Name
Enabling 32-bit extension of the set addresses	Display extended function enable register (DEFR)	DSAE

2.3.4 Plane Priority Settings

The DU allows the priority for plane superposition to be set.

Also, the enabled/disabled state of prioritized display can be set.

Table 7 lists the registers used for setting the DU plane priorities and the prioritization enabled/disabled state.

Table 7 Plane Priorities and Display Prioritization Enabled/Disabled Settings

Function	Register
Priority settings	Display plane priority register (DPPR)
Display prioritization enable/disable setting	

2.3.5 Background Settings

The DU supports setting the color displayed when the display size or use of the transparent color results in an area where there is no plane data to be displayed.

Table 8 lists the registers used for setting the background color and figure 8 shows the background color display.

Table 8 Background Color Settings

Function	Register
Background color settings	Background color register (BPOR)

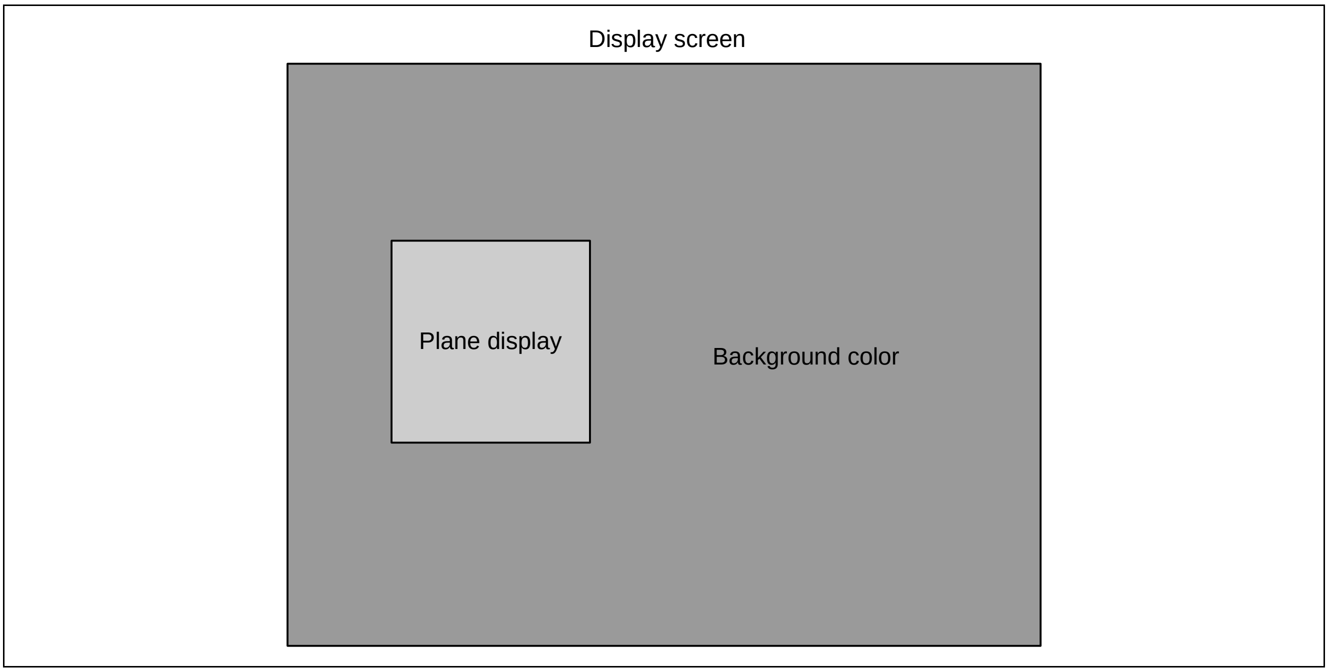


Figure 8 Background Color Display

2.3.6 Sync Signal Settings

The sync signal parameter shown in figure 9 must be set to match the specifications (A) to (E) and (a) to (e) in figure 4 for the target display (e.g. LCD display) used.

Also, table 9 lists registers used to set the DU display timing parameters.

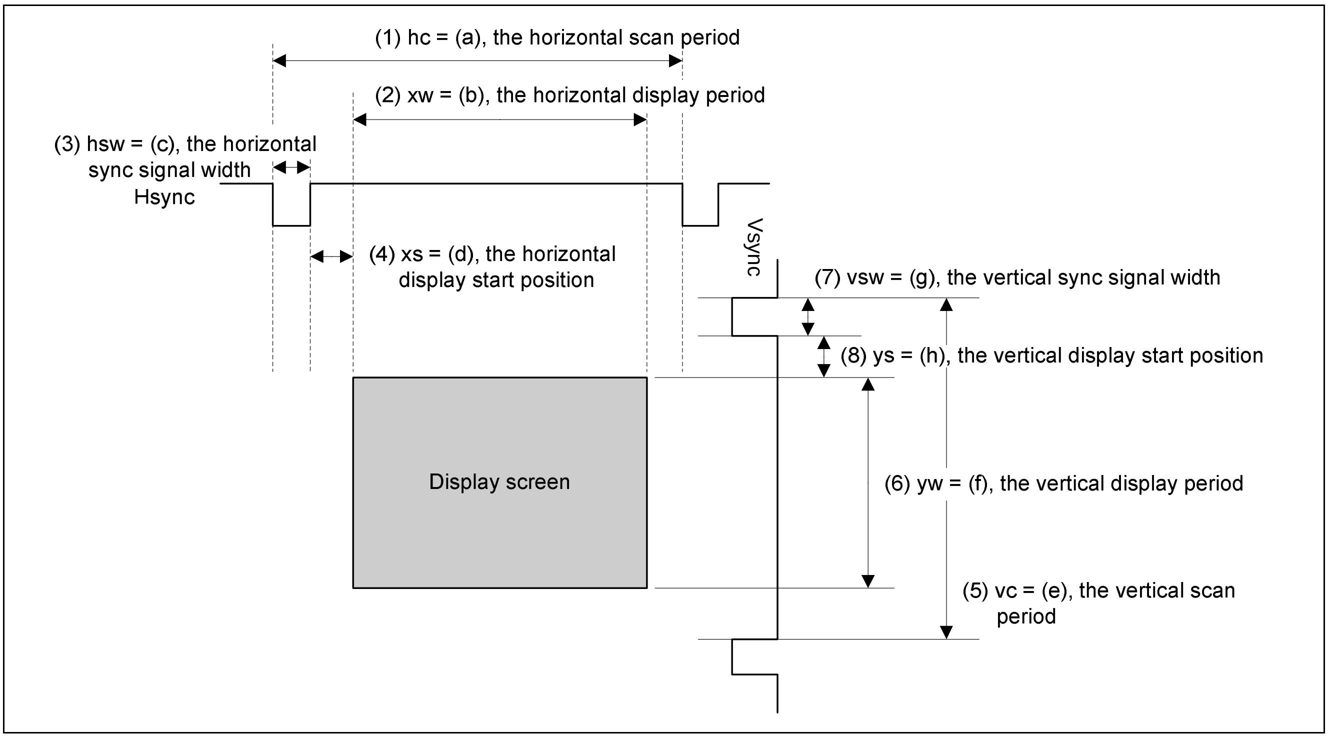

(1) hc = (a), the horizontal scan period
(2) xw = (b), the horizontal display period
(3) hsw = (c), the horizontal sync signal width
Hsync
(4) xs = (d), the horizontal display start position
Vsync
(7) vsw = (g), the vertical sync signal width
(8) ys = (h), the vertical display start position
Display screen
(6) yw = (f), the vertical display period
(5) vc = (e), the vertical scan period

Figure 9 Sync Signal Structure

Table 9 Display Timing Settings Registers

Register	Bit Name	Sync Type (Master Mode)
Horizontal display start position register (HDSR)	HDS	hsw + xs – 19
Horizontal display end position register (HDER)	HDE	hsw + xs – 19 + xw
Vertical display start position register (VDSR)	VDS	ys – 2
Vertical display end position register (VDER)	VDE	ys – 2 + yw
Horizontal sync signal pulse width register (HSWR)	HSW	hsw – 1
Horizontal scan period register (HCR)	HC	hc – 1
Vertical sync position register (VSPR)	VSP	vc – vsw – 1
Vertical scan period register (VCR)	VC	vc – 1

Notes on these settings:

These values must be set to meet this condition: (hsw + xs + xw) < (hc + 18 (decimal))
These values must be set to meet this condition: (vsw + ys + yw) < vc
VDS must be set to 1 or larger.
These values must be set to meet this condition: HDE < HC

2.3.7 Dot Clock Settings

The dot clock must be set up according to the AC characteristics of the target display (e.g. LCD display) used. The dot clock is generated by dividing the source clock.

- Source clock: Either the DU clock (DUck) or an external clock input to the DCLKIN pin can be selected. If the DU clock is selected, the DCKE bit in the display extended function enable register (DEFR) must be set to 1. Since only an integer divisor (1/1, 1/2, 1/3, ...) can be set, a source clock that is an integer multiple of the desired dot clock frequency must be selected. The following restrictions also apply.
  - If the DU clock is selected, the frequency of the divided dot clock generated by the dot clock generation circuit must be 50 MHz or lower.
  - If an external clock input is used, its frequency must be 50 MHz or lower.
- Either a divisor of 1 (1/1), or a divisor other than 1 (1/2, 1/3, ... 1/32) can be selected. If a divisor of 16 or larger is used, the DCKE bit in the display extended function enable register (DEFR) must be set to 1.

Table 10 lists the registers used for DU clock settings.

Table 10 Clock Settings Registers

Function	Register	Bit Name
DU clock setting	Frequency control register (FRQCR1)	S3FC3 to S3FC 0
DUck selection enable Enables the use of divisors from 17 to 32.	Display extended function control register (DEFR)	DCKE
Source clock selection Dot clock output enable Dot clock divisor setting	External sync control register (ESCR)	DCLKSEL DCLKDIS FRQSEL

2.3.8 Display Interface Settings

The output mode, signal selections, and signal polarity must be set according to the specifications of the target display (e.g. LCD display) used.

- Output mode: Select either master mode or TV sync mode. This application note presents a sample program that selects master mode.
- Signal selection: Select the following signals: VSYNC, ODDF, DISP, and CSYNC.
  - VSYNC pin: Select VSYNC or CSYNC output from the VSYNC pin. In this application note, VSYNC is selected.
  - ODDF: Select ODDF or CLAMP output from the ODDF pin. Since this pin is not used for this application note, the initial value is not changed.
  - DISP: Select DISP, CSYNC, or DE output from the DISP pin. In this application note, DISP is selected.
  - CSYNC: Select CSYNC or HSYNC output from the HSYNC pin. In this application note, HSYNC is selected.
- Signal polarity: Select the polarity for DISP, HSYNC, and VSYNC.
  - DISP: Select the display period polarity. In this application note, active high is selected.
  - HSYNC: Select the horizontal sync signal polarity. In this application note, active low is selected.
  - VSYNC: Select the vertical sync signal polarity. In this application note, active low is selected.

See sections 19.3.1, Display System Control Register (DSYSR), and 19.3.2, Display Mode Register (DSMR), in section 19, Display Unit (DU), in the SH7785 Hardware Manual (REJ09B0261) for details on these settings.

Note: Nothing will be displayed if the image goes outside the display memory area or outside the monitor display area. Also, display data cannot be guaranteed (contaminated data may be displayed) if the display position is displaced and areas outside the display memory are referenced.

Table 11 lists the DU registers used to set up the display interface.

Table 11 Display Interface Settings

Function	Register
Output mode	Display system control register (DSYSR)
Signal selection Signal polarity	Display mode register (DSMR)

3. Application Example

This section describes the sample pin settings and sample settings code as a sample program for displaying images on an LCD display using the DU module.

3.1 LCD Display Specifications

This section presents the specifications of the LCD display used in this application note. The display used is a VESA standard conforming LCD display connected with a DVI cable. An RGB to DVI converter IC (the Texas Instruments TFP410) is used in the DVI output.

3.1.1 DVI Converter Specifications

Table 12 lists the specifications of the TFP410 IC used in this application note.

Table 12 TFP410 Specifications (From the product's data sheet)

Item	Specification
Resolution	From VGA through UXGA (This sample program uses WVGA.)
Frequency	From 25 to 125 MHz (This sample program uses 33 MHz.)
Input signals	CMOS, 8-bit digital RGB outputs

Table 13 lists the TFP410 pin function used by this application note.

Table 13 TFP410 Pin Functions

Item	Specification
IDCK+	Dot clock input
HSYNC	Horizontal sync signal input
VSYNC	Vertical sync signal input
DE	Display start signal input
D23 to D16 (R7 to R0)	Red data signal input (8 bits, MSB: R7, LSB: R0)
D15 to D8 (G7 to G0)	Green data signal input (8 bits, MSB: G7, LSB: G0)
D7 to D0 (B7 to 0)	Blue data signal input (8 bits, MSB: B7, LSB: B0)
TXC+, TXC–	Differential pair clock output
TX0+, TX0–	Differential pair data 0 output
TX1+, TX1–	Differential pair data 1 output
TX2+, TX2–	Differential pair data 2 output

3.1.2 DVI Connector Pin Functions

Table 14 lists the pin functions of the DVI connector used in this application note.

Table 14 DVI Connector Pin Functions

Item	Specification
TXC+, TXC–	Differential pair clock
TX0+, TX0–	Differential pair data 0
TX1+, TX1–	Differential pair data 1
TX2+, TX2–	Differential pair data 2

3.1.3 Interface Timing

Table 15 lists the WVGA timing used in this application note.

Also, figure 9 shows the sync signal structure and correspondence.

Table 15 WVGA Timing

Item		Value	Unit	Correspondence with Figure 9
CLK	Frequency	37.5	MHz	
HSYNC	Total	1258	CLK	Horizontal sync period (hc)
	Back porch	110		Horizontal display start position (xs)
	Front porch	220		hc – (hsw + xs + xw)
	Valid display period	800		Horizontal display period (xw)
	Sync signal width	128		Horizontal sync signal width (hsw)
VSYNC	Total	525	HSYNC	Vertical sync period (yc)
	Back porch	35		Vertical display start position (ys)
	Front porch	5		vc – (vsw + ys + yw)
	Valid display period	480		Vertical display period (yw)
	Sync signal width	5		Vertical sync signal width (ysw)

Table 16 lists sample calculations for values set in registers.

Table 16 WVGA Timing Register Settings

Function	Register	Bit Name	Value	Sample Setting Value
Horizontal timing settings	Horizontal display start position register (HDSR)	HDS	hsw + xs – 19 = 128 + 110 – 19 = 219	0xDB
	Horizontal display end position register (HDER)	HDE	hsw + xs –19 + xw = 128 + 110 – 19 + 800 = 1019	0x3FB
	Horizontal sync signal pulse width register (HSWR)	HSW	hsw – 1 = 128 – 1 = 127	0x7F
	Horizontal scan period register (HCR)	HC	hc – 1 = 1258 – 1 = 1257	0x4E9
Vertical timing settings	Vertical display start position register (VDSR)	VDS	ys – 2 = 5 – 2 = 3	0x3
	Vertical display end position register (VDER)	VDE	ys – 2 + yw = 5 – 2 + 480 = 483	0x1E3
	Vertical sync position register (VSPR)	VSP	vc – vsw – 1 = 525 – 5 – 1 = 519	0x207
	Vertical scan period register (VCR)	VC	vc – 1 = 525 – 1 = 524	0x20c

Table 17 lists the sample clock settings used in this application note.

Table 17 Clock Settings

Function	Register	Bit Name	Value	Sample Setting Value
Dot clock	Frequency control register (FRQCR1)	S3FC3 to S3FC0	Input CLK = 33.33 MHz 33.33 * 9/2 = 150 MHz	0x4
	Display extended function control register (DEFR)	DCKE	DUck enabled	0x1
	External sync control register (ESCR)	DCLKSEL	Source clock: DUck	0x1
		DCLKDIS	DCLKOUT	0x1
		FRQSEL	150 MHz/4 = 37.5 MHz	0x3

Since the LCD dot clock specifications call for a frequency of about 34 MHz with a horizontal period of 31.5 kHz. Since the dot clock actually used is 37.5 MHz, the frequency is somewhat high and as a result the image will be shifted left or right somewhat. To correct this image displacement, the horizontal sync period can be increased to adjust the frequency horizontal sync period frequency to be closer to 31.5 kHz and the front porch and back porch adjusted by one pixel at a time to correct the left/right displacement. In the evaluation board used in this application note, since the DVI conversion IC has a DISP signal (display start signal) input, display position displacement can be avoided without adjustment by using this DISP signal.

Whether or not the DISP signal is present depends on the specifications of the LCD module and the signal conversion IC (analog RGB, DVI, or other standard).

3.2 Sample DVI Connection Circuit

3.2.1 Sample Pin Connections

Figure 10 shows the sample DVI connection circuit used in the application note.

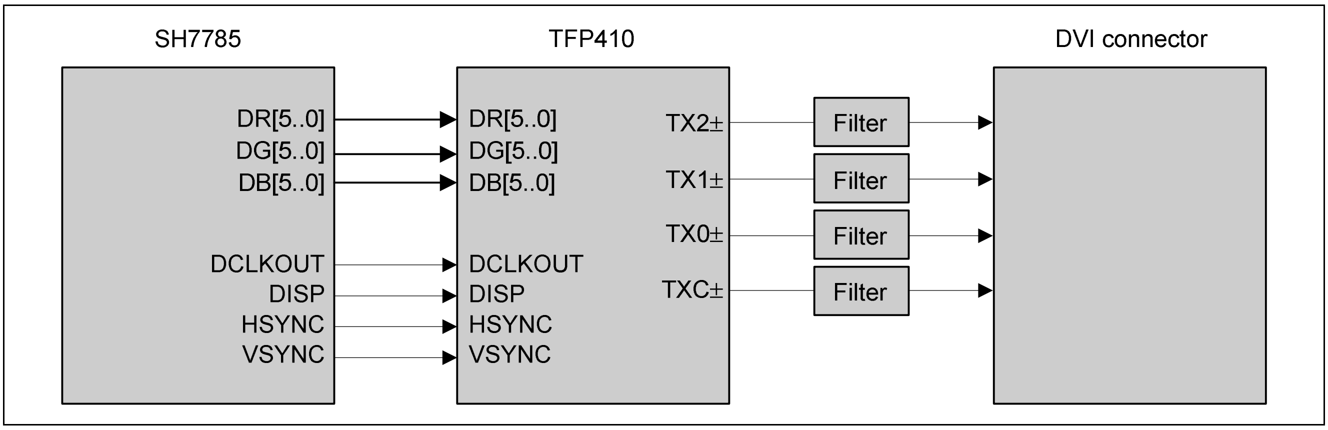


Figure 10 Sample Connection Circuit

3.3 Sample Program Specifications

This section presents the specifications of the sample program and its flowcharts.

3.3.1 Specifications

(1) Five BMP files are displayed using four planes in a WVGA size display.

- Plane 1: Draws an 80×80 size image.
- Plane 2: Draws an 80×80 size image using two BMP files.
- Plane 3: Draws an 80×80 size image.
- Plane 4: Draws a 100×80 size image.

(2) The DU is started and the following processing is performed at each frame interrupt.

In this sample program, the scrolling display function is not used for changing the display position of the images, but rather the display position on the monitor is changed using the plane display position X and Y registers. As a result, this sample program implements functionality equivalent to that of a mouse pointer.

- Plane 1: The initial position is at the far upper right of the monitor. The following operations (1) to (4) are then repeated.
  (1): When the image reaches the far upper right of the monitor, the value of the plane 1 display X register (P1DPXR) is decremented by 2 to move the image to the far upper left of the monitor.
  (2): When the image reaches the far upper left of the monitor, the value of the plane 1 display Y register (P1DPYR) is incremented by 2 to move the image to the far lower left of the monitor.
  (3): When the image reaches the far lower left of the monitor, the value of the plane 1 display X register (P1DPXR) is incremented by 2 to move the image to the far lower right of the monitor.
  (4): When the image reaches the far lower right of the monitor, the value of the plane 1 display Y register (P1DPYR) is decremented by 2 to move the image to the far upper right of the monitor.

- Plane 2: The initial position is at the far upper left of the monitor. The following operations (1) to (4) are then repeated. Every 20 frames, P2DC in the plane 2 mode register (P2MR) is set to 1 to switch the frame buffer and update the image.
  (1): When the image reaches the far upper left of the monitor, the value of the plane 2 display X register (P2DPXR) is incremented by 1 to move the image to the far upper right of the monitor.
  (2): When the image reaches the far upper right of the monitor, the value of the plane 2 display Y register (P2DPYR) is incremented by 1 to move the image to the far lower right of the monitor.
  (3): When the image reaches the far lower right of the monitor, the value of the plane 2 display X register (P2DPXR) is decremented by 1 to move the image to the far lower left of the monitor.
  (4): When the image reaches the far lower left of the monitor, the value of the plane 2 display Y register (P2DPYR) is incremented by 1 to move the image to the far upper left of the monitor.

- Plane 3: The initial position is at the far lower left of the monitor. The following operations (1) to (6) are then repeated.
  (1): When the image reaches the far upper left of the monitor, the values of the plane 2 display X register (P2DPXR) and the plane 2 display Y register (P2DPYR) are both incremented by 2 to move the image diagonally to the far lower right.
  (2): When the image reaches the far lower right of the monitor, the values of the plane 2 display X register (P2DPXR) and the plane 2 display Y register (P2DPYR) are both decremented by 2 to move the image diagonally to the far upper left.
  (3): When the image reaches the left edge of the monitor, the value of the plane 2 display X register (P2DPXR) is incremented by 2 and the plane 2 display Y register (P2DPYR) increment/decrement operation remains the same to move the image diagonally to the right edge.
  (4): When the image reaches the top edge of the monitor, the value of the plane 2 display Y register (P2DPYR) is incremented by 2 and the plane 2 display X register (P2DPXR) increment/decrement operation remains the same to move the image diagonally to the lower edge.
  (5): When the image reaches the right edge of the monitor, the value of the plane 2 display X register (P2DPXR) is decremented by 2 and the plane 2 display Y register (P2DPYR) increment/decrement operation remains the same to move the image diagonally to the left edge.

(6): When the image reaches the lower edge of the monitor, the value of the plane 2 display Y register (P2DPYR) is decremented by 2 and the plane 2 display Y register (P2DPYR) increment/decrement operation remains the same to move the image diagonally to the top edge.

— Plane 4: The initial position is at the far upper left of the monitor. The following operations (1) to (6) are then repeated. The displayed image has a size of 8 horizontally by 6 vertically, for a total of 48, on the monitor.

(1): The image is added to image memory.

(2): A DMA transfer from image memory to the frame buffer set in plane 4 display area start address 0 or 1 (P4DSA0R or P4DSA1R) is performed.

(3): Every 60 frames, the image is updated by switching the frame buffer by setting P4DC in the plane 4 mode register (P4MR) to 1.

(4): If 48 images have been displayed on the monitor, images are deleted one at a time from image memory.

(5): The same operation as step (2).

(6): The same operation as step (3).

(3) The processing of (2) operates as an infinite loop.

(4) The on/off state of the display of planes 1 to 4 is controlled from the console.

3.3.2 Sample Program Main Flowchart

Figure 11 shows the flowchart for the main section of the sample program.

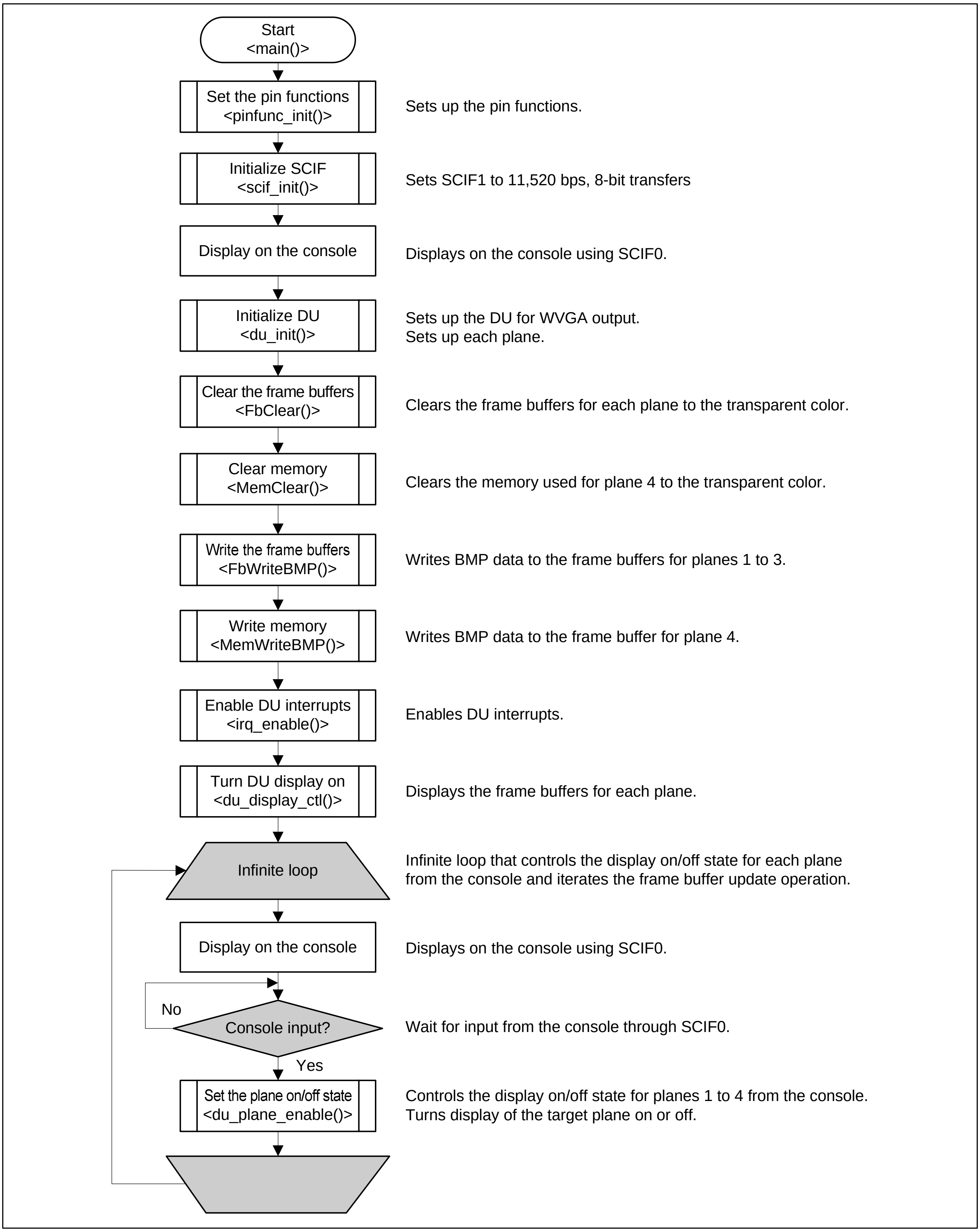

Start
<main()>
Set the pin functions
<pinfunc_init()>
Sets up the pin functions.
Initialize SCIF
<scif_init()>
Sets SCIF1 to 11,520 bps, 8-bit transfers
Display on the console
Displays on the console using SCIF0.
Initialize DU
<du_init()>
Sets up the DU for WVGA output.
Sets up each plane.
Clear the frame buffers
<FbClear()>
Clears the frame buffers for each plane to the transparent color.
Clear memory
<MemClear()>
Clears the memory used for plane 4 to the transparent color.
Write the frame buffers
<FbWriteBMP()>
Writes BMP data to the frame buffers for planes 1 to 3.
Write memory
<MemWriteBMP()>
Writes BMP data to the frame buffer for plane 4.
Enable DU interrupts
<irq_enable()>
Enables DU interrupts.
Turn DU display on
<du_display_ctl()>
Displays the frame buffers for each plane.
Infinite loop
Infinite loop that controls the display on/off state for each plane from the console and iterates the frame buffer update operation.
Display on the console
Displays on the console using SCIF0.
No
Console input?
Wait for input from the console through SCIF0.
Yes
Set the plane on/off state
<du_plane_enable()>
Controls the display on/off state for planes 1 to 4 from the console.
Turns display of the target plane on or off.

Figure 11 Sample Program Main Flowchart

3.3.3 Pin Function Settings

Figure 12 shows the flowchart for setting the pin functions.

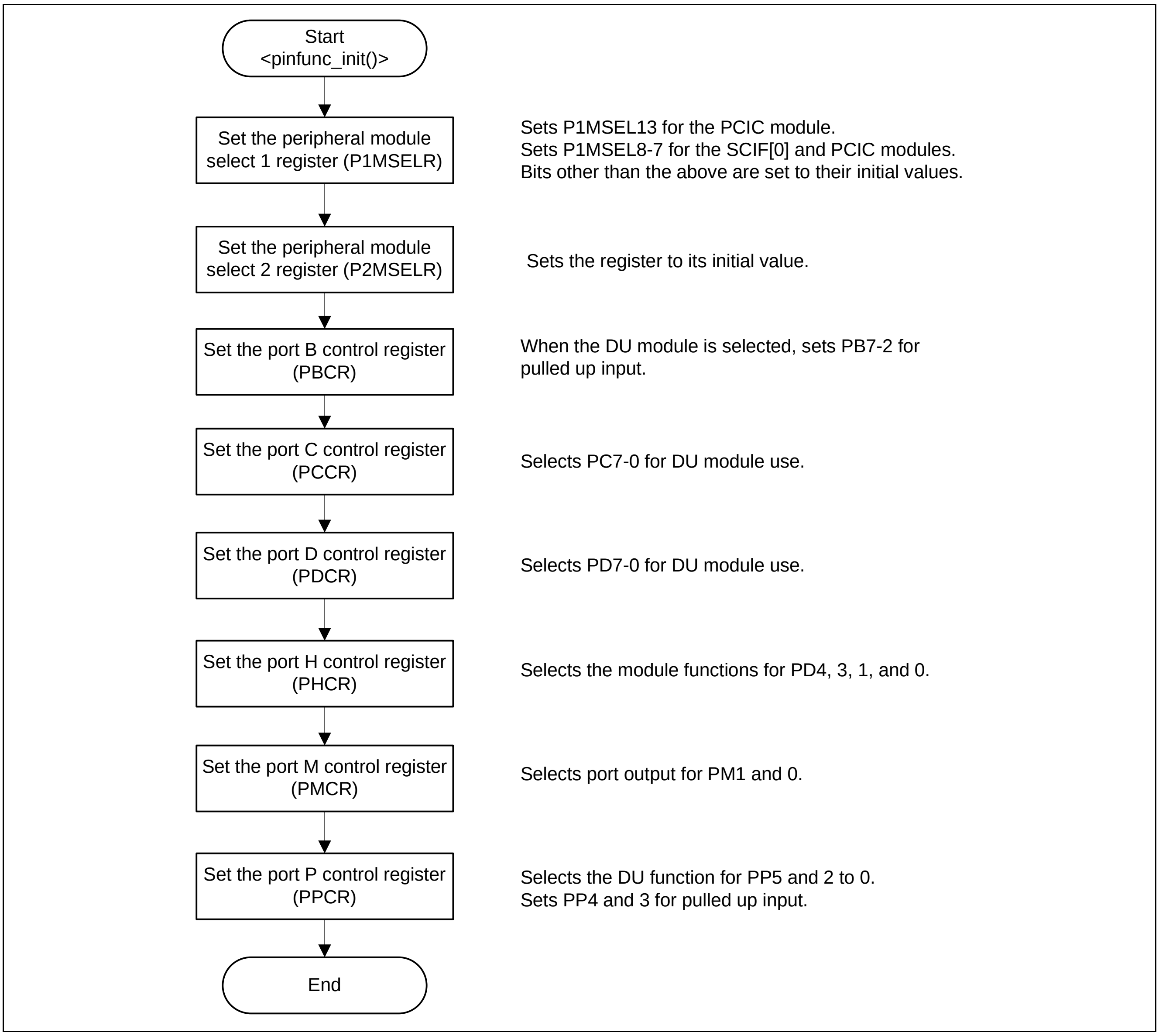

Start
<pinfunc_init()>
Set the peripheral module select 1 register (P1MSELR)
Sets P1MSEL13 for the PCIC module.
Sets P1MSEL8-7 for the SCIF[0] and PCIC modules.
Bits other than the above are set to their initial values.
Set the peripheral module select 2 register (P2MSELR)
Sets the register to its initial value.
Set the port B control register (PBCR)
When the DU module is selected, sets PB7-2 for pulled up input.
Set the port C control register (PCCR)
Selects PC7-0 for DU module use.
Set the port D control register (PDCR)
Selects PD7-0 for DU module use.
Set the port H control register (PHCR)
Selects the module functions for PD4, 3, 1, and 0.
Set the port M control register (PMCR)
Selects port output for PM1 and 0.
Set the port P control register (PPCR)
Selects the DU function for PP5 and 2 to 0.
Sets PP4 and 3 for pulled up input.
End

Figure 12 Pin Function Setting Flowchart

3.3.4 SCIF Initialization

Figure 13 shows the flowchart for SCIF initialization.

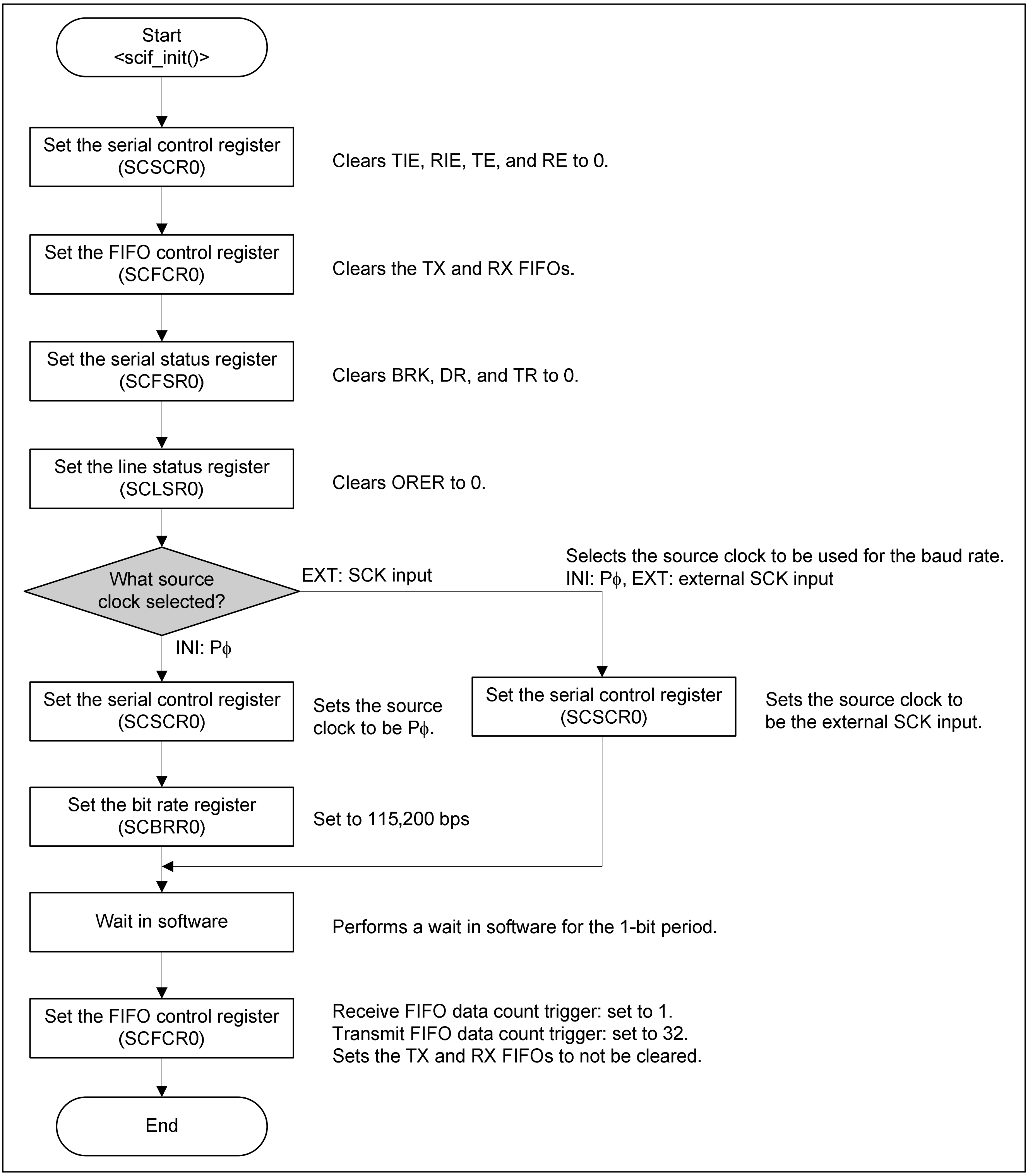

Start
<scif_init()>
Set the serial control register (SCSCR0)
Clears TIE, RIE, TE, and RE to 0.
Set the FIFO control register (SCFCR0)
Clears the TX and RX FIFOs.
Set the serial status register (SCFSR0)
Clears BRK, DR, and TR to 0.
Set the line status register (SCLSR0)
Clears ORER to 0.
What source clock selected?
EXT: SCK input
Selects the source clock to be used for the baud rate.
INI: Pϕ, EXT: external SCK input
INI: Pϕ
Set the serial control register (SCSCR0)
Sets the source clock to be Pϕ.
Set the serial control register (SCSCR0)
Sets the source clock to be the external SCK input.
Set the bit rate register (SCBRR0)
Set to 115,200 bps
Wait in software
Performs a wait in software for the 1-bit period.
Set the FIFO control register (SCFCR0)
Receive FIFO data count trigger: set to 1.
Transmit FIFO data count trigger: set to 32.
Sets the TX and RX FIFOs to not be cleared.
End

Figure 13 SCIF Initialization Flowchart

3.3.5 DU Initialization

Figures 14 to 16 show the flowchart for DU initialization.

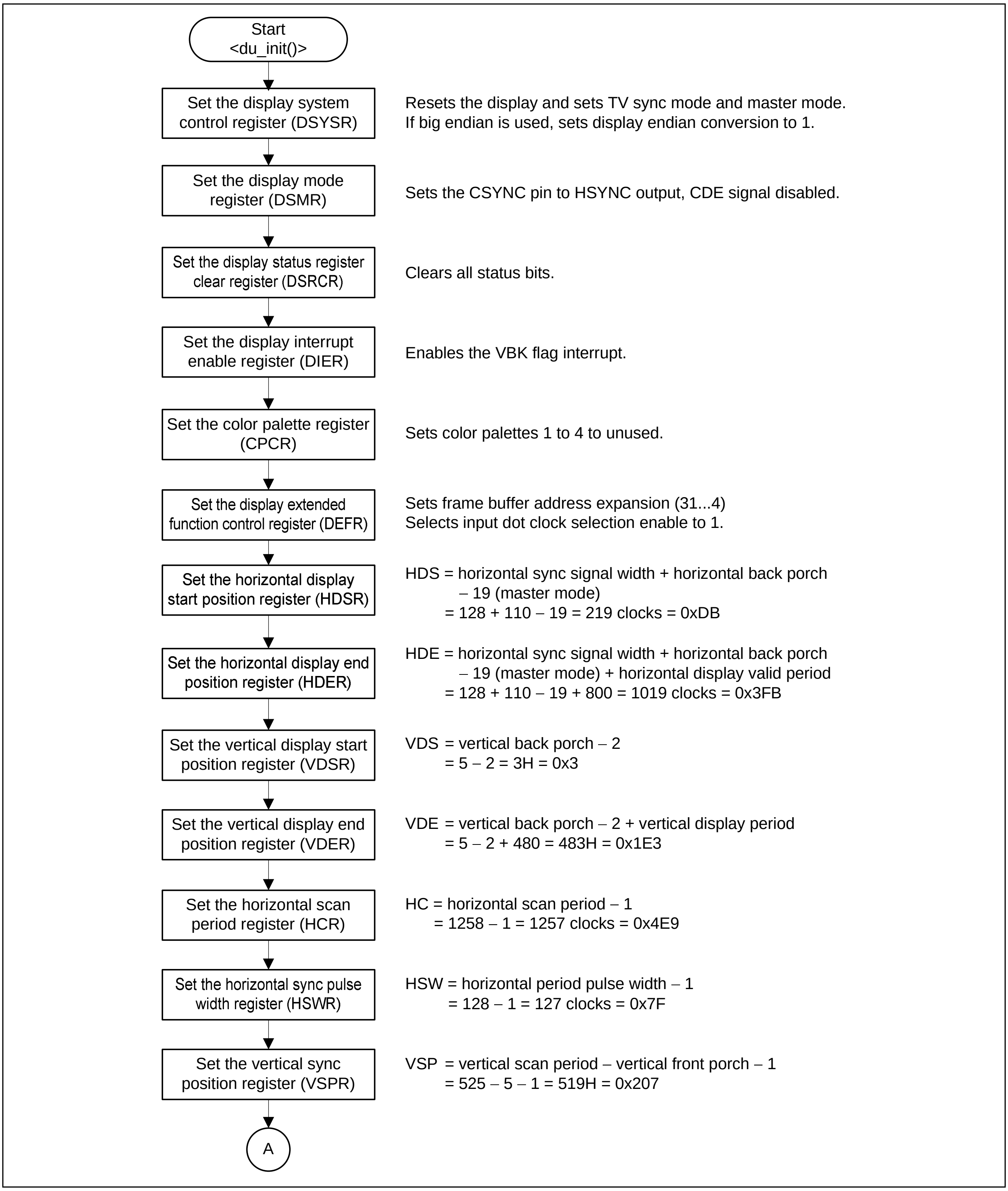
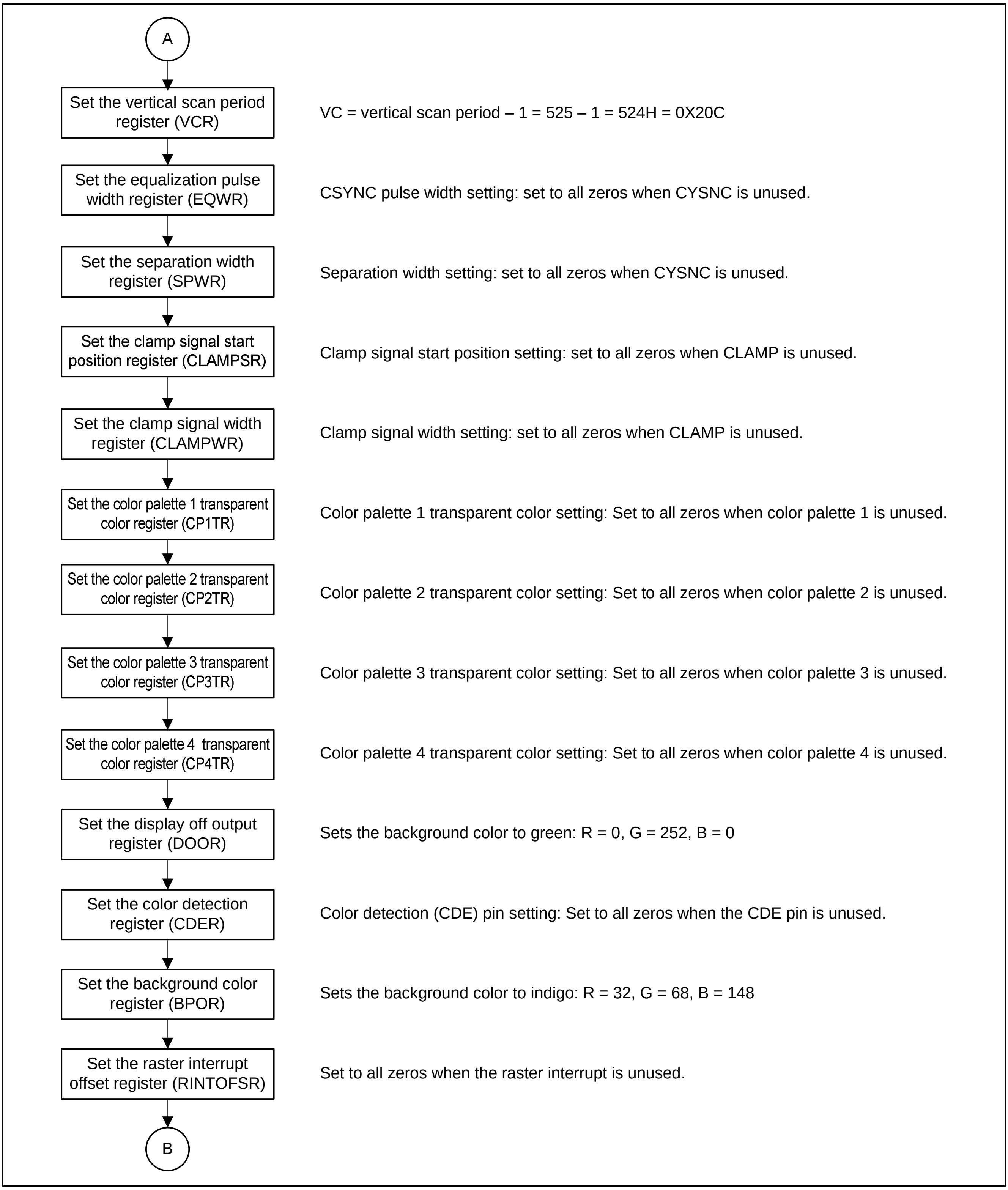

Start
<du_init()>
Set the display system control register (DSYSR)
Resets the display and sets TV sync mode and master mode.
If big endian is used, sets display endian conversion to 1.
Set the display mode register (DSMR)
Sets the CSYNC pin to HSYNC output, CDE signal disabled.
Set the display status register clear register (DSRCR)
Clears all status bits.
Set the display interrupt enable register (DIER)
Enables the VBK flag interrupt.
Set the color palette register (CPCR)
Sets color palettes 1 to 4 to unused.
Set the display extended function control register (DEFR)
Sets frame buffer address expansion (31...4)
Selects input dot clock selection enable to 1.
Set the horizontal display start position register (HDSR)
HDS = horizontal sync signal width + horizontal back porch
– 19 (master mode)
= 128 + 110 – 19 = 219 clocks = 0xDB
Set the horizontal display end position register (HDER)
HDE = horizontal sync signal width + horizontal back porch
– 19 (master mode) + horizontal display valid period
= 128 + 110 – 19 + 800 = 1019 clocks = 0x3FB
Set the vertical display start position register (VDSR)
VDS = vertical back porch – 2
= 5 – 2 = 3H = 0x3
Set the vertical display end position register (VDER)
VDE = vertical back porch – 2 + vertical display period
= 5 – 2 + 480 = 483H = 0x1E3
Set the horizontal scan period register (HCR)
HC = horizontal scan period – 1
= 1258 – 1 = 1257 clocks = 0x4E9
Set the horizontal sync pulse width register (HSWR)
HSW = horizontal period pulse width – 1
= 128 – 1 = 127 clocks = 0x7F
Set the vertical sync position register (VSPR)
VSP = vertical scan period – vertical front porch – 1
= 525 – 5 – 1 = 519H = 0x207
A

Figure 14 DU Initialization Flowchart 1


A
Set the vertical scan period register (VCR)
Set the equalization pulse width register (EQWR)
Set the separation width register (SPWR)
Set the clamp signal start position register (CLAMPSR)
Set the clamp signal width register (CLAMPWR)
Set the color palette 1 transparent color register (CP1TR)
Set the color palette 2 transparent color register (CP2TR)
Set the color palette 3 transparent color register (CP3TR)
Set the color palette 4 transparent color register (CP4TR)
Set the display off output register (DOOR)
Set the color detection register (CDER)
Set the background color register (BPOR)
Set the raster interrupt offset register (RINTOFSR)
B

VC = vertical scan period – 1 = 525 – 1 = 524H = 0X20C

CSYNC pulse width setting: set to all zeros when CYSNC is unused.

Separation width setting: set to all zeros when CYSNC is unused.

Clamp signal start position setting: set to all zeros when CLAMP is unused.

Clamp signal width setting: set to all zeros when CLAMP is unused.

Color palette 1 transparent color setting: Set to all zeros when color palette 1 is unused.

Color palette 2 transparent color setting: Set to all zeros when color palette 2 is unused.

Color palette 3 transparent color setting: Set to all zeros when color palette 3 is unused.

Color palette 4 transparent color setting: Set to all zeros when color palette 4 is unused.

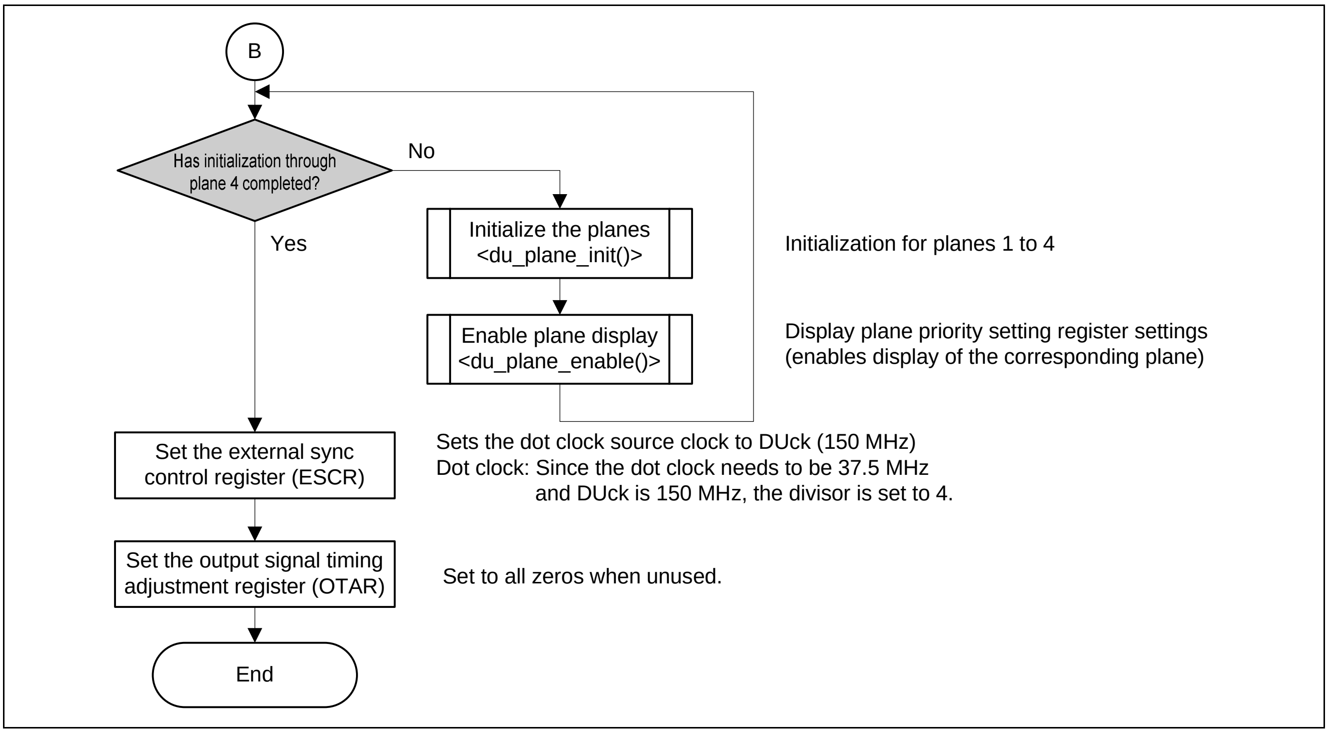
Sets the background color to green: R = 0, G = 252, B = 0

Color detection (CDE) pin setting: Set to all zeros when the CDE pin is unused.

Sets the background color to indigo: R = 32, G = 68, B = 148

Set to all zeros when the raster interrupt is unused.

Figure 15 DU Initialization Flowchart 2


B
Has initialization through plane 4 completed?
No
Yes
Initialize the planes <du_plane_init()>
Enable plane display <du_plane_enable()>
Set the external sync control register (ESCR)
Set the output signal timing adjustment register (OTAR)
End

Initialization for planes 1 to 4

Display plane priority setting register settings (enables display of the corresponding plane)

Sets the dot clock source clock to DUck (150 MHz)
Dot clock: Since the dot clock needs to be 37.5 MHz
and DUck is 150 MHz, the divisor is set to 4.

Set to all zeros when unused.

Figure 16 DU Initialization Flowchart 3

3.3.6 Plane Initialization

Figures 17 and 18 show the flowcharts for plane initialization.

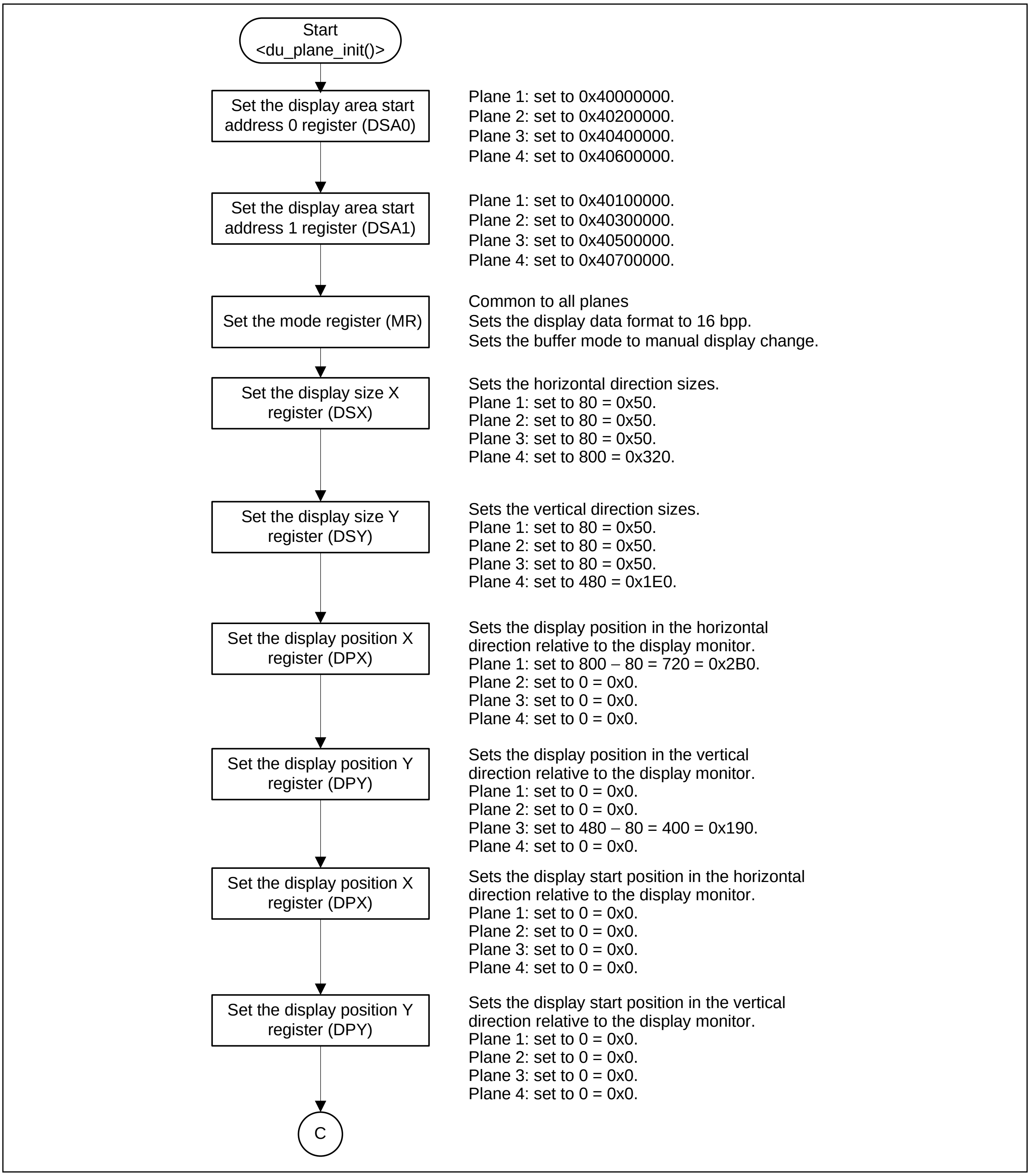
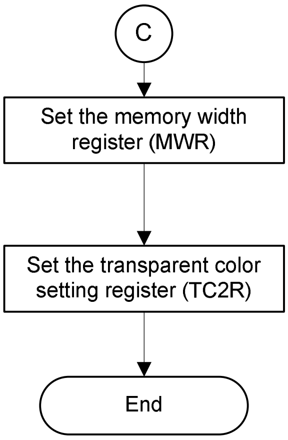

Start
<du_plane_init()>
Set the display area start address 0 register (DSA0)
Plane 1: set to 0x40000000.
Plane 2: set to 0x40200000.
Plane 3: set to 0x40400000.
Plane 4: set to 0x40600000.
Set the display area start address 1 register (DSA1)
Plane 1: set to 0x40100000.
Plane 2: set to 0x40300000.
Plane 3: set to 0x40500000.
Plane 4: set to 0x40700000.
Set the mode register (MR)
Common to all planes
Sets the display data format to 16 bpp.
Sets the buffer mode to manual display change.
Set the display size X register (DSX)
Sets the horizontal direction sizes.
Plane 1: set to 80 = 0x50.
Plane 2: set to 80 = 0x50.
Plane 3: set to 80 = 0x50.
Plane 4: set to 800 = 0x320.
Set the display size Y register (DSY)
Sets the vertical direction sizes.
Plane 1: set to 80 = 0x50.
Plane 2: set to 80 = 0x50.
Plane 3: set to 80 = 0x50.
Plane 4: set to 480 = 0x1E0.
Set the display position X register (DPX)
Sets the display position in the horizontal direction relative to the display monitor.
Plane 1: set to 800 – 80 = 720 = 0x2B0.
Plane 2: set to 0 = 0x0.
Plane 3: set to 0 = 0x0.
Plane 4: set to 0 = 0x0.
Set the display position Y register (DPY)
Sets the display position in the vertical direction relative to the display monitor.
Plane 1: set to 0 = 0x0.
Plane 2: set to 0 = 0x0.
Plane 3: set to 480 – 80 = 400 = 0x190.
Plane 4: set to 0 = 0x0.
Set the display position X register (DPX)
Sets the display start position in the horizontal direction relative to the display monitor.
Plane 1: set to 0 = 0x0.
Plane 2: set to 0 = 0x0.
Plane 3: set to 0 = 0x0.
Plane 4: set to 0 = 0x0.
Set the display position Y register (DPY)
Sets the display start position in the vertical direction relative to the display monitor.
Plane 1: set to 0 = 0x0.
Plane 2: set to 0 = 0x0.
Plane 3: set to 0 = 0x0.
Plane 4: set to 0 = 0x0.
C

Figure 17 Plane Initialization Flowchart 1


C
Set the memory width register (MWR)
Set the transparent color setting register (TC2R)
End

Sets the memory widths in the horizontal direction.
Plane 1: set to the image size: 80 = 0x50
Plane 2: set to the display size: 80 = 0x50
Plane 3: set to the display size: 80 = 0x50
Plane 4: set to the image size: 800 = 0x320

Common to all planes:
Sets the transparent color to green: R = 0, G = 63, B = 0.

Figure 17 Plane Initialization Flowchart 2

3.3.7 Turning Plane Display On

Figure 19 shows the flowchart for turning on display for each plane.

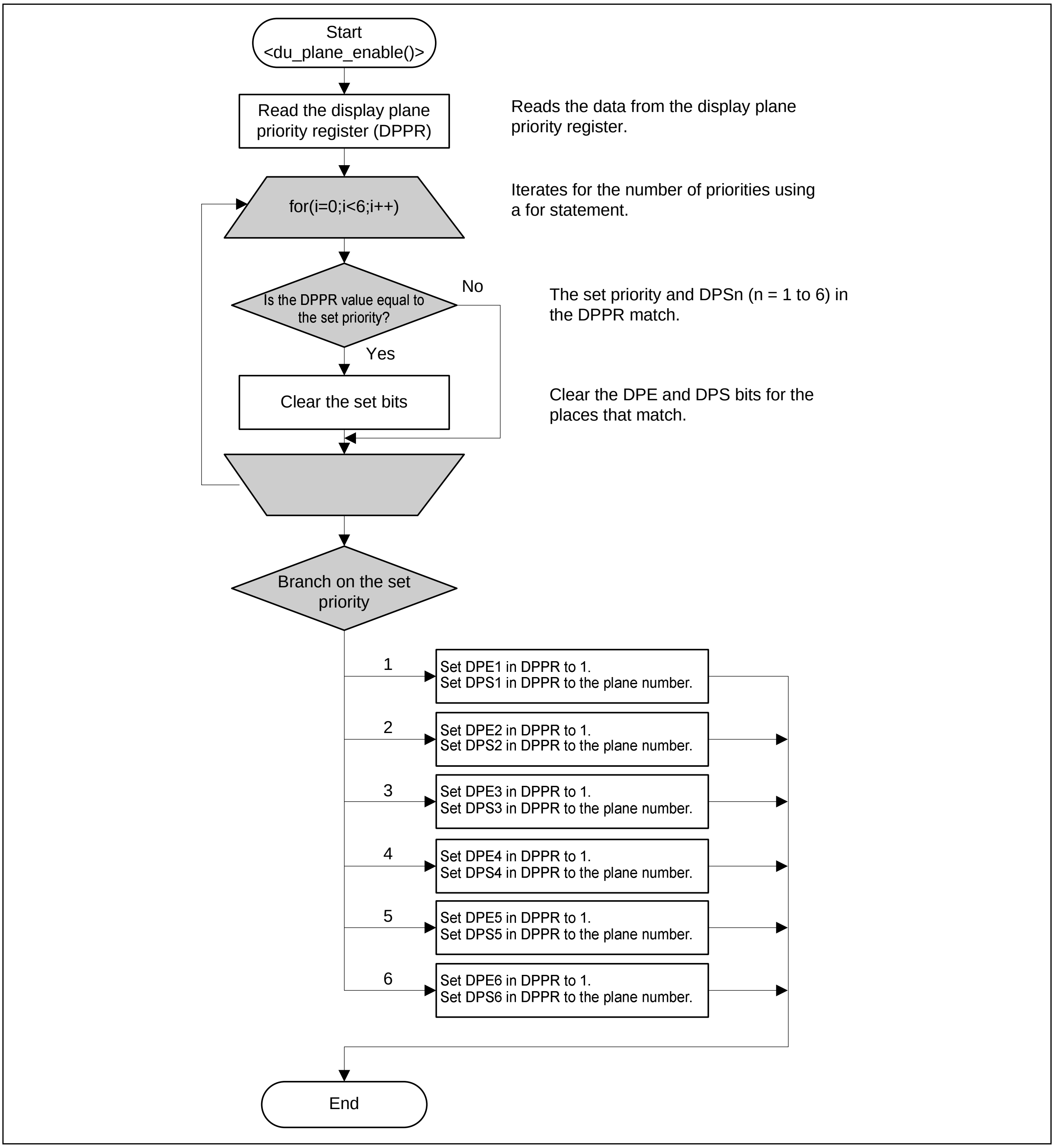

Start
<du_plane_enable()>
Read the display plane priority register (DPPR)
Reads the data from the display plane priority register.
for(i=0;i<6;i++)
Iterates for the number of priorities using a for statement.
Is the DPPR value equal to the set priority?
No
Yes
The set priority and DPSn (n = 1 to 6) in the DPPR match.
Clear the set bits
Clear the DPE and DPS bits for the places that match.
Branch on the set priority
1
Set DPE1 in DPPR to 1.
Set DPS1 in DPPR to the plane number.
2
Set DPE2 in DPPR to 1.
Set DPS2 in DPPR to the plane number.
3
Set DPE3 in DPPR to 1.
Set DPS3 in DPPR to the plane number.
4
Set DPE4 in DPPR to 1.
Set DPS4 in DPPR to the plane number.
5
Set DPE5 in DPPR to 1.
Set DPS5 in DPPR to the plane number.
6
Set DPE6 in DPPR to 1.
Set DPS6 in DPPR to the plane number.
End

Figure 19 Turning Plane Display On Flowchart

3.3.8 Turning Plane Display Off

Figure 20 shows the flowchart for turning off display for each plane.

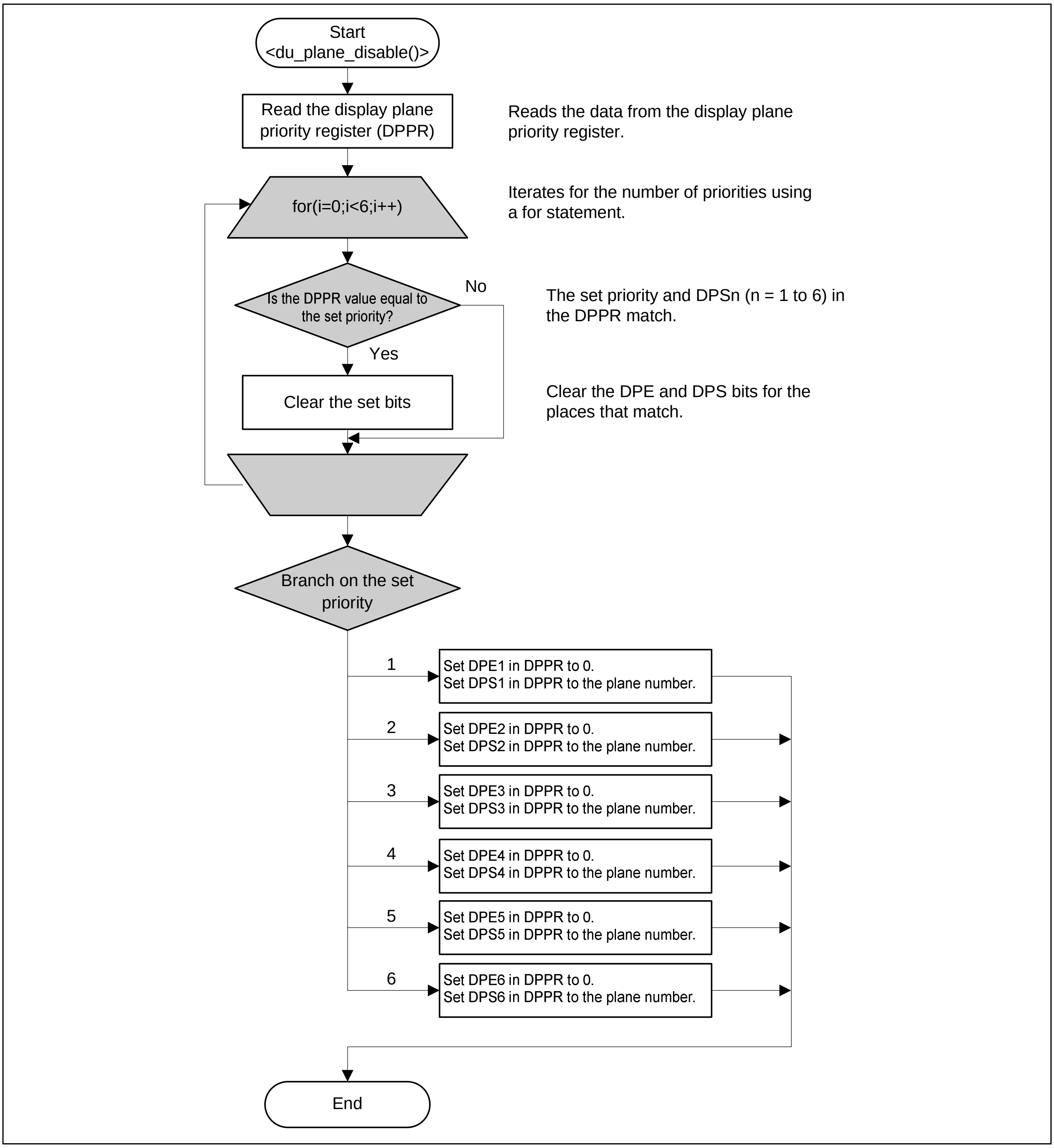
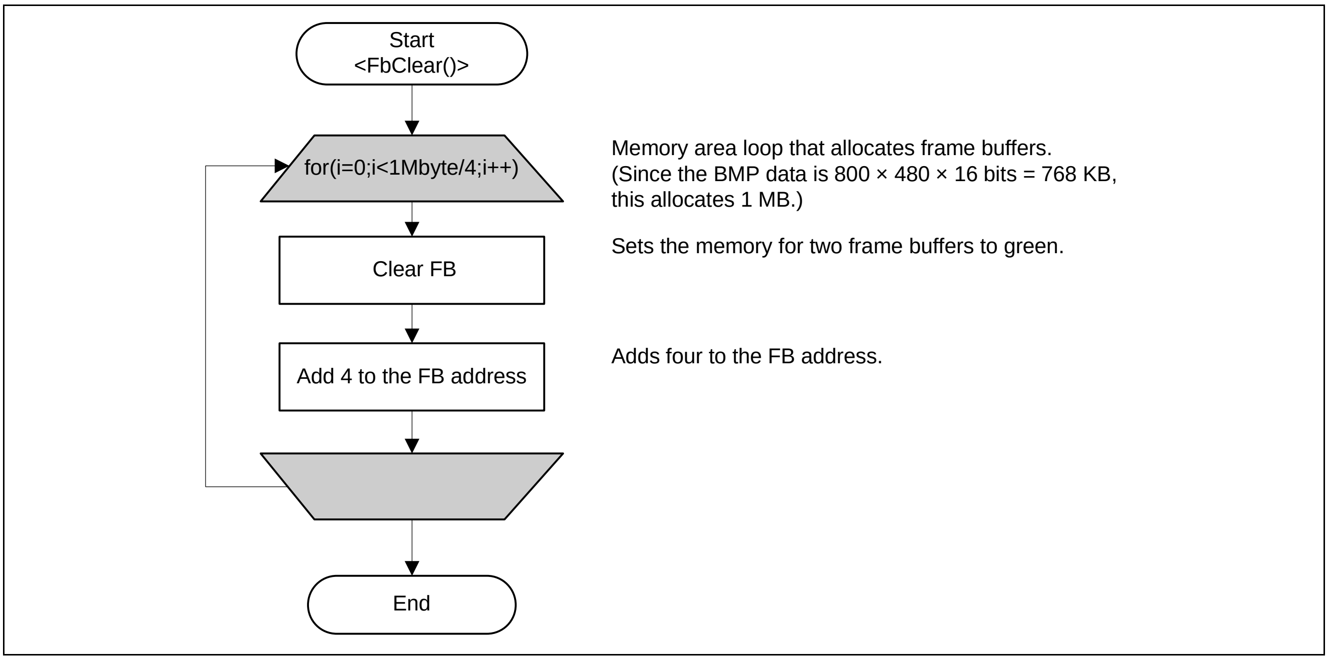

Start
<du_plane_disable()>
Read the display plane priority register (DPPR)
Reads the data from the display plane priority register.
for(i=0;i<6;i++)
Iterates for the number of priorities using a for statement.
Is the DPPR value equal to the set priority?
No
Yes
The set priority and DPSn (n = 1 to 6) in the DPPR match.
Clear the set bits
Clear the DPE and DPS bits for the places that match.
Branch on the set priority
1
Set DPE1 in DPPR to 0.
Set DPS1 in DPPR to the plane number.
2
Set DPE2 in DPPR to 0.
Set DPS2 in DPPR to the plane number.
3
Set DPE3 in DPPR to 0.
Set DPS3 in DPPR to the plane number.
4
Set DPE4 in DPPR to 0.
Set DPS4 in DPPR to the plane number.
5
Set DPE5 in DPPR to 0.
Set DPS5 in DPPR to the plane number.
6
Set DPE6 in DPPR to 0.
Set DPS6 in DPPR to the plane number.
End

Figure 20 Turning Plane Display Off Flowchart

3.3.9 Frame Buffer Clear

Figure 21 shows the flowchart for clearing the frame buffers for planes 1 to 3.



Memory area loop that allocates frame buffers. (Since the BMP data is 800 × 480 × 16 bits = 768 KB, this allocates 1 MB.)

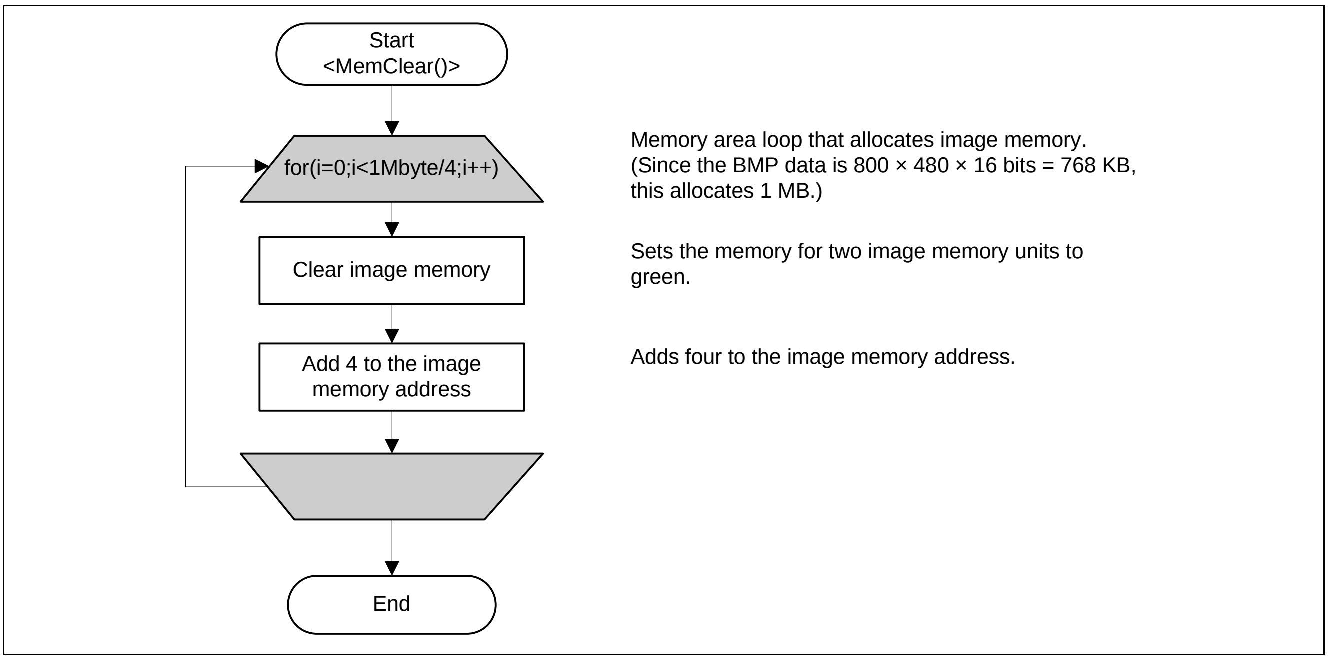
Sets the memory for two frame buffers to green.

Adds four to the FB address.

Figure 21 Frame Buffer Clear Flowchart

3.3.10 Memory Clear

Figure 22 shows the flowchart for clearing the image memory used for the four planes.



Memory area loop that allocates image memory. (Since the BMP data is 800 × 480 × 16 bits = 768 KB, this allocates 1 MB.)

Sets the memory for two image memory units to green.

Adds four to the image memory address.

Figure 22 Memory Clear Flowchart

3.3.11 Frame Buffer Write

Figure 23 shows the flowchart for writing BMP files to the frame buffers for planes 1 to 3.

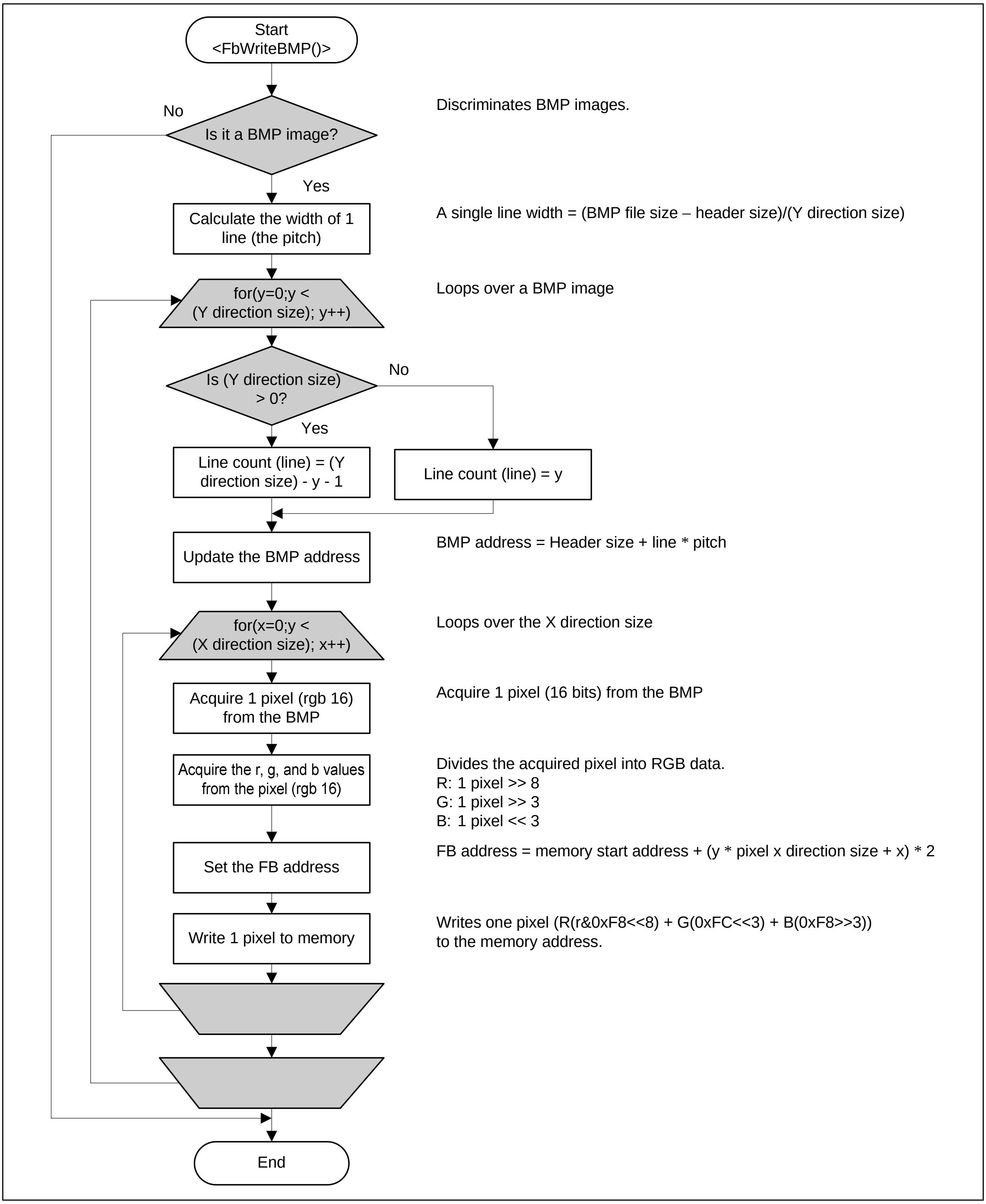

Start
<FbWriteBMP()>
No
Is it a BMP image?
Yes
Discriminates BMP images.
Calculate the width of 1 line (the pitch)
A single line width = (BMP file size – header size)/(Y direction size)
for(y=0;y < (Y direction size); y++)
Loops over a BMP image
Is (Y direction size) > 0?
No
Yes
Line count (line) = (Y direction size) - y - 1
Line count (line) = y
Update the BMP address
BMP address = Header size + line * pitch
for(x=0;y < (X direction size); x++)
Loops over the X direction size
Acquire 1 pixel (rgb 16) from the BMP
Acquire 1 pixel (16 bits) from the BMP
Acquire the r, g, and b values from the pixel (rgb 16)
Divides the acquired pixel into RGB data.
R: 1 pixel >> 8
G: 1 pixel >> 3
B: 1 pixel << 3
Set the FB address
FB address = memory start address + (y * pixel x direction size + x) * 2
Write 1 pixel to memory
Writes one pixel (R(r&0xF8<<8) + G(0xFC<<3) + B(0xF8>>3)) to the memory address.
End

Figure 23 Frame Buffer BMP Write Flowchart

3.3.12 Memory Write

Figure 24 shows the flowchart for writing BMP images to the memory used by plane 4.

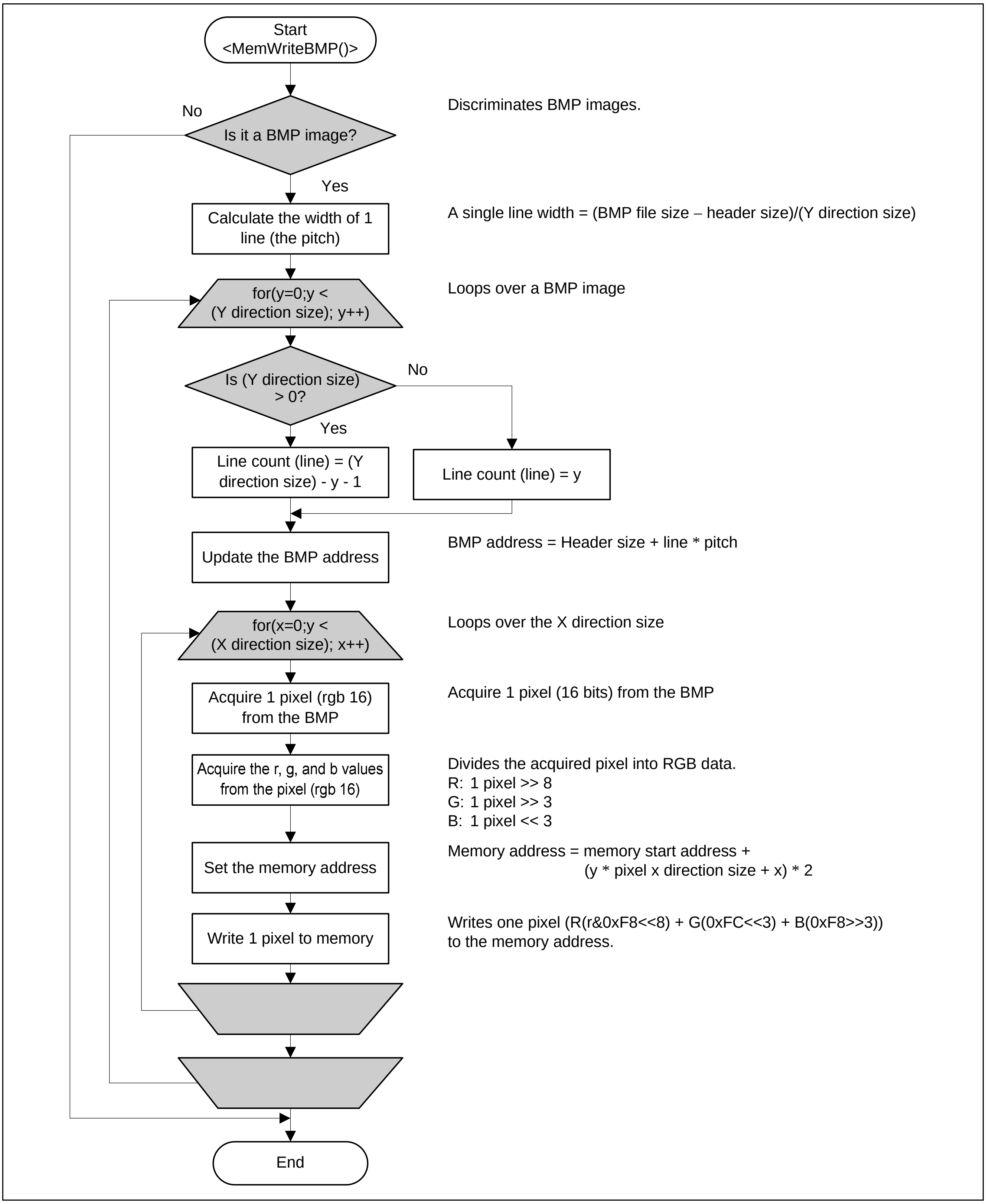

Start
<MemWriteBMP()>
Is it a BMP image?
No
Yes
Discriminates BMP images.
Calculate the width of 1 line (the pitch)
A single line width = (BMP file size – header size)/(Y direction size)
for(y=0;y <
(Y direction size); y++)
Loops over a BMP image
Is (Y direction size) > 0?
No
Yes
Line count (line) = (Y direction size) - y - 1
Line count (line) = y
Update the BMP address
BMP address = Header size + line * pitch
for(x=0;y <
(X direction size); x++)
Loops over the X direction size
Acquire 1 pixel (rgb 16) from the BMP
Acquire 1 pixel (16 bits) from the BMP
Acquire the r, g, and b values from the pixel (rgb 16)
Divides the acquired pixel into RGB data.
R: 1 pixel >> 8
G: 1 pixel >> 3
B: 1 pixel << 3
Set the memory address
Memory address = memory start address +
(y * pixel x direction size + x) * 2
Write 1 pixel to memory
Writes one pixel (R(r&0xF8<<8) + G(0xFC<<3) + B(0xF8>>3))
to the memory address.
End

Figure 24 Write BMP Image to Memory Flowchart

3.3.13 DU Display On/Off

Figure 25 shows the flowchart for turning DU display on or off.

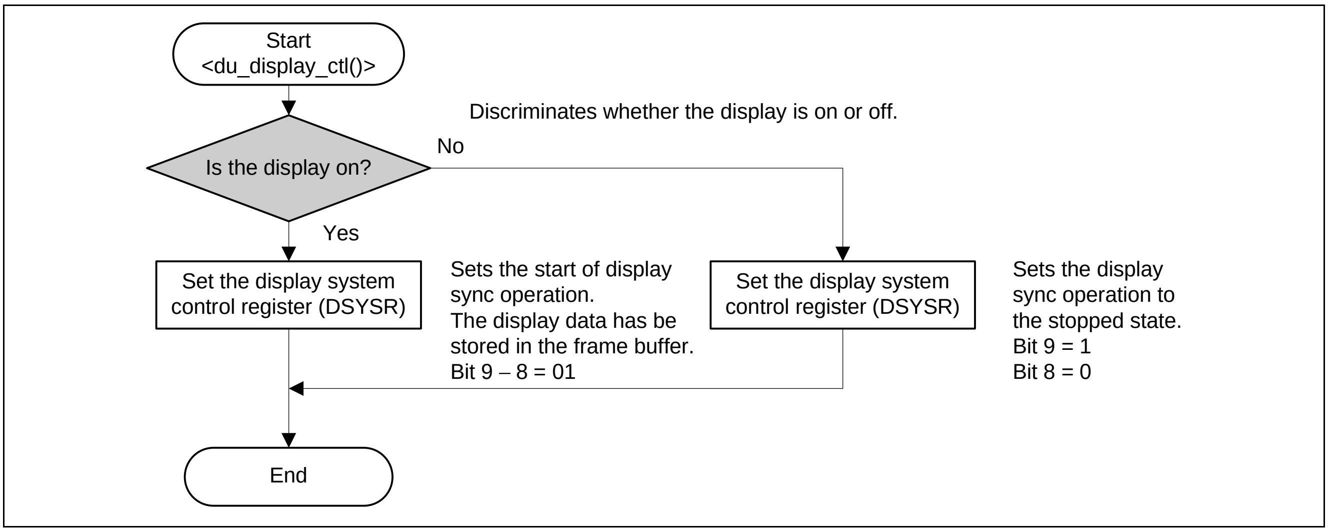
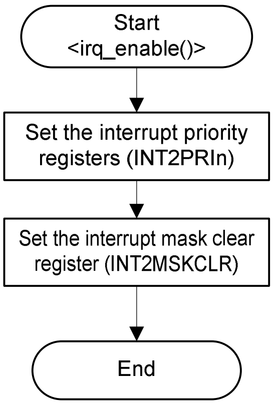

Start
<du_display_ctl()>
Discriminates whether the display is on or off.
Is the display on?
No
Yes
Set the display system control register (DSYSR)
Sets the start of display sync operation.
The display data has be stored in the frame buffer.
Bit 9 – 8 = 01
Set the display system control register (DSYSR)
Sets the display sync operation to the stopped state.
Bit 9 = 1
Bit 8 = 0
End

Figure 25 DU Display On/Off Control Flowchart

3.3.14 Enabling Interrupts

Figure 26 shows the flowchart for enabling peripheral module interrupts.


Start
<irq_enable()>
Set the interrupt priority registers (INT2PRIn)
Set the interrupt mask clear register (INT2MSKCLR)
End

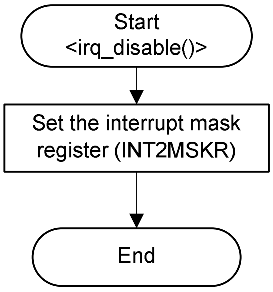
Sets the interrupt priorities for the internal peripheral modules used.
n: Values in the range 0 to 9 depending on the modules used.

Clears the interrupt masks for the peripheral modules used.

Figure 26 Peripheral Module Interrupt Enabling Flowchart

3.3.15 Disabling Interrupts

Figure 27 shows the flowchart for disabling peripheral module interrupts.



Masks the internal peripheral module interrupts used.

Figure 27 Peripheral Module Interrupt Disabling Flowchart

3.3.16 DU Interrupts

Figure 28 to 33 show the flowcharts for using the DU interrupts.

The sample program uses only the VBK interrupt. Although interrupt handlers for interrupts other than the VBK interrupt are provided, they are not used.

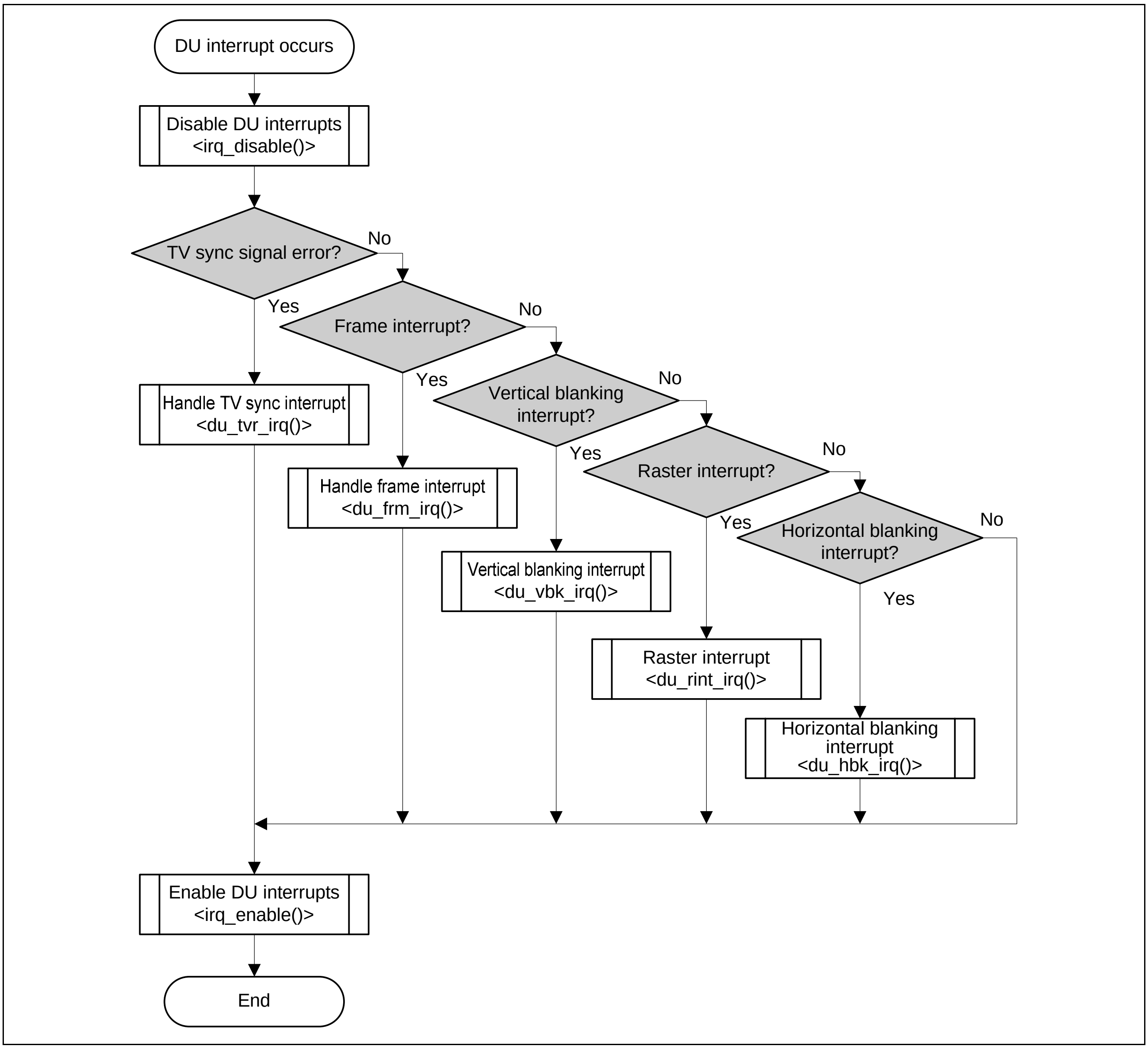

DU interrupt occurs
Disable DU interrupts <irq_disable()>
TV sync signal error?
No
Yes
Frame interrupt?
No
Yes
Vertical blanking interrupt?
No
Yes
Raster interrupt?
No
Yes
Horizontal blanking interrupt?
No
Yes
Handle TV sync interrupt <du_tvr_irq()>
Handle frame interrupt <du_frm_irq()>
Vertical blanking interrupt <du_vbk_irq()>
Raster interrupt <du_rint_irq()>
Horizontal blanking interrupt <du_hbk_irq()>
Enable DU interrupts <irq_enable()>
End

Figure 28 DU Interrupt Flowchart

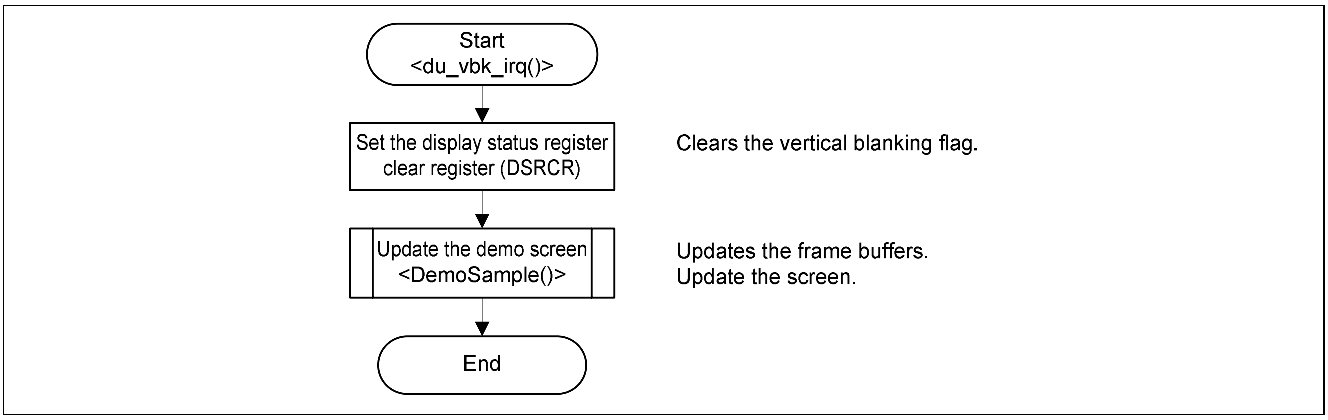

Start
<du_vbk_irq()>
Set the display status register clear register (DSRCR)
Clears the vertical blanking flag.
Update the demo screen <DemoSample()>
Updates the frame buffers.
Update the screen.
End

Figure 29 Vertical Blanking Flag Interrupt Flowchart

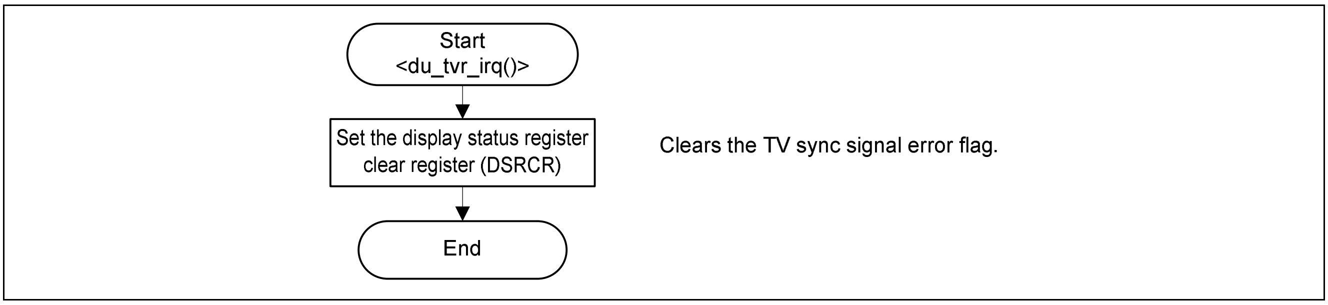

Start
<du_tvr_irq()>
Set the display status register clear register (DSRCR)
Clears the TV sync signal error flag.
End

Figure 30 TV Sync Signal Error Flag Interrupt Flowchart

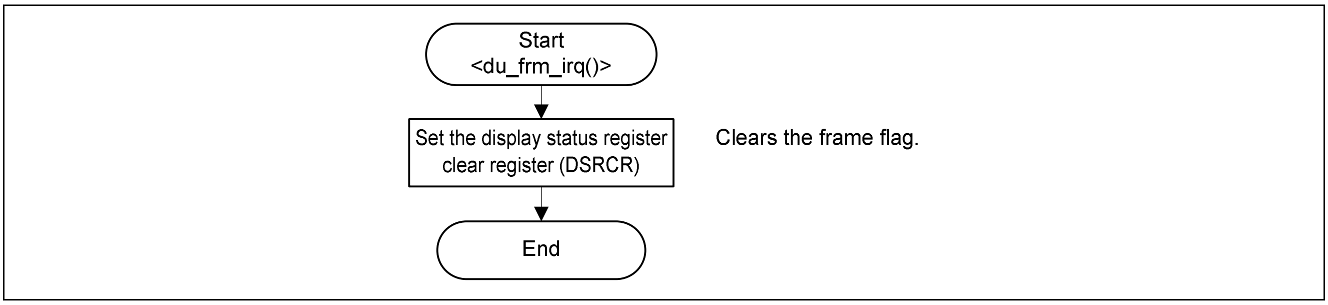


Figure 31 Frame Flag Interrupt Flowchart

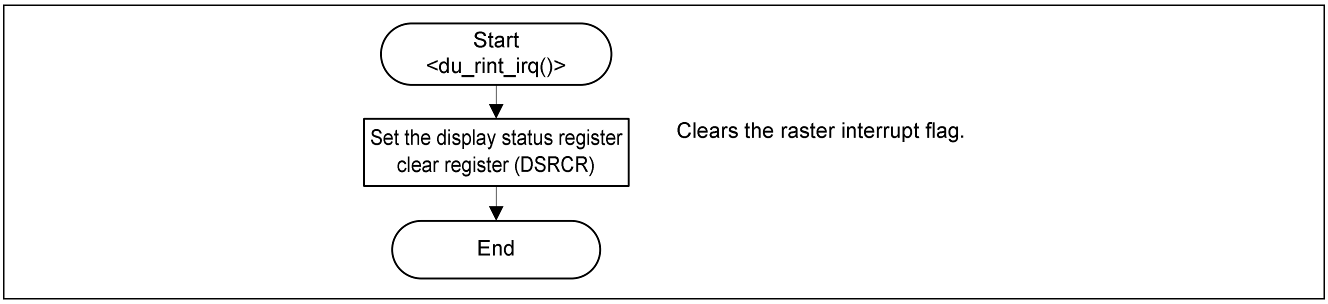

Start
<du_rint_irq()>
Set the display status register clear register (DSRCR)
End
Clears the raster interrupt flag.

Figure 32 Raster Flag Interrupt Flowchart

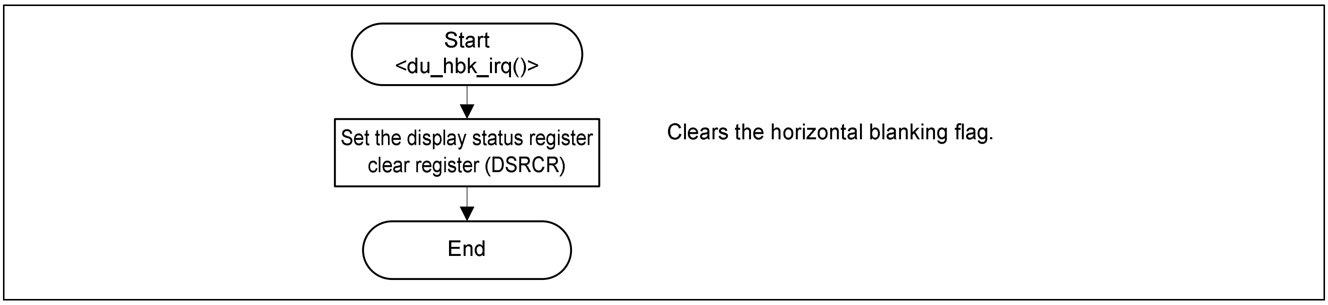

Start
<du_hbk_irq()>
Set the display status register clear register (DSRCR)
End
Clears the horizontal blanking flag.

Figure 33 Horizontal Blanking Flag Interrupt Flowchart

3.3.17 The DemoSample() Function

Figures 34 to 37 show the flowcharts for the DemoSample() function.

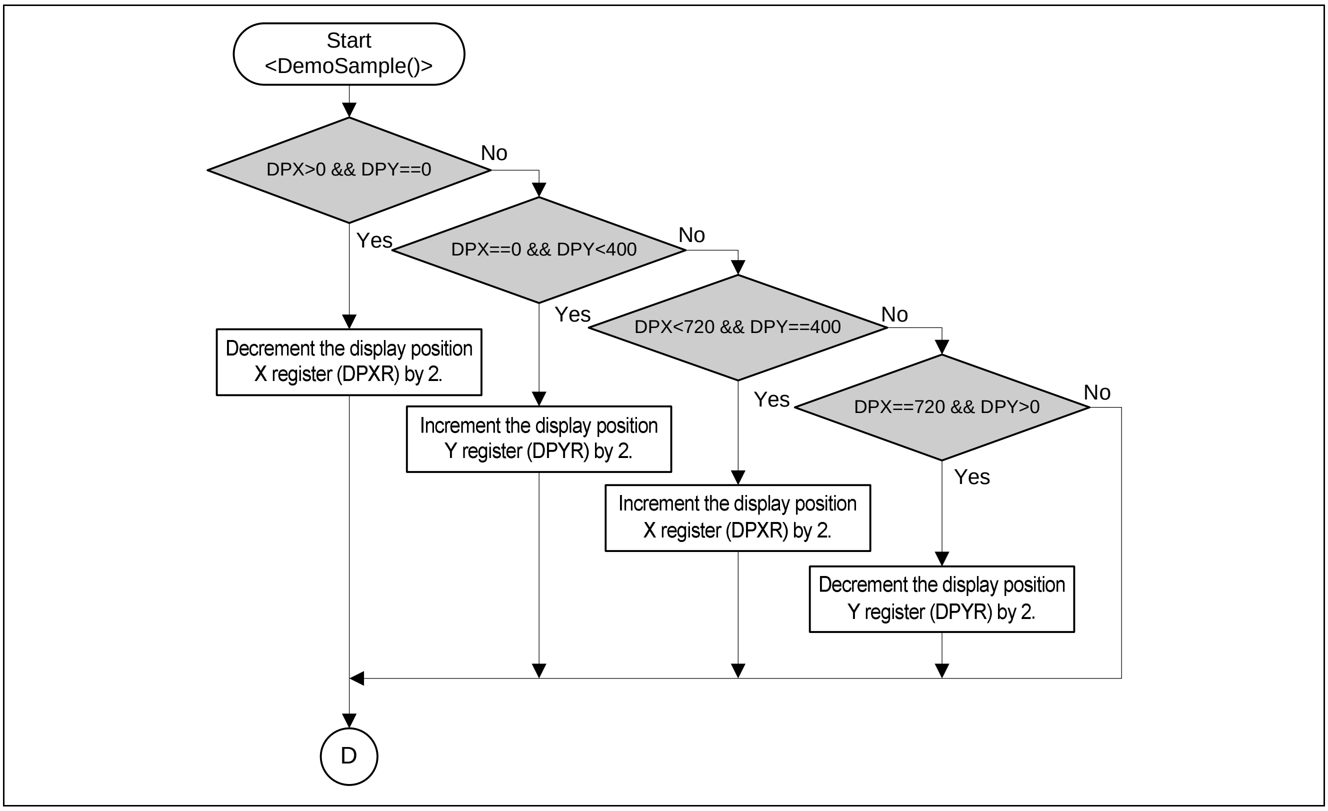

Start
<DemoSample()>
DPX>0 && DPY==0
No
Yes
DPX==0 && DPY<400
No
Yes
DPX<720 && DPY==400
No
Yes
DPX==720 && DPY>0
No
Yes
Decrement the display position X register (DPXR) by 2.
Increment the display position Y register (DPYR) by 2.
Increment the display position X register (DPXR) by 2.
Decrement the display position Y register (DPYR) by 2.
D

Figure 34 DemoSample() Plane 1 Flowchart

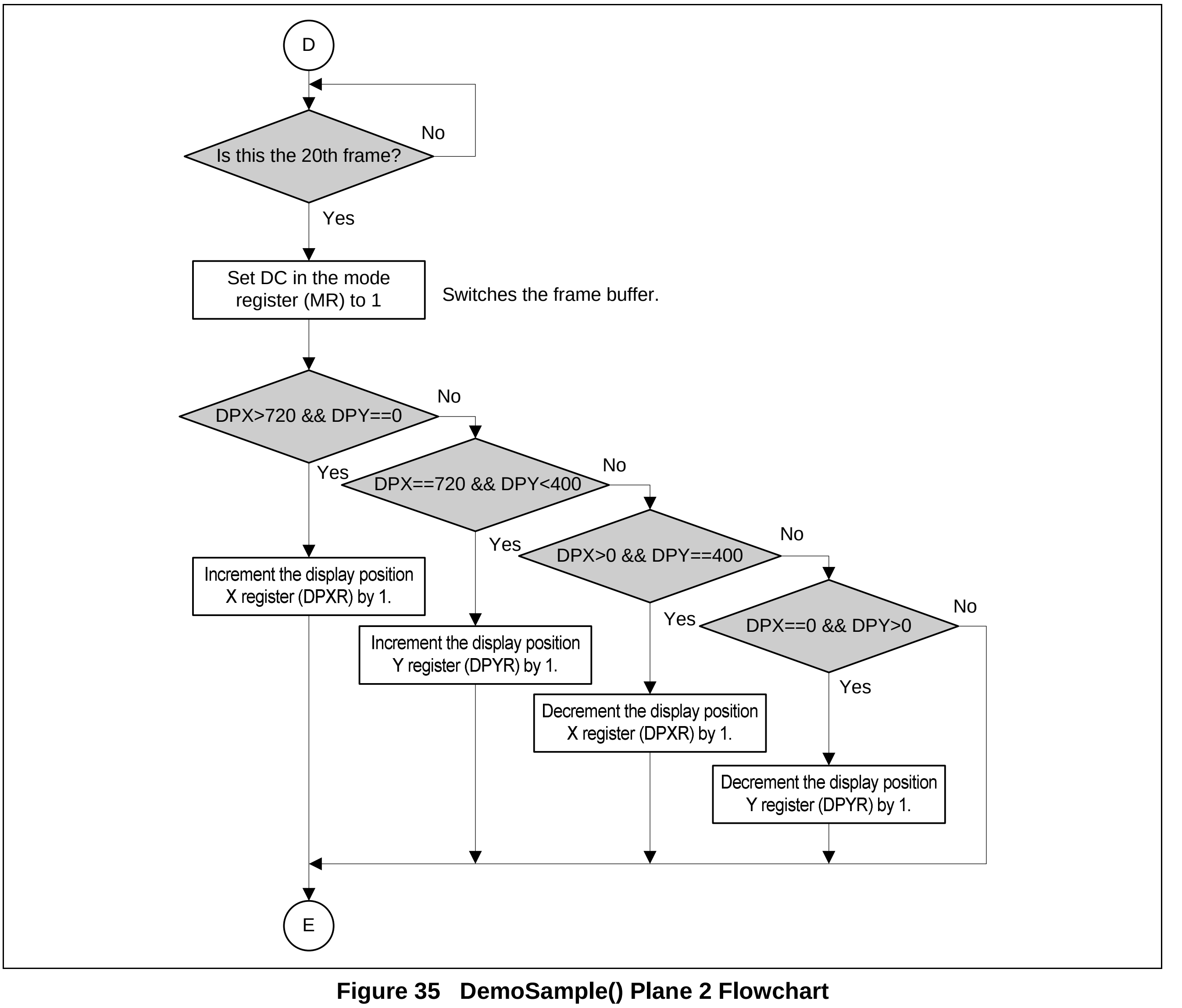

D
Is this the 20th frame?
No
Yes
Set DC in the mode register (MR) to 1
Switches the frame buffer.
DPX>720 && DPY==0
No
Yes
DPX==720 && DPY<400
No
Yes
DPX>0 && DPY==400
No
Yes
DPX==0 && DPY>0
No
Yes
Increment the display position X register (DPXR) by 1.
Increment the display position Y register (DPYR) by 1.
Decrement the display position X register (DPXR) by 1.
Decrement the display position Y register (DPYR) by 1.
E

Figure 35 DemoSample() Plane 2 Flowchart

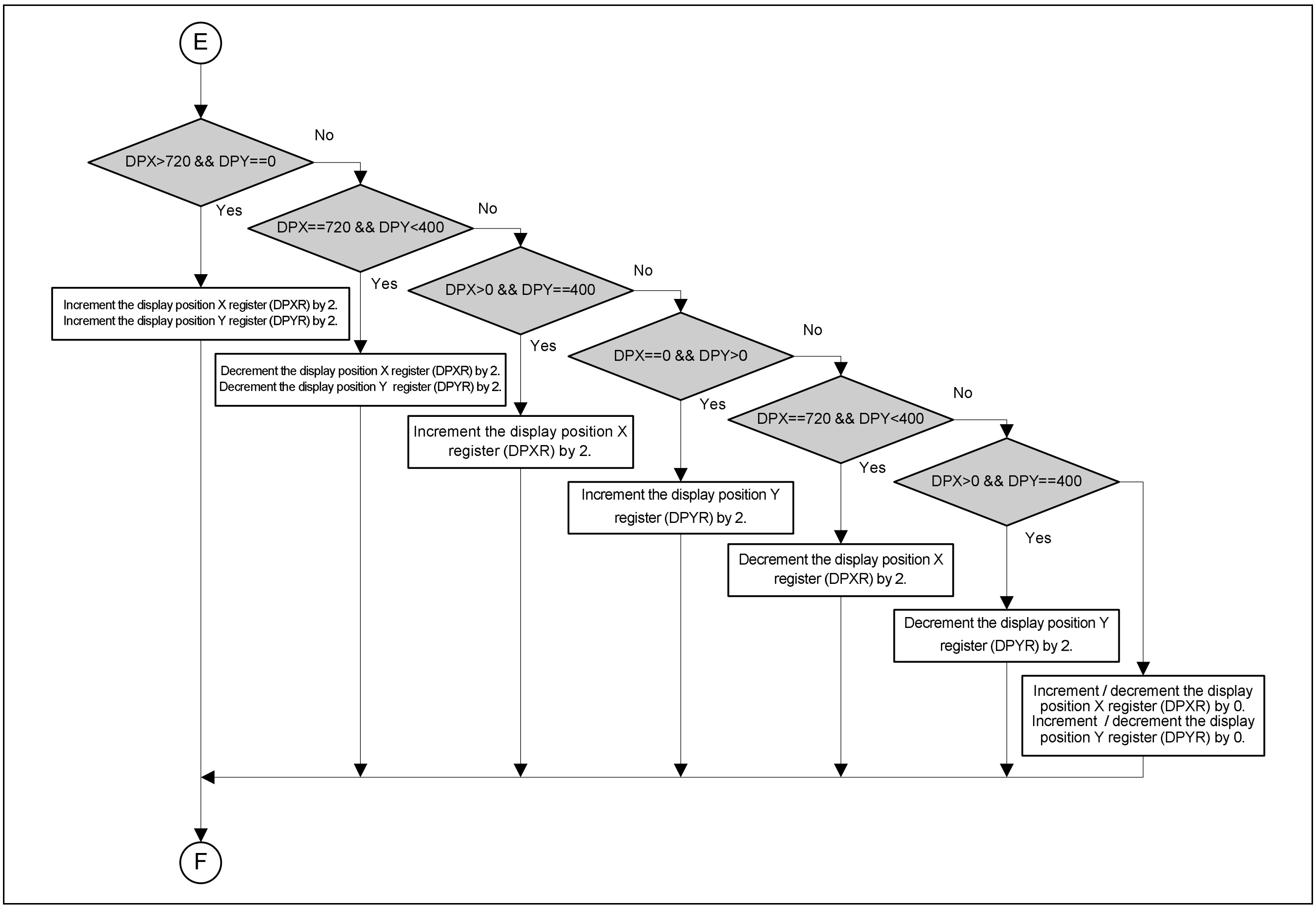

E
DPX>720 && DPY==0
No
Yes
Increment the display position X register (DPXR) by 2.
Increment the display position Y register (DPYR) by 2.
DPX==720 && DPY<400
No
Yes
Decrement the display position X register (DPXR) by 2.
Decrement the display position Y register (DPYR) by 2.
DPX>0 && DPY==400
No
Yes
Increment the display position X register (DPXR) by 2.
DPX==0 && DPY>0
No
Yes
Increment the display position Y register (DPYR) by 2.
DPX==720 && DPY<400
No
Yes
Decrement the display position X register (DPXR) by 2.
DPX>0 && DPY==400
Yes
Decrement the display position Y register (DPYR) by 2.
Increment / decrement the display position X register (DPXR) by 0.
Increment / decrement the display position Y register (DPYR) by 0.
F

Figure 36 DemoSample() Plane 3 Flowchart

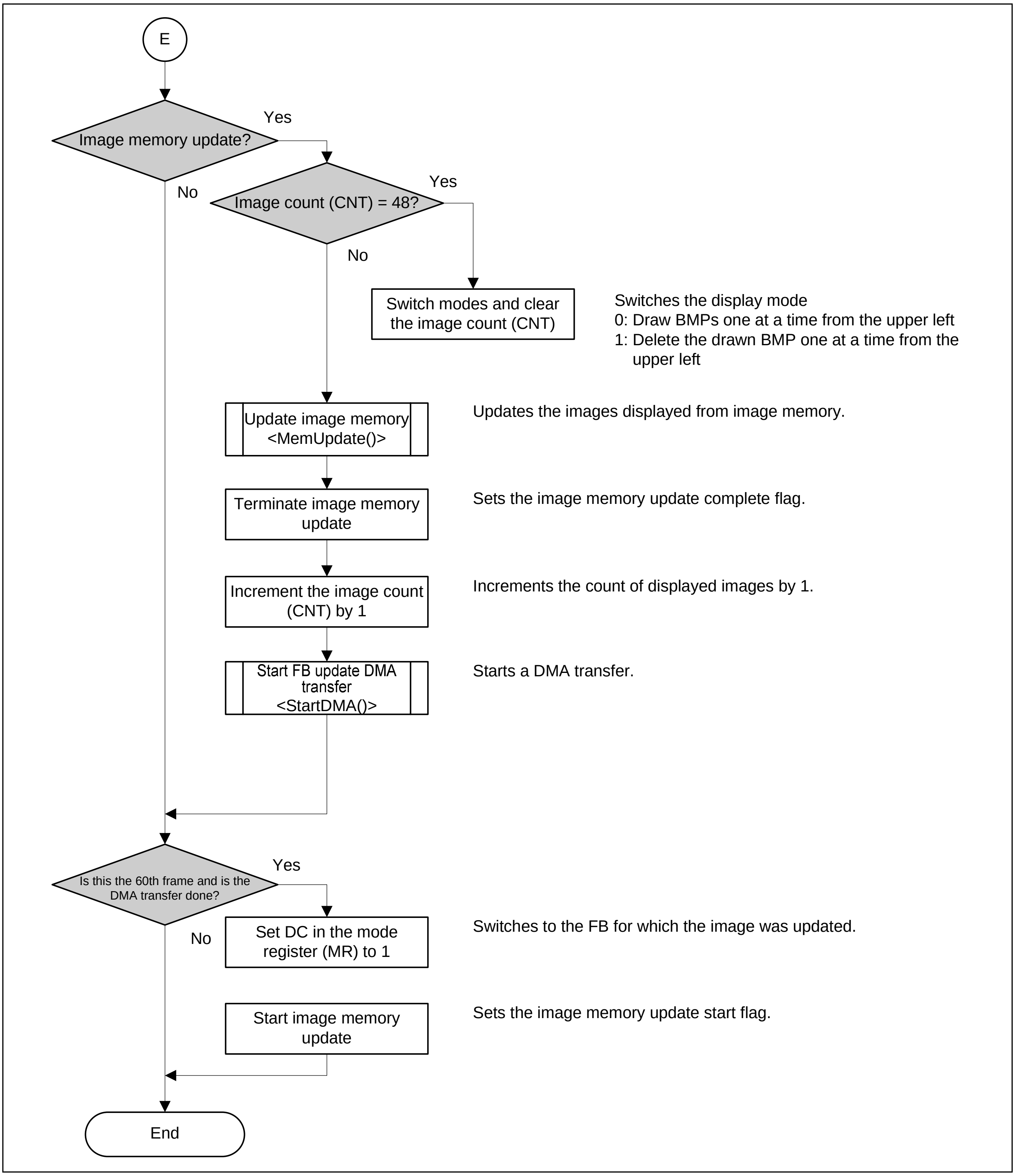

E
Image memory update?
Yes
No
Image count (CNT) = 48?
Yes
No
Switch modes and clear the image count (CNT)
Switches the display mode
0: Draw BMPs one at a time from the upper left
1: Delete the drawn BMP one at a time from the upper left
Update image memory <MemUpdate()>
Updates the images displayed from image memory.
Terminate image memory update
Sets the image memory update complete flag.
Increment the image count (CNT) by 1
Increments the count of displayed images by 1.
Start FB update DMA transfer <StartDMA()>
Starts a DMA transfer.
Is this the 60th frame and is the DMA transfer done?
Yes
No
Set DC in the mode register (MR) to 1
Switches to the FB for which the image was updated.
Start image memory update
Sets the image memory update start flag.
End

Figure 37 DemoSample() Plane 4 Flowchart

3.3.18 Image Memory Update

Figure 38 shows the flowchart for update the image memory used by plane 4.

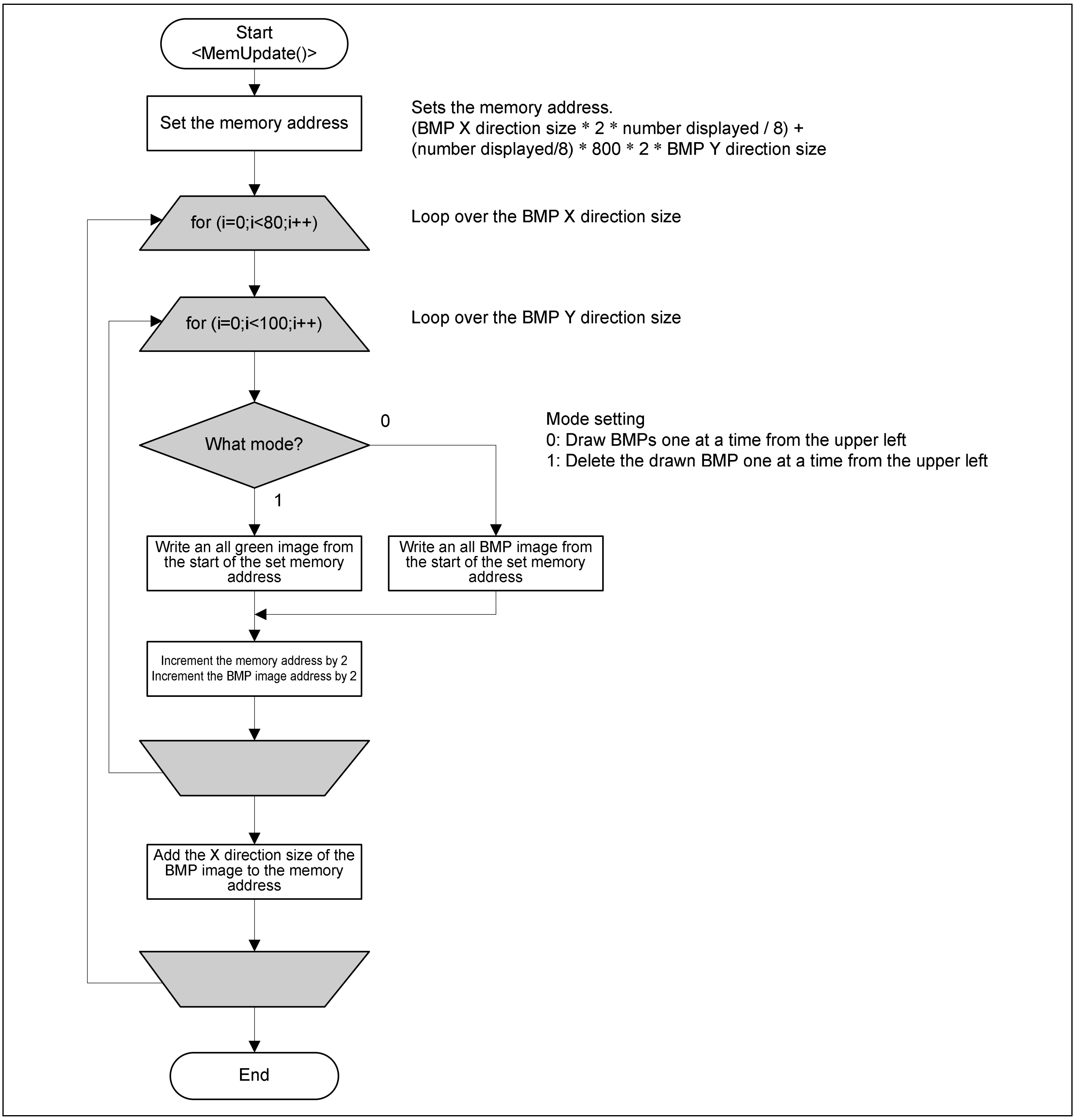

Start
<MemUpdate()>
Set the memory address
Sets the memory address.
(BMP X direction size * 2 * number displayed / 8) +
(number displayed/8) * 800 * 2 * BMP Y direction size
for (i=0;i<80;i++)
Loop over the BMP X direction size
for (i=0;i<100;i++)
Loop over the BMP Y direction size
What mode?
0
1
Mode setting
0: Draw BMPs one at a time from the upper left
1: Delete the drawn BMP one at a time from the upper left
Write an all green image from the start of the set memory address
Write an all BMP image from the start of the set memory address
Increment the memory address by 2
Increment the BMP image address by 2
Add the X direction size of the BMP image to the memory address
End

Figure 38 Plane 4 Image Memory Update Flowchart

3.3.19 Start of the FB Update DMA Transfer

Figure 39 shows the flowchart for starting the DMA transfer used to update the plane 4 frame buffer.

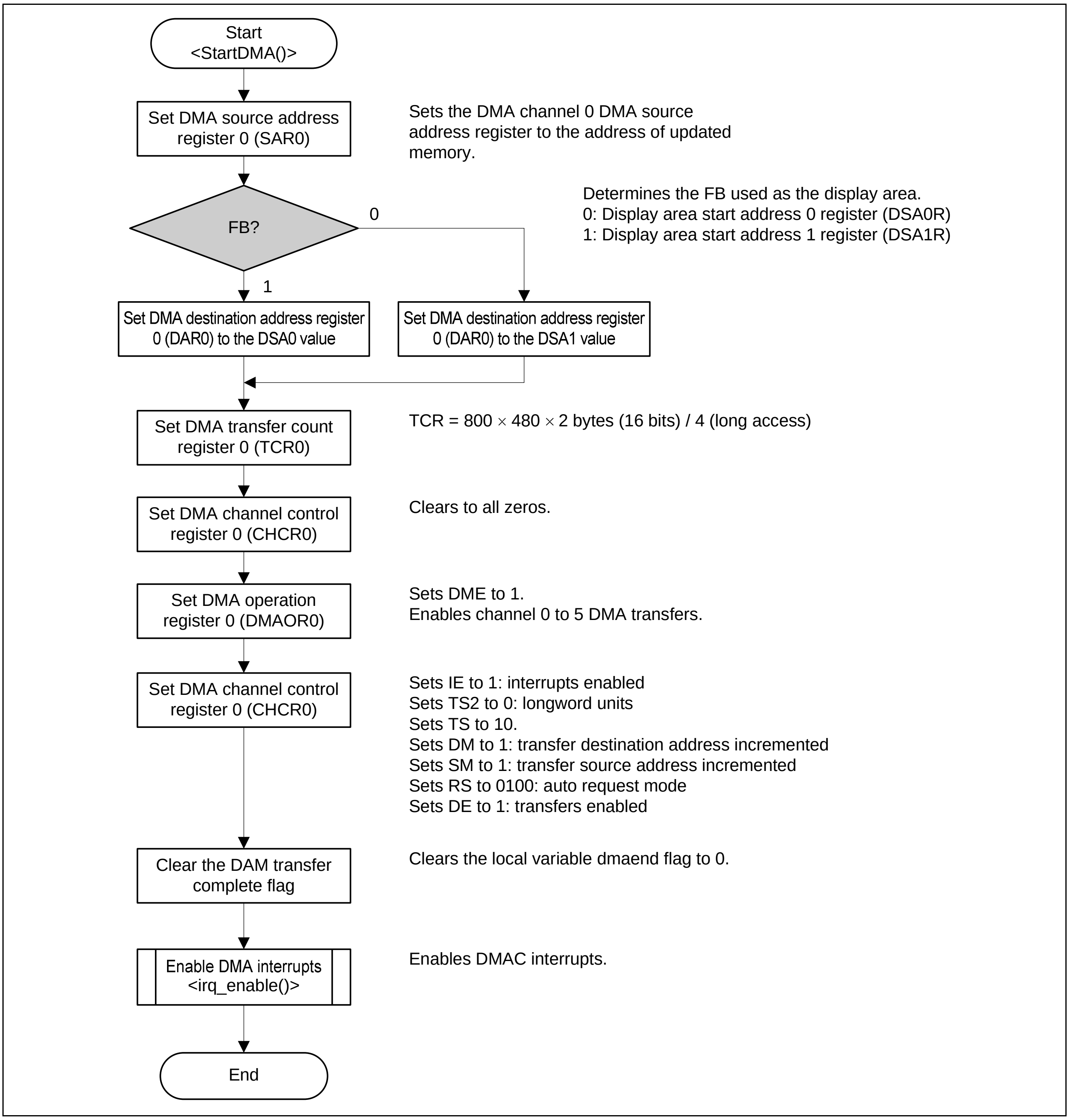

Start
<StartDMA()>
Set DMA source address register 0 (SAR0)
Sets the DMA channel 0 DMA source address register to the address of updated memory.
FB?
0
1
Determines the FB used as the display area.
0: Display area start address 0 register (DSA0R)
1: Display area start address 1 register (DSA1R)
Set DMA destination address register 0 (DAR0) to the DSA0 value
Set DMA destination address register 0 (DAR0) to the DSA1 value
Set DMA transfer count register 0 (TCR0)
TCR = 800 × 480 × 2 bytes (16 bits) / 4 (long access)
Set DMA channel control register 0 (CHCR0)
Clears to all zeros.
Set DMA operation register 0 (DMAOR0)
Sets DME to 1.
Enables channel 0 to 5 DMA transfers.
Set DMA channel control register 0 (CHCR0)
Sets IE to 1: interrupts enabled
Sets TS2 to 0: longword units
Sets TS to 10.
Sets DM to 1: transfer destination address incremented
Sets SM to 1: transfer source address incremented
Sets RS to 0100: auto request mode
Sets DE to 1: transfers enabled
Clear the DAM transfer complete flag
Clears the local variable dmaend flag to 0.
Enable DMA interrupts
<irq_enable()>
Enables DMAC interrupts.
End

Figure 39 Frame 4 Frame Buffer Update DMA Transfer Start Flowchart

3.3.20 DMA Interrupts

Figure 40 shows the flowchart for handling the DAM transfer complete interrupt.

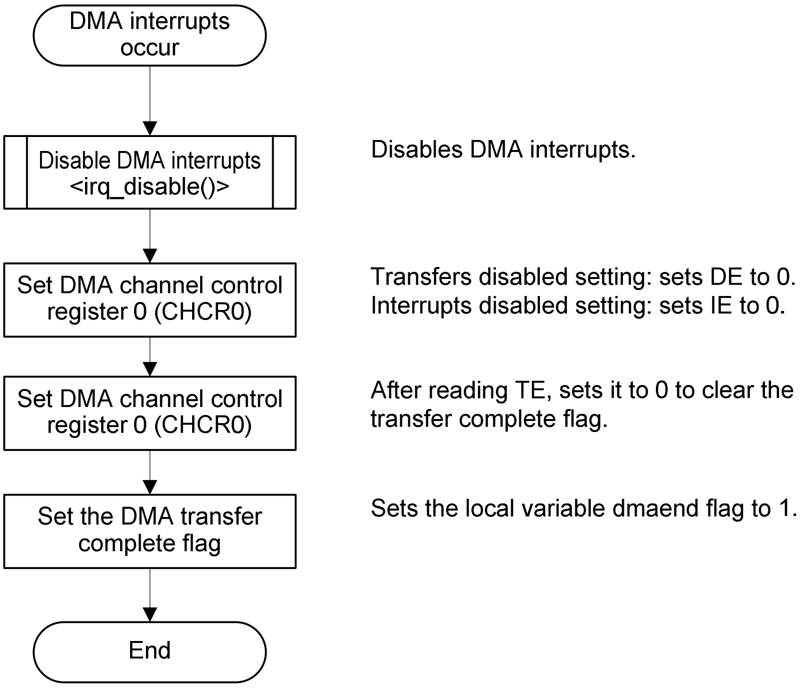

DMA interrupts occur
Disable DMA interrupts <irq_disable()>
Disables DMA interrupts.
Set DMA channel control register 0 (CHCR0)
Transfers disabled setting: sets DE to 0.
Interrupts disabled setting: sets IE to 0.
Set DMA channel control register 0 (CHCR0)
After reading TE, sets it to 0 to clear the transfer complete flag.
Set the DMA transfer complete flag
Sets the local variable dmaend flag to 1.
End

Figure 40 DMA Transfer Complete Interrupt Handling Flowchart

3.3.21 Section Allocations

Table 18 lists the section allocations used in this sample program.

Table 18 Section Allocations

Section	Section Usage	Area	Allocation Address (Virtual Address)	
P	Program area	ROM	0x00002000	P0 area (Can be cached, MMU address conversion possible)
C	Constant area	ROM		
C$BSEC	Uninitialized data area address structure	ROM		
C$DSEC	Initialized data area address structure	ROM		
D	Initialized data	ROM		
BMP	BMP file	ROM	0x00010000	
B	Uninitialized data area	RAM	0x0C000000	
R	Initialized data area	RAM		
S	Stack area	RAM	0x0DFF8000	
INTHandler	Exception/interrupt handler	ROM	0x80001000	P1 area (Can be cached, MMU address conversion not possible)
VECTTBL	Reset vector table Interrupt vector table	ROM		
INTTBL	Interrupt mask table	ROM		
PIntPRG	Interrupt function	ROM		
FRAMEBUF	Frame buffer	RAM	0x8C100000	
RSTHandler	Reset handler	ROM	0xA0000000	P2 area (Can not be cached, MMU address conversion not possible)
PResetPRG	Reset program	ROM		
PnonCACHE	Program area (Cache invalid access)	ROM		

4. Sample Program

Sample program listing: DU_SampleProgram.c

This is the main function in sample program.

```
/************************************************************************
* DISCLAIMER

* This software is supplied by Renesas Electronics Corporation. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Electronics Corporation. and is
* protected under all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* ELECTRONICS CORPORATION. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
**********************************************************************/
/* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved. */
/*""FILE COMMENT""*********** Technical reference data ****************
* System Name  : SH7785 Sample Program
* File Name    : DU_DemoSample.c
* Abstract     : SH7785 DU Demo Sample Program
* Version      : Ver 1.00
* Device       : SH7785
* Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
*              : C/C++ Compiler Package for SuperH Family (V.9.3.2.0)
* OS           : None
* H/W Platform : Alpha Project AP-SH4A-3A, an SH-4A board
* Description  : The SH7785 DU demo sample program.
*              :
* Operation    :
* Limitation   :
*              :
**********************************************************************
* History      : 30.SEP.2010 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/
/**********************************************************************/
/*                                                                    */
/*  FILE        :DU_DemoSample.c                                      */
/*  DATE        :Tue, Jul 20, 2010                                    */
/*  DESCRIPTION :Main Program                                         */
/*  CPU TYPE    :Other                                                */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.16).   */
/*                                                                     */
/***********************************************************************/



#ifdef __cplusplus
//#include <ios>                    // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;   // Remove the comment when you use ios
#endif

#include "config.h"
#include "du.h"
#include "bitmap.h"
#include "intc.h"

void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

/* ==== Variable Declarations ==== */
#define BUFF_MAX 7
#define yposcnt     XRES * 2 * 80
#define xupdate     (XRES - 100) * 2
#define pio_start0
#define pio_end     1
#define membase     5
int ImageCnt = 0;
int xshift=0, yshift=0;
static int modify = 0;
static int update = 0;
static int dmaend = 0;
static int mode = 0;
unsigned long start_address = (unsigned long)__sectop("BMP");
#define offsadd     fb_base + FRAME_SIZE*(membase*2)
#define offdadd     fb_base + FRAME_SIZE*(membase*2+1)

/* ==== Macro Declarations ==== */
#define xposcnt(a)  100 * 2 * (a)

/* ==== Function Declarations ==== */
void pinfunc_init( void );
int FbWriteBMP( int planenum, int fb );
void FbUpdate( int planenum, int fb);
void FbClear( int planenum );
void FbCopy( int planenum );
void MemClear( void );
void MemUpdate( int cnt, int mode );
void BuffClear(char *pBuff, int size);
/*** DU ***/
extern void du_init(void);
extern void du_display_ctl(int on_off);
extern void du_vbk_irq(void);
extern struct plane_info du_plane_info;
extern void du_plane_enable( int planenum, int pri );
extern void du_plane_disable( int planenum, int pri );
/*** SCIF ***/
extern int scif_init(void);
extern charscif_recive_data( char  *Data );
extern void scif_transmit_data( char *Data );
extern void scif_transmit_data_byte( char *Data );


/*""FUNC COMMENT""****************************************************
* ID                 :
* Outline            : Sample program main() function
*                    : (DU Display)
* Include            :
* Declaration        : void main(void)
* Description        : After SCIF initialization, displays "SH7785 DU DEMO Sample" on the console.
*                    : After DU initialization, clears the frame buffers for four planes of data
*                    : and writes BMP files to the frame buffers for each plane.
*                    : It then displays a WVGA image on the LCD display and enables the VSYNC interrupt.
*                    : The displayed/not displayed state of each plane can be controlled from the console.
*                    :
* Limitation         :
*                    :
* Argument           : none
* Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void main(void)
{
  int ret;
  int p1=1, p2=1, p3=1, p4=1;
  char KeyBuff[BUFF_MAX];

  ret = scif_init();
  if( ret == 0 )
     scif_transmit_data("\n\rSH7785 DU DEMO Sample\n");

  pinfunc_init();
  du_init();
  FbClear(PLANE1);
  FbClear(PLANE2);
  FbClear(PLANE3);
  FbClear(PLANE4);
  MemClear();
  if(FbWriteBMP(PLANE1, 0) != 0)
     scif_transmit_data("\rBMP Write Error Plane1 FB1\n");
  if(FbWriteBMP(PLANE2, 0) != 0)
     scif_transmit_data("\rBMP Write Error Plane2 FB0\n");
  if(FbWriteBMP(PLANE2, 1) != 0)
     scif_transmit_data("\rBMP Write Error Plane2 FB1\n");
  if(FbWriteBMP(PLANE3, 0) != 0)
     scif_transmit_data("\rBMP Write Error Plane3 FB1\n");
  if(MemWriteBMP() != 0)
     scif_transmit_data("\rBMP Write Error Memory\n");

  du_display_ctl(DISP_ON);
```

```
irq_enable(_DU);
while(1) {
   scif_transmit_data("\rPLANE ON/OFF SETTING(TOGGLE)\n");

   if(p1) {
      scif_transmit_data("\r### PLANE1 DISPLAY = ON  ###\n");
      du_plane_enable( PLANE1, du_plane_info.plane[PLANE1].pri );
   } else {
      scif_transmit_data("\r### PLANE1 DISPLAY = OFF ###\n");
      du_plane_disable( PLANE1, du_plane_info.plane[PLANE1].pri );
   }

   if(p2) {
      scif_transmit_data("\r### PLANE2 DISPLAY = ON  ###\n");
      du_plane_enable( PLANE2, du_plane_info.plane[PLANE2].pri );
   } else {
      scif_transmit_data("\r### PLANE2 DISPLAY = OFF ###\n");
      du_plane_disable( PLANE2, du_plane_info.plane[PLANE2].pri );
   }

   if(p3) {
      scif_transmit_data("\r### PLANE3 DISPLAY = ON  ###\n");
      du_plane_enable( PLANE3, du_plane_info.plane[PLANE3].pri );
   } else {
      scif_transmit_data("\r### PLANE3 DISPLAY = OFF ###\n");
      du_plane_disable( PLANE3, du_plane_info.plane[PLANE3].pri );
   }

   if(p4) {
      scif_transmit_data("\r### PLANE4 DISPLAY = ON  ###\n");
      du_plane_enable( PLANE4, du_plane_info.plane[PLANE4].pri );
   } else {
      scif_transmit_data("\r### PLANE4 DISPLAY = OFF ###\n");
      du_plane_disable( PLANE4, du_plane_info.plane[PLANE4].pri );
   }

   scif_transmit_data("\r-- Please Select Number --\n");
   scif_transmit_data("\r-- 1 : PLANE1           --\n");
   scif_transmit_data("\r-- 2 : PLANE2           --\n");
   scif_transmit_data("\r-- 3 : PLANE3           --\n");
   scif_transmit_data("\r-- 4 : PLANE4           --\n");
   scif_transmit_data("\r-- >");

   BuffClear( KeyBuff, BUFF_MAX );            // Clears Buff.
   while( scif_recive_data( KeyBuff ) != 0);
   switch( KeyBuff[0] ) {
      case '1' :
               p1 = !p1;
               break;
      case '2' :
               p2 = !p2;
               break;
      case '3' :
               p3 = !p3;
               break;
      case '4' :
               p4 = !p4;
```

```
                 break;
        default :
                 break;
     }
     scif_transmit_data("\n\n");


  }

}
/*""FUNC COMMENT""*****************************************************
 * ID                 :
 * Outline            : Sample program main() function
 *                    : (DU Display)
 * Include            :
 * Declaration        : void pinfunc_init( void )
 * Description        : Sets the pin functions.
 *                    :
 *                    :
 *                    :
 *                    :
 *                    :
 * Limitation         :
 *                    :
 * Argument           : none
 * Return Value       : none
 * Calling Functions :
 *""FUNC COMMENT END""**************************************************/

void pinfunc_init( void )
{
  GPIO.P1MSELR.WORD = 0x2180;
  GPIO.P2MSELR.WORD = 0x0000;
  GPIO.PBCR.WORD = 0xFFF0;
  GPIO.PCCR.WORD = 0x0000;
  GPIO.PDCR.WORD = 0x0000;
  GPIO.PHCR.WORD = 0xFC30;
  GPIO.PMCR.WORD = 0xFFF5;
  GPIO.PPCR.WORD = 0x03C0;
}

/* FB Functions */
/*""FUNC COMMENT""*****************************************************
 * ID                 :
 * Outline            : Sample program main() function
 *                    : (DU Display)
 * Include            :
 * Declaration        : void pset(int planenum, int fb, int x,int y,unsigned char r,unsigned char g,unsigned char b)
 * Description        : Writes one pixel (16 bits) of data to a frame buffer.
 *                    :
 *                    :
 *                    :
 *                    :
 *                    :
 * Limitation         :
 *                    :
 * Argument           : planenum: plane number, fb: FB plane, x: x coordinate, y: y coordinate
*                    : r: pixel red value, g: pixel green value, b: pixel blue value
* Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void  pset(int planenum, int fb, int x,int y,unsigned char r,unsigned char g,unsigned char b)
{
  intoffset, c, mask;
  int memadd;

  if ((x >= du_plane_info.xw) || (y >= du_plane_info.yw)) {
     return;
  }
  offset = (y * du_plane_info.plane[planenum].dsx + x) * 2;

  if (fb)
     memadd = DUP(planenum).DSA1R.LONG;
  else
     memadd = DUP(planenum).DSA0R.LONG;

  memadd += offset;
  *(volatile unsigned short *)memadd = RGB16(r, g, b);
}

#if defined(_BIG)
/*""FUNC COMMENT""*****************************************************
* ID                 :
* Outline            : Sample program main() function
*                    : (DU Display)
* Include            :
* Declaration        : unsigned short swap_endian16(unsigned short value)
* Description        : Performs a byte swap on a 16-bit datum.
*                    :
*                    :
*                    :
*                    :
*                    :
* Limitation         :
*                    :
* Argument           : Value before conversion
 Return Value        : Value after conversion
* Calling Functions :
*""FUNC COMMENT END""**************************************************/

unsigned short swap_endian16(unsigned short value)
{
    return ((value & 0xFF) << 8) | ((value >> 8) & 0xFF);
}

/*""FUNC COMMENT""*****************************************************
* ID                 :
* Outline            : Sample program main() function
*                    : (DU Display)
* Include            :
* Declaration        : unsigned short swap_endian32(unsigned long value)
* Description        : Performs a byte swap on a 32-bit datum.
*                    :
*                    :
```

```
*                    :
*                    :
*                    :
* Limitation         :
*                    :
* Argument           : Value before conversion
 Return Value        : Value after conversion
* Calling Functions :
*""FUNC COMMENT END""*************************************************/
unsigned long swap_endian32(unsigned long value)
{
  return (value >> 24) | (value << 24) | ((value >> 8) & 0xFF00) | ((value << 8) & 0xFF0000);
}

/*""FUNC COMMENT""*****************************************************
* ID                 :
* Outline            : Sample program main() function
*                    : (DU Display)
* Include            :
* Declaration        : int FbWriteBMP( int planenum, int fb )
* Description        : Writes a BMP image into a frame buffer.
*                    : (big endian)
*                    :
*                    :
*                    :
*                    :
* Limitation         :
*                    :
* Argument           : planenum: plane number, fb: FB plane
 Return Value        : -1 BMP file error
* Calling Functions :
*""FUNC COMMENT END""*************************************************/
int FbWriteBMP( int planenum, int fb )
{
  bitmap_file_header_t *file_header = (bitmap_file_header_t *)start_address;
  bitmap_header_t *bitmap_header = (bitmap_header_t *)(start_address + 14);
  long x, y;
  unsigned long bitmap_pitch, line;
  unsigned char red, green, blue, index;
  unsigned short rgb16;
  unsigned long offsetadd;

  if (swap_endian16(file_header->bitmap_file_type) != 0x4D42) {
     scif_transmit_data("\n\rERROR_INVALID_BITMAP_FILE\n");
     return(-1);
  }

  // Calculate pitches.
  bitmap_pitch = swap_endian32(file_header->bitmap_file_size);
  bitmap_pitch -= swap_endian32(file_header->bitmap_file_bits_offset);
  bitmap_pitch /= swap_endian32(bitmap_header->bitmap_height);

  // Load all lines.
  for (y = 0; y < swap_endian32(abs(bitmap_header->bitmap_height)); y++)
  {
     // Seek to line.
     if (swap_endian32(bitmap_header->bitmap_height) > 0)
     {
        line = swap_endian32(bitmap_header->bitmap_height) - y - 1;
     }
     else
     {
        line = y;
     }

     /* Address update */
     offsetadd = swap_endian32(file_header->bitmap_file_bits_offset) +
line * bitmap_pitch;

     // Load all pixels.
     for (x = 0; x < swap_endian32(bitmap_header->bitmap_width); x++)
     {
        rgb16 = swap_endian16(*(unsigned short *)(start_address + offsetadd + (x * 2)));
        red = ((rgb16 & 0xF800) >> 8);
        green = ((rgb16 & 0x07E0) >> 3);
        blue = ((rgb16 & 0x001F) << 3);
        pset(planenum, fb, x, y, red, green, blue);
     }
  }
  start_address += swap_endian32(file_header->bitmap_file_size);
  return 0;
}

#else
/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : int FbWriteBMP( int planenum, int fb )
* Description       : Writes a BMP image into a frame buffer.
*                   : (little endian)
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : planenum: plane number, fb: FB plane
 Return Value       : -1: BMP file error
* Calling Functions :
*""FUNC COMMENT END""***********************************************/
int FbWriteBMP( int planenum, int fb )
{
  bitmap_file_header_t *file_header = (bitmap_file_header_t *)start_address;
  bitmap_header_t *bitmap_header = (bitmap_header_t *)(start_address + 14);
  long x, y;
  unsigned long bitmap_pitch, line;
  unsigned char red, green, blue, index;
  unsigned short rgb16;
  unsigned long offsetadd;

  if (file_header->bitmap_file_type != 0x4D42) {
     scif_transmit_data("\n\rERROR_INVALID_BITMAP_FILE\n");
     return(-1);
  }

  // Calculate pitches.
  bitmap_pitch = file_header->bitmap_file_size;
  bitmap_pitch -= file_header->bitmap_file_bits_offset;
  bitmap_pitch /= bitmap_header->bitmap_height;

  // Load all lines.
  for (y = 0; y < abs(bitmap_header->bitmap_height); y++)
  {
     // Seek to line.
     if (bitmap_header->bitmap_height > 0)
     {
        line = bitmap_header->bitmap_height - y - 1;
     }
     else
     {
        line = y;
     }

     /* Address update */
     offsetadd = file_header->bitmap_file_bits_offset + line * bitmap_pitch;

     // Load all pixels.
     for (x = 0; x < bitmap_header->bitmap_width; x++)
     {
        rgb16 = *(unsigned short *)(start_address + offsetadd + (x * 2));
        red = ((rgb16 & 0xF800) >> 8);
        green = ((rgb16 & 0x07E0) >> 3);
        blue = ((rgb16 & 0x001F) << 3);
        pset(planenum, fb, x, y, red, green, blue);
     }
  }

  start_address += file_header->bitmap_file_size;
  return 0;
}
#endif
/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void FbClear( int planenum )
* Description       : Initializes a frame buffer to the transparent color (green).
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : planmenum: plane number
 Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""***************************************************/
```

```
void FbClear( int planenum )
{
  unsigned long address0 = DUP(planenum).DSA0R.LONG;
  unsigned long address1 = DUP(planenum).DSA1R.LONG;
  int i;
  unsigned long data = (unsigned long)(color(0, 63, 0)) << 16 | color(0, 63, 0);

  for (i = 0; i < FRAME_SIZE/4; i++) {
     *(unsigned long *)address0 = data;
     *(unsigned long *)address1 = data;
     address0 += 4;
     address1 += 4;
  }
}


#if defined(_BIG)
/*""FUNC COMMENT""*******************************************************
* ID                 :
* Outline            : Sample program main() function
*                    : (DU Display)
* Include            :
* Declaration        : int MemWriteBMP( void )
* Description        : (big endian)
*                    :
*                    :
*                    :
*                    :
*                    :
* Limitation         :
*                    :
* Argument           : none
 Return Value        : -1: BMP file error
* Calling Functions :
*""FUNC COMMENT END""***************************************************/
int MemWriteBMP( void )
{
  bitmap_file_header_t *file_header = (bitmap_file_header_t *)start_address;
  bitmap_header_t *bitmap_header = (bitmap_header_t *)(start_address + 14);
  long x, y;
  unsigned long bitmap_pitch, line;
  unsigned char red, green, blue, index;
  unsigned short rgb16;
  unsigned long offsetadd;
  intoffset;
  int memadd;

  if (swap_endian16(file_header->bitmap_file_type) != 0x4D42) {
     scif_transmit_data("\n\rERROR_INVALID_BITMAP_FILE\n");
     return(-1);
  }

  // Calculate pitches.
  bitmap_pitch = swap_endian32(file_header->bitmap_file_size);
  bitmap_pitch -= swap_endian32(file_header->bitmap_file_bits_offset);
  bitmap_pitch /= swap_endian32(bitmap_header->bitmap_height);


  // Load all lines.
  for (y = 0; y < swap_endian32(abs(bitmap_header->bitmap_height)); y++)
  {
     // Seek to line.
     if (swap_endian32(bitmap_header->bitmap_height) > 0)
     {
        line = swap_endian32(bitmap_header->bitmap_height) - y - 1;
     }
     else
     {
        line = y;
     }

     /* Address update */
     offsetadd = swap_endian32(file_header->bitmap_file_bits_offset) + line * bitmap_pitch;

     // Load all pixels.
     for (x = 0; x < swap_endian32(bitmap_header->bitmap_width); x++)
     {
        rgb16 = swap_endian16(*(unsigned short *)(start_address + offsetadd + (x * 2)));
        red = ((rgb16 & 0xF800) >> 8);
        green = ((rgb16 & 0x07E0) >> 3);
        blue = ((rgb16 & 0x001F) << 3);
        offset = (y * swap_endian32(bitmap_header->bitmap_width) + x) * 2;
        memadd = offsadd;
        memadd += offset;
        *(volatile unsigned short *)memadd = RGB16(red, green, blue);
     }
  }

  start_address += swap_endian32(file_header->bitmap_file_size);
  return 0;

}
#else
/*""FUNC COMMENT""******************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : int MemWriteBMP( void )
* Description       : (little endian)
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
 Return Value       : -1: BMP file error
* Calling Functions :
*""FUNC COMMENT END""*************************************************/
int MemWriteBMP( void )
{
  bitmap_file_header_t *file_header = (bitmap_file_header_t *)start_address;
  bitmap_header_t *bitmap_header = (bitmap_header_t *)(start_address + 14);
  long x, y;
  unsigned long bitmap_pitch, line;
  unsigned char red, green, blue, index;
  unsigned short rgb16;
  unsigned long offsetadd;
  intoffset;
  int memadd;

  if (file_header->bitmap_file_type != 0x4D42) {
     scif_transmit_data("\n\rERROR_INVALID_BITMAP_FILE\n");
     return(-1);
  }

  // Calculate pitches.
  bitmap_pitch = file_header->bitmap_file_size;
  bitmap_pitch -= file_header->bitmap_file_bits_offset;
  bitmap_pitch /= bitmap_header->bitmap_height;


  // Load all lines.
  for (y = 0; y < abs(bitmap_header->bitmap_height); y++)
  {
     // Seek to line.
     if (bitmap_header->bitmap_height > 0)
     {
        line = bitmap_header->bitmap_height - y - 1;
     }
     else
     {
        line = y;
     }

     /* Address update */
     offsetadd = file_header->bitmap_file_bits_offset + line * bitmap_pitch;

     // Load all pixels.
     for (x = 0; x < bitmap_header->bitmap_width; x++)
     {
        rgb16 = *(unsigned short *)(start_address + offsetadd + (x * 2));
        red = ((rgb16 & 0xF800) >> 8);
        green = ((rgb16 & 0x07E0) >> 3);
        blue = ((rgb16 & 0x001F) << 3);
        offset = (y * bitmap_header->bitmap_width + x) * 2;
        memadd = offsadd;
        memadd += offset;
        *(volatile unsigned short *)memadd = RGB16(red, green, blue);
     }
  }

  start_address += file_header->bitmap_file_size;
  return 0;

}
#endif

/*""FUNC COMMENT""*******************************************************
```

```
 * ID                 :
 * Outline            : Sample program main() function
 *                    : (DU Display)
 * Include            :
 * Declaration        : void MemClear( void )
 * Description        : Initializes image memory to the transparent color (green).
 *                    :
 *                    :
 *                    :
 *                    :
 *                    :
 * Limitation         :
 *                    :
 * Argument           : none
   Return Value       : none
 * Calling Functions :
 *""FUNC COMMENT END""**************************************************/
void MemClear( void )
{
  unsigned long address0 = offsadd;
  unsigned long address1 = offdadd;
  int i;
  unsigned long data = (unsigned long)(color(0, 63, 0)) << 16 | color(0, 63, 0);

  for (i = 0; i < FRAME_SIZE/4; i++) {
     *(unsigned long *)address0 = data;
     *(unsigned long *)address1 = data;
     address0 += 4;
     address1 += 4;
  }
}

/*""FUNC COMMENT""*****************************************************
 * ID                 :
 * Outline            : Sample program main() function
 *                    : (DU Display)
 * Include            :
 * Declaration        : void MemUpdate( int cnt, int mode )
 * Description        : Adds a single BMP image to image memory.
 *                    :
 *                    :
 *                    :
 *                    :
 *                    :
 * Limitation         :
 *                    :
 * Argument           : cnt: number of BMP display images, mode: 0 = increment/1 = decrement
   Return Value       : none
 * Calling Functions :
 *""FUNC COMMENT END""**************************************************/
void MemUpdate( int cnt, int mode )
{
  unsigned long sadd = offsadd;
  unsigned long dadd = offdadd;
  int i, j;

  /* Sets the transfer target start address. */
  dadd += (xposcnt(cnt%8)) + ((cnt/8) * yposcnt);

  for(i = 0;i < 80; i++) {
     for(j = 0;j < 100; j++) {
        if( mode )
           *(unsigned short *)dadd = color(0, 63, 0);
        else
           *(unsigned short *)dadd = *(unsigned short *)sadd;

        sadd += 2;
        dadd += 2;

     }
     dadd += xupdate;
  }
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void StartDMA( int planenum, int fb )
* Description       : Initializes DMAC channel 0 and starts a DMA transmission
*                   : Enables DMA interrupts
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : planenum: plane number, fb: FB plane
 Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void StartDMA( int planenum, int fb )
{
  *(unsigned long *)0xFC808020 = offdadd | 0xA0000000;
  if ( fb )
     *(unsigned long *)0xFC808024 = DUP(planenum).DSA0R.LONG | 0xA0000000;
  else
     *(unsigned long *)0xFC808024 = DUP(planenum).DSA1R.LONG | 0xA0000000;

  DMAC0.TCR.BIT.CNT = XRES * YRES * 2 / 4;
  DMAC0.CHCR.LONG   = 0;
  DMAC.DMAOR0.BIT.DME  = 1;  /* Enable DMAC0 to DMAC5 */
  DMAC0.CHCR.BIT.IE    = 1;  /* Enable interrupts */
  DMAC0.CHCR.BIT.TS2   = 0;  /* long access */
  DMAC0.CHCR.BIT.TS    = 2;  /* long access */
  DMAC0.CHCR.BIT.DM    = 1;  /* Increment the transfer target */
  DMAC0.CHCR.BIT.SM    = 1;  /* Increment the transfer source */
  DMAC0.CHCR.BIT.RS    = 4;  /* Auto request */
  DMAC0.CHCR.BIT.DE    = 1;  /* Enable transfers */
  dmaend = 0;
  irq_enable( _DMAC0 );
}

```

```
/*""FUNC COMMENT""******************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void dmac0_irq( void )
* Description       : Disables DMAC interrupts and disables transfers
*                   : Sets the DMA transfer complete flag.
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
 Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void dmac0_irq( void )
{
  int tmp;
  DMAC0.CHCR.BIT.IE = 0;  /* Disable interrupts */
  DMAC0.CHCR.BIT.DE = 0;  /* Disable transfers */
  tmp = DMAC0.CHCR.BIT.TE;
  DMAC0.CHCR.BIT.TE = 0;  /* Flag clear*/
  dmaend = 1;
}

/* Demo Sample Program */

/*""FUNC COMMENT""******************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void DemoSample( void )
* Description       : Updates the frame buffers for planes 1 to 4.
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
 Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void DemoSample( void )
{
  int i = 0;
  int xpos = XRES - 80;
  int ypos = YRES - 80;

  /* PLANE1 */
  if ((DUP(i).DPXR.BIT.DPX > 0) && (DUP(i).DPYR.BIT.DPY == 0)) /* Move left (←) */
     DUP(i).DPXR.BIT.DPX -= 2;
```

```
else if((DUP(i).DPXR.BIT.DPX == 0) && (DUP(i).DPYR.BIT.DPY < ypos)) /* Move down (↓) */
   DUP(i).DPYR.BIT.DPY += 2;
else if((DUP(i).DPXR.BIT.DPX < xpos) && (DUP(i).DPYR.BIT.DPY == ypos)) /* Move right (→)*/
   DUP(i).DPXR.BIT.DPX += 2;
else if((DUP(i).DPXR.BIT.DPX == xpos) && (DUP(i).DPYR.BIT.DPY > 0)) /* Move up (↑) */
   DUP(i).DPYR.BIT.DPY -= 2;

i++;

/* PLANE2 */
if((modify%20) == 0)
   DUP(i).MR.BIT.DC = 1;

if ((DUP(i).DPXR.BIT.DPX < xpos) && (DUP(i).DPYR.BIT.DPY == 0)) /* Move right (→) */
   DUP(i).DPXR.BIT.DPX += 1;
else if((DUP(i).DPXR.BIT.DPX == xpos) && (DUP(i).DPYR.BIT.DPY < ypos)) /* Move down (↓) */
   DUP(i).DPYR.BIT.DPY += 1;
else if((DUP(i).DPXR.BIT.DPX > 0) && (DUP(i).DPYR.BIT.DPY == ypos)) /* Move left (←) */
   DUP(i).DPXR.BIT.DPX -= 1;
else if((DUP(i).DPXR.BIT.DPX == 0) && (DUP(i).DPYR.BIT.DPY > 0)) /* Move up (↑)*/
   DUP(i).DPYR.BIT.DPY -= 1;

modify++;
i++;

/* PLANE3 */
if ((DUP(i).DPXR.BIT.DPX == 0) && (DUP(i).DPYR.BIT.DPY == 0)) {
   xshift = 2;
   yshift = 2;
} else if ((DUP(i).DPXR.BIT.DPX == xpos) && (DUP(i).DPYR.BIT.DPY == ypos)) {
   xshift = -2;
   yshift = -2;
} else if (DUP(i).DPXR.BIT.DPX == 0) {
   xshift = 2;
} else if(DUP(i).DPYR.BIT.DPY == 0) {
   yshift = 2;
} else if(DUP(i).DPXR.BIT.DPX == xpos) {
   xshift = -2;
} else if(DUP(i).DPYR.BIT.DPY == ypos) {
   yshift = -2;
} else {
   xshift = xshift;
   yshift = yshift;
}

DUP(i).DPXR.BIT.DPX += xshift;
DUP(i).DPYR.BIT.DPY += yshift;

i++;
/* PLANE4 */
/* Transfer the original image to image memory using PIO. */
if( update == pio_start ) {
   if( ImageCnt == 48 ) {
      mode = !mode;
      ImageCnt = 0;
   }

```

```
     MemUpdate( ImageCnt, mode );
     update = pio_end;
     ImageCnt++;
     StartDMA( 3, DU.DSSR.BIT.DFB4 );
  }

  if(((modify%60) == 0) && dmaend) {
     DUP(i).MR.BIT.DC = 1;
     update = pio_start;
     dmaend = 0;
  }
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void BuffClear(char *pBuff, int size)
* Description       : Serial reception data buffer initialization
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : *pBuff: Buffer, size: Buffer size
 Return Value       : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void BuffClear(char *pBuff, int size)
{
  int i;
  for( i = 0; i < size; i++ )   /* Clear the serial data reception work area. */
  {
     *( pBuff + i ) = 0;
  }
}



#ifdef __cplusplus
void abort(void)
{

}
#endif
```

Sample program listing: du.c

Initializes the DU.

```
/*************************************************************************
* DISCLAIMER

* This software is supplied by Renesas Electronics  Corporation. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Electronics  Corporation. and is
* protected under all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* ELECTRONICS CORPORATION. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
*************************************************************************/
/* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
/*""FILE COMMENT""*********** Technical reference data ****************
* System Name  : SH7785 Sample Program
* File Name    : du.c
* Abstract     : SH7785 DU Initial Settings Sample Program
* Version      : Ver 1.00
* Device       : SH7785
* Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
*              : C/C++ Compiler Package for SuperH Family (V.9.3.2.0)
* OS           : None
* H/W Platform : Alpha Project AP-SH4A-3A, an SH-4A board
* Description  : The SH7785 DU initial settings sample program.
*              :
* Operation    :
* Limitation   :
*              :
***********************************************************************
* History      : 30.Sep.2010 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/

#include "du.h"

/* ==== Function Declarations ==== */
extern void DemoSample( void );
extern void delay( int cnt );

struct plane_info du_plane_info = {
```

```
DU_CLK_INTERNAL,           /* clock_source */
HTOTAL,                    /* hc - X direction total */
VTOTAL,                    /* vc – Y direction total */
XRES,                      /* xw - X direction display size */
YRES,                      /* yw – Y direction display size */
X_FRONT,                   /* xe - X direction front porch */
X_BACK,                    /* xs - X direction back porch */
Y_FRONT,                   /* ye - Y direction front porch */
Y_BACK,                    /* ys - Y direction back porch */
H_WIDTH,                   /* hsw Hsync width */
V_WIDTH,                   /* vsw Vsync width */
{{ /* plane1 */
   DISP_ON,             /* plane_enable */
   0,                   /* dsa0 - frame buffer address 0 */
   0,                   /* dsa1 - frame buffer address 1 */
   80,                  /* dsx - plane display size in the X direction */
   80,                  /* dsy - plane display size in the Y direction */
   XRES-80,             /* dpx - Separation in the X direction from the dsa origin display position */
   0,                   /* dpy - Separation in the Y direction from the dsa origin display position */
   0,                   /* spx - Separation in the X direction from the dsa origin start position */
   0,                   /* spy - Separation in the Y direction from the dsa origin start position */
   0,                   /* waspy - Y direction start position for the wraparound area */
   80,                  /* wamwy - Y direction memory width for wraparound (240 to 4095) */
   1,                   /* pri - Priority */
   80,                  /* mwx - X direction memory width for the plane (16 to 4096) */
   80,                  /* mly - Y direction memory area */
},
{  /* plane2 */
   DISP_ON,             /* plane_enable */
   0,                   /* dsa0 - frame buffer address 0 */
   0,                   /* dsa1 - frame buffer address 1 */
   80,                  /* dsx - plane display size in the X direction */
   80,                  /* dsy - plane display size in the Y direction */
   0,                   /* dpx - Separation in the X direction from the dsa origin display position */
   0,                   /* dpy - Separation in the Y direction from the dsa origin display position */
   0,                   /* spx - Separation in the X direction from the dsa origin start position */
   0,                   /* Spy - Separation in the Y direction from the dsa origin start position */
   0,                   /* waspy - Y direction start position for the wraparound area */
   80,                  /* wamwy - Y direction memory width for wraparound (240 to 4095) */
   2,                   /* pri - Priority */
   80,                  /* mwx - X direction memory width for the plane (16 to 4096) */
   80,                  /* mly - Y direction memory area */
},
{  /* plane3 */
   DISP_ON,             /* plane_enable */
   0,                   /* dsa0 - frame buffer address 0 */
   0,                   /* dsa1 - frame buffer address 1 */
   80,                  /* dsx - plane display size in the X direction */
   80,                  /* dsy - plane display size in the Y direction */
   0,                   /* dpx - Separation in the X direction from the dsa origin display position */
   YRES-80,             /* dpy - Separation in the Y direction from the dsa origin display position */
   0,                   /* spx - Separation in the X direction from the dsa origin start position */
   0,                   /* spy - Separation in the Y direction from the dsa origin start position */
   0,                   /* waspy - Y direction start position for the wraparound area */
   80,                  /* wamwy - Y direction memory width for wraparound (240 to 4095) */
   3,                   /* pri - Priority */
   80,                  /* mwx - X direction memory width for the plane (16 to 4096) */
```

```
     80,                  /* mly - Y direction memory area */
  },
#if !defined(CONFIG_SCREEN_SVGA)
  {  /* plane4 */
     DISP_ON,             /* plane_enable */
     0,                   /* dsa0 - frame buffer address 0 */
     0,                   /* dsa1 - frame buffer address 1 */
     XRES,                /* dsx - plane display size in the X direction */
     YRES,                /* dsy - plane display size in the Y direction */
     0,                   /* dpx - Separation in the X direction from the dsa origin display position */
     0,                   /* dpy - Separation in the Y direction from the dsa origin display position */
     0,                   /* spx - Separation in the X direction from the dsa origin start position */
     0,                   /* spy - Separation in the Y direction from the dsa origin start position */
     0,                   /* waspy - Y direction start position for the wraparound area */
     YRES,                /* wamwy - Y direction memory width for wraparound (240 to 4095) */
     4,                   /* pri - Priority */
     XRES,                /* mwx - X direction memory width for the plane (16 to 4096) */
     YRES,                /* mly - Y direction memory area */
  },
#if defined(CONFIG_SCREEN_VGA)
  {  /* plane5 */
     DISP_ON,             /* plane_enable */
     0,                   /* dsa0 - frame buffer address 0 */
     0,                   /* dsa1 - frame buffer address 1 */
     40,                  /* dsx - plane display size in the X direction */
     40,                  /* dsy - plane display size in the Y direction */
     0,                   /* dpx - Separation in the X direction from the dsa origin display position */
     0,                   /* dpy - Separation in the Y direction from the dsa origin display position */
     0,                   /* spx - Separation in the X direction from the dsa origin start position */
     0,                   /* spy - Separation in the Y direction from the dsa origin start position */
     0,                   /* waspy - Y direction start position for the wraparound area */
     YRES,                /* wamwy - Y direction memory width for wraparound (240 to 4095) */
     5,                   /* pri - Priority */
     0,                   /* mwx - X direction memory width for the plane (16 to 4096) */
     0,                   /* mly - Y direction memory area */
  },
  {  /* plane6 */
     DISP_ON,             /* plane_enable */
     0,                   /* dsa0 - frame buffer address 0 */
     0,                   /* dsa1 - frame buffer address 1 */
     40,                  /* dsx - plane display size in the X direction */
     40,                  /* dsy - plane display size in the Y direction */
     0,                   /* dpx - Separation in the X direction from the dsa origin display position */
     0,                   /* dpy - Separation in the Y direction from the dsa origin display position */
     0,                   /* spx - Separation in the X direction from the dsa origin start position */
     0,                   /* spy - Separation in the Y direction from the dsa origin start position */
     0,                   /* waspy - Y direction start position for the wraparound area */
     0,                   /* wamwy - Y direction memory width for wraparound (240 to 4095) */
     6,                   /* pri - Priority */
     0,                   /* mwx - X direction memory width for the plane (16 to 4096) */
     0,                   /* mly - Y direction memory area */
  },
#endif /* CONFIG_SCREEN_VGA | CONFIG_SCREEN_WVGA */
#endif /* CONFIG_SCREEN_SVGA */
  },
};

```

```
/*""FUNC COMMENT""*****************************************************
 * ID                :
 * Outline           : Sample program main() function
 *                   : (DU Display)
 * Include           :
 * Declaration       : void du_plane_init( int planenum )
 * Description       : Plane initialization
 *                   :
 *                   :
 *                   :
 *                   :
 *                   :
 * Limitation        :
 *                   :
 * Argument          : planenum: Plane number
 * Return Value      : none
 * Calling Functions :
 *""FUNC COMMENT END""*************************************************/
void du_plane_init( int planenum )
{
     /* DSA settings for each plane */
     DUP(planenum).DSA0R.LONG = fb_base + FRAME_SIZE*(planenum*2);
     DUP(planenum).DSA1R.LONG = fb_base + FRAME_SIZE*(planenum*2+1);

     /* Mode settings for each plane */
     DUP(planenum).MR.LONG = FMT_BPP16 | BUF_MODE_MANU;

     /* Screen size settings for each plane */
     DUP(planenum).DSXR.BIT.DSX = du_plane_info.plane[planenum].dsx;
     DUP(planenum).DSYR.BIT.DSY = du_plane_info.plane[planenum].dsy;

     /* Display position settings for each plane */
     DUP(planenum).DPXR.LONG = du_plane_info.plane[planenum].dpx;
     DUP(planenum).DPYR.LONG = du_plane_info.plane[planenum].dpy;

     /* Start position settings for each plane */
     DUP(planenum).SPXR.BIT.SPX = du_plane_info.plane[planenum].spx;
     DUP(planenum).SPYR.BIT.SPY = du_plane_info.plane[planenum].spy;

     /* X direction memory width setting for each plane */
     DUP(planenum).MWR.LONG = du_plane_info.plane[planenum].mwx;

     /* Transparent color settings for each plane */
     DUP(planenum).TC2R.BIT.TC2 = color(0, 63, 0);

}

/*""FUNC COMMENT""*****************************************************
 * ID                :
 * Outline           : Sample program main() function
 *                   : (DU Display)
 * Include           :
 * Declaration       : void du_plane_enable( int planenum, int pri )
 * Description       : Turns plane display on.
 *                   :
 *                   :
 *                   :
```

```
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : planenum: Plane number, pri: Priority
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_plane_enable( int planenum, int pri )
{
  unsigned long tmp;
  int i, dpe, dps;
  tmp = DU.DPPR.LONG;


  for(i = 0;i < 6; i++) {
     if( ((tmp >> (i * 4)) & 0x7) == (pri - 1))
        tmp &= ~(0xf << (i * 4));
  }

  switch (pri) {
     case 1:
           dpe = 3;
            dps = 0;
            break;
     case 2:
           dpe = 7;
            dps = 4;
            break;
     case 3:
           dpe = 11;
            dps = 8;
            break;
     case 4:
           dpe = 15;
            dps = 12;
            break;
     case 5:
           dpe = 19;
            dps = 16;
            break;
     case 6:
           dpe = 23;
            dps = 20;
            break;
     default:
           break;
  }
  tmp &= ~(0xf << dps);
  tmp |= (1 << dpe) + (planenum << dps);
  DU.DPPR.LONG = tmp;
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_plane_init( int planenum )
* Description       : Turns plane display off.
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : planenum: Plane number, pri: Priority
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_plane_disable( int planenum, int pri )
{
  unsigned long tmp;
  int i, dpe, dps;
  tmp = DU.DPPR.LONG;


  for(i = 0;i < 6; i++) {
     if( ((tmp >> (i * 4)) & 0x7) == (pri - 1))
        tmp &= ~(0xf << (i * 4));
  }

  switch (pri) {
     case 1:
           dpe = 3;
            dps = 0;
            break;
     case 2:
           dpe = 7;
            dps = 4;
            break;
     case 3:
           dpe = 11;
            dps = 8;
            break;
     case 4:
           dpe = 15;
            dps = 12;
            break;
     case 5:
           dpe = 19;
            dps = 16;
            break;
     case 6:
           dpe = 23;
            dps = 20;
            break;
     default:
           break;
  }
  tmp &= ~(0xf << dps);
  tmp |= (0 << dpe) + (planenum << dps);
  DU.DPPR.LONG = tmp;
```

```
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_init( void )
* Description       : DU initialization
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_init( void )
{
  int i;

  /* Initialization */
  DU.DSYSR.BIT.DRES = 1;
  DU.DSYSR.BIT.TVM = MASTER_MODE;
#if defined(_BIG)
  DU.DSYSR.BIT.DSEC = 1;        /* Big endian*/
#endif

  DU.DSMR.BIT.CSPM = 1;      /* HSYNC signal output */
  DU.DSMR.BIT.CDED = 1;      /* Disables the CDE signal */

  DU.DSRCR.LONG = 0x0000CB00;   /* Clears the display status register */

  DU.DIER.BIT.VBE = 1;       /* Enables vertical interrupts */

  DU.CPCR.LONG = 0x00000000;    /* Color palettes 1 to 4 are unused */

//   DU.DEFR.BIT.DSAE = 1;         /* Frame address expansion (31 to 4) */
  DU.DEFR.BIT.DCKE = 1;

  /* Horizontal display start position*/
  DU.HDSR.BIT.HDS = du_plane_info.hsw + du_plane_info.xs - REVISE;

  /* Horizontal display end position */
  DU.HDER.BIT.HDE = DU.HDSR.BIT.HDS + du_plane_info.xw;

  /* Vertical display start position */
  DU.VDSR.BIT.VDS = du_plane_info.ys - 2;

  /* Vertical display end position */
  DU.VDER.BIT.VDE = DU.VDSR.BIT.VDS + du_plane_info.yw;

  /* Horizontal scan period */
  DU.HCR.BIT.HC = du_plane_info.hc - 1;

  /* Horizontal sync signal pulse width */
  DU.HSWR.BIT.HSW = du_plane_info.hsw - 1;

  /* Vertical sync position */
  DU.VSPR.BIT.VSP = du_plane_info.vc - du_plane_info.vsw - 1;

  /* Vertical scan period */
  DU.VCR.BIT.VC = du_plane_info.vc - 1;

  DU.EQWR.LONG = 0;
  DU.SPWR.LONG = 0;
  DU.CLAMPSR.LONG = 0;
  DU.CLAMPWR.LONG = 1;

  DU.CP1TR.LONG = 0;      /* Color palette 1 is unused */
  DU.CP2TR.LONG = 0;      /* Color palette 2 is unused */
  DU.CP3TR.LONG = 0;      /* Color palette 3 is unused */
  DU.CP4TR.LONG = 0;      /* Color palette 4 is unused */
  DU.DOOR.LONG = 0x0000FC00; /* Color setting when the plane is unused: green */
  DU.CDER.LONG = 0;       /* Color detection is not set. */
  DU.BPOR.LONG = 0x00204494; /* Background color setting: indigo */
  DU.RINTOFSR.LONG = 0;   /* Raster interrupts are not set. */


  /* Initialization for each plane */
  for (i=0;i<PLANE_NUM;i++) {
     du_plane_init( i );
     du_plane_enable( i, du_plane_info.plane[i].pri );
  }

  DU.ESCR.BIT.DCLKSEL = du_plane_info.clk_sorce;
  DU.ESCR.BIT.FRQSEL = 3;    /* Divides the input clock by 4. */

  DU.OTAR.LONG = 0;
}

/*""FUNC COMMENT""****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_display_ctl( int on_off )
* Description       : DU display on/off control
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : on_off: On when 1, off when 0.
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_display_ctl( int on_off )
{
  if (on_off) {
     DU.DSYSR.BIT.DRES = 0;
     DU.DSYSR.BIT.DEN = 1;
  } else {
     DU.DSYSR.BIT.DRES = 1;
     DU.DSYSR.BIT.DEN = 0;
  }
}

/*""FUNC COMMENT""******************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_tvr_irq(void)
* Description       : TVR interrupt handling
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_tvr_irq(void)
{
  DU.DSRCR.BIT.TVCL = 1;
}

/*""FUNC COMMENT""******************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_frm_irq(void)
* Description       : FRM interrupt handling
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_frm_irq(void)
{
  DU.DSRCR.BIT.FRCL = 1;
}


/*""FUNC COMMENT""******************************************************
```

```
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_vbk_irq(void)
* Description       : VBK interrupt handling
*                   : Demo Sample program 1 execution
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""*************************************************/
void du_vbk_irq(void)
{
  DU.DSRCR.BIT.VBCL = 1;

  DemoSample();
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_rint_irq(void)
* Description       : RINT interrupt handling
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""*************************************************/
void du_rint_irq(void)
{
  DU.DSRCR.BIT.RICL = 1;

}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void du_hbk_irq(void)
* Description       : HBK interrupt handling
*                   :
*                   :
```

```
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void du_hbk_irq(void)
{
  DU.DSRCR.BIT.HBCL = 1;

}

```

Sample program listing: du.h

Header file for "du.c"

```

#ifndef _DU_H_
#define _DU_H_

#include "config.h"
#include "iodefine.h"

#define PLANE10
#define PLANE21
#define PLANE32
#define PLANE43
#define PLANE54
#define PLANE65

#if defined(CONFIG_SCREEN_VGA)
#define XRES        640
#define YRES        480
#define X_FRONT     105
#define X_BACK      16
#define Y_FRONT     33
#define Y_BACK      10
#define H_WIDTH     39
#define V_WIDTH     2
#define PLANE_NUM4
#elif defined(CONFIG_SCREEN_WVGA)
#define XRES        800
#define YRES        480
#define X_FRONT     220
#define X_BACK      110
#define Y_FRONT     35
#define Y_BACK      5
#define H_WIDTH     128
#define V_WIDTH     5
#define PLANE_NUM4
#elif defined(CONFIG_SCREEN_SVGA)
#define XRES        800
#define YRES        600
#define X_FRONT     0
#define X_BACK      0
#define Y_FRONT     0
#define Y_BACK      0
#define H_WIDTH     0
#define V_WIDTH     0
#define PLANE_NUM3
#else
#define XRES        480
#define YRES        234
#define X_FRONT     0
#define X_BACK      0
#define Y_FRONT     0
#define Y_BACK      0
#define H_WIDTH     0
#define V_WIDTH     0
#define PLANE_NUM6
```

```
#endif

#define HTOTALXRES + X_FRONT + X_BACK + H_WIDTH
#define VTOTALYRES + Y_FRONT + Y_BACK + V_WIDTH

/* Transparent color settings: RGB = 5:6:5, R/B = 0 to 31, G = 0 to 63 */
#define color(R, G, B) (R << 11) | (G << 5) | B


#define REVISE19    /* Master */
//#define  REVISE24 /* TV synchronization */

#define MASTER_MODE 0
#define TV_MODE     2

#define DU_CLK_INTERNAL1
#define DU_CLK_EXTERNAL0

#define DISP_ON     1
#define DISP_OFF    0

static int fb_base = (int)__sectop("FRAMEBUF");
#define FRAME_SIZE  1024 * 1024 * 1/* 1MB */

#define DUP(ch)    (*(volatile struct st_dup     *)((unsigned long )0xFFF80000 + ((ch+1) << 8)))

/* PnMR */
#define FMT_BPP8    0
#define FMT_BPP16   1
#define FMT_ARGB    2
#define FMT_YC      3
#define BUF_MODE_AUTO  (2 << 4)
#define BUF_MODE_MANU  (0 << 4)
#define DC_ON       (1 << 7)
#define DC_OFF      (0 << 7)
#define WAE_ON      (1 << 16)
#define WAE_OFF     (0 << 16)




struct plane_cfg {
  int plane_enable; /* Plane display on: 1, off: 0 */
  intdsa0;          /* Frame buffer address 0 */
  intdsa1;          /* Frame buffer address 1 */
  int dsx;          /* Plane display size in the X direction*/
  int dsy;          /* Plane display size in the Y direction */
  intdpx;           /* Separation in the X direction from the dsa origin display position */
  intdpy;           /* Separation in the Y direction from the dsa origin display position */
  int spx;          /* Separation in the X direction from the dsa origin start position */
  int spy;          /* Separation in the Y direction from the dsa origin start position */
  int waspy;        /* Y direction start position for the wraparound area */
  int wamwy;        /* Y direction memory width for wraparound (240 to 4095) */
  int pri;          /* pri - Priority */
  int mwx;          /* X direction memory width for the plane (16 to 4096) */
  int mly;          /* Y direction memory area */
};
```

```

struct plane_info {
  int clk_sorce;
  int hc;           /* X direction total */
  int vc;           /* Ydirection total */
  int xw;        /* X direction display size */
  int yw;        /* Y direction display size */
  int xe;        /* X direction back porch */
  int xs;        /* X direction front porch */
  int ye;        /* Y direction back porch */
  int ys;        /* Y direction front porch */
  int hsw;    /* Hsync width */
  int vsw;    /* Vsync width */
  struct plane_cfg plane[PLANE_NUM];
};




#endif /* _DU_H_ */
```

Sample program listing: scif.c

Sample program that demonstrates serial communications channel 1 initialization.

```
/***********************************************************************
* DISCLAIMER

* This software is supplied by Renesas Electronics  Corporation. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Electronics  Corporation. and is
* protected under all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* ELECTRONICS CORPORATION. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
***************************************************************************/
/* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
/*""FILE COMMENT""*********** Technical reference data ****************
* System Name  : SH7785 Sample Program
* File Name    : scif.c
* Abstract     : SH7785 SCIF Initial Settings Sample Program
* Version      : Ver 1.00
* Device       : SH7785
* Tool-Chain   : High-performance Embedded Workshop (Version 4.07.00.007)
*              : C/C++ Compiler Package for SuperH Family (V.9.3.2.0)
* OS           : None
* H/W Platform : Alpha Project AP-SH4A-3A, an SH-4A board
* Description  : The SH7785 SCIF initial settings sample program.
*              :
* Operation    :
* Limitation   :
*              :
*********************************************************************
* History      : 01.Sep.2010 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/


#include"scif.h"

/*""FUNC COMMENT""**************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : int delay( int cnt )
* Description       : Software wait
*                   : Iterates a for statement loop cnt times.
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : cnt
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
void delay( int cnt )
{
  int i;
  for(i=0;i<cnt;i++);
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : int scif_init(void)
* Description       : SCIF initialization
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : -1: baud rate clock calculation error
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
int scif_init(void)
{
  unsigned short data;
  int t = -1, cnt = 0;

  SCIF.SCSCR.WORD = 0x0000;  /* TIE, RIE, TE, RE Clear */

  SCIF.SCFCR.BIT.TFCL = 1;/* Tx FIFO Clear */
  SCIF.SCFCR.BIT.RFCL = 1;/* Rx FIFO Clear */

  SCIF.SCFSR.WORD = 0x0000;  /* BRK, DR, TR Clear */
  SCIF.SCLSR.BIT.ORER = 0;/* ORER Clear */

#if defined(CONFIG_SCIF_CLK_EXTERNAL)
  SCIF.SCSCR.BIT.CKE = 2; /* Clock source:SCK */
#elif  defined(CONFIG_SCIF_CLK_PCLK)
  SCIF.SCSCR.BIT.CKE = 0; /* Clock source:PCLK */
  t = SCBRR_VALUE(CONFIG_BPS, CONFIG_SCIF_CLK_PCLK);
#endif /* CONFIG_SCIF_CLK */

  if(t > 0) {
     while(t >= 256) {
        cnt++;
        t >> 2;
     }
     if(cnt > 3)
        return -1;

     SCIF.SCSMR.BIT.CKS = cnt;
     SCIF.SCBRR = t;
  }
  delay(1000);

  SCIF.SCFCR.BIT.RTRG = 0;
  SCIF.SCFCR.BIT.TTRG = 0;
  SCIF.SCFCR.BIT.TFCL = 1;/* Tx FIFO Clear */
  SCIF.SCFCR.BIT.RFCL = 1;/* Rx FIFO Clear */

  SCIF.SCFCR.BIT.TFCL = 0;/* Tx FIFO Not Clear */
  SCIF.SCFCR.BIT.RFCL = 0;/* Rx FIFO Not Clear */
  SCIF.SCSCR.BIT.TE  = 1;
  SCIF.SCSCR.BIT.RE  = 1;
  return 0;
}

/*""FUNC COMMENT""*****************************************************
* ID                : 
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           : 
* Declaration       : void scif_transmit_data( char *Data )
* Description       : SCIF multi-byte data transmission
*                   : 
*                   : 
*                   : 
*                   : 
*                   : 
* Limitation        : 
*                   : 
* Argument          : *Data: Transmit data storage area
* Return Value      : none
* Calling Functions : 
*""FUNC COMMENT END""**************************************************/
void scif_transmit_data( char*Data )
{
  while( *Data )
  {
     while(!(SCIF.SCFSR.BIT.TDFE));/* Wait until transmit data becomes write enabled. */
     SCIF.SCFTDR = *Data;             /* Set the transmit data.   */
     Data++;
     while(!(SCIF.SCFSR.BIT.TEND));/* Wait for transmit complete.*/
     SCIF.SCFSR.BIT.TDFE = 0;
     SCIF.SCFSR.BIT.TEND = 0;
  }
}
```

```

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : void scif_transmit_byte_data( char *Data )
* Description       : SCIF single-byte data transmission
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : *Data: Transmit data storage area
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""***************************************************/
void scif_transmit_data_byte( char *Data )
{
     while(!(SCIF.SCFSR.BIT.TDFE));/* Wait until transmit data becomes write enabled. */
     SCIF.SCFTDR = *Data;              /* Set the transmit data.   */
     while(!(SCIF.SCFSR.BIT.TEND));/* Wait for transmit complete.*/
     SCIF.SCFSR.BIT.TDFE = 0;
     SCIF.SCFSR.BIT.TEND = 0;
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : char scif_recive_data( char  *Data )
* Description       : SCIF data reception
*                   :
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : *Data: Receive data storage area
* Return Value      : -1: Receive error
* Calling Functions :
*""FUNC COMMENT END""***************************************************/
char scif_recive_data( char  *Data )
{
  unsigned char  ReadData, i = 0;
  char  ret_cd = 0;

  for(;;)
  {
     if(( SCIF.SCFSR.BIT.ER  ) ||
        ( SCIF.SCFSR.BIT.BRK ) ||
        ( SCIF.SCFSR.BIT.DR  ))    /* Did an error occur?  */
     {
        ReadData = SCIF.SCFRDR; /* Dummy data read*/
        ret_cd = -1;/* Set that a receive error occurred.   */
        SCIF.SCFSR.WORD &= 0x0000; /* Clear the error.   */
        SCIF.SCLSR.WORD &= 0x0000;
     }
     else if( SCIF.SCFSR.BIT.RDF ) /* Was data received?*/
     {
        *Data = SCIF.SCFRDR; /* Get the data.   */
        SCIF.SCFSR.BIT.RDF = 0; /* Clear the reception sign.   */
        SCIF.SCFSR.BIT.DR  = 0; /* Clear the reception sign.   */
        scif_transmit_data_byte( Data );
        if( *Data == '\n' )  /* Is the acquired data a CR? */
        {
           break;/* Processing has completed.   */
        }
        if( *Data == 0x0d )  /* Is the acquired data a CR? */
        {
           break;/* Processing has completed.   */
        }
        Data++;  /* Set the next data acquisition address. */
        if( ++i == 4 )
        {
           ret_cd = -1;
        }
     }
     if( ret_cd == -1 )
     {
        break;
     }
  }
  return( ret_cd );
}


```

Sample program listing: scif.h

Header file for " scif.h ".

```

#ifndef _SCIF_H
#define _SCIF_H

#include "iodefine.h"
#include "config.h"

#if defined(CONFIG_SCIF0)
#define SCIF   (*(volatile struct st_scif    *)0xFFEA0000) /* SCIF0   Address */
#elif defined(CONFIG_SCIF1)
#define SCIF   (*(volatile struct st_scif    *)0xFFEB0000) /* SCIF1   Address */
#elif defined(CONFIG_SCIF2)
#define SCIF   (*(volatile struct st_scif    *)0xFFEC0000) /* SCIF2   Address */
#elif defined(CONFIG_SCIF3)
#define SCIF   (*(volatile struct st_scif    *)0xFFED0000) /* SCIF3   Address */
#elif defined(CONFIG_SCIF4)
#define SCIF   (*(volatile struct st_scif    *)0xFFEE0000) /* SCIF4   Address */
#elif defined(CONFIG_SCIF5)
#define SCIF   (*(volatile struct st_scif    *)0xFFEF0000) /* SCIF5   Address */
#endif /* CONFIG_SCIFn */

//#define SCBRR_VALUE(bps, clk) ((clk+16*bps)/(16*bps)-1)
#define SCBRR_VALUE(bps, clk) ((clk)/(32*bps)-1)

/* SCFCR */
#define  RTRG1 0
#define  RTRG161
#define  RTRG322
#define  RTRG483
#define  TTRG320
#define  TTRG161
#define  TTRG2 2
#define  TTRG0 3



#endif /* _SCIF_H */
```

Sample program listing: intprg.c

Registers the DU interrupt handler and the DMAC channel 0 interrupt handler as handlers.

```
... Code omitted ...
/* H'620 DMAC0 interrupt */
void INT_DMAC_DMINT0(void)
{
  irq_disable( _DMAC0 );
  dmac0_irq();
}
... Code omitted ...
/* H'D80 DU interrupt */
void INT_DU_DUI(void)
{
  irq_disable( _DU );

  if( DU.DSSR.BIT.TVR & DU.DIER.BIT.TVE)
     du_tvr_irq();
  else if( DU.DSSR.BIT.FRM  & DU.DIER.BIT.FRE)
     du_frm_irq();
  else if( DU.DSSR.BIT.VBK  & DU.DIER.BIT.VBE)
     du_vbk_irq();
  else if( DU.DSSR.BIT.RINT  & DU.DIER.BIT.RIE)
     du_rint_irq();
  else if( DU.DSSR.BIT.HBK  & DU.DIER.BIT.HBE)
     du_hbk_irq();

  irq_enable( _DU );
}
... Code omitted ...
```

Sample program listing: intc.h

Sets the peripheral module interrupt enabled/disabled states and priorities.

```
/*********************************************************************
*
* Device       : SH-4A/SH7785
*
* File Name    : intc.h
*
* Abstract     : INTC .
*
* History      : 1.00  (2010-09-30)  [Hardware Manual Revision : 1.00]
*
* Copyright(c) 2010 Renesas Electronics Corp.
*               And Renesas Solutions Corp.,All Rights Reserved.
*
*********************************************************************/

#ifndef _INTC_H_
#define _INTC_H_

static enum {
  _TMU0,
  _TMU1,
  _TMU2,
  _TMU2_IC,
  _TMU3,
  _TMU4,
  _TMU5,
  _SCIF0,
  _SCIF1,
  _SCIF2,
  _SCIF3,
  _SCIF4,
  _SCIF5,
  _WDT,
  _H_UDI,
  _DMAC0,
  _DMAC1,
  _HAC0,
  _HAC1,
  _PCIC0,
  _PCIC1,
  _PCIC2,
  _PCIC3,
  _PCIC4,
  _PCIC5,
  _SIOF,
  _HSPI,
  _MMCIF,
  _FLCTL,
  _GPIO,
  _SSI0,
  _SSI1,
  _DU,
  _GDTA
}int_num;
```

```

static enum {
  PRI0, PRI1, PRI2, PRI3, PRI4, PRI5, PRI6, PRI7, PRI8, PRI9, PRI10,
  PRI11, PRI12, PRI13, PRI14, PRI15, PRI16, PRI17, PRI18, PRI19, PRI20,
  PRI21, PRI22, PRI23, PRI24, PRI25, PRI26, PRI27, PRI28, PRI29, PRI30,
  PRI31
}priority;

struct intc2_table {
  int pri;       /* Priority */
  int pri_pos;   /* Priority bit position */
  char pri_add;     /* Priority address offset */
  int st_pos;       /* Interrupt bit positions */
};

static struct intc2_table intc_table[] = {
  /* pri,  pri_pos, pri_add, st_pos */
  {  PRI0, 24,      0x00,    0  },    /* TMU0 */
  {  PRI0, 16,      0x00,    0  },    /* TMU1 */
  {  PRI0,  8,      0x00,    0  },    /* TMU2 */
  {  PRI0,  0,      0x00,    0  },    /* TMU2_IC */
  {  PRI0, 24,      0x04,    1  },    /* TMU3 */
  {  PRI0, 16,      0x04,    1  },    /* TMU4 */
  {  PRI0,  8,      0x04,    1  },    /* TMU5 */
  {  PRI0, 24,      0x08,    2  },    /* SCIF0 */
  {  PRI0, 16,      0x08,    3  },    /* SCIF1 */
  {  PRI0,  8,      0x08,    4  },    /* SCIF2 */
  {  PRI0,  0,      0x08,    5  },    /* SCIF3 */
  {  PRI0, 24,      0x0C,    6  },    /* SCIF4 */
  {  PRI0, 16,      0x0C,    7  },    /* SCIF5 */
  {  PRI0,  8,      0x0C,    8  },    /* WDT */
  {  PRI0, 24,      0x10,    9  },    /* H_UDI */
  {  PRI15,16,      0x10,    10 },    /* DMAC0 */
  {  PRI0,  8,      0x10,    11 },    /* DMAC1 */
  {  PRI0, 24,      0x14,    12 },    /* HAC0 */
  {  PRI0, 16,      0x14,    13 },    /* HAC1 */
  {  PRI0,  8,      0x14,    14 },    /* PCI0 */
  {  PRI0,  0,      0x14,    15 },    /* PCI1 */
  {  PRI0, 24,      0x18,    16 },    /* PCI2 */
  {  PRI0, 16,      0x18,    17 },    /* PCI3 */
  {  PRI0,  8,      0x18,    18 },    /* PCI4 */
  {  PRI0,  0,      0x18,    19 },    /* PCI5 */
  {  PRI0, 24,      0x1C,    20 },    /* SIOF */
  {  PRI0, 16,      0x1C,    21 },    /* HSPI */
  {  PRI0,  8,      0x1C,    22 },    /* MMCIF */
  {  PRI0, 24,      0x20,    23 },    /* FLCTL */
  {  PRI0, 16,      0x20,    24 },    /* GPIO */
  {  PRI0,  8,      0x20,    25 },    /* SSI0 */
  {  PRI0,  0,      0x20,    26 },    /* SSI1 */
  {  PRI14,24,      0x24,    27 },    /* DU */
  {  PRI0, 16,      0x24,    28 },    /* GDTA */
};

#define INTC2_OFFSET0xFFD40000
/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : static void irq_enable( int module )
* Description       : Enables the INT2 internal peripheral module
*                   : interrupts and sets their priorities.
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
static void irq_enable( int module )
{
  unsigned long tmp;
  unsigned long address;
  /* Set the priorities. */
  address = INTC2_OFFSET + intc_table[module].pri_add;
  tmp = *(unsigned long *)address;
  tmp |= (intc_table[module].pri << intc_table[module].pri_pos);
  *(unsigned long *)address = tmp;

  /* Clear the interrupt masks. */
  INTC.INT2MSKCLR.LONG = (1 << intc_table[module].st_pos);
}

/*""FUNC COMMENT""*****************************************************
* ID                :
* Outline           : Sample program main() function
*                   : (DU Display)
* Include           :
* Declaration       : static void irq_disable( int module )
* Description       : Disable the INT2 internal peripheral module
*                   : interrupts.
*                   :
*                   :
*                   :
*                   :
* Limitation        :
*                   :
* Argument          : none
* Return Value      : none
* Calling Functions :
*""FUNC COMMENT END""**************************************************/
static void irq_disable( int module )
{
  unsigned long address;

  /* Set the interrupt masks. */
  INTC.INT2MSKR.LONG = (1 << intc_table[module].st_pos);
}

#endif /* _INTC_H_ */
```

Sample program listing: lowlevelinit.inc

Differs from the SH7785 Initialization Example (R01AN0242EJ0101) SH7785 Group Application Note in that minor changes have been made.

- The DBSC2 setting values have been changed to conform to the conditions listed in section 1.3.

```
... Code omitted ...
DBSC2_DBCONF_D:                .equu  H'009A0002
DBSC2_DBTR0_D:                 .equ  H'050D1604
DBSC2_DBTR1_D:                 .equ  H'00040204
DBSC2_DBTR2_D:                 .equ  H'02120708
DBSC2_DBFREQ_D1:               .equ  H'00000000
DBSC2_DBFREQ_D2:               .equ  H'00000100
DBSC2_DBDICODTOCD_D:           .equ  H'00000E07

DBSC2_DBMRCNT_D_EMRS2:         .equ  H'00020000
DBSC2_DBMRCNT_D_EMRS3:         .equ  H'00030000
DBSC2_DBMRCNT_D_EMRS1_1:       .equ  H'00010004
DBSC2_DBMRCNT_D_EMRS1_2:       .equ  H'00010384
DBSC2_DBMRCNT_D_MRS_1:         .equ  H'00000952
DBSC2_DBMRCNT_D_MRS_2:         .equ  H'00000852
... Code omitted ...
```

5. Results of Program Execution

The following occur when this program is run.

- Four planes are displayed.
- The display positions of planes 1 to 3 are changed.
- The plane 4 image is increased or decreased.
- The on/off state of display for each plane is controlled iteratively from the console.

6. Reference Documents

- Software Manual
  SH4-A Software Manual (REJ09B0003)
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Hardware Manual
  SH7785 Group Hardware Manual (REJ09B0261)
  (The latest version can be downloaded from the Renesas Electronics Web site.)

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.01	Mar.17.11	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Renesas Electronics Corporation

http://www.renesas.com

SALES OFFICES

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0