
DSPSTAR Video Module User's Manual

Revision 1.2

Literature Number: NDT050302

2005. 03.



List of Contents

1.	DSPSTAR VIDEO MODULE.....	1-1
1.1.	VM480 COMPONENTS	1-3
1.2.	VIDEO BUFFER IN VM480.....	1-5
1.3.	VIDEO INPUT	1-6
1.3.1.	EMIF (EXTERNAL MEMORY INTERFACE) SETTING	1-7
1.4.	EXAMPLE PROGRAMS.....	1-8

List of Figures

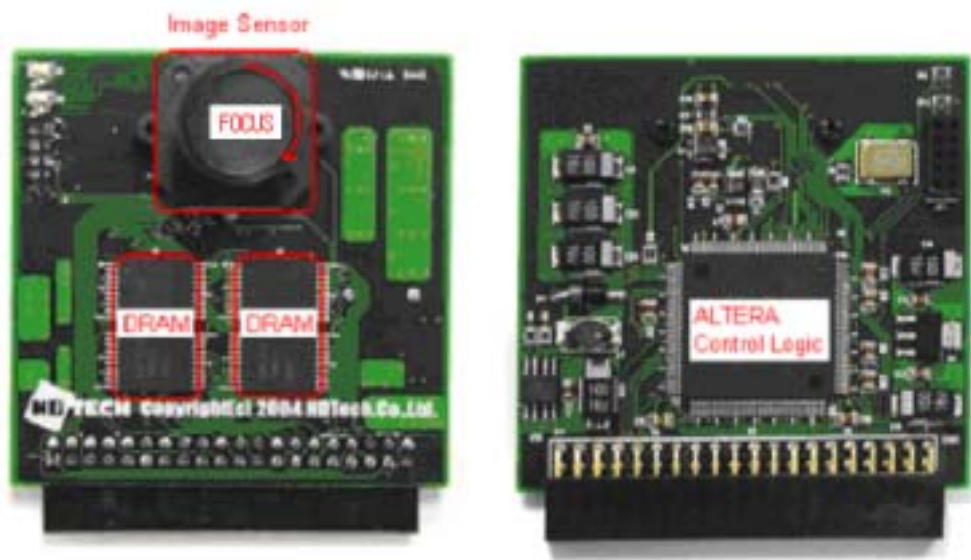
FIGURE 1.	REAL IMAGES OF VM480.....	1-1
FIGURE 2.	INSTALLATION OF VM480 INTO EXPANSION CONNECTOR	1-2
FIGURE 3.	INSTALLATION OF VM480 WITH PROCESSOR MODULE.....	1-2
FIGURE 4.	SCREEN BUFFER STRUCTURE.....	1-5
FIGURE 5.	IMAGE OF 640X480 OBTAINED FROM THE VM480	1-7

List of Tables

TABLE 1.	CPU EXPANSION PORT SIGNALS.....	1-4
TABLE 2.	CPU EXPANSION PORT PIN DESCRIPTION.....	1-4
TABLE 3.	IMAGE DATA ORGANIZATION.....	1-7
TABLE 4.	LIST OF EXAMPLE PROGRAMS	1-8

1. DSPSTAR Video Module

The DSPSTAR Video Module VM480 shown in Figure 1 is a video capturing board which is an add-on module to DSPSTAR Personal hardware board (DS6xxxP) through the expansion port connector. The VM480 houses CCD camera sensor, through which 30 Hz video signal with 640x480 image resolution is entered as 8-bit byte serial data in YCrCb 4:2:2 format. As shown in the front view, CCD sensor focus can be adjusted by turning it clockwise or counter clockwise.



(a) Front View

(a) Rear View

Figure 1. Real images of VM480

When placing the video module into the expansion port, make sure that the camera directs outward as shown in Figure 2.

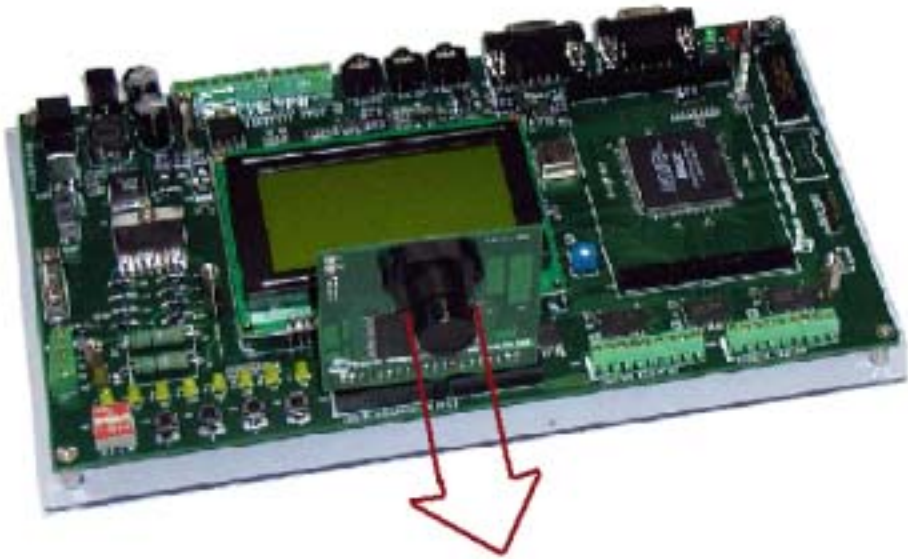


Figure 2. Installation of VM480 into Expansion Connector

The VM480 can also be used with DSPSTAR Processor Module as shown in Figure 3.

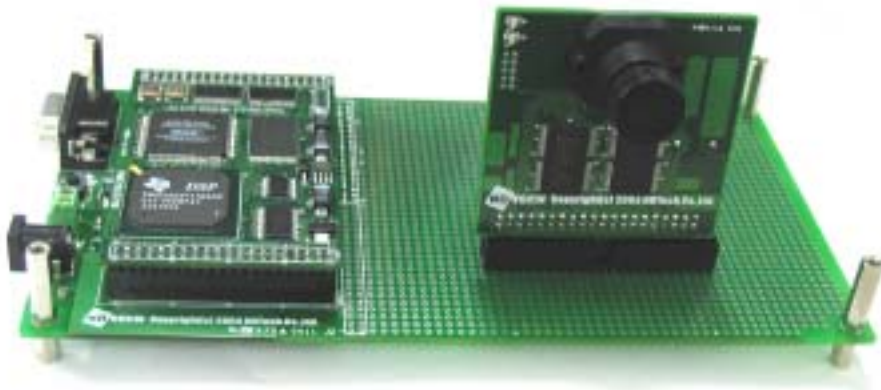


Figure 3. Installation of VM480 with Processor Module

1.1. VM480 Components

A 60 Hz color video is entered to the VM480 as eight bit data through the camera image sensor as in Figure 1. The camera sensor OV7640 of OMNIVISION TECHNOLOGIES INC. is used in the video module, which provides various video format such as 60 Hz YCrCb 4:2:2, RGB 4:2:2, RGB raw data 8 bit format, and etc. However, the VM480 product supports YCrCb 4:2:2 8 bit mode.

The VM480 includes two DRAM chips T224162B of TM Technology Inc. to span a total memory of 1 Mbyte. Each DRAM chip is 256K x 16 bit in size.

Interface connector of VM480, which is plugged into expansion port of DSPSTAR personal hardware includes 16 bit address bus, 16 bit data bus, chip enable, read enable, write enable, 5V output, and etc. The detail pin descriptions appear in the following two tables.

Table 1. CPU Expansion Port Signals

Pin No.	Pin Name	Pin No.	Pin Name
1	VCC	2	GND
3	CD0	4	CD1
5	CD2	6	CD3
7	CD4	8	CD5
9	CD6	10	CD7
11	CD8	12	CD9
13	CD10	14	CD11
15	CD12	16	CD13
17	CD14	18	CD15
19	CA0	20	CA1
21	CA2	22	CA3
23	CA4	24	CA5
25	CA6	26	CA7
27	CA8	28	CA9
29	CA10	30	CA11
31	CA12	32	CA13
33	CA14	34	CA15
35		36	/URE
37	/UCE	38	/UWE
39	GND	40	VCC

Table 2. CPU Expansion Port Pin Description

Pin Name	Description
VCC	VCC (5V)
GND	Ground
CD[15..0]	Data Bus
CA[15..0]	Address Bus
/UCE	Chip Select Signal
/URE	Read Enable
/UWE	Write Enable

1.2. Video Buffer in VM480

Image data of size 640x480 serially supplied from the CMOS image sensor is collected and stored in screen buffer. When stored, 8-bit sensor output is assembled to 16-bit data. A total of 640x480x2 bytes memory is needed to hold image data corresponding to image size 640x480. But, a total of 1 Mbyte memory is present in the video module, which requires 19 bit word address. Since CPU expansion port carries only 16-bit address bus, 5-bit page address (VPAGE) and 14-bit offset (VBUFF) are used for accessing 1-Mbyte screen buffer, as in Figure 4.

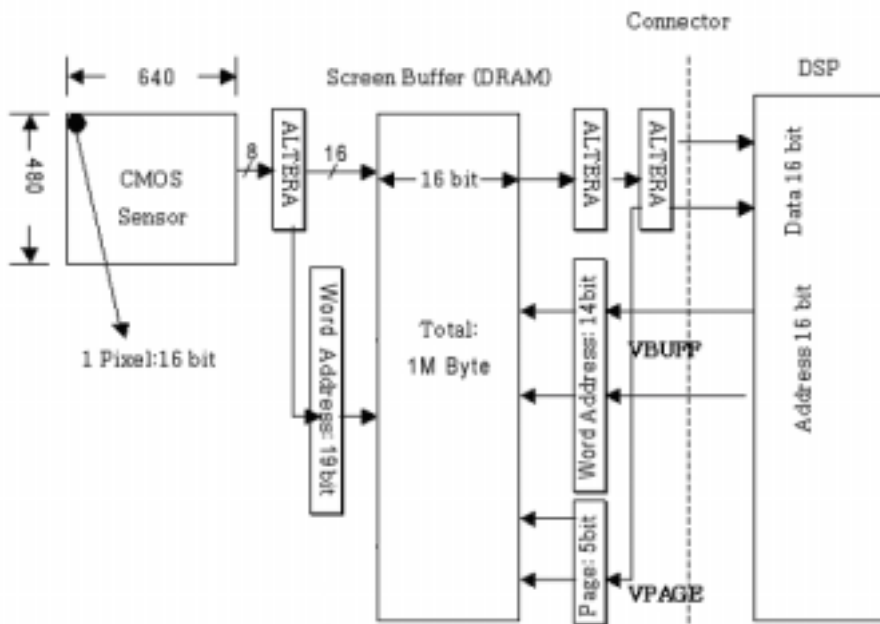


Figure 4. Screen Buffer Structure

1.3. Video Input

Video buffer takes an input of 640x480 image signal in 4:2:2 YUV format. The total 1KByte video buffer consists of 16 64-kbyte pages.

The following C function reads the screen buffer to get 640x480 image data and convert it to the image size as given in as two function arguments X and Y. The local integer pointer variable VBUFF indicates the start of screen buffer, while the VPAGE indicates address offset.

```
void Getnzm(short *buffer,int X, int Y)
{
    int i,j,k,l,d,s,*VBUFF,*VPAGE;

    VBUFF = (int *) (CE1BASE+0x003e0000); /* Video Buffer(DRAM) start address */
    VPAGE = (int *) (CE1BASE+0x003b0000); /* Video Buffer Page Register */

    if (X>640 || Y>480) {
        printf("Notice: Image Size can be upto 640x480.\n"); return;}
    k = 640/X;
    l = 480/Y;
    d=0;

    for(i=0;i<Y;i++)
    {
        for (j=0;j<X/2;j++)
        {
            s = i*640 + j*k*2;
            while(*VPAGE & 0x0000); // check dram data valid
            *VPAGE = (s >> 14);
            buffer[d++] = VBUFF[s&0x3fff];

            s++;
            while(*VPAGE & 0x0000); // check dram data valid
            *VPAGE = (s >> 14);
            buffer[d++] = VBUFF[s&0x3fff];
        }
    }
}
```

After execution of the function Getnzm, the data stored in the array variable buffer is organized as in Table 3. The first data buffer[0] stores Y_0 and V_{01} , and the second data buffer[1] stores Y_1 and U_{01} , where Y_i indicates luminance information at the i th pixel, and $U_{i(i+1)}$ and $V_{i(i+1)}$ indicate chrominance information of the i th & $(i+1)$ th pixels. In this way, the zeroth pixel is determined by buffer[0] and one lower byte of buffer[1], and the first pixel is determined by low byte of buffer[0] and buffer[1]. In this way, the last pixel

can be determined from low byte of buffer[640x480-2] and buffer[640x480-1].

Table 3. Image Data organization

Buffer data	High Byte	Low Byte
buffer[0]	Y0	V01
buffer[1]	Y1	U01
buffer[2]	Y2	V23
buffer[3]	Y3	U23
buffer[4]	Y4	V45
buffer[5]	Y5	U45

1.3.1. EMIF (External Memory InterFace) Setting

EMIF initialization is needed since screen buffer in the VM480 is allocated as asynchronous memory in CE1 space. Figure 5 shows EMIF initialization routine. Note that EMIF initialization (CE0 and SDRAM) is also included for SDRAM since memory address of 0x80000000 for the storage of image data is in external SDRAM.

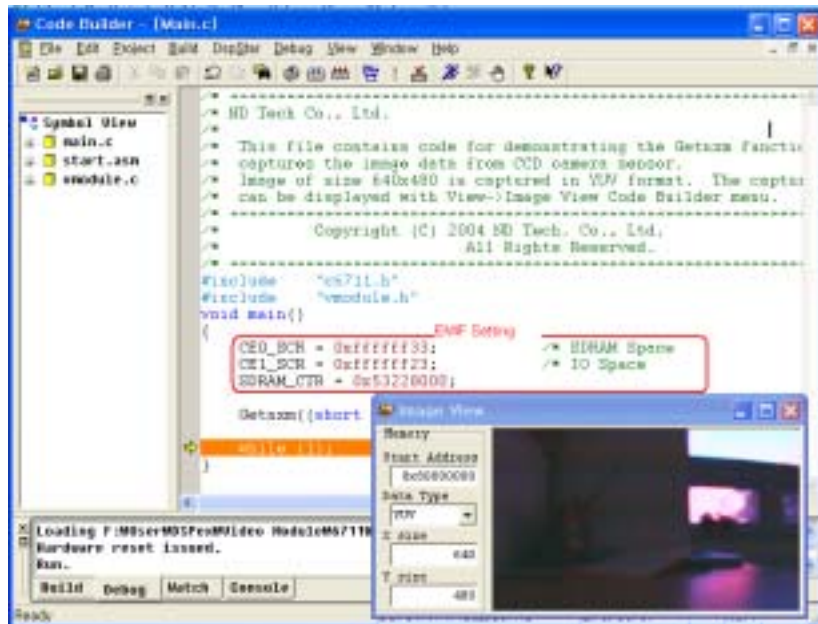


Figure 5. Image of 640x480 obtained from the VM480

1.4. Example Programs

Shown in Table 4 is the list of example programs accompanied in the product CD.

Table 4. List of Example Programs

Application Area	Example Projects	Descriptions
Compression Decompression	dcidct	Forward Discrete Cosine Transform Inverse Discrete Cosine Transform
	img_dct_8x8	Forward Discrete Cosine Transform Inverse Discrete Cosine Transform
	img_quantize	Matrix Quantization with Rounding
	img_sad_8x8	Sum of Absolute Differences on Single 8x8 block
	img_mad_8x8	8x8 Minimum Absolute Difference
	zigzag	Zigzag Reordering
Image Analysis	colorbar	Vertical Color Bar
	img_boundary	Boundary Structural Operator
	img_histogram	Histogram Computation
	img_thr_gt2max	Thresholding – Clamp to 255
	img_thr_gt2thr	Thresholding – Clip above threshold
	img_thr_le2min	Thresholding – Clamp to zero
Picture Filtering Format Conversions	img_thr_le2thr	Thresholding – Clip under threshold
	img_conv_3x3	3x3 Convolution
	img_corr_3x3	3x3 Correlation
	img_corr_gen	Generalized Correlation
	img_median_3x3	3x3 Median Filter
	img_pix_exp_sat	Pixel Expand & Saturation
	vmodule	Image Capture from Image Sensor
	yuv2rgb	YUV->RGB Conversion
yuv2u	Extraction of individual y, u, v components from YUV signal	