

# Interfacing a CMOS Image Sensor to a D.Module

Doc #1.0

## Application Note

**D. SignT**

Digital Signalprocessing Technology

Kane Computing Ltd

7 Theatre Court, London Road,  
Northwich, Cheshire, CW9 5HB, UK

phone +44 (0)1606 351006

fax +44 (0)1606 351007/8

email [sales@kanecomputing.com](mailto:sales@kanecomputing.com)

web [www.kanecomputing.com](http://www.kanecomputing.com)

## **Revision History**

1.0 April 2003

---

<b>1 ABSTRACT .....</b>	<b>3</b>
<b>2 CMOS SENSOR BASICS .....</b>	<b>4</b>
<b>3 DSP SELECTION.....</b>	<b>6</b>
<b>4 CAMERA INTERFACE.....</b>	<b>8</b>
4.1 Pixel Data Interface .....	10
4.2 Control Interface .....	11
4.3 Image Sensor Initialization .....	14
4.4 DSP Initialization .....	17
<b>5 NETWORKING INTERFACE.....</b>	<b>20</b>
<b>6 CONCLUSION .....</b>	<b>21</b>

## **1 Abstract**

Nowadays image processing systems are mostly based on PC, CompactPCI, or VME-Bus systems. They are typically made up from analog cameras with CCD sensors, a frame grabber, and the actual image processing processor.

For many new applications this architecture however cannot be used due to size, power consumption, and cost constraints. This is particularly true for biometric systems (e.g. face recognition) and automotive applications (driver assistance systems). These new developments can only be marketed successfully if an extremely compact, powerful, and reasonably priced hardware is available.

CMOS image sensors have become a popular choice for machine vision applications requiring low power, small size and medium image resolution. They integrate the analog and digital circuit parts on a single chip, providing a direct interface to a processor's data bus and hence significantly reducing component count, power consumption and board real-estate compared to the 'classical' solutions.

This application note describes an image processing system based on the Omnivision OV6120 sensor and the Texas Instruments TMS320C6203B digital signal processor. The D.Module.C6203 forms the hardware basis since it readily includes all additional components required for stand-alone operation. The system is rounded off with an Ethernet controller to add networking capabilities which are frequently required for communications, remote control, or database access.

## 2 CMOS Sensor Basics

A CMOS image sensor uses photodiodes or photogate transistors as the light detecting element. In contrast to CCD sensors, each photo element is individually addressable in a row and column matrix, similar to a DRAM structure, which allows partial image readout (windowing, area of interest). Windowing is especially helpful in motion detection and object tracking algorithms. It reduces intermediate storage memory requirements and allows higher frame rates.

Image quality of modern CMOS sensors is almost comparable to CCD devices, with slightly higher noise levels, mainly caused by the integration of analog and digital circuitry on the same chip. CMOS however does not suffer from blooming and smearing effects that are caused by charge leakage in CCD devices.

CMOS sensors can be built with logarithmic characteristics, achieving a dynamic range up to 100 dB and more, which is essential for many 'outdoor' applications like object detection and tracking in a vehicle.

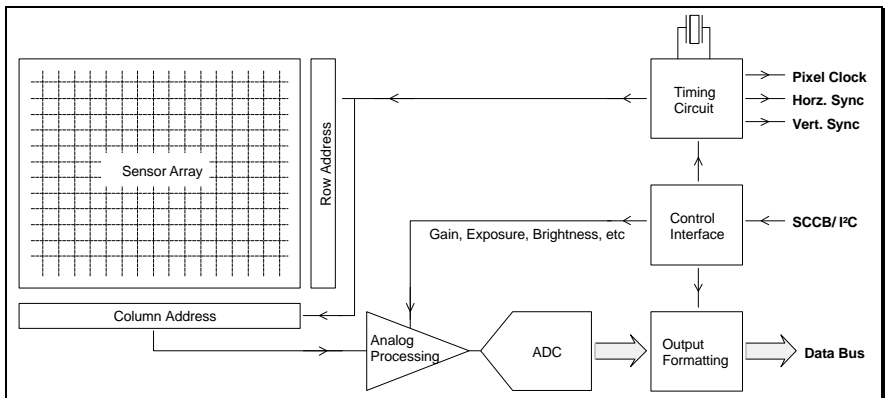


Figure 2-1 CMOS Image Sensor

Color sensors include a Bayer color filter on top of the inherently monochrome sensor array. The analog pre-processor is complemented with gamma correction circuits and with a RGB to YCrCb matrix. An additional ADC converts the CrCb color difference signal.

Image data is transmitted via a parallel bus interface, qualified by a pixel clock and horizontal and vertical synchronization pulses.

A CMOS sensor is typically controlled via a SCCB (I<sup>2</sup>C) bus, a serial two-wire interface. Depending on the sensor the host processor can control exposure, gain, brightness, contrast, gamma correction, windowing, frame rate and output data format.

Besides the described progressive-scan architecture which outputs image data sequentially line by line, random-access sensors exist which are connected to the host via row and column address lines, similar to a DRAM device. These sensors provide fully random access to any pixel within the array and are particularly useful for very fast object tracking systems.

CMOS sensors exist in various resolutions. Typical are CIF (352x288 pixels), VGA (640x480), SVGA (800x600) and XGA (1024x768) devices, but also a number of special formats are available. Low resolution sensors (128x128) allow hundreds of frames per second and are used for very fast motion detection, whereas material inspection applications call for highest resolution up to several Megapixels.

The OV6120 monochrome sensor was chosen for this application for no particular reason except the availability of an evaluation board, readily populated with a lens holder and some passive components. Most CMOS image sensors do have an identical architecture and should work with minor adjustments, mainly concerning the software setup and sensor configuration.

### **3 DSP Selection**

Image processing is quite a challenge for every processor. Depending on the resolution hundreds of kilobytes of data must be processed in a few milliseconds. Many algorithms frequently access not only the pixel actually being processed but also the neighborhood pixels. A simple filter based on a 3x3 matrix requires 9 memory accesses for each pixel being filtered. To implement this filter in real-time (25 frames per second) VGA resolution, a memory bandwidth of 66 Mbytes per second is required. More complex image processing tasks are possible only if the processor architecture provides sufficient memory bandwidth and supports parallel access and processing of multiple pixels simultaneously. External memory devices are limited in speed, hence large internal memories and/or a multi-path cache architecture is essential for demanding applications.

Because processing itself often consumes most of the available memory bus bandwidth an additional data path for image acquisition is helpful to prevent bottlenecks. Since we want to avoid costly additional buffering (FIFOs) we must be able to read and store each pixel from the camera in time. At VGA resolution and 25 frames per second the image sensor will output a new pixel every 70 nano-seconds. This speed obviously prohibits interrupt-driven data acquisition. A DMA (direct memory access) controller is implicitly required. The DMA controller must be synchronized to the image sensor's pixel qualifier clock, i.e. each clock pulse will trigger a DMA transfer.

The separate data bus for image acquisition is especially important if real-time continuous processing is required as in object tracking algorithms. All real-world applications need some kind of communications to transmit their results to other process control devices. Most communication devices do have a slow bus interface: a typical UART requires an access time of 100 nsecs or more, the same applies to network controllers (Ethernet or CAN). If the DSP bus interface is blocked by reading or writing such a communication device it is not possible to acquire an image from the CMOS sensor in real-time without data loss. If no separate bus for image acquisition exists a FIFO memory is needed to buffer the image data, or communications must be limited to the inter-frame gap.

The Texas Instruments C6000 processors fulfill all these requirements and are an excellent choice for the described system. Especially the fixed-point devices C6203 and C6416 are to mention. Should floating-point dynamic be required, the C6713 may be used. This processor lacks the separate bus for image acquisition, but on the D.Module.C6713 the processor's Host Port Interface, in conjunction with the user-programmable CPLD, can be used as an additional expansion port.

Although these processors are extremely fast, straightforward C-coding will not always provide the desired performance results. Today's C-Compiler are highly efficient and optimized, but elaborate Assembler coding, fully exploiting a processor's capabilities, will still deliver significantly higher performance, especially on image processing algorithms. Therefore a library of optimized basic image processing functions is of invaluable importance to speed-up development and reduce hardware requirements. Texas Instruments provides such an optimized image processing library for their processors. Simultaneous reading and intermediate register storage of multiple pixels reduce memory bandwidth requirements. Optimized loop constructs and data read-ahead techniques guarantee minimum overhead. On the C6203 for example, a Sobel-filter for edge detection completes in only one millisecond for a complete CIF-sized image.

This application note is based on the D.Module.C6203. Besides the processor, power supply, and clock circuits, the D.Module provides additional features required for a complete standalone system: Flash Memory to boot-load the application program and store parameters, 16 Mbytes of SDRAM for intermediate image storage, two high-speed UARTs for communications, and a user-programmable CPLD to implement the SCCB / I<sup>2</sup>C bus interface. Hence, the D.Module.C6203 provides the versatility of a typical micro-controller, combined with the impressive 2400 MIPS DSP processing power.

## 4 Camera Interface

This chapter describes the hardware and software interface of the CMOS image sensor.

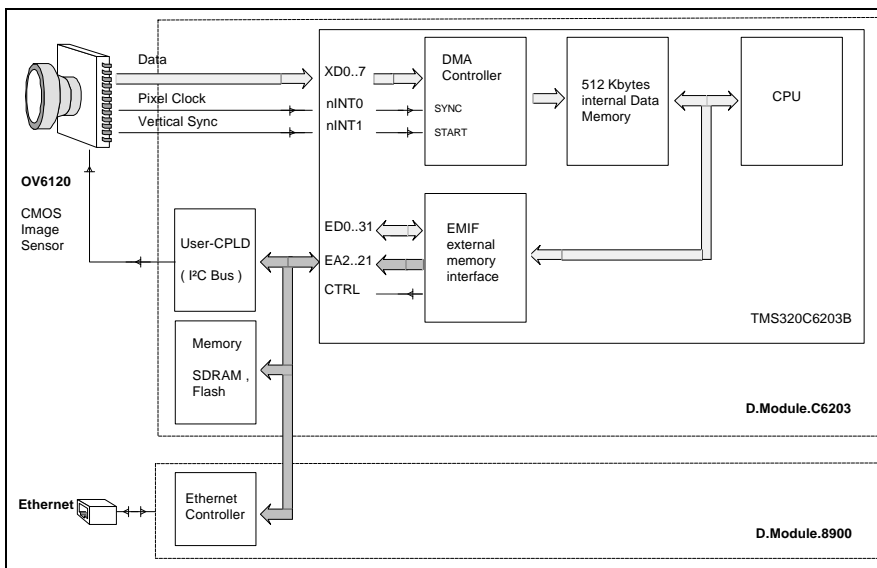


Figure 4-1 Image Processing System

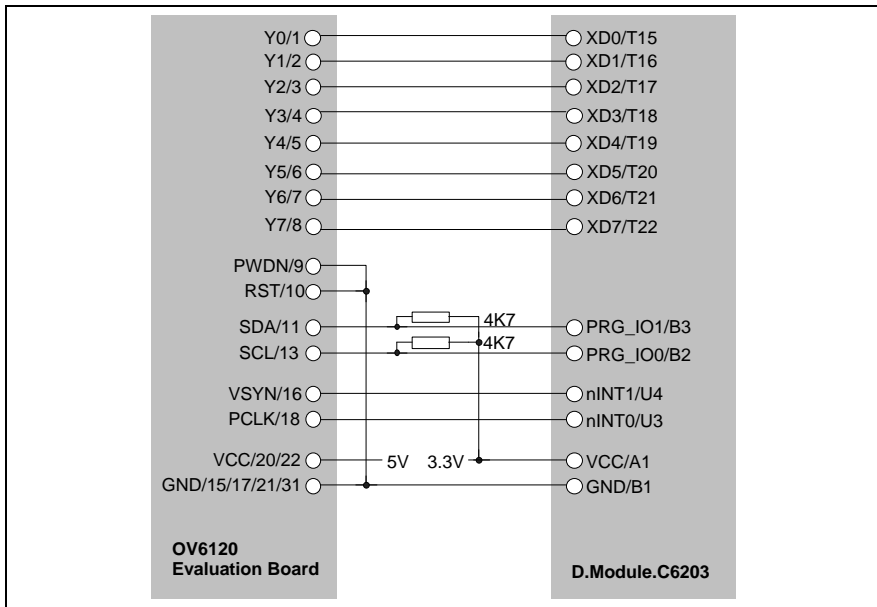


Figure 4-2 OV6120 to D.Module.C6203 Interface Wiring

## 4.1 Pixel Data Interface

As mentioned before, a progressive scan CMOS image sensor will output the image data frame-based, qualified by horizontal and vertical synchronization signals. Each pixel itself is qualified by the pixel qualifier clock.

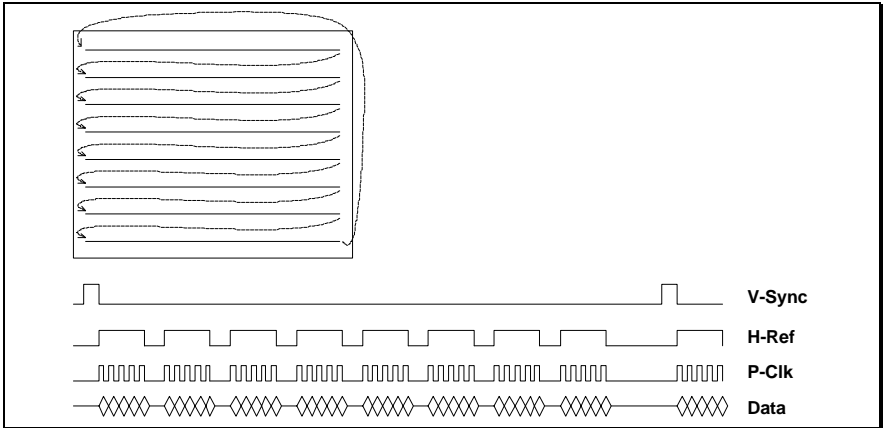


Figure 4-3 Progressive Scan Data Readout

The figure above shows a typical timing of a progressive-scan CMOS image sensor: starting on the upper left image (or window) corner, a vertical sync pulse (V-Sync) is generated which qualifies the start of a new image. The sensor now reads one row of the image and outputs the pixel data sequentially from left to right with each P-Clk (pixel qualifier clock). The H-Ref Signal is active while valid pixels are output. H-Ref is de-asserted while the next row is being addressed. If all image rows are output the sensor restarts with the next image.

To interface such a sensor to the DSP we only need to connect the sensor data bus to the DSP bus, connect the V-Sync signal to an interrupt input, and connect the P-Clk to a DMA-Request input.

H-Ref is not required if the sensor can be programmed to qualify valid pixels only. Otherwise (if only a continuous P-Clk is available) the DMA request signal must be generated externally by gating P-Clk with H-Ref. The V-Sync interrupt is used to set-up and start the DMA-Controller, the remaining image acquisition takes place in background.

If the DSP lacks an expansion port which is solely used to connect the image sensor we have to share the common bus interface with memories and communication peripherals. In this case the camera must provide a 3-stateable data bus, or an external 3-stateable bus driver must be used. The camera's data bus (rsp. the bus driver) is enabled by an address decoder, similar to any other memory-mapped device.

The OV6120 (like most other available sensors) and TMS320C6203 can be connected without additional glue logic: The data bus is connected to the DSP Expansion Bus XD0..7, V-Sync and P-Clk are both connected to external interrupt inputs. The C6203 allows to synchronize a DMA transfer with an external interrupt, i.e. the interrupt input in this case in actuality acts as a DMA request.

The described system relies on the DMA being able to read every pixel from the camera interface in time. The timing constraints of the DMA controller must carefully be checked to avoid losing pixels. If image data is DMAed into external memory, the external memory interface timing needs to be considered too. With the current high-speed DSPs it is possible to acquire SVGA sized images at full 25 frames per second. For higher frame rates the resolution must be decreased, higher resolution requires to lower the frame rate.

## 4.2 Control Interface

As most CMOS image sensors the OV6120 is controlled and configured via a SCCB (serial camera control bus) interface, which is identical to the standard I<sup>2</sup>C bus interface. This two-wire interface uses a clock (SCL) and a data (SDA) signal. Both SCL and SDA are open drain signals. A low level is actively driven, a high level is passively generated by a pull-up resistor. This allows any device on the I<sup>2</sup>C bus to overwrite a high level, which is important to request wait states and for multi-master bus arbitration. Multiple peripherals can be connected to an I<sup>2</sup>C bus, which are distinguished by their individual addresses. Typically the upper address bits are hard-coded into the peripheral device, while the lower bits are pin-programmable. I<sup>2</sup>C uses a master-slave architecture: the master generates the SCL clock and addresses the slave by sending it's address and a control bit which indicates if data should be written to the slave or read from the slave. If

data is read, the master generates SCL and the slave drives SDA. Data on SDA is valid and remains unchanged while SCL is high, except for the START and STOP conditions.

The maximum transfer speed depends on the slave device. 100 kHz SCL clock is common, but most devices support a 400 kHz high-speed mode too.

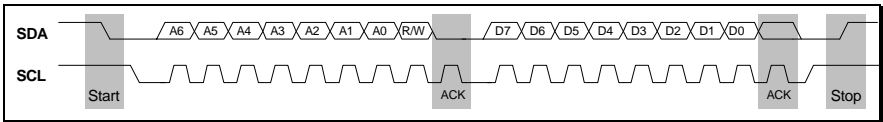


Figure 4-4 SCCB / I<sup>2</sup>C Transmission

The figure above shows a typical I<sup>2</sup>C bus transfer. The master generates a START condition and sends the slave device address (7 bits) followed by the R/W direction bit. The slave acknowledges this by pulling SDA low during the ACK bit time. Then the master continues generating SCL clock pulses, and either the slave or the master transmit a data byte. If the slave device has multiple registers, like a CMOS image sensor, this first byte typically specifies the slave register (sub-address) to access. After the last transfer of a sequence is completed the master terminates the transfer by generating a STOP sequence. If the number of data bytes in a transfer is undetermined a 'not acknowledge' on the last byte is often used to signal no more data to follow.

A slave can request wait states by pulling SCL low until ready. A master always has to check this condition and wait until SCL is released and pulled high again. I<sup>2</sup>C also provides arbitration mechanisms for multi-master mode and 11 bit extended address identifier - but these features are not required for the application described in this document.

An I<sup>2</sup>C bus interface can always be emulated using bit-programmable I/O port pins. On the C6000 DSPs, unused McBSP (serial port) pins can be used for example. To mimic the open-drain circuit the pin is configured as an output and a '0' is written to it to drive an active low-level. To generate the passive high-level the pin is configured as an input. One of the DSP timers is used to periodically generate interrupts at four times the I<sup>2</sup>C bus clock rate.

In the first bit cycle SCL is driven low and the SDA pin is either driven low if the master writes a 0-bit, or configured as an input if the master writes a 1-bit or reads a bit from the slave.

In the second cycle SDA remains unchanged and SCL is configured to input mode to generate the passive high-level.

In the third cycle the master checks the state of SCL. If found high, the master reads the slave data bit or leaves SDA unchanged on write transfers.

In the last cycle finally SCL is driven low again.

Interrupt load is quite high in this scenario. For a 400 kHz I<sup>2</sup>C bus clock, 1.6 million interrupts are generated per second. Also the interrupt code is quite complex which results in a large overhead for context save and restore. This is acceptable if the SCCB / I<sup>2</sup>C bus is mainly used during initialization and not during image acquisition and processing. If however control data is frequently send, e.g. to adjust gain and exposure to adapt to varying light conditions, a dedicated I<sup>2</sup>C bus interface should be used. I<sup>2</sup>C controller devices are readily available, but on the D.Module a more simple solution exists: the user-programmable CPLD is used to implement an I<sup>2</sup>C master interface, suitable programming files exist for all modules. This solution cuts the DSP interrupt load by 36 to one interrupt per byte.

### 4.3 Image Sensor Initialization

The OV6120 provides multiple configuration options which must be set-up to match the described hardware interface. The same applies to the C6203. The following initialization code example assumes these functions for I<sup>2</sup>C communications:

```
int i2c_send (unsigned char state, char data); // send one byte, return ACK bit
char i2c_receive (unsigned char state);      // receive one byte
```

Parameter state is a bit-field that defines special I<sup>2</sup>C communication states: START, STOP and ACK. If the START bit is set, a START condition is generated prior to sending the data byte. If the STOP bit is set, a STOP condition is generated following the data transmission. ACK is used for receive operations only. If this bit is set, the received data is acknowledged, otherwise the data is not acknowledged.

These functions must be implemented according to the I<sup>2</sup>C hardware interface actually used (external I<sup>2</sup>C controller, User-CPLD implementation on D.Module, or bit-I/O emulation).

The following code shows a typical initialization sequence. On each I<sup>2</sup>C transmission, the ACK (acknowledge) bit as returned by i2c\_send() should be checked. This has been omitted for clarity in the sample code.

1. Make sure the OV6120 is available, read it's manufacturer ID:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (STOP, 0x1C); // set sub-address to 0x1C = manufacturer ID reg.
i2c_send (START, 0xC1); // send device address, R/nW = 1
id_hi = i2c_receive (ACK); // read high byte and acknowledge
id_lo = i2c_receive (STOP); // read low byte, do not ack, generate STOP
id = (id_hi << 8) || id_lo;
if (id != 0x7FA2) exit (1); // exit program if manufacturer ID is not 0x7FA2
```

2. Generate an OV6120 software reset by writing 0xA4 to the Common Control A register at sub-address 0x12:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x12);        // set sub-address to 0x12
i2c_send (STOP, 0xA4); // write 0xA4, generate STOP
```

3. wait approx. 100 msecs

```
delay (100);
```

4. Configure pixel clock to qualify valid data only by writing 0x40 to Common Control L register at sub-address 0x39:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x39);        // set sub-address to 0x39
i2c_send (STOP, 0x40); // write 0x40, generate STOP
```

5. Enable automatic Black Level Calibration by writing 0xB2 to the Common Control F register at sub-address 0x26:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x26);        // set sub-address to 0x26
i2c_send (STOP, 0xB2); // write 0xB2, generate STOP
```

6. Set Clock Pre-Scaler to the desired frame rate (maximum) by writing 0 to the Clock Rate Control register at sub-address 0x11:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x11);        // set sub-address to 0x11
i2c_send (STOP, 0x00); // write 0, generate STOP
```

By default the OV6120 uses auto-exposure mode. If a user-defined exposure level should be set, write the desired value to the AEC register at sub-address 0x10. The default value following reset is 0x94:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x10);        // set sub-address to 0x10
i2c_send (STOP, exp);   // write exposure value, generate STOP
```

To enable or disable auto exposure mode write a 0 (auto exposure off) or a 1 (auto exposure on) to the Common Control B register at sub-address 0x13:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x13);        // set sub-address to 0x13
i2c_send (STOP, on_off); // write auto-exposure on/off, generate STOP
```

To control brightness write the desired value to the Brightness Control Register at sub-address 0x06. The default level following reset is 0x80:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x06);        // set sub-address to 0x06
i2c_send (STOP, brt);   // write brightness level, generate STOP
```

To define a window (area of interest) write the coordinates of the window (x\_left, x\_right, y\_top, y\_bot) to the HREFStart, HREFEnd, VREFStart, and VREFEnd Registers at sub-addresses 0x17 .. 0x1A. The default coordinates for a full screen are 56, 234, 3, and 146. Each increment / decrement corresponds to two pixels:

```
i2c_send (START, 0xC0); // generate START, send device address, R/nW = 0
i2c_send (0x17);        // set sub-address to 0x17
i2c_send (x_left);      // write window coordinates, generate STOP on last transfer
i2c_send (x_right);
i2c_send (y_top);
i2c_send (STOP, y_bot);
```

## 4.4 DSP Initialization

On the DSP side the Expansion Bus must be configured to match the OV6120 pixel data port timing: 1 setup cycle, 3 read strobe cycles, and 1 hold cycle on the 300 MHz C6203B processor.

```
* (volatile unsigned int *) 0x1880008 = 0x10D10321;
```

The interrupts for frame start and DMA synchronization (VSYN and PCLK) must be set to the desired polarity. The Vertical Sync Interrupt, which starts a new acquisition, must be active on a 0-to-1 transition. A new pixel is output on a 0-to-1 transition on PCLK. This matches the default configuration of the C6203. On the D.Module.C6203 interrupt polarity is reversed by external inverters and the DSP Interrupt Polarity register at address 0x19C0008 must be set appropriately (polarity of these signals can also be reversed in the OV6120):

```
* (volatile unsigned int *) 0x19C0008 = 3;
```

The VSYN (EXT\_INT5 = nINT1 on the D.Module) interrupt service function is used to setup and start the DMA transfer of an image. We use DMA channel 0 in this example. DMA is synchronized by the pixel clock connected to the DSP's external interrupt 4 (nINT0 on the D.Module.C6203). Since an image is larger than the maximum 64K words a DMA is able to transfer in a single frame, we use a block transfer consisting of multiple frames: The element count is set to the number of pixels in an image row, the number of frames is set to number of rows of the image.

1. Configure the DMA Primary Control Register. Read synchronization is set to EXT\_INT4, use Global Count Register A for element count reload, element size is 8 bits, destination address is incremented on each transfer, the source address remains unchanged:

```
* (volatile unsigned int *) 0x1840000 = 0x10240;
```

2. Clear all pending events in the DMA Secondary Control Register, enable interrupt if transfer completed:

```
* (volatile unsigned int *) 0x1840008 = 0x2080;
```

3. Set the DMA Source Address to the Expansion Bus XCE0:  
\* (volatile unsigned int \*) 0x1840010 = 0x40000000;
4. Set the DMA Destination Address to the image buffer start address:  
\* (volatile unsigned int \*) 0x1840018 = (unsigned int) image\_buffer;
5. Set the DMA Transfer Counter and the Global Counter Reload A register to the image size: write image rows to bit 31..16, write image columns to bit 15..0:  
\* (volatile unsigned int \*) 0x1840020 =  
\* (volatile unsigned int \*) 0x1840028 = (rows << 16) + cols;
6. Start the DMA:  
\* (volatile unsigned int \*) 0x1840000 |= 1;

After enabling interrupts, the first image will be acquired. To wait for completion we can either use an interrupt service function for the DMA channel 0 transfer complete interrupt, or poll for this condition in the DSP Interrupt Flag Register. If polling is used, the interrupt flag must be manually cleared by writing to the Interrupt Clear Register.

A common technique for continuous image acquisition and processing, as required by object tracking applications, uses a ping-pong buffer scheme to acquire an image in background while processing the previous image. Since the current acquisition buffer pointer (acq\_buf) is used in the VSYN interrupt service function as the DMA destination address, at least this variable must be declared global. For fastest processing the image buffers should be located in on-chip data memory. The #pragma DATA\_SECTION is used on the C6000 to define a 'special' data section, that is then mapped to internal data memory in the linker command file. Also most functions of the image library provided by TI require image buffers to be word-aligned, which is achieved by #pragma DATA\_ALIGN.

```
#pragma DATA_SECTION (img_buf1, ".idram");
#pragma DATA_SECTION (img_buf2, ".idram");
#pragma DATA_ALIGN (img_buf1, 32);
#pragma DATA_ALIGN (img_buf2, 32);

unsigned char img_buf1 [ROWS*COLS];
unsigned char img_buf2 [ROWS*COLS];
unsigned char * acq_buf = img_buf1;
unsigned char * proc_buf = img_buf2;
unsigned char * temp_ptr;

enable_vsyn_interrupt();      // start interrupts
for (;;)                      // repeat forever
{
    while (! dma0_complete() ) ; // wait for acquisition to complete
    temp_ptr = acq_buf;        // swap buffers
    acq_buf = proc_buf;
    proc_buf = temp_ptr;
    image_proc (proc_buf);     // process image
}
```

In the above example it is assumed that image processing has completed before the next image is available. If this is not guaranteed interrupts must be disabled until processing is done. Otherwise the next VSYN interrupt will start acquisition into the same buffer that is still being processed.

## **5 Networking Interface**

Many image processing applications do require a fast interface to additional system components to communicate their results and/or access to a central database. Biometric systems for face recognition will extract the characteristic vectors of a face and query a database for a matching entry. This is typically done via an Ethernet network interface. A driver assistance system will need to communicate with the car's security control and display systems via CAN bus. In industrial control applications a network connection allows remote control and configuration, e.g. sending new sets of parameters depending on the actually processed materials.

A system quite similar to the one described in this application note is used in printing industry to verify the correct sorting of pages before final binding. The image processing system calculates the characteristic features of each page and receives it's nominal values from a central database via Ethernet.

An Ethernet connection is also useful during algorithm development and testing. Many elaborate tools like MatLab® and LabView® provide a TCP/IP interface to insert data into their processing chain or to output stimuli. Image processing chains developed with these tools can be implemented step-by-step on the DSP, and verified on-the-fly.

For the D.Module.8900 Ethernet Controller Daughter Card a TCP/IP protocol stack has been developed that does not require a RTOS (real-time operating system) and minimizes resource requirements on the DSP. This allows to integrate networking functionality without interfering or degrading the DSP's actual task: image processing.

## **6 Conclusion**

Combining a C6000 series DSP with a CMOS image sensor is straightforward without the need for external glue logic. This results in a high-performance, compact, low-cost, and low-power image processing system, as required for many new promising applications. The C6000 processor's DMA controller is able to handle image acquisition in background, hence making frame grabber hardware or large FIFO buffers unnecessary. Connecting the image sensor via the expansion bus leaves the full external memory bus bandwidth available to the CPU for intermediate storage and communication tasks. The large internal data memories (especially C6203 and C6416) and/or multi-path cache architecture provide sufficient high-speed storage space for efficient image processing. The available library of optimized basic image processing functions significantly cuts development time and reduces hardware requirements,

The limiting factor of such a system is the maximum DMA speed the processor is able to handle. Currently SVGA resolution at 25 frames per second is possible. Higher resolution requires to lower the frame rate and vice-versa. Future processor's will of course raise these limits.