V1KU

Hardware Design User's Manual

Version 1.4.1
Revised 09/23/2011

COGNIMEM
Technologies, Inc.

Limitation of Liability

CogniMem Technologies, Inc. (CTI) assumes no liability whatsoever and disclaims any express, implied or statutory warranty relating to the product described in this manual and accompanying materials ("Product") including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or non-infringement. In no event shall CTI be liable for any direct, indirect, consequential, punitive, special or incidental damages (including, without limitation, damages for loss of profits, business interruption, or loss of information) arising out of the use or inability to use the Product, even if CTI has been advised of the possibility of such damages. CTI makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice.

This Product is not designed, manufactured or intended by CTI for incorporation into products intended for use or resale in equipment in hazardous, dangerous to life or potentially life-threatening environments, such as in the operation of nuclear facilities, aircraft navigation or communication systems or direct life support machines, in which the failure of products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). The inclusion of the Product as critical component in High-Risk Activities implies that the manufacturer assumes all risk of such use and in doing so agrees to fully indemnify CTI for any damages resulting from such application.

Trademarks and Copyrights

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

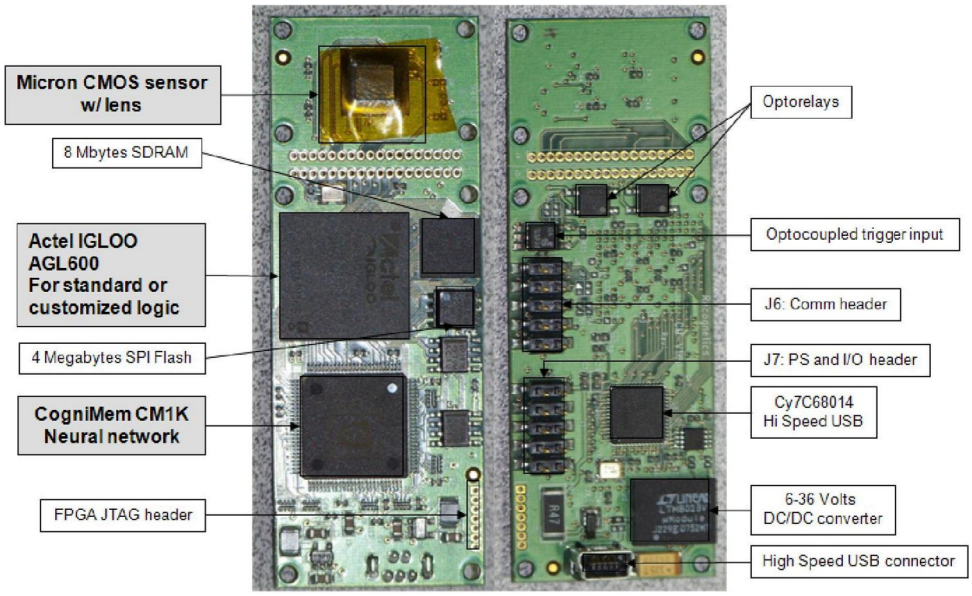
Contact Information

www.cognimem.com

Table of Contents

Hardware Description

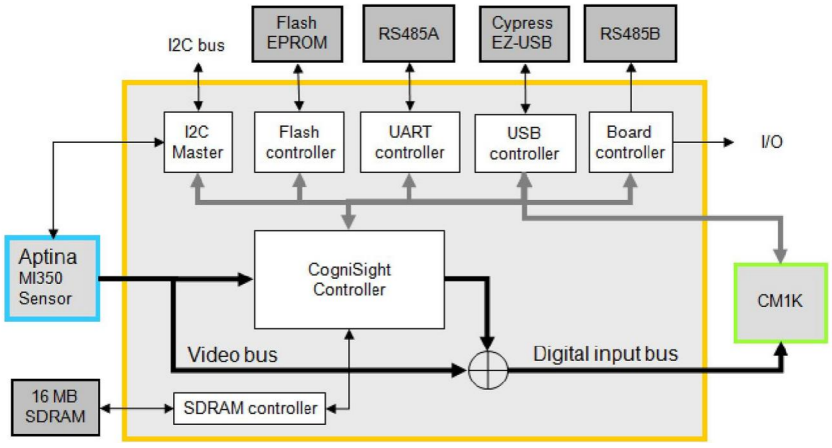
V1KU is an evaluation module for the CogniMem technology applied to video and image recognition. The board features a CogniMem chip with 1024 neurons, a high quality Aptina monochrome video sensor, a reconfigurable Actel Field Programmable Gate Array (FPGA), 8 MB of SDRAM, 4 MB of Flash memory, one high-speed USB2 port, two RS485 ports, 2 opto relays, and one opto-isolated input line.



The CogniMem CM1K chip can learn and recognize pixel data coming directly from the Micron sensor or previously manipulated in the FPGA to produce a feature vector. In the former case, the feature vector is automatically extracted by the CM1K chip from a region of interest defined by the user. The FPGA can also consolidate and format the response of the neural network for transmission to the outside world. The SRAM is partially used to store the memory frame but can also hold user data.

Default factory programming:

V1KU is delivered with an FPGA programmed with the components presented in the diagram below.



The components

Aptina MT9V022 video sensor

Monochrome or color, Progressive scan
752x480 pixels at 60 frames per second
Global shutter, Automatic exposure control (AEC), Automatic gain correction (AGC)
External trigger input
I2C sensor control

Lens (default)

60 degrees horizontal field of view angle
M7 (7 mm thread)
Adjustable to 10 mm horizontal Field of view (8 mm distance) to infinity

CogniMem CM1K chip

1024 silicon neurons working in parallel with automatic model generator
Classify vectors of up to 256 bytes, Up to 16382 categories
Radial Basis Function (Restricted Coulomb Energy) or K-Nearest neighbor classifier

FPGA hosting recognition and decision logic

Actel IGLOO FPGA 600 (600,000 gates)

Flash memory

Atmel Flash Memory 2048 pages of 264 bytes. SPI access.

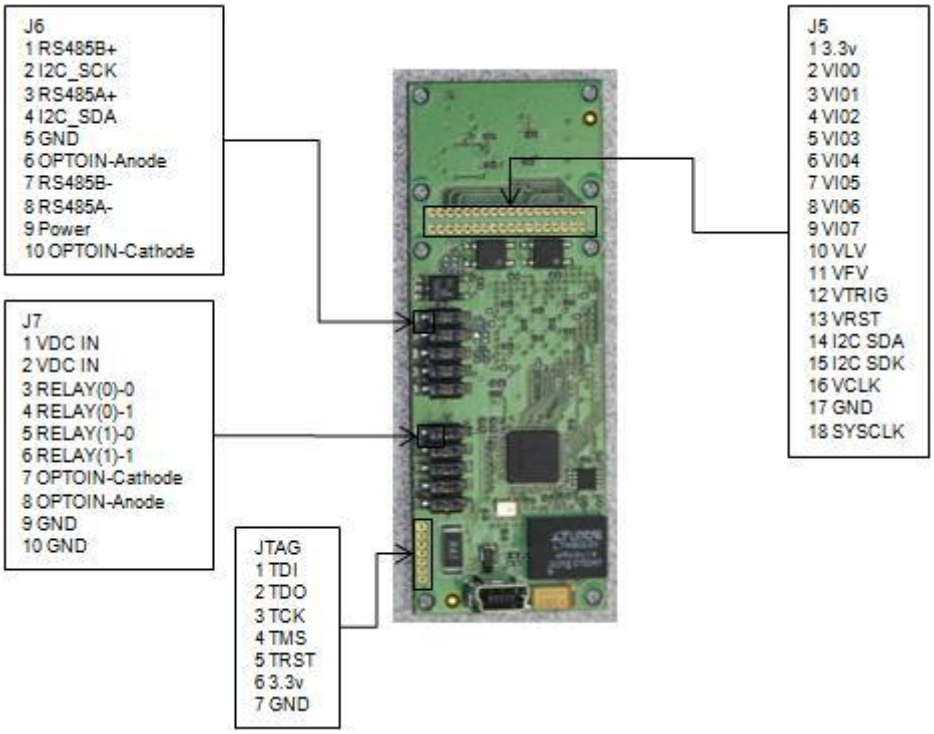
SDRAM

Micron MT48LC8M16A2. Synchronous 8 MB DRAM: 2 Meg x16 x 4 banks

Cypress USB chip

The Cypress Semiconductor Cypress CY7C68013A_8 USB Microcontroller chip supports the high bandwidth offered by the USB 2.0.

Connectivity and I/Os



Power supply

V1KU can be powered between 6 and 36 VDC. The recommended power is 6 Volts. Typical consumption is 750 mW.

The power can be supplied through the USB port or through an external supply depending on the configuration of jumper J1.

- USB: solder bridge between pins 3 and 4 of J1 (factory default)
- External: solder bridge between pins 1 and 2 of J1

Both bridges can be soldered, BUT in this case, pay attention NOT to connect an external power supply if V1KU is connected to a host via its USB connector.

USB Port

The Easy USB chip from Cypress (Cypress CY7C68013A -56LFXC) mounted on the V1KU has the following features:

- An integrated, high-performance CPU based on the industry-standard 8051 processor.
- A soft (RAM-based) architecture that allows unlimited configuration and upgrades.
- Full USB throughput. USB devices that use EZ-USB chips are not limited by number of endpoints, buffer sizes, or transfer speeds.
- Automatic handling of most of the USB protocol, which simplifies code and accelerates the USB learning curve.

The chip is connected to the FPGA via an 8-bit data bus and also via its I2C serial lines. In the default configuration, the Cypress chip uses its I2C lines only to interface with a Microchip Flash memory (8 kbytes) holding its configuration settings. Its slave address is 0xA0 (160). After the initialization of the board, the i2c_master controller programmed in the FPGA at factory settings accesses the Aptina sensor with the slave address 140 following the simple Register Transfer Level protocol described in the next chapter of this manual.

Drivers and USB development tools can be downloaded from the Cypress web site. Cypress includes an evaluation version of the 8051 Keil Software Tools in the USB 2.0 development kit.

RS485 serial ports

The V1KU board features two pairs of bi-directional serial lines RS485A and RS485B supporting up to 20 Mbits and compatible with Profibus.

Configuration at factory settings:

- RS485 port A: Interface with the RS485 controller programmed in the FPGA and supporting the protocol described in the next chapter of this manual.

- RS485 port B: Interface with an output of the FPGA for the optional serial transmission of the lowest 8 bits of the CM_CAT register after each CAT_VAL pulse @230,400 baud with Data bit=8, Stop bit=1, Parity= None, Handshake=None.

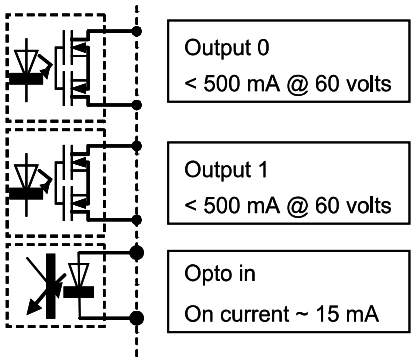
I2C lines

Two lines on the jumper J6 are reserved for I2C serial communication. They are disabled by default to ensure that the internal I2C communication between the Cypress chip and the Micron sensor are not disrupted. You can enable the communication between V1KU and two external I2C lines by configuring the jumper JP2 as follows:

SCK enable: solder bridge between pins 1 and 2 of JP2
SDA enable:solder bridge between pins 3 and 4 of JP2

At factory settings, the I2C master programmed in the FPGA interfaces to the I2C Slave address 144 or the Aptina sensor. Assuming that you want to connect to an external I2C device following the same protocol as the Aptina sensor, you can change the slave address by writing the register 0xA1. If your device supports a different protocol, you will have to load your I2C IP core in the FPGA. The I2C protocol of the Aptina sensor is described in the Aptina User's Manual.

Opto-isolated relay outputs



Two opto isolated relay outputs can drive up to 60 volts 500 mA continuously or up to 1500 mA during 100 milliseconds pulse.

Trigger Input

A current flow (25 mA max) can be applied between the anode and cathode of the Opto_In line.

I/O Connectors

J6 pins	Signal	Description
1	RS485B+	RS485 Channel B data+
2	I2C_SCK	I2C Clock
3	RS485A+	RS485 Channel A data+
4	I2C_SDA	I2C_Data
5	GND	GND power
6	OPTOIN-1	Opto In anode
7	RS485B-	RS485 Channel B data-
8	RS485A-	RS485 Channel A data-
9	Power	6-36 VDC power
10	OPTOIN-0	Opto In cathode
J7 pins		
1	VDC IN	6-36 VDC power
2	VDC IN	6-36 VDC power
3	RELAY(0)-0	Relay(0) output line 0
4	RELAY(0)-1	Relay(0) output line 1
5	RELAY(1)-0	Relay(1) output line 0
6	RELAY(1)-1	Relay(1) output line 1
7	OPTOIN-0	Opto In cathode
8	OPTOIN-1	Opto in anode
9	GND	GND power
10	GND	GND power

JTAG connector

At factory settings, the various components of the V1KU board are easily accessible through the Register Transfer Level (RTL) protocol programmed on the FPGA and described in the V1KU_RTL_SDK manual. This protocol is interpreted by both the USB controller and RS485 controllers.

Programming the V1KU FPGA requires the FlashPro software and FlashPro3 programming device from Actel (http://www.actel.com). You will have to make an adaptor/connector to plug into the V1KU J2 JTAG:

J2 pins	Signal	Description
1*	TDI	Transmit Data In
2	TDO	Transmit Data Out
3	TCK	Transmit Clock
4	TMS	Transmit Mode Select
5	TRST	Transmit Reset
6	VCC	3.3v
7	GND	Ground

*pin with the square print

Firmware Description

The Communication protocol

The communication protocol programmed in the FPGA of the V1KU board is based on the following packet sequence:

1 byte	4 bytes	3 bytes	N words
Device ID* Default =1	bit[31]= RW command 0, Read 1, Write bit[30-24]= Module ID bit[23-0]= Register	Datalength expressed in word Default = 1	Sequence of N data words N must be equal to Datalength * 2. Sent if bit[31]=1, Received if bit[31]=0

A single read or write command takes 10 bytes. A multiple read or write command takes 8+N*2 bytes.

The protocol is interpreted by both the USB and RS485 controllers. In the case of the RS485 protocol, the Write command returns one byte of acknowledge equal to the DeviceID.

*The notion is DeviceID is only useful if multiple V1KU boards are connected in a chain through the RS485 bus. In this case each board must have a unique identification number to decode its own instructions. A identification number equal to 0xF5 is universal and decoded by all devices at one.

Write Example

Write the value 0x0055 to the register 0x0B of the module 0x01 of the device 0x01

0x01 0x0100000B 0x000001 0x0055

Returns Nothing in the case of USB, 0x01 in the case of an RS485

Read Example

Read the value of the register 0x23 of the module 0x02 of the device 0x01

0x01 0x82000023 0x000001

Returns 0x0040 (i.e. board's serial number)

USB interface

The USB bulk transfers of the V1KU board are handled by the Cypress EZ_USB chip (CY7C68014, 56-pin model). Packets from host to V1KU must be sent to the end point EP2_OUT. Packets from the device to the host are received on the end point EP6_IN.

RS485 Port A

Bus specifications: 921,600 baud with Data bit=8, Stop bit=1, Parity= None, Handshake=None.

V1KU address mapping

The various components of the V1KU board are accessible through the following address map:

Address[31-24]=Module[7:0], see column 2 below
Address[23:8]=0x0000
Address[7:0]=Reg[7:0], see registers per module described in this chapter

Address Range	Module	Description
0x01000000 0x0100001F	1= CogniMem	Access to the CM1K neurons to learn and recognize vectors, save and restore knowledge. Also access to the recognition logic in bypass and video mode.
0x02000020 0x20000024	2= Board info	Access to information about the hardware and configuration of the opto-isolated lines.
0x03000030 0x03000036	3= Flash memory	Access to the Flash memory to read and write pages of data, but also to read and decode sequences of instructions for the other modules.
0x04000000 0x0400FFFF	4= Sensor	Access to the registers of the Aptina sensor.
0x05000000 0x05FFFFF	5= Memory	Access to the SRAM address bank. Recognition uses the RAM to store video frames digitized by the Aptina sensor, or to images transferred from a host.
0x06000000 0x06000076	6= Recognition Engine	Access to the CogniMem recognition engine to learn and recognize a region of interest, find and report known patterns in a region of scan.

The code of this module is 0x06.

Functions

The Recognition registers can be accessed using the following methods:

- CS.Comm.Write(byte Module, byte Register, word Data)
- CS.Comm.Read(byte Module, byte Register)

Registers

Register	Hex	Description	Default	Access
CS_CSR	0x60	Control Status Register (AUTO_RESET register) Bit [0], grab Bit [1], recognize ROI Bit [2], learn ROI Bit [3], recognize ROS, scan and append the position, category and distance of all recognized ROIs in a hit list. (3)	0	R/W
CS_ROILEFT (1)	0x61	Left position of the ROI	200	R/W
CS_ROITOP (1)	0x62	Top position of the ROI	120	R/W
CS_RECODIST (2)	0x63	Smallest distance of the last processed ROI	0xFFFF	R
CS_RECOCAT (2)	0x64	Best match category of the last processed ROI	0	R
CS_CATL	0x65	Category to learn	1	R/W
CS_ROSLEFT	0x66	Left position of the ROS	0	R/W
CS_ROSTOP	0x67	Top position of the ROS	0	R/W
CS_ROSWIDTH	0x68	Width of the ROS	752	R/W
CS_ROSHEIGHT	0x69	Height of the ROS	480	R/W
CS_HITCOUNT (3)	0x6A	Number of identified ROIs in the ROS		R
CS_HITX (3)	0x6B	Left position of the next ROI in the Hit list. Reading this register triggers the update of the associated registers HITY, HITDIST and HITCAT.		R
CS_HITY (3)	0x6C	Top position of the next identified ROI. Must be read after HITX.		R
CS_HITDIST (3)	0x6D	Distance of the next identified ROI. Must be read after HITX		R
CS_HITCAT (3)	0x6E	Category of the next identified ROI. Must be read after HITX.		R
CS_INIT	0x6F	Reset all the above to their default values		W

CS_FWIDTH (4)	0x71	Width of the image in memory	752	RW
CS_FHEIGHT (4)	0x72	Height of the image in memory	480	RW
CS_ROSSTEPX	0x73	Horizontal scanning step	16	RW
CS_ROSSTEPY	0x74	Vertical scanning step	16	RW
CS_RSR	0x75	Recognition Status Register Bit[2-0] = what to report 000, all recognized objects 001, all unknown objects 010, objects recognized with certainty 100, objects recognized with uncertainty Bit[3]=reserved Bit[8]=1, do not use the hardware reco-logic, but use the CogniMem reco-logic (results must be read from the RTDIST and RTCAT registers of the CogniMem controller, automated Search mode is not available)		RW

(1) The registers of the region of interest (ROI) are divided between the hardware and CogniMem modules. The position is defined in the CS registers. The Width, Height, BWidth and BHeight of the ROI are defined as CM registers. Refer to the Appendix “Feature Extraction” for more details about the block width and height.

(2) The registers CS_ROICAT and CS_ROIDIST reports the response of the neuron with the best match and are updated after each RecognizeROI command (CS_CSR=2). If the responses of additional firing neurons are needed, they can be retrieved by successive Read CM_DIST followed by Read CM_CAT.

(1) The instruction (Write CS_CSR=8) launches the automatic displacement of the ROI over the ROS and accumulates in a FIFO the XY locations of the ROI where a recognition occurs along with the associated distance and recognized category. Because the FIFO of the V1KU is sized to store 1024 records at a time you may have to divide the ROS into smaller portions if it contains more than 1024 hit points. This limitation is also enforced in the simulation in order to maintain compatibility with the hardware but it is very easy to circumvent as follows: After the instruction CS_CSR=8, Read CS_HITCOUNT, followed by the position and category of the Hit Points (CS_HITX, CS_HITY, CS_HITDIST, CS_HITCAT). If CS_HITCOUNT is equal to 1024, modify the starting top position of the ROS so that CS_ROSTOP is equal to the CS_HITY of the last point read in the FIFO. Reduce the remaining area height ROSHEIGHT accordingly and launch a new scanning with CS_CSR=8. Repeat the procedure if necessary. The advanced function CS_Search() implements this work-around.

(2) CS_FWIDTH and CS_FHEIGHT are used by the hardware engine to locate the ROI and ROS in the video frame. They must be updated when a new image is transferred from the host to the Recognition Bitmap. In the case of the V1KU, they must also be updated if the acquisition window of the sensor is changed, etc. The GrabImage() and SetWindowfunction of the DLL update CS_FWIDTH and CS_FHEIGHT automatically.

(3) Setting the bits 1, 2 or 3 of the CS_CSR register triggers commands which use the CogniMem reco-logic and write CM_CSR[0]=1. However, the Recognition controller has no means to know when to turn this reco- logic off. It is the programmer’s duty to do so when applicable by writing CM_RSR[0]=0;

(4) A speed limiting factor of the V1KU is the access to its SDRAM where the image is stored. The size of the ROI determines how many pixel values must be read from the memory at each instruction RecognizeROI or LearnROI. This number is multiplied in the case of RecognizeROS by a factor N equal to (ROSWIDTH ÷ ROSSTEPX) times (ROSHEIGHT ÷ ROSSTEPY). The feature

extraction on the other end is performed by the recognition logic of the CM1K chip which means that the number of blocks inside the ROI has no impact on the speed of the feature extraction.

Programming examples

Size a region of interest with a size of 32 x 32 divided into internal blocks of 2 x 2

Commands	USB protocol
Write CM_NWIDTH, 32	0x01 0x81000013 0x000001 0x0020
Write CM_NHEIGHT, 32	0x01 0x81000014 0x000001 0x0020
Write CM_BWIDTH, 2	0x01 0x81000015 0x000001 0x0002
Write CM_BHEIGHT, 2	0x01 0x81000016 0x000001 0x0002

Grab a video frame, move the region of interest to the location (10,12) and learn it as category 33

Commands	USB protocol
Write CS_CSR, 1	0x01 0x86000060 0x000001 0x0001
Write CS_LEFT, 10	0x01 0x86000061 0x000001 0x000A
Write CS_TOP, 12	0x01 0x86000062 0x000001 0x000C
Write CS_CATL, 33	0x01 0x86000065 0x000001 0x0033
Write CS_CSR, 4	0x01 0x86000060 0x000001 0x0004

Grab a new frame and recognize the region of interest

Commands	USB protocol
Write CS_CSR, 1	0x01 0x86000060 0x000001 0x0001
Write CS_CSR, 2	0x01 0x86000060 0x000001 0x0002
Read CS_RECOCAT	0x01 0x06000064 0x000001 (returns the recognized category)
Read CS_RECODIST	0x01 0x06000063 0x000001 (returns the distance)

Define a region of search (5, 3, 128, 256), and set its scanning step to 4.

Commands	USB protocol
Write CS_ROSLEFT, 5	0x01 0x86000066 0x000001 0x0005
Write CS_ROSTOP, 3	0x01 0x86000067 0x000001 0x0003
Write CS_ROSWIDTH, 128	0x01 0x86000068 0x000001 0x0080
Write CS_ROSHEIGHT, 256	0x01 0x86000069 0x000001 0x0100
Write CS_ROSSTEPX, 4	0x01 0x86000073 0x000001 0x0004
Write CS_ROSSTEPY, 4	0x01 0x86000074 0x000001 0x0004

Grab a new frame, scan the region of search and read how many objects are detected.

Commands	USB protocol
Write CS_CSR, 1	0x01 0x86000060 0x000001 0x0001
Write CS_CSR, 8	0x01 0x86000060 0x000001 0x0008
Read CS_HITCOUNT	0x01 0x0600006A 0x000001 (returns N, number of recognized ROIs)

Read the position and category of next recognized ROI in the list (can repeat N times)

Commands	*USB protocol*
Read CS_HITX	*0x01 0x0600006B 0x000001 (returns X position)*
Read CS_HITY	*0x01 0x0600006C 0x000001 (returns(Y position)*
Read CS_HITCAT	*0x01 0x0600006E 0x000001 (returns Category)*

Considerations about the processing speed

A speed limiting factor of the V1KU is the access to its SDRAM where the image is stored. As a consequence, the sizes of the ROS and ROI have a direct impact on the time of execution of the Search function since it determines the number of pixel values to read in the SDRAM. Given two sizes of ROI, the scanning of a same ROS with a same Step increment will execute faster with the smaller ROI.

The feature extraction on the other end is performed by the CM1K chip which means that the number of blocks inside the ROI has no impact on the speed of the feature extraction. It is done on the flow as the Recognition module sends the pixel values to the CM1K digital input bus.

Functions

The CogniMem registers can be accessed using the following methods and the module 0x01:

- CS.Comm.Write(byte Module, byte Register, word Data)
- CS.Comm.Read(byte Module, byte Register)

When reading or writing the components of a vector, the following two functions optimized the transfer of a data array to the CM_COMP register, thus replacing multiple CM_Read or CM_Write.

- CS.Comm.Read_Memory(long Address, int Length_inBytes, ref byte[] Data);
- CS.Comm.Write_Memory(long Address, int Length_inBytes, byte[] Data);

Registers

For more information about the CogniMem registers and programming examples, refer to the CM1K hardware user's manual and the CogniMem technology Reference Guide.

Register	Hex	Description	Default	Access
CM_NCR	0x00	Neuron Context	0	R/W in SR mode
CM_COMP	0x01	Component	0	W, R/W in SR mode
CM_LCOMP	0x02	Last Component	0	W, R/W in SR mode
CM_DIST(2)	0x03	Distance	0xFFFF	R
CM_CAT(2)	0x04	Category Must be read after CM_DIST	0xFFFF	R/W
CM_AIF	0x05	Active Influence Field	0x4000	R/W in SR mode
CM_MINIF	0x06	Minimum Influence Field	2	R/W
CM_MAXIF	0x07	Maximum Influence Field	0x4000	R/W
CM_NID	0x0A	Neuron identifier Must be read after CM_CAT	0	R
CM_GCR	0x0B	Global Norm and Context	1	W
CM_RESET CHAIN	0x0C	Point to the 1st neuron in SR mode		W
CM_NSR	0x0D	Network Status Register	0	R/W
CM_FORGET	0x0F	Clear the neuron registers, the Minif, Maxif and GCR global registers. Does NOT reset the NSR register.		W
CM_NCOUNT	0x0F	Return the number of committed neurons		R
CM_RSR	0x1C	Recognition Status Register	0	R/W
CM_RTDIST(2)	0x1D	Real-Time distance	0xFFFF	R
CM_RTCAT(2)	0x1E	Real-Time category	0xFFFF	R
CM_LEFT	0x11	Left position of the ROI	200	R/W
CM_TOP	0x12	Top position of the ROI	120	R/W
CM_NWIDTH(1)	0x13	Width of the ROI*	340	R/W
CM_NHEIGHT(1)	0x14	Height of the ROI*	220	R/W
CM_BWIDTH(1)	0x15	Width of the inner block	20	R/W
CM_BHEIGHT(1)	0x16	Height of the inner block	20	R/W

CM_ROIINIT	0x1F	Reset the ROI to default		W

(1) Refer to the Appendix “Feature Extraction” for more details about the block width and height.
(2) During a recognition, the register CM_CAT must be read after CM_DIST. On the other end, the values of the registers CM_RTDIST and CM_RTCAT are “latched” by the recognition logic and can be accessed individually and in any order.

Programming examples

Learn a vector (1,2,3,4) as category 5.

Commands	**USB protocol**
Write CM_COMP, 1	0x01 0x81000001 0x000001 0x0001
Write CM_COMP, 2	0x01 0x81000001 0x000001 0x0002
Write CM_COMP, 3	0x01 0x81000001 0x000001 0x0003
Write CM_LCOMP, 4	0x01 0x81000002 0x000001 0x0004
Write CM_CAT, 5	0x01 0x81000004 0x000001 0x0005

Recognize a vector (2,3,4,5) and read its best-match distance and category

Commands	**USB protocol**
Write CM_COMP, 1	0x01 0x81000001 0x000001 0x0002
Write CM_COMP, 2	0x01 0x81000001 0x000001 0x0003
Write CM_COMP, 3	0x01 0x81000001 0x000001 0x0004
Write CM_LCOMP, 4	0x01 0x81000002 0x000001 0x0005
Read CM_DIST	0x01 0x01000003 0x000001 (return value 4)
Read CM_CAT	0x01 0x01000004 0x000001 (return value 5)

Board module

The Board registers can be accessed using the following methods and the module 0x02:

- CS.Comm.Write(byte Module, byte Register, word Data)
- CS.Comm.Read(byte Module, byte Register)

Register	Hex	Description	Default	Access
EB_CSR	0x20	Control Status Register Bit [0], opto lines output 0, disabled; 1, enabled If enabled, Relay0 = CM_RTCAT[0], Relay1= CM_RTCAT[1] Bit [1], opto lines format 0, sustained 1, pulsed (de-asserted at rising edge of CM_FV) Bit [8], disable UART controller	0	RW
EB_CHAINID	0x21	Reserved (chainID number)		
EB_SLA	0x22	Address of the I2C slave address of Aptina sensor	140	RW
EB_SN	0x23	Board serial number		R
EB_FPGAREV	0x24	Version of the firmware in the FPGA		R

Sensor I2C Master module

The Sensor registers can be accessed using the following methods and the module 0x04:

- CS.Comm.Write(byte Module, byte Register, word Data)
- CS.Comm.Read(byte Module, byte Register)

The table below is a list of the most commonly used sensor registers. Refer to the Aptina User's Manual for a complete list.

Register	Hex	Description	Default	Access
MI_RESET	0x0C	Software reset causing the sensor to abandon the current frame capture. Does NOT Sensor reset the sensor registers	0	RW
MI_GAIN	0x35	Gain between 16 and 64.	16	RW
MI_SHUTTER	0x0B	Exposure time in number of row-times. The value can range between 1 and 480. Default row-time is 31.72 usec.	480	RW
MI_AGC	0xAF	Automatic Gain (bit0) and Automatic Exposure (bit1)	3	RW
MI_LEFT	0x01	Column start	0	RW
MI_TOP	0x02	Row start	0	RW
MI_HEIGHT	0x03	Window height	480	RW
MI_WIDTH	0x04	Window width	752	RW

Example 1: Read the shutter speed of the Micron sensor on the device 0x01

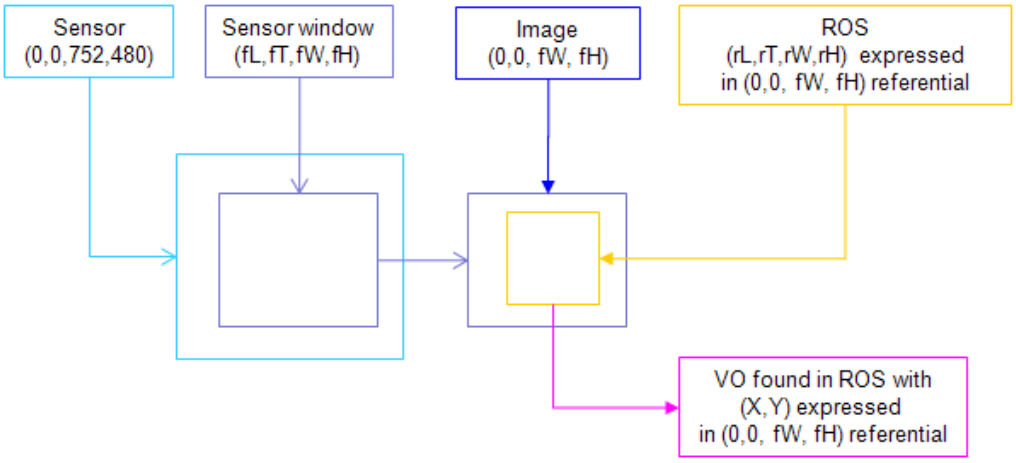
0x01 0x0400000B 0x000001 (returns 480 by default)

Example 2: Set gain to value 16 (0x10)

0x01 0x04000035 0x000001 0x00010

returns 0x01 in the case of RS485 protocol

The following diagram illustrates how the image frame referential is changed when you set a window of digitization different than the default (0,0,752,480)



Flash module

The Flash registers can be accessed using the following methods and the module 0x03:

- CS.Comm.Write(byte Module, byte Register, word Data)
- CS.Comm.Read(byte Module, byte Register)

A page can range from 0 to 2048 since the Flash memory is organized in 2048 pages of 264 bytes. The Flash Module can execute three advanced functions described in the list below.

- **Simple Read/Write** commands can be used to store and retrieve user data in the Flash memory of the board such as a project title, a date, sensor settings, definitions of regions of interest and even the contents of neurons.
- **Read and execute** a series of single Write commands addressing any of the controllers of the V1KU board. This command reads the flash memory per increment of four bytes and decodes them as follows:
    - Flash Byte0= Module[7:0] with bit[7]=RW
    - Flash Byte1= Register[7:0]
    - Flash Byte2-3= Data[15:0]
    - If the Module is different than 0xFF, the corresponding internal command is executed and the next four bytes are read. Otherwise the transaction is terminated.
- **Read and restore the neurons** starting at the current page. This command reads the flash memory per increment of 264 bytes and decodes them as follows:
    - Byte0-255= CM_COMP
    - Byte 256-257= CM_NCR
    - Byte 258-259= CM_AIF
    - Byte 260-261= CAT
    - Byte 262-263= 0x0000 (default) to go to next page and repeat, or 0xFFFF to stop
- **Restore All.** This command looks for a code in page0 to execute the following sequence automatically:
    - **Code = 0x6252:** Read_and_Restore the saved N neurons starting at page 1, and Read_and_Execute the settings saved in page N+1.
    - **Code = 0x6352:** Read_and_Execute the settings saved in page1.

Register	Hex	Description	Default	Access
FLASH_STOP	0x30	Release the Chip_Select line of the Flash (to terminate one of the following transactions)		
FLASH_R-STA	0x31	Page number from which the next data will be read.		W
FLASH_R_DATA	0x32	Read pair of bytes from the Flash memory		R
FLASH_W_STA	0x33	Page number to which the next data will be written		W
FLASH_W_DATA	0x34	Write pair of bytes to the Flash memory		W
FLASH_R_EXEC	0x35	Read the current page per increment of four bytes and decode as an internal command until Module=0xFF. Releases the Chip_Select line when done.		W
FLASH_R_NEUR	0x36	Read the current page, restore as the content of a neuron and repeat for next page unless Byte 262-263=0xFFFF. Releases the Chip_Select line when done.		W
FLASH_RESTORE	0x37	Restore a complete project if stamped as Valid.		W

Example 1: Write an array of data to a page of Flash memory
Write FLASH_W_STA, page_num
For i=0 to N: Write FLASH_W_DATA, data(i): Next i
Write FLASH_STOP, 0

Example 2: Read an array of N data from a page of Flash memory
Write FLASH_R_STA, page_num
For i=0 to N: Read FLASH_W_DATA, data(i): Next i
Write FLASH_STOP, 0

Example 3: Read a page of Flash memory and execute the corresponding internal Write commands
Write FLASH_R_STA, page_num
Write FLASH_R_EXEC, 0

Example 4: Read and restore the contents of the neurons starting at the page page_num
Write FLASH_R_STA, page_num
Write FLASH_R_NEUR, 0

Appendix A: Lens selection

V1KU comes with an M7 lens with a 6mm focal length. Optionally V1KU can be mounted with a faceplate compatible with M12 lenses.

Cleaning the lens

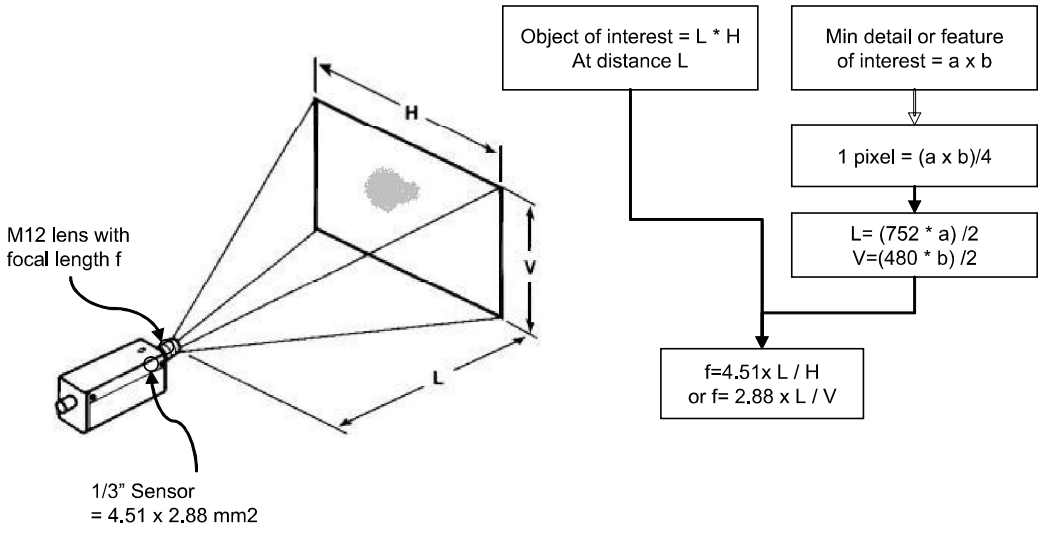
- To clean the lens, you will need a small screw driver, a Qtips and some rubbing alcohol
- Remove the 2 small screws located under the lens holder and extract the V1KU from the enclosure.
- Remove the two screws behind the holder
- Gently pull the lens holder straight (be careful the sensor is somewhat "clipped" inside)
- Rub the sensor with the wet QTips
- Clip back the lens holder over the sensor and connect to the Diagnostics or IKB software to view the image and verify that no dust spot appears in the field of view.
- When satisfied, screw the lens holder back

Changing the lens

If your object of interest is contained in a Field of View (FOV) with the dimensions HxV when placed at a distance L from the sensor, then one pixel represents H/752 x V/480 mm^2.

The minimum details to detect in the field of view should be at least equal to 2x2 pixels. From there you can find calculate the parameters H, V and also L, if applicable.

The focal length of the lens is then equal to f=4.51x L / H or f= 2.88 x L / V. The multiplication factors derive from the fact that the Aptina sensor mounted on the V1KU has a 1/3" optical format, that is 4.51 x 2.88 mm^2.



Appendix B: What is new in this manual ?

Revision from 09/23/2011

- The registers VOCOUNT, VOLEFT, VOTOP< VODIST and VOCAT have been replaced with HITCOUNT, HITX, HITX, HITDIST and HITCAT to match the terminology used in the .Net API delivered with the board
- Correction of an erroneous description: HITX and HITY do not refer to the top left corner of a recognized ROI but to its center
- Addition of a comment regarding the [0, 0x7FFE] value range of the registers CM_CAT and CS_CATL

Erratum

Noisy feature vector if image width is not modulo 16

V1KU with FPGA rev 3 to 5

The feature vector extracted by theRecognitionengine shows noisy components if the width of the image stored in memory is not a multiple of 16 (CS_FWIDTH).

This problem is especially noticeable when the width of the blocks inside the region of interest are small (CM_BWIDTH). The larger the width of the blocks, the more attenuated is the noise due to the averaging of the pixel values per block.

The commands using the faulty feature extraction are: CS_CSR=2, CS_CSR=4, CS_CSR=8.

Remedy:

- If loading an image from the host to the V1KU, crop it to a width which is a multiple of 16 prior to the transfer to memory.
- If the sensor is set to acquire a window area lesser than the full resolution of 752x480, make sure that the width of the window is a multiple of 16 (register 0x04 of the Aptina sensor)
- Do not set the sensor to a binning mode of 2 or 4 whenever possible, or enlarge the block width.