

Cyclops

# Cyclops:

## Image Sensing and Interpretation in Wireless Sensor Networks

---

*Reference Manual*

Mohammad Rahimi  
mhr@cens.ucla.edu

Rick Baer  
rick.baer@agilent.com

6 February 2005  
Preliminary Documentation for Cyclops 1.0

---

Cyclops is an embedded image capturing and inference module. It is designed for extremely light-weight imaging and interpretation. It by no means have capabilities for classical image processing. It is designed with power economy in mind. Each component sleeps deeply when it is in no use.  
welcome to the world of Cyclops...

---

**Note: This document is preliminary. Please check out for updates  
February 2005**

## License

---

Agilent Technology  
Center for Embedded Network Sensing

Copyright (c) 2003 Agilent Corporation. All rights reserved.

Copyright (c) 2003 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Neither the name of the University nor Agilent Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS , AGILENT CORPORATION AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR AGILENT CORPORATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction and Overview</b>                  | <b>1</b>  |
| <b>2</b> | <b>Cyclops Hardware</b>                           | <b>3</b>  |
| 2.1      | Imager . . . . .                                  | 4         |
| 2.2      | Microcontroller . . . . .                         | 5         |
| 2.3      | CPLD . . . . .                                    | 6         |
| 2.4      | SRAM and FLASH . . . . .                          | 6         |
| 2.5      | Other Components . . . . .                        | 6         |
| 2.6      | Schematic . . . . .                               | 6         |
| <b>3</b> | <b>Cyclops Firmware</b>                           | <b>8</b>  |
| 3.1      | Operation System . . . . .                        | 8         |
| 3.2      | Dedicated CPLD Logic . . . . .                    | 9         |
| 3.3      | CPLD and Direct Memory Access . . . . .           | 9         |
| 3.4      | Cyclops Imaging Layer . . . . .                   | 10        |
| 3.5      | Libraries . . . . .                               | 12        |
| 3.5.1    | Matrix Operation . . . . .                        | 12        |
| 3.5.2    | Histogram Operation . . . . .                     | 12        |
| 3.6      | Directory Organization . . . . .                  | 12        |
| 3.7      | Naming Convention . . . . .                       | 12        |
| 3.8      | Version Numbers . . . . .                         | 13        |
| <b>4</b> | <b>Getting started</b>                            | <b>14</b> |
| 4.1      | Requirement . . . . .                             | 14        |
| 4.1.1    | Hardware . . . . .                                | 14        |
| 4.1.2    | Software . . . . .                                | 14        |
| 4.2      | Installation . . . . .                            | 14        |
| 4.3      | Image Visualization Tools on PC . . . . .         | 14        |
| 4.4      | Getting the First Image . . . . .                 | 14        |
| 4.5      | Getting the Image Through Wireless Link . . . . . | 14        |
| <b>5</b> | <b>Code Contribution and Sharing</b>              | <b>15</b> |
| <b>6</b> | <b>Contact Information</b>                        | <b>15</b> |

# 1 Introduction and Overview

Vision is a rich sensing experience. It provides a degree of attachment to the physical world that may hardly be accomplished with any other sensing modality. Combination of intensity and color may reveal information about the shape, geometry and condition of our surroundings. This capability can be further enhanced if it can be deployed in large numbers, for long period of time. It provides multiple perspective about the world and avoids any occlusion effects. Such capability can be used extensively in our surrounding environment, in office for occupancy control, in buildings for light level measurement and actuation and in habitat for irrigation control or phenology.

Today as we learn how to integrate sensors in our environment we still face many challenges with imaging sensor technology. To Couple imagers with the environment with high degree of longevity, we need low power image capturing and processing capability. Emergence of CMOS imaging with low power consumption and moderate imaging quality brings such dreams closer to realm of reality. These imagers are widely used in electronic gadgets such as cell phones and typically include a lens, an image sensor, and image processing in a small package. In spite of this high level of integration, these devices are still difficult to use in typical lightweight nodes. Image data must be captured synchronously at high rates of speed, and programming is complex.

Cyclops (Figure-1) attempts to overcome these difficulties. It is an electronic interface between a camera module and a lightweight (potentially wireless) host. Cyclops contains programmable logic and memory circuits for high-speed data transfer. It also contains a micro-controller (MCU) that presents a simple interface to the outside world. This combination will potentially serve as a smart and readily available sensor in optical domain. It isolates the high-

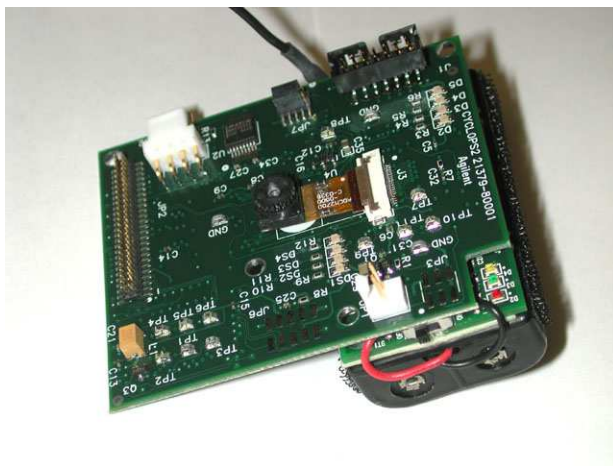


Figure 1: Cyclops is essentially a smart vision sensor that will be controlled by a host. This picture shows cyclops with a host Mote

speed requirement of connection to an imager and can be called by an embedded bus to get a frame in lower speed or as we envision, particular sensing information that the host application is interested in.

Cyclops power consumption is minimal to enable extended lifetime. This, on the other hand results in extreme constraints in its computational power and imaging quality. Cyclops strength is in 1) deployment quantity which results in better coverage in the face of occlusion and 2) lifetime for extended deployment. Its weakness is its 1) constrained computation and 2) image size. This makes cyclops appealing for particular classes of application. Any application that may have access to abundant power or need classical image processing capabilities may consider other imaging platforms.

## 2 Cyclops Hardware

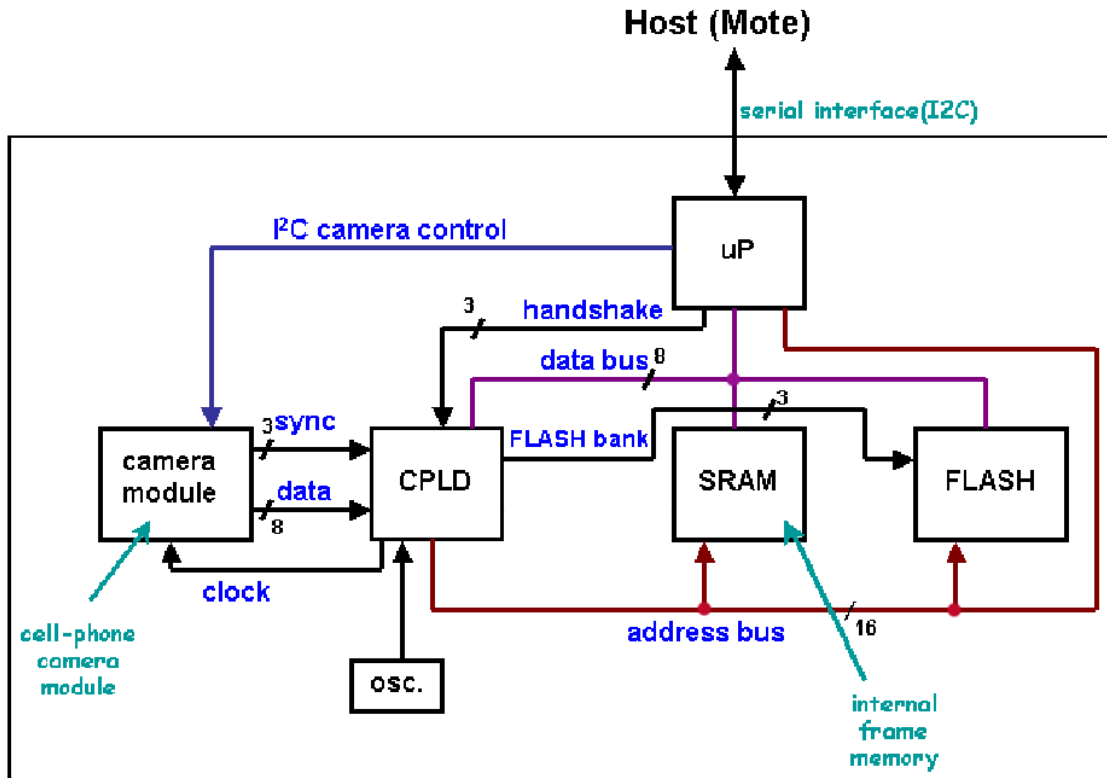


Figure 2: Diagram shows Hardware architecture of first generation of Cyclops.

Cyclops (Figure- 2) consists of an imager, microcontroller (MCU), a complex logice device (CPLD), external SRAM and external FLASH. The MCU serves as the main brain of the Cyclops sensor. It can set parameters of the imager, instruct imager to capture a frame and run local computation on the image to produce inference. Image capturing needs high frequency clocking of the imager which can be provided by CPLD. MCU uses CPLD for grabbing frames from the imager and saving them in memory. In essence the MCU sets the CPLD in imager communication mode to feed the clock to the imager and it sets the parameters of the imager via its I2C interface. It also set the imager in the video mode streaming images constantly. The MCU then instruct CPLD to capture the most recent frame indicating the starting location of the image in the memory. CPLD captures the next coming frame and copies it one byte at a time in the memory. At the end of the operation, CPLD interrupts the MCU to indicate that the capture operation was successfully accomplished. MCU may then access the memory to do any image specific operation. CPLD can accommodate specific expert logic for faster



Figure 3: Agilent ADCM-1700 is a CIF resolution ultracompact module that is only  $8.0 \times 7.0 \times 5.4mm$

hardware operations. These operations may even happen as the image capturing is in process. For example it can perform background subtraction, or frame differentiation at capturing time. This results in extremely economical use of resources since the CPLD is already clocking at the capture time.

Cyclops uses external SRAM to increase the limited amount of internal MCU memory. This provides MCU the necessary memory for image storage and manipulation. Although this memory resides external to MCU, it still can be accessed as normal memory location preserving a cohesive view of Cyclops memory space. In addition Cyclops has external FLASH. The FLASH accommodates large permanent data storage which extends the sensing potential to such sensing that needs permanent storage (e.g. Template matching). Both SRAM and FLASH are shared between MCU and CPLD and they communicate over a shared memory bus. This facilitates easy data transfer between the imager, shared SRAM and FLASH memory but it also requires mechanism that guarantees synchronized access to such shared resources. This will be further described in Cyclops firmware discussion.

Each module in Cyclops can go to different level of sleep. Typically lower consumption sleep level have higher wake up cost. An application should set the sleep level based on the amortized wake-up cost.

## 2.1 Imager

Imager is the most important component of the Cyclops. It is an ultra compact CIF resolution CMOS camera module (ADCM-1700) from Agilent Technology. It combines Agilent CMOS

image sensing and processing design with high quality lens. The lens has a focal length of  $2.10mm$  with practical focal depth of almost  $100mm$  to infinity and has  $52$  deg full angle field of view.

The image sensor has CIF resolution of  $352$  but in practice due to memory constrains Cyclops can use limited part of this resolution. The imager is also capable of adjusting the size of the image. It has two parameter for setting the window size. Input window size determines the array of sensor pixels and output window size determines the number of pixels in output window. In practice in cases that input window size is greater than the output window size, the imager will averages the input pixel to generate the result. On the other hand, if input size is smaller than the output size the imager will extrapolate the input data to generate the target output window. Finally the imager is capable of panning across the feild of view. If the image input and output sizes both are smaller than CIF image then the image is normaly taken from the center of the feild of view. On the other hand one can set a determined offset so that the image can pan across the entire feild of view.

The image processing unit is capable of generating variaety of image formats. Currently Cyclops support 3 image format of intensity image, RGB and YCbCr format with 8 bit depth. The imager can produce images in still or video mode. It is programmed through a serial I2C compatible for setting parameters and prodeuced data either in serial or parallel mode. In addition many low level parameters such as exposure or white balancing may be accessed for sensing specific adjustment.

## 2.2 Microcontroller

The microcontroller(MCU) is ATmega128L<sup>1</sup> conventionally receives so much attention, is not particularly noteworthy. It is an 8-bit Harvard architecture with 16-bit addresses. It provides 32 8-bit general registers and runs at 4 MHz and 3.0 V. The system is very memory constrained: it has 8 KB of ash as the program memory, and 512 bytes of SRAM as the data memory. The MCU is designed such that a processor cannot write to instruction memory; our prototype uses a coprocessor to perform that function. Additionally, the processor integrates a set of timers and counters which can be con gured to generate interrupts at regular time intervals. More noteworthy are the three sleep modes: idle, which just shuts o the processor, power down, which shuts o everything but the watchdog and asynchronous interrupt logic necessary for wake up, and power save, which is similar to the power down mode, but leaves an asynchronous timer running.

Atmel's AVR microcontrollers have a RISC core running single cycle instructions and a well-defined I/O structure that limits the need for external components. Internal oscillators, timers, UART, SPI, pull-up resistors, pulse width modulation, ADC, analog comparator and watch-dog timers are some of the features you will find in AVR devices. AVR instructions are tuned to decrease the size of the program whether the code is written in C or Assembly. With

---

<sup>1</sup><http://www.atmel.com/>

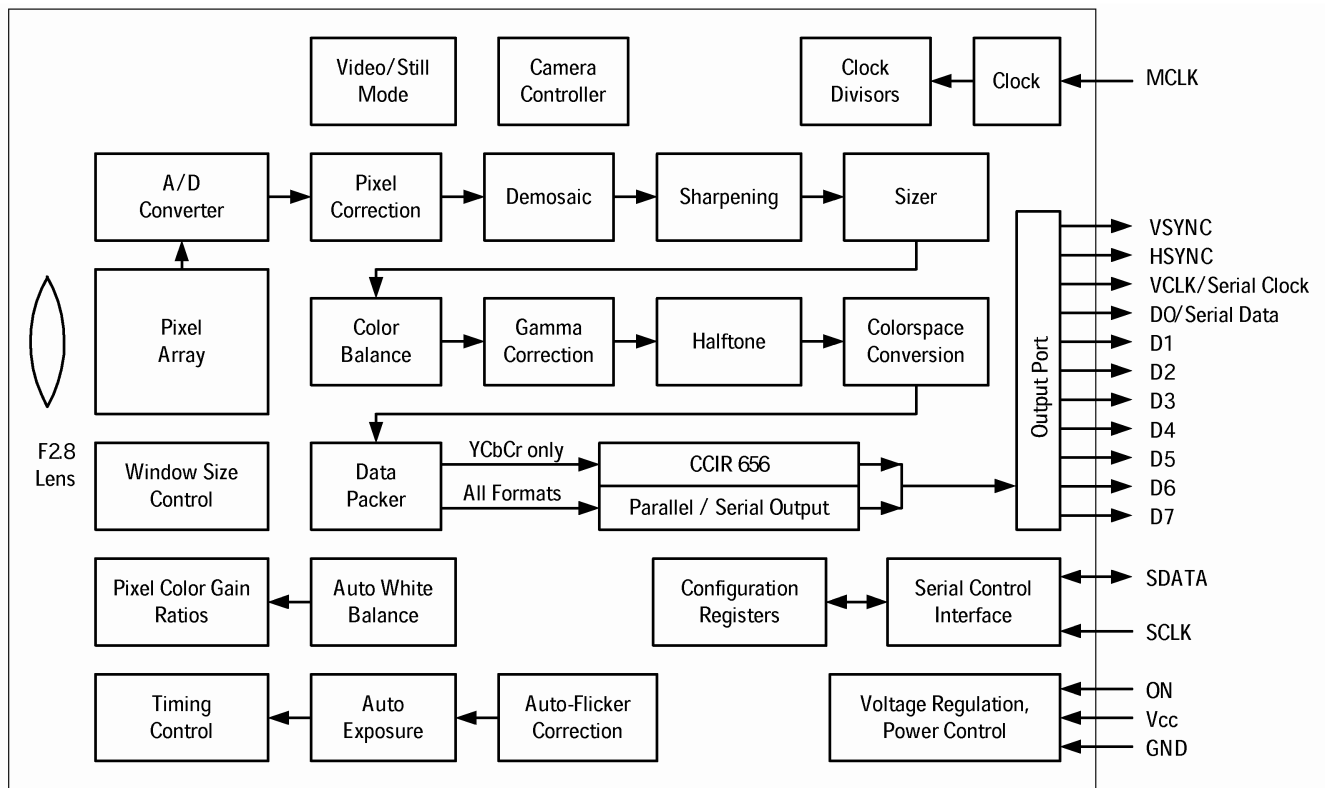


Figure 4: ADCM-1700 block diagram. It consist of F2.8 lens, image sensor and digitizer, image processing units and data communication units.

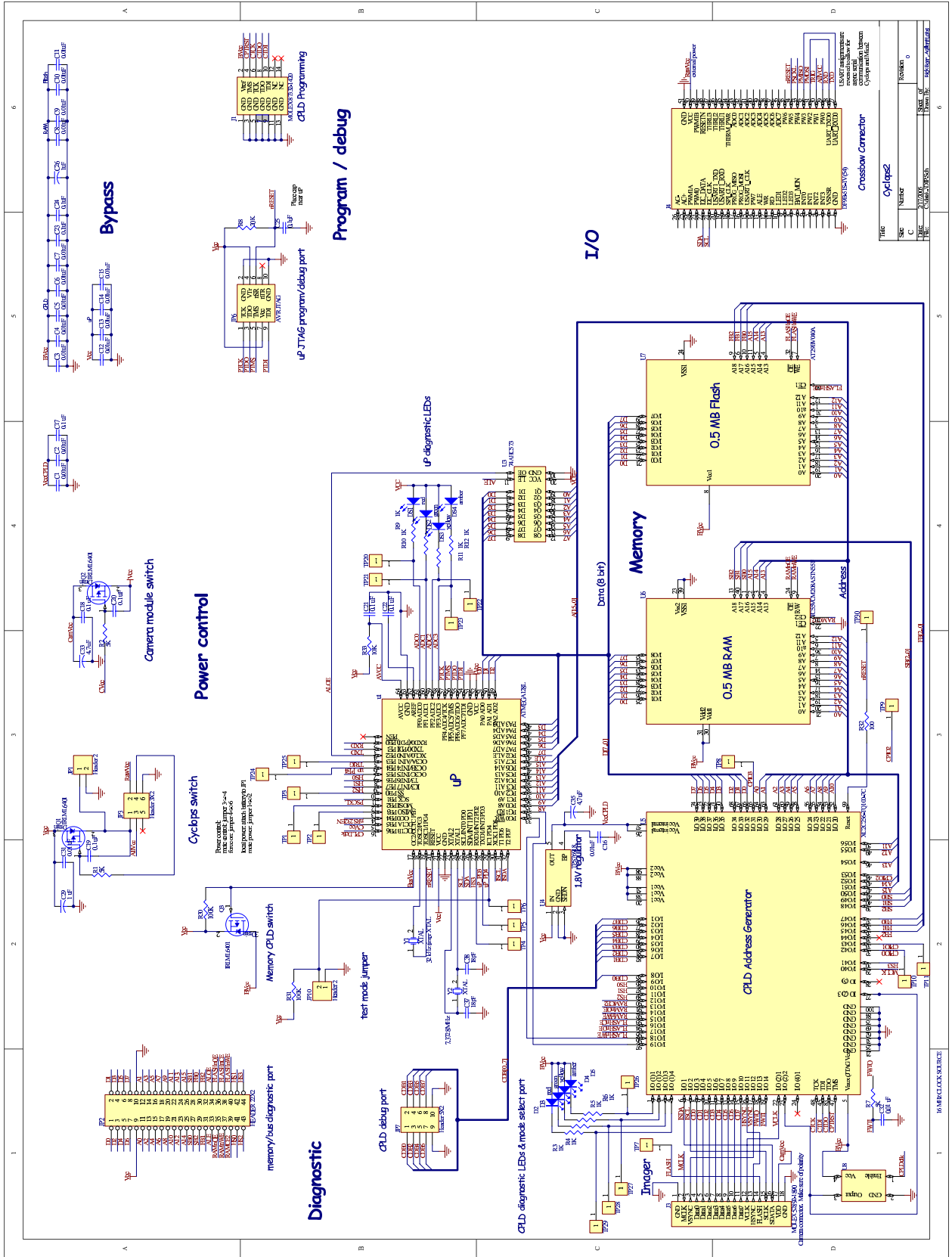
on-chip in-system programmable Flash and EEPROM, the AVR is a perfect choice in order to optimize cost and get product to the market quickly.

### 2.3 CPLD

### 2.4 SRAM and FLASH

### 2.5 Other Components

### 2.6 Schematic



### 3 Cyclops Firmware

#### 3.1 Operation System

Cyclops software is written in nesC language and runs in TinyOS<sup>2</sup> operating system environment. TinyOS is an open-source operating system designed for wireless embedded sensor networks. Its software architecture consists of different objects or components that communicate via defined interfaces. It has an event-driven execution model which initiates from lower layer components attached directly to the hardware and resonates across the chain of components all the way to application layer.

There are three sets of components in Cyclops (Figure-2): drivers, libraries and sensor application. Drivers are the set of components that enable the MCU to communicate with different peripherals. Example of drivers are the imager setting and capturing drivers, CPLD drivers, FLASH drivers or drivers that enable communication with the host Mote. There are also different set of libraries. Among the very important libraries are the variety of matrix operation, statistics and histogram. These libraries are key to do any processing on raw image. Other libraries may be closer to application (algorithm) such as object detection or motion vector recognition libraries that are designed do specific high level tasks. Sensor Application is the main sensing component on Cyclops. It receives proper command from the host and is responsible for necessary sensing operation as requested by the host. In fact sensing application makes the cyclops a smart sensor. For information about the hierarchy of the components and directories please consult the firmware hierchy section.

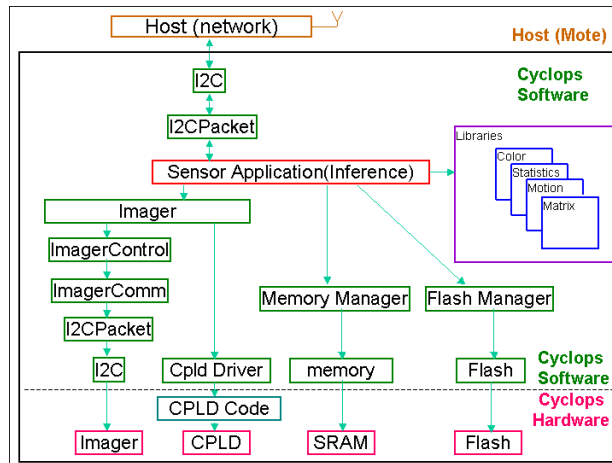


Figure 5: Component layouts in the Cyclops. The components consist of the main sensing application, the libraries that encapsulate image manipulation logic and the devices that enable communication with hardware components.

<sup>2</sup><http://www.tinyos.net/>

### 3.2 Dedicated CPLD Logic

CPLD code is written in Verilog Descriptive Language. It is a big state machine (Figure- 6) that has several different modes that can be programmed by the microcontroller. Microcontroller selects the operating mode by programming the CPLD's internal registers and by asserting control over CPLD handshake line. The operations can be divided into five classes:

- Facilitate MCU access to memory
- Capture image data to SRAM (with or without averaging)
- Copy data between SRAM and Flash
- Standby (low power mode)
- Program mode (set parameters for the next operation)

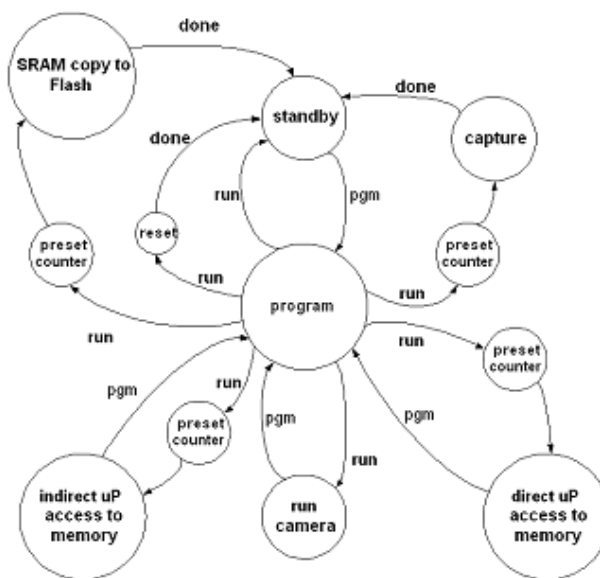
Four registers are used to define all CPLD operations. These register values are established by the microcontroller, via data transfer through the common data bus. The first register contains the operation code, second register is reserved, third register stores starting page address and the fourth register stores the ending address page. Pages are 256 byte and addresses should be specified with such granularity. Start page address is important in capture and copy operation and end page is important in copy operations between the SRAM and Flash memories.

In some states (such as capture image) the CPLD state machine returns to the program mode by end of operation. In such cases it notifies the microcontroller for further action. In other cases (such as direct memory access) it will continue on that mode until explicitly asserted by microcontroller for a new mode.

### 3.3 CPLD and Direct Memory Access

In Cyclops external memory is shared among many entities including MCU, SRAM, FLASH and CPLD. This potentially creates contention for bus access. On the other hand SRAM and FLASH are only passive components on the bus, hence only a mechanism is necessary for synchronization of MCU and CPLD access to the bus. We implemented Direct Memory Access mechanism to synchronize bus usage. In firmware design of the MCU every component assumes that it has access to the external memory. In times that CPLD needs access to the BUS for example for capturing an image the peripheral (in this case CPLD driver) request the scheduler a DMA transaction. When access for such transaction is granted by scheduler the drivers set the CPLD to the proper operation mode. At the end it will notify scheduler for normal operation. In DMA period the MCU is in sleep mode to consume minimal energy.

There are two mechanisms that guarantee safe access of each component to memory. 1) The default CPLD mode is direct memory mode access. 2) Cyclops scheduler sleeps in DMA



CPLD State Diagram

Figure 6: CPLD state machine. In program mode CPLD will be able to receive commands from microcontroller and perform the task it has been programmed.

mode. This combination means that any external memory access in task routine will happen while CPLD is in memory access mode. Libraries should be implemented in task context and this leverages access to the external memory without any requirement.

**WARNING: No external memory access should be performed in interrupt service routine.**

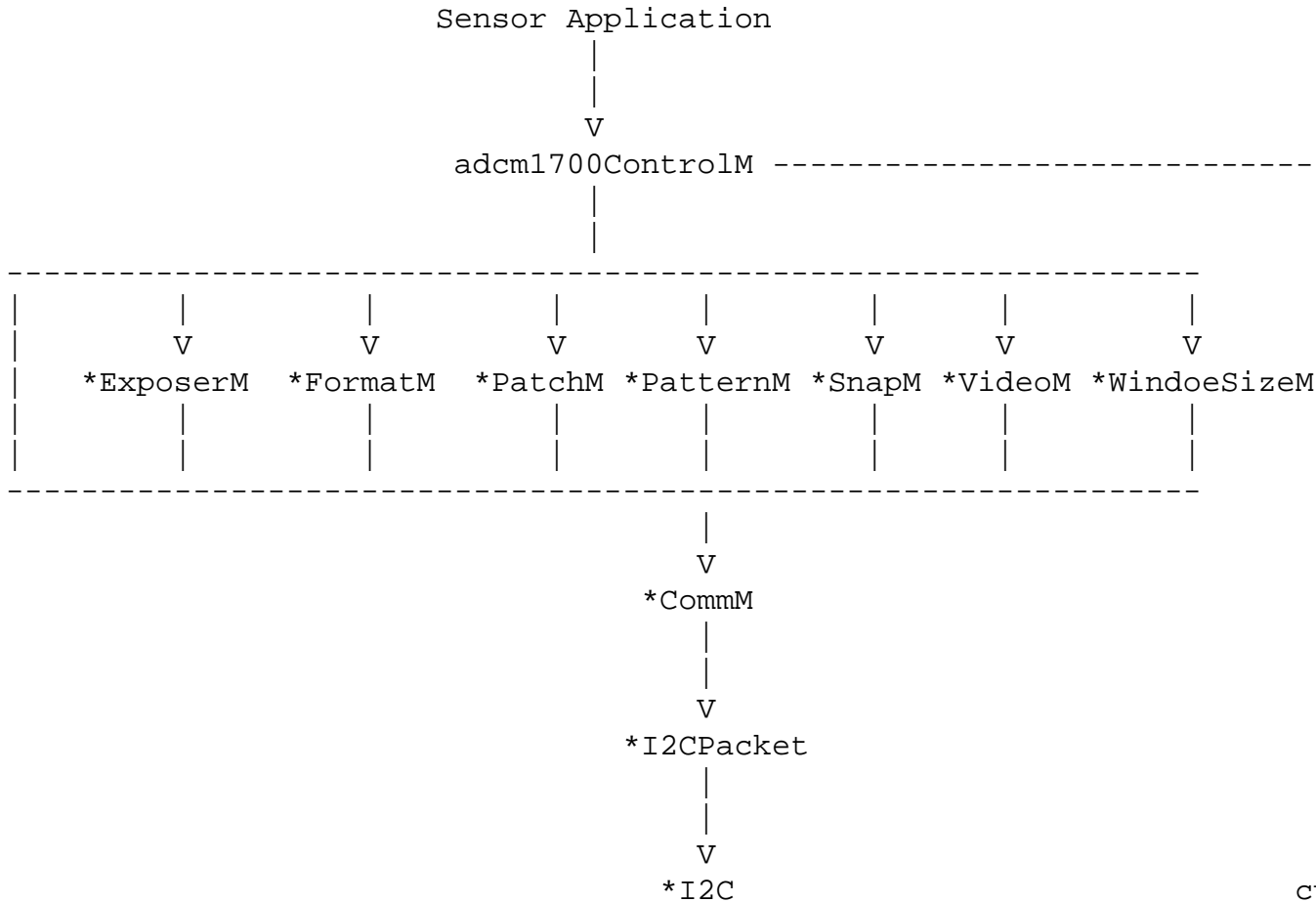
### 3.4 Cyclops Imaging Layer

Cyclops imaging(CI) layer consist of a main component (adcm1700ControlM) and number of smaller components. The adcm1700ControlM module is responsible for 1)synchronization between smaller modules 2)keeping the state of the imager. To asset control over different parametrs of the imager this module exploits module who are expert in changing particular behaviour of the imager. The expert modules are:

- adcm1700ExposerM : setting exposure parameter
- adcm1700FormatM : setting image format such as intensity or RGB
- adcm1700PatchM : patching initialization parameter
- adcm1700PatternM : setting the imager to generate particular test patterns
- adcm1700SnapM : snampping the image

- adcm1700VideoM : setting the imager in the video or still image mode
- adcm1700WindoeSizeM: setting input and output window size or panning

All the components in CI layer communicate with the imager over a dedicated I2C bus. This communication is managed by adcm1700CommM that implements imager specific protocol on top of the dedicated I2C bus. In addition the adm1700Control can communicates with cpldM component to capture the image.



### Image Data Structure

```

typedef struct CYCLOPS_Image
{
  uint8_t type;
  uint16_t width;
  uint16_t height;
  uint8_t *imageData;
  bufferPtr imageHandle;
}CYCLOPS_Image;
  
```

### Image Capture Parameters

```
typedef struct CYCLOPS_Capture_Parameters
{
int offsetWidth;
int offsetHeight;
int inputSizeWidth;
int inputSizeHeight;
}CYCLOPS_Capture_Parameters;
```

---

## 3.5 Libraries

### 3.5.1 Matrix Operation

#### Matrix Data Structure

```
typedef struct{
uint8_t depth;
union
{
uint8_t* ptr8;
uint16_t* ptr16;
uint32_t* ptr32;
}data;
uint16_t rows;
uint16_t cols; } CyclopsMat;
```

---

- Matrix
- MatrixLogic
- MatrixArithmetic
- MatrixStat

### 3.5.2 Histogram Operation

## 3.6 Directory Organization

## 3.7 Naming Convention

Cyclops software uses the following naming conventions:

- Constant identifiers are in uppercase; for example, CYCLOPS\_1BYTE\_MATRIX  
CYCLOPS\_IMAGE\_TYPE\_RGB\_888,

where the former represents matrix with depth of 1 byte and the latter represents RGB image with 3 bytes format.

- All names of used have the CYCLOPS prefix; for example CYCLOPS\_Image is the data structure of an image and CYCLOPS\_Matrix is data structure of a matrix.

### **3.8 Version Numbers**

Cyclops firmware version numbering follows the Linux operating system version convention. Version numbers consist of major release number and minor release number. Major numbers are based on major changes in the firmware architectures or new hardware release support. Minor odd release numbers are experimental and even numbers are stable. The first cyclops release version will be 1.0. This version will be tested and verified for tinyOS 1.0. The next expected stable release is 1.2 that will comply the TinyOS-1.2 requirements.

## 4 Getting started

### 4.1 Requirement

#### 4.1.1 Hardware

To start working with cyclops you need the following:

- Cyclops board
- Cyclops programming cable
- Cyclops programmer:XBOW MIB510 serial programmer or avr series equivalent
- A pair of Mica2 or MicaZ Motes from XBOW for wireless experimentation

Cyclops can be powered from the host through programming and communication connector. In addition it has connectors to connect directly to external power sources such as batteries.

**WARNING: Connecting power from both connectors can create permanent failure in the device.**

#### 4.1.2 Software

### 4.2 Installation

### 4.3 Image Visualization Tools on PC

### 4.4 Getting the First Image

### 4.5 Getting the Image Through Wireless Link

## **5 Code Contribution and Sharing**

## **6 Contact Information**

for purchasing please contact (rick\_baeragilent.com) (rick\_baeragilent.com) (rick\_baeragilent.com)