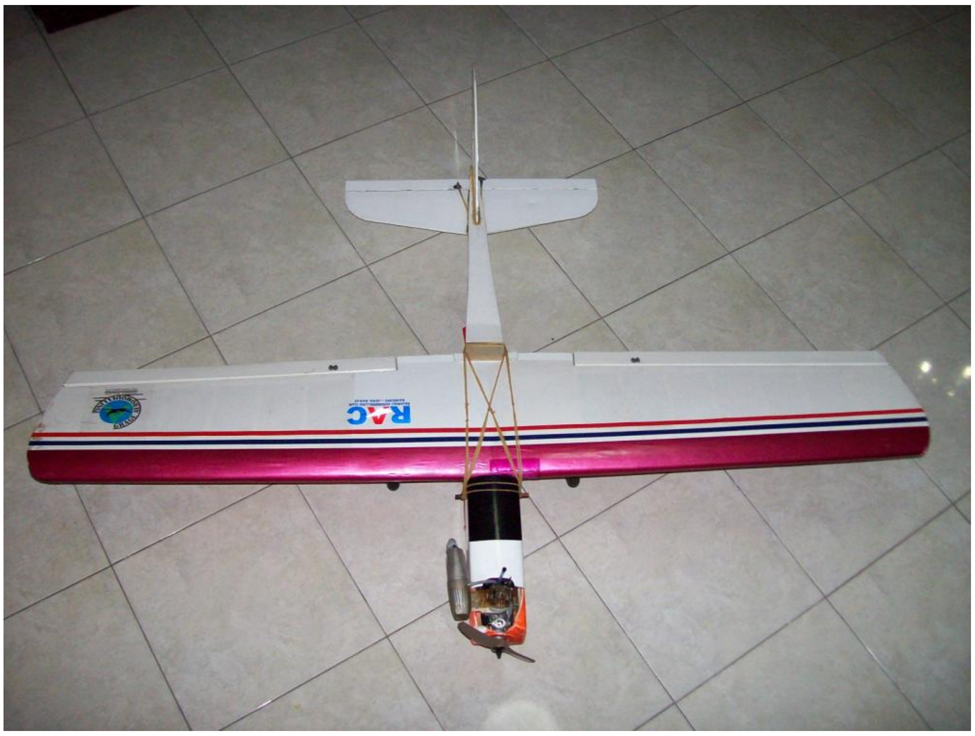
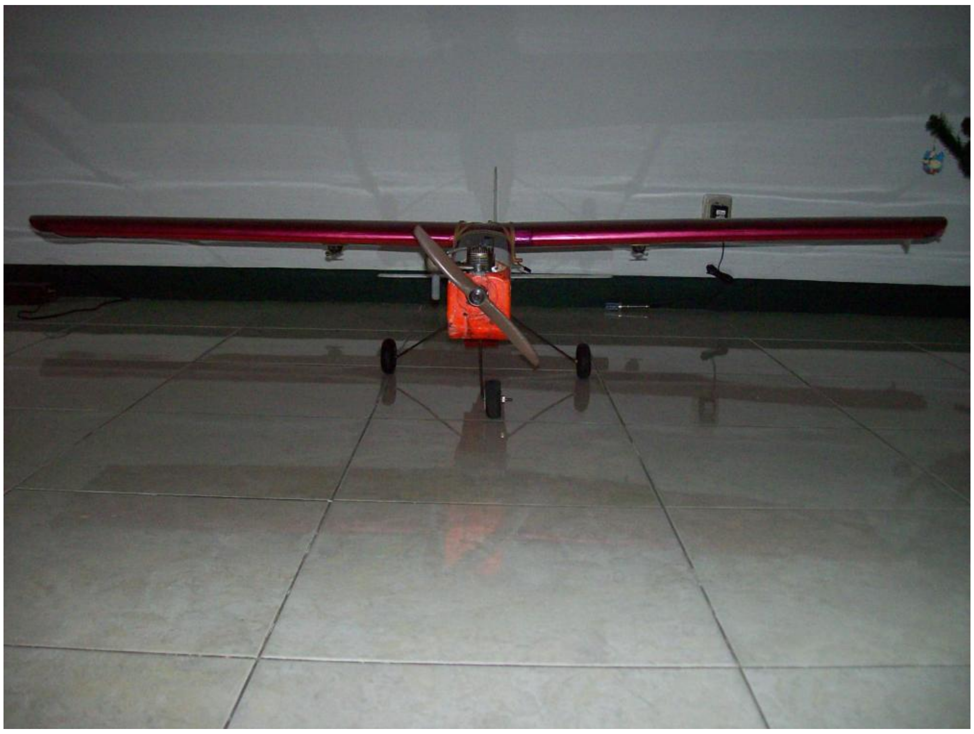
LAMPIRAN A
FOTO ROBOT TERBANG

Tampak Atas



Tampak Depan



Tampak Samping Kiri

Tampak Samping Kanan

LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO ATTINY2313

PROGRAM UTAMA

PENGONTROL MIKRO ATTINY2313

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 12/17/2009
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATtiny2313
Clock frequency     : 11.059200 MHz
Memory model        : Tiny
External SRAM size  : 0
Data Stack size     : 32
*****************************************************/

#include <tiny2313.h>
#include <delay.h>
#include <stdio.h>

unsigned int x,s;
unsigned char dat[32];

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
```

```
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
    if(data==255)// carry return
   {
   x=0;
   s=1;
   }
   if(s==1)
   {
   dat[x]=data;
   x++;
   }

}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
```

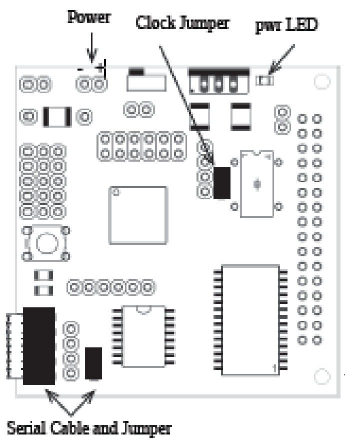
```
#include <stdio.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port D initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// Interrupt on any change on pins PCINT0-7: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Universal Serial Interface initialization
// Mode: Disabled
// Clock source: Register & Counter=no clk.
// USI Counter Overflow Interrupt: Off
USICR=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 38400
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x06;
UBRRH=0x00;
UBRRL=0x11;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

// Global enable interrupts
#asm("sei")
printf("RS\r"); // Reset the camera
delay_ms(100); // must delay …can delay less than 100ms
printf("\r");
delay_ms(100);
printf("sv 0 122\r");
```

```
delay_ms(100);
printf("PM 0\r"); // turn poll mode on
delay_ms(100);
printf("RM 3\r"); // turn on raw data mode for packets received from camera
delay_ms(100);
printf("SW 1 1 80 143\r"); // not necessary this is default window
delay_ms(100);
printf("tc 100 200 16 60 16 60 \r");//@#$%^&*percobaab
//printf("tc 16 50 60 180 60 160 \r");
delay_ms(300);
while (1)
      {
      // Place your code here
       if(dat[9]>=1)
        {
        printf("RS\r"); // Reset the camera
        delay_ms(100); // must delay …can delay less than 100ms
        printf("sv 0 0\r");
        delay_ms(800);
        }
//          else
//          {
//          printf("RS\r"); // Reset the camera
//          delay_ms(100); // must delay …can delay less than 100ms
//          printf("sv 0 200\r");
//          delay_ms(600);
//          }
        printf("RS\r"); // Reset the camera
        delay_ms(100);
        printf("sv 0 190\r");
        delay_ms(600);
        printf("tc 100 200 16 60 16 60 \r");//@#$%^&*percobaab
        delay_ms(200);
      };
}
```

LAMPIRAN C
DATASHEET

Setting Up the Hardware

In order to initially test your CMUcam2, you will need a serial cable, a power adapter and a computer. The CMUcam2 can use a power supply which produces anywhere from 6 to 15 volts of DC power capable of supplying at least 200mA of current. This can be provided by either an AC adapter (possibly included) or a battery supply. These should be available at any local electronics store. The serial cable should have been provided with your CMUcam2. Make sure that you have the CMOS sensor board connected to the CMUcam2 board so that it is in the same orientation as the picture shows on the cover of this manual.



First, connect the power. Make sure that the positive side of your power plug is facing away from the main components on the board. If the camera came with an AC adapter, make sure that the connector locks into the socket correctly.

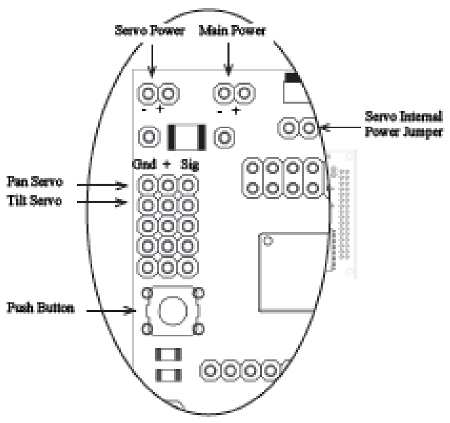
Now that the camera has power, connect the serial link between the camera and your computer. This link is required initially so that you can test and focus your camera. The serial cable should be connected so that the ribbon part of the cable faces away from the board. You must also connect the serial pass through jumper.

Check to make sure that the clock jumper is connected. This allows the clock to actively drive the processor.

Once everything is wired up, try turning the board on. The power LED should illuminate green and only one LED should remain on. Both LEDs turn on upon startup, and one turns off after the camera has been sucessfully configured.

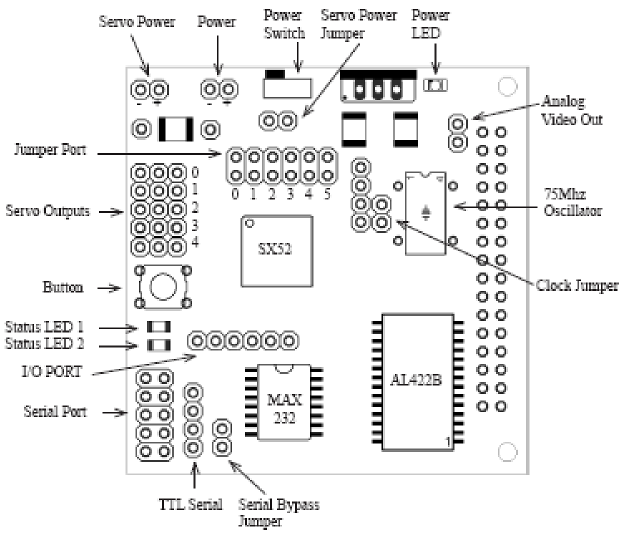
Demo Mode

Demo mode causes the camera to call track window and then drive two standard hobby servos towards the object being tracked. This can be initiated autonomously at startup. First you need to plug a pan and/or tilt servo into servo ports 0 and 1. Servo port 0 is for the pan, while 1 is for the tilt. Next, make sure that the servos are being powered by either the internal servo power jumper or by an external power source. While holding down the push button, turn the camera on. The tracking LED should begin rapidly blinking. Immediately release the push button and wait for the LED to stop blinking. Next, point the camera at a colored object and press the push button again. This should grab the color of the object and begin automatically servoing towards it. If the servos appear to be driving in the reverse direction, add the appropriate servo direction jumper. During the period when the LED is blinking, the camera is adjusting to the light conditions in the room. Try not to hold the object in front of the camera while this is occurring. Experiment with different colors and lighting. You will notice that some work much better than others.



The following steps are performed during power up in demo mode:
1. RS is sent to the camera
2. Pause 5 seconds while blinking the LED to allow the camera to stabilize
3. The Camera register string "CR 18 32 19 32" is sent.
4. Auto Servo Mode is enabled.
5. TW is called.

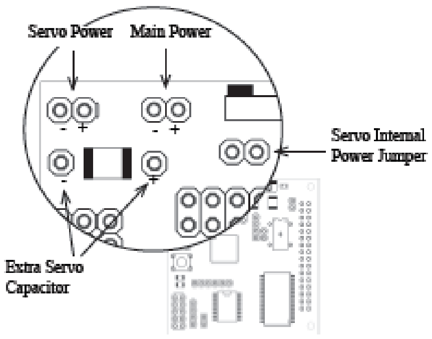
Board Layout



Ports

Power

The input power to the board goes through a 5 volt regulator. It is ideal to supply the board with between 6 and 15 volts of DC power that is capable of supplying at least 200 milliamperes of current.



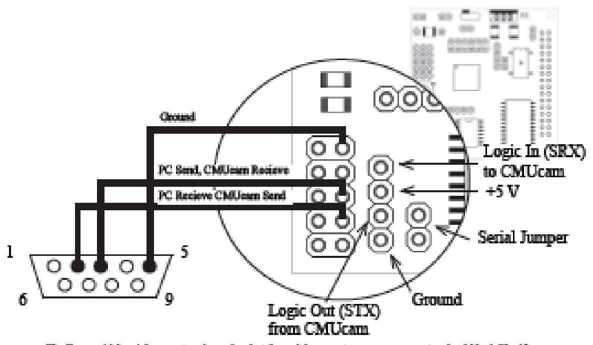
The servos can either be powered by internal power, or by the external servo power connector. To run them off of internal power, connect a jumper across the "servo internal power jumper". To run them off of external power, leave the jumper open, and connect another 5volt supply to the servo power connector. Do not connect an external servo supply while the servo power jumper is in place. If the servos are drawing too much power or seem to be noisy, try soldering a large valued capacitor across the "Extra Servo Capacitor" connections. The external servo power is not switched by the main power switch.

Serial Port

The CMUcam2 has a standard level shifted serial port to talk to a computer as well as a TTL serial port for talking to a microcontroller.

The level shifted serial port only uses 3 of the 10 pins. It is in a 2x5 pin configuration that fits a standard 9 pin ribbon cable clip-on serial sockets and 10 pin female clip on serial headers that can both attach to a 10 wire ribbon cable. If this initially does not work, try flipping the direction that the ribbon cable connects to the CMUcam2 board. Make sure the serial jumper is in place when you use this mode.

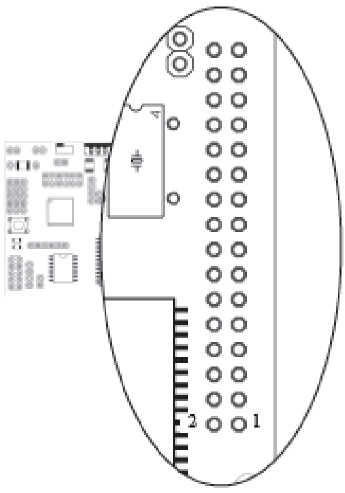
The TTL connector can be used to talk to a micrcontroller without the use of a level shifting chip. It operates between 0 and 5 volts. Remove the Serial Jumper when you use this mode.



The Trapezoidal serial connector shown is what the serial connector on your computer should look like if drawn in an annoying line art drawing program.

Camera Bus

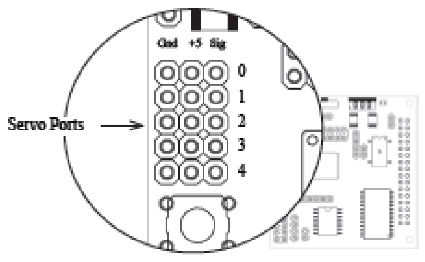
This bus interfaces with the CMOS camera chip. The CMOS camera board is mounted parallel to the processing part of the board and connects starting at pin 1. The female camera header should be soldered on the back of the board.



1-8	Y0-Y7	Digital Output Y Bus
9	PWDN	Power Down Mode
10	RST	Reset
11	SDA	I2C Serial Data
12	FODD	Odd Field Flag
13	SCL	I2C Serial Clock
14	HREF	Horizontal Ref
15	AGND	Analog Ground
16	VSYNC	Vertical Sync
17	AGND	Analog Ground
18	PCLK	Pixel Clock
19	EXCLK	External Clock
20	VCC	+5 VDC
21	AGND	Analog Ground
22	VCC	+5 VDC
23-30	UV0-UV7	Digital Output UV BUS
31	GND	Common Ground
32	VTO	Video Out (75Ohm)

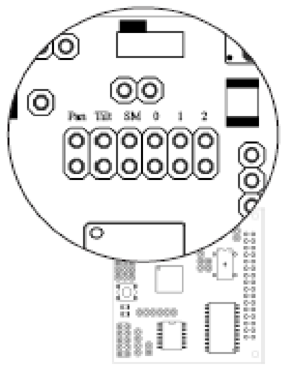
Servo Port

The CMUcam2 has the ability to control 5 servos. This can be useful if you do not wish to use a separate servo controller. The servo port can also be used as a general purpose digital outputs.



Configuration Jumpers

The jumpers can be used to set the camera's baudrates or configure various modes of operation.



Jumpers 0 1 and 2 set the camera into the following baudrates:

Baud Rate	Pin 0	Pin 1	Pin 2
115,200 Baud	_	_	_
57,600 Baud	_	_	X
38,400 Baud	_	X	_
19,200 Baud	_	X	X
9,600 Baud	X	_	_
4,800 Baud	X	_	X
2,400 Baud	X	X	_
1,200 Baud	X	X	X

X - jumper closed
_ - jumper open

Pan and Tilt Reverse Jumpers

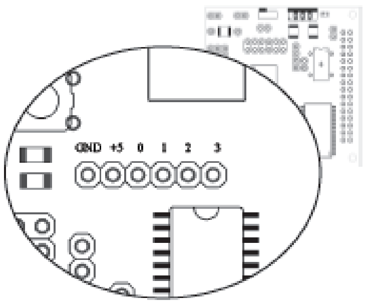
During Auto Servo Mode, or demo mode it may be necessary to reverse the direction of the pan or tilt servo. Connecting the pan and/or tilt jumper will cause auto servo mode to send the opposite commands to each servo. Note, this only works for auto servo mode, and not for normal servo operations.

Slave Mode Jumper

The CMUcam2 supports a mode of operation that allows multiple boards to process data from the same camera. If a PC104 style pass-through header is used instead of the standard double row female header, it is possible to rack multiple boards along the same camera bus. Upon startup, if the "SM" jumper is set, the camera becomes a slave. Slave mode stops the camera board from being able to configure or interfere with the CMOS camera's settings. Instead it processes the format setup by the master vision board. When linking the buses together you must only have one master; all other boards should be setup to be in slave mode. In this current version of the system there is no message passing between boards other than the image data from the camera bus. This means you have to communicate to each slave board via a separate serial link. This communication to the board should be identical to using a single CMUcam2. For example, you could have the master board tracking some color while the slave board could be told to get mean color data. Each board runs independently of one another and only the master can control camera registers.

Axuliary I/O

The CMUcam has 4 auxiliary Input Output ports that can be used for reading data from external devices. Note, that pin 3 is used for the sleep deeply command.

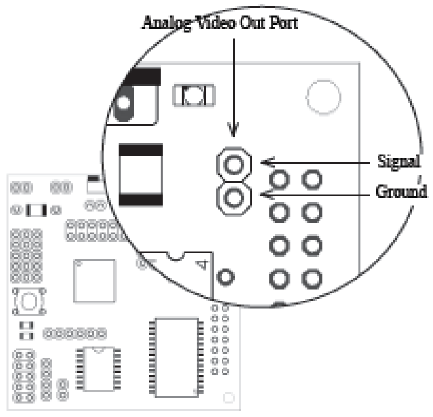


Analog Video Port

Using the OV6620 camera module, you will be able to get a PAL video signal from the analog port of the CMUcam2. This would sync up with any PAL monitor, but will not work with a standard NTSC monitor.

The OV7620 camera module will output a standard black and white NTSC video signal.

To use this output, it is necessary to keep the camera at its maximium frame rate (the default) and switch it into YCrCb mode in order to see the image on a monitor.



Functionally Grouped Command Listing

Buffer Commands

BM	Buffer Mode	30
RF	Read Frame	47

Camera Module Commands

CR	Camera Register	31
CP	Camera Power	32
CT	Camera Type	32

Data Rate Commands

DM	Delay Mode	33
PM	Poll Mode	46
PS	Packet Skip	47
RM	Raw Mode	48
PF	Packet Filter	46
OM	Output Packet Mask	45

Servo Commands

SV	Servo Position	53
SP	Servo Parameters	52
GS	Get Servo Position	36
SM	Servo Mask	51
SO	Servo Output	51

Image Windowing Commands

SF	Send Frame	50
DS	Down Sample	33
VW	Virtual Window	55
FS	Frame Stream	34
HR	HiRes Mode	38
GW	Get Window	38
PD	Pixel Difference	46

Auxiliary I/O Commands

GB	Get Button	35
GI	Get Auxiliary I/O	35
L0(1)	LED control	39

Color Tracking Commands

TC	Track Color	53
TI	Track Inverted	53
TW	Track Window	54
NF	Noise Filter	44
LM	Line Mode	40
GT	Get Tracking Parameters	37
ST	Set Tracking Parameters	52

Histogram Commands

GH	Get Histogram	35
HC	Histogram Config	38
HT	Histogram Track	39

Frame Differencing Commands

FD	Frame Difference	34
DC	Difference Channel	32
LF	Load Frame	39
MD	Mask Difference	44
UD	Upload Difference	55
HD	HiRes Difference	38
LM	Line Mode	40

Color Statistics Commands

GM	Get Mean	36
LM	Line Mode	40

System Level Commands

SD	Sleep Deeply	49
SL	Sleep	50
RS	Reset	49
GV	Get Version	37

Alphabetical Command Listing

BM	Buffer Mode	30
CR	Camera Register	31
CP	Camera Power	32
CT	Set Camera Type	32
DC	Difference Channel	32
DM	Delay Mode	33
DS	Down Sample	33
FD	Frame Difference	34
FS	Frame Stream	34
GB	Get Button	35
GH	Get Histogram	35
GI	Get Aux IO inputs	35
GM	Get Mean	36
GS	Get Servo Positions	36
GT	Get Tracking Parameters	37
GV	Get Version	37
GW	Get Window	38
HC	Histogram Configure	38
HD	High Resolution Difference	38
HR	HiRes Mode	38
HT	Set Histogram Track	39
L0 (1)	Led Control	39
LF	Load Frame to Difference	39

LM	Line Mode	40
MD	Mask Difference	44
NF	Noise Filter	44
OM	Output Packet Mask	45
PD	Pixel Difference	46
PF	Packet Filter	46
PM	Poll Mode	46
PS	Packet Skip	47
RF	Read Frame into Buffer	47
RM	Raw Mode	48
RS	Reset	49
SD	Sleep Deeply	49
SF	Send Frame	50
SL	Sleep Command	50
SM	Servo Mask	51
SO	Servo Output	51
SP	Servo Parameters	52
ST	Set Track Command	52
SV	Servo Position	53
TC	Track Color	53
TI	Track Inverted	53
TW	Track Window	54
UD	Upload Difference buffer	55
VW	Virtual Window	55