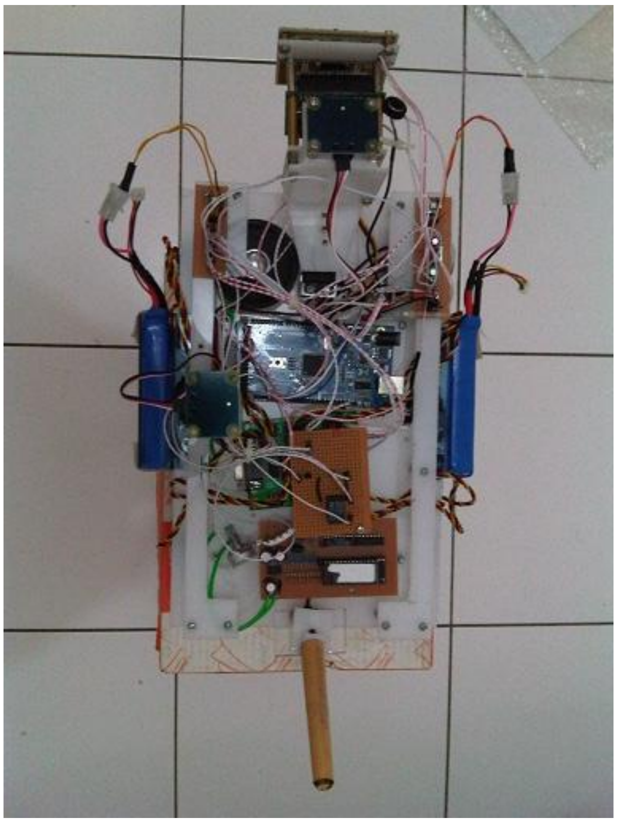
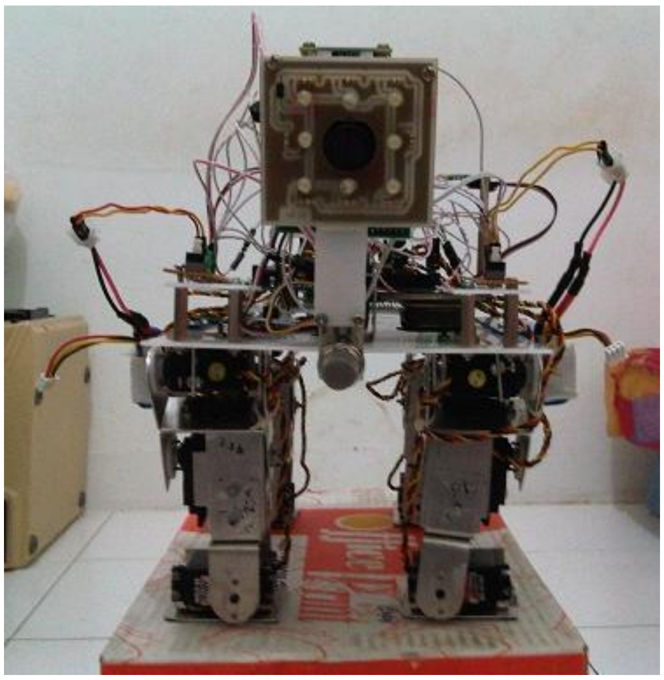
LAMPIRAN A
FOTO ROBOT ANJING

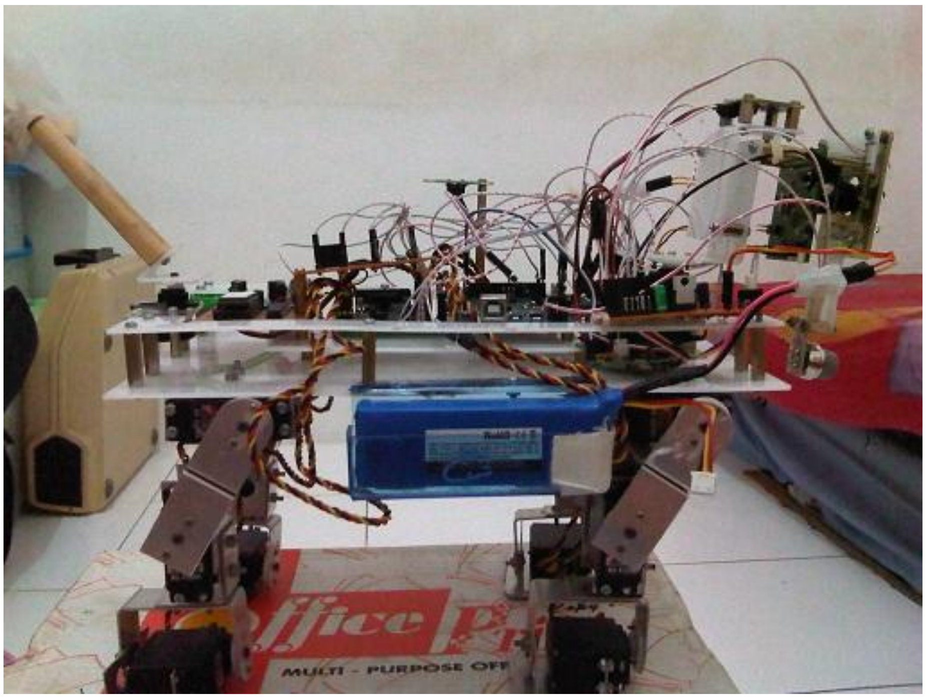
Tampak Atas



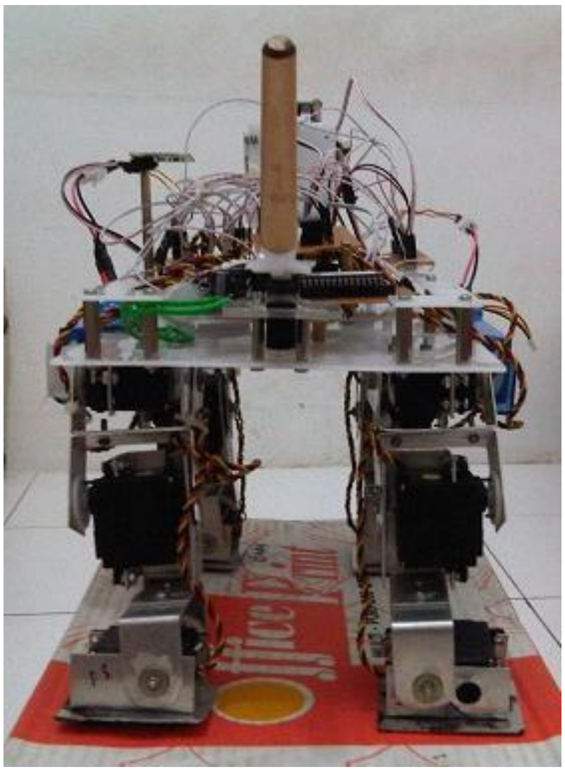
Tampak Depan



Tampak Samping



Tampak Belakang



LAMPIRAN B
PROGRAM PADA PENGENDALI MIKRO ARDUINO

PROGRAM UTAMA

PENGENDALI MIKRO ARDUINO

```
/***********************************************************************
   ROBOT ANJING
   CREATED BY RIKIAN WANJAYA DS
   0622020
   MARANATHA
***********************************************************************/

int x,y,a,b,c,n;
int mx, my, x1, x2, y1, y2, pixels, confidence;

void berdiri()
{
  Serial1.println("#13 P778 T1000");
  Serial1.println("#14 P778 T1000");
  Serial1.println("#15 P1722 T1000");
  Serial1.println("#2 P2000 T1000");
  Serial1.println("#1 P2222 T1000");
  Serial1.println("#0 P1278 T1000");
  Serial1.println("#29 P778 T1000");
  Serial1.println("#30 P1133 T1000");
  Serial1.println("#31 P1578 T1000");
  Serial1.println("#18 P2056 T1000");
  Serial1.println("#17 P2000 T1000");
  Serial1.println("#16 P1500 T1000");
  Serial1.println("#7 P1500 T1000");
  Serial1.println("#23 P1300 T1000");
}

void maju()
{
  Serial1.println("#0 P1556 T1000 #16 P1778 T1000");
  delay(1000);
  Serial1.println("#15 P2000 T1000 #31 P1856 T1000");
  delay(1000);
  Serial1.println("#2 P1389 T1000");
  delay(1000);
  Serial1.println("#1 P1611 T1000");
  delay(1000);
  Serial1.println("#15 P1722 T1000 #31 P1578 T1000");
  delay(1000);
  Serial1.println("#0 P1278 T1000 #16 P1500 T1000");
  delay(1000);
  Serial1.println("#15 P1333 T1000 #31 P1189 T1000");
  delay(1000);
  Serial1.println("#0 P1000 T1000 #16 P1222 T1000");
  delay(1000);
  Serial1.println("#29 P1500 T1000");
  delay(1000);
  Serial1.println("#30 P1744 T1000");
  delay(1000);
  Serial1.println("#13 P1500 T1000");
  delay(1000);
  Serial1.println("#14 P1389 T1000");
  delay(1000);
  Serial1.println("#0 P1278 T1000 #16 P1500 T1000");
  delay(1000);
  Serial1.println("#15 P1722 T1000 #31 P1578 T1000");
  delay(1000);
  Serial1.println("#0 P1556 T1000 #16 P1778 T1000");
  delay(1000);
  Serial1.println("#15 P2000 T1000 #31 P1856 T1000");
  delay(1000);
  Serial1.println("#18 P1444 T1000");
  delay(1000);
  Serial1.println("#17 P1389 T1000");
  delay(1000);
  Serial1.println("#15 P1722 T1000 #31 P1578 T1000");
  delay(1000);
  Serial1.println("#0 P1278 T1000 #16 P1500 T1000");
  delay(1000);
}

void kepalaekor()
{
  Serial1.println("#7 P1000 T1000 #23 P1600 T1000");
  delay(1000);
  Serial1.println("#7 P2000 T1000 #23 P1000 T1000");
  delay(1000);
  Serial1.println("#7 P1500 T1000");
  Serial1.println("#23 P1300 T1000");
}

void VRdetect()
{
   Serial2.print("b");
  delay(500);
  while(Serial2.available() > 0)
  {
    x = Serial2.read();  // buat ngeditek alat
    if ( x =='o')
      Serial.println ("Board Detected");
  }
}
void VRlanguage()
{
 Serial2.print("l");
  delay(10);
  Serial2.write(0x41); // english
  delay(100);
  while(Serial2.available() > 0)
  {
    x = Serial2.read();
    if (x == 'o')
      Serial.println("Set language Ok");
  }
}

void VRtimeout()
{
  Serial2.print("o");
  delay(10);
  Serial2.write(0x46); //  time out 5 detik : 0x41+5 english
  delay(100);
  while(Serial2.available() > 0)
  {
    x = Serial2.read();
    if (x == 'o')
      Serial.println("Set timeout Ok");
  }
}

void VRtrigger()
{
  Serial.println("Trigger Word...");
  Serial2.print("i");
  delay(10);
  Serial2.write(0x41); // trigger word
  delay(100);
  while(Serial2.available() == 0);
  x = Serial2.read();
  if (x == 't')
      Serial.println("timeout");
  else if (x == 'e')
      Serial.println("error");
  else if (x == 's')
```

```
     Serial.println("Recognized");
}

void VRwordset1()
{
    Serial.println("Word Set #1...");
    Serial2.print("i");
    delay(10);
    Serial2.write(0x42); // word set #1
    delay(100);
    while(Serial2.available() == 0);
    x = Serial2.read();
    if (x == 't')
        Serial.println("timeout");
    else if (x == 'e')
        Serial.println("error");
   else if (x == 3)
        {
          Serial.println("Run");
          maju();
        }
    else if (x == 6)
        {
          Serial.println("Stop");
          berdiri();
        }
       else
        Serial.println("???");
}

int cmuSet2(char *command)
{
  unsigned char byteCount;
  Serial3.print(command);
  Serial3.print("\r");
  delay(10);
  while(Serial3.available() == 0);
}

int cmuSet(char *command)
{
  int byteCount;
  Serial3.print(command);
  Serial3.print("\r");
  delay(10);
  while(Serial3.available() == 0);
  while(Serial3.read() != ':');
  Serial3.flush();
}

void camSetup()
{
   Serial.println("Ready");
   delay(1000);
   cmuSet("RS");
   delay(500);
   cmuSet("CR 18 44 19 33");
   delay(500);
   cmuSet("CR 18 40 19 32");
   delay(500);
   cmuSet("PM 1");  //polled mode on
   delay(500);
   cmuSet("RM 3");
//   wait for camera to settle
   delay(500);
   cmuSet2("TC 239 241 15 17 15 17");
   delay(1000);
   Serial.println("Go");
   Serial3.flush();
   delay(500);
}

void camTracking()
{
   unsigned char buffer[30];
   int nByte = 0;
   cmuSet2("TC");
   delay(100);
// 11 bytes are as follows -
// array position --> 0 1 2 3 4 5 6 7 8 9 10
// value --> 255 T mx my x1 y1 x2 y2 pixels confidence \r
//   Serial.print("Data: ");
//   Serial.println(Serial3.available(),DEC);
//   delay(500);
   while ( Serial3.available() > 0 )
   {
      buffer[nByte] = Serial3.read();
      nByte++;
      delay(1);
   }
   mx = buffer[2];
   my = buffer[3];
   x1 = buffer[4];
   y1 = buffer[5];
   x2 = buffer[6];
   y2 = buffer[7];
   pixels = buffer[8];
   confidence = buffer[9];
/*
   for ( int i = 0; i < 11; i++ ) //untuk menampilkan data array pada terimnal
   {
     Serial.print(buffer[i],DEC);
     Serial.print(" ");
   }
   Serial.println(" ");
   delay(1000);
*/

}

void setup()
{
  Serial.begin(9600);
  delay(500);
  Serial2.begin(9600);
  delay(500);
  Serial1.begin(9600);
  delay(500);
  Serial3.begin(19200);
  delay(500);
  pinMode(52,OUTPUT);
  digitalWrite(52,LOW);
  delay(500);
  camSetup();
  VRdetect();
  VRlanguage();
  VRtimeout();
  VRtrigger();
  Serial2.flush();
  if ( x == 's')
  {
    VRwordset1();
  }
  delay(500);

}

void loop()
```

```
{
 berdiri();
 a = analogRead(1);
 b = analogRead(2);
 c = analogRead(3);
 camTracking();
 if ( ( x2 - x1 > 20 ) && ( y2 - y1 > 50 ) )
 {
    kepalaekor();
   digitalWrite(52,HIGH);
 }
 if ( a > 800 )
   {
     kepalaekor();
     for ( n=0; n<5; n++)
     {
        maju();
        berdiri();
     }
   }
 delay(500);
 if ( b > 250 )
   {
     kepalaekor();
     digitalWrite(52,HIGH);
     delay (5000);
   }
 else
   {
     digitalWrite(52,LOW);
   }
 delay(500);
 if ( c > 800 )
   {
     kepalaekor();
     digitalWrite(52,HIGH);
     delay(5000);
   }
  delay(500);
  VRwordset1();
  delay(500);

}
```

LAMPIRAN C
DATASHEET

SENSOR SENTUH (*Phidgets Capacitive Touch Sensor*)

Product Manual

1129 - Touch Sensor

Technical Information

The Touch Sensor changes value from 0 to 1000 when it is touched. More specifically, this sensor is actually a capacitive change sensor. When the capacitance changes the sensor reports a thousand. If the sensor remains at 1000 for longer than 60 seconds, it will recalibrate back down to zero, regardless if the sensor is still being touched. This recalibration can also be done manually by unplugging and plugging the sensor back into the Interface kit.

On the bottom side of the Touch Sensor there is a small exposed metallic pad. A soldered connection can be made to the pad to increase the size and dimensions of the touchable area, such as attaching the sensor to a metallic object or some wire. Once the sensor is recalibrated, the sensor's value will increase to 1000 if the attached object is touched anywhere.

Although there is an exposed metallic pad on the bottom of the board, the pad does not have to be touched directly to activate the sensor - touching anywhere on the board will activate the sensor. The sensor can work as a close proximity sensor, sensing objects at a distance of up to 1/2" from the board in all directions without direct contact.

The Touch sensor will also work through a thickness of up to 1/2" of glass, plastic, or paper.

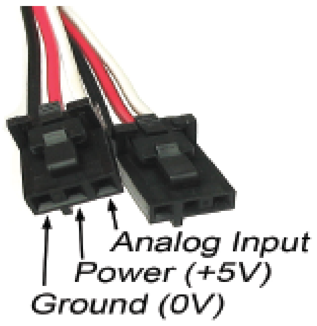
Other Interfacing Alternatives

If you want maximum accuracy, you can use the RawSensorValue property from the PhidgetInterfaceKit. To adjust a formula, substitute (SensorValue) with (RawSensorValue / 4.095)

If the sensor is being interfaced to your own Analog to Digital Converter and not a Phidget device, our formulas can be modified by replacing (SensorValue) with (Vin * 200). It is important to consider the voltage reference and input voltage range of your ADC for full accuracy and range.

Analog Input Cable Connectors

Each Analog Input uses a 3-pin, 0.100 inch pitch locking connector. Pictured here is a plug with the connections labeled. The connectors are commonly available - refer to the Table below for manufacturer part numbers.



Cable Connectors		
Manufacturer	**Part Number**	**Description**
Molex	50-57-9403	3 Position Cable Connector
Molex	16-02-0102	Wire Crimp Insert for Cable Connector
Molex	70543-0002	3 Position Vertical PCB Connector
Molex	70553-0002	3 Position Right-Angle PCB Connector (Gold)
Molex	70553-0037	3 Position Right-Angle PCB Connector (Tin)
Molex	15-91-2035	3 Position Right-Angle PCB Connector - Surface Mount

Note: Most of the above components can be bought at www.digikey.com

Device Specifications

Characteristic	Value
Current Consumption	110μA
Output Impedance	10K ohms
Supply Voltage	2.0VDC to 5.25VDC
Maximum Proximity Sensing Distance	1.27 cm (0.5")

Product History

Date	Board Revision	Comment
November 2009	0	Product Release

Support

Call the support desk at 1.403.282.7335 9:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00

or

E-mail us at: support@phidgets.com

SENSOR KAMERA (CMUCam3)

CMUcam3 Datasheet

September 22, 2007

1. INTRODUCTION

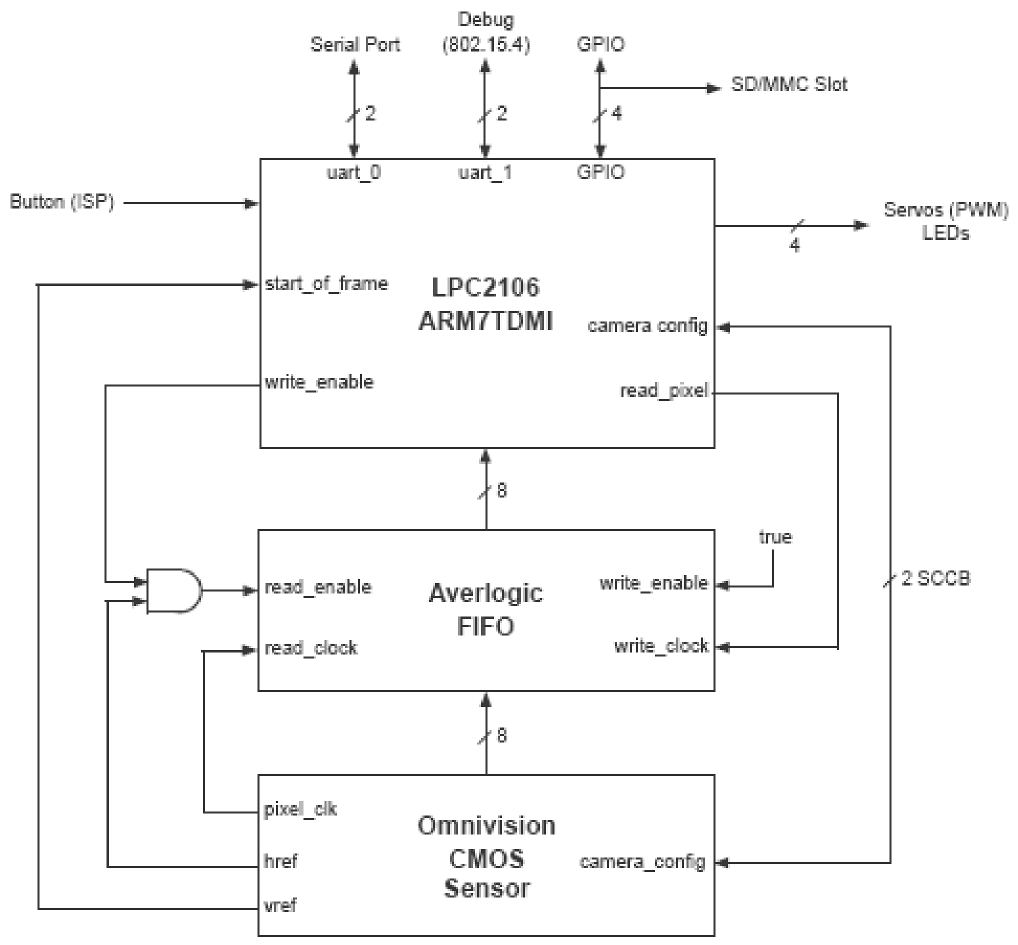
The CMUcam3 is an ARM7TDMI based fully programmable embedded computer vision sensor. The main processor is the Philips LPC2106 connected to an Omnivision CMOS camera sensor module. Custom C code can be developed for the CMUcam3 using a port of the GNU toolchain along with a set of open source libraries and example programs. Executables can be flashed onto the board using the serial port with no external downloading hardware required. Make sure to check www.cmucam.org for the most up to date documentaton and Quick-Start guides.

Features

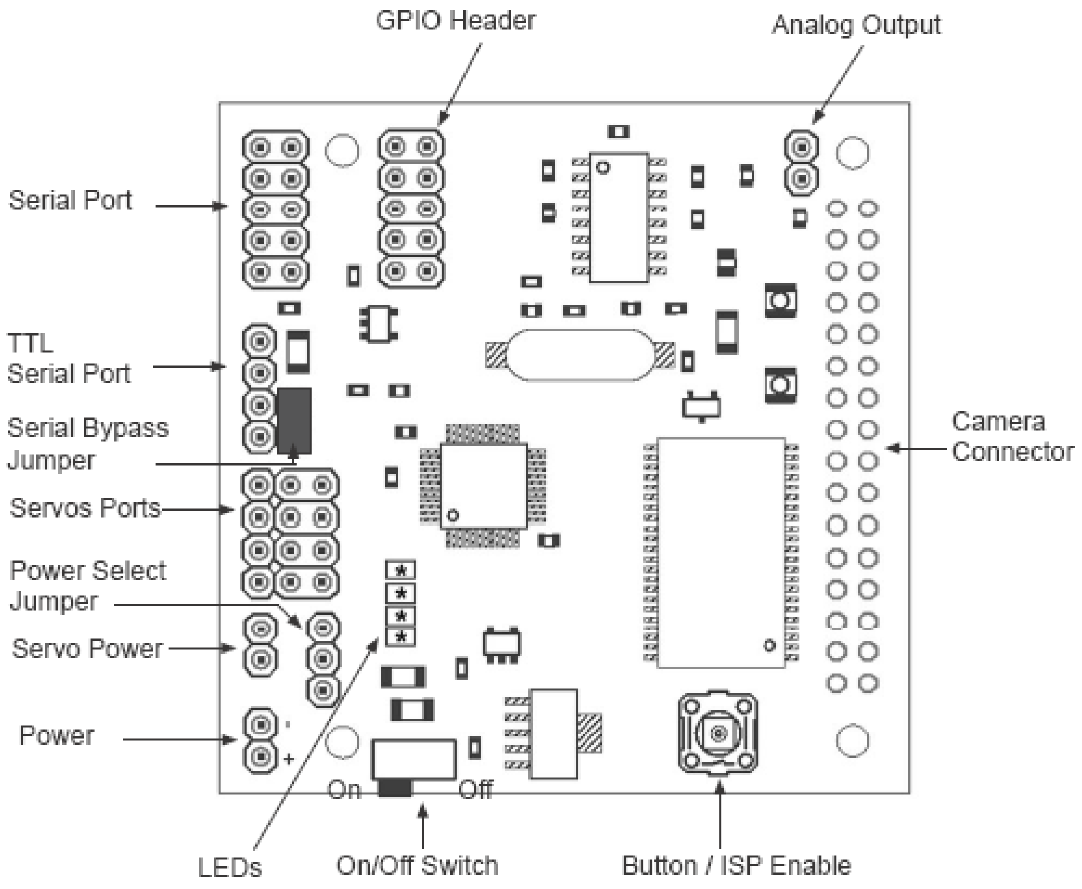
- CIF Resolution (352x288) RGB color sensor
- Open Source Development Environment for Windows and Linux
- MMC Flash Slot with FAT16 driver support
- Four-port Servo Controller
- Load Images into Memory at 26 Frames Per Second
- LUA light-weight language interpreter allows for rapid prototyping
- Software JPEG compression
- Basic Image Manipulation Library
    - Arbitrary Image Clipping
    - Image Downsampling
    - Mutable camera image properties
    - Threshold and Convolution Functions
    - RGB, YCrCb and HSV Color Space
- CMUcam2 Emulation
    - User defined color blobs
    - Frame differencing
    - Mean and variance data collection
    - Raw images dumps over serial
    - Histogram Generation
- B/W Analog video output (PAL or NTSC)*
- FIFO image buffer for multiple pass hi-res image processing
- Compatible Connector with Wireless Motes (Tmote Sky, FireFly, 802.15.4)

2. BLOCK DIAGRAM

Below is a high level block diagram of the major components found on the CMUcam3.



3. HARDWARE CONNECTIONS

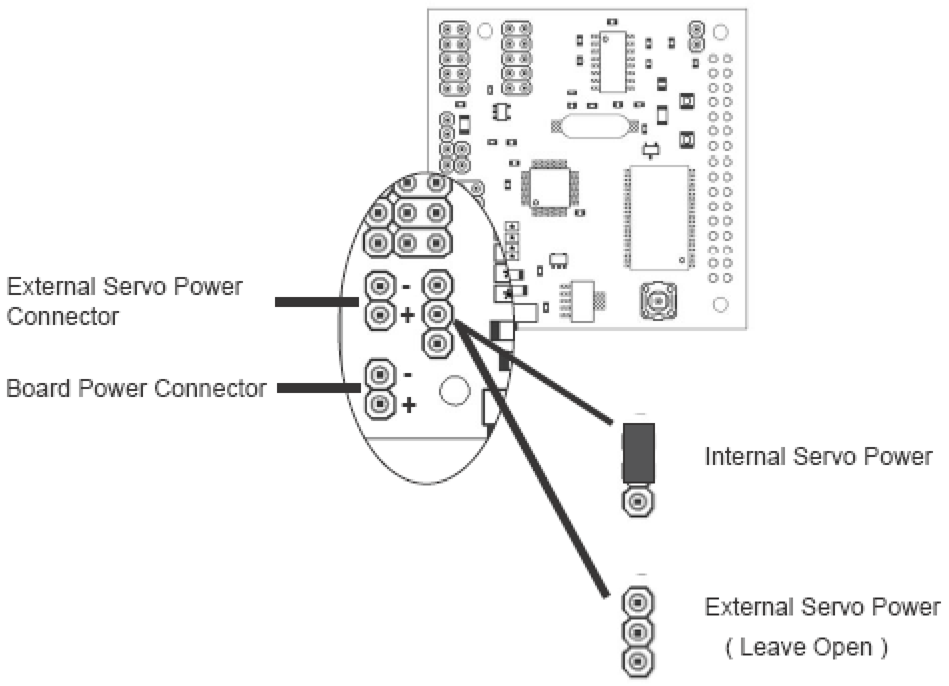


Power

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 150 milliamperes of current.

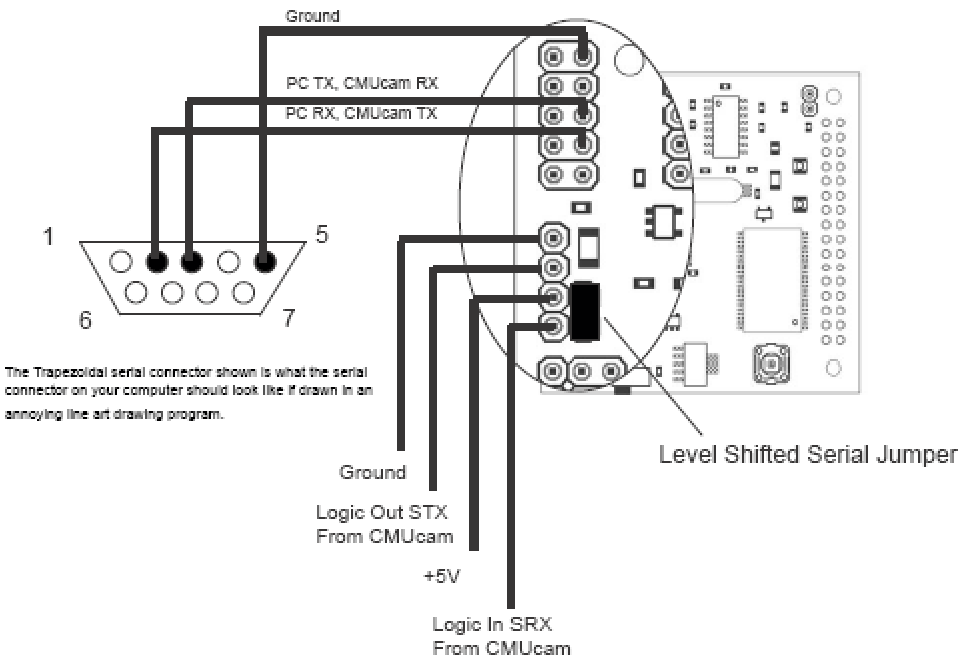
The servos can either be powered by internal power, or by the external servo power connector To run them off of external power, remove the internal servo power jumper like shown below. Then connect a second power supply to the “External Servo Power Connector”. To run off of internal power, connect the “Internal Servo Power” jumper and disconnect any external servo power.

Warning: powering the servos from internal power means that if the servos require more current than is available to power both servos and the processor, then the processor may reset or fail to operate at all. Running three or more servos off of internal power will likely not succeed.



Serial Port

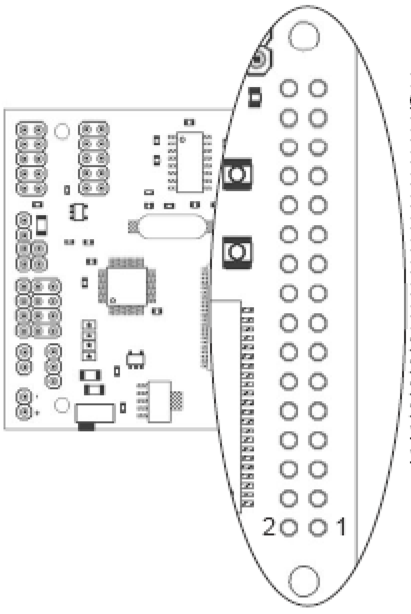
The CMUcam3 has a standard level shifted serial port to talk to a computer as well as a TTL serial port for talking to a microcontroller. The level shifted serial port only uses 3 of the 10 pins. It is in a 2x5 pin configuration that fits a standard 9 pin ribbon cable clip-on serial sockets and 10 pin 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. The TTL pins output between 0 and 3.3volts, but are 5 volt tolerant for input. Remove the Serial Jumper when you use this mode.


Ground
PC TX, CMUcam RX
PC RX, CMUcam TX
1
5
6
7
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.
Ground
Logic Out STX From CMUcam
+5V
Logic In SRX From CMUcam
Level Shifted Serial Jumper

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 main CMUcam3 board.

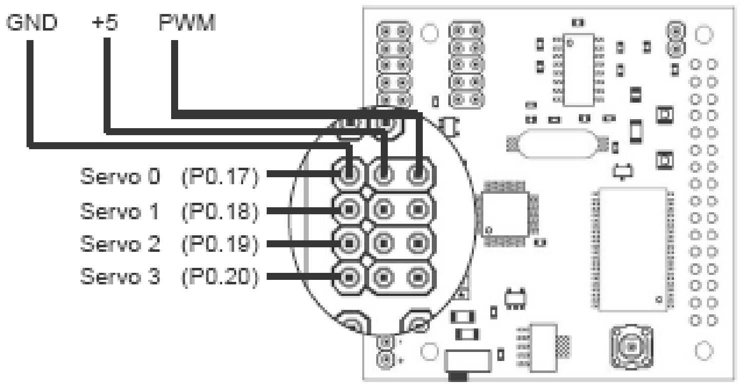
The CMUcam3 currently works with the OV6620 and OV7630 camera modules.



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 CMUcam3 has the ability to control 4 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.

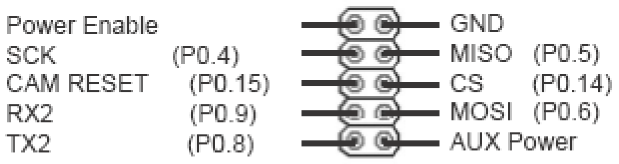


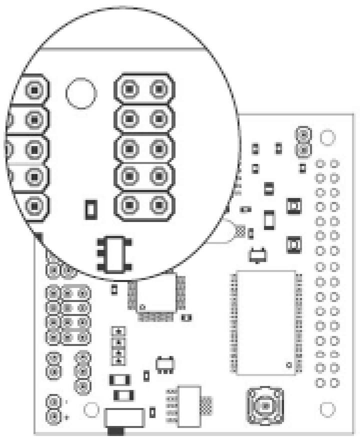
Expansion Port GPIO

The general purpose I/O header allows access to the second UART, various power control pins and the SPI pins.

Power Enable - When pulled low, the main CMUcam3 regulator is disabled causing the board to draw less than 0.01uA. All devices are shutoff and lose all active state information. When the line is released or pulled high, the board will reboot. By default, the pin is internally pulled high.

AUX Power - This pin can be configured to either externally power the board, or power an expansion board. By default, the pin is connected to the 3.3volt internal supply. By removing resistor R11 and adding a jumper resistor in place of R6, the pin is connected to the main power before the 5 volt regulator.





CAM RESET - This pin can be used as external I/O if the camera state is not required. Normally this pin resets the camera module and should not be used.

TX2 - The transmit pin on UART2 is not level shifted and hence can not be directly connected to a PC or none TTL external device. This pin can also be used as GPIO.

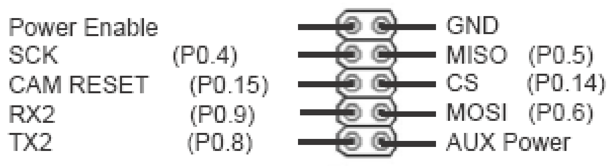
RX2 - The receive pin on UART2 is not level shifted and hence can not be directly connected to a PC or none TTL external device. This pin can also be used as GPIO.

CS - On reboot, if this pin is held low, the LPC2106 will enter bootstrap mode. Reboot can be externally induced by pulsing the power enable pin. Normally this pin will be controlled by the MMC driver. This pin can also be used as GPIO or as the SPI chip select when an MMC card is not inserted.

MOSI - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as an SPI output when an MMC card is not inserted.

MISO - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as an SPI input when an MMC card is not inserted.

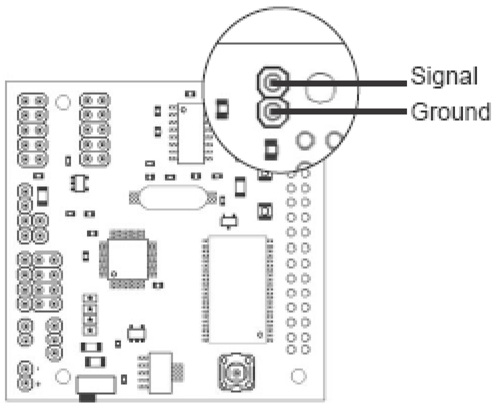
SCK - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as the SPI clock pin when an MMC card is not inserted.

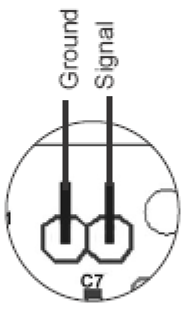


Analog Output Port

Using the OV6620 camera module, you will be able to get a PAL video signal from the analog port of the CMUcam3. 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 maximum frame rate (the default) and switch it into YCrCb mode in order to see the image on a monitor.





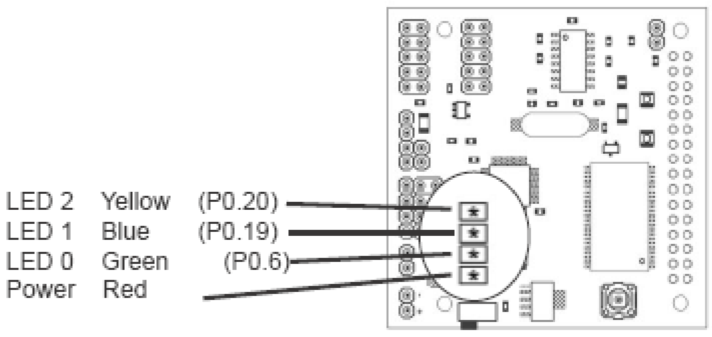
Board Revision B

LEDs

LED 0 - This pin is shared with the MOSI pin and should be used only when CS is disabled while an MMC card is inserted.

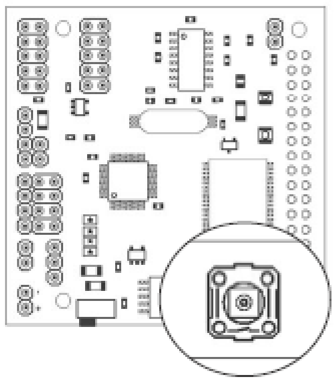
LED 1 - This pin is shared with the Servo 2 pin. When using Servo 2, the LED will flash.

LED 2 - This pin is shared with the Servo 3 pin. When using Servo 3, the LED will flash.



ISP Button

When held down on power up, the ISP button will enable the built in LPC2106 bootloader. After the processor has started up, the button can be read as normal GPIO. It is internally pulled high, and set low when depressed. The button shares the CS pin (P0.14) with the MMC. When using the MMC, CS is active low and hence the button cannot adversely affect data transfers.



4. HARDWARE CHARACTERISTICS

Power State	Active Current	Idle Current	Voltage
All Active	130mA	25mA	5 V
External Regulator Disabled	n/a	0.01uA	n/a
CPU (@60Mhz)	30mA	10uA	1.8 V
CPU Peripherals	30	10uA	3.3 V
CMOS camera	25mA	10uA	5 V
MAX232	8mA	n/a	3.3 V
FIFO	52mA	14mA	3.3 V
MMC card	4mA	4mA	3.3 V
misc	10mA	10mA	3.3 V

Table 1: Power consumption information for various components.

Component		Description
CPU		
	RAM	64KB
	ROM	128KB (8KB taken by bootloader)
	frequency	(14-60 Mhz)
FIFO		
	Capacity	1MB
	max rate	50 FPS
CMOS camera		
	max resolution	352x288
	color depth	8bits per pixel
Misc		
	max serial rate	115,200 bits per second
	full resolution image load and pixel touch rate	26 FPS
	servo frequency	50 Hz

Table 2: CMUcam3 Characteristics

Testing the Firmware

Once you have set the board up and downloaded the firmware, a good way to test the system is to connect it to the serial port of a computer.

Step 1: If one does not already exist, build a serial and/or power cable

Step 2: Plug both of them in.

Step 3: Open the terminal emulator of your choice.

Step 4: Inside the terminal emulator set the communication protocol to 115,200 Baud, 8 Data bits, 1 Stop bit, no parity, local echo on, no flow control and if possible turn on "add line feed" (add \n to a received \r). These setting should usually appear under "serial port" or some other similar menu option.

Step 5: Turn on the CMUcam2 board; the Power LED should light up and only one of the two status LEDs should remain on.

Step 6: You should see the following on your terminal emulator:

```
CMUcam2 v1.0 c6
:
```

If you have seen this, the board was able to successfully configure the camera and start the firmware.

Step 7: Type gv followed by the enter key. You should see the following:

```
:gv
ACK
CMUcam2 v1.0 c6
:
```

This shows the current version of the firmware. If this is successful, your computer's serial port is also configured correctly and both transmit and receive are working.

Serial Commands

The serial communication parameters are as follows:

- 1,200 to 115,200 Baud
- 8 Data bits
- 1 Stop bit
- No Parity
- No Flow Control (Not Xon/Xoff or Hardware)

All commands are sent using visible ASCII characters (123 is 3 bytes "123"). Upon a successful transmission of a command, the ACK string should be returned by the system. If there was a problem in the syntax of the transmission, or if a detectable transfer error occurred, a NCK string is returned. After either an ACK or a NCK, a \r is returned. When a prompt ('\r' followed by a ':') is returned, it means that the camera is waiting for another command in the idle state. White spaces do matter and are used to separate argument parameters. The \r (ASCII 13 carriage return) is used to end each line and activate each-command. If visible character transmission causes too much overhead, it is possible to use varying degrees of raw data transfer.

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

\r

This command is used to set the camera board into an idle state. Like all other commands, you should receive the acknowledgment string "ACK" or the not acknowledge string "NCK" on failure. After acknowledging the idle command the camera board waits for further commands, which is shown by the ':' prompt. While in this idle state a \r by itself will return an "ACK" followed by \r and : character prompt. This is how you stop the camera while in streaming mode.

Example of how to check if the camera is alive while in the idle state:

```
:
ACK
:
```

BM active \r

This command sets the mode of the CMUcam's frame buffer. A value of 0 (default) means that new frames are constantly being pushed into the frame buffer. A value of 1, means that only a single frame remains in the frame buffer. This allows multiple processing calls to be applied to the same frame. Instead of grabbing a new frame, all commands are applied to the current frame in memory. So you could get a histogram on all three channels of the same image and then track a color or call get mean and have these process a single buffered frame. Calling **RF** will then read a new frame into the buffer from the camera. When **BM** is off, **RF** is not required to get new frames.

Example of how to track multiple colors using buffer mode:

```
:BM 1
ACK
:PM 1
ACK
:TC 200 240 0 30 0 30
ACK
T 20 40 10 30 30 50 20 30
:RF
ACK
:TC 0 30 200 240 0 30
ACK
T 30 50 20 40 40 60 22 31
```

CR [reg1 value1 [reg2 value2 ... reg16 value16]]\r

This command sets the Camera's internal Register values directly. The register locations and possible settings can be found in the Omnivision CMOS camera documentation. All the data sent to this command should be in decimal visible character form unless the camera has previously been set into raw mode. It is possible to send up to 16 register-value combinations. Previous register settings are not reset between CR calls; however, you may overwrite previous settings. Calling this command with no arguments resets the camera and restores the camera registers to their default state. This command can be used to hard code gain values or manipulate other low level image properties.

Register		Value	Effect
5	Contrast	0-255	
6	Brightness	0-255	
18	Color Mode		
		36	YCrCb Auto White Balance On
		32	YCrCb Auto White Balance Off
		44	RGB Auto White Balance On
		40	*RGB Auto White Balance Off
17	Clock Speed		
		0	*50 fps
		1	26 fps
		2	17 fps
		3	13 fps
		4	11 fps
		5	9 fps
		6	8 fps
		7	7 fps
		8	6 fps
		10	5 fps
19	Auto Exposure		
		32	Auto gain off
		33	*Auto gain on

*** indicates the default state**

Example of switching into YCrCb mode with White Balance off

```
:CR 18 32
ACK
:
```

CP boolean \r

This command toggles the **C**amera module's **P**ower. A value of 0, puts the camera module into a power down mode. A value of 1 turns the camera back on while maintaining the current camera register values. This should be used in situations where battery life needs to be extended, while the camera is not actively processing image data. Images in the frame buffer may become corrupt when the camera is powered down.

CT boolean \r

This command toggles the **C**amera **T**ype while the camera is in slave mode. Since the CMUcam2 can not determine the type of the camera without communicating with the module, it is not possible for it to auto-detect the camera type in slave mode. A value of 0, sets the CMUcam2 into ov6620 mode. A value of 1 sets it into ov7620 mode. The default slave mode startup value assumes the ov6620.

DC value \r

This command sets the **C**hannel that is used for frame **D**ifferencing commands. A value of 0, sets the frame differencing commands LF and FD to use the red (Cr) channel. A value of 1 (default) sets them to use the green (Y) channel, and 2 sets them to use the blue (Cb) channel.

DM value \r

This command sets the **D**elay **M**ode which controls the delay between characters that are transmitted over the serial port. This can give slower processors the time they need to handle serial data. The value should be set between 0 and 255. A value of 0 (default) has no delay and 255 sets the maximum delay. Each delay unit is equal to the transfer time of one bit at the current baud rate.

DS x_factor y_factor \r

This command allows **D**own **S**ampling of the image being processed. An x_factor of 1 (default) means that there is no change in horizontal resolution. An x_factor of 2, means that the horizontal resolution is effectively halved. So all commands, like send frame and track color, will operate at this lower down sampled resolution. This gives you some speed increase and reduces the amount of data sent in the send frame and bitmap linemodes without clipping the image like virtual windowing would. Similarly, the y_factor independently controls the vertical resolution. (Increasing the y_factor downsampling gives more of a speed increase than changing the x_factor.) The virtual window is reset to the full size whenever this command is called.

Example of down sampling the resolution by a factor of 2 on both the horizontal and vertical dimension.

```
:DS 2 2
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

FD threshold \r

This command calls Frame Differencing against the last loaded frame using the LF command. It returns a type T packet containing the middle mass, bounding box, pixel count and confidence of any change since the previously loaded frame. It does this by calculating the average color intensity of an 8x8 grid of 64 regions on the image and comparing those plus or minus the user assigned *threshold*. So the larger the threshold, the less sensitive the camera will be towards differences in the image. Usually values between 5 and 20 yield good results. (In high resolution mode a 16x16 grid is used with 256 regions.)

FS boolean \r

This command sets the Frame Streaming mode of the camera. A value of 1, enables frame streaming, while a 0 (default) disables it. When frame streaming is active, a send frame command will continuously send frames back to back out the serial connection.

GB \r

This command Gets a Button press if one has been detected. This command returns either a 1 or a 0. If a 1 is returned, this means that the button was pressed sometime since the last call to Get Button. If a 0 is returned, then no button press was detected.

GH <channel> \r

This command Gets a Histogram of the *channel* specified by the user. The histogram contains 28 bins each holding the number of pixels that occurred within that bin's range of color values. So bin 0 on channel 0 would contain the number of red pixels that were between 16 and 23 in value. If no arguments are given, get histogram uses the last channel passed to get histogram. If get histogram is first called with no arguments, the green channel is used. The value returned in each bin is the number of pixels in that bin divided by the total number of pixels times 256 and capped at 255.

GI \r

This command Gets the auxiliary I/O Input values. When get inputs is called, a byte is returned containing the values of the auxiliary IO pins. This can be used to read digital inputs connected to the auxiliary I/O port. The aux I/O pins are internally lightly pulled high. See page 22 for pin numbering. Note that the pins are pulled up internally by the processor.

Example of how to read the auxiliary I/O pins. (in this case, pins 0 and 1 are high, while pins 2 and 3 are low).

```
:GI
3
ACK
:
```

GM \r

This command will Get the Mean color value in the current image. If, optionally, a subregion of the image is selected via virtual windowing, this function will only operate on the selected region. The mean values will be between 16 and 240 due to the limits of each color channel on the CMOS camera. It will also return a measure of the average absolute deviation of color found in that region. The mean together with the deviation can be a useful tool for automated tracking or detecting change in a scene. In YCrCb mode RGB maps to CrYCb.

This command returns a Type S data packet that by default has the following parameters:

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation\r

Example of how to grab the mean color of the entire window:

```
:SW 1 1 40 143
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

GS servo \r

This command will Get the last position that was sent to the Servos.

Example of how to use get servo:

```
:GS 1
ACK
128
:
```

GT \r

This command Gets the current Track color values. This is a useful way to see what color values track window is using.

This example shows how to get the current tracking values:

```
:TW
ACK
T 12 34 ....
:GT
ACK
200 16 16 240 20 20
:
```

GV \r

This command Gets the current Version of the firmware and camera module version from the camera. It returns an ACK followed by the firmware version string. c6 means that it detects an OV6620, while c7 means that it detected an OV7620.

Example of how to ask for the firmware version and camera type:

```
:GV
ACK
CMUcam2 v1.00 c6
```

GW \r

This command Gets the current virtual Windowing values. This command allows you to confirm your current window configuration. It returns the x1, y1, x2 and y2 values that bound the current window.

HC #_of_bins scale \r

This command lets you Configure the Histogram settings. The first parameter takes one of three possible values. A value of 0 (default) will cause GH to output 28 bins. A value of 1 will generate 14 bins and a value of 2 will generate 7 bins. The scale parameter (default 0) allows you to better examine bins with smaller counts. Bin values are scaled by 2^{scale} where scale is the second parameter of the command.

#_of_bins

Input	Bins
0	28
1	14
2	7

HD boolean \r

This command enables or disables HiRes frame Differencing. A value of 0 (default) disables the high resolution frame differencing mode, while a value of 1 enables it. When enabled, frame differencing will operate at 16x16 instead of 8x8. The captured image is still stored internally at 8x8. The extra resolution is achieved by doing 4 smaller comparisons against each internally stored pixel. This will only yield good results when the background image is relatively smooth, or has a uniform color.

HR state \r

This sets the camera into HiRes mode. This is only available using the OV6620 camera module. A *state* value of 0 (default) gives you the standard 88x143, while 1 gives you 176x287. HiRes mode truncates the image to 176x255 for tracking so that the value does not overflow 8 bits.

HT boolean \r

This command enables or disables Histogram Tracking. When histogram tracking is enabled, only values that are within the color tracking bounds will be displayed in the histograms. This allows you to select exact color ranges giving you more detail, and ignoring any other background influences. A value of 0 (default) will disable histogram tracking, while a value of 1 will enable it. Note that the tracking noise filter applies just like it does with the TC and TW commands.

L0 boolean \r
L1 boolean \r

These commands enable and disable the two tracking LEDs. A value of 0 will turn the LED off, while a value of 1 will turn it on. A value of 2 (default) will leave the LED in automatic mode. In this mode, LED 1 turns on when the camera confidently detects an object while tracking and provides feedback during a send frame. In automatic mode, LED 0 does nothing, so it can be manually set.

LF \r

This command Loads a new Frame into the processor's memory to be differenced from. This does not have anything to do with the camera's frame buffer. It simply loads a baseline image for motion differencing and motion tracking.

LM type mode \r

This command enables Line Mode which transmits more detailed data about the image. It adds prefix data onto either T or S packets. This mode is intended for users who wish to do more complex image processing on less reduced data. Due to the higher rate of data, this may not be suitable for many slower microcontrollers. These are the different types and modes that line mode applies to different processing functions:

Type	**Mode**	**Effected Command**	**Description**
0	0	TC TW	Default where line mode is disabled
0	1	TC TW	Sends a binary image of the pixels being tracked
0	2	TC TW	Sends the Mean, Min, Max, confidence and count for every horizontal line of the tracked image.
1	0	GM	Default where line mode is disabled
1	1	GM	Sends the mean values for every line in the image
1	2	GM	Sends the mean values and the deviations for every line being tracked in the image
2	0	FD	Default where line mode is disabled
2	1	FD	Returns a bitmap of tracked pixels much like type 0 mode 0 of track color
2	2	FD	Sends the difference between the current image pixel value and the stored image. This gives you delta frame differenced images.
2	3	LF FD	This gives you the actual averaged value for each element in a differenced frame. It also returns these values when you load in a new frame. This can be used to give a very high speed gray scale low resolution stream of images.

Note, that the "mode" of each "type" of linemode can be controlled independently.

Data Packet Description

When raw mode is disabled all output data packets are in ASCII viewable format except for the F frame and prefix packets.

ACK

This is the standard acknowledge string that indicates that the command was received and fits a known format.

NCK

This is the failure string that is sent when an error occurred. The only time this should be sent when an error has not occurred is during binary data packets.

Type **F** data packet format:

```
1 Xsize Ysize 2 r g b r g b ... r g b r g b 2 r g b r g b ... r g b r g b 3
```

1 - new frame 2 - new row 3 - end of frame
RGB (CrYCb) ranges from 16 - 240
RGB (CrYCb) represents two pixels color values. Each pixel shares the red and blue.
176 cols of R G B (Cr Y Cb) packets (forms 352 pixels)
144 rows
To display the correct aspect ratio, double each column so that your final image is 352x144

Type **H** packet:

H bin0 bin1 bin2 bin3 ... bin26 bin27 \r

This is the return packet from calling get histogram (GH). Each bin is an 8 bit value that represents the number of pixels that fell within a set range of values on a user selected channel of the image.

Bin0 – number of pixels between 16 and 23
Bin1 – number of pixels between 24 and 31
.
.
.
Bin27 – number of pixels between 232 and 240

Type T packet:

```
T mx my x1 y1 x2 y2 pixels confidence\r
```

This is the return packet from a color tracking or frame differencing command.

mx - The middle of mass x value
my - The middle of mass y value
x1 - The left most corner's x value
y1 - The left most corner's y value
x2 - The right most corner's x value
y2 -The right most corner's y value
pixels –Number of Pixels in the tracked region, scaled and capped at 255: (pixels+4)/8
confidence -The (# of pixels / area)*256 of the bounded rectangle and capped at 255

Type S data packet format:

```
S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation \r
```

This is a statistic packet that gives information about the camera's view

Rmean - the mean Red or Cr (approximates r-g) value in the current window
Gmean - the mean Green or Y (approximates intensity) value found in the current window
Bmean - the mean Blue or Cb (approximates b-g) found in the current window
R*deviation* - the *deviation of red or Cr found in the current window
G*deviation*- the *deviation of green or Y found in the current window
B*deviation*- the *deviation of blue or Cb found in the current window

*deviation: The mean of the absolute difference between the pixels and the region mean.

Sensor Gas (MQ-4)

MQ-4 Semiconductor Sensor for Natural Gas

Sensitive material of MQ-4 gas sensor is SnO_2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-4 gas sensor has high sensitity to Methane, also to Propane and Butane. The sensor could be used to detect different combustible gas, especially Methane, it is with low cost and suitable for different application.

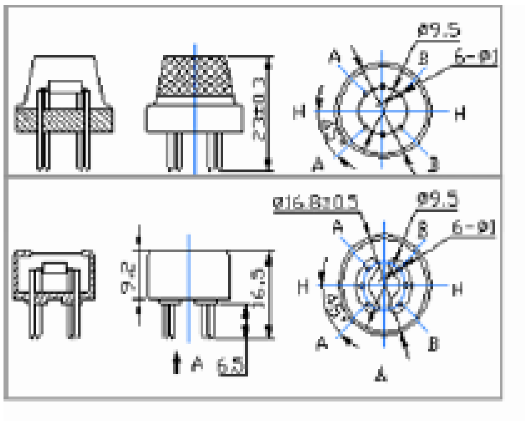
Character

* Good sensitivity to Combustible gas in wide range
* High sensitivity to Natural gas
* Long life and low cost
* Simple drive circuit

Application

* Domestic gas leakage detector
* Industrial Combustible gas detector
* Portable gas detector

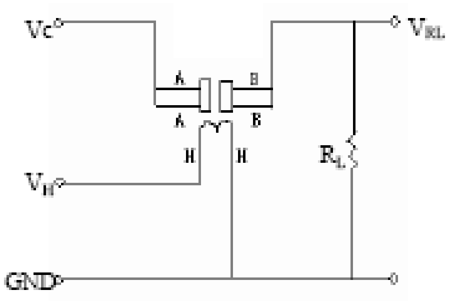
Configuration



Technical Data

Model No.			MQ-4
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Natural gas/ Methane
Concentration			300-10000ppm (Natural gas / Methane)
Circuit	Loop Voltage	V_c	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V ACorDC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 5000ppm CH_4)
	Sensitivity	S	Rs(in air)/Rs(5000ppm CH_4)≥5
	Slope	α	≤0.6($R_{5000ppm}/R_{3000ppm}$ CH_4)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V； V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage（VH）and test voltage（VC）. VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance （RL）whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps):

$Ps=Vc^2 \times Rs/(Rs+RL)^2$

Resistance of sensor(Rs): $Rs=(Vc/VRL-1) \times RL$

Sensitivity Characteristics

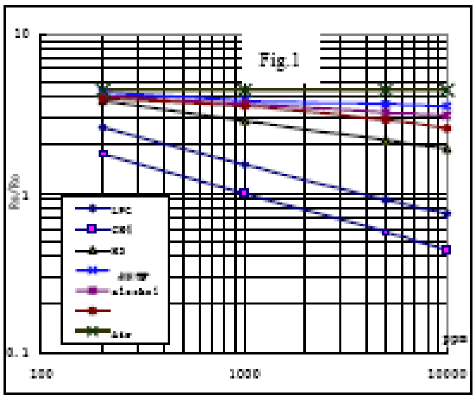


Fig.1 shows the typical sensitivity characteristics of the MQ-4, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm Methane. All test are under standard test conditions.

P.S.: Sensitivity to smoke is ignite 10pcs cigarettes in $8m^3$ room, and the output equals to 200ppm Methane

Influence of Temperature/Humidity

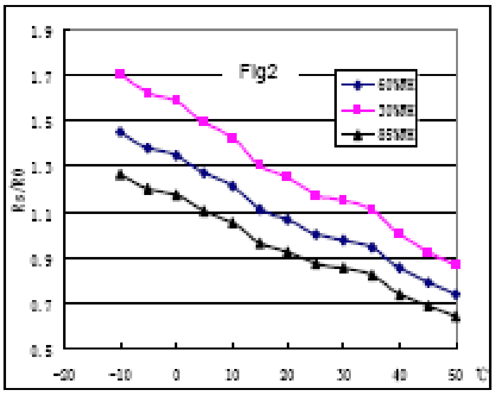
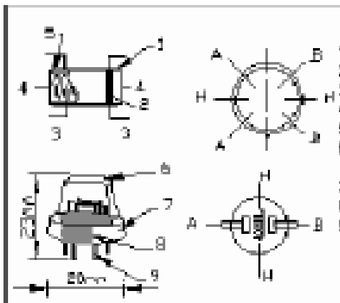


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 1000ppm Methane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Methane, 20℃/65%RH

Structure and configuration



	Parts	Materials
1	Gas sensing layer	SnO2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al2O3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Fig. 3

Structure and configuration of MQ-4 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-4 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

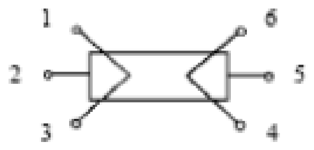
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then repture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine
2.7.2 Speed: 1-2 Meter/ Minute
2.7.3 Warm-up temperature：100±20℃
2.7.4 Welding temperature：250±10℃
2.7.5 1 time pass wave crest welding machine

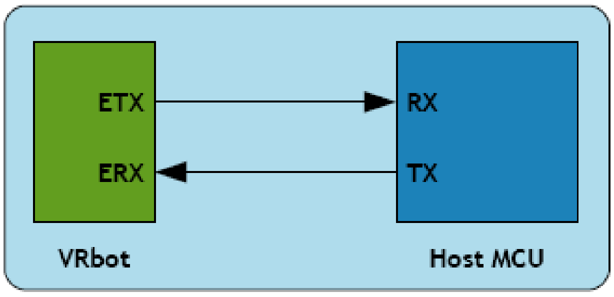
If disobey the above using terms, sensors sensitivity will be reduced.

Sensor Suara (VRBot *Voice Recognition Sensor*)

Protocol and Interface Basics

Communication with the VRbot module uses a standard UART interface compatible with 3.3-5V TTL logical levels, according to the powering voltage VCC.

A typical connection to an MCU-based host:



The initial configuration at power on is 9600 baud, 8 bit data, No parity, 1 bit stop. The baud rate can be changed later to operate in the range 9600 - 115200 baud.

The communication protocol only uses printable ASCII characters, which can be divided in two main groups:

- Command and status characters, respectively on the TX and RX lines, chosen among lower-case letters
- Command arguments or status details, again on the TX and RX lines, spanning the range of capital letters

Each command sent on the TX line, with zero or more additional argument bytes, receives an answer on the RX line in the form of a status byte followed by zero or more arguments.

There is a minimum delay before each byte sent out from the VRbot module to the RX line, that is initially set to 20 ms and can be selected later in the ranges 0 - 9 ms, 10 - 90 ms, 100 ms - 1 s. That accounts for slower or faster host systems and therefore suitable also for software-based serial communication (bit-banging).

The communication is host-driven and each byte of the reply to a command has to be acknowledged by the host to receive additional status data, using the *space* character. The reply is aborted if any other character is received and so there is no need to read all the bytes of a reply if not required.

Invalid combinations of commands or arguments are signaled by a specific status byte, that the host should be prepared to receive if the communication fails. Also a reasonable timeout should be used to recover from unexpected failures.

If the host does not send all the required arguments of a command, the command is ignored by the module, without further notification, and the host can start sending another command.

The module automatically goes to lowest power sleep mode after power on. To initiate communication, send any character to wake-up the module.

Command Details

Format of command strings accepted by the module. Please note that numeric arguments of command requests are mapped to upper-case letters (see the related section).

CMD_BREAK	
`'b'` (62h)	Abort recognition in progress if any or do nothing **Known issues:** In firmware ID 0, any other character received during recognition will prevent this command from stopping recognition, that will continue until timeout or other recognition results.
Expected replies: STS_SUCCESS, STS_INTERR	

CMD_SLEEP	
`'s'` (73h)	Go to the specified power-down mode
`[1]`	Sleep mode (0-8): 0 = wake on received character only 1 = wake on whistle or received character 2 = wake on loud sound or received character 3-5 = wake on double clap (with varying sensitivity) or received character 6-8 = wake on triple clap (with varying sensitivity) or received character
Expected replies: STS_SUCCESS	

CMD_KNOB	
`'k'` (6Bh)	Set SI knob to specified level
`[1]`	Confidence threshold level (0-4): 0= loosest:more valid results 2= typical value (default) 4= tightest:fewer valid results NOTE: knob is ignored for trigger words
Expected replies: STS_SUCCESS	

CMD_LEVEL	
`'v'` (76h)	Set SD level
`[1]`	Strictness control setting (1-5): 1 = easy, 2 = default, 5 = hard A higher setting will result in more recognition errors.
Expected replies: STS_SUCCESS	

CMD_LANGUAGE	
`'l'` (6Ch)	Set SI language
`[1]`	Language (0 = English, 1 = Italian, 2 = Japanese, 3 = German)
Expected replies: STS_SUCCESS	

CMD_TIMEOUT	
`'o'` (6Fh)	Set recognition timeout
`[1]`	Timeout (-1 = default, 0 = infinite, 1-31 = seconds)
Expected replies: STS_SUCCESS	

CMD_RECOG_SI	
`'i'` (69h)	Activate SI recognition from specified wordset
`[1]`	Wordset index (0-3)
Expected replies: STS_SIMILAR, STS_TIMEOUT, STS_ERROR	

CMD_TRAIN_SD	
`'t'` (74h)	Train specified SD/SV command
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Command position (0-31)
Expected replies: STS_SUCCESS, STS_RESULT, STS_SIMILAR, STS_TIMEOUT, STS_ERROR	

CMD_GROUP_SD	
`'g'` (67h)	Insert new SD/SV command
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Position (0-31)
Expected replies: STS_SUCCESS, STS_OUT_OF_MEM	

CMD_UNGROUP_SD	
`'u'` (75h)	Remove SD/SV command
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Position (0-31)
Expected replies: STS_SUCCESS	

CMD_RECOG_SD	
`'d'` (64h)	Activate SD/SV recognition
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
Expected replies: STS_RESULT, STS_SIMILAR, STS_TIMEOUT, STS_ERROR	

CMD_ERASE_SD	
`'e'` (65h)	Erase training of SD/SV command
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Command position (0-31)
Expected replies: STS_SUCCESS	

CMD_NAME_SD	
`'n'` (6Eh)	Label SD/SV command
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Command position (0-31)
`[3]`	Length of label (0-31)
`[4-n]`	Text for label (ASCII characters from `'A'` to `` '`' ``)
Expected replies: STS_SUCCESS	

CMD_COUNT_SD	
`'c'` (63h)	Request count of SD/SV commands in the specified group
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
Expected replies: STS_COUNT	

CMD_DUMP_SD	
`'p'` (70h)	Read SD/SV command data (label and training)
`[1]`	Group index (0 = trigger, 1-15 = generic, 16 = password)
`[2]`	Command position (0-31)
Expected replies: STS_DATA	

CMD_MASK_SD	
`'m'` (6Dh)	Request bit-mask of non-empty groups
Expected replies: STS_MASK	

CMD_RESETALL	
`'r'` (72h)	Reset all commands and groups
`'R'` (52h)	Confirmation character
Expected replies: STS_SUCCESS	

CMD_ID	
`'x'` (78h)	Request firmware identification
Expected replies: STS_ID	

CMD_DELAY	
`'y'` (79h)	Set transmit delay
`[1]`	Time (0-10 = 0-10 ms, 11-19 = 20-100 ms, 20-28 = 200-1000 ms)
Expected replies: STS_SUCCESS	

CMD_BAUDRATE	
`'a'` (61h)	Set communication baud-rate
`[1]`	Speed mode (1 = 115200, 2 = 57600, 3 = 38400, 6 = 19200, 12 = 9600)
Expected replies: STS_SUCCESS	

Status Details

Replies to commands follow this format. Please note that numeric arguments of status replies are mapped to upper-case letters (see the related section).

STS_MASK	
'k' (6Bh)	Mask of non-empty groups
[1-8]	4-bit values that form 32-bit mask, LSB first
In reply to: CMD_MASK_SD	

STS_COUNT	
'c' (63h)	Count of commands
[1]	Integer (0-31)
In reply to: CMD_COUNT_SD	

STS_AWAKEN	
'w' (77h)	Wake-up (back from power-down mode)
In reply to: Any character after power on or sleep mode	

STS_DATA	
'd' (64h)	Provide command data
[1]	Training information (0-7 = training count, +8 = SD/SV conflict, +16 = SI conflict)
[2]	Conflicting command position (0-31)
[3]	Length of label (0-31)
[4-n]	Text of label (ASCII characters from 'A' to '`')
In reply to: CMD_DUMP_SD	

STS_ERROR	
'e' (65h)	Signal recognition error
[1-2]	Two 4-bit values that form 8-bit error code (80h = NOTA, otherwise see FluentChip error codes)
In reply to: CMD_RECOG_SI, CMD_RECOG_SD, CMD_TRAIN_SD	

STS_INVALID	
'v' (76h)	Invalid command or argument
In reply to: Any invalid command or argument	

STS_TIMEOUT	
't' (74h)	Timeout expired
In reply to: CMD_RECOG_SI, CMD_RECOG_SD, CMD_TRAIN_SD	

STS_INTERR	
'i' (69h)	Interrupted recognition
In reply to: CMD_BREAK while in training or recognition	

STS_SUCCESS	
`'o'` (6Fh)	OK or no errors status
In reply to: CMD_BREAK, CMD_DELAY, CMD_BAUDRATE, CMD_TIMEOUT, CMD_KNOB, CMD_LEVEL, CMD_LANGUAGE, CMD_SLEEP, CMD_GROUP_SD, CMD_UNGROUP_SD, CMD_ERASE_SD, CMD_NAME_SD, CMD_RESETALL	

STS_RESULT	
`'r'` (72h)	Recognised SD/SV command or Training similar to SD/SV command
`[1]`	Command position (0-31)
In reply to: CMD_RECOG_SD, CMD_TRAIN_SD	

STS_SIMILAR	
`'s'` (73h)	Recognised SI word or Training similar to SI word
`[1]`	Word index (0-31)
In reply to: CMD_RECOG_SI, CMD_RECOG_SD, CMD_TRAIN_SD	

STS_OUT_OF_MEM	
`'m'` (6Dh)	Memory full error
In reply to: CMD_GROUP_SD	

STS_ID	
`'x'` (78h)	Provide firmware identification
`[1]`	Version identifier (0)
In reply to: CMD_ID	

Arguments Mapping

These are the characters used to represent integer values in the range -1 to 31 for command or status arguments.

ARG_MIN	
`'@'` (40h)	Minimum argument value (-1)

ARG_MAX	
`` '`' `` (60h)	Maximum argument value (+31)

ARG_ZERO	
`'A'` (41h)	Zero argument value (0)

ARG_ACK	
`' '` (20h)	Read more status arguments

Built-in Command Sets

In the tables below a list of all built-in commands for each supported language, along with group index (trigger or wordset), command index and language identifier to use with the communication protocol.

		Language			
		0	1	2	3
Trigger/Wordset	**Command Index**	**English (US)**	**Italian**	**Japanese**	**German**
0	0	robot	robot	ロボット	roboter
1	0	action	azione	アクション	aktion
	1	move	vai	ススメ	gehe
	2	turn	gira	マガレ	wende
	3	run	corri	ハシレ	lauf
	4	look	guarda	ミロ	schau
	5	attack	attacca	コーゲキ	attacke
	6	stop	fermo	トマレ	halt
	7	hello	ciao	こんにちわ	hallo
2	0	left	a sinistra	ヒダリ	nach_links
	1	right	a destra	ミギ	nach_rechts
	2	up	in alto	ウエ	hinauf
	3	down	in basso	シタ	hinunter
	4	forward	avanti	マエ	vorwärts
	5	backward	indietro	ウシロ	rückwärts
3	0	zero	zero	ゼロ	null
	1	one	uno	いち	eins
	2	two	due	二	zwei
	3	three	tre	サン	drei
	4	four	quattro	ヨン	vier
	5	five	cinque	ゴ	fünf
	6	six	sei	ロク	sechs
	7	seven	sette	ナナ	sieben
	8	eight	otto	ハチ	acht
	9	nine	nove	クュー	neun
	10	ten	dieci	ジュー	zehn

Error codes

In the table below a list of some (the most useful) error codes that may be returned by training or recognition commands.

03h	ERR_DATACOL_TOO_NOISY	too noisy
04h	ERR_DATACOL_TOO_SOFT	spoke too soft
05h	ERR_DATACOL_TOO_LOUD	spoke too loud
06h	ERR_DATACOL_TOO_SOON	spoke too soon
07h	ERR_DATACOL_TOO_CHOPPY	too many segments/too complex
11h	ERR_RECOG_FAIL	recognition failed
12h	ERR_RECOG_LOW_CONF	recognition result doubtful
13h	ERR_RECOG_MID_CONF	recognition result maybe
14h	ERR_RECOG_BAD_TEMPLATE	invalid SD/SV command stored in memory
17h	ERR_RECOG_DURATION	bad pattern durations
80h	ERR_NOT_A_WORD	recognized word is not in vocabulary

The first group of codes (03h – 07h) are due to errors in the way of speaking to the VRbot or disturbances in the acquired audio signal, that may depend on the surrounding environment.

The second group (11h – 13h) indicate an insufficient score of the recognized word (from lowest to highest). Acceptance of lower score results may be allowed by lowering the "knob" or "level" settings, respectively for built-in and custom commands (see CMD_KNOB and CMD_LEVEL).

A third group of codes (14h – 17h) reports errors in the stored commands, that may be due to memory corruption. We suggest you check power level and connections, then erase all the commands in the faulty group and train them again.

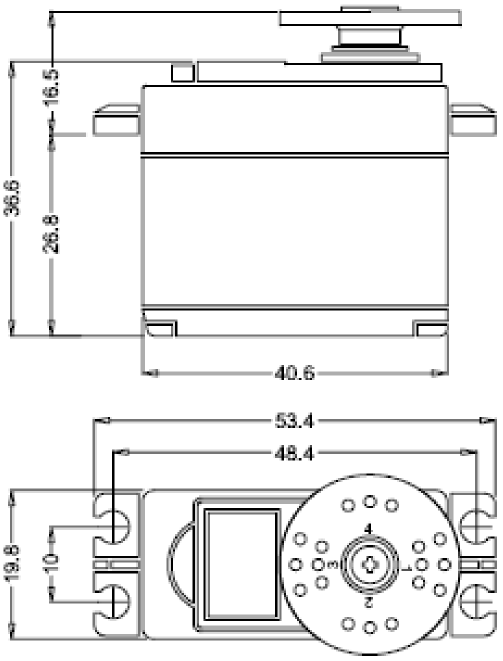
The last code (80h) means that a word has been recognized that is not in the specified built-in sets. This is due to how Speaker Independent recognition works and should be ignored.

Motor Servo Hitech HS-425HB

ANNOUNCED SPECIFICATION OF HS–425BB STANDARD DELUXE BALL BEARING SERVO

1.TECHNICAL VALUES

CONTROL SYSTEM	: +PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	: 4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	: -20 T0 +60°C	
TEST VOLTAGE	: AT 4.8V	AT 6.0V
OPERATING SPEED	: 0.21sec/60° AT NO LOAD	0.16sec/60° AT NO LOAD
STALL TORQUE	: 3.3kg.cm(45.82oz.in)	4.1kg.cm(56.93oz.in)
OPERATING ANGLE	: 45°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
CURRENT DRAIN	: 8mA/IDLE AND 150mA/NO LOAD RUNNING	
DEAD BAND WIDTH	: 8usec	
CONNECTOR WIRE LENGTH	: 300mm(11.81in)	
DIMENSIONS	: 40.6x19.8x36.6mm(1.59x0.77x1.44in)	
WEIGHT	: 45.5g(1.6oz)	



2.FEATURES
- 3-POLE FERRITE MOTOR
- LONG LIFE POTENTIOMETER
- DUAL BALL BEARING
- INDIRECT POTENTIOMETER DRIVE

3.APPLICATIONS
- AIRCRAFT 20-60 SIZE
- 30 SIZE HELICOPTERS
- STEERING AND THROTTLE SERVO FOR CARS
- TRUCK AND BOATS

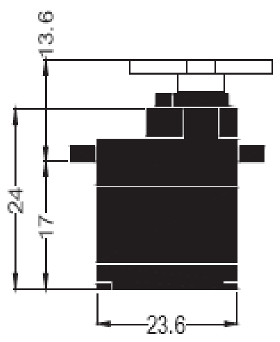
Motor Servo Hitech HS-65HB

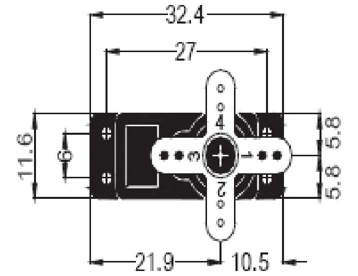
PREPARED BY JUN HEE, LEE
UPDATE: AUG 01, 2003

GENERAL SPECIFICATION OF HS-65HB MIGHTY FEATHER SERVO

1.TECHNICAL VALUE

CONTROL SYSTEM	:+PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	:−20°C TO +60°C(−68°F TO +140°F)	
TEST VOLTAGE	:AT 4.8V	AT 6.0V
OPERATING SPEED	:0.16sec/60° AT NO LOAD	0.13sec/60° AT NO LOAD
STALL TORQUE	:1.6kg.cm(22.21oz.in)	1.9kg.cm(26.38oz.in)
STANDING TORQUE	:1.3kg.cm(18.05oz.in)/5° HOLD OUT	1.5kg.cm(20.83oz.in)/5° HOLD OUT
IDLE CURRENT	:7.4mA AT STOPPED	7.7mA AT STOPPED
RUNNING CURRENT	:180mA/60° AT NO LOAD	220mA/60° AT NO LOAD
STALL CURRENT	:960mA	1200mA
DEAD BAND WIDTH	:5usec	5usec
OPERATING TRAVEL	:40°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	:CORED METAL BRUSH/Nd MAGENT	
POTENTIOMETER TYPE	:2 SLIDER/DIRECT DRIVE	
AMPLIFIER TYPE	:ANALOG S.M.T	
DIMENSIONS	:23.6x11.6x24mm(0.92x0.45x0.94in)	
WEIGHT	:11.2g(0.39oz)	
BALL BEARING	:SINGLE/MR85	
GEAR MATERIAL	:HEAVY DUTY RESIN	
HORN GEAR SPLINE	:25 SEGMENTS/ø5	
SPLINED HORNS	:MICRO23:M23-I, M23-X	
CONNECTOR WIRE LENGTH	:250mm(9.84in)	
CONNECTOR WIRE STRAND COUNTER	:20 EA	
CONNECTOR WIRE GAUGE	:28AWG	





2.FEATURES
HEAVY DUTY RESIN GEARS WITH BALL BEARING
HIGH EFFICIENCY Nd MAGNET MOTOR

3.APPLICATIONS
FAST PARK FLYING MODELS. ELECTRIC MICRO PLANE, MAV
1/18 SCALE CARS

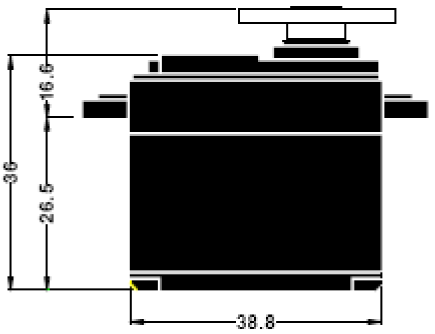
Motor Servo Hitech HS-475HB

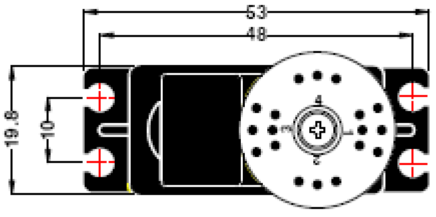
PREPARED BY JUN HEE, LEE
UPDATE:APR 01, 2002

ANNOUNCED SPECIFICATION OF HS-475HB STANDARD DELUXE SERVO

1.TECHNICAL VALUE

CONTROL SYSTEM	:+PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
TEST VOLTAGE	:AT 4.8V	AT 6.0V
OPERATING SPEED	:0.23sec/60? AT NO LOAD	0.18sec/60? AT NO LOAD
STALL TORQUE	:4.4kg.Cm(61.10oz.in)	5.5kg.Cm(76.37oz.in)
IDLE CURRENT	:7.4mA AT STOPPED	7.7mA AT NO LOAD
RUNNING CURRENT	:160mA/60? AT NO LOAD	180mA/60? AT NO LOAD
STALL CURRENT	:900mA	1100mA
DEAD BAND WIDTH	:5usec	5usec
OPERATING TRAVEL	:40? /ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	:CORED METAL BRUSH	
POTENTIOMETER TYPE	:6 SLIDER/INDIRECT DRIVE	
AMPLIFIER TYPE	:ANALOG CONTROLLER & TRANSISTOR DRIVER	
DIMENSIONS	:38.8x19.8x36mm(1.52x0.77x1.41in)	
WEIGHT	:40g(1.41oz)	
BALL BEARING	:TOP/MR106	
GEAR MATERIAL	:HEAVY DUTY RESIN	
HORN GEAR SPLINE	:24 SEGMENTS/?5.76	
SPLINED HORNS	:REGULAR/R-C,R-D,R-I,R-O,R-X,SUPER/R-XA	
CONNECTOR WIRE LENGTH	:300mm(11.81in)	
CONNECTOR WIRE STRAND COUNTER	:60EA	
CONNECTOR WIRE GAUGE	:22AWG	





2.FEATURES

HEAVY DUTY RESIN GEARS, TOP BALL BEARING

3.APPLICATIONS

AIRCRAFT 20-60 SIZE, 30 SIZE HELICOPTER, STEERING AND THROTTLE SERVO, TRUCK AND BOATS

4.ACCESSORY & OPTION

CASE SET/	GEAR SET/	BALL BEARING SET/	HORN SET/
HS475T:1EA	HS475G1:1EA	MR106:1EA	R-C:1EA
HS475M:1EA	HS475G2:1EA	CU 7.99x6:1EA	R-D:1EA
HS475L:1EA	HS475G3:1EA	FE 10x8:1EA	R-I:1EA
PH/T-2 2x30 NI:4EA	HS475G4:1EA		R-O:1EA
	HS75PG:1EA		R-X:1EA
			R-XA:1EA
			WH/W 2.1x15 NI:4EA
			BST 3x5.5:4EA
			NBR 9x6.5x6:4EA

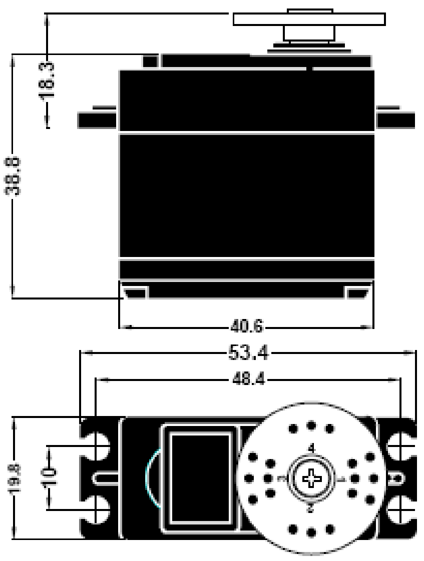
Motor Servo Hitech HS-635HB

PREPARED BY JUN HEE, LEE
UPDATE: JULY 21, 2002

GENERAL SPECIFICATION OF HS-635HB HIGH TORQUE SERVO

1.TECHNICAL VALUE

CONTROL SYSTEM	:+PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	:-20?C TO +60?C(-46?F TO +86?F)	
TEST VOLTAGE	:AT 4.8V	AT 6.0V
OPERATING SPEED	:0.18sec/60? AT NO LOAD	0.15sec/60? AT NO LOAD
STALL TORQUE	:5kg.cm(69.43oz.in)	6kg.cm(83.32oz.in)
STANDING TORQUE	:4kg.cm(55.54oz.in)/5? HOLD OUT	5kg.cm(69.43oz.in)/5? HOLD OUT
IDLE CURRENT	:7.4mA AT STOPPED	7.7mA AT STOPPED
RUNNING CURRENT	:400mA/60? AT NO LOAD	500mA/60? AT NO LOAD
STALL CURRENT	:2000mA	2500mA
DEAD BAND WIDTH	:5usec	5usec
OPERATING TRAVEL	:40?/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	:CORED METAL BRUSH	
POTENTIOMETER TYPE	:6 SLIDER/INDIRECT DRIVE	
AMPLIFIER TYPE	:ANALOG CONTROLLER & MOSFET DRIVER	
DIMENSIONS	:40.6x19.8x38.8mm(1.59x0.77x1.52in)	
WEIGHT	:50g(1.76oz)	
BALL BEARING	:DUAL/MR106	
GEAR MATERIAL	:HEAVY DUTY RESIN	
HORN GEAR SPLINE	:24 SEGMENTS/?5.76	
SPLINED HORNS	:REGULAR/R-C,R-D,R-I,R-O,R-X, SUPER/R-XA	
CONNECTOR WIRE LENGTH	:300mm(11.81in)	
CONNECTOR WIRE STRAND COUNTER	:60EA	
CONNECTOR WIRE GAUGE	:22AWG	



2.FEATURES
HEAVY DUTY RESIN GEAR, DUAL BALL BEARING

3.APPLICATIONS
AIRCRAFT 20-60 SIZE, 60 SIZE HELICOPTERS, STEERING AND THROTTLE SERVO, TRUCK AND BOATS

4.ACCESSORY & OPTION

CASE SET/	GEAR SET/	BALL BEARING SET/	HORN SET/
HS635T:1EA	HS635G1:1EA	MR106:2EA	R-C:1EA
HS635M:1EA	HS635G2:1EA		R-D:1EA
HS635L:1EA	HS635G3:1EA		R-I:1EA
PH/T-2 2x33 NI:4EA	HS635G4:1EA		R-X:1EA
	HS75PG:1EA		R-XA:1EA
			WH/W 2.1x15 NI:4EA
			BST 3x5.5:4EA
			NBR 9x6.5x6:4EA

IC NM27C512

National Semiconductor

February 1994

NM27C512
524,288-Bit (64K x 8) High Performance CMOS EPROM

General Description

The NM27C512 is a high performance 512K UV Erasable Electrically Programmable Read Only Memory (EPROM). It is manufactured using National's proprietary 0.8 micron CMOS AMG™ EPROM technology for an excellent combination of speed and economy while providing excellent reliability.

The NM27C512 provides microprocessor-based systems storage capacity for portions of operating system and application software. Its 90 ns access time provides no-wait-state operation with high-performance CPUs. The NM27C512 offers a single chip solution for the code storage requirements of 100% firmware-based equipment. Frequently-used software routines are quickly executed from EPROM storage, greatly enhancing system utility.

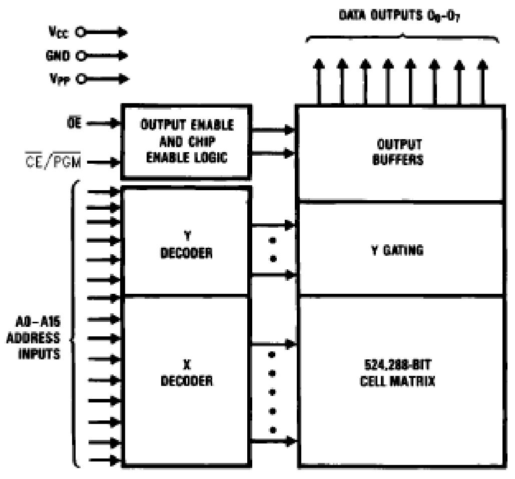
The NM27C512 is configured in the standard JEDEC EPROM pinout which provides an easy upgrade path for systems which are currently using standard EPROMs.

The NM27C512 is one member of a high density EPROM Family which range in densities up to 4 Megabit.

Features

- High performance CMOS
  - 90 ns access time
- Fast turn-off for microprocessor compatibility
- Manufacturers identification code
- JEDEC standard pin configuration
  - 28-pin DIP package
  - 32-pin chip carrier

Block Diagram

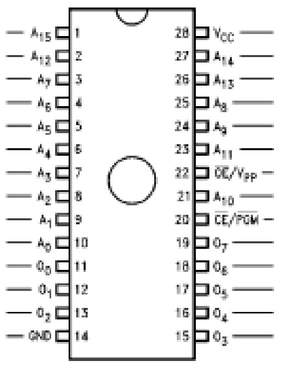


TL/D/10834–1

Connection Diagrams

27C080	27C040	27C020	27C010	27C256
A_{19}	XX/V_{PP}	XX/V_{PP}	XX/V_{PP}	
A_{16}	A_{16}	A_{16}	A_{16}	
A_{15}	A_{15}	A_{15}	A_{15}	V_{PP}
A_{12}	A_{12}	A_{12}	A_{12}	A_{12}
A_7	A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0	A_0
O_0	O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2	O_2
GND	GND	GND	GND	GND

DIP
NM27C512



27C256	27C010	27C020	27C040	27C080
	V_{CC}	V_{CC}	V_{CC}	V_{CC}
	$XX/\overline{PGM}$	$XX/\overline{PGM}$	A_{18}	A_{18}
V_{CC}	XX	A_{17}	A_{17}	A_{17}
A_{14}	A_{14}	A_{14}	A_{14}	A_{14}
A_{13}	A_{13}	A_{13}	A_{13}	A_{13}
A_8	A_8	A_8	A_8	A_8
A_9	A_9	A_9	A_9	A_9
A_{11}	A_{11}	A_{11}	A_{11}	A_{11}
$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$
A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
$\overline{CE}/\overline{PGM}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}/\overline{PGM}$	$\overline{CE}/\overline{PGM}$
O_7	O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3	O_3

TL/D/10834–2

Note: Compatible EPROM pin configurations are shown in the blocks adjacent to the NM27C512 pins.

Commercial Temp Range (0°C to +70°C)

Parameter/Order Number	Access Time (ns)*
NM27C512 Q, N, V 90	90
NM27C512 Q, N, V 120	120
NM27C512 Q, N, V 150	150
NM27C512 Q, N, V 200	200

Extended Temp Range (−40°C to +85°C)

Parameter/Order Number	Access Time (ns)*
NM27C512 QE, NE, VE 90	90
NM27C512 QE, NE, VE 120	120
NM27C512 QE, NE, VE 150	150
NM27C512 QE, NE, VE 200	200

Military Temp Range (−55°C to +125°C)

Parameter/Order Number	Access Time (ns)*
NM27C512 QM 200	200

Note: Surface mount PLCC package available for commercial and extended temperature ranges only.

*All versions are guaranteed to function for slower speeds.

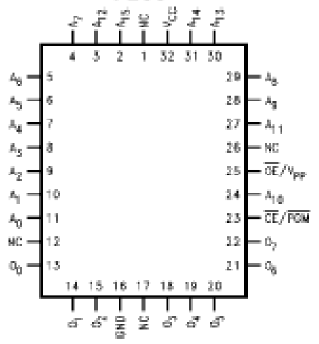
Package Types: NM27C512 Q, N, V XXX
Q = Quartz-Windowed Ceramic DIP Package
N = Plastic OTP DIP Package
V = PLCC Package

- All packages conform to the JEDEC standard.

Pin Names

A0–A15	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
O0–O7	Outputs
$\overline{PGM}$	Program
XX	Don't Care (During Read)

PLCC



TL/D/10834–3

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Storage Temperature	$-65^{\circ}C$ to $+150^{\circ}C$
All Input Voltages Except A9 with Respect to Ground	$-0.6V$ to $+7V$
V_{PP} and A9 with Respect to Ground	$-0.7V$ to $+14V$
V_{CC} Supply Voltage with Respect to Ground	$-0.6V$ to $+7V$
ESD Protection (MIL Std. 883, Method 3015.2)	$>2000V$
All Output Voltages with Respect to Ground	V_{CC} + 1.0V to GND $-0.6V$

Operating Range

Range	Temperature	V_{CC}	Tolerance
Comm'l	$0^{\circ}C$ to $+70^{\circ}C$	+5V	±10%
Industrial	$-40^{\circ}C$ to $+85^{\circ}C$	+5V	±10%
Military	$-55^{\circ}C$ to $+125^{\circ}C$	+5V	±10%

Read Operation

DC Electrical Characteristics

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{IL}	Input Low Level		−0.5	08	V
V_{IH}	Input High Level		2.0	V_{CC} + 1	V
V_{OL}	Output Low Voltage	I_{OL} = 2.1 mA		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −2.5 mA	3.5		V
I_{SB1}	V_{CC} Standby Current (CMOS)	$\overline{CE}$ = V_{CC} ±0.3V		100	µA
I_{SB2}	V_{CC} Standby Current	$\overline{CE}$ = V_{IH}		1	mA
I_{CC1}	V_{CC} Active Current	$\overline{CE}$ = $\overline{OE}$ = V_{IL} f = 5 MHz		40	mA
I_{CC2}	V_{CC} Active Current CMOS Inputs	$\overline{CE}$ = GND, f = 5 MHz Inputs = V_{CC} or GND, I/O = 0 mA C, I Temp Ranges		35	mA
I_{PP}	V_{PP} Supply Current	V_{PP} = V_{CC}		10	µA
V_{PP}	V_{PP} Read Voltage		V_C − 0.7	V_{CC}	V
I_{LI}	Input Load Current	V_{IN} = 5.5V or GND	−1	1	µA
I_{LO}	Output Leakage Current	V_{OUT} = 5.5V or GND	−10	10	µA

AC Electrical Characteristics

Symbol	Parameter	90		120		150		200		Units
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{ACC}	Address to Output Delay		90		120		150		200	
t_{CE}	$\overline{CE}$ to Output Delay		90		120		150		200	
t_{OE}	$\overline{OE}$ to Output Delay		40		50		50		50	
t_{DF}	Output Disable to Output Float		35		25		45		55	ns
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$, Whichever Occurred First	0		0		0				

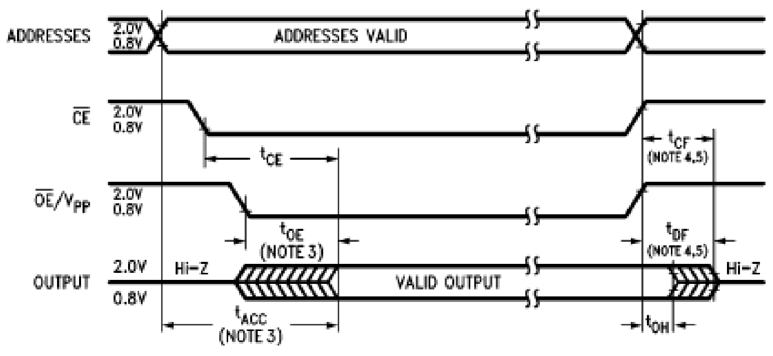
Capacitance $T_A = +25°C$, f = 1 MHz (Note 2)

Symbol	Parameter	Conditions	Typ	Max	Units
C_{IN1}	Input Capacitance except $\overline{OE}/V_{PP}$	$V_{IN} = 0V$	6	12	pF
C_{OUT}	Output Capacitance	$V_{OUT} = 0V$	9	12	pF
C_{IN2}	$\overline{OE}/V_{PP}$ Input Capacitance	$V_{IN} = 0V$	20	25	pF

AC Test Conditions

Output Load	1 TTL Gate and $C_L = 100$ pF (Note 8)
Input Rise and Fall Times	≤ 5 ns
Input Pulse Levels	0.45V to 2.4V
Timing Measurement Reference Level (Note 9)	
Inputs	0.8V and 2V
Outputs	0.8V and 2V

AC Waveforms (Notes 6, 7)



TL/D/10834–4

Note 1: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note 2: This parameter is only sampled and is not 100% tested.

Note 3: $\overline{OE}$ may be delayed up to $t_{ACC} - t_{CE}$ after the falling edge of $\overline{CE}$ without impacting t_{ACC}.

Note 4: The t_{DF} and t_{CF} compare level is determined as follows:
High to TRI-STATE, the measured V_{OH1} (DC) $-$ 0.10V;
Low to TRI-STATE, the measured V_{OL1} (DC) $+$ 0.10V.

Note 5: TRI-STATE may be attained using $\overline{OE}$ or $\overline{CE}$.

Note 6: The power switching characteristics of EPROMs require careful device decoupling. It is recommended that at least a 0.1 µF ceramic capacitor be used on every device between V_{CC} and GND.

Note 7: The outputs must be restricted to V_{CC} + 1.0V to avoid latch-up and device damage.

Note 8: 1 TTL Gate: $I_{OL} = 1.6$ mA, $I_{OH} = -400$ µA.
C_L: 100 pF includes fixture capacitance.

Note 9: Inputs and outputs can undershoot to $-2.0V$ for 20 ns Max.