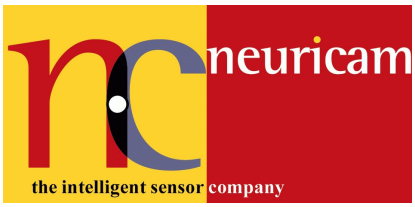


Via S. M. Maddalena, 12 - 38100 Trento - Italy
Tel. +39 0461 260552 - Fax +39 0461 260617
email: info@neuricam.com - http://www.neuricam.com

PCCam

PC-based Smart Camera

USER MANUAL

Rel. 04/05

TABLE OF CONTENTS

1 INTRODUCTION

PCCam is a complete PC-based vision system that combines the functions of image acquisition and image processing in a compact form factor. Processing results or images can be transmitted remotely through a 10/100-Mbit Ethernet connection. A host USB 1.1 interface and a serial RS232 provide standard ready-to-use communication channels. Furthermore, four opto isolated I/O lines (two inputs and two outputs) allow synchronized image acquisition and control of industrial devices. The Operative System mounted on PCCam is Linux.

PCCam is build around the SC2200 Geode CPU, a high performance and low-power consumption processor from National Semiconductor. The logic core is an x86 architecture running at 266MHz, with a 16-Kbytes cache and MMX and FPU units. A keyboard and a mouse PS/2 ports facilitate programming of the smart camera. For easy calibration and control, images can be directly visualized on a RGB monitor. A stereo audio I/O is also available. The memory and capabilities of the PCCam system can be expanded using a mini PCI connector and a CompactFlash socket.

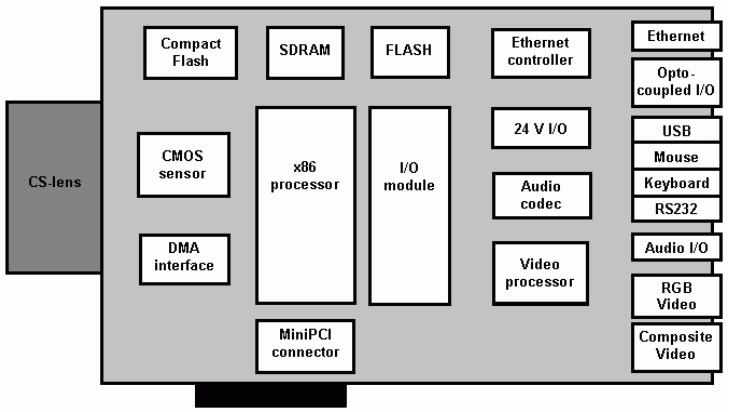


Figure 1.1: Block diagram of PCCam

The sensor mounted on the camera is a CMOS array of size 640x480 pixels from National Semiconductor, colour LM9628 or monochrome LM9618. This sensor is especially suited for outdoor or non-controlled light applications because of its high dynamical range (110 dB). A FPGA provides the interface between sensor and processor. The FPGA is programmed to acquire entire images or blocks of pixels, Region Of Interest (ROI). The maximum frame rate for full size images is 30 f/s. The frame rate can be increased using smaller ROI.

The 266MHz processor, coupled with the MMX istruction set and the Integrated Floating Point Unit (FPU), allows performing computationally intensive image processing algorithms. Several basic image-processing functions are included in the PCCam Image Processing Library. These functions, written in the x86 assembly language, are optimised to exploit the MMX capabilities of the processor.
The camera communicates through: 1) a 10/100Mbit Ethernet, 2) a serial RS232 interface, 3) a USB1.1 port and 4) four opto isolated I/O lines. The Ethernet connection is used to transmit images or processing results. Because of the 100 Mbits bandwidth, the maximum frame rate achievable without compression is 13 color frames per second (13x3x640x480x8 bits ≈ 100 Mbits). The RS232 and the USB1.1 interfaces offer alternative standard channels to communicate with PCCam. The four optoisolated I/O port interface PCCam with industrial devices: the two input lines allow synchronizing image acquisition with an external trigger, whereas the two output lines can be used to drive external devices.

2 TECHNICAL AND MECHANICAL CHARACTERISTICS

- SCx200 Geode CPU: x86 architecture, 266 MHz, 16 KB cache, FPU, MMX;
- CMOS sensor of size 640x480 pixels from National Semiconductors
- Suitable for Different Image sensor types Through a FPGA Interface;
- Colour interpolated or sensor raw data storage capabilities;
- Real time image acquisition, direct in SDRAM through Video Input Port;
- Frame Grabber and/or Live Camera modality;
- SRAM on SODIMM socked for easy upgrades;
- Removable Compact Flash for Image storage;
- Non Volatile I2C Flash for parameter storage;
- OS on easy removable Disk On Module (44 Pin);
- MiniPCI TypeIII Socked for add on modules (Wireless/Bluetooth);
- Graphics Controller with interface for NTSC/PAL TV and RGB colour monitors on DB15;
- Analog Stereo Audio I/O on;
- Host miniUSB standard connection;
- PS/2 keyboard and mouse;
- 2x RS232 + 2x Input/2x Output Trigger 24V command;
- 10/100BaseT Ethernet port on RJ45 Connector;
- Single unregulated 12V Power Supply (consumption below 15W);
- Compact Case.

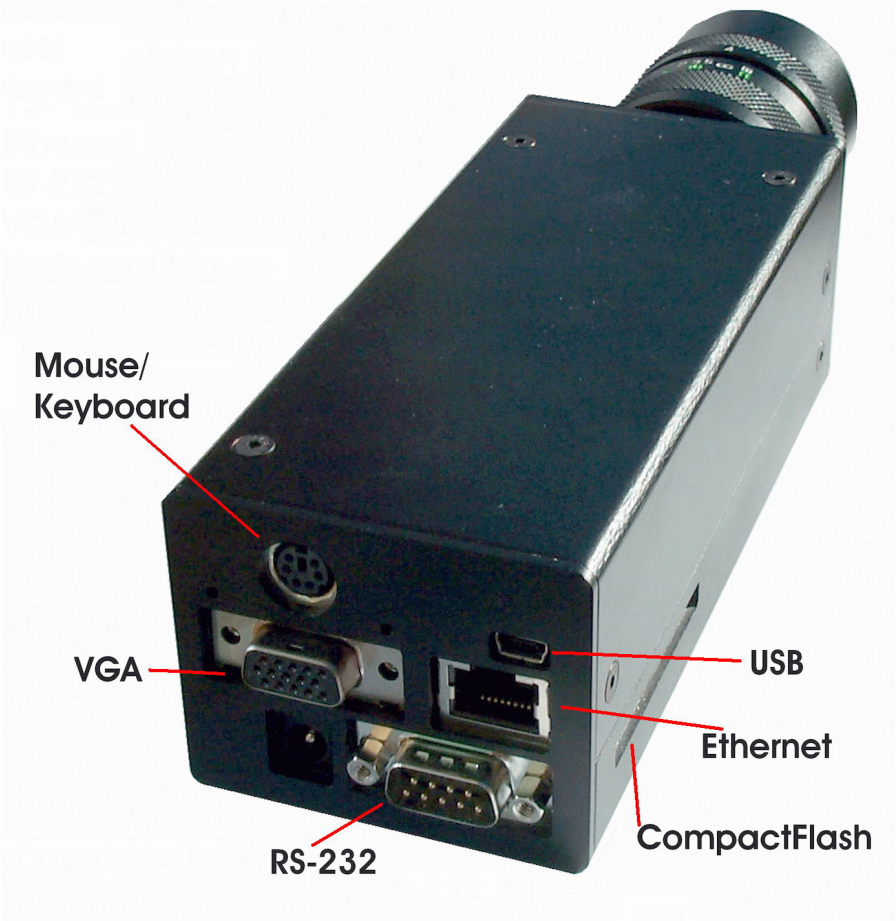


Figure 2.1: Rear View of PCCam.

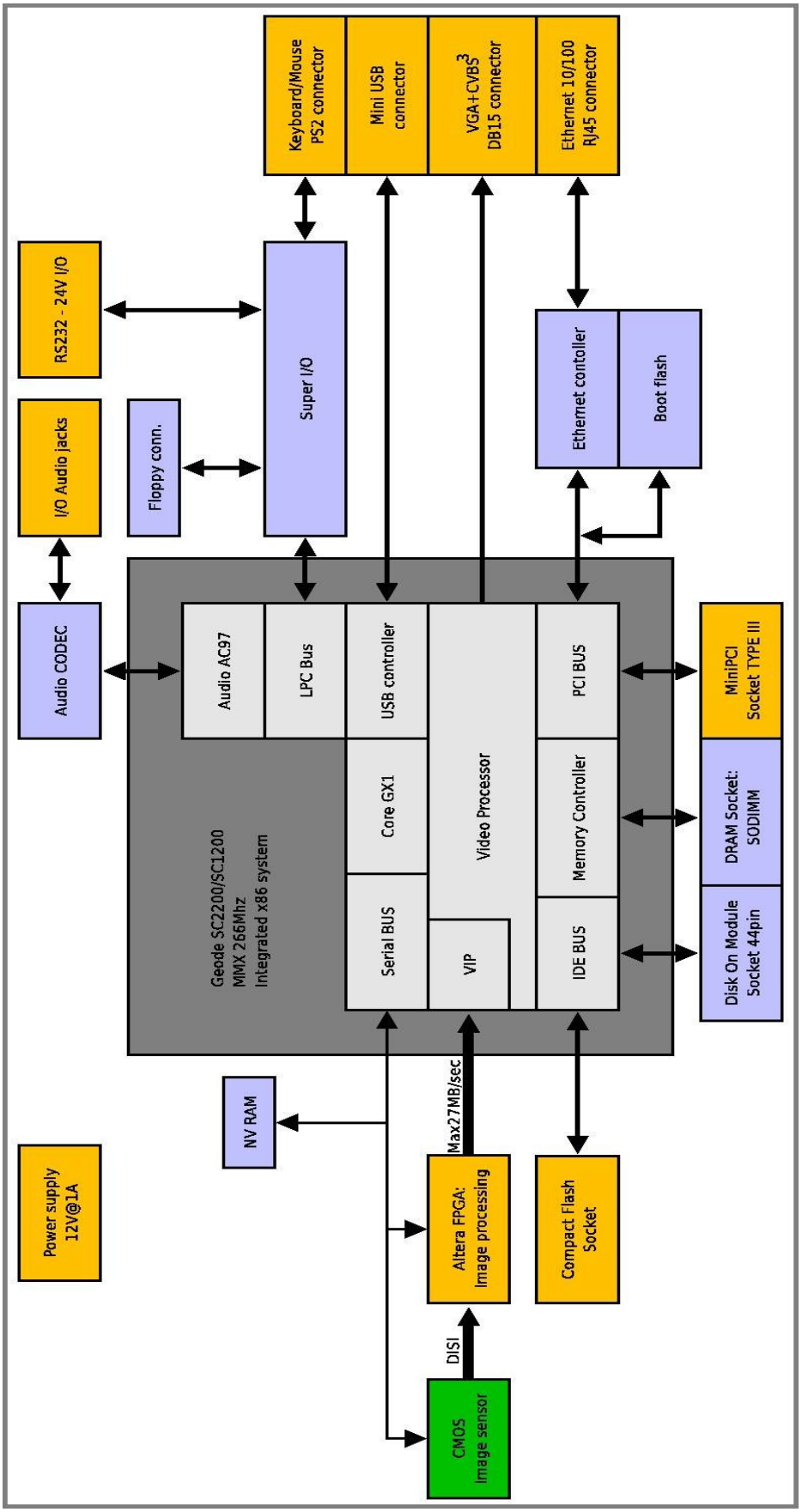


Figure 2.2: Block Diagram of PCCam

2.1 CMOS IMAGE SENSOR

Feature	Specifications
Type	National sensor (colour LM9628 or monochrome LM9618)
Micro-lenses	Yes
Pixels	640x480
Array Diameter	1/3”
Full Res. Frame Rate	30 frame/sec
Sensitivity	12-bit A/D converter
Dynamic Range (dB)	62-110

2.2 CPU & CORE LOGIC

Feature	Specifications
CPU Type	NS Geode SC1200 or SC2200
CPU Features	266 MHz, FPU, Pentium MMX instruction set
Internal Bus	64-bit, 100 MHz
Cache	16 KB unified L1 cache, WB
PC core logic	DMA controller, Interrupt controller, Timers
RTC	Real Time Clock, powered by external lithium battery
Watchdog	Interfaces to INTR, SMI, Reset

2.3 PERIPHERALS

Feature	Specifications
VGA Output	Resolution up to 1024 x 768 at 85 Hz, up to 16 bpp Frame buffer in system memory NTSC-M & PAL-M/B/D/G/H/I/N and RGB interface options Frame buffer driver for STN panels with built-in controller
USB	Standard USB port, 1.5 / 12 Mbps, OHCI v-1.0 compliant
Serial Ports	COM1 - RS232 together with 24V opto-coupled I/O COM2 – UART2 (SIN, SOUT, DTR) on VGA connector
Keyboard Interface	StandardPS/2 together with Mouse Interface
Mouse Interface	PS/2 together with Keyboard Interface
Ethernet	Realtek 8139, MAC & PHY, 10/100BaseT
Audio	2x Jack Connectors: STEREO IN/OUT
24V I/O (opto-coupled)	Together with COM1 on DB9 connector

2.4 MEMORY AND BUSSES

Feature	Specifications
DRAM	SDRAM 100 MHz, 64-bit, SODIMM socket
BOOT Flash	Socked for PLCC32
FLASH DISK	Standard Disk On Module 44Pin
Compact Flash	Standard Connector Type II
PCI bus	Standard Connector Mini PCI Type II

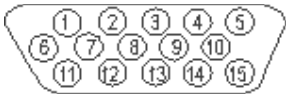
2.5 ELECTRICAL, MECHANICAL AND ENVIRONMENTAL SPECIFICATIONS

Supply Voltage	Single 12V@1A
Power consumption	6-10 W, depending on configuration
Dimensions	188 x 65 x 53 mm

2.6 CONNECTORS PINOUT

- **VGA+TV/RS232 on DB15**

Pin Number	Description
1	RED_OUT
2	GREEN_OUT
3	BLUE_OUT
4	NC / DTR (TV / COM3)
5	GND
6	GND
7	GND
8	GND
9	CVBS_OUT / SIN (TV / COM3)
10	GND
11	GND_CVBS / SOUT (TV / COM3)
12	DDC DAT
13	Horizontal Synchronization
14	Vertical Synchronization
15	DDC CLOCK



- **Keyboard + Mouse MINIDIN**

PS2 Mouse and Keyboard must be connected through the included Y cable adapter.

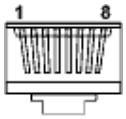
Pin Number	Description
1	MOUSE DATA
2	KEYB DATA
3	VDD (5V)
4	GND
5	MOUSE CLOCK
6	KEYB CLOCK

- **Host Mini USB**

Pin Number	Description
1	VDD (5V)
2	+ DATA
3	- DATA
4	NC
5	GND

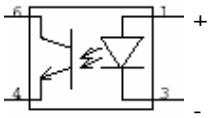
- **Ethernet RJ45**

Pin Number	Description
1	TX_D1+
2	TX_D1-
3	RX_D2+
4	75 ohm termination
5	75 ohm termination
6	RX_D2-
7	75 ohm termination
8	75 ohm termination

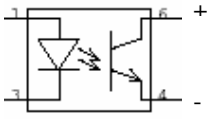


- **RS232 /24V (I/O)**

Pin Number	Description
1	DCD / IN24V_1+
2	SIN / IN24V_1-
3	SOUT / IN24V_2+
4	DTR / IN24V_2-
5	GND
6	DSR / OUT24V_1+
7	RTS / OUT24V_1-
8	CTS / OUT24V_2+
9	RI / OUT24V_2-



24V Input

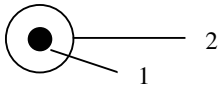


24V Output

RS232 or 24V I/O modality is selected through HW jumper resistors.

- **Power JACK**

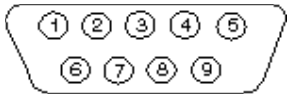
Pin Number	Description
1	VDD (12V) unregulated
2	GND



3 OPTOCOUPLED INTERFACE

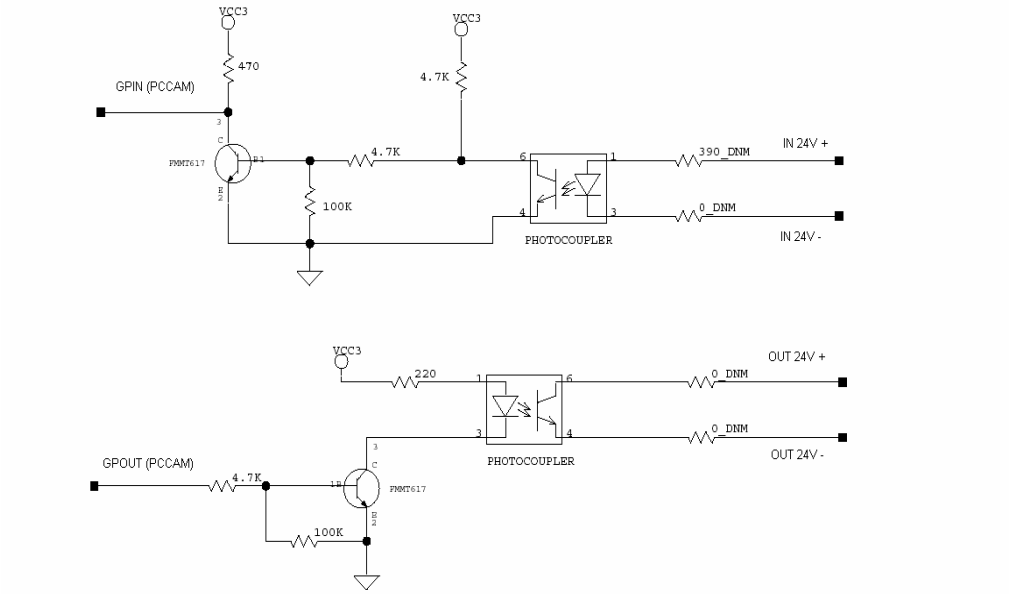
The PCCAM serial connector can be set to provide an RS232 communication protocol or a generic optocoupled I/O. This setting requires a hardware modification and should be performed before purchasing the camera.
In the optocoupled mode the following table describes the role of each pin of the serial connector.

Serial Connector 24V (I/O)



Pin Number	Description
1	IN24V_1+
2	IN24V_1-
3	IN24V_2+
4	IN24V_2-
5	GND
6	OUT24V_1+
7	OUT24V_1-
8	OUT24V_2+
9	OUT24V_2-

The following scheme shows the electrical schematics of the 2 optocouplers inside PCCAM. The signals on the right side are the ones from the connector and the ones on the left are the internal signal for the Altera FPGA. The first scheme is related to a GPIN and the second to a GPOUT.



The general pourpose inputs are designed to be interfaced with standard TTL ports (5V) . Its Power consumption is about 10mA.

The general pourpose outputs are standard Open-Collector that support a maximum of 100mA of output current.

4 THE SENSOR INTERFACE

The interface between the Image Sensor and the SCx200 VIP (Video Input Port) is performed by a Altera FPGA. A master clock is provided to the FPGA and the Sensor. The Sensor divides the clock to 12 or 8 MHz, and this frequency is used to acquire images.
The Sensor sends to Altera the image, the pixel clock (clock used for the acquisition) and a copy of the signals (h_sync_i and v_sync_i) which contain information on the beginning and the end of rows and frames. Altera processes these data and sends to VIP a clock at 12 or 24 MHz and the image data. The FPGA's internal structure is schematized in Figure 2.

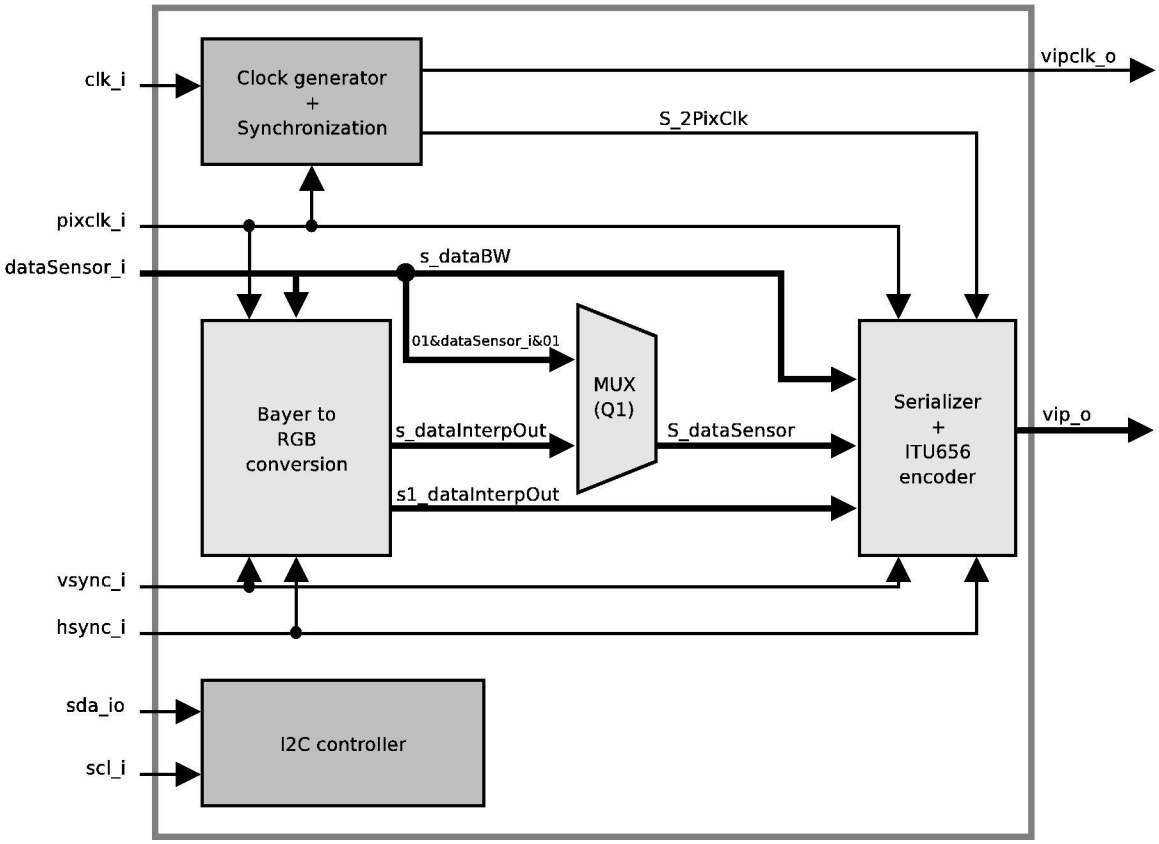


Figure 4.1: Sensor interface

4.1 ACQUISITION MODES

There are several image acquisition modes available, some for the b/w sensor, and some for the colour sensor.
Both sensors support the16-bit 656 RGB mode (useful for video overlay). The black and white sensor supports also the 8 bit fast, the 12 bit, and the 16 bit acquisition mode.
In the 8 bit fast mode, the 12-bit pixel values are directly taken from the sensor (with frequency 12 MHz) and then Altera removes the 4 least significant bits in order to get a 8 bit value. Because of the ITU656 standard (see the section 4.3), the pixel value can range from 1 to 254.
In the 12 bit mode, data are directly taken from the sensor (with frequency 12 MHz) and then Altera adds two bits at the beginning and at the end of the pixel value in order to comply with the ITU656 standard: 01XXXXXXXXXXXX01.

In the 16-bit mode Altera preserves the 5 most significant bits to obtain a 16-bit signal, which is the concatenation of a 5-bit data (data red), a 6-bit data (data green), and a 5-bit data (data blue): XXXXXXXXXX0XXXXX.

The color sensor supports two color modes: the 16-bit and the 24-bit RGB. A conversion from the Bayer pattern to the RGB format is used.

4.2 BAYER TO RGB CONVERSION

Several colour sensors, such as the National LM9628, provide colour images that are encoded with a Bayer pattern. To obtain the colour image in RGB format, one must first decode the image using the Bayer algorithm.

The Altera FPGA takes the data coming from the sensor in raw Bayer sequence and converts them in a normal RGB array .
The color sensor is a colored checkerwork composed of red, blue and green cells (the Bayer pattern, see **Figure 4.2**). A number is associated to each colour. The classification is performed by the signal ***s_row_column_toggle*** which can be equal to the following configurations : “00”, ” 01”, ”10” and “11”.
At each clock cycle, the LSB of the precedent signal is inverted. At each change of row, the MSB of the signal is inverted and the LSB is forced to1.

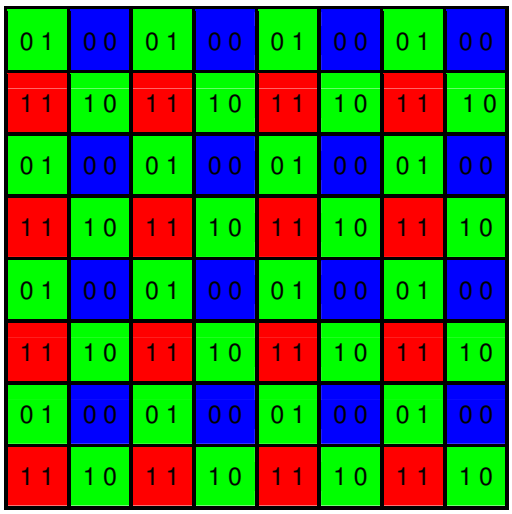
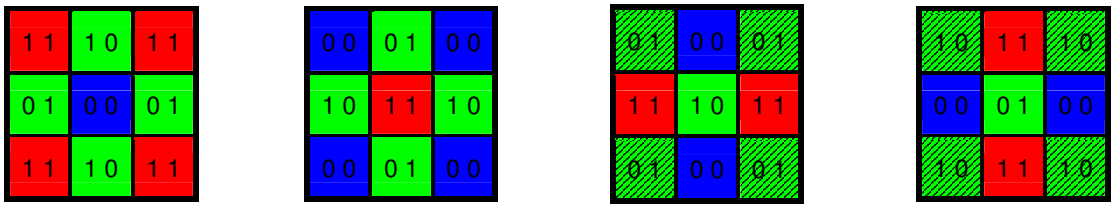


Figure 4.2: Bayer pattern

Each pixel contains only one colour, that is why an interpolation is necessary. Given a pixel, consider the 3x3 matrix, in which the pixel is the central one. Four or eight neighbour pixels (cells without stripes) are necessary for the interpolation, as shown in **Figure 4.3**:



Blue “00” Red “11” Green “10” Green “01”

Figure 4.3

Since eight pixels are memorized for the interpolation, two fifos and six flip-flop are used. Taking three consecutive rows, while the data of the third row enter the two fifo, the data of the first row go out from fifo1 and the data of the second row go out from fifo2. Therefore, the reading of fifo1 is the reading of fifo2 one row delayed. The two first rows are read only when the writing of the third starts.

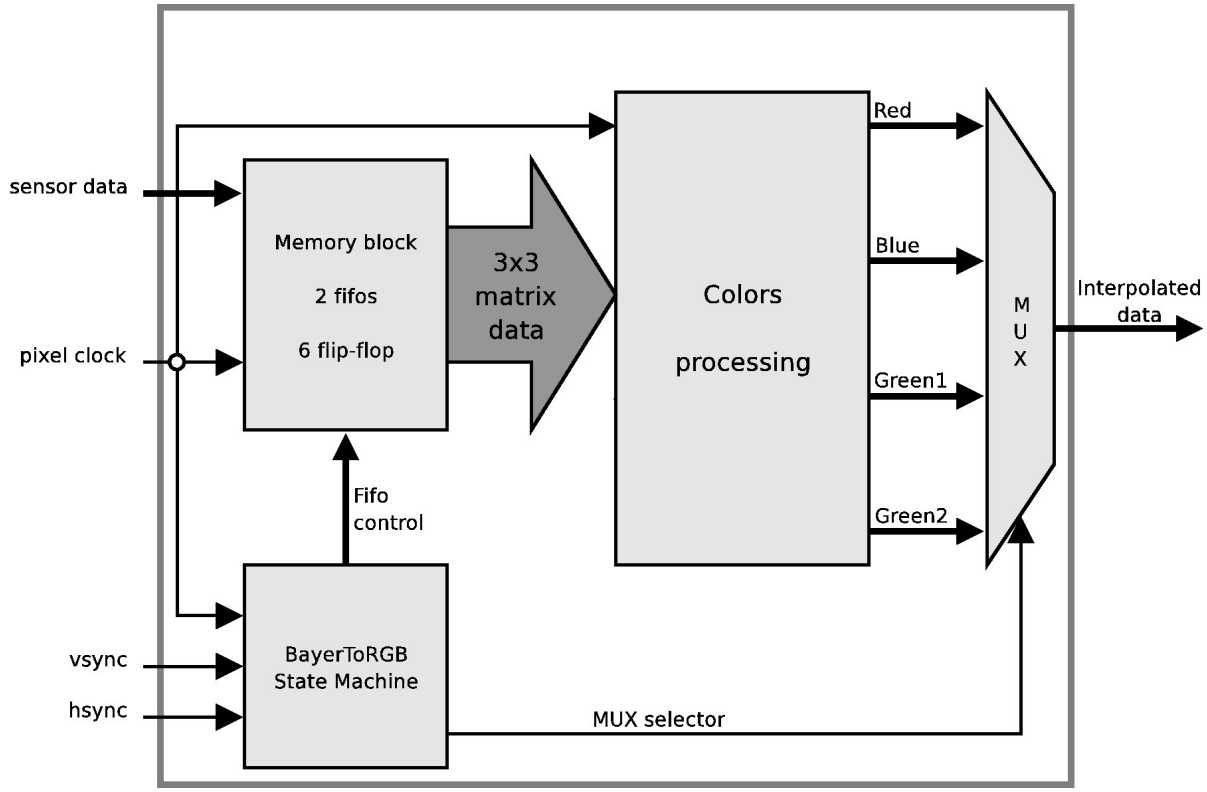


Figure 4.4: Bayer pattern conversion

4.3 OUTPUT ON VIDEO INPUT PORT

ITU656 is a standard that allows the transfer of digital video through an 8 bit bus (***vip_o***) and a clock (***vipclk_o***).
The standard uses the two key words, ***00h*** and ***FFh***, to insert two video timing reference coding: SAV (start of active video) and EAV (end of active video). These bit configurations are not allowed during the active video line, this means that some additional check on the coming data is required.

Acquisition mode	Original data	ITU656 compliant data
8 bit fast	00000000	00000001
	11111111	11111110
16 bit	00000000 00000000	00001000 00000001
	11111111 11111111	11110111 11111110
24 bit	00000000 00000000 00000000	00000001 00000001 00000001
	11111111 11111111 11111111	11111110 11111110 11111110
12 bit	XXXXXXXXXXXX	01XXXXXX XXXXXX01

When an invalid bit configuration is detected, the original data is replaced by a ITU656 compliant data (see the table above).
For the 12 bit mode, two bits are added at the beginning and at the end. The pixel value is sent to the video input port as: “01XXXXXX””XXXXXX01” avoiding the 00h and FFh values.

5 THE PCCAM SOFTWARE DEVELOPMENT ENVIRONMENT

The PCCam development environment consists of a smart camera with monitor (CRT or TFT), keyboard and mouse.
Users can develop their application software on PCCam using standard software tools (editor, compiler). Programs can be written in C/C++ or x86-assembly languages. Alternatively, programs can be developed and compiled on a host workstation and then they are transferred to PCCam. Once a program is stored in the non-volatile memory (flash memory), PCCam system becomes a completely autonomous unit that can acquire and process images and communicates the processing results via Ethernet, USB, or RS232.

The PCCam system supports the Linux Operative System. The precompiled kernel supports all the the devices mounted on the camera and several network services and protocols like http, ftp, telnet, nfs, etc. The user can recompile the kernel adding and/or removing features.

A CD is supplied with the camera. Its directory structure is

- Linux contains the 2.4.22 kernel source code, configuration files and drivers;
- Recovery contains a OS image useful to restore the system;
- Demo contains programs that allows the user to display images captured from PCCam;
- Docs contains manuals, datasheets and related documents.

5.1 OPERATING SYSTEM SPECIFICATIONS

Operating system	Slackware Linux 10.0
Kernel	2.4.22 patched (see chapter 5.9)
File System	ReiserFS
gcc version	3.3.4

5.2 LOGGING ON PCCAM

The PCCam system works exactly as a standard personal computer: once the user has connected keyboard, mouse, monitor and Ethernet cable, PCCam is ready to be turned on.
Firstly the system checks its devices (memory, disks, etc.); during this stage the user can access the bios settings pressing the key “F2”.
After this step the Linux operating system starts its booting procedure. At the end the user is asked for username and password. Possible choices:

Username: root
Password: pccam2003

Username: pccam
Password: pccam2003.

“root” is a special user that can perform almost every operation on the system. Inexpert users should log on as “pccam” in order to avoid accidental damages to the Linux environment. Remember, it can be dangerous to work in the root account unless you really need to.

Another way to connect to PCCam is using network protocols such as telnet and ssh (see section 5.5).
The default IP address is **10.0.10.240**. If your network uses DHCP, ask to your system administrator in order know the IP address assigned to the camera. Otherwise PCCam_Viewer will find the address automatically (see section 5.4.3);

PCCam can be rebooted using the command `reboot`, or turned off with the command `poweroff` (or simply unplugging the power supply).

5.3 USEFUL COMMANDS

One of the easiest ways to find out how to use commands and applications is through the **man** command.
The word "man" stands for "manual," a series of online pages which tell you the purpose of many commands. In a highly condensed format, man pages provide a summary of a command's purpose, the options available, and the syntax which is used to issue the command.
Of course, like any good help system, the man command has its own man page. At the prompt, type:

```
# man man
```

Listed below are some of the more useful commands that are used under the PCCam Linux system. A complete command list can be found at the Appendix A.

5.3.1 useradd and userdel

The user “root” can add and delete users with these commands. For example

```
# useradd -m -s /bin/bash bob
```

adds user *bob* to the system, creates the /home/bob home directory and sets the login shell for this user. The user is made a member of the *users* group. By default the **useradd** only adds users, it does not set a password for the added user. Passwords can be set using the **passwd** command;

```
# userdel bob
```

removes the user’s account, not the user's home directory. Just add the *-r* parameter to delete the user's home directory too. For example

```
# userdel -r bob
```

5.3.2 passwd

Changes user’s password. Example:

```
# passwd bob
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

5.3.3 ifconfig

Changes network settings (only for user “root”). For example

```
# ifconfig eth0 10.0.10.190 netmask 255.255.255.0
```

sets the ip address to 10.0.10.190 and the network mask to 255.255.255.0. Shutting down PCCam removes the new network settings. In order to make the changes permanent use the program **netconfig** instead.

5.3.4 netconfig

This program sets up networking scripts under the PCCam system (only for user "root").
First, netconfig will ask you for a hostname and a domain name. Your hostname is the name by which your network sees your individual computer, and the domain name is the name you want to give your to network.
After setting up a hostname and domain name, you will be asked whether you want to configure a loopback only system, a DHCP system, or a static IP. You want to choose a static IP if you have been assigned one by your systems administrator or internet service provider or if you are setting up a network yourself. Static ip means the ip associated with your computer never changes. If your network uses DHCP to assign computers dynamic IPs, you will want to choose that option. A dynamic IP means that you get a new IP every time you sign on to the network.
Once the configuration is completed reboot PCCam.

5.3.5 startx

startx is a script located in */usr/bin* that starts an X session on PCCam. When executed from the console, startx sets up the appropriate environment variables, then runs xinit to start the X server.

5.3.6 loadkeys

The keyboard layout is controlled by the program `loadkeys`. This program accepts a keymap file as parameter.
All available keymaps are stored in `'/usr/share/kbd/keymaps/i386/'`. For example:

```
# loadkeys us.map
```

switches to the US layout (default), while

```
# loadkeys it.map
```

switches to the Italian keyboard layout.
The user can change the keymap loaded at boot-time by editing the file `/etc/rc.d/rc.keymap`.

5.3.7 sudo

`sudo` (superuser do) allows the system administrator to assign to a user (or to groups of users) the capability to run some (or all) commands as root or as another user. `sudo` determines who is an authorized user by consulting the file `/etc/sudoers`. Use the command `visudo` to edit the sudoers file. The sudoers file on PCCam is

```
# Cmnd alias specification
Cmnd_Alias      KILL = /sbin/poweroff
Cmnd_Alias      REBOOT = /sbin/reboot
# Defaults specification

# User privilege specification
root      ALL=(ALL) ALL
pccam     ALL=(ALL) NOPASSWD: ALL
ALL       ALL = NOPASSWD: KILL,REBOOT
```

The `sudoers` file is composed of two types of entries: aliases (basically variables) and user specifications (which specify who may run what). In the example above, the user pccam may run any command without entering a password, and any user may turn off or reboot PCCam without authenticating himself. For example, the user `pccam` could use `sudo` privileges to install a new software package

```
# sudo installpkg  xv-3.10a-i386-1.tgz
```

5.4 DISPLAYING LIVE IMAGES

There are three ways to view images captured by the CMOS sensor. Two of them need mouse, keyboard and monitor connected to PCCam. The third one takes advantage of Ethernet connection and needs only the Ethernet cable and the power supply.

5.4.1 pccam_viewer for PCCam Linux OS

If the monitor supports the 1024x768 resolution at 75 Hz, type `startx` and then press Enter. This command opens an X session. Press the right mouse button and open an xterm. On the command line type `pccam_viewer` and press Enter. The following window will appear:

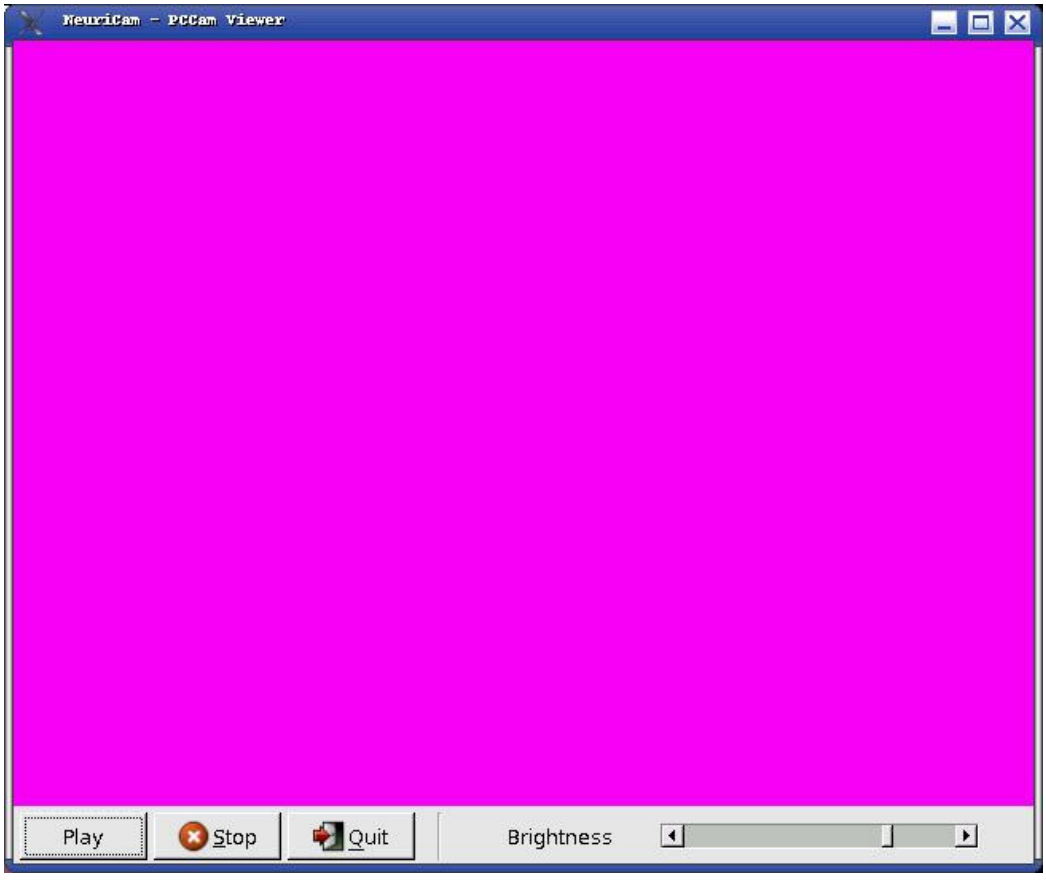


Figure 5.1: pccam_viewer screenshot

5.4.2 progressive_image for PCCam Linux OS

Without starting an X session, the root user can run `progressive_image` + Enter. The images will cover the whole display (even the shell prompt). Type `stop` + Enter to go back to the command line.

5.4.3 PCCam Viewer for MS Windows 98/2000/XP

PCCam_Viewer allows the user to acquire images from PCCam through the Ethernet connection.
Before trying to run the demo, it is necessary to check the PCCam network configuration. By default the PCCam settings are obtained using the DHCP protocol. DHCP enables automatic configuration of TCP/IP parameters for the camera. Your network must have a DHCP server running to use this feature.
If you are not sure whether your network has a DHCP server, ask your system administrator. Otherwise you can configure the network manually using **netconfig** (see section 5.3.4).

Installation steps:

1. Unzip **pccam_viewer-windows.zip** within a temporary folder on your computer.
2. Double click on "Setup.exe" to run the automatic installation program and follow the instructions.
3. The Setup program creates a program shortcut icon on the computer desktop automatically. A double-click on the icon will launch the program.

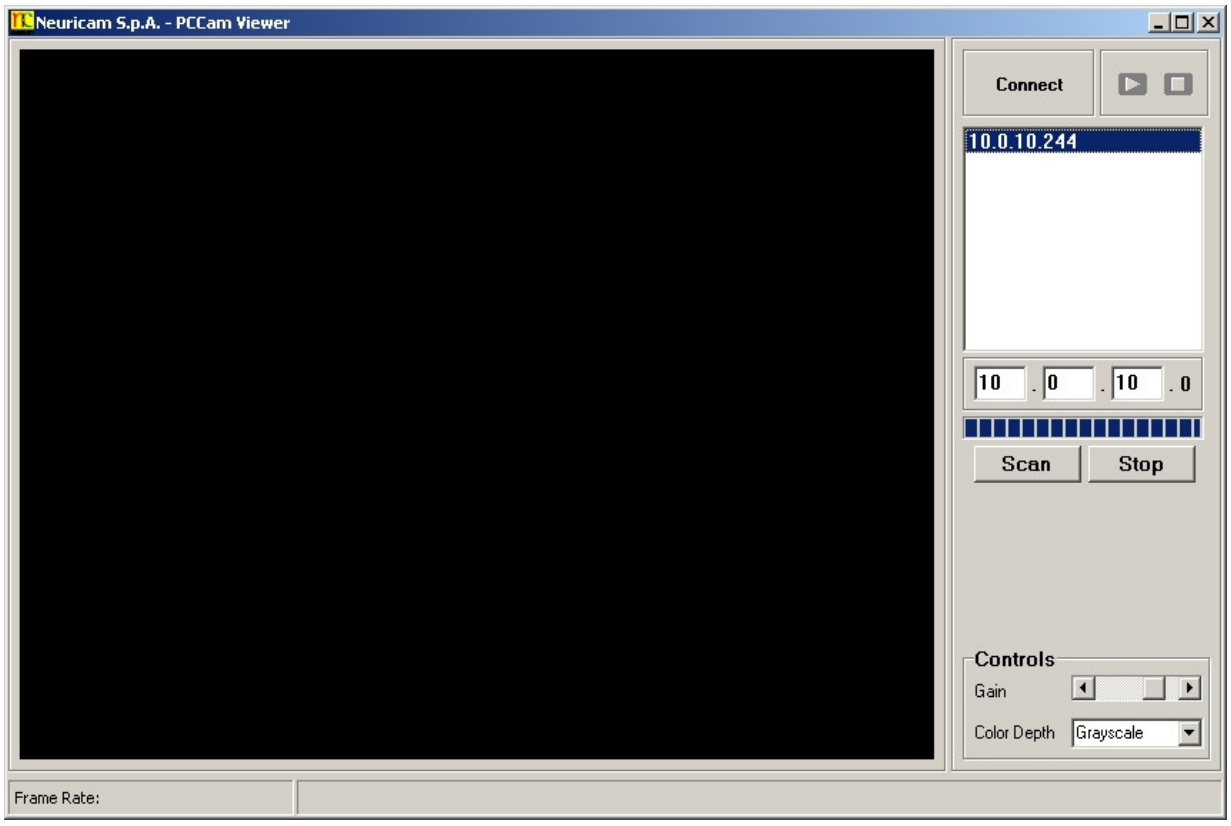


Figure 5.2: PCCam_Viewer screenshot

4. First of all fill the network address fields and then press the button "Scan".

5. If you have configured correctly the PCCam network settings, the camera IP address will appear on the list.
6. Select the address with the mouse, press “Connect” and “►”.
7. During the acquisition you can change the sensor gain and the color depth using the controls tools .
8. On the bottom of the window the current frame rate is displayed. It strongly depends on the network traffic and it can reach up to 12 frames per second.

5.5 PCCAM NETWORKING CAPABILITIES

PCCam supports several network protocols such as telnet, ftp, secure shell (ssh), nfs and samba.

5.5.1 Telnet and ftp

Telnet allows the user to connect to other computers from PCCam and vice versa and use materials (files, programs, etc) on the remote system. Syntax:

```
# telnet [computername]
```

Ftp allows the user to retrieve files from PCCam to other computers and vice versa. Syntax:

```
# ftp [computername]
```

Server and client of these protocols are both installed on PCCam.

5.5.2 Secure Shell (Ssh)

Ssh is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over unsecure channels. Server and client are both installed on PCCam.
The syntax is:

```
# ssh -l username [computername]
```

For more information, visit the website www.openssh.com.

5.5.3 NFS

Abbreviation of ***N****etwork* ***F****ile* ***S****ystem,* it is a client/server application that allows all network users to access shared files stored on computers of different types.

NFS was developed to allow machines to mount a disk partition on a remote machine as if it were on a local hard drive. This allows fast file sharing across a network.

With NFS, computers connected to a network operate as clients while accessing remote files, and as servers while providing remote users access to local shared files.

PCCam can both operate as client and server. The recovering of the system (chapter 5.7) is an example of a NFS client (PCCam) which accesses to shared files of a NFS server.

For more information, visit the website nfs.sourceforge.net.

5.5.4 Samba

Samba is a software that allows for interoperability between Linux/Unix servers and Windows-based clients.

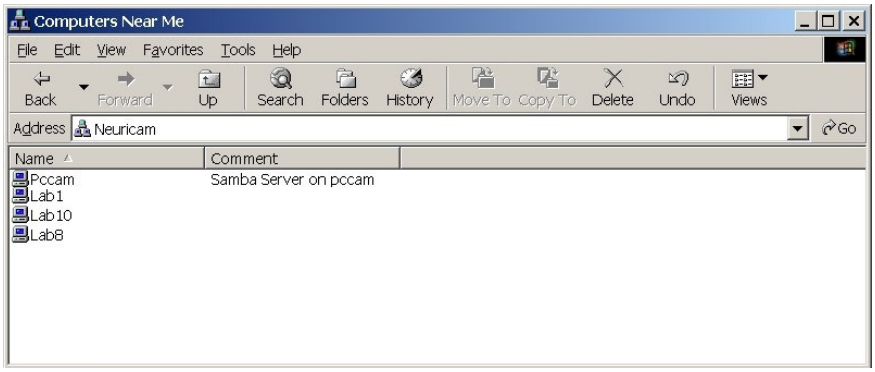


Figure 5.3: PCCam within the Windows network neighbourhood

`/etc/samba/smb.conf` is the configuration file for the samba suite. On PCCam it allows to browse only the users home directories. It is possible to add further folders editing that file.
For more information, visit the website www.samba.org.

5.6 ADDING/REMOVING PACKAGES

The PCCam OS is based on Slackware Linux 10.0, therefore it provides the same tools for the system management.
On the camera there are several programs already installed:

a. compilers (gcc, g++);
b. network utilities (nfs, ssh, ftp, ftpd, telnet);
c. the Xwindow system (Xfree86);
d. Xwindow utilities (xterm, xv, nedit);
e. file compressors (gzip, bzip2);
f. Linux manual pages (man).

New packages can be downloaded for free from Internet. Check for your nearest mirror site on www.slackware.com.
Once you have the package (e.g. xv-3.10a-i386-1.tgz) put it on PCCam via ftp, samba or ssh.
As root type:

```
# installpkg  xv-3.10a-i386-1.tgz
```

Before adding new packages, check the disk space usage with command **df**.

The complete list of installed packages can be found within /var/adm/packages.

The **removepkg** can be used to remove installed packages. For example, if you want to remove the "xv" package, you can execute:

```
# removepkg  xv
```

As you can see only the name of the package is specified in this example. You can also remove a package by specifying its full name:

```
# removepkg  xv-3.10a-i386-1
```

5.7 RECOVERY THE SYSTEM

Within the CD there is a large file called pccam.img.gz. This is the exact copy of the filesystem mounted on PCCam when the user turns it on for the first time.
This file is needed when the OS doesn't start. This usually happens when the user logs on the system as root and he/she removes or modifies system files.

5.7.1 Recovery PCCam booting from the recovery compact flash

The operations performed by this procedure is booting the Linux installed on the recovery compact flash and then downloading a working operating system from a nfs server to the primary disk of PCCam.

Tools needed:
- pccam.img.gz package (included in the PCCam CDROM);
- computer with Linux operating system and a network adapter;
- portmap and nfs-utils packages installed and configured on that computer.

1. Unzip pccam.img.gz and put pccam.img within an exported folder (e.g. "/pccam") on the nfs server. For example

```
# gunzip -c /mnt/cdrom/pccam.img.gz > /pccam/pccam.img
```

2. Insert the Compact Flash module and turn on the camera. Press F2 and enter the Bios setup.
3. In the "ATAPI Units 1st" section of the main menu, set Master to "None" and Slave to "Auto" using PageUp and PageDown keys (see **Figure 5.4**);

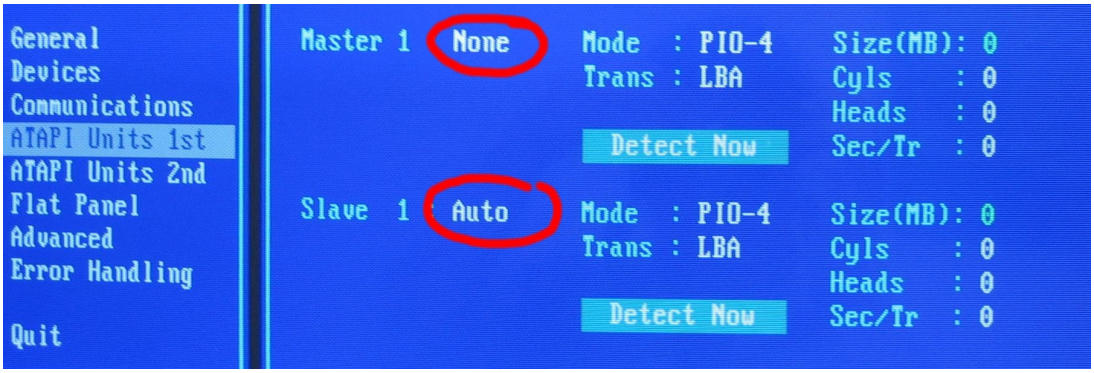


Figure 5.4: Bios setup

4. Select "Quit" in the main menu and save the changes. Now the system boots from the recovery Compact Flash.
5. Once the boot is completed log in as root (passwd: pccam2003) and mount the exported folder with command:

```
# mount 10.0.10.1:/pccam /mnt/hd
```

   where 10.0.10.1 is the IP address of the nfs server, /pccam is the exported folder on the server and /mnt/hd is the local folder on the camera where /pccam will be mounted.
6. Now it is possible to copy the recovery image on the Disk on Module with command

```
# dd if=/mnt/hd/pccam.img of=/dev/hda
```

It takes about twenty minutes. When the command prompt reappears, turn off the camera with the command "poweroff" and unplug the power supply.

7. Remove the Compact Flash and turn on PCCam.
8. Press the key F2 and enter in the Bios setup.
9. In the "ATAPI Units 1st" section of the main menu, set Master to "Auto" and Slave to "None" (see **Figure 5.5**);

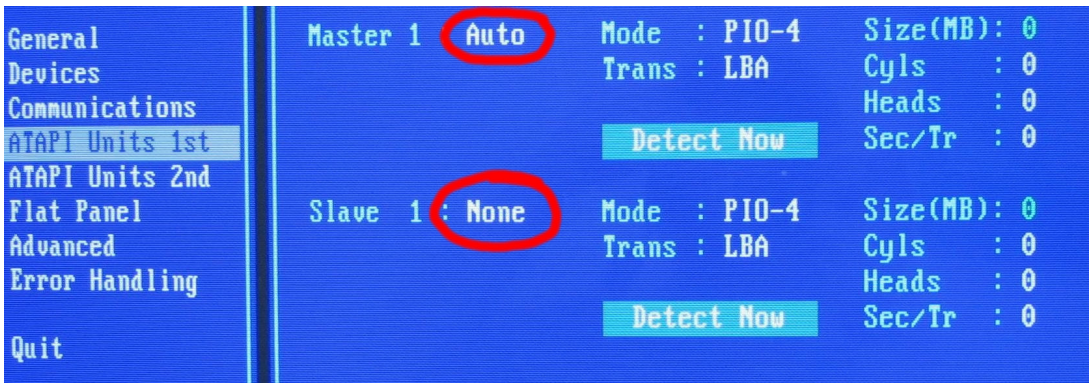


Figure 5.5: Bios setup

10. Select "Quit" in the main menu and save the changes. Now the system is completely restored.

5.8 USING A USB FLASH DRIVE WITH PCCAM

A USB flash drive is a small self-powered drive that connects to a computer directly through a USB port. Flash drives can hold any type of data, including image, video, and text files.
`Flash drives are also commonly referred to as key drives, thumb drives, jump drives, USB drives and pen drives. They come with varying amounts of memory (128MB seems to be the most popular choice).`
Using a USB flash drive under the PCCam operating system is very simple. First of all plug the device to the camera using the adapter included in the PCCam kit.
Now you need to mount the device to folder `/mnt/usb-drive` using command

```
# mount /mnt/usb-drive
```

After that, you may create, copy and delete files on the flash drive doing these actions within folder `/mnt/usb-drive`.
Once the drive is no longer needed, do not immediately remove it from the USB port. Instead, use the `umount` command in the following way

```
# umount /mnt/usb-drive
```

Otherwise, the files could not be copied into the drive.
After that you can unplug the device from the USB port of PCCam.
Note: NTFS-formatted flash drives are unsupported.

5.9 COMPILING THE KERNEL (ONLY FOR LINUX EXPERTS)

General information on how to compile Linux kernel can be found on the net. This section explains how to configure the kernel source code for the PCCam system.
Since the official Linux kernel doesn't provide a driver for the camera frame buffer, it is necessary to apply a patch to the kernel source code before compile it. This patch can be found on the CDROM (nsc-kfb-driver-2.7.7.tar.gz).
On the PCCam CDROM, there is a package containing the kernel source code already patched and configured. In case you need to add or remove some drivers, unzip the kernel archive within a temporary folder on your Linux workstation

```
#tar xvfj linux-2.4.22.tar.bz2
```

Configure and compile the kernel as usual.

6 PROGRAMMING PCCAM

The PCCam environment provides a set of tool for developing software applications. Programs can be written in the C language or in the x86 assembly language. The assembly language, and in particular the MMX instructions, should be used for computationally intensive and repetitive operations. These core functions can then be included in C/C++ programs that perform complex, high-level control operations.

6.1 HOW TO WRITE AND COMPILE PROGRAMS

To write programs in C/C++ or assembly language, a common text editor can be used. For example, the ***emacs*** and ***vi*** editors can be found in every Linux distribution. Another editor is **NEdit** (already installed on PCCam). NEdit is a multi-purpose text editor for X Window Systems with a standard, easy-to-use graphical user interface.
C files have to be saved with the *.c* extension, C++ files with the *.cc* extension. Assembly language files have to be saved with the *.s* extension.

To compile a C program simply type

```
# gcc -O2 -mmmx myProg.c -o myProg
```

To compile a C++ program type

```
# g++ -O2 -mmmx myProg.cc -o myProg
```

On the system there are both compilers and all the libraries needed. Also users can compile the source code on their own Linux workstation (usually faster and more user friendly than an embedded system) and place the executable on PCCam via ftp, nfs or ssh protocols.

6.2 USING THE ACQUISITION DRIVER

The Video4Linux driver for PCCam combines features for video overlay and video data capture. This kernel module allows the user to interact with the frame grabber and acquire images from the CMOS sensor.
The driver name is **pccamfg.** The examples below assume that the module has been successfully loaded before use (command `lsmod`, for the root user, lists all the loaded drivers).

Applications using this driver must include the headers:

```
#include <linux/videodev.h>
#include "pccam_v4l.h"
```

The first one contains the Video4Linux driver definitions, the second one (it has to be present in the working directory) contains additional ioctl controls for PCCam.

6.2.1 Opening and closing the device

The device may be opened and closed as follows

```
int vid = 0;

vid = open("/dev/video",O_RDWR);
if(vid < 0)
{
  printf("open /dev/video: failed!!!\n");
  exit(0);
}
…
…
…
close(vid);
```

Only one application at a time can open the device; the function "open" returns -1 in case "/dev/video" is already opened.

6.2.2 Configuring the device

Before acquiring images, the user must configure the device. There are two main steps

- setting color depth and brightness;

```
struct video_picture vid_pict;
vid_pict.depth = 8;                  /*  8 bits grey image */
vid_pict.brightness = 0x30;          /*  min=0x00 max=0x3F */
ioctl(vid, VIDIOCSPICT,&vid_pict);    /* sending settings to the driver */
```

- setting image width and height.

```
struct video_window vid_win;
vid_win.width = 640;
vid_win.height = 480;
ioctl(vid,VIDIOCSWIN,&vid_win);      /* sending settings to the driver */
```

See the table below for more details on `video_picture` and `video_window` structures.
After the device is configured, the user must wait for about 33 milliseconds before acquiring images, otherwise the first image can be corrupted.

6.2.3 Capturing images

The user can capture a single image using the `read` function. If the device is not configured as described in the previous section, `read` returns a negative number.

```
unsigned char image[640*480];
lseek(vid,0,SEEK_SET);
if(read(vid,image,sizeof(image))<0)
{
  printf("read: /dev/video not configured!\n");
  exit(0);
}
```

More details on errors can be obtained running the program `dmesg` on the command line.

6.2.4 Video overlay setup

Video overlay enables the display of full-motion video on monitor screen. When using video overlay, the CPU does not process information, saving valuable processing power for other applications.
Before the driver can display an overlay window, it needs to know where the window should be placed, and also how large it should be. The video_window structure is used to describe how the image is displayed. `VIDIOCSWIN` is the related IOCTL.
Video overlay appears wherever graphics pixel match the chromakey value within the window:

```
struct video_picture vid_pict;
vid_pict.depth = 16;                /*  16 bit RGB image */
vid_pict.brightness = 0x30;
ioctl(vid, VIDIOCSPICT,&vid_pict);

struct video_window vid_win;
vid_win.x = x;
vid_win.y = y;
vid_win.width = 640;
vid_win.height = 480;
vid_win.chromakey = 0x00F000F0
ioctl(pccam,VIDIOCSWIN,&vid_win);
```

When the video window is configured, it is possible to turn on the capture/overlay. This is done with the `VIDIOCCAPTURE` IOCTL. This takes a single integer argument, where 1 is on and 0 is off.

```
int arg = 1;
ioctl(pccam,VIDIOCCAPTURE,&arg);
```

The pccam_viewer source code ("Demo" folder on the CD ROM) is a good example of an application that uses the video overlay technique.

Struct video_picture fields

Depth	Image color depth. The user can choose one of the following values: 7 – "fast 8-bit"; fast acquisition of a grayscale image with values in the 1-254 range; 8 – grayscale image with values ranging from 0 to 255; 10 – grayscale image with values ranging from 0 to 1023; 12 – grayscale image with values ranging from 0 to 4095; 16 – color image in the 565 16-bit RGB format; 24 – color image in the 24-bit RGB format.
Brightness	is the image brightness with values ranging from 0 to 63.

Struct video_window fields

Width	The width in pixel of the desired image. The value has to be a multiple of 4 ranging between 64 and 640.
Height	The height in pixel of the desired image. The value has to be a multiple of 4 ranging between 48 and 480.
X	The X position of the top-left corner of the window where the picture is displayed.
Y	The Y position of the top-left corner of the window where the picture is displayed.
Chromakey	The colour (expressed in RGB32 format) for the chromakey colour if chroma keying in being used

6.2.5 Additional ioctl controls

The `pccamfg` driver supports additional ioctl controls out of the Video4Linux standard.

The **`VIDIOCSIMAGE`** ioctl is used when it is necessary to display images captured using the `read` function. Before capturing, the user has to set the video window (chromakey included) beyond the video picture structure:

```
#include <linux/videodev.h>
#include "pccamfg_v4l.h"
unsigned char Frame[640*480];



struct video_picture vid_pict;
vid_pict.depth = 8;                  /*  8 bit gray image (7,16 and 24  */
vid_pict.brightness = 0x30;          /*  also possible)                 */
ioctl(vid, VIDIOCSPICT,&vid_pict);

vid_win.x = x;
vid_win.y = y;
vid_win.width = 640;
vid_win.height = 480;
vid_win.chromakey = 0x00F000F0;
ioctl(pccam,VIDIOCSWIN,&vid_win);

unsigned char arg = 1;            /* set arg=0 to disable this feature  */
ioctl(pccam,VIDIOCSIMAGE,&arg);

lseek(pccam,0,SEEK_SET);
read(pccam,Frame,sizeof(Frame));
```

After calling `read`, the image is stored in `Frame` and displayed on the window of colour 0x00F000F0 placed at the point (x,y).

The general purpose inputs and outputs (see Chapter 3) can be controlled using the ioctl commands **`VIDIOCGIO`** and **`VIDIOCSIO`**. The syntax is very simple:

```
unsigned char value;
ioctl(pccam,VIDIOCGIO,&value);  /* reading input 1=IN24V_1, 2=IN24V_2 */
```

or

```
unsigned char value;
value = 1;                        /* 1=OUT24V_1, 2=OUT24V_2 */
ioctl(pccam,VIDIOCSIO,&value);    /* writing output */
```

However, the best way to synchronize image acquisition with an external trigger (hooked, let us say, to the input `IN24V_1`), is putting the camera in the triggered mode using the ioctl command **`VIDIOCSTRIGGER`**. This command takes a single argument that indicates the pin used for the trigger. For example, 1 means `IN24V_1`, 2 means `IN24V_2`, 3 both pins and 0 returns to the continuous mode.

```
int arg = 1;
ioctl(pccam,VIDIOCSTRIGGER,&arg);
```

In the triggered mode the `read` function doesn't exit until a rising edge arrives on the selected pin.
In order to avoid hanging the program when the `read` function is waiting for the trigger, it is recommended to put that function within a separated thread (see the example below).

By default each pixel of the sensor integrates light incident on it for the duration of a frame acquisition (33 milliseconds).
In some case is desiderable to reduce this time without changing the frame rate (e.g. when shooting moving objects).
Users can set the integration time through the **`VIDIOCSINT`** ioctl command.

```
unsigned short value = 400;
ioctl(pccam,VIDIOCSINT,&value);
```

`value` can range between 0 and 504:
- `value = 0` means full frame integration (33 ms);
- `1 ≤ value ≤ 504` means (33*value)/504 ms.

To read the current integration time use the **`VIDIOCGINT`** ioctl command.

```
unsigned short value;
ioctl(pccam,VIDIOCGINT,&value);
```

6.2.6 Examples

Here is an example of triggered acquisition using the `read` function within a separated thread. A second example can be found within the "Demo" folder on the CD-Rom (pccam_viewer-src.tgz). It describes how to implement a video overlay application .

```
/*-------------------------------------------------------------------------
Neuricam SpA   - Trento - Italy
http://www.neuricam.com
email:  techsupport@neuricam.com , info@neuricam.com
-------------------------------------------------------------------------
Filename: acq_example.c
-------------------------------------------------------------------------
Project:       <PCCam SDK>
Author(s):     <Roberto Doriguzzi>
First release: <May 2004>
Description:   <simple acquisition program>
-------------------------------------------------------------------------
<gcc options>
 gcc -O2 -mmmx -Wall acq_example.c -o acq_example -lpthread
-------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/videodev.h>
#include "pccamfg_v4l.h"

#define NX 640
#define NY 480
```

```
void *functionC();
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
int  pccam = 0;
FILE *out;

int  main()
{
   int rc1;
   pthread_t thread1;
   struct video_picture vid_pict;
   struct video_window vid_win;
   unsigned char arg;

   /* opening the device */

   pccam = open("/dev/video",O_RDWR);
   if(pccam < 0)
   {
     printf("open /dev/video: failed!!!\n");
     exit(0);
   }

   /* setting the picture properties */

   vid_pict.depth = 7;
   vid_pict.brightness = 0x1F;
   ioctl(pccam,VIDIOCSPICT,&vid_pict);

   /* setting the image size */

   vid_win.width = NX;
   vid_win.height = NY;
   ioctl(pccam,VIDIOCSWIN,&vid_win);

   /* setting the triggered mode on */

   arg = 1; /* pin IN24V_1 */
   ioctl(pccam,VIDIOCSTRIGGER,&arg);

   /* waiting for 33 milliseconds */

   usleep(33000);

   /* Creating independent thread, which will execute functionC */

   if( (rc1=pthread_create( &thread1, NULL, &functionC,NULL) ) )
   {
     printf("Thread creation failed: %d\n", rc1);
   }

   /* Wait till threads are complete before main continues. Unless we  */
   /* wait, we run the risk of executing an exit that terminates       */
   /* the process and all threads before the threads have completed.   */

   pthread_join( thread1, NULL);

   /* setting the triggered mode off */

   arg = 0;
   ioctl(pccam,VIDIOCSTRIGGER,&arg);

   /* closing the device */

   close(pccam);
   return 0;
}
```

```
void *functionC()
{
  unsigned char Frame[NX*NY];

  lseek(pccam,0,SEEK_SET);
  read(pccam,Frame,NX*NY);        /* acquiring the image */

  out = fopen("img.raw","wb");
  fwrite(Frame,NX*NY,1,out);      /* saving the image    */
  fclose(out);
}
```

Saved images can be displayed using the software **xv**. This program cannot open the "raw" images (sequences of pure binary numbers) acquired from the PCCam frame grabber. These images have to be converted to a standard image format (GIF, JPEG, TIFF, PBM, PGM, PPM, etc.).
Linux provides a set of useful functions for converting raw images to pgm (greyscale) or ppm (24-bit RGB) formats.

rawtopgm - converts raw greyscale bytes to a PGM image. Example for a 640x480 raw image:

```
# rawtopgm 640 480 img.raw > img.pgm
```

rawtoppm - converts a stream of raw RGB bytes to a PPM image. Example

```
# rawtoppm 640 480 img.raw > img.pgm
```

Displaying the image in a window on the screen using **xv**

```
# xv img.pgm
```

6.3 OPENING A SOCKET CONNECTION WITH PCCAM

As described in section 5.4.3, the PCCam CDROM contains a simple program (PCCam_Viewer) which enables acquiring images through the Ethernet connection. On the camera, there is a server (`/usr/sbin/imgserver`) that waits for TCP/IP requests on port 5351. It accepts only one connection at a time. When the connection is established, the client (e.g. PCCam_Viewer) can send commands to acquire images or to set the gain of the sensor and the colour depth of the image.

6.3.1 Commands table

pic	Sets some picture properties like color depth and brightness (changing the sensor gain). The command has to be followed by two 8-bit values, the color depth (7 for grayscale and 16 for 16-bit color) and the sensor gain (ranging from 0 to 63). See section 6.3.3.
img	Start the image acquisition process on the camera. When the image is ready, the server sends it to the client.(see section 6.3.4).

The main steps for creating a client application are explained in the following part (see also the PCCam_Viewer source code).

6.3.2 Opening the connection

The client connects to the server using the standard socket library:

```
#define PORT 5351
SOCKET sock;
struct sockaddr_in serv_addr;
sock = socket(AF_INET,SOCK_STREAM,0);

memset ((char *)&serv_addr,0,sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port   = htons(PORT);                    // service port
serv_addr.sin_addr.s_addr = inet_addr("10.0.10.1");  // PCCam address

if(connect(sock,(struct sockaddr *)&serv_addr, sizeof(sockaddr)) == -1)
{
  exit(0);
}
```

For windows clients, `<winsock2.h>` must be included; for Linux include `<sys/socket.h>`.

6.3.3 Setting the color depth and the sensor gain

Before acquiring images from the camera, it is necessary to set the image color depth (grayscale or 16-bit color) and the sensor gain level (ranging from 0 to 63).

```
unsigned char pic_depth;
unsigned char pic_gain;

pic_depth = 7;               // 7 = greyscale, 16 = 16-bit color
pic_gain  = 50;              // range 0-63

SendString(sock,"pic");                    // sending the command
Send(sock,&pic_depth,sizeof(pic_depth));  // sending the color depth
Send(sock,&pic_gain,sizeof(pic_gain));    // sending the sensor gain
```

Send, SendString, Recv and RecvString are special function defined in “Main.h” (PCCam_Viewer source code). They have a very similar structure, e.g.:

```
int Send(int fd,void *buf,int len)
{
  int sum = 0;
  int bytes = 0;
  int bytesleft = len;
  char * buffer;

  buffer = (char *)buf;
  while(sum < len)
  {
    bytes = send(fd,buffer+sum,bytesleft, 0);
    if(bytes > 0) {sum += bytes; bytesleft -= bytes;}
    else return bytes;
  }
  return sum;
}
```

`send` is a function of the standard socket library. The while loop is required to send all the data contained in buffer `buf`.

6.3.4 Asking for images

This command is usually executed within a loop of a dedicated thread. In this way the acquisition process can not stall the main program (see the PCCam_Viewer source code).

```
int imagesize = 640*480;

void ThreadLoop::Execute()
{
  while(!Terminated)
  {
    SendString(sock,"img");                // sending the command
    Recv(sock,(char *)Buffer, imagesize);  // receiving the image
  }
}
```

For a single image the loop in not needed.

6.3.5 Closing the connection

```
shutdown(sock,2);
```

APPENDIX A LINUX COMMAND LIST

This appendix briefly describes the Linux commands and programs used for various functions on the system. For more details read the **man** pages (e.g. `#man chgrp`).

alias	Create an alias
awk	Find and Replace text within file(s)
break	Exit from a loop
builtin	Run a shell builtin
cal	Display a calendar
case	Conditionally perform a command
cat	Display the contents of a file
cd	Change Directory
chgrp	Change group ownership
chmod	Change access permissions
chown	Change file owner and group
chroot	Run a command with a different root directory
cksum	Print CRC checksum and byte counts
clear	Clear terminal screen
cmp	Compare two files
comm	Compare two sorted files line by line
command	Run a command - ignoring shell functions
continue	Resume the next iteration of a loop
cp	Copy one or more files to another location
cron	Daemon to execute scheduled commands
crontab	Schedule a command to run at a later time
csplit	Split a file into context-determined pieces
cut	Divide a file into several parts
date	Display or change the date & time
dc	Desk Calculator
dd	Data Dump - Convert and copy a file
declare	Declare variables and give them attributes
df	Display free disk space
diff	Display the differences between two files
diff3	Show differences among three files
dir	Briefly list directory contents
dircolors	Colour setup for `ls'
dirname	Convert a full pathname to just a path
dirs	Display list of remembered directories
du	Estimate file space usage
echo	Display message on screen
ed	A line-oriented text editor (edlin)
egrep	Search file(s) for lines that match an extended expression
enable	Enable and disable builtin shell commands
env	Display, set, or remove environment variables
eval	Evaluate several commands/arguments
exec	Execute a command
exit	Exit the shell
expand	Convert tabs to spaces
export	Set an environment variable
expr	Evaluate expressions
factor	Print prime factors
false	Do nothing, unsuccessfully
fdformat	Low-level format a floppy disk
fdisk	Partition table manipulator for Linux
fgrep	Search file(s) for lines that match a fixed string

find	Search for files that meet a desired criteria
fmt	Reformat paragraph text
fold	Wrap text to fit a specified width.
for	Expand words, and execute commands
format	Format disks or tapes
free	Display memory usage
fsck	Filesystem consistency check and repair.
function	Define Function Macros
gawk	Find and Replace text within file(s)
getopts	Parse positional parameters
grep	Search file(s) for lines that match a given pattern
groups	Print group names a user is in
gzip	Compress or decompress named file(s)
hash	Remember the full pathname of a name argument
head	Output the first part of file(s)
history	Command History
hostname	Print or set system name
id	Print user and group id's
if	Conditionally perform a command
import	Capture an X server screen and save the image to file
info	Help info
install	Copy files and set attributes
join	Join lines on a common field
kill	Stop a process from running
less	Display output one screen at a time
let	Perform arithmetic on shell variables
ln	Make links between files
local	Create variables
locate	Find files
logname	Print current login name
logout	Exit a login shell
ls	List information about file(s)
m4	Macro processor
man	Help manual
mkdir	Create new folder(s)
mkfifo	Make FIFOs (named pipes)
mknod	Make block or character special files
more	Display output one screen at a time
mount	Mount a file system
mtools	Manipulate MS-DOS files
mv	Move or rename files or directories
nice	Set the priority of a command or job
nl	Number lines and write files
nohup	Run a command immune to hangups
passwd	Modify a user password
paste	Merge lines of files
pathchk	Check file name portability
popd	Restore the previous value of the current directory
pr	Convert text files for printing
printenv	Print environment variables
printf	Format and print data
ps	Process status
pushd	Save and then change the current directory
pwd	Print Working Directory
ram	ram disk device
rcp	Copy files between two machines.

read read a line from standard input
readonly Mark variables/functions as readonly
return Exit a shell function
rm Remove files
rmdir Remove folder(s)
rpm Remote Package Manager
rsync Remote file copy (Synchronize file trees)
screen Terminal window manager
sdiff Merge two files interactively
sed Stream Editor
select Accept keyboard input
seq Print numeric sequences
set Manipulate shell variables and functions
shift Shift positional parameters
shopt Shell Options
shutdown Shutdown or restart linux
sleep Delay for a specified time
sort Sort text files
source Run commands from a file `.'
split Split a file into fixed-size pieces
su Substitute user identity
sum Print a checksum for a file
sync Synchronize data on disk with memory
tac Concatenate and write files in reverse
tail Output the last part of files
tar Tape ARchiver
tee Redirect output to multiple files
test Evaluate a conditional expression
time Measure Program Resource Use
times User and system times
touch Change file timestamps
top List processes running on the system
traceroute Trace Route to Host
trap Run a command when a signal is set(bourne)
tr Translate, squeeze, and/or delete characters
true Do nothing, successfully
tsort Topological sort
tty Print filename of terminal on stdin
type Describe a command
ulimit Limit user resources
umask Users file creation mask
umount Unmount a device
unalias Remove an alias
uname Print system information
unexpand Convert spaces to tabs
uniq Uniquify files
units Convert units from one scale to another
unset Remove variable or function names
unshar Unpack shell archive scripts
until Execute commands (until error)
useradd Create new user account
usermod Modify user account
users List users currently logged in
uuencode Encode a binary file
uudecode Decode a file created by uuencode
vdir Verbosely list directory contents (`ls -l -b')

REFERENCES

- Manual and application notes for the Geode processor development boards can be found at: http://wwwd.amd.com/amd/developer.nsf/ (registration required).
- More information about Video4Linux can be found at: http://www.exploits.org/v4l/.
- The Slackware Linux project home page: http://www.slackware.com.