5-31-1990

A micro-based machine vision system

Hsueh-Chung Michael Feng
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

A MICRO-BASED MACHINE VISION SYSTEM

by

Hsueh-Chung Michael Feng

Thesis submitted to the faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1990

APPROVAL SHEET

Title of Thesis: A Micro-based Machine Vision System

Name of Candidate: Hsueh-Chung Michael Feng
Master of Science in Electrical Engineering, 1990

Thesis and Abstract Approved :

5/29/90

Dr. Anthony Robbi, Advisor — Date
Associate Professor
Department of Electrical and Computer Engineering
New Jersey Institute of Technology

5-22-90

Dr. C. Manikopoulos — Date
Associate Professor
Department of Electrical and Computer Engineering
New Jersey Institute of Technology

5-22-90

Dr. E. Hou — Date
Assistant Professor
Department of Electrical and Computer Engineering
New Jersey Institute of Technology

VITA

Name: Hsueh-Chung Michael Feng

Address:

Degree and date to be conferred: M.S.E.E., 1990

Date of birth:

Place of birth:

Collegiate institutions attended	Dates	Degree	Date of Degree
Fu-Jen Catholic University, Taiwan, ROC	9/80-6/84	B.E.E.E.	July, 1984
New Jersey Institute of Technology	9/88-5/90	M.S.E.E	May, 1990

Major: Electrical Engineering

Position held: Engineer

ABSTRACT

Title of Thesis: A Micro-based Machine Vision System

Hsueh-Chung Michael Feng, Master of Science in Electrical Engineering, 1990

Thesis directed by: Dr. Anthony Robbi

Department of Electrical and Computer Engineering

In this thesis, a low-cost general purpose machine vision system based on a personal computer of the PC-XT/AT family and an electronic camera controller based upon a Motorola 68HC11 single-chip microcontroller has been designed and implemented. This work is focused on image formation. The system supports a long distance camera-computer configuration. A picture editor, generic image manipulation software was designed and implemented. A single computer can supervise multiple cameras.

ACKNOWLEDGEMENT

I am grateful to Dr. Anthony Robbi for his continued support and guidance during this thesis. I would like to thank Dr. Manikopoulos for his providing research equipment, and Texas Instruments Inc. for their donating the TC211 image sensor chips and technical consulting. This work was partially funded by the Separately Budgeted Research Program of NJIT. Thanks also go to Mr. Xiao-Fang Qi, Mr. Zhen Zhu, Mr. Jason Chen and Mr. Mirko Delgado for their help.

FOREWORD

The concept of machine vision began about 30 years ago. It often refers to as computer vision or intelligent vision, is a means of electro-mechanically simulating the image recognition capability of the human eye system. It represents relatively complex subject drawing upon many technical disciplines such as image processing, pattern recognition, and artificial intelligence. Compared with the human eye system, machine vision might be to consider it a type of sensing capability, similar to the way the human eye acts as the body's vision sensor. The human eye receives light from an object and then converts this light into electrical signals. It does not interpret these signals. Image interpretation is performed by the brain. In the same way, a machine vision system includes both a visual sensing and an interpretive capability. An imaging device, such as a vidicon camera, is nothing more than a visual sensor which receives light and converts it into electrical signals. When a computer is employed, these signals can be refined and analyzed to allow interpretation of the scene which generated the signals. In other words, imaging alone can be though of a means of providing visual input signals to a computer for processing. The signal that is received by the computer is actually no different than the electrical signal generated through a keyboard or other input device, exact that the data rate is orders of magnitude greater.

Contents

List of Figures

Chapter 1

Introduction

1.1 Difference between machine vision system and optical sensing equipment

A complete machine vision system requires at least the following three basic capabilities: image formation, image analysis, and image interpretation. Image formation is the reception of incoming light from an object or a scene, conversion of the light into electrical signals, and then processing of the signals until they are in a form that is compatible with a computer. Image analysis is the computer's ability to analyze and measure various characteristics of the captured image. Image interpretation is the ability to interpret the image to support decision making. These requirements distinguish the machine vision system from the optical sensing equipment such as closed-circuit television systems used by human operators for inspection.

1.2 Applications of machine vision

In general, machine vision systems are suitable for use in three categories of manufacturing applications. First, they can be used for visual inspection

of a variety of parts, sub-assemblies, and finished products to insure that certain standards such as the dimensions are met or to verify parts features, such as inspection [20] of part profile and contour, crack detection, and measurement of length, width, and area. Second, they can be used to identify parts for sorting them into groups. For instance, optical character recognition, identification of parts for spray painting, and keyboard and display verification. Third, they are suitable for guidance and control applications, such as vision-assisted robot assembly or material handling. See [9] for the details.

As machine vision technology continues to develop and becomes more widely used within industry, additional applications are beginning to materialize. Many of these applications combine more than one of the functions mentioned before, while others represent special purpose equipment which has been designed to satisfy a particular need.

1.3 Thesis description

There are many optical sensor manufactures providing varieties of sensing equipments in the world. The one most related to this thesis is EDC-1000, a television-like monochrome camera, made by ELECTRIM Corporation in Princeton, New Jersey. The EDC-1000 operates under computer control. A TC211, CCD (charge-coupled device) detector, is used to operate the full frame image sensing. All timing and video signals are carried on a single multiconductor cable which connects the camera to a PC add-on card.

Image acquisition systems have also been discussed recently. One of these systems is the image analysis and data- acquisition techniques designed by K. G. Young and D. L. Hillis [21]. They coupled two video cameras with fast acquisition and display systems, developed for a Micro VAX-II. Data may be acquired at rates of 16.7 ms per one 60×80 eight-bit-pixel image frame or lower rates, up to about 0.5 s per frame.

In this thesis, a low-cost general purpose machine vision system based on a personal computer of PC-XT/AT family and an electronic camera controller based-upon a Motorola 68HC11 single-chip microcontroller have been designed and implemented. This work is focused on image formation. Image analysis and interpretation are primarily based upon application-specific software or sophisticated hardware [10]. The system supports a long distance camera-computer configuration. A picture editor in the PC acts as generic image manipulation software. A single computer can supervise multiple cameras.

Chapter 2

System Description

2.1 System configuration

The system discussed in this thesis consists of a camera with a CCD chip TC211 in it, an image grabber board with Motorola 68HC11 microcontroller on it, and a PC add-on card. The camera and the image grabber are connected together by a 9-pin short cable, which conducts the pulse trains to shift the image data out and leads the analog pixel signal down to the grabber board. In principle they could be incorporated into the same module. The connection between the image grabber and the PC add-on card is a 15-pin long cable, which provides a digital, serial channel for transmitting image data from the image grabber to the PC add-on card and receiving the commands from the PC add-on card to the image grabber.

Figure 2.1 is an overview of single camera configuration. For development purposes, another connection is built to attach a Motorola 68HC11 EVB (Evaluation Board) to the image grabber board.

A system with real applications could be more powerful than the one described above. A much bigger board or a set of boards might be used not

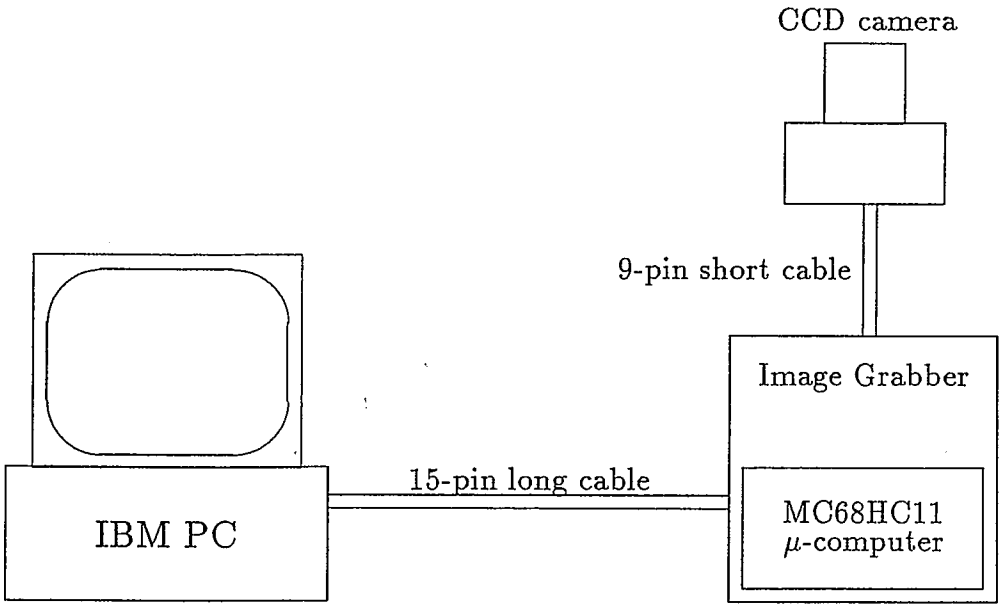


Figure 2.1: System overview

only for image grabbing but also for some other control functions because with the microprocessor, the image grabber has the intelligence to perform other functions. For instance, a stepping motor can be controlled by the 68HC11 to aim at a specified object. Since the line drivers and receivers used in the serial interface between the image grabber and PC add-on card satisfy the requirements of RS-422, the length of the cable can be up to 1200 meters depending on the transmission rate [2]. Besides that, a multi-camera configuration is also possible. With a proper arbitration mechanism, more than one image grabber board can share the same channel. However, the count of pins which can fit in the edge of a add-on card are quite limited. In this case, the author would like to suggest the use of a HUB board to collect all the signals and then send them to the add-on card as shown in Figrue 2.2. The circuits on the HUB should be very much similar to the multiple master SPI configuration described in [18]. The maximum rate of an image transmission to the PC is inversely related to the number of

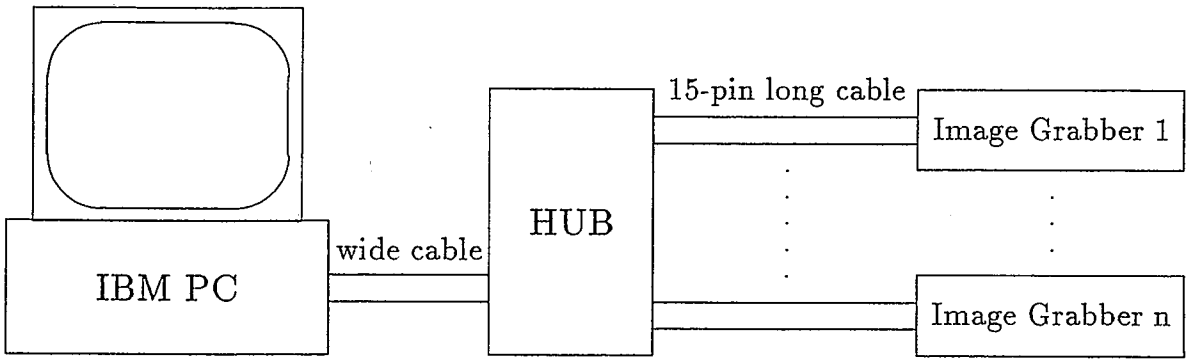


Figure 2.2: Multi camera configuration

cameras at a HUB.

2.2 Image grabber

In an early work, Mr. Chi-Hsin Lee described the use of microprocessor to drive a CCD chip to build a camera[4]. This work was extended by ordering a EDC-1000 from ELECTRIM Corporation and replacing the inside PCB by a new board, which contains a CCD chip TC211 and an OPAMP working as a voltage follower. The level shift circuits used by Mr. Lee in the image grabber are also changed to be much simpler, and the pulse trains for exposing the image and shifting out the video data are generated totally by software. The analog signal which represents the brightness of a pixel is connected to the Motorola 68HC11's built-in ADC in order to convert it to 8-bit digital pixel data. Figure 2.3 shows the block diagram of the image grabber.

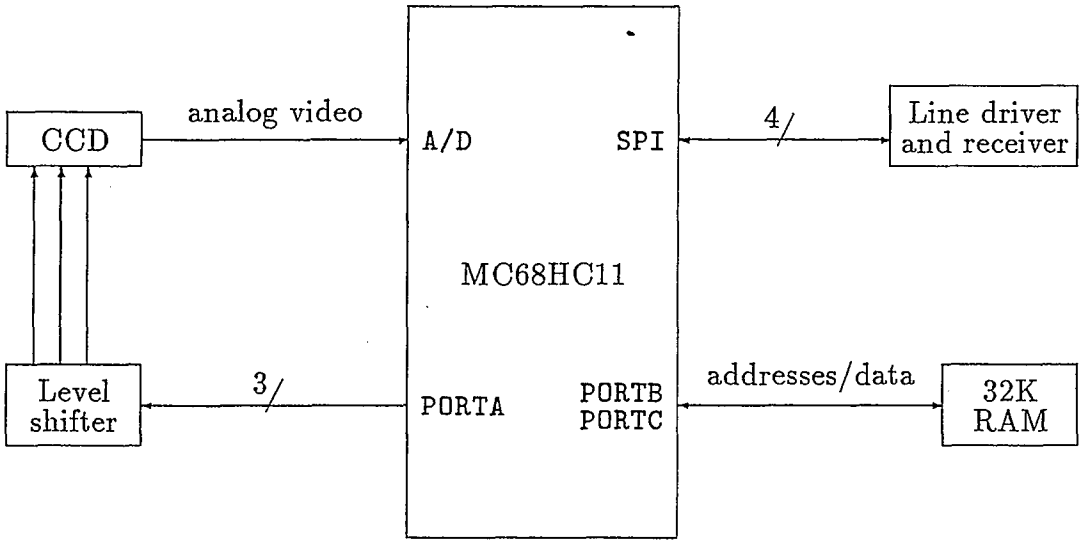


Figure 2.3: Block diagram of the image grabber

2.3 Communications between image grabber and PC

Inclusion of a microcomputer in the camera permits digital image transmission. The image grabber in this system converts the analog video signal to digital locally and then sends it out in digital, so the length of the cable can be much longer. With the help of 26LS31 and 26LS32 transceiver chips, which can support RS-422 requirements, the length can be dramatically increased up to 1000 meters. The EDC-1000 sends analog video output from the CCD chip to a PC add-on card and then converts there. Because of the noise sensitivity of the analog signals, the length of the cable should be short. That means the camera should be kept close to the PC.

Another advantage of adding a microprocessor inside optical sensing equipments is the general increase of capability. For instance, an intelligent

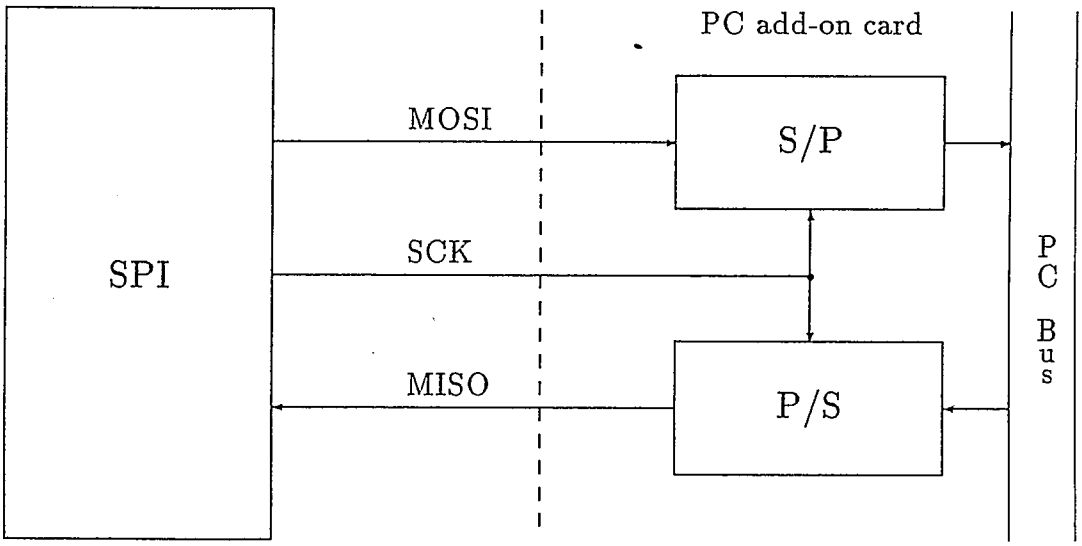


Figure 2.4: Block diagram of the communication interface

camera can change the exposure time as desired, perform autofocus, and do autoaiming in a predetermined pattern or under dynamic control of the PC.

The Motorola 68HC11 has a SPI (Serial Peripheral Interface) built in. The SPI performs synchronous serial data communications. Data streams can go out and come in synchronous with SCK simultaneously through ports MOSI and MISO, *i.e.* the channel is full duplex . The Motorola 68HC11 is assigned as the master, and it generates SCK clock during the transmission. Because the analog to digital conversion, which is being done at same time, is quite time related, it is better to have master authority at the image grabber side such that sending data on fly will be possible. Converted image data are sent out on MOSI, and commands from PC are picked up on MISO. Figure 2.4 shows the block diagram of this interface.

2.4 Image processing at the PC

Even though a microcontroller had been put in the image sensor, a PC-based system has much greater processing capability. Therefore, image data are transferred to PC to be processed rather than processed by the image sensor. In this work the processes implemented are display, editing, and print out.

2.4.1 Image display

In the PC, screen output relies on two components: the display adapter and the display monitor. The "adapter" is a hardware card that you plug into one of the slots inside the PC, and the "monitor" is the display screen where the actual characters and graphics appear. There are four major adapters available for PC: MDA(Monochrome Display Adapter), CGA(Color Graphics Adapter), EGA (Enhanced Graphics Adapter), and HGC(Hercules Graphics Card).

Like most peripheral devices in the IBM PC, the display adapters are programmed via 8-bit registers that are accessible by unique input port addresses. Besides, all display adapters are "memory-mapped." Each pixel on the display screen corresponds to 1 or more bits in a memory region called "video memory" or "video RAM". In graphics modes, for a black and white monitor, each pixel simply has to be either on or off, which means that a single bit is enough to display a pixel.

The details of the physical organization of the video RAM are bypassed by using the Turbo C graphics routines. Every adapter can display graphics at different resolutions, except the MDA, and the resolutions are twice in each dimension of the TC211, 192×165 image sensor. Therefore, the image can be displayed in pseudo gray level which turns different numbers of points on in a certain area to represent different gray level of a pixel. Therefore, we use four points on the screen to present one pixel in the image, and turn on three of them to represent the pixel has a gray level 3. This is usually called halftoning. It compensates for the fact that PC does not support a full gray level display.

2.4.2 Image editor

The image we got from the sensor is sometimes not desirable. For instance, there may be spots on it. The image editor gives the user the ability of modifying the image on screen. Not only can it correct the spots, but also it can add some characters or symbols to the image, *e.g.* a title or date.

2.4.3 Image printout

There are many desk top publishing packages which can print out image files. For instance, PageMaker, and Microsoft Word, provided by the Microsoft Corporation, can merge text and image files and print them out. There are also certain standard file formats for such image files, *e.g.*, the TIFF and PCX. As long as the usage file is properly formatted, we can take the advantages of those software packages instead of writing printer driver programs ourselves.

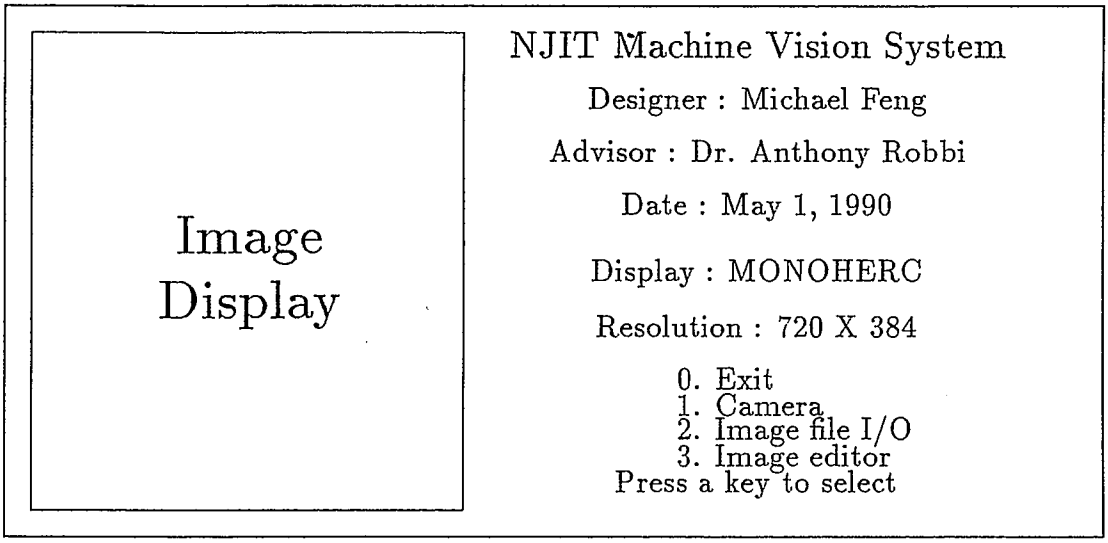


Figure 2.5: The main menu of NMVS

The TIFF (Tag Image File Format) is used in this system. The TIFF was defined jointly by Aldus and Microsoft in conjunction with leading scanner vendors and other interested parties. It is a tag based file format that is designed to promote the interchange of digital image data. The general scenario for which TIFF was invented assumes that applications software for scanning or painting creates a TIFF file, which can then be read and incorporated into a document or publication by an application such as a desktop publishing package.

2.5 System operations

The system is operated under the software NMVS (NJIT Machine Vision System). It is an execution file named NMVS.EXE, created by linking several programs written in Turbo C and Intel 8088 assembly language.

When the user types in NMVS, a main menu screen will be shown. See

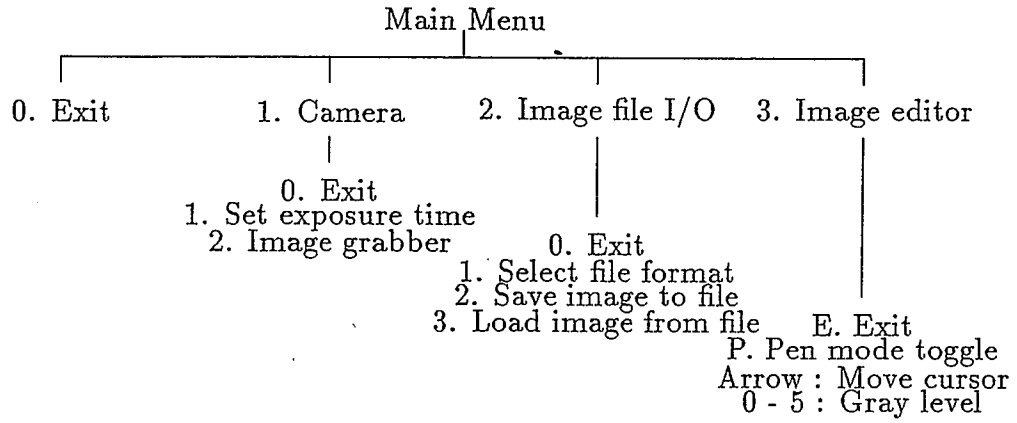


Figure 2.6: The selection tree

Figure 2.5. The left half side of the screen is the image display area, while the right half side is the selection menu. The user can give commands to the system by making selections from the menu instead of typing the commands. Almost every command that the user has to type is reduced to one key selection on screen. Figure 2.6 shows the selection tree of the system. Note that every screen has a consistent exit point.

The camera entry on the main menu is directly related to the image grabber. It invokes exposure time changing and image data reception routines which were written in assembly language. The image can be stored in a file for later processing, and can be read back. These are done under the image I/O entry. Two file formats, which are binary and TIFF, are provided. The binary file is a direct collection of the gray level values of each pixel, 31680 bytes for the TC211. The TIFF file is a standard image exchange format which is supported by other software. It includes header

information so more than 31680 bytes are needed for a TC211 image. The third selection is an image editor. It gives the user a chance to modify the image on screen. A blinking cursor can be moved by pressing the arrow keys on the keyboard. The pixel covered by the cursor can be modified to a desired gray level if the user turns on the pen mode. Refer to Appendix A for User's Guide.

Chapter 3

System Design

As described in the previous chapter, the system comprises an image grabber, which is a Motorola 68HC11-based system, and a host computer, which is an Intel 8088-based system. The major work done on this system is implementation of the SPI channel for data and command transfer. It can be extended to single-host, multiple-camera communication. Image display, image editing, and image printout file preparation are software modules specially developed for the host.

3.1 SPI and data/command transfer

3.1.1 Serial peripheral interface

The SPI (Serial Peripheral Interface) of 68HC11 is a synchronous interface which allows the microcontroller to communicate with other SPI-type devices. The clock is not included in the data stream and must be furnished as separate signal of the SPI channel. Three SPI signals is used here. They are MISO, MOSI, and SCK.

The MISO line is configured as an input in a master device and as an

output in a slave device, while the MOSI line is configured as an output in a master device and as an input in a slave device. Both of these lines transfer serial data with the most significant bit sent first. No framing bits are required because the mode is synchronous. The SCK clock synchronizes data flow both in and out through the MOSI and MISO lines. The device which generates the clock is called master. See Figure 3.2 for timing. Another signal used in the interface is the SCKREQ. It is a request signal and also handshakes signal from PC.

The connection between the image grabber and the PC is a 15- pin cable. Eight of the 15 pins are used, because every SPI signal are transceived in a differential mode over balanced lines. The other 7 pins are reserved for the arbitration in the multi-camera configuration. Pin assignment of cable is shown in sheet 4 of Appendix B.

3.1.2 Data/Command transfer

The circuit diagrams of the data and command transfer channel with the SPI of 68HC11 are shown in sheet 2 and 3 of Appendix B. The 68HC11 SPI is set to master mode to provide SCK. PC slave circuits are implemented by discrete chips such as 74166 PISO (parallel- in-serial-out), 74164 SIPO (serial-in-parallel-out) registers, and 7493 counter.

Command transfer always occurs before the data transfer. It is more complicated than the data transfer because it is initiated by PC. Therefore, it is explained after the data transfer is illustrated in the two succeeding

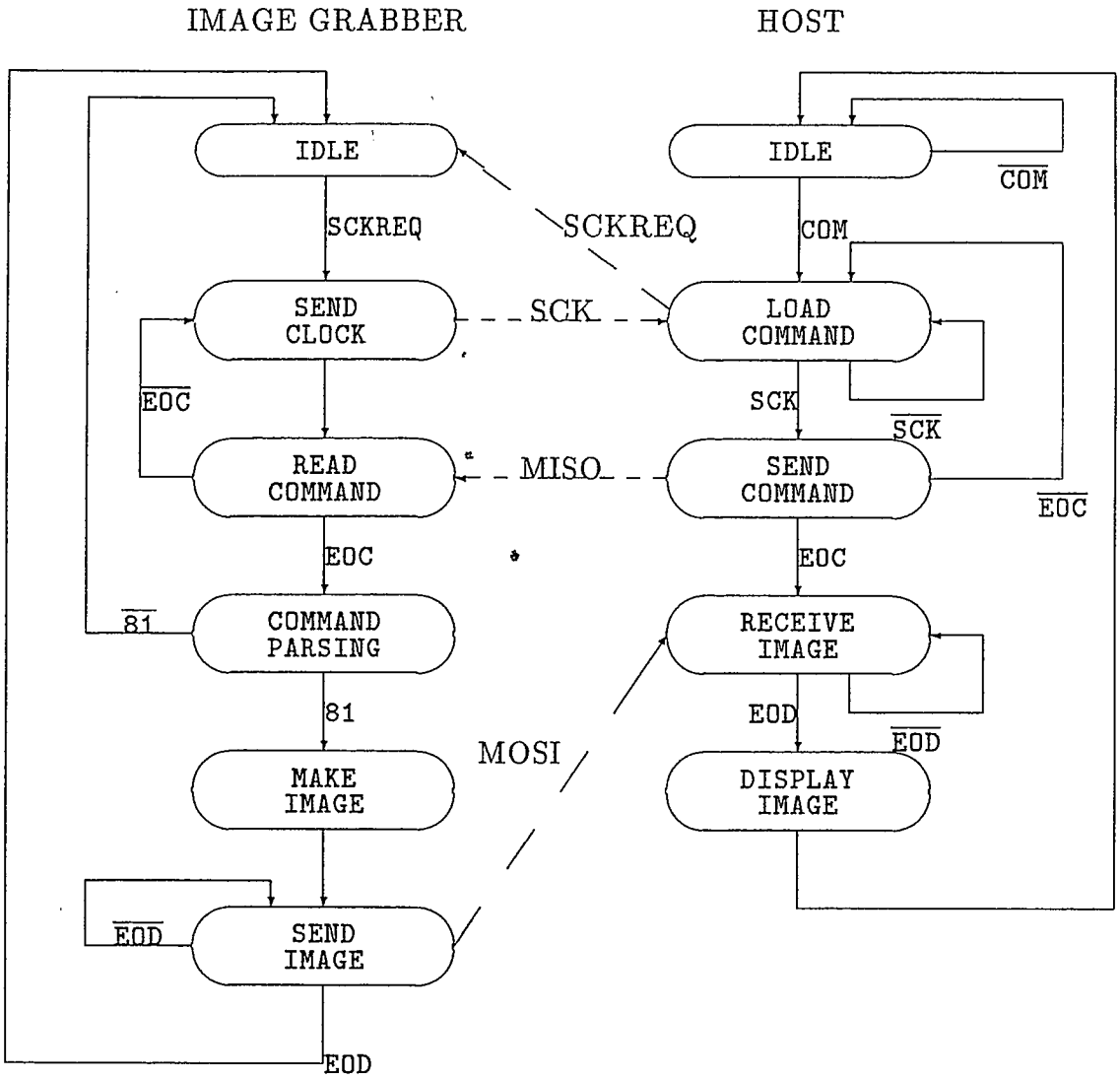


Figure 3.1: State Diagram of SPI

subsections. A command consists of a command code followed by a byte or a series of information and a CR byte. The information portion is written in 7-bit ASCII, while the command code and CR are codes in 8-bit ASCII with bit 7 set. For instance, the command for taking a picture with 100 μs exposure time is: `81, 31, 8D`, where `81` is the command code, `31` is the hundred digit of 100 in ASCII, and `8D` is the CR with bit 7 set. At this time only one command code, 81, is defined. More other commands can be defined in this format.

Figure 3.1 is the state diagram of the interface. The image data are transmitted on MOSI, and commands are received on MISO simultaneously. Only the master device can initiate data transmission or reception. If the PC needs to initiate a transmission, SCKREQ is used to request a command reception cycle while the image grabber is not sending data. Since the channel is full duplex, the data on MOSI should be thrown away in this case. On the other hand, a dummy command 00 is fed back during the data transmission period when PC has nothing to say.

3.1.3 Image grabber side

The SPI control register (SPCR), SPI status register (SPSR), and SPDR are software accessible registers used to configure and operate the SPI system. The port D data direction control register (DDRD) also influences SPI activities.

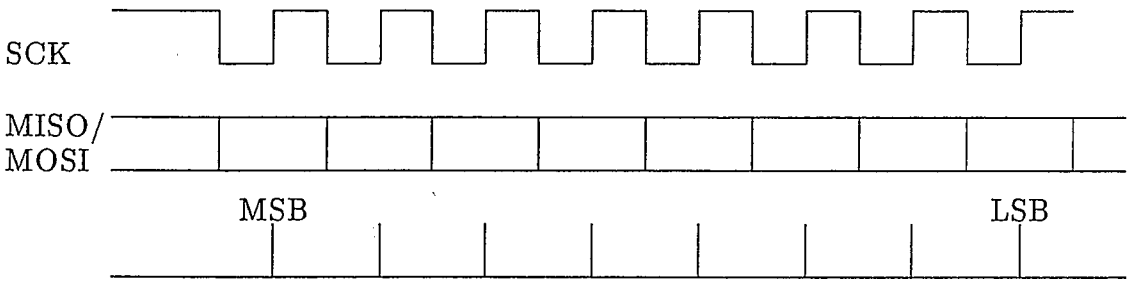


Figure 3.2: Data clock timing diagram

```
LDAA #%00111100
STAA DDRD
LDAA #%01011110
STAA SPCR
```

The 68HC11 program segment listed above initializes the SPI and transfers the data and commands. Bit 4, 3, and 2 of port D are used as SCK, MOSI, and MISO by the SPI in this system. Bits DDRD4 and DDRD3 must be set to one to enable the SCK and MOSI output as a master[8]. When the SPI system is enabled as a slave, the DDRD2 bit must be set to one to enable the slave serial data output. When the SPI system is enabled as a master, the MISO acts as the master serial data input, regardless of the state of DDRD2. The SPCR configures the SPI. Storing %01011100 into SPCR selects SPI interrupt disable, SPI enable, SPI master, and SCK idle high. Figure 3.2 shows the timing relationship between data and the clock. The master device always places data on the MOSI line a half- cycle before

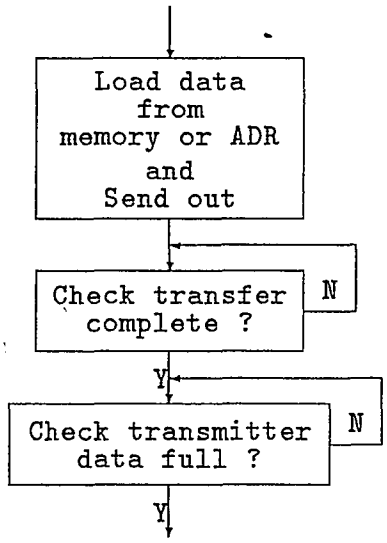


Figure 3.3: Conceptual flowchart segment of data transfer

the positive clock edge (SCK), in order for the slave device to latch the data.

Two status bits must be checked in between each byte of data sending. They are transfer complete flag and the transmitter data full detector. Bit 7 of the SPSR is the SPI transfer complete flag. If it is set upon completion of each byte of data transfer. Unless SPSR is read (with bit 7 set) first, attempts to write to SPDR are inhibited. Bit 0 of PORTA is connected to the SCKREQ of PC side and used as a transmitter data full detector during the data transfer. When a high voltage is detected on this bit, one byte of data is ready to be read on PC data port. Clearing this bit is accomplished by the data reading action on PC. No attempt should be made to write until this bit is cleared, or an overrun condition will exist.

```
.....  ....  ....  ..........................
        LDAB  DATA ; Load data from memory or ADR
```

```
        STAB  SPDR ; Send to SPI
WAIT    LDAA  SPSR ; Check transfer complete flag
        BPL   WAIT ; Jump if clear (incomplete)
WAIT1   LDAA  PORTA ; Check transmitter data full
        ASRA
        BLO   WAIT1 ; Jump if set (full)
......  ....  .....  ........................
```

The data transfer program shown above is the direct translation from the flowchart in Figure 3.3. A byte of data is read from the A/D result register or a memory location (in this example), depending on it is been sending on fly or from buffer, and then sent by a store to SPDR instruction, followed by two statuses checking action. If transfer complete and transmitter data not full, then program continues until the whole frame is sent.

The following program segment, based upon polling, is for command reception.

```
......  ....  .....  ............
POLL    LDAA  PORTA ; Check SCKREQ
        ASRA
        BCS   POLL ; Jump if set (no request)
WAIT2   STAA  SPDR ; Send a dummy byte
        LDAA  SPSR ; Check if read ready
        BPL   WAIT2 ; Jump if clear (not ready)
        LDAA  SPDR ; Load a command byte
        STAA  ,X ; Store to command buffer
```

```
        CMPA   #CR1 ; End of command ?
        BNE    POLL ; Loop back if not finished
......  ....   .....  ........................
```

In this system, since no other task is assigned, the MC68HC11 polls the PA0, the SCKREQ signal from PC, to see whether PC requests the transfer clock. Once a request is detected, MC68HC11 tries to read the command byte from SPDR after it sent out a dummy byte. If this is a CR (with bit 7 set), program goes on to command parsing; otherwise this is an element of a command and should be stored in the command buffer. In future work, the SCKREQ checking should be handled by interrupt, permitting the MC68HC11 to do something else.

3.1.4 PC side

The simplest way to select I/O ports is the fixed address method. This method checks the system address map, then selects one or a group of unused addresses with some proper circuits. The major disadvantage of this method is the selected address might overlap with some other add-on cards address. Figure 3.4 shows a more flexible alternate. Bits SW2 to SW8 of the DIP switch are compared one to one correspondent to PC address A9 to A3, while bit SW1 is compared with AEN. AEN is used to degate the CPU and other devices from the I/O channel to allow DMA transfers to take. When this line is active, the DMA controller has the control of the address bus, the data bus Read command lines (memory and I/O), and the Write command lines (memory and I/O). Since we don't use the DMA in this system, SW1 should be set to 0. When the value on the switch

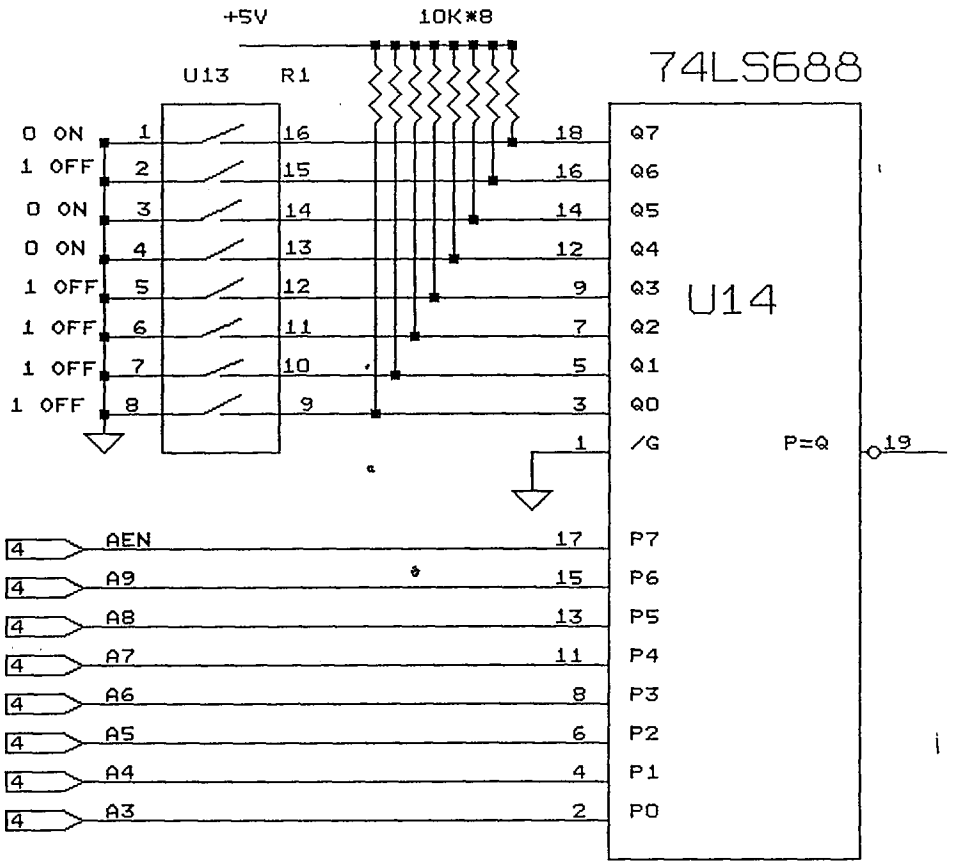


Figure 3.4: Switch selected decoding

equals the P input of U14, the octal comparator, in Figure 3.4, the compare equal output is active. This is the group select control signal, and can be treated as high level decoding to enable U10, the decoder. By changing the combination of the DIP switch, the selected address can be varied in a certain address space.

The default setting, 01001111 at SW1 to SW8, selects 278h as the starting address of the group. The data receiving and command transfer program segments and shown below is based on this setting, so the address of the data port is 278h and that of the status/control port is 27ah. An address refers to two different ports, depending on the executed instruction is input or output. That is, 278h refers to SIPO register for input, or PISO register for output; while 27ah refers to status port for input, or control port for output.

```
start:  mov dx,27ah     ;Select status port
ready?: in  al,dx
        shl al,1        ;Check data ready flag
        jnc ready?      ;Jump if clear (not ready)
        mov dx,278h     ;Select data port
        in  al,dx
        mov ds:[bx],al  ;Store data into memory
        inc bx
        dec cx
        jne start       ;Loop back if not finished
```

In the data transfer, bit 7 of the status port 27ah is a input data ready

indicator. It will be set after system reset or a reading action to the data port. It will be cleared at every byte data transferred. The PC reads one byte of data from port 278h once this bit is checked set, then stores the data at a memory location pointed by ds:[bx]. The cx is a data counter. It was initialized to 31680 by the invoking program.

The following program segment is an example of sending the taking picture with a specified exposure time command.

```
....... .... .......... ......................
        mov  al,81h      ; Load the command code
        call send
        mov  al,[bp+6]   ; Load the information byte
        add  al,30h      ; Convert to ASCII
        call send
        mov  al,8dh      ; CR (with bit 7 set)
        call send
....... .... .......... ...................
send    mov  dx,278h     ;Select data port
        out  dx,al       ;Prepare command to image grabber
        in   dx,al       ;Take previous dummy data away
        mov  dx,27ah     ;Select control port
        mov  al,7fh      ;
        out  dx,al       ;Assert SCKREQ
ready?: in   al,dx
        shl  al,1        ;Check command received flag
        jnc  ready?      ;Jump if clear (not received)
```

```
mov   al,0ffh     ;
out   dx,al       ;Clear SCKREQ
ret
```

Each byte of command is preloaded into al, and then sent out by calling the send subroutine. In command transfer, bit 7 of control port is the SCKREQ. A 0 at this bit informs the image grabber that PC request a command reception cycle. The send subroutine stores the command into PISO register, and resets the bit 7 of the control port. The input data ready indicator, bit 7 of the status port, in data transfer is used as a command received flag. It will be set if the command is received. After the command byte is received, the send subroutine clears the SCKREQ, and returns.

3.2 Image manipulation

In graphics modes, the screen is seen as a matrix of points each capable of displaying one or more colors. Depending on the graphics modes and display adapter, the width and height of the matrix, in pixels, varies. The Hercules Graphics Card is a monochrome graphics adapter capable of displaying graphics on the monochrome monitor. It can display graphics at a resolution of 720×384.

3.2.1 Image display

The Turbo C graphics coordinate system has its origin at the upper left hand corner of the physical screen with the x- axis positive to the right and the y-axis positive going downward. All graphics functions in the library

work with a coordinate frame whose origin is located at the upper left hand corner of the current "viewport," a rectangular region within which all current graphics output appears. A viewport can be defined by the *setviewport* function.

```
clearviewport();
for (j=0; j<165; j++)
{
  for (i=0; i<192; i++)
  {
    level = (*pixel-threshold)/regionwidth;
    switch (level)
    {
      case 5:
      case 4:  putpixel(i*2, j*2+1, max_color);
      case 3:  putpixel(i*2+1, j*2, max_color);
      case 2:  putpixel(i*2+1, j*2+1, max_color);
      case 1:  putpixel(i*2, j*2, max_color);
      case 0:  break;
  }
  pixel++;
}
}
```

Listed above, the core of the display routine is straightforward. The *clearviewport* function clears the current viewport, fills it with the back-

ground color, and resets the current position to the origin (the upper left corner) of the current viewport. The *putpixel* function fills the pixel with a specified color, which is white in this case. Since only black and white are provided by HGC, a halftoning which represents 5 gray levels by counting 1 CCD pixel as 2×2 points is used. Every pixel is positioned by specifying its x and y coordinates, which are `i` and `j` in the program. Corresponding to the Turbo C graphics coordinate system, the origin is located at the upper left corner, and the `i` positive goes to the right and the `j` positive goes to the bottom. The image data from the image grabber are 8-bit per pixel, which means 256 gray level values are possible. However, the distribution of the number of pixel with respect to the intensity is usually not uniform. How to pick up a proper threshold value for image display is a popular subject has been discussing [3]. In this system, the lowest pixel data is selected as the `threshold`, and the one fifth of the difference between the highest and the lowest pixel value is selected as the `regionwidth`. Each of the pixel data is divided by the `regionwidth` for grading. The *switch* instruction in Turbo C compares a expression with the condition value in each *case*, then executes the instructions that the matched case assigns to. Since no *breaks* are put between *cases*, the instructions in later *cases* will be executed continuously after that in matched *case* is completed. Figure 3.5 shows the patterns for each gray level.

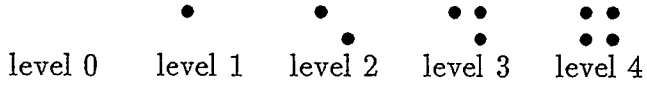


Figure 3.5: The patterns for each gray level

3.2.2 Image editing

The 192×165 bytes of pixel are stored in the memory location given by MS-DOS. They are mapped starting from the upper left corner on the screen to the right, and from the top to the bottom. So the first byte is located at the origin, the 192nd byte is located at the upper right corner, and the 193rd byte at is the first location of the second row.

The image editor program is listed in Appendix C, and the flowchart is shown in Figure 3.6. The Turbo C function *bioskey* scans the keyboard. It returns the IBM keyboard code of the pressed key, or 0 for no pressed key. If the pressed key is not a function key ("E", "P", "0"-"5") or no key is pressed, the program does nothing but blinking the cursor. The "E" key exit the routine, and the "P" key toggles the pen mode. If the pen mode id ON, the routine modifies the pixel pointed by the cursor according to the current gray value. The number key, "0" to "5", changes the current gray value. If the pen mode is ON, it affects the current pointed pixel. The arrow key moves the cursor one step to the direction. The "Home", "End", "PageUp", and "PageDown" are also valid as moving to four corner positions. All operations are followed by blinking the cursor, and then go

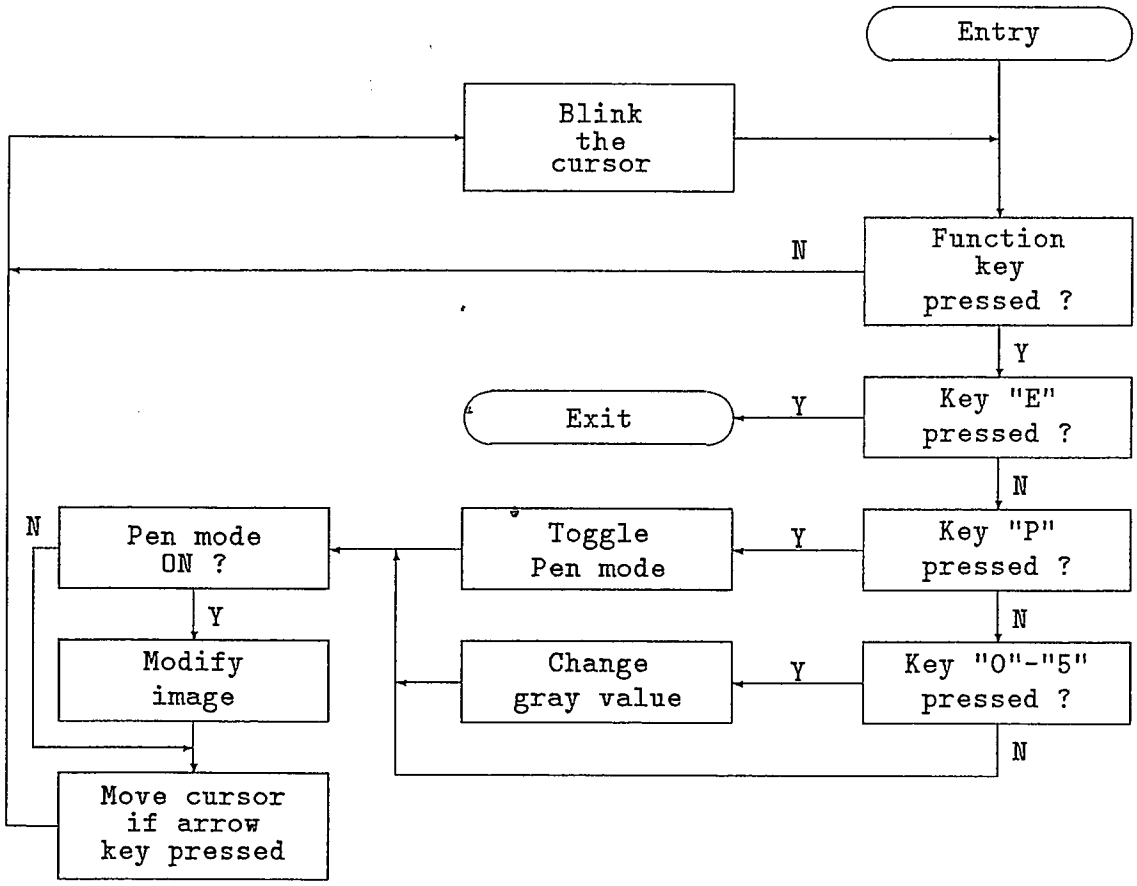


Figure 3.6: Conceptual flowchart of image editor

back to the keyboard scanning.

3.3 Image file preparation

A file format is defined by both structure and content. The simplest structure is binary. It is a direct collection of the gray level value of each pixel. The values are written sequentially.

Another structure available in the system is the TIFF. It was defined jointly by Aldus and Microsoft in conjunction with leading scanner vendors and other interested parties. It is one of the standard image exchange formats. The content of TIFF consists of definitions of individual fields. It is therefore the content that we are ultimately interested in, and the structure merely tells us how to find the fields. In TIFF, individual fields are identified with a unique tag. This allows particular fields to be present or absent from the file as required by the application. Appendix D. give a brief illustration of TIFF structure. See [16] for details.

There are many fields described in the TIFF because it was designed to be powerful and flexible. It takes a fair amount of effort to handle all the options (probably no application does a complete job). In the TIFF file generated by ELECTRIM EDC-1000, the following fields are used: **SubfileType, ImageWidth, ImageLength, BitsPerSample, Compression, PhotometricInterpretation,** and **StripOffsets.** They describe the file contains uncompressed, full resolution image data with the dimension 192 $\times$ 165 pixels, 8 bits per pixel, and value 0 is imaged as black, and value $2^8 - 1$ is imaged as white. Those fields, the image file header, and image

file directory are the first 98 bytes of the file, followed by 31680 bytes of uncompressed pixel data. These also apply to our system. When a TIFF output is requested, the 98 bytes of non-image data will be written to a file, followed by 31680 bytes of gray level image data, to form a 31778 bytes long binary file with the name provided by the user and the extension "TIF".

The Turbo C function *open* opens a file with the given file name and the specified attribute for unbuffered and unformatted I/O operations. The *read* and *write* instructions perform the saving and retrieving a specified number of bytes of data at the current position in the opened, unbuffered and unformatted file.

Chapter 4

System Analysis of Limits and Performance

4.1 Image grabbing

[19] describes the details of the timing, clocking, and operation of TC211 with a television monitors. In this thesis, the chip is operated at much lower frame rate to fit the time consuming A/D conversion in MC68HC11. It takes 32 cycles to convert analog pixel data to digital, *i.e.*16 μs, if 1 cycle equals to 0.5 μs.

4.2 Data transmission

4.2.1 Sending data from buffer

From the viewpoint of hardware, SPI can transfer image data to PC at a bit rate up to 1 Mhz. It takes

$$\frac{8 \times 165 \times 192}{10^6} = 0.25sec$$

to send a full uncompressed frame. However, there are two status bits that have to be checked during the transfer: transfer complete and trans-

mitter data full. The following program segment is a rewrite from the one mentioned in the previous chapter with instruction time in cycles.

```
.....  ......  ....  ....
( 2 )          LDAB  DATA ; Load data from memory or ADR
( 4 )          STAB  SPDR ; Send to SPI
( 4 )  WAIT    LDAA  SPSR ; Check transfer complete flag
( 3 )          BPL   WAIT ; Jump if clear (incomplete)
( 4 )  WAIT1   LDAA  PORTA ; Check transmitter data full
( 2 )          ASRA
( 3 )          BLO   WAIT1 ; Jump if set (full)
.....  ......  ....  .....
```

Assumed all tests are passed at the first check, the segment takes 22 cycles for execution. If 1 cycle equals to 0.5 μsec, a full frame transmission time becomes

$$22\mu sec \times 165 \times 192 \sim 0.35sec.$$

The transmission rate, the reciprocal of the transmission time, will be 2.87 frames per second. This makes it possible for a machine to inspect and recognize sample parts at a rate of higher than 2 items per sec.

PC/XT is running at the speed of about 210 ns per cycle. Using a polling technique, it takes about 70 cycles, 14.7 μsec, to read a byte of data. It is fast enough to handle the image reception.

4.2.2 Sending data on fly

An A/D sequence begins one E clock cycle after a write to the ADCTL (A/D control/status register). It takes 32 cycle to convert analog pixel data to digital. During the 32 cycles, MC68HC11 does nothing but turning the SRG (serial register gate) of the TC211 ON and OFF [4]. Therefore, it will be more efficient to send the digital data on fly. The following program segment is one of the possible ways.

```
.....  ......  ....  ............; ................
( 4 )          STAA  ADCTL         ; Begin A/D 32 cycle
.....          ....  ..............; ................
( 4 )          STAA  PORTA         ; Turn OFF the SRG
( 4 )          LDAA  SPSR          ; Clear bit 7 of SPSR
( 4 )          LDAB  ADRn          ; Get the A/D result
( 4 )          STAB  SPDR          ; Send to SPI
( 4 )          LDAA  PORTA         ;
( 2 )          ANDA  #%10111111    ;
( 4 )          STAA  PORTA         ; Turn ON the SRG
( 2 )          NOP                 ; 4 cycles
( 2 )          NOP                 ; Time delay
( 2 )          ORA   #%01000000    ;
( 4 )          STAA  PORTA         ; Turn OFF the SRG
.....  ......  ....  ............; ................
```

In this example, MC68HC11 takes data from the A/D result register after it turns off the SRG, then sends to SPI. The required time consump-

tion for shifting out the next pixel, *i.e.* the time interval between the two SRGs, or the execution time the two instructions with the "Turn OFF the SRG" comment in the program segment is 28 cycles. Four more cycles consumption should be added to match the A/D conversion time, 32 cycles. Now, the transmission rate is equal to the data generation rate, which is

$$\frac{1}{16\mu sec \times 165 \times 192} \sim 1.97 frames/sec$$

if 1 cycle equals to 0.5 μsec.

4.3 Long distance transmission

The most popular serial interface of computer systems is the EIA RS-232. It supports transmission distance up to 50 feet and it is suitable for a one to one dedicated configuration. The line driver and receiver used in this system, 26LS31 and 26LS32, meet all the requirements of RS-422, which supports long distance transmission up to 1200 meters, and up to 10 fan outs. Figure 4.1 shows the transmission ability.

4.4 Image display

Displaying image on the screen takes host system time. The display program is written in C. Turbo C provides over a hundred graphics routines making the task of creating graphics-based programs much simpler, and dramatically decrease the development time. Turbo C graphics routines perform better than the BIOS video routines and they provide many more

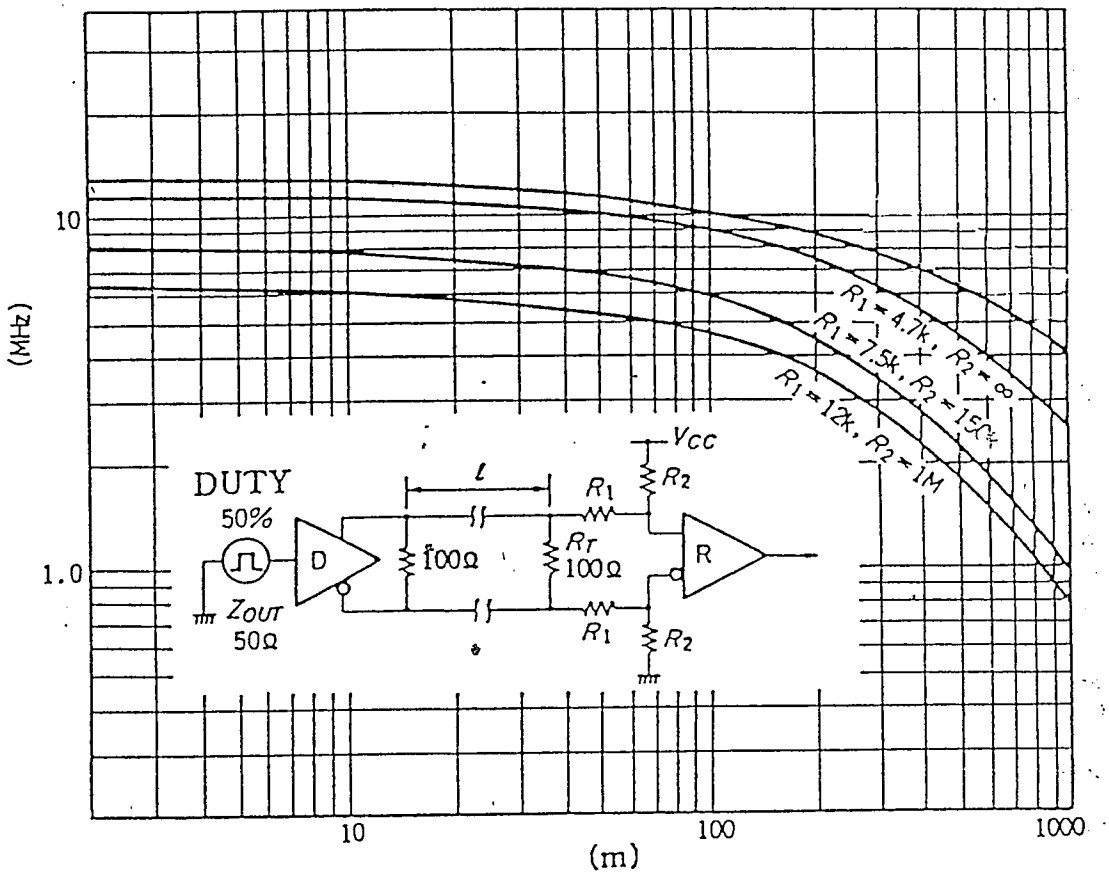


Figure 4.1: Transmission ability of RS-422

basic capabilities than does BIOS interrupt. However, the major disadvantage of a high level language is the execution speed. For very fast screen drawing, directly writing to the video RAM may be preferred.

On the other hand, when writing a program in assembly language, the programmer must take care almost everything by himself. A skilled photographer takes pictures with a manual multi-function camera because a fully automatic camera can not produce many special effects. Similarly, there is a tradeoff between system development time and program execution time.

Image display is not the main point of this thesis, so the author chose to use assembly language only in writing time critical routines, and writing more routines in Turbo C. From the point of view of a real product, much faster image display will be required. Then, writing our own routines in assembly might be necessary.

4.5 Experiments

Similar to the TC210 in Ref[12], TC211 can be operated under constant light, however, smearing is present due to the incident light on the whole chip while the pixel data is transferred out of the CCD. To reduce the smear, reducing the pixel readout time as much as possible is one of the solutions. The recommended clock rate is 7.16 Mhz. The maximum clock rate we have now is

$$\frac{1}{16\mu sec} = 62.5Khz,$$

Figure 4.2: The picture of Feng

so smear problem happens all the time. Figure 4.2 is the best picture we got from the image grabber. It was taken under fairly bright constant light source, with f/16 aperture, 1 meter focus distance, 900 μsec exposure time.

Chapter 5

Conclusions and possible future work

5.1 Conclusion

In this thesis work, a low-cost general purpose machine vision system has been designed and implemented. The system demonstrates the application of a Motorola single-chip microcontroller to implement an intelligent camera system with an interconnected personal computer PC-XT/AT. Synchronous digital data transmission using a line driver and receiver provide the capability to separate the camera and computer by a distance. The SPI of Motorola 68HC11 also provides the possibility to configure a single host computer with multiple cameras.

To display the image on the screen of a personal computer, a variety of display adapters are studied. A halftoning algorithm is used to represent a gray-level picture on a two level (ON and OFF) screen. The image data can be stored in a binary file, which can be easily retrieved by custom image display software, or a TIFF file, which is compatible with many desk-top

publishing software packages.

5.2 Suggestions for future work

There are many approaches to upgrade the system performance. They are listed below:

The polling technique is used for the SPI. It can be modified to be interrupt driven, so the image transfer can go in background.

Some digital signal process chips such as delta modulator or other data compression chips can be used to increase the capabilities on the camera side. This may reduce considerably the amount of data transmission between camera and the host computer to represent a frame.

There are many processes which could be added on the PC side. For example, the product inspection an edge detecting function would be desirable. In security applications, motion detection would be useful.

The only command executed by the camera is changing the exposure time now. Other commands could be defined in the future. For instance, adding some I/O ports at the image grabber side to control a motor to aim the target, trace the picture or to focus.

A RGB (Red-Blue-Green) wheel can be installed in front of the lens. Combining the three pictures taken through the RBG wheel, we can get a

color picture without a color CCD chip as long as the image is static for three exposure times.

Appendix A

The User's Guide

A.1 System description

NMVS (NJIT Machine Vision System) is a black and white, computer controlled, machine vision system. It consists of a PC add-on card and a micro-based image grabber. It supports long distance camera-computer configuration. It is a compact and low-cost alternate for the acquisition of digital image data and inspection.

A.2 Features

CCD (Charge Coupled Device) detector

Low light level capability

Free I/O port addressing

Long distance transmission

Light weight

Long life

A.3 Specifications

image sensing area	2.64 mm × 2.64 mm
pixel size	13.75 × 16 microns
pixel array (resolution)	192(H) × 165(V)
exposure time	100 μs to 999 μs
image transmission rate	up to 2.87 frames per sec
transmission distance	up to 1000 meters
operating temperature	32 - 110 $F°$
power requirements	+5V, +12V, -12V

A.4 Compatibility

The NMVS requires an IBM PC/XT/AT or compatible computer with a free expansion slot in which to install a full size card. The NMVS software requires a minimum of a double density 5.25 in floppy disc driver, 512 K RAM, and PC/MS-DOS version 3.1 or higher. The Hercules video adapter is supported for display of images.

A.5 Installation

The NMVS consists of a CCD camera, an image grabber board, a PC add-on card, a 9-pin and a 15-pin connecting cables, and a 5.25 in diskette containing the system software.

A.5.1 Port address selection

In most installations there will be no conflict between the default settings of the NMVS port addresses and the port addresses of other peripheral device.

For the unusual situation in which a port address conflict exists, the NMVS provides a DIP switch for selecting alternative sets of port addresses. The SW1 of the DIP switch should be always set to ON. Switches 2-8 select one of 128 possible contiguous 8 byte port address ranges to which the NMVS interface card responds. The location of the first port address in the range is determined as follows:

Switch	hex value when "off"	hex value when "on"
8	8	0
7	10	0
6	20	0
5	40	0
4	80	0
3	100	0
2	200	0

The "hex values" of the switches are additive. For instance, the default setting of the DIP switches is as shown below:

Switch	position	hex value
8	off	8
7	off	10
6	off	20
5	off	40
4	on	0
3	on	0
2	off	200

Thus the port address locations to which the NMVS add-on card responds (at the default switch settings) are 278 and 27a.

A.5.2 Add-on card insertion

Turn off the computer and any devices connected to it, before you do anything. Open the cover of the computer, referring to the instructions that came with the computer if necessary. Select an unused expansion slot in

which to install the card. Insert the edge connector into the socket of the expansion slot, making sure that the card is correctly seated in the slot.

A.5.3 Cable connection:

The NMVS use two cables to connect three major parts of the system. The camera and the image grabber are connected together by a 9-pin short cable. The connection between the image grabber and the PC add-on card is a 15-pin long cable. Hook up these two cables with the computer power turned off.

A.5.4 Software setup

The NMVS software is supplied for use only with the NMVS. The software is not copy protected. It is recommended that a backup copy be made of the NMVS software and the original diskette be stored in a safe place. Make sure all of files are in the same directory.

A.5.5 Image grabber configuration

This part is for the prototype image grabber only. The prototype image grabber consists of a MC68HC11 EVB and a wire wrapped board. The two boards are connected with a 60- pin multicolored flat cable. Download and run the program IG according to the EVB manual.

A.6 Menu software

The system is operated under the software NMVS (NJIT Machine Vision System). It is an execution file named NMVS.EXE, created by linking several programs written in Turbo C and Intel 8088 assembly language.

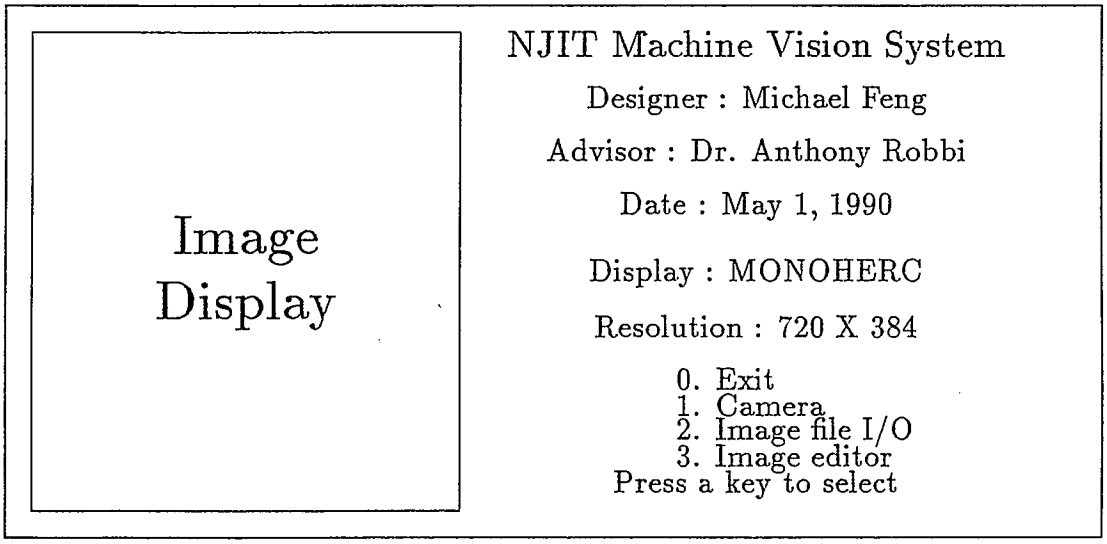


Figure A.1: The main menu of NMVS

When the user types in NMVS, a main menu screen will be shown. See Figure A.1. The left half side of the screen is the image display area, while the right half side is the selection menu. The user can give commands to the system by making selections from the menu instead of typing the commands. Almost every command that the user has to type is reduced to one key selection on screen.

A.6.1 Main menu

The choices from the "main" menu and their actions are as follows:

0. Exit	exits the NMVS software
1. Camera	enters the "camera" menu
2. Image file I/O	enters the "image file I/O" menu
3. Image editor	enters the "image editor" menu

A.6.2 Camera menu

The choices from the "camera" menu and their actions are as follows:

0. Exit	exits the "camera" menu
1. Set exposure time	sets the exposure time of the camera
2. Image grabber	begins image grabbing and display

The acceptable exposure time varies from 100 μs to 900 μs. It can be set in the "Set exposure time" entry by typing in the number. The camera starts to take a picture with the given exposure time and transfers to the PC side after the image grabber entry is selected.

A.6.3 Image file I/O menu

The choices from the "image file I/O" menu and their actions are as follows:

0. Exit	exits the "image file I/O" software
1. Select file format	enters the "image file format menu"
2. Save image to file	enters the "save image file menu"
3. Load image from file	enters the "load image file menu"

In the "Select file format" entry, key "T" is used to toggle the file format between binary and TIFF. A "BIN" or a "TIF", the default file name extensions, will be displayed as a indicator. The "Save image to file" menu prompts the user a file name for later current image saving. If the file name is given without a extension, the default extension will be used. The "Load image from file" menu does the same thing as the "save image to file" does, except it is for loading.

A.6.4 Image editor

The choices from the "Image editor" menu and their actions are as follows:

E. Exit	exits the "image editor" software
P. Pen mode toggle	toggles the Pen mode
Arrow : Move cursor	moves the cusor
0 - 5 : Gray level	changes gray level value

A.7 Operation

Having installed the NMVS hardware, according to the proceeding instructions, place the diskette with the copy of the NMVS software in the disc driver. Make the directory containing the NMVS the current directory, and enter "nmvs" to start.

A.7.1 Getting a picture

With moderate indoor illumination, typical of most homes or offices, set the camera lens to an aperture of f/16. Set the focus adjustment of the camera lens to a distance of about 1 meter. Point the camera toward a still object a few feet from the lens.

Select "image grabber" entry under the "camera" menu, a rectangular image frame will be displayed at the left side of the screen. Slowly adjust the focus of the lens for the best picture.

A.7.2 Setting exposure time

To allow a large aperture without detector overload it is desirable to reduce the room illumination or to reduce the exposure time. The acceptable exposure time varies from 100 μs to 999 μs. It can be set after the user select the "set exposure time" entry.

A.7.3 Image editing

The image editor program is provided to modify the current displayed image. A blinking cursor is displayed on the screen. Use the arrow key to move the cursor in four directions. The "Home", "End", "PageUp", "PageDown" are also valid as popping the cursor to the four corner positions. Pen mode is toggled by the "P" key. When the pen mode is OFF, no track will be left when the cursor is moved. If the pen mode is ON, the pixel pointed by the cursor will be modified according to the current gray value. A particular symbol or character can be made by moving the cursor when the pen mode is ON. The number key "0" to "5" set the current gray vale. The gray level 0 is the darkest, and the gray level 5 is the brightest, related to the current displayed image.

A.7.4 Saving/loading a image

Another way to get a picture displayed is loading the picture from a file. This can be done in the "load image from file" entry under the "image file I/O" menu. The file can be binary or TIFF, but it has to be generated by the NMVS before. The user will be prompted a file name. Any file name acceptable in DOS is acceptable in the NMVS. If no file name extension

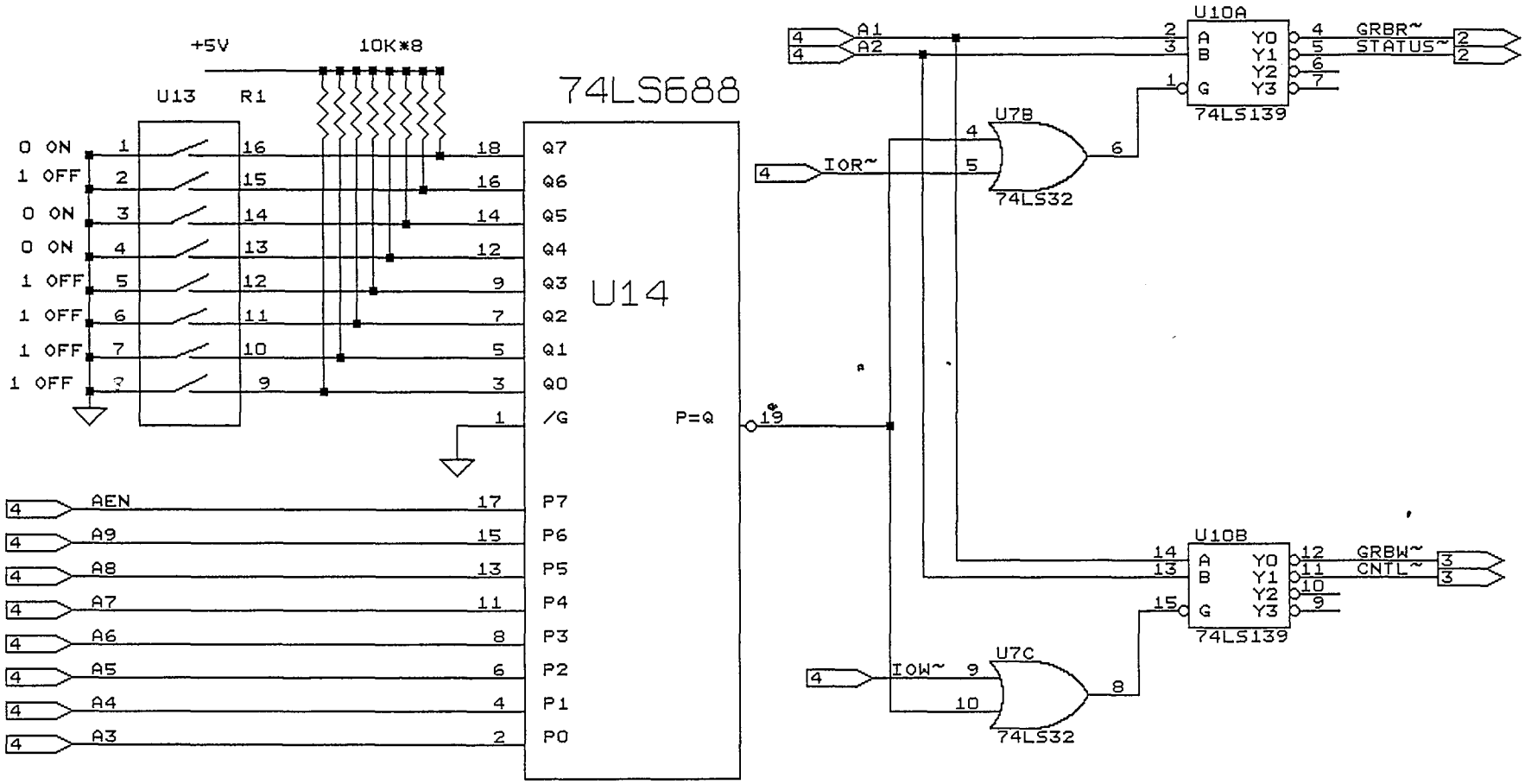
is given, a default extension, according to the file format selected, will be added. The current picture can be store into a file with the similar procedure in the "save image to file" entry.

A.8 Printing Images

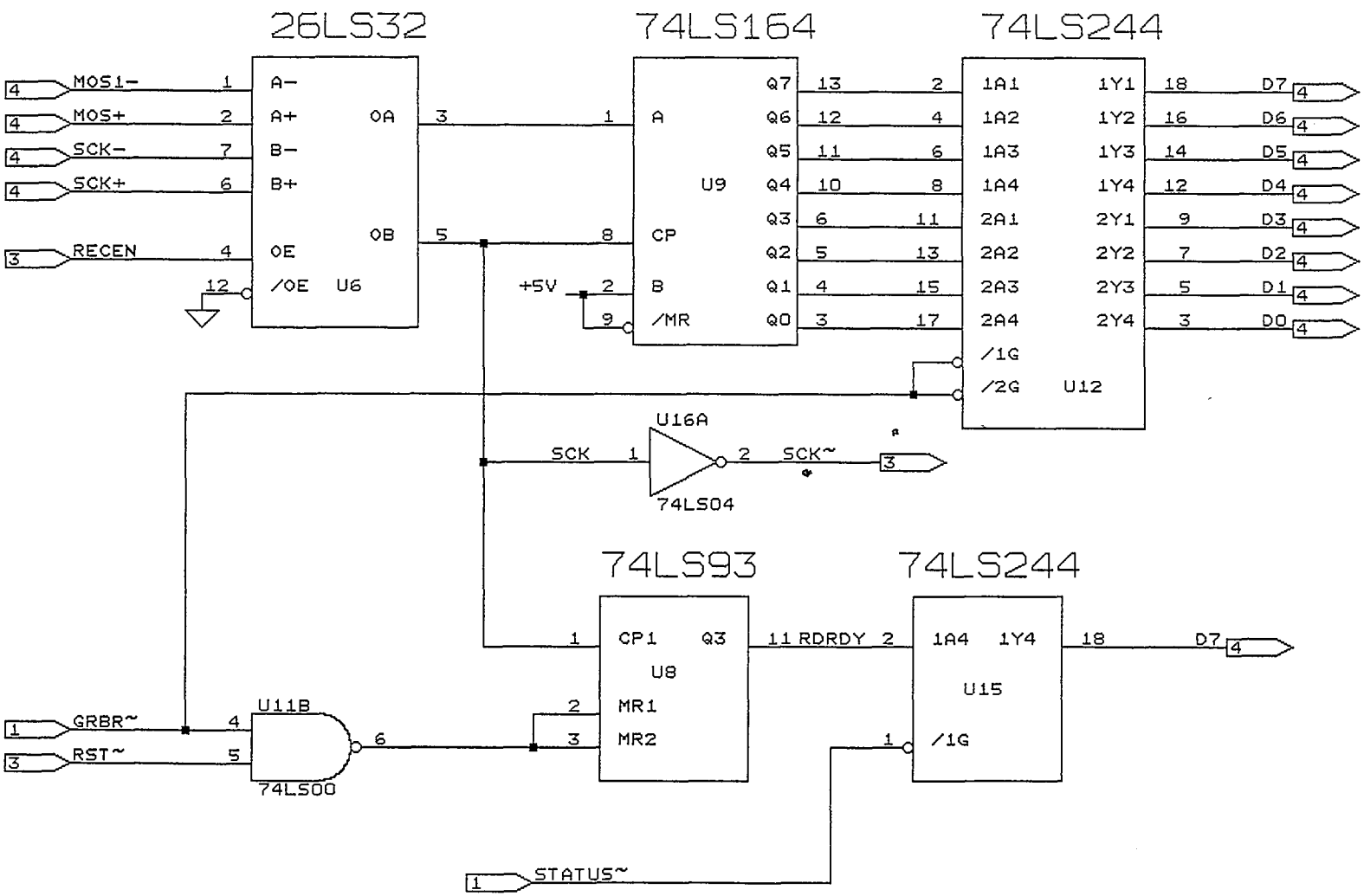
Printing of images directly from the NMVS is presently not supported; however, NMVS pictures may be converted to "TIFF" files and printed using appropriate commercial software packages, such as PageMaker and Microsoft Word.

Appendix B

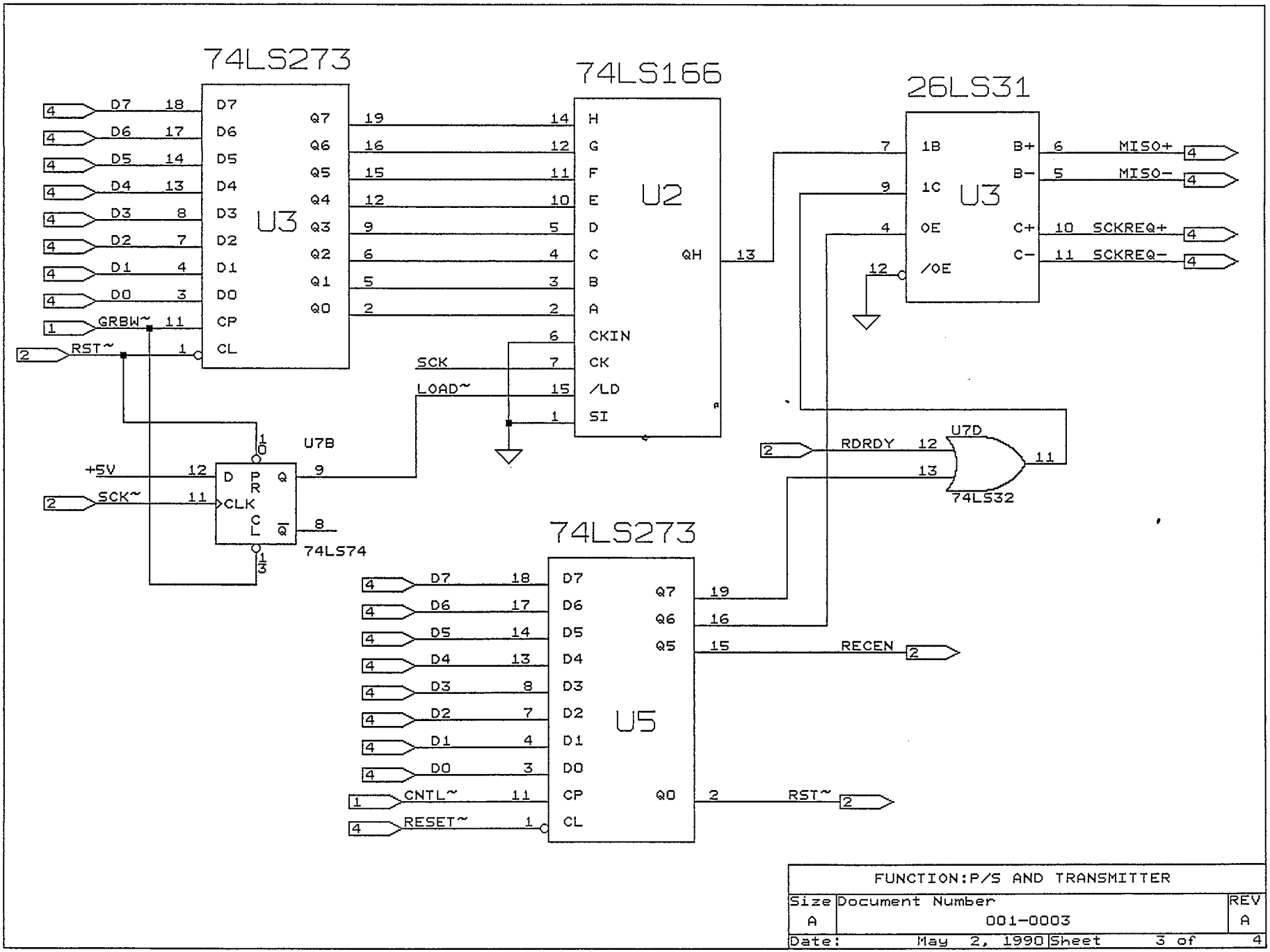
Circuit Diagrams

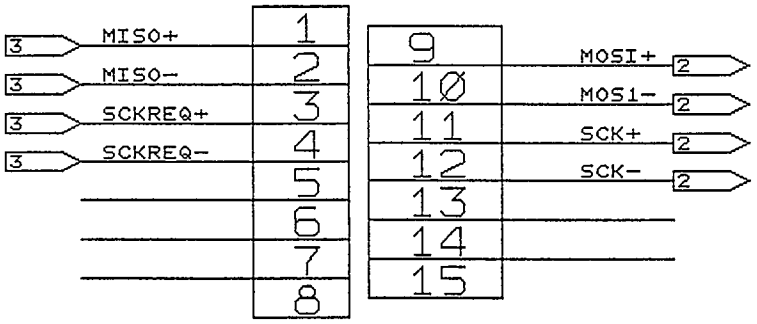


FUNCTION: GROUP SELECT AND DECODING		
Size A	Document Number 001-0001	REV A
Date: May 10, 1990	Sheet 1 of 4	

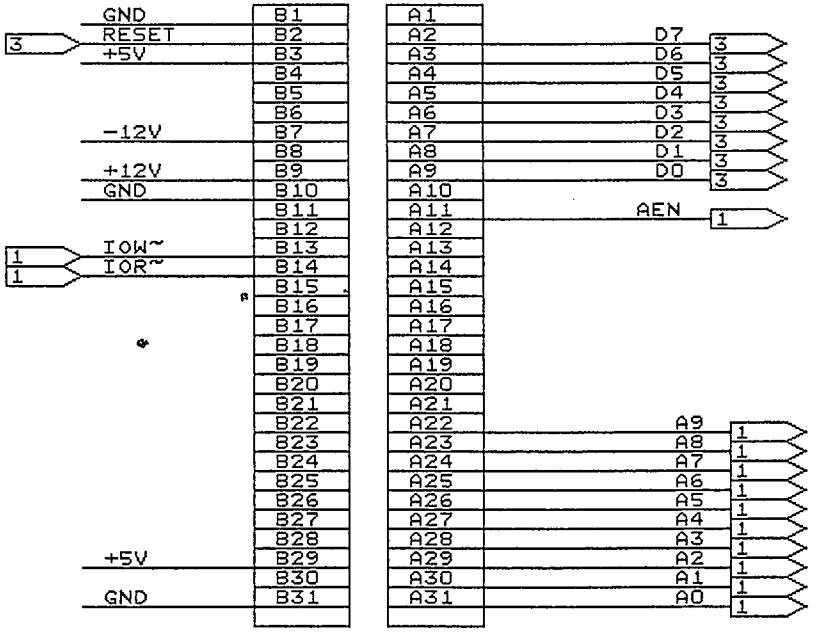


FUNTION: RECEIVE AND S/P		
Size A	Document Number 001-0002	REV A
Date: April 30, 1990	Sheet 2 of 4	


74LS273
U3
D7 18
D6 17
D5 14
D4 13
D3 8
D2 7
D1 4
D0 3
GRBW~ 11
RST~ 1
Q7 19
Q6 16
Q5 15
Q4 12
Q3 9
Q2 6
Q1 5
Q0 2
74LS166
U2
H 14
G 12
F 11
E 10
D 5
C 4
B 3
A 2
CKIN 6
CK 7
/LD 15
SI 1
QH 13
SCK
LOAD~
26LS31
U3
1B 7
1C 9
OE 4
/OE 12
B+ 6 MISO+
B- 5 MISO-
C+ 10 SCKREQ+
C- 11 SCKREQ-
U7B
+5V 12
SCK~ 11
D P R Q
CLK
C L Q
9
8
74LS74
U7D
RDRDY 12
13
11
74LS32
74LS273
U5
CNTL~ 11
RESET~ 1
Q7 19
Q6 16
Q5 15
Q0 2
RECEN
RST~
FUNCTION:P/S AND TRANSMITTER
Size A
Document Number 001-0003
REV A
Date: May 2, 1990
Sheet 3 of 4

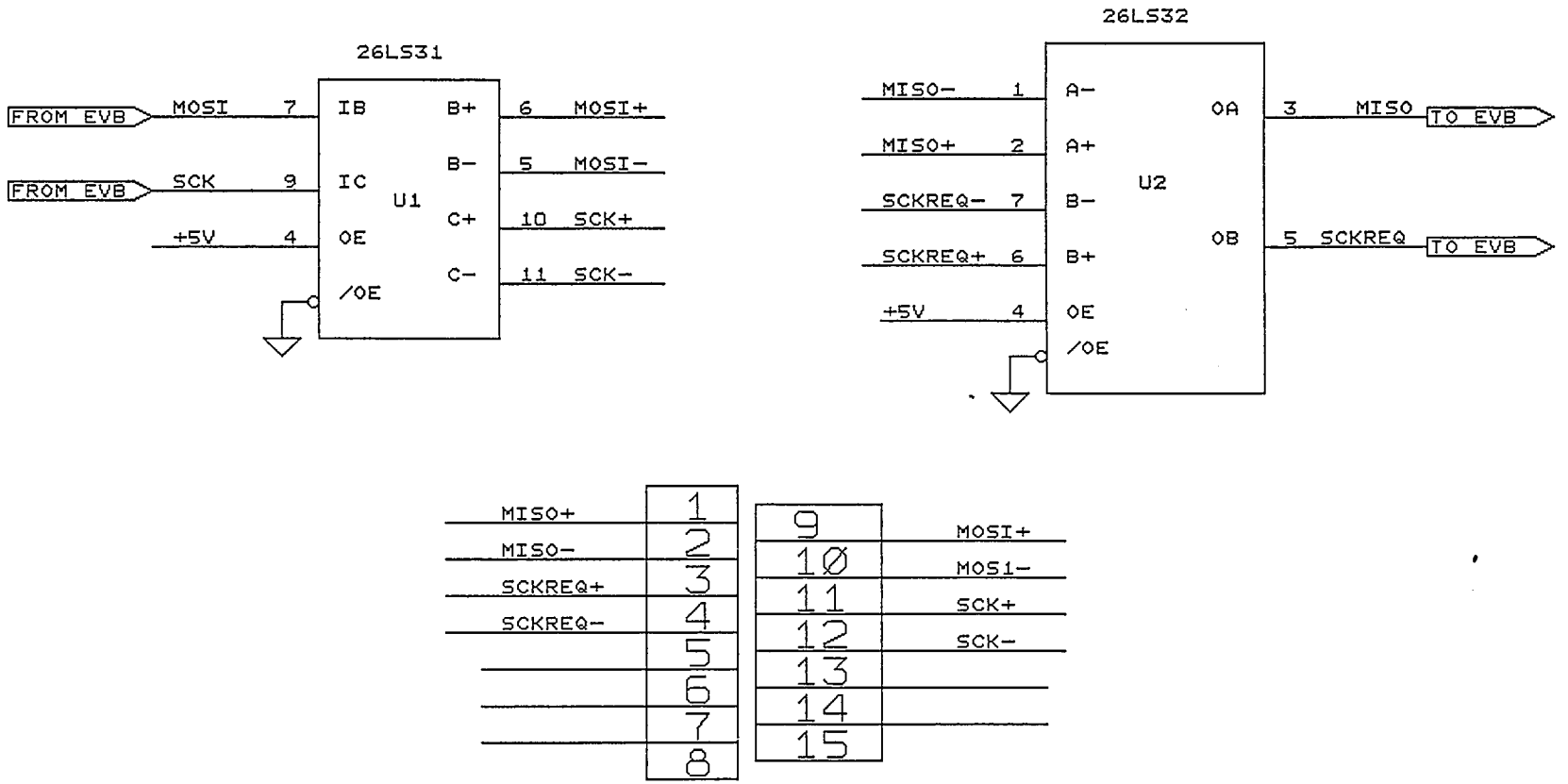


J1



J2

FUNCTION: CONNECTORS		
Size: A	Document Number: 001-0004	REV: A
Date: May 10, 1990	Sheet 4 of 4	



FUNCTION:LINE TRANSCEIVERS OF IG		
Size A	Document Number 002-0001	REV A
Date: May 11, 1990	Sheet 1 of 1	

Part List of the NMVS

U1, 5	74LS273	2	8-bit register with clear
U2	74LS166	1	8-bit shift registers
U3	26LS31	1	RS-422 line driver
U4	74LS74	1	Dual D-type flip-flop
U6	26LS32	1	RS-422 line receiver
U7	74LS32	1	Quad 2-input OR gate
U8	74LS93	1	Decade counter 4-bit counter
U9	74LS164	1	Serial-in parallel-out register
U10	74LS139	1	Dual 1-of-4-decoder
U11	74LS00	1	Quad 2-input NAND gate
U12, 15	74LS244	2	Octal 3-state buffer/line driver
U14	74LS688	1	8-bit magnitude comparators
U15	74LS04	1	Hex inverter

Appendix C

Source Program Listing

```
                                ;*************************************************
                                ;   Taking picture with the specified exposure time *
                                ; Copyright 1990 New Jersey Institute of Technology *
                                ; Programmer: Michael Feng   Date: May 1, 1990      *
                                ;*************************************************
                                        public  _exposure
0000                    _text   SEGMENT para public 'code'
                                        ASSUME  CS:_text,DS:_text
0000                    _exposure       proc    far
0000  55                                push    bp
0001  8B EC                             mov     bp,sp
0003  1E                                push    ds
0004  06                                push    es
0005  57                                push    di
0006  56                                push    si
0007  BA 027A                           mov     dx,27ah
000A  B0 FF                             mov     al,0ffh
000C  EE                                out     dx,al
000D  B0 81                             mov     al,81h
000F  E8 001C                           call    send
0012  8B 46 06                          mov     ax,[bp+6]
0015  04 30                             add  al,30h
0017  E8 0014                        call       send
001A  B0 8B                             mov     al,8bh
001C  E8 000F                           call    send
001F  BA 027A                    mov  dx,27ah
0022  B0 7F                      mov  al,7fh
0024  EE                         out  dx,al
0025  B8 0000                          mov     ax,0
0028  5E                              pop     si
0029  5F                              pop     di
002A  07                              pop     es
002B  1F                              pop     ds
002C  5D                              pop     bp
002D  CB                              ret
002E                    _exposure       endp
```

```
                                ;*************************************************
                                ;           Image receiption subroutine         *
                                ; Copyright 1990 New Jersey Institute of Technology *
                                ; Programmer: Michael Feng   Date: May 1, 1990   *
                                ;*************************************************
                                        public  _fill1
0000                            _text   SEGMENT para public 'code'
                                        ASSUME  CS:_text,DS:_text
0000                            _fill1  proc    far
0000  55                                push    bp
0001  8B EC                             mov     bp,sp
0003  1E                                push    ds
0004  06                                push    es
0005  57                                push    di
0006  56                                push    si
0007  8B 46 06                          mov     ax,[bp+6]
000A  8B D8                             mov     bx,ax
000C  8B 46 08                          mov     ax,[bp+8]
000F  8B C8                             mov     cx,ax
0011  BA 027A                           mov     dx,27ah
0014  B0 7F                             mov     al,7fh
0016  EE                                out     dx,al
0017  BA 0278                           mov     dx,278h
001A  EC                                in      al,dx
001B  FA                                cli
001C  BA 027A                   start:  mov     dx,27ah
001F  EC                        ready?: in      al,dx
0020  D0 E0                             shl     al,1
0022  73 FB                             jnc     ready?
0024  BA 0278                           mov     dx,278h
0027  EC                                in      al,dx
0028  88 07                             mov     ds:[bx],al
002A  43                                inc     bx
002B  49                                dec     cx
002C  75 EE                             jne     start
002E  B0 FF                             mov  al,0ffh
```

```
/*****************************************************************/
/*                      Image file I/O routine                   */
/*    Copyright 1990 New Jersey Institute of Technology Ver 1.0   */
/*    This routine save/load image file to/from disc              */
/*****************************************************************/
#include <errno.h>
#include <graphics.h>
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
#include <string.h>
#include <sys\stat.h>
#include <sys\types.h>
int extern x, y, x1, y1, x2, height_of_text;
unsigned char extern *buffer, tif_header[98];
char extern file_ext[5];
char extern filetype[];
char extern *filename_ext;
char filename[12];
void display();
void ggets(char *s);
char select(int select_group);
fileio()
{
 int y2;
 int bytes, filehandle;
 char selection, ans, header[98];

 y2 = height_of_text + 20;
 setviewport((x-x2)/2+x1, y-y2, (x+x2)/2+x1, y, 0);
 clearviewport();
 outtext("File Type: "); outtext(filetype);
 while ((selection = select(2)) != '0')
 {
  switch (selection)
  {
   case '1': outtext("Press T to toggle,");
          moveto(0, y2); outtext("or other key to exit.");
          ans = getch();
```

```
/*****************************************************************/
/*                  Get character string subroutine              */
/*    Copyright 1990 New Jersey Institute of Technology Ver 1.0   */
/*    This subroutine is a counterpart of gets function in text   */
/*    text mode. It returns the address of the character string   */
/*    entered from keyborad.                                      */
/*****************************************************************/
#include <stdio.h>
#include <ctype.h>
#include <graphics.h>
#define BS  0x08
#define CR  0x0d
#define DEL 0x7f

ggets(char *s)
{
 struct viewporttype
 {
  int left, top;
  int right, bottom;
  int clip;
 };
 struct viewporttype cur_view;
 char c;
 char temp[] = " \0";
 int count, i, x, y;
 count = 0;
 getviewsettings(&cur_view);
 while ((c = getch()) != CR)
 {
   if (isprint(c))
   {
     temp[0] = c;
     outtext(temp);
   }
   if (c ==  DEL || c == BS)
   {
     if (count)
     {
```

```
/*****************************************************************/
/*                    Image editor subroutine                    */
/*    Copyright 1990 New Jersey Institute of Technology Ver 1.0   */
/*    This routine modifies the current displayed image           */
/*****************************************************************/
#include <bios.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#define KEYBRD_READY 1
#define KEYBRD_READ 0
#define KEY_E 0x1265
#define KEY_e 0x1245
#define KEY_P 0x1950
#define KEY_p 0x1970
#define KEY_0 0x0b30
#define KEY_1 0x0231
#define KEY_2 0x0332
#define KEY_3 0x0433
#define KEY_4 0x0534
#define UP 0x48
#define DOWN 0x50
#define RIGHT 0x4d
#define LEFT 0x4b
#define HOME 0x47
#define END 0x4f
#define PGUP 0x49
#define PGDN 0x51

int extern height_of_text, maxx, maxy, x, y, x0, y0, x1, y1;
unsigned char extern *buffer;
unsigned char extern regionwidth, lowest, highest;
ieditor()
{
 struct viewporttype
 {
  int left, top;
  int right, bottom;
  int clip;
```

```
/*****************************************************************/
/*                        Select subroutine                      */
/*   Copyright 1990 New Jersey Institute of Technology Ver 1.0   */
/*   This subroution displays the selection informations,        */
/*   prompts the user, and returns the select. Selection         */
/*   informations are grouped by select_group.                   */
/*   1: main menu, 2: camera menu, 3: image file I/O menu        */
/*****************************************************************/

#include <ctype.h>
#include <graphics.h>
#include <stdio.h>
#include <string.h>
int extern x, y, x1, y1, height_of_text;
char select(int select_group)
{
 int i, x2, y2;
 char ch;
 char prompt[] = "Press a key to select";
 static char *fun_name[3][4]
                = {
                {"0. Exit           ",
                 "1. Camera         ",
                 "2. Image file I/O ",
                 "3. Image editor   "
                },
                {
                 "0. Exit             ",
                 "1. Set exposure time",
                 "2. Image grabbing   ",
                 "                    ",
                },
                {
                 "0. Exit              ",
                 "1. Select file format",
                 "2. Save current image",
                 "3. Load image file   ",
                }
               };
```

```
/*************************************************************/
/*    NJIT Machine Vision System Main Program  Ver 1.0       */
/*    Copyright 1990 New Jersey Institute of Technology      */
/*    Programmer: Michael Feng          Date: May 1, 1990    */
/*************************************************************/
#include <stdio.h>
#include <graphics.h>
#include <alloc.h>
#include <string.h>
int maxx, maxy, x, y, x0, y0, x1, y1, x2, y2, height_of_text;
int time = 900;
char file_ext[5] = ".BIN";
char filetype[] = "BIN";
char *filename_ext = "Enter file name [.BIN]: ";
unsigned char tif_header[98] = {'I', 'I', '*', 0, 8, 0, 0, 0, 7, 0, 0xff,
                    0, 3, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 3,
                    0, 1, 0, 0, 0, 0xc0, 0, 0, 0, 1, 1, 3, 0,
                    1, 0, 0, 0, 0xa5, 0, 0, 0, 2, 1, 3, 0, 1,
                    0, 0, 0, 8, 0, 0, 0, 3, 1, 3, 0, 1, 0, 0,
                    0, 1, 0, 0, 0, 6, 1, 3, 0, 1, 0, 0, 0, 1,
                    0, 0, 0, 0x11, 1, 3, 0, 1, 0, 0, 0, 0x62,
                    0, 0, 0, 0, 0, 0, 0};
unsigned char *buffer;
char *time_string = "900";
char select(int select_group); /* defined in select.c */
void camera(); /* defined in camera.c */
void fileio(); /* defined in fileio.c */
void ieditor(); /* defined in ieditor.c */
void ggets(char *s); /* defined in gget.c */
main()
{
 const length = 165*2, width = 192*2;
 int graphdriver = DETECT, graphmode, success;
 char filename[81];
 char title[] = "NJIT Machine Vision System";
 char advisor[] = "Advisor: Dr. Anthony Robbi";
 char designer[] = "Designer: Michael Feng";
 char date[] = "May 1, 1990";
 char display[20] = "Display: ";
```

```
/*****************************************************************/
/*                 Image display subroutine                      */
/*    Copyright 1990 New Jersey Institute of Technology Ver 1.0  */
/*    This subroutine display the image in 5 gray level          */
/*****************************************************************/
#include <graphics.h>
int extern maxx, maxy, x0, y0, x1, y1;
unsigned char extern *buffer;
unsigned char regionwidth, lowest = 126, highest = 131;
display()
{
 struct viewporttype
 {
  int left, top;
  int right, bottom;
  int clip;
 };
 int i, j, level, max_color;
 struct viewporttype cur_view;
 unsigned char *pixel;
 lowest = 126; highest = 131;
 getviewsettings(&cur_view);
 setviewport(x0, y0, x1, y1, 1);
 clearviewport();
 max_color = getmaxcolor();
 pixel = buffer;
 for (i=0; i<31680; i++)
 {
  if (*pixel < lowest) lowest = *pixel;
  if (*pixel > highest) highest = *pixel;
  pixel++;
 }
 regionwidth = (highest - lowest)*0.2;
 rectangle(0, 0, 192*2, 165*2);
 pixel = buffer;
 for (j=0; j<165; j++)
 {
  for (i=0; i<192; i++)
  {
```

```
/*****************************************************************/
/*                    Camera subroutine                          */
/*    Copyright 1990 New Jersey Institute of Technology Ver 1.0  */
/*    This subroutine sends command to the image grabber and     */
/*    receives the image frame                                   */
/*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
int extern x, y, x1, y1, x2, height_of_text, time;
char extern *time_string;
unsigned char extern *buffer;
char extern ggets(char *s);
int extern fill1(unsigned char *a, int count);
int extern exposure(int time);
void extern display();
char extern select(int select_group);
camera()
{
 int a, i, y2;
 char exposure_time[] = "Exposure Time :";
 char ms[] = "ms";
 char selection;

 y2 = height_of_text + 20;
 setviewport((x-x2)/2+x1, y-y2, (x+x2)/2+x1, y, 0);
 clearviewport();
 outtext(exposure_time); outtext(time_string); outtext(ms);
 while ((selection = select(1)) != '0')
 {
   switch(selection)
   {
    case '1': setviewport(x1+20, y, x+x1-20, y1, 0);
           outtext("Type in the exposure time (ms)");
           moveto(0,y2); ggets(time_string);
           /* if (isdigit(*time_string)) */
           time = atoi(time_string);
           itoa(time, time_string, 10);
           clearviewport();
```

Appendix D

Brief Illustration of TIFF

D.1 TIFF structure

A TIFF file begins with an 8-byte "image file header" that points to one or more "image file directories." The directories contains informations about the image, as well as pointer to the actual image data. Figure D.1 shows the structure of the TIFF file. The address of each item is shown at left side of the item, and the description is shown at right. A sample of TIFF G (gray-leveled) image is appended at the end. For more informations, see Ref [16].

D.1.1 Image file header

The 8-byte image file header contains the following information:

Bytes 0-1 : specify the byte order used within the file.

"II" (Hex 4949) - from least significant to most significant.

"MM" (Hex 4D4D) - from most significant to least significant.

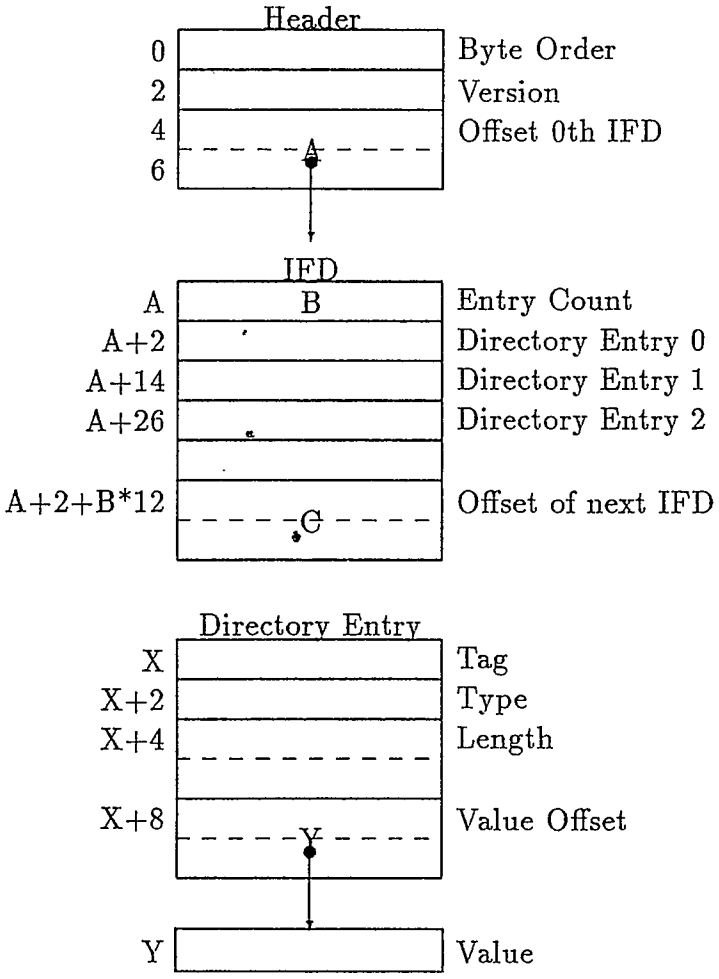


Figure D.1: The structure of TIFF

Bytes 2-3 : version number 42 (Hex 002A).

A TIFF file does not have a real version/revision number. The number 42 was chosen for its deep philosophical significance. It can and should be used as additional verification that this is indeed a TIFF file.

Bytes 4-7 : offset of the first IFD. The directory may be at any location in the file after the header. In particular, an image file directory may follow the image data it describes. (The term "offset" is always with respect to the beginning of the file.)

D.1.2 Image file directory

IFD consists of a 2-byte count of the number of entries, followed by a sequence of 12-byte field entries, followed by a 4-byte offset of the next IFD or 0 if none. Each 12-byte IFD entry has the following format:

Bytes 0-1 : tag for the field.

Bytes 2-3 : field type (3=SHORT 2 bytes, 4 LONG 4 bytes)

Bytes 4-7 : length of the field

Bytes 8-11: file offset of the value for the field

In order to save time and space, the value offset is interpreted to contain the value instead of pointing to the value if the value fits into 4 bytes. If the value is less than 4 bytes, it is left-justed within the 4-byte value offset, *i.e.* stored in the lower-number bytes. The entries in an IFD must be stored in ascending order by Tag. The values to which directory entries point need not be in any particular order in the file.

D.2 The required field

There are many fields described in the TIFF spec. Because we are interested in grayscale images, only the requirements for class G (grayscale) are discussed here.

SubfileType

Tag = 255 (FFh)

Type = SHORT

N = 1

A general indication of the kind of data that is contained in this subfile.

1 = full resolution image data.

ImageWidth

Tag = 256 (100h)

Type = SHORT or LONG

N = 1

The image's width in pixel (X : horizontal).

ImageLength

Tag = 257 (101h)

Type = SHORT or LONG

N = 1

The image's length (height) in pixels (Y : vertical).

StripOffsets

Tag = 273 (111h)

Type = SHORT or LONG

N = number of strips

Offset of the stripoffsets, or offset of the strip if N=1.

BitsPerSample

Tag = 258 (102h)

Type = SHORT

Number of bits per sample.

Compression

Tag = 259 (103h)

Type = SHORT

N = 1

1 : No compression.

PhotometricInterpretation

Tag = 262 (106h)

Type = SHORT

N = 1

1 : For grayscale images : 0 is imaged as black. $2^{\mathbf{BitPerSample}-1}$ is imaged as white.

D.3 A TIFF G image example

The materials listed below are the first 112 bytes dumped out from a TIFF file.

```
0000 49 49 2A 00 08 00 00 00 07 00 FF 00 03 00 01 00
0010 00 00 01 00 00 00 00 01 03 00 01 00 00 00 C0 00
0020 00 00 01 01 03 00 01 00 00 00 A5 00 00 00 02 01
0030 03 00 01 00 00 00 08 00 00 00 03 01 03 00 01 00
0040 00 00 01 00 00 00 06 01 03 00 01 00 00 00 01 00
0050 00 00 11 01 03 00 01 00 00 00 62 00 00 00 00 00
0062 00 00 78 74 74 77 79 76 76 76 77 76 77 77 76 77
```

Reorganizing these materials to be more intuitive, we get the followings:

Offset	Name	Value
Header:		
0000	Byte Order	4949
0002	Version	002A
0004	1st IFD pointer	00000008
IFD:		
0008	Entry Count	0007
000A	SubfileType	00FF 0003 00000001 00000001
0016	ImageWidth	0100 0003 00000001 000000C0
0022	ImageLength	0101 0003 00000001 000000A5
002E	BitsPerSample	0102 0003 00000001 00000008
003A	Compression	0103 0003 00000001 00000001

0046	PhotometricInterpretation	0106 0003 00000001 00000001
0052	StripOffsets	0111 0003 00000001 00000062

Image Data

0062	Uncompressed data	78 74 74 77 79 76 76 77 76 77
....		.. ..

Appendix E

A Low-cost Machine Vision System with Intelligent Camera

E.1 Abstract

In this paper, a low-cost general purpose machine vision system based on a personal computer of PC-XT/AT family and an electronic camera controller based-upon a Motorola 68HC11 single-chip microcontroller have been designed and implemented. This work is focused on image formation. Image analysis and interpretation are primarily based upon application-specific software or sophisticated hardware [10]. The system supports a long distance camera-computer configuration. A picture editor in the PC acts as generic image manipulation software. A single computer can supervise multiple cameras.

E.2 Introduction

A complete machine vision system requires that at least the following three basic capabilities: image formation, image analysis, image interpretation.

Image formation is the reception of incoming light from an object or a scene, conversion of the light into electrical signals, and then processing of the signals until they are in a form that is compatible with a computer.

Image acquisition systems have also been discussed recently. One of these systems is the image analysis and data- acquisition techniques designed by K. G. Young and D. L. Hillis [21]. They coupled two video cameras with fast acquisition and display systems, developed for a Micro VAX-II. Data may be acquired at rates of 16.7 ms per one 60×80 eight-bit-pixel image frame or lower rates, up to about 0.5 s per frame.

In this paper, a low-cost general purpose machine vision system based on a personal computer of PC-XT/AT family and an electronic camera controller based-upon a Motorola 68HC11 single-chip microcontroller have been designed and implemented. This work is focused on image formation. Data may be acquired at rates of 0.35 sec per one 192×165 eight-bit-pixel image frame.

The system supports an up to 1000 meter long distance camera-computer configuration. A picture editor in the PC acts as generic image manipulation software. A single computer can supervise multiple cameras.

E.3 System configuration

The system discussed in this thesis consists of a camera with a CCD chip TC211 in it, an image grabber board with Motorola 68HC11 microcontroller on it, and a PC add-on card. The camera and the image grabber are

connected together by a 9-pin short cable, which conducts the pulse trains to shift the image data out and leads the analog pixel signal down to the grabber board. In principle they could be incorporated into the same module. The connection between the image grabber and the PC add-on card is a 15-pin long cable, which provides a digital, serial channel for transmitting image data from the image grabber to the PC add-on card and receiving the commands from the PC add-on card to the image grabber. Figure 2.1 is an overview of single camera configuration.

A system with real applications could be more powerful than the one described above. A much bigger board or a set of boards might be used not only for image grabbing but also for some other control functions because with the microprocessor, the image grabber has the intelligence to perform other functions. For instance, a stepping motor can be controlled by the 68HC11 to aim at a specified object. A multi-camera configuration is also possible. With a proper arbitration mechanism, more than one image grabber board can share the same channel. However, the count of pins which can fit in the edge of a add-on card are quite limited. In this case, the author would like to suggest the use of a HUB board to collect all the signals and then send them to the add-on card as shown in Figrue 2.2.

E.4 System operations

The system is operated under the software NMVS (NJIT Machine Vision System). It is an execution file named NMVS.EXE, created by linking several programs written in Turbo C and Intel 8088 assembly language.

When the user types in NMVS, a main menu screen will be shown. See Figure 2.5. The left half side of the screen is the image display area, while the right half side is the selection menu. The user can give commands to the system by making selections from the menu instead of typing the commands. Almost every command that the user has to type is reduced to one key selection on screen. Figure 2.6 shows the selection tree of the system. Note that every screen has a consistent exit point.

The camera entry on the main menu is directly related to the image grabber. It invokes exposure time changing and image data reception routines which were written in assembly language. The image can be stored in a file for later processing, and can be read back. These are done under the image I/O entry. Two file formats, which are binary and TIFF, are provided. The binary file is a direct collection of the gray level values of each pixel, 31680 bytes for the TC211. The TIFF file is a standard image exchange format which is supported by other software. It includes header information so more than 31680 bytes are needed for a TC211 image. The third selection is an image editor. It gives the user a chance to modify the image on screen. A blinking cursor can be moved by pressing the arrow keys on the keyboard. The pixel covered by the cursor can be modified to a desired gray level if the user turns on the pen mode.

E.5 Communications between image grabber and PC

Inclusion of a microcomputer in the camera permits digital image transmission. The image grabber in this system converts the analog video signal to digital locally and then sends it out in digital, so the length of the cable can be much longer. With the help of 26LS31 and 26LS32 transceiver chips, which can support RS-422 requirements, the length can be dramatically increased up to 1000 meters.

Another advantage of adding a microprocessor inside optical sensing equipments is the general increase of capability. For instance, an intelligent camera can change the exposure time as desired, perform autofocus, and do autoaiming in a predetermined pattern or under dynamic control of the PC.

The Motorola 68HC11 has a SPI (Serial Peripheral Interface) built in. The SPI performs synchronous serial data communications. Data streams can go out and come in synchronous with SCK simultaneously through ports MOSI and MISO, *i.e.* the channel is full duplex . The Motorola 68HC11 is assigned as the master, and it generates SCK clock during the transmission. Converted image data are sent out on MOSI, and commands from PC are picked up on MISO. Figure 2.4 shows the block diagram of this interface.

E.5.1 Data/Command transfer

The circuit diagrams of the data and command transfer channel with the SPI of 68HC11 are shown in sheet 2 and 3 of Appendix B. The 68HC11 SPI is set to master mode to provide SCK. PC slave circuits are implemented by discrete chips such as 74166 PISO (parallel- in-serial-out), 74164 SIPO (serial-in-parallel-out) registers, and 7493 counter.

Command transfer always occurs before the data transfer. It is more complicated than the data transfer, because it is initiated by PC. Therefore, it is explained after the data transfer is illustrated in the two succeeding subsections. A command consists of a command code followed by a byte or a series of information and a CR byte. The information portion is written in 7-bit ASCII, while the command code and CR are codes in 8-bit ASCII with bit 7 set. For instance, the command for taking a picture with 100 μs exposure time is: `81, 31, 8D`, where `81` is the command code, `31` is the hundred digit of 100 in ASCII, and `8D` is the CR with bit 7 set. At this time only one command code, 81, is defined. More other commands can be defined in this format.

The image data are transmitted on MOSI, and commands are received on MISO simultaneously. Only the master device can initiate data transmission or reception. If the PC needs to initiate a transmission, SCKREQ is used to request a command reception cycle while the image grabber is not sending data. Since the channel is full duplex, the data on MOSI should be thrown away in this case. On the other hand, a dummy command 00 is

fed back during the data transmission period when PC has nothing to say.

E.5.2 Image grabber side

The SPI control register (SPCR), SPI status register (SPSR), and SPDR are software accessible registers used to configure and operate the SPI system. The port D data direction control register (DDRD) also influences SPI activities.

```
LDAA #%00111100
STAA DDRD
LDAA #%01011110
STAA SPCR
```

The 68HC11 program segment listed above initializes the SPI and transfers the data and commands. Bit 4, 3, and 2 of port D are used as SCK, MOSI, and MISO by the SPI in this system. Bits DDRD4 and DDRD3 must be set to one to enable the SCK and MOSI output as a master[8]. When the SPI system is enabled as a slave, the DDRD2 bit must be set to one to enable the slave serial data output. When the SPI system is enabled as a master, the MISO acts as the master serial data input, regardless of the state of DDRD2. The SPCR configures the SPI. Storing %01011100 into SPCR selects SPI interrupt disable, SPI enable, SPI master, and SCK idle high. Figure 3.2 shows the timing relationship between data and the clock. The master device always places data on the MOSI line a half- cycle before the positive clock edge (SCK), in order for the slave device to latch the data.

Two status bits must be checked in between each byte of data sending. They are transfer complete flag and the transmitter data full detector. Bit 7 of the SPSR is the SPI transfer complete flag. If it is set upon completion of each byte of data transfer. Unless SPSR is read (with bit 7 set) first, attempts to write to SPDR are inhibited. Bit 0 of PORTA is connected to the SCKREQ of PC side and used as a transmitter data full detector during the data transfer. When a high voltage is detected on this bit, one byte of data is ready to be read on PC data port. Clearing this bit is accomplished by the data reading action on PC. No attempt should be made to write until this bit is cleared, or an overrun condition will exist.

```
......  ....  ....  ........................
        LDAB  DATA ; Load data from memory or ADR
        STAB  SPDR ; Send to SPI
WAIT    LDAA  SPSR ; Check transfer complete flag
        BPL   WAIT ; Jump if clear (incomplete)
WAIT1   LDAA  PORTA ; Check transmitter data full
        ASRA
        BLO   WAIT1 ; Jump if set (full)
......  ....  .....  ............................
```

The data transfer program shown above is the direct translation from the flowchart in Figure 3.3. A byte of data is read from the A/D result register or a memory location (in this example), depending on it is been sending on fly or from buffer, and then sent by a store to SPDR instruction, followed by two statuses checking action. If transfer complete and transmitter data not full, then program continues until the whole frame is sent.

The following program segment, based upon polling, is for command reception.

```
.....   ....   .....   ............
POLL    LDAA   PORTA ; Check SCKREQ
        ASRA
        BCS    POLL ; Jump if set (no request)
WAIT2   STAA   SPDR ; Send a dummy byte
        LDAA   SPSR ; Check if read ready
        BPL    WAIT2 ; Jump if clear (not ready)
        LDAA   SPDR ; Load a command byte
        STAA   ,X ; Store to command buffer
        CMPA   #CR1 ; End of command ?
        BNE    POLL ; Loop back if not finished
.....   ....   .....   ........................
```

In this system, since no other task is assigned, the MC68HC11 polls the PA0, the SCKREQ signal from PC, to see whether PC requests the transfer clock. Once a request is detected, MC68HC11 tries to read the command byte from SPDR after it sent out a dummy byte. If this is a CR (with bit 7 set), program goes on to command parsing; otherwise this is an element of a command and should be stored in the command buffer. In future work, the SCKREQ checking should be handled by interrupt, permitting the MC68HC11 to do something else.

E.5.3 PC side

The simplest way to select I/O ports is the fixed address method. This method checks the system address map, then selects one or a group of unused addresses with some proper circuits. The major disadvantage of this method is the selected address might overlap with some other add-on cards address. Figure 3.4 shows a more flexible alternate. Bits SW2 to SW8 of the DIP switch are compared one to one correspondent to PC address A9 to A3, while bit SW1 is compared with AEN. AEN is used to degate the CPU and other devices from the I/O channel to allow DMA transfers to take. When this line is active, the DMA controller has the control of the address bus, the data bus Read command lines (memory and I/O), and the Write command lines (memory and I/O). Since we don't use the DMA in this system, SW1 should be set to 0. When the value on the switch equals the P input of U14, the octal comparator, in Figure 3.4, the compare equal output is active. This is the group select control signal, and can be treated as high level decoding to enable U10, the decoder. By changing the combination of the DIP switch, the selected address can be varied in a certain address space.

The default setting, 01001111 at SW1 to SW8, selects 278h as the starting address of the group. The data receiving and command transfer program segments and shown below is based on this setting, so the address of the data port is 278h and that of the status/control port is 27ah. An address refers to two different ports, depending on the executed instruction is input or output. That is, 278h refers to SIPO register for input, or PISO

register for output; while 27ah refers to status port for input, or control port for output.

```
start:  mov dx,27ah     ;Select status port
ready?: in  al,dx
        shl al,1        ;Check data ready flag
        jnc ready?      ;Jump if clear (not ready)
        mov dx,278h     ;Select data port
        in  al,dx
        mov ds:[bx],al  ;Store data into memory
        inc bx
        dec cx
        jne start       ;Loop back if not finished
```

In the data transfer, bit 7 of the status port 27ah is a input data ready indicator. It will be set after system reset or a reading action to the data port. It will be cleared at every byte data transferred. The PC reads one byte of data from port 278h once this bit is checked set, then stores the data at a memory location pointed by `ds:[bx]`. The `cx` is a data counter. It was initialized to 31680 by the invoking program.

The following program segment is an example of sending the taking picture with a specified exposure time command.

```
....... .... .......... ......................
        mov  al,81h     ; Load the command code
        call send
```

```
        mov   al,[bp+6]   ; Load the information byte
        add   al,30h      ; Convert to ASCII
        call  send
        mov   al,8dh      ; CR (with bit 7 set)
        call  send
....... .... .......... ..................
send    mov   dx,278h     ;Select data port
        out   dx,al       ;Prepare command to image grabber
        in    dx,al       ;Take previous dummy data away
        mov   dx,27ah     ;Select control port
        mov   al,7fh      ;
        out   dx,al       ;Assert SCKREQ
ready?: in    al,dx
        shl   al,1        ;Check command received flag
        jnc   ready?      ;Jump if clear (not received)
        mov   al,0ffh     ;
        out   dx,al       ;Clear SCKREQ
        ret
```

Each byte of command is preloaded into al, and then sent out by calling the send subroutine. In command transfer, bit 7 of control port is the SCKREQ. A 0 at this bit inform the image grabber that PC request a command reception cycle. The send subroutine stores the command into PISO register, and resets the bit 7 of the control port. The input data ready indicator, bit 7 of the status port, in data transfer is used as a command received flag. It will be set if the command is received. After the command

byte is received, the send subroutine clears the SCKREQ, and returns.

E.6 Image processing at the PC

E.6.1 Image display

In the PC, screen output relies on two components: the display adapter and the display monitor. The "adapter" is a hardware card that you plug into one of the slots inside the PC, and the "monitor" is the display screen where the actual characters and graphics appear.

Like most peripheral devices in the IBM PC, the display adapters are programmed via 8-bit registers that are accessible by unique input port addresses. Besides, all display adapters are "memory-mapped." Each pixel on the display screen corresponds to 1 or more bits in a memory region called "video memory" or "video RAM". In graphics modes, for a black and white monitor, each pixel simply has to be either on or off, which means that a single bit is enough to display a pixel.

The details of the physical organization of the video RAM are bypassed by using the Turbo C graphics routines.

The Turbo C graphics coordinate system has its origin at the upper left hand corner of the physical screen with the x- axis positive to the right and the y-axis positive going downward. All graphics functions in the library work with a coordinate frame whose origin is located at the upper left hand corner of the current "viewport," a rectangular region within which all cur-

rent graphics output appears. A viewport can be defined by the *setviewport* function.

```
clearviewport();
for (j=0; j<165; j++)
{
  for (i=0; i<192; i++)
  {
    level = (*pixel-threshold)/regionwidth;
    switch (level)
    {
      case 5:
      case 4:  putpixel(i*2, j*2+1, max_color);
      case 3:  putpixel(i*2+1, j*2, max_color);
      case 2:  putpixel(i*2+1, j*2+1, max_color);
      case 1:  putpixel(i*2, j*2, max_color);
      case 0:  break;
  }
  pixel++;
}
}
```

Listed above, the core of the display routine is straightforward. The *clearviewport* function clears the current viewport, fills it with the background color, and resets the current position to the origin (the upper left corner) of the current viewport. The *putpixel* function fills the pixel with a

specified color, which is white in this case. Since only black and white are provided by HGC, a halftoning which represents 5 gray levels by counting 1 CCD pixel as 2×2 points is used. Every pixel is positioned by specifying its x and y coordinates, which are `i` and `j` in the program. Corresponding to the Turbo C graphics coordinate system, the origin is located at the upper left corner, and the `i` positive goes to the right and the `j` positive goes to the bottom. The image data from the image grabber are 8-bit per pixel, which means 256 gray level values are possible. However, the distribution of the number of pixel with respect to the intensity is usually not uniform. How to pick up a proper threshold value for image display is a popular subject has been discussing [3]. In this system, the lowest pixel data is selected as the `threshold`, and the one fifth of the difference between the highest and the lowest pixel value is selected as the `regionwidth`. Each of the pixel data is divided by the `regionwidth` for grading. The *switch* instruction in Turbo C compares a expression with the condition value in each *case*, then executes the instructions that the matched case assigns to. Since no *breaks* are put between *cases*, the instructions in later *cases* will be executed continuously after that in matched *case* is completed. Figure 3.5 shows the patterns for each gray level.

E.6.2 Image editor

The image we got from the sensor is sometimes not desirable. For instance, there may be spots on it. The image editor gives the user the ability of modifying the image on screen. Not only can it correct the spots, but also

it can add some characters or symbols to the image, *e.g.* a title or date.

The 192 × 165 bytes of pixel are stored in the memory location given by MS-DOS. They are mapped starting from the upper left corner on the screen to the right, and from the top to the bottom. So the first byte is located at the origin, the 192nd byte is located at the upper right corner, and the 193rd byte at is the first location of the second row.

The image editor program is listed in Appendix C, and the flowchart is shown in Figure 3.6. The Turbo C function *bioskey* scans the keyboard. It returns the IBM keyboard code of the pressed key, or 0 for no pressed key. If the pressed key is not a function key ("E", "P", "0"-"5") or no key is pressed, the program does nothing but blinking the cursor. The "E" key exit the routine, and the "P" key toggles the pen mode. If the pen mode id ON, the routine modifies the pixel pointed by the cursor according to the current gray value. The number key, "0" to "5", changes the current gray value. If the pen mode is ON, it affects the current pointed pixel. The arrow key moves the cursor one step to the direction. The "Home", "End", "PageUp", and "PageDown" are also valid as moving to four corner positions. All operations are followed by blinking the cursor, and then go back to the keyboard scanning.

E.6.3 Image printout

There are many desk top publishing packages which can print out image files. For instance, PageMaker, and Microsoft Word, provided by the Mi-

crosoft Corporation, can merge text and image files and print them out. There are also certain standard file formats for such image files, *e.g.*, the TIFF and PCX. As long as the usage file is properly formatted, we can take the advantages of those software packages instead of writing printer driver programs ourselves.

The TIFF (Tag Image File Format) is used in this system. The TIFF was defined jointly by Aldus and Microsoft in conjunction with leading scanner vendors and other interested parties. It is a tag based file format that is designed to promote the interchange of digital image data. The general scenario for which TIFF was invented assumes that applications software for scanning or painting creates a TIFF file, which can then be read and incorporated into a document or publication by an application such as a desktop publishing package.

There are many fields described in the TIFF because it was designed to be powerful and flexible. It takes a fair amount of effort to handle all the options (probably no application does a complete job). The following fields are applicable to us: **SubfileType, ImageWidth, ImageLength, BitsPerSample, Compression, PhotometricInterpretation,** and **StripOffsets.** They describe the file contains uncompressed, full resolution image data with the dimension 192 $\times$ 165 pixels, 8 bits per pixel, and value 0 is imaged as black, and value $2^8 - 1$ is imaged as white. Those fields, the image file header, and image file directory are the first 98 bytes of the file, followed by 31680 bytes of uncompressed pixel data. These also apply to our system. When a TIFF output is requested, the 98 bytes of non-image

data will be written to a file, followed by 31680 bytes of gray level image data, to form a 31778 bytes long binary file with the name provided by the user and the extension "TIF".

The Turbo C function *open* opens a file with the given file name and the specified attribute for unbuffered and unformatted I/O operations. The *read* and *write* instructions perform the saving and retrieving a specified number of bytes of data at the current position in the opened, unbuffered and unformatted file.

E.7 System Analysis of Limits and Performance

E.7.1 Image grabbing

[19] describes the details of the timing, clocking, and operation of TC211 with a television monitors. In this thesis, the chip is operated at much lower frame rate to fit the time consuming A/D conversion in MC68HC11. It takes 32 cycles to convert analog pixel data to digital, *i.e.*16 μs, if 1 cycle equals to 0.5 μs.

E.7.2 Sending data from buffer

From the viewpoint of hardware, SPI can transfer image data to PC at a bit rate up to 1 Mhz. It takes

$$\frac{8 \times 165 \times 192}{10^6} = 0.25sec$$

to send a full uncompressed frame. However, there are two status bits that have to be checked during the transfer: transfer complete and trans-

mitter data full. The following program segment is a rewrite from the one mentioned in the previous chapter with instruction time in cycles.

```
.....  ......  ....  ....
( 2 )          LDAB  DATA ; Load data from memory or ADR
( 4 )          STAB  SPDR ; Send to SPI
( 4 )  WAIT    LDAA  SPSR ; Check transfer complete flag
( 3 )          BPL   WAIT ; Jump if clear (incomplete)
( 4 )  WAIT1   LDAA  PORTA ; Check transmitter data full
( 2 )          ASRA
( 3 )          BLO   WAIT1 ; Jump if set (full)
.....  ......  ....  .....
```

Assumed every tests are passed at the first check, the segment takes 22 cycles for execution. If 1 cycle equals to 0.5 μsec, a full frame transmission time becomes

$$22\mu sec \times 165 \times 192 \sim 0.35sec.$$

The transmission rate, the reciprocal of the transmission time, will be 2.87 frames per second. This makes it possible for a machine to inspect and recognize sample parts at a rate of higher than 2 items per sec.

PC/XT is running at the speed of about 210 ns per cycle. Using a polling technique, it takes about 70 cycles, 14.7 μsec to read a byte of data. It is fast enough to handle the image reception.

E.7.3 Sending data on fly

An A/D sequence begins one E clock cycle after a write to the ADCTL (A/D control/status register). It takes 32 cycle to convert analog pixel data to digital. During the 32 cycles, MC68HC11 does nothing but turning the SRG (serial register gate) of the TC211 ON and OFF [4]. Therefore, it will be more efficient to send the digital data on fly. The following program segment is one of the possible ways.

```
.....  ......  ....  ............; ..................
( 4 )          STAA  ADCTL       ; Begin A/D 32 cycle
.....          ....  ............ ; ..................
( 4 )          STAA  PORTA       ; Turn OFF the SRG
( 4 )          LDAA  SPSR        ; Clear bit 7 of SPSR
( 4 )          LDAB  ADRn        ; Get the A/D result
( 4 )          STAB  SPDR        ; Send to SPI
( 4 )          LDAA  PORTA       ;
( 2 )          ANDA  #%10111111  ;
( 4 )          STAA  PORTA       ; Turn ON the SRG
( 2 )          NOP               ; 4 cycles
( 2 )          NOP               ; Time delay
( 2 )          ORA   #%01000000  ;
( 4 )          STAA  PORTA       ; Turn OFF the SRG
.....  ......  ....  ............ ; ..................
```

In this example, MC68HC11 takes data from the A/D result register after it turns off the SRG, then sends to SPI. The required time consump-

tion for shifting out the next pixel, *i.e.* the time interval between the two SRGs, or the execution time the two instructions with the "Turn OFF the SRG" comment in the program segment is 28 cycles. Four more cycles consumption should be added to match the A/D conversion time, 32 cycles. Now, the transmission rate is equal to the data generation rate, which is

$$\frac{1}{16\mu sec \times 165 \times 192} \sim 1.97 frames/sec$$

if 1 cycle equals to 0.5 μsec.

The most popular serial interface of computer system is the EIA RS-232. It supports transmission distance up to 50 feet and it is suitable for a one to one dedicated configuration. The line driver and receiver used in this system, 26LS31 and 26LS32, meet all the requirements of RS-422[2], which supports long distance transmission up to 1200 meters, and up to 10 fan outs. Figure 4.1 shows the transmission ability.

E.7.4 Experiments

Similar to the TC210 in Ref[12], TC211 can be operated under constant light, however, smearing is present due to the incident light on the whole chip while the pixel data is transferred out of the CCD. To reduce the smear, reducing the pixel readout time as much as possible is one of the solutions. The recommended clock rate is 7.16 Mhz. The maximum clock rate we have now is

$$\frac{1}{16\mu sec} = 62.5 Khz,$$

so smear problem happens all the time. Figure 4.2 is the best picture we got from the image grabber now. It was taken under fairly bright constant light source, with f/16 aperture, 1 meter focus distance, 100 μsec exposure time.

E.8 Conclusion

In this paper, a low-cost general purpose machine vision system has been designed and implemented. The system demonstrates the application of a Motorola single-chip microcontroller to implement an intelligent camera system with an interconnected personal computer PC-XT/AT. Synchronous digital data transmission using a line driver and receiver provides the capability to separate the camera and computer by a distance. The SPI of Motorola 68HC11 also provides the possibility to configure a single host computer with multiple cameras.

To display the image on the screen of a personal computer, a variety of display adapters are studied. A halftoning algorithm is used to represent a gray-level picture on a two level (ON and OFF) screen. The image data can be stored in a binary file, which can be easily retrieved by custom image display software, or in a TIFF file, which is compatible with many desk-top publishing software packages.

Appendix F

Bibliography

[1] Barkakati, Nabajyti, The Waite Group's Turbo C Bible. Howard W. Sams & Company, Indianapolis. Indiana. 1989.

[2] Bertine, H. V., "Physical Level Protocols," IEEE Transactions on Communication, April 1980.

[3] Hertz, Lois and Schafer, Ronald W., "Multilevel Thresholding Using Edge Matching," Computer Vision, Graphics, and Image Processing, Vol. 44, No. 3, Dec., 1988.

[4] Li, Chishin, "Microcomputer-based CCD Imager," Master Thesis, NJIT, Newark, New Jersey, 1989.

[5] M68HC11EVB Evaluation Board User's Manual, Motorola Inc., 1986.

[6] M68HC11 HCMOS Single-Chip Microcontroller Programmer's Reference Manual, Motorola Inc., 1986.

[7] M68HC11 Reference Manual, Motorola Inc., 1989.

[8] M68HC11 Single-Chip Microcontroller Reference Manual, Motorola Inc., 1986.

[9] Machine Vision Systems: A Summary and Forecast, Tech Tran Consultants, Inc., 1985.

[10] Ngan, K. N., Kassim, A. A. and Singh, H. S.: "Parallel Image-processing System Based on the TMS32010 Digital Signal Processor," Insititution Electrical Engineering Proceedings, Vol. 134, Part E, No. 2, Mar. 1987.

[11] Optoelectronics and Image-Sensor Data Book, Texas Instruments Inc., 1987.

[12] Parten, M. E. and Mau, K. Y., "A CCD Image Sensor Frame Grabber," IEEE Conference, June ,1988.

[13] Shoap, Steve, "Build a Robotic Vision System with a Digital Signal Processor," Electronic Design, June 23, 1988.

[14] Reference to Microsoft Word, Microsoft Corporation, 1989.

[15] Turbo C REFERENCE GUIDE, Borland International, Inc., 1987.

[16] "Tag Image File Format Specification," Revision 5.0, Aldus Corporation. , 1988.

[17] Turbo C USER'S GUIDE, Borland International, Inc., 1987.

[18] "Using the Serial Peripheral Interface to Communicate Between Multiple Microcomputer," Motorola Semiconductor Application Note, AN991, 1987.

[19] Weber, Roger L., "Small-area CCD Imager Evaluation Using Television Monitors," Electronic Imaging '87 Conference, February, 1987.

[20] Wray, A., "Computer-controlled Manipulator System for Autonomous Inspection," Institution of Electrical Engineering Proceedings, Vol 134, Part A, No. 3, Mar. 1989.

[21] Young, K. G. and Hillis, D. L., "Image Analysis and Data-acquisition Techniques for Infrared and CCD cameras for ATF," Review of Scientific Instruments, Vol. 59, No. 8, August, 1988.