SECTION 11.6: I/O PROGRAMMING WITH C/C++ AND VB

With the rise in popularity of C/C++ and Visual Basic in recent decade
it is fitting to explore how these languages are used in x86 I/O programming.
this section we discuss I/O operation in C/C++ and Visual Basic. We discuss I/
programming in Microsoft's Visual C/C++, Borland's Turbo C, Linux C/C++, a
Visual Basic environments.

Visual C/C++ I/O programming

Microsoft Visual C++ programming is one of the most widely used pr
gramming languages on the Windows platform. Since Visual C++ is an obje
oriented language, it comes with many classes and objects to make programmi
easier and more efficient. Unfortunately, there is no object or class for direct
accessing I/O ports in the full Windows version of Visual C++. The reason f
that is that Microsoft wants to make sure the x86 system programming is und
full control of the operating system. This precludes any hacking into the syste
hardware. This applies also to Windows NT. In other words, none of the syste
INT instructions such as INT 21H and I/O operations that we have discussed
previous chapters are applicable in Windows NT and its subsequent versions.
access the I/O and other hardware features of the x86 PC in the NT environme
you must use MS Developer's Software provided by Microsoft. The situation
different in the Windows 9x (95 and 98) environment. While INT 21H and oth
system interrupt instructions are blocked in Windows 9x, direct I/O addressing
available. To access I/O directly in Windows 9x, you must program Visual C
in console mode. The instruction syntax for I/O operations is shown in Table 1
5. Notice the use of the undersign (_) in both the _outp and _inp instructions.
must also noted that while the x86 Assembly language makes a distincti
between the 8-bit and 16-bit I/O addresses by using the DX register, there is
such distinction in C programming, as shown in Table 11-5. In other words,
the instruction "outp(port#,byte)" the port# can take any address value betwe
0000 - FFFFH.

Table 11-5. Input/Output Operations in Microsoft Visual C++

x86 Assembly	Visual C++
OUT port#,AL	_outp(port#,byte)
OUT DX,AL	_outp(port#,byte)
IN AL,port#	_inp(port#)
IN AL,DX	_inp(port#)

Visual C++ output example

Next we give some examples of I/O programming in Visual C+
Reexamine Example 11-7 in Assembly language. The Visual C++ version of th
program is given in Example 11-8. In Example 11-8, we are toggling all the b
of PA and PB of the 8255 in the PC Trainer. Notice the following points.

1. The use of the _sleep function to create a delay.
2. The use of kbhit to exit upon any key press.
3. The use of 0x in _outp(0x300,0x80) to indicate that the values are in hex.

Visual C++ input example

As an example of inputting data in Visual C++, examine Example 11
We wrote the Assembly language version of this program in Example 11-5.
Example 11-9, we are getting a byte of data from port A and sending it to both
and PC. Notice that when we bring a byte of data in, we save it using the varia

Example 11-8

Write a Visual C++ program to toggle all bits of PA and PB of the 8255 chip on the PC Trainer. Use the kbhit function to exit if there is a keypress.

Solution:

```
//Tested by Dan Bent
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>
#include<windows.h>
void main()
{
cout<<setiosflags(ios::unitbuf);          // clear screen buffer
cout<<"This program toggles the bits for Port A and Port B.";
_outp(0x303,0x80);          //MAKE PA,PB of 8255 ALL OUTPUT
do
{

  _outp(0x300,0x55);          //SEND 55H TO PORT A
  _outp(0x301,0x55);          //SEND 55H TO PORT B
  _sleep(500);                //DELAY of 500 msec.
  _outp(0x300,0xAA);          //NOW SEND AAH TO PA, and PB
  _outp(0x301,0xAA);
  _sleep(500);
}
while(!kbhit());
}
```

Example 11-9

Write a Visual C++ program to get a byte of data from PA and send it to both PB and PC of 8255 in PC Trainer.

Solution:

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>
#include<windows.h>
#include<process.h>
//Tested by Dan Bent
void main()
{
     unsigned char mybyte;
      cout<<setiosflags(ios::unitbuf);     // clear screen buffer
     system("CLS");
     _outp(0x303,0x90);     //PA=in, PB=out, PC=out
     _sleep(5);     //wait 5 milliseconds
     mybyte=_inp(0x300);     //get byte from PA
      _outp(0x301,mybyte);     //send to PB
      _sleep(5);
      _outp(0x302,mybyte);     //send to Port C
      _sleep(5);
      cout<<mybyte;          //send to PC screen also
      cout<<"\n\n";
}
```

mybyte before we send it out. Make a habit of doing this every time you input data. Avoid combining a bunch of input and output operations together in a single line. That kind of dense code is very difficult for other programmers to read. Also, notice how the "unsigned char mybyte" line dictates the size of data as unsigned character. This allows the mybyte variable to be an 8-bit data, taking values of 00 - FFH.

I/O programming in Turbo C/C++

Borland is a major provider of software for the x86 PC. The company was founded in the early 1980s and became one of the early pioneers in the development of software for the x86 PC. They are also known by the name of Inprise. Check their web site www.borland.com. Their Turbo C/C++ is a widely used program compiler for x86 PCs. It supports I/O programming of ports as shown in Table 11-6. It must be noted that these I/O functions are no longer supported in Borland C++ Builder and you must write your own I/O functions using Assembly language. See the Micro Digital Education web site www.microdigitaled.com for more information on this and other topics.

Table 11-6: Input/Output Operation in Borland C++

x86 Assembly	Turbo C++
OUT port#,AL	outp(port#,byte)
OUT DX,AL	outp(port#,byte)
IN AL,port#	inp(port#)
IN AL,DX	inp(port#)

Example 11-10

Write a Borland (Inprise) Turbo C program to toggle all bits of PA and PB of the 8255 chip on the PC Trainer. Put a 500 ms (milliseconds) delay between the "on" and "off" states. Use the kbhit function to exit if there is a keypress.

Solution:

```
#include<conio.h>
#include<stdio.h>
void main()
 {
 printf("This program toggles the bits for Port A and Port B.");
 outp(0x303,0x80);        //MAKE PA,PB of 8255 ALL OUTPUT
 do
  {
  outp(0x300,0x55);       //SEND 55H TO PORT A
  outp(0x301,0x55);       //SEND 55H TO PORT B
  delay(500);             //DELAY of 500 msec.
  outp(0x300,0xAA);       //NOW SEND AAH TO PA, and PB
  outp(0x301,0xAA);
  delay(500);
  }
  while(!kbhit());
 }
```

Example 11-11

Write a Turbo C/C++ program to get a byte of data from PA and send it to both PB and PC of 8255 in the PC Trainer.

Solution:

```
#include<conio.h>
#include<stdio.h>
void main()
{
    unsigned char mybyte;
    clrscr();
    printf("This program gets a byte from PA and sends it to PB,PC and screen\n.");
    outp(0x303,0x90);     //PA=in, PB=out, PC=out
    delay(5);             //wait 5 milliseconds
    mybyte=inp(0x300);    //get byte from PA
    outp(0x301,mybyte);   //send to PB
    delay(5);
    outp(0x302,mybyte);   //send to Port C
    delay(5);
    printf("The input from PA is equal to %X in hex \n",mybyte);
```

I/O programming in Linux C/C++

Linux is a popular operating system for the x86 PC. You can get a copy of the latest C/C++ compiler from http://gcc.gnu.org. Table 11-7 provides the C/C++ syntax for I/O programming in the Linux OS environment.

Table 11-7. Input/Output Operations in Linux

x86 Assembly	Linux C/C++
OUT port#,AL	outb(byte,port#)
OUT DX,AL	outb(byte,port#)
IN AL,port#	inb(port#)
IN AL,DX	inb(port#)

Compiling and running Linux C/C++ programs with I/O functions

To compile the I/O programs of Examples 11-12 and 11-13, the following points must be noted.

1. To compile with a keypress loop, you must link to library ncurses as follows:
   > gcc -lncurses toggle.c -o toggle
2. To run the program, you must either be root or root must change permissions on executable for hardware port access.
   Example: (as root or superuser)
   > chown root toggle
   > chmod 4750 toggle

Now toggle can be executed by users other than root. More information on this topic can be found at www.microdigitaled.com.

Example 11-12

Write a C/C++ program for a PC with the Linux OS to toggle all bits of PA and PB of the 8255 chip on the PC Trainer. Put a 500 ms delay between the "on" and "off" states. Pressing any key should exit the program.

Solution:

```
 //      This program demonstrates low level I/O
 //      using C language on a Linux based system.
  //     Tested by Nathan Noel //
 #include <stdio.h>        // for printf()
 #include <unistd.h>       // for usleep()
 #include <sys/io.h>       // for outb() and inb()
 #include <ncurses.h>      // for console i/o functions
 int main ()
  {
  int n=0;                  // temp char variable
  int delay=5 e5;           // sleep delay variable

  ioperm(0x300,4,0x300); // get port permission
  outb(0x80,0x303);       // send control word

  //----- begin ncurses setup ----------
  //--- (needed for console i/o) -------

   initscr();               // initialize screen for ncurses
   cbreak();                // do not wait for carriage return
   noecho();                // do not echo input character
   halfdelay(1);            // only wait for 1ms for input
                            // from keyboard
  //----- end ncurses setup ----------

   do                       // main toggle loop
     {
     printf("0x55 \n\r");   // display status to screen
     refresh();             // refresh() to update console
     outb(0x55,0x300);      // send 0x55 to PortA (01010101B)
     outb(0x55,0x301);      // send 0x55 to PortB (01010101B)
     usleep(delay);                  // wait for 500ms (5 e5 microseconds)
     printf("0xAA \n\r");   // display status to screen
     refresh();             // refresh() to update console
     outb(0xaa,0x300);      // send 0xAA to PortA (10101010B)
     outb(0xaa,0x301);      // send 0xAA to PortB (10101010B)
     usleep(delay);                  // wait for 500ms
                            // get input from keyboard
     n=getch();             // if no keypress in 1ms, n=0
                            // due to halfdelay()
    }
  while(n<=0);              // test for keypress
                            // if keypress, exit program
    endwin();               // close program console for ncurses
    return 0;                        // exit program
    }
```