Giriş

Bu kısımda, ağırlıklı olarak Windows işletim sistemlerinde çalışan (Turbo C, Dev-C++ gibi) derleyicilerin bünyesinde bulunan port fonksiyonları ve kullanımları anlatılacaktır. Linux işletim sisteminde benzer uygulamaların nasıl yapılacağı bölüm sonunda verilmiştir.

Bir program içerisinden donanımsal birimlere erişmek veya onları kullanmak için birçok yol vardır. En basiti, bu gibi birimlere aynı bellek gözüne erişilmiyormuş gibi gösterici kullanarak erişmektir; ancak bu durum sistem mimarisinden dolayı her zaman mümkün olmayabilir. Bu durumda, ilgili birimlere erişmek için derleyicilerin sahip olduğu hazır kütüphane fonksiyonları kullanılır[1].

NOT

- Windows işletim sistemi, güvenlik nedeniyle, C derleyicilerine ait portlara erişim fonksiyonlarını kullanmaya da izin vermeyebilir.
- Linux işletim sistemi kullanıcıları ana kullanıcı (root) olmadığı sürece, portlara erişim izni yoktur.

18.1 Port Kavramı

Anakartın üzerinde bir bilgisayarın en önemli bileşenleri (Veriyolları, Portlar, CPU, RAM, BIOS, ChipSet, ROM, I/O devrelerinin çoğu) bulunur. Anakart, sistemin çalışmasını organize eder. Bu organizasyon anakart üzerinde bulunan yongalar (entegre devreler) sayesinde gerçekleşir. Anakart üzerinde bilgisayara veri giriş/çıkış için kullanılan pinlere veya elektriksel bağlantı noktalarına **port** denir. Örneğin: Paralel port (LPT), seri port (COM), AGP portu, PCI portları gibi.

Daha fazla bilgi için **burayı** tıklayın.

18.2 Port Giriş/Çıkış Fonksiyonları

Bir bilgisayarın portlarına erişmek için birçok fonksiyon vardır. Tablo 18.1'de, Turbo C derleyicisinde bulunan ve bu konu ile ilgili birkaç fonksiyon tanıtılmıştır.

NOT

Turbo C derleyicisinde, port fonksiyonları kullanılabilmesi için `dos.h` başlık dosyası programa ilave edilmelidir.

Tablo 18.1: *dos.h'te tanımlı bazı port erişim fonksiyonları*

Port Fonksiyonu	Açıklama
`void outp(int port_adresi,int bayt_degeri);`	Porta bir baytlık veri yazar
`void outport(int port_adresi,int deger);`	Porta bir kelime* yazar
`void outportb(int port_adresi,unsigned char deger);`	Porta bir baytlık veri yazar
`int inp(int port_adresi);`	Porttan bir baytlık veri okur
`int inport(int port_adresi);`	Porttan bir kelime okur
`char inportb(int port_adresi);`	Porttan bir baytlık veri okur
(*) *kelime (word)* : Porta yazılacak veya porttan okunacak, bir tamsayının bellekte kaplayacağı alanı temsil eder. (Bu alan `sizeof()` operatörü ile öğrenilebilir)	

Port foksiyonlarının kullanımı, örnek programlar üzerinde, bir sonraki bölümlerde incelenmiştir. Bütün programlar Turbo C derleyicisinde denemiştir. Eger bu derleyiciye sahip degilseniz, **buradan** inderbilirsiniz.

18.3 Paralel Port Örnekleri

Bu bölümde, bir önceki kısımda verilen port fonksiyonları ile, bir PC'nin paralel portunun nasıl denetleneceği 6 tane örnek programda anlatılmıştır.

NOT

Standart bir PC'de LPT nin alt portlarının adresleri, DATA için **0x378**, STATUS için **0x379** ve CONTROL **0x37A** dır;

Program 18.1: *outp fonksiyonunun kulanımı*

```
01:  /* 18prg01.c: outp örneği */
02:
03:  #include <stdio.h>
04:  #include <dos.h>
05:
06:  #define DATA 0x0378
07:
08:  int main()
09:  {
10:      int deger = 25;
11:
12:      outp(DATA, deger);
```

```
13:
14:      printf("\n%X nolu adrese %d degeri yazildi.",
15:  DATA, deger);
16:
17:   return 0;
     }
```

ÇIKTI

```
378 adresine 25 degeri yazildi.
```

Program 18.1'de 6. satırda tanımlanan porta, 12.satırda 25 değeri yazılmaktadır. Bu değer PC paralel portunun DATA uçlarına yazılır. Bu sebeple 25 değeri ikili sistemde (binary) ifade edilip 8 bite bölünür, yani 25 = 00011001 şekinde DATA portuna yazılır. Burada 1 portun ilgili bacağına +5V DC sinyalini gönderir. 0 olan bağlantı noktalarına ise sinyal gönderilmez. Bu değerler basit bir voltmetre ile ölçülüp test edilebilir.

Porta yazılmak veya porttan okunmak istenen veriyi ikili (binary) olarak görüntülemek yararlı olabilir. Program 18.2'de `cevir_taban2` fonksiyonu bu amaçla yazılmıştır.

Program 18.2: *outportb fonksiyonun kullanımı*

```
01: /* 18prg02.c: outportb fonksiyonu */
02:
03: #include <stdio.h>
04: #include <dos.h>
05: #include <math.h>
06:
07: #define DATA 0x0378
08:
09: long cevir_taban2(int);
10:
11: int main()
12: {
13:    int deger = 0x19; /* deger = 25 */
14:
15:    outportb(DATA,deger);
16:
17:    printf("\nDATA portuna gonderilen deger %d :
18: %08ld",deger, cevir_taban2(deger));
19:
20:   return 0;
21: }
22:
23:  /* Bu fonksiyon 10 tabanındaki bir sayıyı
24:     2 tabınındaki karşılığını hesaplar. */
25: long cevir_taban2(int x)
26: {
27:     int  i = 0, k;
28:     long bin = 0;
29:
30:     while( x>0 )
31:     {
32:        if(x%2) k = 1;
33:        else    k = 0;
34:        bin += k*pow(10,i++);
```

```
35:          x /= 2;
36:       }
37:
38:    return bin;
     }
```

ÇIKTI

```
DATA portuna gonderilen deger 25 : 00011001
```

9. satırdaki `cevir_taban2` fonksiyonu, kendisine parametere olarak gelen bir tamsayıyı iki tabana çevirir. Ekranda porta yazılan değer ve onun iki tabanındaki karşılığı, uygun bir formatla, 8 bit halinde gösterilmiştir.

`inp()` ve `inportb()` fonksiyonları, PC bağlantı noktalarından sırasıyla bir karakter ve bir baytlık veri okumak mümkündür. Program 18.3, bu fonksiyonlar ile nasıl veri okunacağına dair iyi bir fikir verir.

Program 18.3: *inp ve inportb fonksiyonlarıyla paralel porta atanan varsayılan değerleri öğrenme*

```
01: /* 18prg03.c: inp ve inportb fonksiyonlarının
02: kullanımı */
03:
04: #include <dos.h>
05: #include <stdio.h>
06:
07: #define DATA    0x0378
08: #define STATUS  DATA+1
09: #define CONTROL DATA+2
10:
11: int main()
12: {
13:    int  veri;
14:
15:    puts("Paralel porta atanan degerler (Hex):");
16:
17:    veri = inp(DATA);
18:    printf( "Data portu    : %X\n",veri );
19:
20:    veri = inp(STATUS);
21:    printf( "Status portu  : %X\n",veri );
22:
23:    veri = inportb(CONTROL);
24:    printf( "Kontrol portu : %X\n",veri );
25:
26:  return 0;
     }
```

ÇIKTI

```
Paralel porta atanan degerler (Hex):
Data portu    : 4
Status portu  : 7F
Kontrol portu : CC
```

Programın elde ettiği değerler, porta hiç bir müdehale olmadan elde edilmiştir ve her bilgisayarda

başka bir sonuç verebilir. Bu fonksiyonların tek parameteresi olduğuna dikkat ediniz.

Bir porta herhangi bir veri yazıldıktan sonra, bu veri o portun saklayıcısına (*register*) yazılır ve yeni bilgi yazılmadıkça orada kalır. Program 18.4 CONTROL portuna `ouportb` ile yazılan bir verinin `inportb` fonksiyonu ile okunması gösterilmiştir.

Program 18.4: *inportb ve outportb fonksiyonlarının kullanımı*

```
01: /* 18prg04.c: inportb ve outportb örneği */
02:
03: #include <stdio.h>
04: #include <dos.h>
05:
06: #define PORT 0x037A
07:
08: int main()
09: {
10:     int deger;
11:
12:     deger = inportb(PORT);  /* varsayılan deger */
13:     printf("\nPorta veri yazilmadan onceki deger :
14: %X",deger);
15:
16:     deger = 0x0A; /* deger = 10 */
17:     outportb(PORT, deger);
18:
19:     deger = inportb(PORT);
20:     printf("\nPorta veri yazdiktan sonraki deger :
21: %X",deger);
22:
     return 0;
    }
```

ÇIKTI

```
Porta veri yazilmadan onceki deger : CC
Porta veri yazdiktan sonraki deger : CA
```

Program 18.4'ün çıktısı incelendiğinde, portta varsayılan değerin CCh, veri yazıldıktan sonraki değerin CAh olduğu görülmektedir. CONTROL portunun ilk 4-bitine müdehale edilebildiği halde ikinci 4-biti değiştirilememiş. Neden?

18.4 Seri Port Örnekleri

Bu bölümde, yine Standart C'de olmayan bilgisayarın seri portları üzerinden iletişim konu edilecektir.

NOT

Standart bir PC'de COM1 için ilk adres **0x3F8**, COM2 için **0x2F8** dir;

Standart PC'lerin seri iletişim portlarına erişim UART olarak adlandırılan bir birim üzerinden gerçekleştirilir. Bu birim anakat üzerindeki bir entegre devredir. Ancak temel olarak bilinmesi gereken alma saklayıcısına ve gönderme saklayıcısına nasıl erişileceği ve UART'ın ayarlarının nasıl yapılacağıdır. Program 18.5'de bir dosya içeriğinin karakter karakter seri port üzerinden karşı tarafa nasıl gönderileceği görülmektedir[1].

Program 18.5: *Bir metin dosyasının içeriğini seri porta aktarır.*

```
/* 18prg05.c: Bir metin dosyasının içeriğini seri
porta aktarır  */

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

int main()
{
   char kr;
   FILE *dosya;

   /* UART'ın ayarlanması */
   outportb(0x3FB,0x80);
   outport (0x3F8,0x0C);
   outportb(0x3FB,0x1B);  /* 9600 bps.dur biti
1.cift eslik, 8 bit veri*/

   /* dosya açılıyor */
   if ( (dosya=fopen("deneme.txt", "r")) == NULL) {
     puts ("Hata olustu! Dosya acilmadi.");
     exit(1);
   }

   while( !feof(dosya) )
   {
       kr=getc(dosya);                      /*
dosyadan bir karakter oku  */
       while ( (inportb(0x3FD) & 0x20)==0 ); /*
gönderme saklayicisi sınanıyor */
                ouportb(0x3F8,kr);          /*
porta gönderiliyor */
   }
   fclose(dosya);                           /*
dosya kapatiliyor */

 return 0;
}
```

Bir UART iletişim işine geçmeden önce ayarlanmalıdır; yani, iletişim hızı, hata biti kullanıp kullanılmayacağı gibi birtakım bilgilerin yerleştirimi yapılmalıdır. UART'ın herhangi bir andaki durumu, yani veri göndermeye hazır olup olmadığı, yeni veri gelip gelmediği gibi bilgiler hat durum saklayıcısı üzerindeki bitlere bakılarak anlaşılır. Örneğin, UART'a gönderilmesi için bir veri yazılmadan önce, göndermek için uygun olup olmadığı sınanmalıdır. Program 18.6'de seri port üzerinden gelen karakterleri alıp ekrana nasıl yazıldığı görülmektedir[1].

Program 18.6: *Seri port üzerinden gelen karakterleri alıp ekrana yazar.*

```
01: /* 18prg06.c: Seri port üzerinden gelen karakterleri
02: alıp ekrana yazar */
03:
04: #include <stdio.h>
05: #include <dos.h>
06:
07: int main()
08: {
09:    char kr;
10:
11:    /* UART'ın ayarlanması */
12:    outportb(0x3FB,0x80);
13:    outport (0x3F8,0x0C);
14:    outportb(0x3FB,0x1B);  /* 9600 bps.dur biti
15: 1.cift eslik, 8 bit veri*/
16:
17:    while(1)
18:    {
19:       while( (inportb(0x3FD) & 0x01)==0 ); /* yeni
20: karakter gelene kadar bekle */
21:          kr=inportb(0x3F8);              /*
22: geleni al ve kr'ye yerleştir  */
23:          printf("%c", kr);
       }

     return 0;
    }
```

Soru: Son iki programı öyle değiştirin ki, birinin klavyesinden girilen, diğerinin ekranında görülsün.

18.5 Linux'de Portlara Erişim

Linux işletim sisteminde portlara erişmek için birkaç yol vardır. Burada, gcc derleyicisinin içine gömülebilen assemble dili kullanılarak oluşturulan port fonksiyonları gösterilecektir.

Linux'de ana kullanıcı (root) olmadıkça veya ana kullanıcı izin vermedikçe portlara erişmeniz mümkün değildir. Bu yüzden, önce programın portlara erişim izni verip vermediği sınanmalıdır. Bunun için, `/usr/include/sys/io.h` dosyası içinde tanımlı `ioperm()` fonksiyonu kullanılabilir.

Program 18.7 de basit bir port erişim programı verilmiştir. Kullanıcıların, programın başına ilave edebileceği başlık dosyaları için ayrıca bkz. Bölüm 20.

Program 18.7: *Linux'de port erişimi*

```
01: /* 18prg07.c
02:    Linux işletim sisteminde portlara erişim */
03:
04: #include <stdio.h>
05: #include <stdlib.h>
06: #include <sys/io.h>
07:
```

```
08: #include "linuxPort.h"
09:
10: int main()
11: {
12:    int deger = 25;
13:
14:    /* porta erişim izni var mı?
15:       sadece root ve onun izin verdiği kullanıcılar
16: erişebilir ) */
17:       if( port_erisim() ) exit(1);
18:
19:     outportb(DATA, deger);
20:
21:     printf("\n%X nolu adrese %d degeri yazildi.",
22: DATA, deger);
23:
24:
     return 0;
    }
```

17. satırdaki `port_erisim()` fonksiyonu `ioperm()` fonksiyonunu çağırıp erişim iznini denetler. `port_erisim()`, aşağıda verilen `linuxPort.h` dosyası içinde tanımlanmıştır. `linuxPort.h` bütün Linux tabanlı işletim sistemlerinde bulunan gcc derleyici ile kullanılabilir.

```
01: /************************************************
02: ********************
03: **
04: **
05: ** linuxPort.h
06: **
07: **
08: **
09: ** Linux işletim sisteminde kullanılabilecek port
10: fonksiyonları     **
11: **
12: **
13: ** Oluşturma tarihi: Haziran 2006
14: **
15: ** Enson güncelleme: Kasım   2008
16: **
17: **
18: **
19: ************************************************
20: ******************/
21:
22:
23: //
24: // Standart PC için Port adresleri
25: ------------------------------------
26: //
27: #define DATA     0x378
28: #define STATUS   0x379
29: #define CONTROL  0x37A
30: #define COM1     0x3F8
31: #define COM2     0x2F8
32: #define MCR      0x3FC   // Modem Control Register
33: (8 bit)
```

```
34: #define MSR      0x3FE   // Modem Status  Register
35: (8 bit)
36: #define KEYB1    0x060
37: #define KEYB2    0x064
38:
39: //
40: // port erişim fonksiyonları
41: ----------------------------------------
42: // Bu fonksiyonlar, gcc derleyicisinde <sys/io.h>
43: dosyasında mevcuttur
44: //
45:
46: void outportb(unsigned short port, unsigned char
47: data)
48: {
49:   __asm__ __volatile__ ("outb %1, %0"
50:           :
51:           : "dN" (port),
52:             "a" (data));
53: }
54:
55: void outport(unsigned short port, unsigned short
56: data)
57: {
58:   __asm__ __volatile__ ("outw %1, %0"
59:           :
60:           : "dN" (port),
61:             "a" (data));
62: }
63:
64: unsigned char inportb(unsigned short port)
65: {
66:   unsigned char rv;
67:   __asm__ __volatile__ ("inb %1, %0"
68:           : "=a" (rv)
69:           : "dN" (port));
70:   return rv;
71: }
72:
73: unsigned short inport(unsigned short port)
74: {
75:   unsigned short rv;
76:   __asm__ __volatile__ ("inw %1, %0"
77:           : "=a" (rv)
78:           : "dN" (port));
79:   return rv;
80: }
81:
82: //
83: // erisim izinlerini kontrol et
84: ------------------------------------
85: //
86: int port_erisim(void)
87: {
88:    int i, erisim = 0;
89:
90:    /* paralel porta erisim izni var mi? */
```

```
91:     for(i=0; i<=2; i++){
92:       if (ioperm(DATA+i,1,1) )
93:       {
94:         fprintf(stderr, "0x%X adresindeki porta
95:  erisim engellenmistir.\n", DATA+i);
96:         erisim++;
97:       }
98:     }
99:
100:    /* seri porta erisim izni var mi? */
101:    for(i=0; i<=6; i++){
102:      if (ioperm(COM1+i,1,1) )
103:      {
104:        fprintf(stderr, "0x%X adresindeki porta
105:erisim engellenmistir.\n", COM1+i);
106:        erisim++;
107:      }
108:    }
109:
110:    /* klavye portuna erisim izni var mi? */
111:    if (ioperm(KEYB1,1,1) )
112:    {
113:        fprintf(stderr, "0x%X adresindeki porta
114:erisim engellenmistir.\n", KEYB1);
115:        erisim++;
116:    }
117:    if (ioperm(KEYB2,1,1) )
118:    {
            fprintf(stderr, "0x%X adresindeki porta
    erisim engellenmistir.\n", KEYB2);
            erisim++;
       }

       if(erisim) printf("erisim # = %d\n",erisim);

       return erisim;
    }

    //
    // Seri port Haberleşme Ayarları
    -----------------------------------
    //
    void com_ayarlari()
    {
       outportb(COM1 + 1 , 0x00);  // Kesmeleri kapat -
    COM1
       outportb(COM1 + 3 , 0x80);  // DLAB açık
       outportb(COM1 + 0 , 0x03);  // Aktarım hızı -
    Divisor Latch Low  Byte
       outportb(COM1 + 1 , 0x00);  // Aktarım hızı -
    Divisor Latch High Byte
       outportb(COM1 + 3 , 0x03);  // 8 bit, parity yok,
    1 dur biti
       outportb(COM1 + 2 , 0xC0);  // FIFO Kontrol
    saklayıcısı
    }
```