DIGITAL INPUT/OUTPUT CARD

IOD-96M

USER MANUAL

File: MIOD-96M.A2a

NOTICES

TABLE OF CONTENTS

INSTALLING THE CARD

Before installing the card carefully read the ADDRESS SELECTION and OPTION SELECTION Sections of this manual and configure the card according to your require-ments. Use the special software program called **Setup.exe** provided on CD with the card. It supplies visual aids to configure all areas of the board.

Be especially careful with address selection. If the addresses of two installed functions overlap, you will experience unpredictable computer behavior. IOD-96M uses 16 address locations on the I/O bus. If you are unsure about which addresses are available in your computer, use the **FINDBASE** program provided on CD by ACCES to determine which addresses are available.

To install the card:

      1. Remove power from the computer.

      2. Remove the computer cover.

      3. Remove blank I/O backplate.

      4. Install jumpers for selected options. See OPTION SELECTION section of this manual.

      5. Select the base address on the card. See ADDRESS SELECTION section of this manual.

      6. Loosen the nuts on the strain relief bar and swing the top end free.

      7. Install the card in an I/O expansion slot. If convenient, select a slot adjacent to a vacant slot because this will make cable installation easier.

      8. Thread the I/O cables, one by one through the cutout in the mounting bracket and plug them into the headers.

      9. Smooth the cables as close as practicable to the card and, while holding them close to the card surface, swing the strain relief bar into position and tighten nuts.

      10. Inspect for proper fit of the card and cables and tighten screws.

      11. Replace the computer cover.

FUNCTIONAL DESCRIPTION

FEATURES

96 Channels of Digital Input/Output.
Four and Eight Bit Groups Independently Selectable for I/O.
Bi-Directional Data Transfers Supported
Supports PPI Modes 0, 1, and 2.
Interrupt and Interrupt-Disable Capability.
+5V Supply Available to User.
Compatible with Industry Standard I/O Racks like Opto-22, Potter & Brumfield etc.

APPLICATIONS

Automatic Test Systems.
Security Systems, Energy Management.
Robotics
Relay Monitoring and Control.
Parallel Data Transfer to PC.
Sensing switch closures or TTL, DTL, CMOS Logic.
Driving Indicator Lights or Recorders.

The IOD-96M board was designed for industrial applications and should be installed in a long slot of an IBM PC/XT/AT or compatible computer. The board contains four Programmable Peripheral Interface chips type 8255-5 (PPI) to provide computer interface to 96 digital I/O lines. Each PPI provides three 8-bit ports A, B, and C. Each 8-bit port can be configured to function as either inputs or output latches. Port C can also be configured as four input and four output latches. Modes 0, 1, and 2 of the 8255 PPI are supported.

Two I/O lines of each port can be used for interfacing User Interrupts to the computer. Interrupts are enabled by jumper installation or by a combination of jumper installation and a digital input line. You can use Interrupts #2 through #7. Interrupts of all ports (one per port) are OR'ed together.

I/O wiring connections are via 50-pin headers on the board. Five flat I/O cables connect IOD-96E to termination panels such as ACCES model STA-50. Also, this provides compatibility with OPTO-22, Gordos, Potter & Brumfield, etc. module mounting racks. Every second conductor of the flat cables is grounded to minimize the effect of crosstalk between signals. If needed for external circuits +5VDC power is available on each I/O connector pin 49. If you use this power, we recommend that you include a 1A fast blow fuse in your circuits in order to avoid possible damage to the host computer.

The board occupies sixteen consecutive bytes within the I/O address space. The base address is selectable via ADDRESS SETUP DIP switches (A5-A9) anywhere within the hex 200-3F0 rangefor XT's and 100-3F0 (except 1F0 to 1F8) for AT's. Refer to the OPTION SELECTION Section of this manual for a detailed description.

Utility software provided with the IOD-96M card is an illustrated setup program. Interactive displays show locations and proper settings of DIP switches and jumpers to set up board address, interrupt levels, and interrupt enable. Additionally, two sample programs and a utility driver for use with VisualBASIC for Windows are provided. See the Software section of this manual for a detailed description of the latter.

IOD-96M BLOCK DIAGRAM
### (Typical of Four Sections)
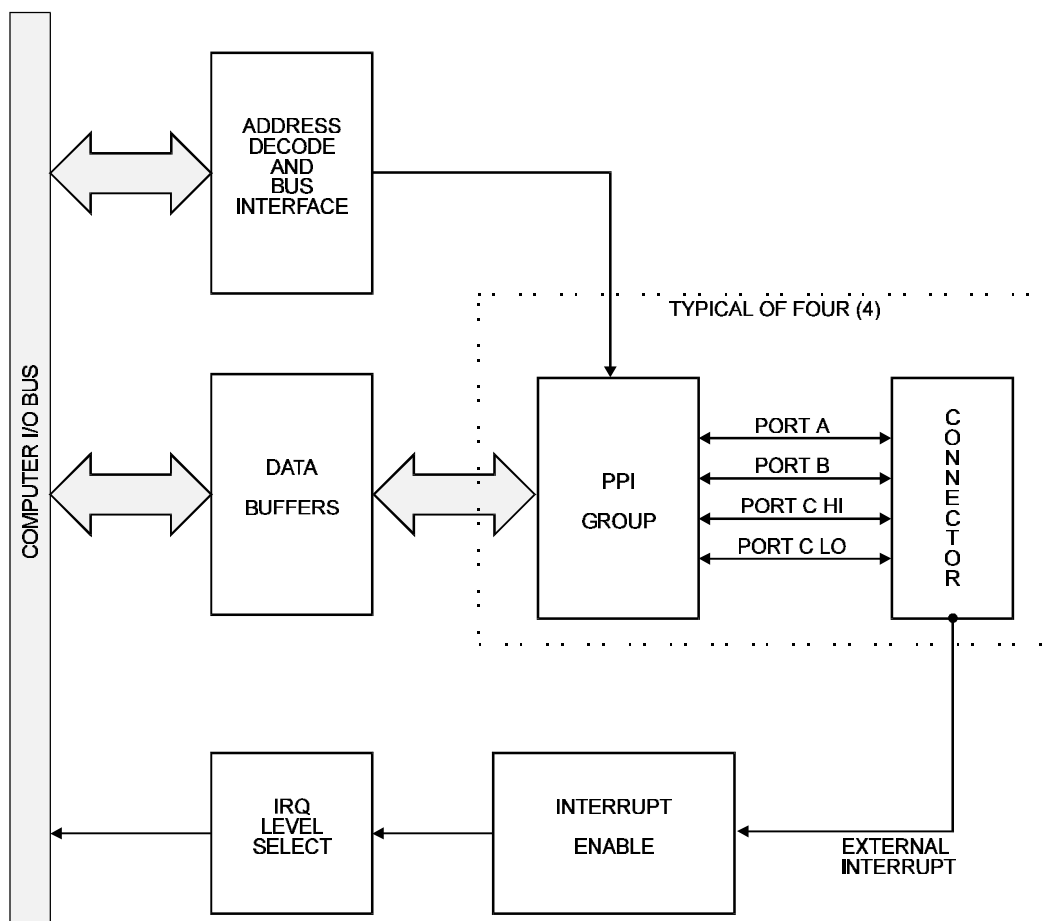
OPTION SELECTION

Refer to the illustrated setup programs on the CD provided with the card when reading this section of the manual. Also, refer to the OPTION SELECTION MAP below. Base address selection is covered both by the CD and in the BASE ADDRESS SELECTION section of this manual.

Interrupts are accepted on the I/O connector, pin 9 (port C3). The interrupt signal is positive true. Interrupts are enabled if I/O connector pin 1 (port C) is held low. Interrupts are disabled if I/O connector pin 1 (port C7) is held high. User interrupts are directed to interrupts #2 through #7 by jumpers marked IN2, 3, 4, 5, 6, 7.

The WAIT jumper should be installed when the host computer I/O bus speed is 8 MHz or more. This will automatically assert one wait state when the card is addressed.

The foregoing are the only manual setups necessary to use the IOD-96E. Input/Output selection is done, via software, by writing to the PPI 8255-5 Control Registers as described in the PROGRAMMING section of this manual.

OPTION SELECTION MAP
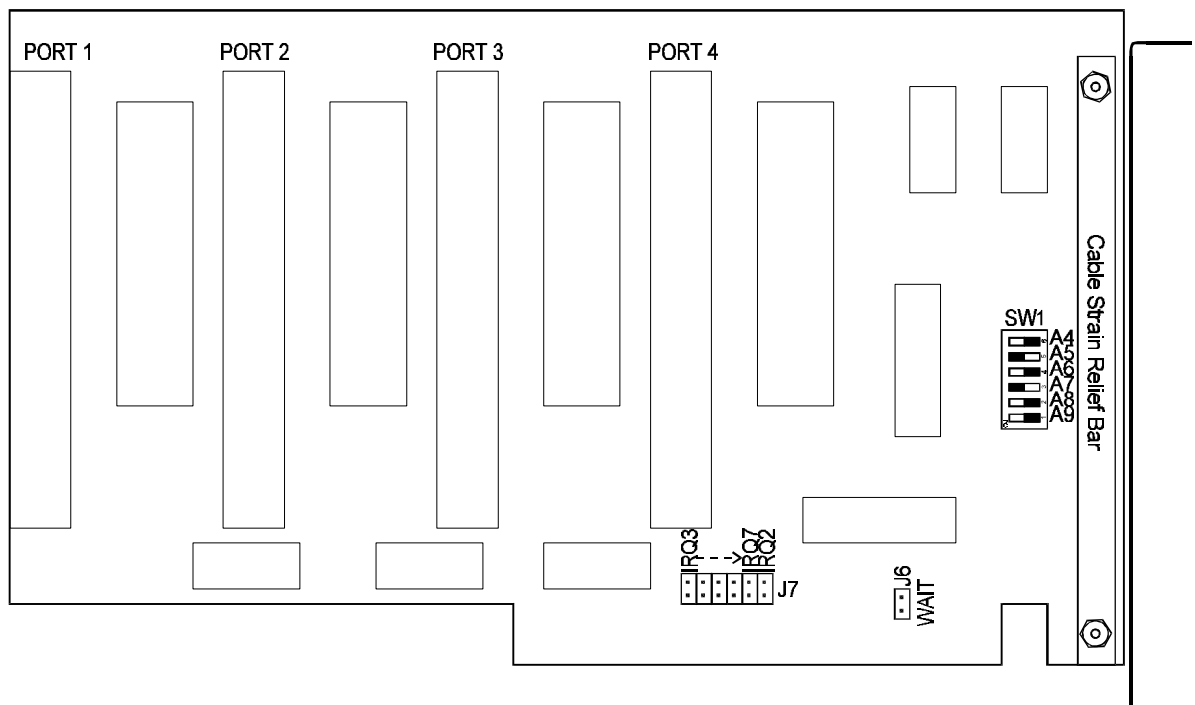
ADDRESS SELECTION

The IOD-96M Input/Output Card occupies 16 bytes of I/O space. The card base address can be selected anywhere within an I/O address range 200-3F0 hex for XTs and 100-3F0 (except 1F0-1F8) for ATs. However two installed options cannot share the same address. If in doubt where to assign the base address of the IOD-96E, refer to the tables below and the FINDBASE utility program to locate an available address for your system.

STANDARD ADDRESS ASSIGNMENTS FOR 286/386/486 COMPUTERS

Hex Range	Usage
000-01F	DMA Controller 1
020-03F	INT Controller 1, Master
040-05F	Timer
060-06F	8042 (Keyboard)
070-07F	Real Time Clock, NMI Mask
080-09F	DMA Page Register
0A0-0BF	INT Controller 2
0C0-0DF	DMA Controller 2
0F0	Clear Math Coprocessor Busy
0F1	Reset Coprocessor
0F8-0FF	Arithmetic Processor
1F0-1F8	Fixed Disk
200-207	Game I/O
278-27F	Parallel Printer Port 2
2F8-2FF	Asynchronous Comm'n (Secondary)
300-31F	Prototype Card
360-36F	Reserved
378-37F	Parallel Printer Port 1
380-38F	SDLC or Binary Synchronous Comm'n 2
3A0-3AF	Binary Synchronous Comm'n 1
3B0-3BF	Monochrome Display/Printer
3C0-3CE	Local Area Network
3D0-3DF	Color/Graphic Monitor
3F0-3F7	Floppy Diskette Controller
3F8-3FF	Asynchronous Comm'n (Primary)

To set desired board address, refer to the illustrated Board Address setup program on the CD provided with the card. Type the desired address in hexadecimal code and the graphic display shows you how to set the ADDRESS SETUP switches. These switches are marked A4-A9 and form a binary representation of the address in negative-true logic. *Assign '0' to all ADDRESS SETUP switches turned ON, and assign '1' to all ADDRESS SETUP switches turned OFF*.

The following example illustrates switch selection corresponding to hex 2D0 (or binary 10 1101 xxxx) The "xxxx" represents address lines A3, A2, A1, and A0 used on the card to select individual registers at the PPI's. See section 5, PROGRAMMING.

HexRepresentation	2		D			
Conversion Multipliers	2	1	8	4	2	1
Binary Representation	1	0	1	1	0	1
Setup	OFF	ON	OFF	OFF	ON	OFF
Switch ID	A9	A8	A7	A6	A5	A4

CAUTION

Review the address selection reference table carefully before selecting the card address. If the addresses of two installed functions overlap you will experience unpredictable computer behavior.

SOFTWARE

ACCES supplies several programs to support the IOD-96M Digital I/O card and, also, to help you develop your applications software. These programs are on a CD that comes with your card and consist of a Setup program and three sample programs. The sample programs are in forms suitable for use with BASIC, QuickBASIC, C, and Pascal. The programs are as follows:

FINDBASE--Locates available I/O bus addresses.

SETUP--Board Setup Program

SAMPLE1--A program that writes a sequence of values to Port A&B reads and displays the values in Ports A & B.

SAMPLE2--A program that displays the bits in Ports A & B and, when an interrupt occurs, polls those same bits.

SETUP

This program is supplied with the IOD-96M card as a tool for you to use in configuring jumpers and switches on the card. It is menu-driven and provides pictures of the card on the computer monitor. You make simple keystrokes to select the functions. In turn, the pictures then change to show how the jumpers or switches should be placed to effect your choices.

The setup program is a stand-alone program that can be run at any time. It does not require the IOD-96M to be plugged into the computer for any part of the setup. The program is self-explanatory with operation instructions and on-line help.

PROGRAMMING

The IOD-96M is an I/O mapped device that is easily configured from any language and any language can easily perform digital I/O through the card's ports. This is especially true if the form of the data is byte or word wide. All references to the I/O ports would be in absolute port addressing. However, a table could be used to convert the byte or word data ports to a logical reference.

DEVELOPING YOUR OWN APPLICATION SOFTWARE

If you wish to gain a better understanding of the programs listed in the previous section, then the information in the following paragraphs will be of interest to you. Follow the 8255-5 Specification in APPENDIX A to program the PPIs on the IOD-96M Digital Input/Output Card.

A total of 16 address locations are used by the IOD-96M for addressing the PPI's; four for each PPI. The PPIs are addressed using address bits A3 through A0 (See Address Selection, section 4 of this manual.) as follows:

CONTROL REGISTER ADDRESS SELECTION TABLE

Address	Port Assignment	Operation
Base Address	PA Port 0	Read/Write
Base Address +1	PB Port 0	Read/Write
Base Address +2	PC Port 0	Read/Write
Base Address +3	Control Port 0	Write Only
Base Address +4	PA Port 1	Read/Write
Base Address +5	PB Port 1	Read/Write
Base Address +6	PC Port 1	Read/Write
Base Address +7	Control Port 1	Write Only
Base Address +8	PA Port 2	Read/Write
Base Address +9	PB Port 2	Read/Write
Base Address +A	PC Port 2	Read/Write
Base Address +B	Control Port 2	Write Only
Base Address +C	PA Port 3	Read/Write
Base Address +D	PB Port 3	Read/Write
Base Address +E	PC Port 3	Read/Write
Base Address +F	Control Port 3	Write Only

The IOD-96M uses four 8255-5 PPIs to provide a total of 96 bits input/output capability. The card may be used in Modes 0, 1, or 2.

Mode 0 is the most frequently used mode of operation wherein:

   a. There are two 8-bit ports (A and B) and two 4-bit ports (C HI and C LO).

b. Any port can be configured as an input or an output.
c. Outputs are latched.
d. Inputs are not latched.

Mode 1 is used for strobed input/output. In this mode, Ports A and B use the lines of Port C to generate and accept control signals associated with data transfer. See the PPI specification sheet in Appendix A for control line assignments.

Mode 2 is used for bi-directional 8-bit bus applications. See the PPI soecification sheet in Appendix A and consult factory if you desire this mode of operation.

Each PPI contains a control register. This Write-only, 8-bit register is used to set the mode and direction of the ports. At Power-Up or Reset, all I/O lines are set as inputs. Each PPI should be configured during initialization by writing to the control registers even if the ports are going to be used as inputs. Output buffers are automatically set by hardware logic according to the control register. Control registers are located at base addresses +03, +07, +0B, and +0F. Bit assignments in each of these control registers are as follows:

CONTROL REGISTER BIT ASSIGNMENTS

Bit	Assignment	Function
D0	Port C Lo (C0-C3)	1 = Input, 0 = Output
D1	Port B	1 = Input, 0 = Output
D2	Mode Selection	1 = Mode 1, 0 = Mode 0
D3	Port C Hi (C4-C7)	1 = Input, 0 = Output
D4	Port A	1 = Input, 0 = Output
D5,D6	Mode Selection	01 = Mode 1, 00 = Mode 0
		1X = Mode 2
D7	Mode Set Flag	1 = Active

NOTE: Contact ACCES for directions if you wish to operate this card in mode 1 or mode 2.

PROGRAMMING EXAMPLE

The following programming example is provided as a guide to assist you in developing your working software. In this example, the card base address is 2D0 hex and I/O lines of Port 0 are to be setup as follows:

port A	= Input	port C hi = Input
port B	= Output	port C lo = Output

Configure bits of the Control Register as:

```
D7 D6   D5    D4    D3    D2    D1    D0
1  0    0     1     1     0     0     0
|  |    |     |     |     |     |     |
|  |    |     |     |     |     |     Port C Lo = output
|  |    |     |     |     |     |
|  |    |     |     |     |     |_____ Port B = output
|  |    |     |     |     |     |
|  |    |     |     |     |_____ Mode 0
|  |    |     |     |     |
|  |    |     |     |_____ Port C Hi = input
|  |    |     |     |
|  |    |     |_____ Port A = input
|  |    |     |
|  |    |_____ Mode 0
|  |    |
|  |_____Mode 0
|
|_____ Active Mode Set
```

This corresponds to 98 hex. If the card base address is 2D0 hex, use the BASIC OUT command to write to the control register as follows:

```
10BASEADDR=&H2D0
20OUT BASEADDR+3,&H98
```

To read the inputs at Port A and the upper nybble of Port C, use the BASIC INPUT command:

```
30X=INP(BASEADDR)                 'Read Port A
40Y=INP(BASEADDR+2)/16            'Read Port C Hi
```

To set outputs high ("1") at Port B and the lower nybble of Port C:

```
50    OUT BASEADDR+1,&HFF         Turn on all Port B bits
60    OUT BASEADDR+2,&HF          'Turn on all bits of  Port C Lo
```

SAMPLE PROGRAMS

The following sample programs are in TURBO-C and TURBO-PASCAL languages. They cover a security system that allows you to monitor the status of 16 switches and to automatically trigger four alarms that can be used to turn on lights, activate a siren, or send a signal to a silent alarm. The alarm system in this demonstration has four arming stations which toggle the alarm on or off. These programs are also provided on the CD that ACCES supplied with your IOD card.

TURBO-PASCAL PROGRAM

```
CONST BASEADDR = $300;          {declare base address for IOD card}
CONST ON       = 1;             {declare some useful constants}
CONST OFF      = 0;             {    "      "    "       "       }

TYPE sensor_array = array[0..15] of integer;
                                {creates a type of variable used for sensor data}
VAR sensors_at_arm : sensor_array;
                                {bit-by-bit status of sensors when alarm is activat-
                                ed.  Used to notify user of open windows, etc}
VAR sensors_now : sensor_now;
                                {bit-by-bit status of sensors at current time.  When}
                                {compared  against  sensors_at_arm,  indicates}
                                {break-in if there is a change.}
VAR arming_stations : integer;   {variables representing all four arming stations. If}
VAR old_arming_stations: integer;  {value changes toggle alarm on/off}
VAR hour, min., sec., hun. : word;  {variables used to retrieve time}
VAR key : char;                 {useful temporary variable}
VAR i : integer;                {useful temporary variable, used in loops}
VAR j : integer;                {    "        "      "       "      "      "  }

procedure initialize_board;     {this procedure sets MODE 0 as active  and sets}
                                {Port A, B, and C LO as input and Port C HI as}
                                {output}
begin
  port[BASEADDR+3] :=$93;       {port[X] is Pascal's method of accessing the port}
                                {memory.   This code sets the port memory at}
                                {address 303 hex, the control register, and to 93}
                                {hex because the bit pattern to set the desired}
                                {mode and port designations is} 10010011 which}
                                {equals 93 hex}

  end; {procedure initialize_board}
```

```
procedure read_sensors(VAR ary:sensor_ary);
  VAR tempA : byte;                    {this procedure fetches data from Ports A and B}
VAR tempB: byte;                      {and  returns a binary representation of each sensor}
begin
  tempA := port[BASEADDR];           {this procedure loads tempA and tempB with corre
                                  tempB:=port[BASEADDR+1]:        {sponding inputs
from the IOD card}
  for i:= 0 to 7 do begin
    if ((tempA shr i) AND ON) > 0 then {this tests to see if bit #i is ON and sets the}
      ary[i]:=ON                              {corresponding array element to ON if it is}
        else                                 {.. else, the array element is set to OFF}
          ary[i]:=OFF
  end;
  for i:=0 to 7 do begin
    if ((tempB shr i) AND ON) > 0 then
      ary[i+8]:=ON                      {n order to get Port B into array, elements 8}
        else                            { thru 15, we add 8 to the bit numbers in the}
          ary[i+8]:=OFF;                        {assignment}
        end;
end;{procedure read_sensors}


function get_status:integer;
      var temp:integer;
begin                                          {this sets status to the lower nybble of Port}
      temp:=port[BASEADDR+2];                  {C;the half defined by Initialize to be input for}
      get_status:=temp AND $0F;                {four arming switches}
end; {function get_arming_status


procedure ALARM
      var temp:longint;
    begin
      sound(2000);                             {this starts the computer's speaker which}
                                               {acts as siren for the alarm}

      temp:=0
      port[BASEADDR+2]:=$F;                    {this sets Port C's lower nybble bits to ON}
      repeat
        arming_stations:=get_status            {this activates four alarm outputs and then}
        if arming_stations <> old_arming_stations then     {toggles Port C hi's LSB which}
          temp:=2000000000; {disarmed}               {might be used with external siren}
        port[BASEADDR+2]:=port[BASEADDR+2] XOR $10;
        temp:=temp+1;
      until temp>=2000000000;
      nosound;
    end; {procedure ALARM}
```

```
begin
 initialize_board;
 clrscr;
 gotoxy(5,5);
 writeln('This is the IOD card demonstration program.  This ');
 writeln('program will simulate an alarm system program for ');
 writeln('sixteen sensors and four arming stations, along with');
 writeln('four separate alarm outputs which could be routed to');
 writeln('a siren, lights, silent alarm,etc');
 writeln;


 writeln('THIS PROGRAM IS INTENDED FOR DEMONSTRATION PURPOSES,');
 writeln('ONLY AND IS NOT MEANT TO BE USED AS AN ACTUAL ALARM ');
 writeln('SYSTEM.');
 writeln;writeln;
 writeln('Press any key to begin program.');
 key:readkey;
 old_arming_stations:=get_status;        {this loads the status of the arming sitches}
 repeat                                  {at the time the program is first activated.}
  clrscr;                                {A change in status indicates arming)

  read_sensors(sensors_now);            {this reads the current status of the sensors}
  for i=0 to 15 do begin                {which is then displayed to indicate open}
    if sensors_now[i]=OFF then          {windows, etc.}
      writeln('Sensor #',i,'is open');
  end;
  writeln;
  writeln('Press ESC to re-scan, RETURN to begin alarm scanning.');
  key:=readkey;
 until key=#13;                         {the repeat/until loop gives the user  an oppor-}
                                        {tunity to shut open windows or doors, and then}
                                        {re-scan the sensors}
 clrscr;

WHILE TRUE do begin                     {this WHILEis used to form an infinite loop}

 Writeln('Waiting to be armed, or press any key to halt program.');

 repeat                                 {this repeat/until-loop continues until arming}
   arming_stations:=get_status;         {station status changes indicating arming or }
   if key pressed then halt(1);         {until a key is pressed indicating program}
                                        {termination}
```

```
until arming stations <> old arming stations;
sound(900);                              {short tone indicating that alarm has been armed}
delay(300);                              {   "     "        "      "     "     "    "    " }
nosound;                                 { "       "        "      "      "    "    "    " }
writeln('Alarm system will activate in 15 seconds');
read_sensors(sensors_at_arm);
old_arming_stations : get_status;
gettime(hour,min,sec,hun);               {this code reads the system clock for the current}
    i:=sec+15;                           {time which is used to delay for 15 seconds}
 if i > 60 then i :=i-60;
    repeat
      gettime(hour,min,sec,hun);
    until sec = i;                       {end of delay loop}
    writeln;
    writeln('ALARM SYSTEM ACTIVE AND ARMED');
     sound(900);                         {short tone indicating that alarm is fully activated}
     delay(300);
     no sound;
     j:=0                                {the following code compares current status of sen-
                                         {sors against status when armed to determine if}
                                         {break-in has occurred..any change indicates}
                                         {break-in}

     repeat
      read_sensors(sensors_now);
      for i:= 1 to 16 do begin
        if sensors_now[i-1] <> sensors_at_arm[i-1]then
          j:=1;
      end;{for}
      arming_stations: get_status;
      if arming_stations <> old_arming_stations then
        j:= -i;                          {flag used to signal that alarm is de-activated}
     until j <> 0;
     if j = -1 then begin        {j was set to -1 in the above loop, then alarm is} {-
                                 de-activated}
      gettime(hour,min,sec,hun);
      writeln('Alarm deactivated at ', hour,':',min,':',sec);

      sound(900);                        {the following code chirps the speaker to indicate}
                                         {disarming}
      delay(100);
      no sound;
      delay(50);
      sound(900);
      delay(100);
```

```
        nosound;
  end                                {end of disarming routine}
  else {if alarm}begin
    writeln('Sensor #', j,' has been activated!!');
    gettime(hour,min,sec,hun);
    writeln('The time of alarm is ',hour,':',min,':',sec);
    ALARM;
  end;                               {else}

  end;                               {WHILE  this "end" sends the program back to wait}
                                     {to be re-armed}

  end.
```

<u>TURBO-C PROGRAM</u>

```
#define BASEADDR 0x300        /*declare base address for IOD card*/
#define ON      1             /*create useful constant*/
#define OFF     0             /*  "    "      "    */
#include "stdio.h"
#include "conio.h"
#include "time.h"
#include "dos.h"

int sensors_at_arm[15];
int sensors_now[15];          /*bit-by-bit status of sensors at current time. When*/
                              /*compared against status of sensors at arm,*/
                              /*indicates break-in if there is a change.*/


int arming_stations;          /*variables representing all four arming stations.If*/
int_old_arming_stations;      /*the value changes,toggle alarm ON/OFF.*/
char key;                     /*useful temporary variable*/
int i;                        /*useful temporary variable used in loops*/
int j;                        /*useful temporary variable*/

initialize(){
  outportb(BASEADDR+3,0x93);  /*outportb(addr,byte) is C's method of accessing
                              /* port memory.  This procedure sets Port A, B,*/
                              /*and  C LO as inputs and Port C hi as outputs */
                              /*Address 303 hex is the control register. The bit*/
                              /*pattern needed to set the desired mode and */
                              /*port designation is 10010011 = 93 Hex*/

} /*procedure initialize*/
read_sensors(int *ary){
```

```
unsigned char tempA;
unsigned char tempB;
  tempA = inportb(BASEADDR);
  tempB = inportb(BASEADDR+1);
  for(i-0;i<8;i++){
   if((tempA>> i) & ON){                      /*this determines if bit #i is on and sets the corres-*/
     *ary++=ON;}                     /*ponding array element to ON if it is. If not, sets */
    else{                                      /*the array element to OFF  */
     *ary++=OFF;}
  }
  for(i=0;i<8;i++){
   if((tempB>> i) & ON){
     *ary++=ON; }
    else
     *ary++=OFF;}
  }
} /*procedure read_sensors*/
get_status(){
  int temp;
  temp=inportb(BASEADDR+2);        /*this sets status to the lower half of Port C, the */
                                    /*the half defined in Initialize to be input, for four */
                                    /*four arming switches*/

  return temp & 0x0F;
} /*function get_arming_status*/
ALARM(){
long int temp=0;
  sound(2000);                      /*this starts the computer's speaker*/
  outportb(BASEADDR+@,0xF0);        /*this sets Port C upper nybble bits to ON (1111 */
                                    /*binary = F hex).*/
  do{
   arming_stations=get_status();       /*this activates 4 alarm outputs and then togles */
   if(arming_stations !=old_arming_stations)  /*Port C  Hi LSB which might be used  */
    temp=2000000000; /*dis-armed*/          /*with an external speaker*/
   outportb(BASEADDR+2,inportb(BASEADDR+2)^0x10);
  }while(temp++ !=2000000000);
  nosound();
} /*procedure ALARM*/
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
main()
{
time_t start;
  initialize();
  clscr();
  goto(5,5);
```

```c
printf("This IOD-card demonstration program simulates an alarm\n");
printf("system program for 16 sensors, four arming stations and\n");
printf("four separate alarm outputs which could be routed to a\n");
printf("siren, lights, silent alarm, etc.\n");
printf("\n");
printf("THIS PROGRAM IS FOR DEMONSTRATION PURPOSES ONLY, AND IS\n");
printf("NOT MEANT TO BE USED AS AN ACTUAL ALARM SYSTEM.\n");
printf("\n");printf("n");
printf("Press any key to begin program.\n");
key=getch();
old_arming_stations=get_status();
do{
  clrscr();
  read_sensors(sensors_now);
  for(i=0;i<=15;i++){
    if (!sensors_now[i]) printf("Sensor #%d %s\n,i,"is open");
  }
  printf("\n");
  printf("Press ESC to re-scan, RETURN to begin alarm scanning.");
  key=getch();
}while(key!=13);
clrscr();
for(;;)                              {/*this creates an infinite loop*/
printf("Waiting to be armed.  Press any key to halt program.\n");
do{
  arming_stations=get_status();
  if(kbhit()) abort(0);
}while(arming_stations== old_arming_stations);
sound(1000);
delay(300);
nosound();
printf("Alarm system will activate in 15 seconds");
read_sensors(sensors_at_arm);
old_arming_stations=get_status();
start=time(NULL);
do{
}while(difftime(time(NULL),start) !=15);

printf("\n");
printf("ALARM SYSTEM ACTIVE AND ARMED\n\n");
sound(900);
delay(300);
nosound();
j=0;
```

```
 do{
  read_sensors(sensors_now);
  for(i=1;i<=16;i++){
    if(sensors_now[i-1] !=sensors_at_arm[i-1])
  j=i;
  }                                            /*for*/
  arming_stations = get_status();
  if (arming_stations != old_arming_stations)
 j=-1                                   /*flag used to signal alarm is de-activated*/
 while(!j);
 if(j == -1){
  start=time(NULL);
  printf("Alarm deactivated at %s,(asctime(gmtime(&start))));
  sound(900); delay(300);
  nosound(); delay(50);
  sound(900); delay(100);
  nosound();
 }
 else {
  printf("Sensor #%d has been activated!!\n\n",j);
  start=time(NULL);
  printf("The time of alarm is %s", asctime(gmtime( &start)));
  old_arming_stations=get_status();
  ALARM();
 }                                        /*else*/
}                                   /* for(;;) this "end" used to send program back*/
                                         /*to await re-arm*/

}
```

CONNECTOR PIN ASSIGNMENTS

Four 50-pin headers are provided on the IOD-96M card; one for each group of 24 I/O lines. The mating connector is an AMP type 1-746285-0 or equivalent. Connector pin assignments are listed below. Notice that every second line is grounded to minimize crosstalk between signals.

PIN	BIT ASSIGNMENT	PIN	BIT ASSIGNMENT
1	PA7	27	PC2
3	PA6	29	PC1
5	PA5	31	PC0
7	PA4	33	PB7
9	PA3	35	PB6
11	PA2	37	PB5
13	PA1	39	PB4
15	PA0	41	PB3
17	PC7 *	43	PB2
19	PC6	45	PB1
21	PC5	47	PB0
23	PC4	49	+5 VDC
25	PC3 **	2 THRU 50	Digital Common

Notes:
* This line is an I/O port and also an Interrupt Enable.
** This line is an I/O port and also a User Interrupt.

SPECIFICATION

Features

    96 Channels of Digital Input/Output.
    Four and Eight Bit Groups Independently Selectable for I/O.
    Supports PPI Modes 1, 2, and 3.
    Interrupt and Interrupt-Disable Capability.
    +5V Supply Available to User.
    Compatible with Industry Standard I/O Racks like Opto-22, Potter & Brumfield, etc.

Digital Inputs

    Logic High: 2.0 to 5.0 VDC.
    Logic Low: -0.5 to +0.8 VDC.
    Input Load (Hi): +10 uA.
    Input Load (Lo): -10 uA.

Digital Outputs

    Logic High: 2.5 VDC min., source 200 uA
    Logic Low: 0.5 VDC max., sink 1.7 mA

Power Output: +5 VDC from computer bus (ext. 1A fast blow fuse recommended).

Power Requirements: +5 VDC at 400 mA typical.

Size: 7" long

Environmental

Operating Temperature Range: 0 degr. to 60 degr.C.
Storage Temperature Range: -50 degr. to +120 degr.C.
Humidity: 0 to 90% RH, non-condensing.

WARRANTY

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

TERMS AND CONDITIONS

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

COVERAGE

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

EQUIPMENT NOT MANUFACTURED BY ACCES

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

GENERAL

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arriving from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

APPENDIX A

PPI DATA SHEETS

The data sheets in this Appendix are provided to help your understanding of the 8255-5 PPI which is manufactured by several companies. These data sheets are reprinted with permission of Mitsubishi Electric Corporation. (Copyright 1987).

The information, diagrams, and all other data included are believed to be correct and reliable. However, no responsibility is assumed by ACCES or Mitsubishi Electric Corp for their use, nor for any infringements of patents or other rights belonging to other parties which may result from their use. Values shown on these data sheets are subject to change for product improvement.