Table of Contents

1. Introduction

The A823.lib is a collection of data acquisition subroutines for A-823PGH/PGL. These subroutines are written with C language and perform a variety of data acquisition operations. These subroutines can be classified as follow:

- Digital I/O
- D/A conversion
- A/D conversion via Polling
- A/D conversion via Interrupt
- A/D conversion via DMA
- Initialization
- Pacer Trigger control
- Machine independent timer

The subroutines in A823.lib are easy understanding as its name standing for. It provides powerful, easy-to-use subroutine for developing your data acquisition application. To speed-up your developing process, some demonstration C source program are provided.

This driver can support 8 cards maximum in one PC based system. But only can support 2 interrupt/DMA boards at most and the other board will be running in polling mode. So the validate configuration of using 8 boards may be as follows:

- **8 polling**
- **7 polling + 1 DMA**
- **7 polling + 1 interrupt**
- **6 polling + 2 DMA**
- **6 polling + 2 interrupt**
- **6 polling + 1 DMA + 1 interrupt**

- **Support 8 cards in one system maximum**
- **Only 2 cards can use DMA/interrupt at most**

2. Installation

It is recommended to install the A-823PHL/PGH application software to your hard disk to get the best performance. Before beginning, to make a backup copy of the A-823PGH/PGL application software. Store the original diskette in a safe place. The A-823PGL/PGH application disk includes the following files:

- A823\A823DIAG.EXE : The diagnostic utility for 823PGL/PGH.
- A823\A823.CFG : The configuration file of A823DIAG.EXE.

- A823\lib\A823.h : The header file for MSC/TC.
- A823\lib\SA823.lib : The small model library for MSC/TC.
- A823\lib\MA823.lib : The medium model library for MSC/TC.
- A823\lib\CA823.lib : The compact model library for MSC/TC.
- A823\lib\LA823.lib : The large model library for MSC/TC.
- A823\lib\HA823.lib : The huge model library for MSC/TC.

- A823\demo\msc*.* : The demo program for MSC 5.X
- A823\demo\tc*.* : The demo program for TC 2.X
- A823\demo\bc*.* : The demo program for BC 3.X

- A823\demo\msc*.c : The source files of demo program
- A823\demo\msc*.exe : The execution files of demo program
- A823\demo\msc*.bat : The compiler&link batch files of demo program

- A823\demo\tc*.c : The source files of demo program
- A823\demo\tc*.exe : The execution files of demo program.
- A823\demo\tc*.prj : The compiler&link project files of demo program

- A823\demo\bc*.c : The source files of demo program
- A823\demo\bc*.exe : The execution files of demo program
- A823\demo\bc*.prj : The compiler&link project files of demo program

- A823\demo\tc\testpoll.c : The demo program for A/D Polling testing
- A823\demo\tc\testint.c : The demo program for A/D Interrupt testing
- A823\demo\tc\testdma.c : The demo program for A/D DMA testing
- A823\demo\tc\testdio.c : The demo program for digital I/O testing
- A823\demo\tc\testda.c : The demo program for D/A testing
- A823\demo\tc\testad.c : The demo program for A/D Polling/INT/DMA testing
- A823\demo\tc\gain1.c : Delay the gain settling time for A823PGH
- A823\demo\tc\gain2.c : Delay the gain settling time for A823PGL
- A823\demo\tc\polling.c : Performance of A/D polling mode
- A823\demo\tc\interrup.c : Performance of A/D interrupt mode
- A823\demo\tc\dma.c : Performance of A/D DMA mode
- A823\demo\tc\pollwave.c : Show wave form by polling mode
- A823\demo\tc\intwave.c : Show wave form by interrupt mode
- A823\demo\tc\dmawave.c : Show wave form by DMA mode

- **A823\demo\msc*.c : only program1 ~ program6 are supported**

- **A823\demo\bc*.c : only program1 ~ program6 are supported**

2.1 Compiler & link using MSC

- The including file is **A823.H**
- There are 5 different mode library files : **S/C/M/L/HA823.LIB**
- Support MSC 6.x compiler
- **SMALL** model compiler & link command : CL /AS program.c SA823.LIB
- **COMPACT** model compiler & link command : CL /AC program.c CA823.LIB
- **MEDIUM** model compiler & link command : CL /AM program.c MA823.LIB
- **LARGE** model compiler & link command : CL /AL program.c LA823.LIB
- **HUGE** model compiler & link command : CL /AH program.c HA823.LIB
- A:\A823\demo\msc*.bat give some examples for compiler&link batch file

2.2 Compiler & link using TC

- The including file is **A823.H**
- There are 5 different model library files : **S/C/M/L/HA823.LIB**
- Support TC 2.x compiler
- Use text editor to create a project file include : program.c ?A823.lib
- Use TC integrated environment to **select the correct compiler model**
- A:\A823\demo\tc*.prj give some examples for compiler&link project file

2.3 Compiler & link using BC

- The including file is **A823.H**
- There are 5 different model library files : **S/C/M/L/HA823.LIB**
- Support BC 3.x compiler
- Use BC integrated environment to create a project file include : program.c ?A823.lib
- Use BC integrated environment to **select the correct compiler model**
- A:\A823\demo\bc*.prj give some examples for compiler&link project file

3. C Language Library

- A823.H
- A823_Initialize
- A823_ActiveBoard
- A823_Check_Address
- A823_DI
- A823_DO
- A823_DA
- A823_AD_SetChGainMode
- A823_AD_PollingVar
- A823_AD_PollingArray
- A823_AD_INT_Start
- A823_AD_INT_Count
- A823_AD_INT_Stop
- A823_AD_DMA_Start
- A823_AD_DMA_Stop
- A823_AD_DMA_Status
- A823_Delay

3.1 A823.H

```
int A823_Initialize(int CardNo,int BaseAddr,int DmaNo, int IrqNo);
int A823_ActiveBoard(int BoardNo);
int A823_Check_Address(int  BaseAddr);
unsigned int A823_DI( void );
void A823_DO(unsigned DigitOutput);
int A823_DA(int Channel, unsigned int data);
void A823_AD_SetChGainMode(int Channel, int Gain, int Mode);
void A823_AD_PollingArray(unsigned int *Buffer, unsigned int Length);
unsigned int A823_AD_PollingVar(void);
int A823_AD_INT_Start(int* Buffer , unsigned int count, int c1 , int
c2);
unsigned A823_AD_INT_Count(void);
void A823_AD_INT_Stop(void);
int A823_AD_DMA_Start(int far *Buffer ,unsigned int count,int c1 ,int
c2);
void A823_AD_DMA_Stop(void);
int A823_AD_DMA_Status(void);
int A823_Delay(unsigned int DownCount);

/***************** DEFINE A823 RELATIVE ADDRESS *****************/
#define   TIMER0       0x00
#define   TIMER1       0x01
#define   TIMER2       0x02
#define   TIMER_MODE          0x03
#define   AD_LO        0x04    /* Analog to Digital, Low Byte */
#define   AD_HI        0x05    /* Analog to Digital, High Byte */
#define   DA_CH0_LO    0x04    /* Digit to Analog, CH 0 */
#define   DA_CH0_HI    0x05
#define   DA_CH1_LO    0x06    /* Digit to Analog, CH 1 */
#define   DA_CH1_HI    0x07
#define   DIGITIN_LO          0x06    /* Digit Input */
#define   DIGITIN_HI          0x07
#define   DIGITOUT_LO  0x0D    /* Digit Output */
#define   DIGITOUT_HI  0x0E

#define   CLEAR_IRQ    0x08
#define   SET_GAIN     0x09
#define   SET_CH       0x0A
#define   SET_MODE     0x0B
#define   SOFT_TRIG    0x0C

#define   POLLING_MODE     1
#define   DMA_MODE         2
#define   INTERRUPT_MODE   6
```

```
/*** define the gain mode ***/
#define   A823_BI_1        0
#define   A823_BI_10             1
#define   A823_BI_100      2
#define   A823_BI_1000           3
#define   A823_UNI_1             4
#define   A823_UNI_10      5
#define   A823_UNI_100           6
#define   A823_UNI_1000          7
#define   A823_BI_05             8
#define   A823_BI_5        9
#define   A823_BI_50             10
#define   A823_BI_500      11

#define   A823_BI_2        1
#define   A823_BI_4        2
#define   A823_BI_8        3
#define   A823_UNI_2             5
#define   A823_UNI_4             6
#define   A823_UNI_8             7

/*** define the error number ***/
#define   NoError                0
#define   CheckBoardError  1
#define   CheckDmaError    2
#define   CheckIrqError    3
#define   CardNumError     4
#define   DAchannelError         5
#define   DelayError       6
```

3.2 A823_Initialize

- **Description :**

A823_Initialize initialize the 823PGL/PGH board. This function should be called before using the other A823.lib subroutines. This function will detect automatically 823PGL/PGH board according to I/O base address. Auto detection failure will occur if I/O base address not match with hardware DIP switch. **This driver supports 8 cards max. But only 2 cards can perform A/D operation through DMA or Interrupt.** The others can perform A/D operation via polling. After this subroutine is executed, **the board which CardNo specified will be active**. If more than one board in a system, the A823_ActiveBoard(int BoardNo) must use to switch to the active board.

- **Syntax :**

int A823_Initialize(int CradNo, int BaseAddr, int DmaNo, int IrqNo);

- **Input Parameter :**

CardNo : The validate card number is from 0 to 7.

IOBase : I/O base address. (must match with hardware DIP switch)

DmaNo : DMA channel number. (1 or 3 must match with hardware jumper)

-1 = NO DMA

IrqNo : Intterrupt channel number. (must match with hardware jumper)

-1= NO Interrupt

- **Return Value :**

CheckBoardError : IO base address detection error

CardNumError : error in card number input (validate numbers are from 0 to 7)

CheckDmaError : error in DMA channel number (validate numbers are 1 or 3)

CheckIrqError : error in IRQ channel number

NoError

- **Demo program :** Sec. 4.1 ~ Sec. 4.14

● Example 1 :

```
#include  "A823.h"
main()
{
int  CradNo=0;              /* only one card */
int  IOBase=0x220;        /* The IO base address for A823 */
int  DmaNo=1;        /* The DMA channel no. */
int  IrqNo=15;        /* The IRQ channel no.  */

A823_Initialize(CardNo,IOBase,DmaNo,IrqNo);  /* board_0 is active */
/* only one card in this system, so no need to call A823_ActiveBoard(0) */
            .
            .
}
```

● Example 2 :

```
#include  "A823.h"
main()
{
A823_Initialize(0,0x200,1,12);      /* now board_0 is active*/
A823_Initialize(1,0x210,3,15);      /* now board_1 is active*/
A823_Initialize(2,0x220,-1,-1);     /* now board_2 is active*/
/* now there are 3 cards in this system, so A823_ActiveBoard(?) need to be called */

A823_ActiveBoard(0);   /* now card_0 is active */


A823_ActiveBoard(1);   /* now card_1 is active */


A823_ActiveBoard(2);   /* now card_2 is active */

}
```

3.3 A823_ActiveBoard

- **Description :**

This driver support 8 different cards in one system max. The A823_Initialize(…) will initialize the separate boards. But only one board can be active at one time. So the software should active the selected card before perform specific function. The A823_ActiveCrad(?) is used to active the desired card in the system. If only one card in the system, there is no need to use this subroutine.

- **Syntax :**

int A823_ActiveBoard(int BoardNo);

- **Input Parameter :**

BoardNo : The validate board number is from 0 to 7. This number is equal to CardNo in A823_Initial(CardNo,…..);

- **Return Value :**

CardNumError : error in card number input (validate numbersare from 0 to 7)
NoError

- **Example 1 :**

See sec. 4.6

3.4 A823_Check_Address

- **Description :**

This subroutine check if the I/O base address is match with the board. This subroutine send a software trigger signal to the A/D converter and check the ready bit of A/D conversion. If the **ready bit can not be clear to zero in a fixed time, this subroutine will return CheckBoardError.**

- **Syntax :**

int A823_Check_Address(int BaseAddr);

- **Input Parameter :**

BaseAddr : base address of the board

- **Return Value :**

CheckBoardError : The BaseAddr does not match with the hardware setting

NoError

- **Demo program :** Sec. 4.1 ~ Sec. 4.14

- **Example 1 :**

```
#include  "A823.h"
main()
{
if (A823_Check_Address(0x220)==NoError)
      {
      printf("\n0x220 find a card');
      }
else
      {
      printf("\n0x220 cannot find any card");
      }
}
```

3.5 A823_DI

- **Description :**

  This function read the digital input data from I/O port Base+6 and Base+7.

- **Syntax :**

  unsigned int A823_DI(void);

- **Input Parameter :**

  None.

- **Return Value :**

  A 16 bit data from digit input port.

- **Demo program :** Sec. 4.4

- **Example 1 :**

```
#include  "A823.H"
main()
{
unsigned DigitInput;

A823_Initialize(0,0x220,1,15);
            .
            .
DigitInput=A823_DI();
            .
}
```

3.6 A823_DO

- **Description :**

  This function write data to I/O port Base+0x0D and Base+0X0E.

- **Syntax :**

  void A823_DO(unsigned DigitalOutput);

- **Input Parameter :**

  DigitalOutput: data send to output port, 0 through 65535.

- **Return Value :**

  None.

- **Demo program :** Sec. 4.4

- **Example 1 :**

```
#include  "A823.H"
main()
{
unsigned DigitOutput=0x1234;

A823_Initialize(0,0x220,1,15);
            .
            .
A823_DO(DigitOutput);
            .
            .
            .
}
```

3.7 A823_DA

- **Description :**

  This function is used to writes 12 bits data to D/A output latch register.

- **Syntax :**

  int A823_DA(int Channel, unsigned int data);

- **Input Parameter :**

  Channel : D/A channel number, 0 = channel 0, 1 = channel 1.

  data : 0 through 4095.

- **Return Value :**

  DAchannelError : validate Channel is only 0/1

  NoError

- **Demo program :** Sec. 4.5

- **Example 1 :**

```
#include  "A823.h"
main()
{
usigned DAOutput=0x1234;

A823_Initialize(0,0x220,15,1);
            .
A823_DA(0,DAOutput);        /* D/A channel 0 */
A823_DA(1,DAOutput);        /* D/A channel 1 */
            .
}
```

3.8 A823_AD_SetChGainMode

- **Description :**

This function is used to set the A/D channel number, gain and operation mode. The detail information is given in “A823 Hardware Manual” section 2.7.

- **Syntax :**

void A823_AD_SetChGainMode(int Channel, int Gain, int Mode)

- **Input Parameter :**

Channel : A/D channel number, 0 ~ 15 for single-ended, 0 ~ 7 for differential
Gain : A/D gain control code, 0 ~ 11 for A823PGH, 0 ~ 8 for A823PGL
Mode : A/D operation mode, 0 through 7

- **Return Value :**

None

- **Demo program :** Sec. 4.1 ~ sec. 4.14

- **Example 1 :**

```
#include  "A823.h"
main()
{
int  Gain,Mode;

A823_Initialize(0,0x220,1,15);

/* user must define Gain, Mode here */

for   (channel=0; channel<16; channel++)
      {
      A823_AD_SetChGainMode(channel,Gain,Mode);
      /* delay settling time if needed */.
      }
            .
}
```

3.9 A823_AD_PollingVar

- **Description :**

  This function performs the A/D conversion by polling .

- **Syntax :**

  unsigned A823_AD_PollingVar(void)

- **Input Parameter :**

  None.

- **Return Value :**

  The result of A/D conversion.

  If there is a timeout occurred, the data will be set to 0xffff.

- **Demo program :** Sec. 4.1/4.7/4.8/4.9/4.12

- **Example 1 :**

```
#include  "A823.h"
#include  "stdio.h"
main()
{
int channel,gain,mode;
unsigned  PollData;
float     volt;
A823_Initialize(0,0x220,1,15);
channel=0;      /* channel 0 */
gain=0;         /* bipolar, gain=1, range=-5V ~ +5V */
mode=1;         /* polling mode */
A823_AD_SetChGainMode(channel,gain,mode);
              .
PollData=A823_AD_PollingVar();
volt=((float)PollData-2048.0)/2048.0*5.0;
printf("\ndata = %xH = %5.3fV\n",PollData,volt);
              .
}
```

3.10 A823_AD_PollingArray

- **Description :**

  This function performs the A/D conversion by polling method.

- **Syntax :**

  void A823_AD_PollingArray(unsigned *Buffer, unsigned Count)

- **Input Parameter :**

  Buffer : address of buffer.

  Count : number of A/D conversions .

- **Return Value :**

  None.

  If there is a timeout occurred, the data will be set to 0xffff.

- **Demo program :** Sec. 4.1/4.7/4.8/4.9/4.12

- **Example 1 :**

```
#include  "A823.h"
#include  "stdio.h"
main()
{
int channel,gain,mode;
unsigned int  i,Buffer[1000],DesireCount=1000;
A823_Initialize(0,0x220,1,15);
channel=0;      /* channel 0 */
gain=0;         /* bipolar, gain=1, range=-5V ~ +5V */
mode=1;         /* polling mode */
A823_AD_SetChGainMode(channel,gain,mode);
            .
A823_AD_PollingArray(Buffer, DesireCount);

for (i=0; i<10; i++) printf("\nBuffer[%d]=%x",i,Buffer[i]);
            .
}
```

3.11 A823_AD_INT_Start

- **Description :**

Transfer A/D conversion data by interrupt. The sampling rate of pacer trigger is 2M/(c1*c2) Hz. The detail information is given in "A823 Hardware Manual" section 2.7.

- **Syntax :**

int A823_AD_INT_Start(int *Buffer, unsigned int Count , int c1 , int c2)

- **Input Parameter :**

Buffer : address of Buffer store A/D conversion data.
Count: number of A/D conversion to perform.
c1 : 16 bits divisor in Timer/Counter 1.
c2 : 16 bits divisor in Timer/Counter 2.

- **Return Value :**

CheckIrqError : IRQ channel number error
NoError : no error

- **Example 1 :**

See sec.4.2/4.10/4.13

- NOTE : The C1 & C2 has some constraints :
  1. 1 is invalidate
  2. 3 is invalidate
  3. 2000/(C1*C2) can not over 110

If C1=4 and C2=5, pacer timer=2000/(4*5)=100K→ validate
if C1=2 and C2=6, pacer timer=2000/111.1K >110K→ invalidate

The test program in sec. 4.10 & 4.13 can test the A/D performance by interrupt mode. In general, the user can get the A/D performance about 60K in the normal PC AT/486 system. **But this performance is machine dependent**, so the user must test to get this maximum performance.

3.12 A823_AD_INT_Count

- **Description :**

  Return the current A/D conversion count number.

- **Syntax :**

  unsigned A823_AD_INT_Count(void)

- **Input Parameter :**

  None.

- **Return Value :**

  Current count value.

- **Example 1 :**

  See sec.4.2/4.10/4.13

3.13 A823_AD_INT_Stop

- **Description :**

  Stop the operation initiate by A823_AD_INT_Start() function.

- **Syntax :**

  void A823_AD_INT_Stop(void)

- **Input Parameter :**

  None.

- **Return Value :**

  None.

- **Example 1 :**

  See sec.4.2/4.10/4.13

3.14 A823_AD_DMA_Start

- **Description :**

Transfer A/D conversion data by DMA. The sampling rate of pacer trigger is 2M/(c1*c2) Hz. The detail information is given in "A823 Hardware Manual" section 2.7.

- **Prototype :**

int A823_AD_DMA_Start(int far *Buffer, unsigned count , int c1 , int c2)

- **Input Parameter :**

Buffer : address of buffer store A/D conversion data.
Count : number of A/D conversion to perform.
c1 : 16 bits divisor in Timer/Counter 1.
c2 : 16 bits divisor in Timer/Counter 2.

- **Return Value :**

CheckDmaError : DMA channel specified in A_823_Initial(…) is error
NoError

- **Example 1 :**

See sec.4.3/4.11/4.14

- NOTE : The C1 & C2 has some constraints :
  1. 1 is invalidate
  2. 3 is invalidate
  3. 2000/(C1*C2) can not over 110

If C1=4 and C2=5, pacer timer=2000/(4*5)=100K→ validate
if C1=2 and C2=6, pacer timer=111.1K >110K→ invalidate

The test program in sec. 4.11 & 4.14 can test the A/D performance by DMA mode. In general, the user can get the A/D performance about 100K in the normal PC AT/486 system. **But this performance is machine dependent**, so the user must test to get this maximum performance.

3.15 A823_AD_DMA_Status

- **Description :**

  Return the status of DMA operation.

- **Syntax :**

  int A823_AD_DMA_Status(void)

- **Input Parameter :**

  None.

- **Return Value :**

  0 : A/D DMA operation is finished

  1 : A/D DMA operation is not finished

- **Example 1 :**

  See sec.4.3/4.11/4.14

3.16 A823_AD_DMA_Stop

- **Description :**

  Stop the operation initiate by A823_AD_DMA_Start() function.

- **Syntax :**

  void A823_AD_DMA_Stop(void)

- **Input Parameter :**

  None.

- **Return Value :**

  None.

- **Example 1 :**

  See sec.4.3/4.11/4.14

3.17 A823_Delay

- **Description :**

Use counter 0 to implement a machine independent timer. The detail information is given in “A823 Hardware Manual” section 2.6.

- **Syntax :**

int A823_Delay(unsigned int DownCount)

- **Input Parameter :**

DownCount : from 1 to 0x7fff, 1 count=0.5 us

- **Return Value :**

DelayError : counter_0 hardware function error

NoError

- **Example 1 :**

See sec.4.7/4.8

- **NOTE :**

1. **The software driver assume the JP6 select the internal 2M stable clock. If the JP6 setting error, the A823_Delay() will not function properly.**
2. **Because the counter0 is used by A823_Delay(), the counter0 can not be use as a user timer/counter.**

4. Demo Program

- A823\demo\tc\testpoll.c : The demo program for A/D Polling testing
- A823\demo\tc\testint.c : The demo program for A/D Interrupt testing
- A823\demo\tc\testdma.c : The demo program for A/D DMA testing
- A823\demo\tc\testdio.c : The demo program for digital I/O testing
- A823\demo\tc\testda.c : The demo program for D/A testing
- A823\demo\tc\testad.c : The demo program for A/D Polling/INT/DMA testing
- A823\demo\tc\gain1.c : Delay the gain settling time for A823PGH
- A823\demo\tc\gain2.c : Delay the gain settling time for A823PGL
- A823\demo\tc\polling.c : Performance of A/D polling mode
- A823\demo\tc\interrup.c : Performance of A/D interrupt mode
- A823\demo\tc\dma.c : Performance of A/D DMA mode
- A823\demo\tc\pollwave.c : Show wave form by polling mode
- A823\demo\tc\intwave.c : Show wave form by interrupt mode
- A823\demo\tc\dmawave.c : Show wave form by DMA mode

4.1 TESTPOLL.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **Support 8 boards at most**

```
#include<stdio.h>
#include<conio.h>
#include"A823.h"

int Base=0x220;
int Buf[1000];

void main(void)
{ int err;
  int i;
  float rate=5.0/2048.0;

  err=A823_Initialize(0,Base,-1,-1);
  if(err){
     if(err==CheckBoardError){
        printf("Address 0 Error!\n");
     }
     else if(err==CardNumError){
        printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  printf("\nCard 0 [polling test]\n");
  printf("Using [A823_AD_PollingArray]\n");
  A823_AD_SetChGainMode(0,A823_BI_1,POLLING_MODE);
  A823_AD_PollingArray(Buf,1000);
  for(i=0;i<1000;i+=100){
         printf("[%3d]%6.3f\n",i,rate*(Buf[i]-2048));
  }
  printf("Using [A823_AD_PollingVar]\n");
  Buf[0]=A823_AD_PollingVar();
  printf("PollingVar=%6.3f\n",rate*(Buf[0]-2048));
}
```

4.2 TESTINT.C

- **I/O ADDRESS=0x220, Interrupt=15, NO DMA**
- **Support 2 boards at most**
- **Press [Esc] key to stop this program**
- **Sampling rate=pacer rate=2000/20*20=5K**

```
#include<stdio.h>
#include<conio.h>
#include"A823.h"

int Base=0x220,Irq=15, Buf[1000];
void main(void)
{ int err,i,c1,c2,NowCount,quit=0;
  float rate=5.0/2048.0;

  err=A823_Initialize(0,Base,-1,Irq);
  if(err){
     if(err==CheckBoardError){
       printf("Address 0 Error!\n");
     }
     else if(err==CardNumError){
       printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  printf("\nCard 0 [interrupt test]\n");
  A823_AD_SetChGainMode(0,A823_BI_1,INTERRUPT_MODE);
  c1=20; c2=20;
  while(!quit){
       A823_AD_INT_Start(Buf,1000,c1,c2);
       do {
          NowCount=A823_AD_INT_Count();
          if(kbhit() && getch()==27) {
             quit=1; break;
          }
       } while( NowCount<1000 );
       A823_AD_INT_Stop( );
       if(kbhit() && getch()==27) break;
       for(i=0;i<1000;i+=100){
           printf("[%3d]%6.3f\n",i,rate*(Buf[i]-2048));
       }
```

4.3 TESTDMA.C

- **I/O ADDRESS=0x220, Interrupt=15, DMA=1**
- **Support 2 boards at most**
- **Press [Esc] key to stop this program**
- **Sampling rate=pacer rate=2000/6*6=55.6K**

```
#include<stdio.h>
#include<conio.h>
#include"A823.h"

int Base=0x220,Dma=1,Irq=15,Buf[1000];
void main(void)
{ int err,i,c1,c2,quit=0;
  float rate=5.0/2048.0;

  err=A823_Initialize(0,Base,Dma,Irq);
  if(err){
     if(err==CheckBoardError){
       printf("Address 0 Error!\n");
     }
     else if(err==CardNumError){
       printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  printf("\nCard 0 [DMA test]\n");
  A823_AD_SetChGainMode(0,A823_BI_1,DMA_MODE);
  c1=6; c2=6; quit=0;
  while(!quit){
     A823_AD_DMA_Start(Buf,1000,c1,c2);
     while(A823_AD_DMA_Status()){
       if(kbhit() && getch()==27) {
          quit=1; break;
       }
     };
     A823_AD_DMA_Stop();
     if(kbhit() && getch()==27) break;
     for(i=0;i<1000;i+=100){
        Buf[i]&=0xfff;   /* in DMA mode must change bit 12 to 0 */
        cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]-2048));
     }
  }
```

4.4 TESTDIO.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **Support 8 boards at most**
- **Connect CN1 to CN2 can get DI=DO**

```
#include<stdio.h>
#include"A823.h"

int Base=0x220;

void main(void)
{ int err;
  unsigned int Dio;

  err=A823_Initialize(0,Base,-1,-1);
  if(err){
     if(err==CheckBoardError){
        printf("Address Error!\n");
     }
     else if(err==CardNumError){
        printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  Dio=0x5A5A;
  A823_DO(Dio);
  printf("DO=%4x(Hex)\n",Dio);

  Dio=A823_DI();
  printf("\nDI=%4X(Hex)\n",Dio);
}
```

4.5 TESTDA.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **Support 8 boards at most**

```
#include<stdio.h>
#include"A823.h"

int Base=0x220;

void main(void)
{ int err;

  err=A823_Initialize(0,Base,-1,-1);
  if(err){
     if(err==CheckBoardError){
        printf("Address Error!\n");
     }
     else if(err==CardNumError){
        printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  A823_DA(0,0x8ff);
  A823_DA(1,0xfff);
  printf("\nDA channel 0=0x8FF(2.5V)\n");
  printf("DA channel 1=0xFFF(5.0V)\n");
}
```

4.6 TESTAD.C

- **CARD1:I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **CARD1:I/O ADDRESS=0x240, Interrupt=11, NO DMA**
- **CARD1:I/O ADDRESS=0x260, Interrupt=15, DMA=1**
- **Can support 5 more cards (polling mode)**

```
#include<stdio.h>
#include<conio.h>
#include"A823.h"

int Base0=0x220;
int Base1=0x240;
int Base2=0x260;
int Buf[1000];

void main(void)
{ int err,Dio,i;
  int quit=0;
  float rate=5.0/2048.0;
  int c1,c2;
  int NowCount;

  err=A823_Initialize(0,Base0,-1,-1); /* card 0 : NO DMA,NO IRQ */
  if(err){
     if(err==CheckBoardError){
        printf("Address 0 Error!\n");
     }
     else if(err==CardNumError){
        printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
  err=A823_Initialize(1,Base1,-1,11);  /* card 1 : NO DMA ,IRQ=11*/
  if(err){
     if(err==CheckBoardError){
        printf("Address 1 Error!\n");
     }
     else if(err==CardNumError){
        printf("Card Number Error!(range:0-7)\n");
     }
     return;
  }
```

```
err=A823_Initialize(2,Base2,1,15);  /* card 2 : DMA=1,IRQ=15 */
if(err){
   if(err==CheckBoardError){
      printf("Address 2 Error!\n");
   }
   else if(err==CardNumError){
      printf("Card Number Error!(range:0-7)\n");
   }
   return;
}
printf("\nCard 0 [polling test]\n");
printf("Press any key to begin test\n");
printf("then Press ESC to end test\n");
getch();
A823_ActiveBoard(0);
A823_AD_SetChGainMode(0,A823_BI_1,POLLING_MODE);
while(!quit){
   if(kbhit() && getch()==27) break;
   A823_AD_PollingArray(Buf,1000);
   for(i=0;i<1000;i+=100){
       cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]-2048));
   }
}

printf("\nCard 1 [interrupt test]\n");
printf("Press any key to begin test\n");
printf("then Press ESC to end test\n");
getch();
A823_ActiveBoard(1);
A823_AD_SetChGainMode(0,A823_BI_1,INTERRUPT_MODE);
c1=20; c2=20;
while(!quit){
      A823_AD_INT_Start(Buf,1000,c1,c2);
      do {
         NowCount=A823_AD_INT_Count();
         if(kbhit() && getch()==27) {
            quit=1;
            break;
         }
      } while( NowCount<1000 );
      A823_AD_INT_Stop( );
      if(kbhit() && getch()==27) break;
      for(i=0;i<1000;i+=100){
          cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]-2048));
      }
}
```

```
  printf("\nCard 2 [DMA test]\n");
  printf("Press any key to begin test\n");
  printf("then Press ESC to end test\n");
  getch();
  A823_ActiveBoard(2);
  A823_AD_SetChGainMode(0,A823_BI_1,DMA_MODE);
  c1=6;
  c2=6;
  quit=0;
  while(!quit){
     A823_AD_DMA_Start(Buf,1000,c1,c2);
     while(A823_AD_DMA_Status()){
        if(kbhit() && getch()==27) {
           quit=1;
           break;
        }
     };
     A823_AD_DMA_Stop();
     if(kbhit() && getch()==27) break;
     for(i=0;i<1000;i+=100){
         Buf[i]&=0xfff;   /* in DMA mode must change bit 12 to 0 */
         cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]-2048));
     }
  }
}
```

4.7 GAIN1.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **Support 8 boards at most**
- **Press any key to stop this program**
- **This program is designed for A823_PGH.**
- **The gain settling time is described in Hardware manual sec. 2.4.6 & sec. 2.4.7.**
- **Validate for single-end or differential signal (must set correct hardware jumper and correct wire connection, that is to say, the software is same but the hardware jumper and wire connection are different)**
- **This program only take care the gain settling time. If the multiplexer settling time is needed, this software must to delay another settling time. (refer to Hardware manual sec. 2.4.7)**
- **If the multiplexer and gain control code are changed nearly at the same time, the settling time is the longer one instead of summation of both settling time.**

```
#include   <stdio.h>
      /***********************************/
#include   "A823.h"              /*
*/
                                 /* for A823PGH              */
unsigned int  Buffer[2000];/*                                 */

      /***********************************/
main(void)
{
int  Ch,G,i,j;
float  volt,temp,v0,v1,v2,v3,v4,v5,v6,v7;
unsigned int PollData;

A823_Initialize(0,0x220,-1,-1);
```

```
for (;;)
{
Ch=0; G=0;                          /* Channel_0, Gain=1, Bipolar */
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v0=volt;

Ch=1; G=1;                          /* Channel_1, Gain=10, Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(28*2);                   /* GAIN settling time=28 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v1=volt/10.0;

Ch=2; G=2;                           /* Channel_2,Gain=100,Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(140*2);                   /* GAIN settling time=140 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v2=volt/100.0;

Ch=3; G=3;                           /*Channel_3,Gain=1000,Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(1300*2);                  /* GAIN settling time=1300 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v3=volt/1000.0;

Ch=4; G=4;                          /* Channel_4, Gain=1, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v4=volt;
```

```
Ch=5; G=5;                          /*Channel_5, Gain=10, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(28*2);                   /* GAIN settling time=28 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v5=volt/10.0;

Ch=6; G=6;                           /*Channel_6,Gain=100,Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(140*2);                   /* GAIN settling time=140 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v6=volt/100.0;

Ch=7; G=7;                           /*Chan_7,Gain=1000,Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(1300*2);                  /* GAIN settling time=1300 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v7=volt/1000.0;

printf("\nA823PGH:[0]=%7.5fV,[1]=%7.5fV,[2]=%7.5fV,[3]=%7.5f",v0,v1,v
2,v3);
printf("\nA823PGH:[4]=%7.5fV,[5]=%7.5fV,[6]=%7.5fV,[7]=%7.5f",v4,v5,v
6,v7);
if (kbhit()!=0) {getch(); return;}
}
}
```

4.8 GAIN2.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **Support 8 boards at most**
- **Press any key to stop this program**
- **This program is designed for A823_PGL.**
- **The gain settling time is described in Hardware manual sec. 2.4.6 & sec. 2.4.7.**
- **Validate for single-end or differential signal (must set correct hardware jumper and correct wire connection, that is to say, the software is same but the hardware jumper and wire connection are different)**
- **This program only take care the gain settling time. If the multiplexer settling time is needed, this software must to delay another settling time. (refer to Hardware manual sec. 2.4.7)**
- **If the multiplexer and gain control code are changed nearly at the same time, the settling time is the longer one instead of summation of both settling time.**

```
#include   <stdio.h>              /*****************************/
#include   "A823.h"             /*                             */
                              /* for A823PGL                 */
unsigned int  Buffer[2000];/*                               */
                              /*****************************/
main(void)
{
int  Ch,G,i,j;
float  volt,temp,v0,v1,v2,v3,v4,v5,v6,v7;
unsigned int PollData;

A823_Initialize(0,0x220,-1,-1);
```

```
for (;;)
{
Ch=0; G=0;                          /* Channel_0, Gain=1, Bipolar */
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v0=volt;

Ch=1; G=1;                          /* Channel_1, Gain=2,  Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v1=volt/2.0;

Ch=2; G=2;                           /* Channel_2,Gain=4,  Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(25*2);                    /* GAIN settling time=25  us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v2=volt/4.0;

Ch=3; G=3;                           /* Channel_3,Gain=8, Bipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(28*2);                    /* GAIN settling time=28   us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=(temp-2048)/2048.0*5.0;
v3=volt/8.0;
```

```
Ch=4; G=4;                          /* Channel_4, Gain=1, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v4=volt;

Ch=5; G=5;                          /* Channel_5, Gain=2, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A823_Delay(23*2);                   /* GAIN settling time=23 us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v5=volt/2.0;

Ch=6; G=6;                           /* Channel_6,Gain=4, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(25*2);                    /* GAIN settling time=25  us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v6=volt/4.0;

Ch=7; G=7;                           /* Channel_7,Gain=8, Unipolar*/
A823_AD_SetChGainMode(Ch,G,POLLING_MODE);  /* Polling */
A823_Delay(28*2);                    /* GAIN settling time=28   us */
PollData=A823_AD_PollingVar();
temp=(float)PollData;
volt=temp/4095.0*10.0;
v7=volt/8.0;

printf("\nA823PGL:[0]=%7.5fV,[1]=%7.5fV,[2]=%7.5fV,[3]=%7.5f",v0,v1,v
2,v3);
printf("\nA823PGL:[4]=%7.5fV,[5]=%7.5fV,[6]=%7.5fV,[7]=%7.5f",v4,v5,v
6,v7);
if (kbhit()!=0) {getch(); return;}
}
}
```

4.9 POLLING.C

- **I/O ADDRESS=0x220, NO Interrupt, NO DMA**
- **This program is designed to evaluate the performance of A/D operation by polling mode.**
- **This program will perform 10*100K times A/D conversion by polling mode. If the performance is 100K, this operation will take about 10 sec.**
- **This program will compute the performance automatically.**
- **performance = (10/operation time)*100K**
- **If operation time = 10 sec → performance =100K**
- **If operation time = 20 sec → performance = 50K**
- **The polling mode performance will depend on the performance of CPU.**
- **If use ASC-TI486/33M CPU board of ICP, the performance is about 100K**

```
#include<stdio.h>
#include<conio.h>
#include<sys\timeb.h>
#include"A823.h"

int Base=0x220;
int Buffer[1000];
struct timeb t1,t2;

main(void)
{
    int  i,j,err;
    float  volt,temp,t;
    unsigned int PollData;
```

```
    err=A823_Initialize(0,Base,-1,-1);
    if(err){
          if(err==CheckBoardError){
             printf("Address 0 Error!\n");
             }
          else if(err==CardNumError){
             printf("Card Number Error!(range:0-7)\n");
             }
          return;
          }

    for(i=0; i<1000; i++)  Buffer[i]=1024;
    A823_AD_SetChGainMode(0,A823_BI_1,POLLING_MODE);
    printf("\nWait about 10 sec. to perform 10*100K polling");
    ftime(&t1);
    for (j=0; j<1000; j++)
    A823_AD_PollingArray(Buffer,1000);
    ftime(&t2);
    t=(float)(t2.time-t1.time);
    if (t2.millitm>t1.millitm)
       {t+=(float)(t2.millitm-t1.millitm)/1000.0;}
       else
       {
       t--;
       t+=(float)(1000.0+t2.millitm-t1.millitm)/1000.0;
       }
    printf("\nStart time : %ld %d",t1.time,t1.millitm);
    printf("\nEnd   time : %ld %d",t2.time,t2.millitm);
    printf("\nTotal time : %f sec",t);
    printf("\nPerformance=%fK\n\n",100.0*10.0/t);
    for(i=0,j=0; i<1000;  i+=20,j++)
    {
        if( Buffer[i]==0xffff )
        {
            putch(0x07);
            printf("A/D convertion timeout in polling mode !!!\n");
            exit(0);
        }
        temp=(float)Buffer[i];
        volt=(temp-2048)/2048.0*5.0;
        printf("[%04d]:%+5.3f  ",i,volt);
        if( j%5==4 )  printf("\n");
    }
}
```

4.10 INTERRUP.C

- **I/O ADDRESS=0x220, Interrupt=15, NO DMA**
- **Assume channel_0, Gain=1, interrupt mode**
- **Input C1 & C2 to set the sampling rate (pacer rate)**
- **Input the minimum & maximum value of input signal**
- **This program will loop test until user stop it.**
- **Press any key to stop this program**
- **This program is designed to evaluate the performance of A/D operation by interrupt mode.**
- **The interrupt mode performance will depend on the performance of CPU.**
- **If performance > 60K, the user must use test program intwave.exe(sec. 4.13) to verify again. Because the interrupt request register must be clear to allow next interrupt (Hardware manual sec. 2.4.5), the interrupt service routine maybe has no enough time to clear this register before next pacer clock generated. So this pacer clock may be skipped and this means the performance is mismatch with pacer timer.**
- **If use ASC-TI486/33M CPU board of ICP, the performance is about 60K**

```
#include   <stdio.h>
#include   "A823.H"

int   Buffer[2000];
main(void)
{
    int  DesireCount=2000;    /* The count which We desire to
transfer */
    int  NowCount;        /* The count which is transfering now */
    int  i,j,c1,c2;
    float  volt,min,max,min2,max2,err,rep;

    for(i=0; i<DesireCount; i++)  Buffer[i]=1024;
    A823_Initialize(0,0x220,-1,15);
    A823_AD_SetChGainMode(0,A823_BI_1,INTERRUPT_MODE);
```

```
    /* The clock rate of pacer trigger is equal to 2M/(c1*c2). */
    printf("\nC1 C2="); scanf("%d %d",&c1,&c2);
    printf("\nMin Max="); scanf("%f %f",&min,&max); /* input range */
    err=0; rep=0; min2=5.0; max2=-5.0;
for(;;)
{
    rep++;

printf("\nc1=%d,c2=%d:rep=%.1f,err=%.1f:[%f,%f]",c1,c2,rep,err,min2,m
ax2);
    if( A823_AD_INT_Start(Buffer,DesireCount,c1,c2)!=0 )
    {
       putch(0x07);
       printf("A823_AD_INT_Start() error !!!\n");
       exit(1);
    }

    do
    {
       NowCount=A823_AD_INT_Count();
    } while( NowCount<DesireCount );
    A823_AD_INT_Stop( );

    for(i=0,j=0; i<DesireCount;  i++)
    {
    volt=(float)(Buffer[i]-2048)/2048.0*5.0;
    if (volt<min2) min2=volt; if (volt>max2) max2=volt;
    if (i==0) printf(" : volt[0]=%f",volt);
    if ((volt<min) || (volt>max))
       {
       err++;
       printf("\nmin=%f, max=%f,volt=%f",min,max,volt);
       break;
       }
    }
    delay(300);
    if (kbhit()!=0) {getch(); return;}
}
}
```

4.11 DMA.C

- **I/O ADDRESS=0x220, Interrupt=15, DMA=1**
- **Assume channel_0, Gain=1, DMA mode**
- **Input C1 & C2 to set the sampling rate (pacer rate)**
- **Input the minimum & maximum value of input signal**
- **This program will loop test until user stop it.**
- **Press any key to stop this program**
- **This program is designed to evaluate the performance of A/D operation by DMA mode.**
- **The DMA mode performance will depend on the performance of CPU.**
- **Unlike interrupt mode, this program can evaluate the performance very precisely. The user can use this program to find the maximum performance of his system. The user also can use this program to test the reliability of DMA operation.**
- **If use ASC-TI486/33M CPU board of ICP, the performance is about 100K**

```
#include   <stdio.h>
#include   "A823.h"

int   Buffer[2000];
main(void)
{
   int DesireCount=2000;     /* The count which We desire to transfer
*/
   int NowCount;           /* The count which is transfering now */
   int ErrorCode;
   int i,j,c1,c2;
   float  volt,min,max,min2,max2,err,rep;
```

```
    for(i=0; i<DesireCount; i++) Buffer[i]=1024;
    A823_Initialize(0,0x220,1,15);
    A823_AD_SetChGainMode(0,A823_BI_1,DMA_MODE);
    /* The clock rate of pacer trigger is equal to 2M/(c1*c2). */
    printf("\nC1 C2="); scanf("%d %d",&c1,&c2);
    printf("\nMin Max="); scanf("%f %f",&min,&max);
    err=0; rep=0; min2=5.0; max2=-5.0;
for(;;)
{
rep++;
printf("\nc1=%d,c2=%d:rep=%.1f,err=%.1f:[%f,%f]",c1,c2,rep,err,min2,m
ax2);
   if( (ErrorCode=A823_AD_DMA_Start(Buffer,DesireCount,c1,c2)) )
   {
       putch(0x07);
       printf("A823_AD_DMA_Start() error !!!=%d\n",ErrorCode);
       exit(1);
   }
   while( DmaGo )
   {};
   A823_AD_DMA_Stop();

    for(i=0,j=0; i<DesireCount;  i++)
    {
    Buffer[i]=Buffer[i]&0x0fff;
    volt=(float)(Buffer[i]-2048)/2048.0*5.0;
    if (volt<min2) min2=volt;
    if (volt>max2) max2=volt;
    if (i==0) printf(" : volt[0]=%f",volt);
    if ((volt<min) || (volt>max))
       {
       err++;
       printf("\nmin=%f, max=%f,volt=%f",min,max,volt);
       break;
       }
    }
    delay(300);
    if (kbhit()!=0) {getch(); return;}
}
}
```

4.12 POLLWAVE.C

- **I/O ADDRESS=Key-in, NO Interrupt, NO DMA**
- **Assume channel_0, Gain=1, polling mode**
- **The user must key-in the base address (in hex format, for example 220)**
- **The execution file need the EGAVGA.BGI in the same directory. Press any key twice can stop this program**
- **This program read the input signal and show the wave form in screen . The first 16 input data are show in the upper screen in text format. The wave form is show in the lower screen in graphics format. The sampling data are shown in RED point.**
- **This program does not use pacer timer to sample the input signal. So it is impossible to reconstruct the input wave form very precisely.**
- **This program can use as a test program to evaluate the performance of polling mode.**

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#include <graphics.h>
#include "A823.h"

#define ZERO 204
int   Buffer[2000];

main()
{
int key;
FILE *stream;
char c,buf[80];
```

```
int    DesireCount=2000;
int    i,j,x1,x2,y1,y2,io_base;
float  volt;

init();
printf("\nio base (in hex format) ="); scanf("%x",&io_base);
A823_Initialize(0,io_base,-1,-1);
A823_AD_SetChGainMode(0,A823_BI_1,POLLING_MODE);

   for(;;)
       {
       cleardevice();

       sprintf(buf,"Press Any Key Twice To Stop");
       outtextxy(5,5+10,buf);

       A823_AD_PollingArray(Buffer,DesireCount);

       x1=0;
       j=Buffer[0]&0x0fff;
       y1=j>>5;
       y1+=ZERO;
       for(i=1; i<127;  i++)
              {
              j=Buffer[i]&0x0fff;
              j=j>>5;
              j+=ZERO;
              x2=i*5; y2=j;
              line(x1,y1,x2,y2);
              x1=x2; y1=y2;
       }

       for(i=0; i<127;  i++)
              {
              j=Buffer[i]&0x0fff;
              j=j>>5;
              j+=ZERO;
              putpixel(i*5,j,RED);
              putpixel(i*5,j+1,RED);
              putpixel(i*5+1,j,RED);
              putpixel(i*5+1,j+1,RED);
              }

       j=ZERO+64;
       for (i=0; i<639; i++)  putpixel(i,j,GREEN);
       for (i=0; i<639; i++)  putpixel(i,j-65,GREEN);
```

```
       for (i=0; i<639; i++)  putpixel(i,j+65,GREEN);
       for (j=ZERO-65; j<ZERO+65; j++) putpixel(0,j+64,GREEN);
       for (j=ZERO-65; j<ZERO+65; j++) putpixel(639,j+64,GREEN);

       for(i=0; i<16;      i++)
       {
       Buffer[i]=Buffer[i]&0x0fff;
       volt=(float)(Buffer[i]-2048)/2048.0*5.0;
       sprintf(buf,"POLLING : [%02d] --> [%03xH]=%+5.3f
Volt",i,Buffer[i],volt);
       outtextxy(5,5+10*(i+3),buf);
       }

       delay(1000);
       if (kbhit()!=0) {getch(); goto ret_label;}
       }
ret_label:
       getch();
       cleardevice();
       closegraph();
       }

/* ---------------------------------- */

init()
{
int driver,mode,char_size;

driver=EGA;
mode=EGAHI;
initgraph(&driver,&mode,"");
setactivepage(0);
setvisualpage(0);
setcolor(15);
setbkcolor(0);
settextstyle(SMALL_FONT,HORIZ_DIR,6);
}
```

4.13 INTWAVE.C

- **I/O ADDRESS=Key-in, Interrupt=Key-in, NO DMA**
- **Assume channel_0, Gain=1, interrupt mode**
- **The user must key-in the base address (in hex format, for example 220), interrupt channel number and C1&C2 to set the sampling rate (pacer rate)**
- **The execution file need the EGAVGA.BGI in the same directory. Press any key twice can stop this program**
- **This program read the input signal and show the wave form in screen . The first 16 input data are show in the upper screen in text format. The wave form is show in the lower screen in graphics format. The sampling data are shown in RED point.**
- **This program use pacer timer to sample the input signal. So it is possible to reconstruct the input wave form very precisely.**
- **This program can use as a test program to evaluate the performance of interrupt mode.**

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#include <graphics.h>
#include "A823.h"

#define ZERO 204
int   Buffer[2000];

main()
{
int key,c1,c2;
FILE *stream;
char c,buf[80];
```

```
   int DesireCount=2000;     /* The count which We desire to transfer
*/
   int NowCount;          /* The count which is transfering now */
   int ErrorCode;
   int i,j,x1,x2,y1,y2,io_base,irq_no;
   float  volt;

   init();
   printf("\nio base  (in hex format, 220) = ");
scanf("%x",&io_base);
   printf("IRQ number (in int format, 15)  = "); scanf("%d",&irq_no);
   printf("(100K --> C1=4) --> C1= "); scanf("%d",&c1);
   printf("(100K --> C2=5) --> C2= "); scanf("%d",&c2);
   A823_Initialize(0,io_base,-1,irq_no);
   A823_AD_SetChGainMode(0,A823_BI_1,INTERRUPT_MODE);

   for(;;)
       {
       cleardevice();

       sprintf(buf,"C1=%d C2=%d --> Sampling
Rate=%5.1fK",c1,c2,2000.0/(float)(c1*c2));
       outtextxy(5,5,buf);
       sprintf(buf,"Press Any Key Twice To Stop");
       outtextxy(5,5+10,buf);

       if( A823_AD_INT_Start(Buffer,DesireCount,c1,c2)!=0 )
           {
           putch(0x07);
           printf("A823_AD_INT_Start() error !!!\n");
           exit(1);
           }

       do
           {
           NowCount=A823_AD_INT_Count();
           if (kbhit()!=0) {getch(); goto ret_label;}
           }while( NowCount<DesireCount );
       A823_AD_INT_Stop( );

       x1=0;
       j=Buffer[0]&0x0fff;
       y1=j>>5;
       y1+=ZERO;
```

```
	for(i=1; i<127;  i++)
		{
		j=Buffer[i]&0x0fff;
		j=j>>5;
		j+=ZERO;
		x2=i*5; y2=j;
		line(x1,y1,x2,y2);
		x1=x2; y1=y2;
	}

	for(i=0; i<127;  i++)
		{
		j=Buffer[i]&0x0fff;
		j=j>>5;
		j+=ZERO;
		putpixel(i*5,j,RED);
		putpixel(i*5,j+1,RED);
		putpixel(i*5+1,j,RED);
		putpixel(i*5+1,j+1,RED);
		}

	j=ZERO+64;
	for (i=0; i<639; i++)  putpixel(i,j,GREEN);
	for (i=0; i<639; i++)  putpixel(i,j-65,GREEN);
	for (i=0; i<639; i++)  putpixel(i,j+65,GREEN);
	for (j=ZERO-65; j<ZERO+65; j++) putpixel(0,j+64,GREEN);
	for (j=ZERO-65; j<ZERO+65; j++) putpixel(639,j+64,GREEN);

	for(i=0; i<16;      i++)
	{
	Buffer[i]=Buffer[i]&0x0fff;
	volt=(float)(Buffer[i]-2048)/2048.0*5.0;
	sprintf(buf,"Interrupt : [%02d] --> [%03xH]=%+5.3f
Volt",i,Buffer[i],volt);
	outtextxy(5,5+10*(i+3),buf);
	}

	delay(1000);
	if (kbhit()!=0) {getch(); goto ret_label;}
	}
ret_label:
	getch();
	cleardevice();
	closegraph();
	}
```

```
/* ------------------------------------------------ */

init()
{
int driver,mode,char_size;

driver=EGA;
mode=EGAHI;
initgraph(&driver,&mode,"");
setactivepage(0);
setvisualpage(0);
setcolor(15);
setbkcolor(0);
settextstyle(SMALL_FONT,HORIZ_DIR,6);
}
```

4.14 DMAWAVE.C

- **I/O ADDRESS=Key-in, Interrupt=Key-in, DMA=Key-in**
- **Assume channel_0, Gain=1, DMA mode**
- **The user must key-in the base address (in hex format, for example 220) , interrupt channel number, DMA channel number and C1&C2 to set the sampling rate (pacer rate)**
- **The execution file need the EGAVGA.BGI in the same directory. Press any key twice can stop this program**
- **This program read the input signal and show the wave form in screen . The first 16 input data are show in the upper screen in text format. The wave form is show in the lower screen in graphics format. The sampling data are shown in RED point.**
- **This program use pacer timer to sample the input signal. So it is possible to reconstruct the input wave form very precisely.**
- **This program can use as a test program to evaluate the performance of interrupt mode.**

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#include <graphics.h>
#include "A823.h"

#define ZERO 204
int   Buffer[2000];

main()
{
int key,c1,c2;
FILE *stream;
char c,buf[80];
```

```
int    DesireCount=2000;     /* The count which We desire to transfer
*/
int    NowCount;          /* The count which is transfering now */
int    ErrorCode;
int    i,j,x1,x2,y1,y2,io_base,irq_no,dma_no;
float  volt;

init();
printf("\nio base (in hex format,220)=");scanf("%x",&io_base);
printf("IRQ number(in int format,15) =");scanf("%d",&irq_no);
printf("DMA number(in int format,1/3)=");scanf("%d",&dma_no);
printf("(100K --> C1=4) --> C1= "); scanf("%d",&c1);
printf("(100K --> C2=5) --> C2= "); scanf("%d",&c2);
A823_Initialize(0,io_base,dma_no,irq_no);
A823_AD_SetChGainMode(0,A823_BI_1,DMA_MODE);

   for(;;)
       {
       cleardevice();

       sprintf(buf,"C1=%d C2=%d --> Sampling
Rate=%5.1fK",c1,c2,2000.0/(float)(c1*c2));
       outtextxy(5,5,buf);
       sprintf(buf,"Press Any Key Twice To Stop");
       outtextxy(5,5+10,buf);

       if( A823_AD_DMA_Start(Buffer,DesireCount,c1,c2)!=0 )
           {
           putch(0x07);
           printf("A823_AD_DMA_Start() error !!!\n");
           exit(1);
           }

       while(DmaGo)
             {
             if (kbhit()!=0) {getch(); goto ret_label;}
             }
       A823_AD_DMA_Stop( );

       x1=0;
       j=Buffer[0]&0x0fff;
       y1=j>>5;
       y1+=ZERO;
```

```
	for(i=1; i<127;  i++)
		{
		j=Buffer[i]&0x0fff;
		j=j>>5;
		j+=ZERO;
		x2=i*5; y2=j;
		line(x1,y1,x2,y2);
		x1=x2; y1=y2;
	}

	for(i=0; i<127;  i++)
		{
		j=Buffer[i]&0x0fff;
		j=j>>5;
		j+=ZERO;
		putpixel(i*5,j,RED);
		putpixel(i*5,j+1,RED);
		putpixel(i*5+1,j,RED);
		putpixel(i*5+1,j+1,RED);
		}

	j=ZERO+64;
	for (i=0; i<639; i++)  putpixel(i,j,GREEN);
	for (i=0; i<639; i++)  putpixel(i,j-65,GREEN);
	for (i=0; i<639; i++)  putpixel(i,j+65,GREEN);
	for (j=ZERO-65; j<ZERO+65; j++) putpixel(0,j+64,GREEN);
	for (j=ZERO-65; j<ZERO+65; j++) putpixel(639,j+64,GREEN);

	for(i=0; i<16;      i++)
	{
	Buffer[i]=Buffer[i]&0x0fff;
	volt=(float)(Buffer[i]-2048)/2048.0*5.0;
	sprintf(buf,"DMA     : [%02d] --> [%03xH]=%+5.3f
Volt",i,Buffer[i],volt);
	outtextxy(5,5+10*(i+3),buf);
	}

	delay(1000);
	if (kbhit()!=0) {getch(); goto ret_label;}
	}
ret_label:
	getch();
	cleardevice();
	closegraph();
	}
```

```
/* ------------------------------------------------- */

init()
{
int driver,mode,char_size;

driver=EGA;
mode=EGAHI;
initgraph(&driver,&mode,"");
setactivepage(0);
setvisualpage(0);
setcolor(15);
setbkcolor(0);
settextstyle(SMALL_FONT,HORIZ_DIR,6);
}
```