

TABLE OF CONTENTS

1. INTRODUCTION	3
2. INSTALLATION	4
2.1 COMPILER & LINK USING MSC	6
2.2 COMPILER & LINK USING TC	6
2.3 COMPILER & LINK USING BC	6
3. C LANGUAGE LIBRARY.....	7
3.1 A826.H.....	8
3.2 A826_INITIALIZE	10
3.3 A826_ACTIVEBOARD	12
3.4 A826_CHECK_ADDRESS.....	13
3.5 A826_DI.....	14
3.6 A826_DO	15
3.7 A826_DA	16
3.8 A826_AD_SETCHGAINMODE	17
3.9 A826_AD_POLLINGVAR.....	18
3.10 A826_AD_POLLINGARRAY	19
3.11 A826_AD_INT_START.....	20
3.12 A826_AD_INT_COUNT.....	21
3.13 A826_AD_INT_STOP	22
3.14 A826_AD_DMA_START	23
3.15 A826_AD_DMA_STATUS	24
3.16 A826_AD_DMA_STOP	25
3.17 A826_DELAY	26
4. DEMO PROGRAM	27
4.1 TESTPOLL.C	28
4.2 TESTINT.C.....	29
4.3 TESTDMA.C	30
4.4 TESTDIO.C	31
4.5 TESTDA.C.....	32
4.6 TESTAD.C.....	33
4.7 GAIN2.C.....	36
4.8 POLLING.C.....	39
4.9 INTERRUPT.C	41
4.10 DMA.C.....	43

4.11	POLLWAVE.C	45
4.12	INTWAVE.C.....	48
4.13	DMAWAVE.C	51

1. Introduction

The xA826.lib is a collection of data acquisition subroutines for A826PG. 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 xA826.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 the developing process for user, some demonstration C source programs 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 A826PG application software to your hard disk to get the best performance. Before beginning, to make a backup copy of the A826PG application software. Store the original diskette in a safe place. The A826PG application disk includes the following files:

- A826\A82xDiag.EXE : The diagnostic utility for A826PG.
- A826\A82x.CFG : The configuration file of A82xDiag.EXE.

- A826\lib\A826.h : The header file for MSC/TC/BC++.
- A826\lib\SA826.lib : The **small** model library for MSC/TC/BC++.
- A826\lib\MA826.lib : The **medium** model library for MSC/TC/BC++.
- A826\lib\CA826.lib : The **compact** model library for MSC/TC/BC++.
- A826\lib\LA826.lib : The **large** model library for MSC/TC/BC++.
- A826\lib\HA826.lib : The **huge** model library for MSC/TC/BC++.

- A826\demo\msc*.* : The demo program for MSC 5.X or 6.X
- A826\demo\tc*.* : The demo program for TC 2.X
- A826\demo\bc*.* : The demo program for BC++ 3.X or TC++3.0

- A826\demo\msc*.c : The source files of demo program
- A826\demo\msc*.exe : The execution files of demo program
- A826\demo\msc*.bat : The compiler & link batch files of demo program

- A826\demo\tc*.c : The source files of demo program
- A826\demo\tc*.exe : The execution files of demo program.
- A826\demo\tc*.prj : The compiler & link project files of demo program

- A826\demo\bc*.c : The source files of demo program
- A826\demo\bc*.exe : The execution files of demo program
- A826\demo\bc*.prj : The project files of demo program

The demo programs on A826\demo\tc are as following:

1. A826\demo\tc\testpoll.c : The demo program for A/D Polling testing
2. A826\demo\tc\testint.c : The demo program for A/D Interrupt testing
3. A826\demo\tc\testdma.c : The demo program for A/D DMA testing
4. A826\demo\tc\testdio.c : The demo program for digital I/O testing
5. A826\demo\tc\testda.c : The demo program for D/A testing
6. A826\demo\tc\testad.c : The demo program for A/D Polling/INT/DMA testing
7. A826\demo\tc\pollwave.c : Test A/D using polling mode & show the wave form
8. A826\demo\tc\intwave.c : Test A/D using interrupt mode & show the wave form
9. A826\demo\tc\dmawave.c : Test A/D using DMA mode & show the wave form
10. A826\demo\tc\gain2.c : Delay the gain settling time for A826PG
11. A826\demo\tc\polling.c : Performance of A/D polling mode
12. A826\demo\tc\interrup.c : Performance of A/D interrupt mode
13. A826\demo\tc\dma.c : Performance of A/D DMA mode

*All of these demo programs are compiled & test using **Large model** library.*

- **A826\demo\msc*.c : only program # 1 ~ 9 are supported**
- **A826\demo\bc*.c : only program #1 ~ 9 are supported**

2.1 Compiler & link using MSC

- The including file is **A826.H**
- There are 5 different mode library files : **S/C/M/L/HA826.LIB**
- Support MSC 5.x or 6.x compiler
- **SMALL** model compiler & link command : CL /AS program.c SA826.LIB
- **COMPACT** model compiler & link command : CL /AC program.c CA826.LIB
- **MEDIUM** model compiler & link command : CL /AM program.c MA826.LIB
- **LARGE** model compiler & link command : CL /AL program.c LA826.LIB
- **HUGE** model compiler & link command : CL /AH program.c HA826.LIB
- A:\a826\demo\msc*.bat are some examples for compiler & link batch file

2.2 Compiler & link using TC

- The including file is **A826.H**
- There are 5 different model library files : **S/C/M/L/HA826.LIB**
- Support TC 2.x compiler
- Use text editor to create a project file include : program.c xA826.lib(x=S,C,M,L,H)
- Use TC integrated environment(**IDE**) to **select the correct compiler model & project file**
- A:\a826\demo\tc*.prj are some examples for project file of TC 2.0

2.3 Compiler & link using BC++

- The including file is **A826.H**
- There are 5 different model library files : **S/C/M/L/HA826.LIB**
- Support BC++ 1.0~5.x or TC++ 3.0 compiler
- Use BC++ integrated environment to create a project file include : program.c xA826.lib
- Use BC++ integrated environment to **select the correct compiler model**
- A:\a826\demo\bc*.prj are some examples for the project file of BC++

3. C Language Library

- A826.H
- A826_Initialize
- A826_ActiveBoard
- A826_Check_Address
- A826_DI
- A826_DO
- A826_DA
- A826_AD_SetChGainMode
- A826_AD_PollingVar
- A826_AD_PollingArray
- A826_AD_INT_Start
- A826_AD_INT_Count
- A826_AD_INT_Stop
- A826_AD_DMA_Start
- A826_AD_DMA_Stop
- A826_AD_DMA_Status
- A826_Delay

3.1 A826.H

```
#ifndef __cplusplus
extern "C" {
#endif

int A826_Initialize(int CardNo, int BaseAddr, int DmaNo, int IrqNo);
int A826_ActiveBoard(int BoardNo);
int A826_Check_Address(int BaseAddr);
unsigned int A826_DI(void);
void A826_DO(unsigned DigitOutput);
int A826_DA(int Channel, unsigned int data);
void A826_AD_SetChGainMode(int Channel, int Gain, int Mode);
void A826_AD_PollingArray(unsigned int *Buffer, unsigned int Length);
unsigned int A826_AD_PollingVar(void);
int A826_AD_INT_Start(int* Buffer, unsigned int count, int c1, int c2);
unsigned A826_AD_INT_Count(void);
void A826_AD_INT_Stop(void);
int A826_AD_DMA_Start(int far *Buffer, unsigned int count, int c1, int c2);
void A826_AD_DMA_Stop(void);
int A826_AD_DMA_Status(void);
int A826_Delay(unsigned int DownCount);

#ifdef __cplusplus
}
#endif

/***** DEFINE A826 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 A826_BI_1 0
#define A826_BI_2 1
#define A826_BI_4 2
#define A826_BI_8 3

/**/ 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 A826_Initialize

- **Description :**

A826_Initialize initialize the A826PG board. This function should be called before using the other A826PG library subroutines. This function will detect automatically A826PG 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, must use the function A826_ActiveBoard() to switch the active board.

- **Syntax :**

int A826_Initialize(int **BoardNo**, int **BaseAddr**, int **DmaNo**, int **IrqNo**);

- **Input Parameter :**

BoardNo : The validate board 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 : Interrupt channel number. (must match with hardware jumper)

-1 = NO Interrupt

- **Return Value :**

CheckBoardError : I/O 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 : no error

- **Demo program :**

Sec. 4.1 ~ Sec. 4.14

● Example 1 :

```
#include "A826.h"
main()
{
    int BoardNo=0;      /* only one card */
    int IOBase=0x220;   /* The IO base address for A826 */
    int DmaNo=1;        /* The DMA channel no. */
    int IrqNo=15;       /* The IRQ channel no. */

    A826_Initialize(BoardNo,IOBase,DmaNo,IrqNo); /* board_0 is active */
    /* only one card in this system, so no need to call A826_ActiveBoard(0) */
    .
    .
}
```

● Example 2 :

```
#include "A826.h"
main()
{
    A826_Initialize(0,0x200,1,12); /* now board_0 is active*/
    A826_Initialize(1,0x210,3,15); /* now board_1 is active*/
    A826_Initialize(2,0x220,-1,-1); /* now board_2 is active*/
    /* now there are 3 cards in this system, so A826_ActiveBoard(?) need to be called */

    A826_ActiveBoard(0); /* now card_0 is active */

    A826_ActiveBoard(1); /* now card_1 is active */

    A826_ActiveBoard(2); /* now card_2 is active */

}
```

3.3 A826_ActiveBoard

- **Description :**

This driver support 8 different cards in one system max. The A826_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 A826_ActiveBoard(?) is used to active the desired card in the system. If only one A826 board in the system, there is no need to use this subroutine.

- **Syntax :**

```
int A826_ActiveBoard(int BoardNo);
```

- **Input Parameter :**

BoardNo : The validate board number is from 0 to 7. This number is equal to BoardNo in A826_Initial(BoardNo,.....);

- **Return Value :**

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

NoError : no error

- **Example 1 :**

See sec. 4.6

3.4 A826_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 A826_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 : no error

- **Demo program :**

Sec. 4.1 ~ Sec. 4.14

- **Example 1 :**

```
#include "A826.h"
main()
{
    if (A826_Check_Address(0x220)==NoError) {
        printf("\n0x220 find a card");
    }
    else {
        printf("\n0x220 cannot find any card");
    }
}
```

3.5 A826_DI

- **Description :**

This function read the 16 bits digital input data from I/O port Base+6 and Base+7.

- **Syntax :**

unsigned int **A826_DI**(void);

- **Input Parameter :**

None.

- **Return Value :**

A 16 bits data from digit input port.

- **Demo program :**

Sec. 4.4

- **Example 1 :**

```
#include "A826.H"
main()
{
    unsigned DigitInput;

    A826_Initialize(0,0x220,1,15);
    .
    .
    DigitInput=A826_DI();
    .
}
```

3.6 A826_DO

- **Description :**

This function write data to I/O port Base+0x0D and Base+0x0E.

- **Syntax :**

```
void A826_DO(unsigned DigitalOutput);
```

- **Input Parameter :**

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

- **Return Value :**

None.

- **Demo program :**

Sec. 4.4

- **Example 1 :**

```
#include "A826.H"
main()
{
    unsigned DigitOutput=0x1234;

    A826_Initialize(0,0x220,1,15);
    .
    .
    A826_DO(DigitOutput);
    .
    .
}
```

3.7 A826_DA

- **Description :**

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

- **Syntax :**

int **A826_DA**(int **Channel**, unsigned int **data**);

- **Input Parameter :**

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

data : 0 through 4095.(0x0FFF)

- **Return Value :**

DAchannelError : validate Channel is only 0 and 1

NoError: no error

- **Demo program :** Sec. 4.5

- **Example 1 :**

```
#include "A826.h"
main()
{
    unsigned DAOutput=0x0234;

    A826_Initialize(0,0x220,15,1);

    A826_DA(0,DAOutput);    /* D/A channel 0 */
    A826_DA(1,DAOutput);    /* D/A channel 1 */

}
```

3.8 A826_AD_SetChGainMode

- **Description :**

This function is used to set the A/D channel number, gain and operation mode. The detail information is given in “A-826PG Hardware Manual” section 2.7.

- **Syntax :**

```
void A826_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 ~ 3(Reference to A826.H)

Mode : A/D operation mode, 0 or 1 or 2 or 6(Reference to A826.H)

- **Return Value :**

None

- **Demo program :**

Sec. 4.1 ~ sec. 4.14

- **Example 1 :**

```
#include "A826.h"
main()
{
    int Gain, Mode, channel;

    A826_Initialize(0,0x220,1,15);

    /* user must define Gain, Mode here */

    for (channel=0; channel<16; channel++){
        A826_AD_SetChGainMode(channel,Gain,Mode);
        /* delay settling time if needed */
    }
}
```

3.9 A826_AD_PollingVar

- **Description :**

This function performs the A/D conversion by polling .

- **Syntax :**

unsigned A826_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 "A826.h"
#include <stdio.h>
main()
{
    int channel,gain,mode;
    unsigned PollData;
    float    volt;
    A826_Initialize(0,0x220,1,15);
    channel=0;    /* channel 0 */
    gain=0;      /* bipolar, gain=1, range = -10V ~ +10V */
    mode=1;     /* polling mode */
    A826_AD_SetChGainMode(channel,gain,mode);
    PollData=A826_AD_PollingVar();
    volt=((float)PollData)/32768.0*10.0;
    printf("\ndata = %xH = %5.3fV\n",PollData,volt);
}
```

3.10 A826_AD_PollingArray

- **Description :**

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

- **Syntax :**

void **A826_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 "A826.h"
#include <stdio.h>
main()
{
    int channel,gain,mode;
    unsigned int i,Buffer[1000],DesireCount=1000;
    A826_Initialize(0,0x220,1,15);
    channel=0; /* channel 0 */
    gain=0; /* bipolar, gain=1, range=-10V ~ +10V */
    mode=1; /* polling mode */
    A826_AD_SetChGainMode(channel,gain,mode);
    A826_AD_PollingArray(Buffer, DesireCount);
    for (i=0; i<10; i++) printf("\nBuffer[%d]=%x",i,Buffer[i]);
}
```

3.11 A826_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 “A-826PG Hardware Manual” section 2.7.

- **Syntax :**

int **A826_AD_INT_Start**(int ***Buffer**, unsigned int **Count** , int **c1** , int **c2**)

- **Input Parameter :**

Buffer : address of Buffer to store the 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= $2000/(2*6)=111.1K > 110K$ → 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 the maximum performance.

3.12 A826_AD_INT_Count

- **Description :**

Return the current A/D conversion count number. When the return value equal to the parameter **Count** of `A826_AD_INT_Start` means the interrupt AD conversion is finished.

- **Syntax :**

unsigned `A826_AD_INT_Count`(void)

- **Input Parameter :**

None.

- **Return Value :**

Current count value.

- **Example 1 :**

See sec.4.2/4.10/4.13

3.13 A826_AD_INT_Stop

- **Description :**

Stop the operation initiate by A826_AD_INT_Start() function.

- **Syntax :**

void A826_AD_INT_Stop(void)

- **Input Parameter :**

None.

- **Return Value :**

None.

- **Example 1 :**

See sec.4.2/4.10/4.13

3.14 A826_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 “A-826PG Hardware Manual” section 2.7.

- **Prototype :**

int A826_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 A826_Initial(...) is error

NoError : no error

- **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= $2000/(2*6)=111.1K$ >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 the maximum performance.

3.15 A826_AD_DMA_Status

- **Description :**

Return the status of DMA operation.

- **Syntax :**

int A826_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 A826_AD_DMA_Stop

- **Description :**

Stop the operation initiate by A826_AD_DMA_Start() function.

- **Syntax :**

void A826_AD_DMA_Stop(void)

- **Input Parameter :**

None.

- **Return Value :**

None.

- **Example 1 :**

See sec.4.3/4.11/4.14

3.17 A826_Delay

- **Description :**

Use counter 0 to implement a machine independent timer. The detail information is given in “A-826PG Hardware Manual” section 2.6.

- **Syntax :**

int **A826_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 **A826_Delay()** will not function properly.

2. Because the counter0 is used by **A826_Delay()**, the counter0 can not be use as a user timer/counter.

4. Demo Program

- A826\demo\tc\testpoll.c : The demo program for A/D Polling testing
- A826\demo\tc\testint.c : The demo program for A/D Interrupt testing
- A826\demo\tc\testdma.c : The demo program for A/D DMA testing
- A826\demo\tc\testdio.c : The demo program for digital I/O testing
- A826\demo\tc\testda.c : The demo program for D/A testing
- A826\demo\tc\testad.c : The demo program for A/D Polling/INT/DMA testing
- A826\demo\tc\gain2.c : Delay the gain settling time for A826PG
- A826\demo\tc\polling.c : Performance of A/D polling mode
- A826\demo\tc\interrupt.c : Performance of A/D interrupt mode
- A826\demo\tc\dma.c : Performance of A/D DMA mode
- A826\demo\tc\pollwave.c : Test A/D using polling mode & show the wave form
- A826\demo\tc\intwave.c : Test A/D using interrupt mode & show the wave form
- A826\demo\tc\dmawave.c : Test A/D using DMA mode & show the wave form

4.1 TESTPOLL.C

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

```
#include<stdio.h>
#include<conio.h>
#include"a826.h"

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

void main(void)
{ int err;
  int i;
  float rate=10.0/32768.0;

  err=A826_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 [A826_AD_PollingArray]\n");
  A826_AD_SetChGainMode(0,A826_BI_1,POLLING_MODE);
  A826_AD_PollingArray(Buf,1000);
  for(i=0;i<1000;i+=100){
    printf("[%3d]%6.3f\n",i,rate*(Buf[i]));
  }
  printf("Using [A826_AD_PollingVar]\n");
  Buf[0]=A826_AD_PollingVar();
  printf("PollingVar=%6.3f\n",rate*(Buf[0]));
}
```

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"a826.h"

int Base=0x220,Irq=15;
int Buf[1000];
void main(void)
{ int err,i,c1,c2,NowCount,quit=0;
  float rate=10.0/32768.0;

  err=A826_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");
  A826_AD_SetChGainMode(0,A826_BI_1,INTERRUPT_MODE);
  c1=20; c2=20;
  while(!quit){
    A826_AD_INT_Start(Buf,1000,c1,c2);
    do {
      NowCount=A826_AD_INT_Count();
      if(kbhit() && getch()==27) {
        quit=1; break;
      }
    } while( NowCount<1000 );
    A826_AD_INT_Stop( );
    if(kbhit() && getch()==27) break;
    for(i=0;i<1000;i+=100){
      printf("[%3d]%6.3f\n",i,rate*(Buf[i]));
    }
  }
}
```

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"a826.h"

int Base=0x220,Dma=1,Irq=15,Buf[1000];
void main(void)
{ int err,i,c1,c2,quit=0;
  float rate=10.0/32768.0;

  err=A826_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");
  A826_AD_SetChGainMode(0,A826_BI_1,DMA_MODE);
  c1=6; c2=6; quit=0;
  while(!quit){
    A826_AD_DMA_Start(Buf,1000,c1,c2);
    while(A826_AD_DMA_Status()){
      if(kbhit() && getch()==27) {
        quit=1; break;
      }
    }
    A826_AD_DMA_Stop();
    if(kbhit() && getch()==27) break;
    for(i=0;i<1000;i+=100){
      Buf[i]&=0xffff; /* in DMA mode must change bit 12 to 0 */
      cprintf("[%3d]%.3f\n\r",i,rate*(Buf[i]));
    }
  }
}
```

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"a826.h"

int Base=0x220;

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

  err=A826_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;
  A826_DO(Dio);
  printf("DO=%4x(Hex)\n",Dio);

  Dio=A826_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"a826.h"

int Base=0x220;

void main(void)
{ int err;

err=A826_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;
}
A826_DA(0,0x8ff);
A826_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"826.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=10.0/32768.0;
  int c1,c2;
  int NowCount;

  err=A826_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=A826_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=A826_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();
A826_ActiveBoard(0);
A826_AD_SetChGainMode(0,A826_BI_1,POLLING_MODE);
while(!quit){
    if(kbhit() && getch()==27) break;
    A826_AD_PollingArray(Buf,1000);
    for(i=0;i<1000;i+=100){
        cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]));
    }
}

printf("\nCard 1 [interrupt test]\n");
printf("Press any key to begin test\n");
printf("then Press ESC to end test\n");
getch();
A826_ActiveBoard(1);
A826_AD_SetChGainMode(0,A826_BI_1,INTERRUPT_MODE);
c1=20; c2=20;
while(!quit){
    A826_AD_INT_Start(Buf,1000,c1,c2);
    do {
        NowCount=A826_AD_INT_Count();
        if(kbhit() && getch()==27) {
            quit=1;
            break;
        }
    } while( NowCount<1000 );
    A826_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]));
    }
}

printf("\nCard 2 [DMA test]\n");
printf("Press any key to begin test\n");
printf("then Press ESC to end test\n");
getch();
A826_ActiveBoard(2);
A826_AD_SetChGainMode(0,A826_BI_1,DMA_MODE);
c1=6;
c2=6;
quit=0;
while(!quit){

```

```

A826_AD_DMA_Start(Buf,1000,c1,c2);
while(A826_AD_DMA_Status()) {
    if(kbhit() && getch()==27) {
        quit=1;
        break;
    }
};
A826_AD_DMA_Stop();
if(kbhit() && getch()==27) break;
for(i=0;i<1000;i+=100){
    Buf[i]&=0xffff; /* in DMA mode must change bit 12 to 0 */
    cprintf("[%3d]%6.3f\n\r",i,rate*(Buf[i]));
}
}
}

```

4.7 GAIN2.C

- I/O ADDRESS=0x220, NO Interrupt, NO DMA
- Support 8 boards at most
- Press any key to stop this program
- The gain settling time is described in A-826PG 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 A-826PG 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 "A826.h"          /*          */
                          /* for A826PG    */
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;

A826_Initialize(0,0x220,-1,-1);

for (;;)
{
Ch=0; G=0;                /* Channel_0, Gain=1, Bipolar */
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(23*2);        /* GAIN settling time=23 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=(temp)/327687.0*10.0;
v0=volt;
```

```

Ch=1; G=1; /* Channel_1, Gain=2, Bipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(23*2); /* GAIN settling time=23 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=(temp)/32768.0*10.0;
v1=volt/2.0;

Ch=2; G=2; /* Channel_2, Gain=4, Bipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(25*2); /* GAIN settling time=25 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=(temp)/32768.0*10.0;
v2=volt/4.0;

Ch=3; G=3; /* Channel_3, Gain=8, Bipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(28*2); /* GAIN settling time=28 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=(temp)/32768.0*10.0;
v3=volt/8.0;

Ch=4; G=4; /* Channel_4, Gain=1, Unipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(23*2); /* GAIN settling time=23 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=temp/32768.0*10.0;
v4=volt;

Ch=5; G=5; /* Channel_5, Gain=2, Unipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(23*2); /* GAIN settling time=23 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=temp/32768.0*10.0;
v5=volt/2.0;

Ch=6; G=6; /* Channel_6, Gain=4, Unipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(25*2); /* GAIN settling time=25 us */
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=temp/32768.0*10.0;
v6=volt/4.0;

Ch=7; G=7; /* Channel_7, Gain=8, Unipolar*/
A826_AD_SetChGainMode(Ch,G,POLLING_MODE); /* Polling */
A826_Delay(28*2); /* GAIN settling time=28 us */

```

```
PollData=A826_AD_PollingVar();
temp=(float)PollData;
volt=temp/32768.0*10.0;
v7=volt/8.0;

printf("\nA826PGL:[0]=%7.5fV,[1]=%7.5fV,[2]=%7.5fV,[3]=%7.5f",
v0,v1,v2,v3);
printf("\nA826PGL:[4]=%7.5fV,[5]=%7.5fV,[6]=%7.5fV,[7]=%7.5f",
v4,v5,v6,v7);
if (kbhit()!=0) {getch(); return;}
}
}
```

4.8 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"826.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=A826_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;
A826_AD_SetChGainMode(0,A826_BI_1,POLLING_MODE);
printf("\nWait about 10 sec. to perform 10*100K polling");
ftime(&t1);
for (j=0; j<1000; j++)
    A826_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 ){
        putchar(0x07);
        printf("A/D conversion timeout in polling mode !!!\n");
        exit(0);
    }
    temp=(float)Buffer[i];
    volt=(temp)/32768.0*5.0;
    printf("[%04d]:%+5.3f  ",i,volt);
    if( j%5==4 ) printf("\n");
}
}

```

4.9 INTERRUPT.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 (A-826PG 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 "A826.H"

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

for(i=0; i<DesireCount; i++) Buffer[i]=1024;
A826_Initialize(0,0x220,-1,15);
A826_AD_SetChGainMode(0,A826_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,e
rr,min2,max2);
if( A826_AD_INT_Start(Buffer,DesireCount,c1,c2)!=0 ){
    putch(0x07);
    printf("A826_AD_INT_Start() error !!!\n");
    exit(1);
}

do{
    NowCount=A826_AD_INT_Count();
} while( NowCount<DesireCount );
A826_AD_INT_Stop( );

for(i=0,j=0; i<DesireCount; i++){
    volt=(float)(Buffer[i])/342768.0*10.0;
    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;}
}
}

```

4.10 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 "A826.h"

int Buffer[2000];
main(void)
{
    int DesireCount=2000; /*The count which We desire to transfer */
    int NowCount; /* The count which is transferring 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;
    A826_Initialize(0,0x220,1,15);
    A826_AD_SetChGainMode(0,A826_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,max2);
if( (ErrorCode=A826_AD_DMA_Start(Buffer,DesireCount,c1,c2))
){
    putchar(0x07);
    printf("A826_AD_DMA_Start() error !!!=%d\n",ErrorCode);
    exit(1);
}
while(A826_AD_DMA_Status() ){};
A826_AD_DMA_Stop();

for(i=0,j=0; i<DesireCount; i++){
    Buffer[i]=Buffer[i]&0x0fff;
    volt=(float)(Buffer[i])/32768.0*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 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 "A826.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);
    A826_Initialize(0,io_base,-1,-1);
```

```

A826_AD_SetChGainMode(0,A826_BI_1,POLLING_MODE);

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

    A826_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])/32768.0*10.0;
        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);
}
```

4.12 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 "A826.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 transferring 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);
A826_Initialize(0,io_base,-1,irq_no);
A826_AD_SetChGainMode(0,A826_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( A826_AD_INT_Start(Buffer,DesireCount,c1,c2)!=0 ) {
    putchar(0x07);
    printf("A826_AD_INT_Start() error !!!\n");
    exit(1);
  }

  do {
    NowCount=A826_AD_INT_Count();
    if (kbhit()!=0) {getch(); goto ret_label;}
  }while( NowCount<DesireCount );
  A826_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])/32768.0*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);
}

```

4.13 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 "A826.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 transferring 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);
A826_Initialize(0,io_base,dma_no,irq_no);
A826_AD_SetChGainMode(0,A826_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( A826_AD_DMA_Start(Buffer,DesireCount,c1,c2)!=0 ) {
    putchar(0x07);
    printf("A826_AD_DMA_Start() error !!!\n");
    exit(1);
  }
  while(A826_AD_DMA_Status()) {
    if (kbhit()!=0) {getch(); goto ret_label;}
  }
  A826_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])/32768.0*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);
    setttextstyle(SMALL_FONT,HORIZ_DIR,6);
}

```