

64ch 16bit USB Data Acquisition SDK

개요

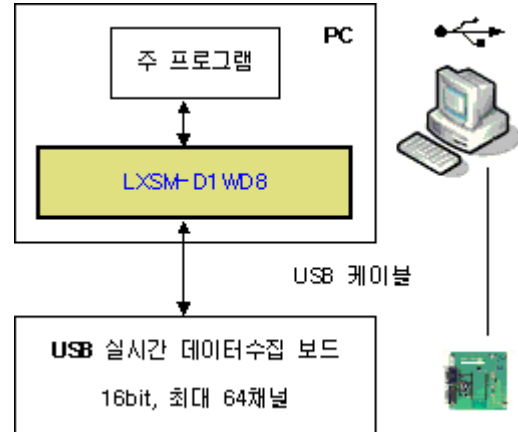
LXSM-D1WD8은 USB 기반 실시간 데이터 수집 장치를 PC에서 사용자가 임의로 제어 가능하게 하는 S/W 모듈이며, win32 API 형식의 DLL 이다. 본 DLL을 사용하여 보드의 모든 기능을 개발자의 메인 프로그램에서 사용할 수 있다.

하드웨어 관련된 모든 번거로운 코드를 DLL내에서 처리하고 있으므로, 개발자는 DLL에서 제공하는 10개의 함수와 1개의 메시지만으로 장치와 통신이 가능하며 실시간 데이터 수집이 가능하다.

D1WD8에서는 보드와의 통신을 달성하기 위하여 필요한 모든 함수 및 메시지를 제공하고 있다.

권장되는 개발 언어는 Visual C++ 6.0 이상이나, 본 DLL은 Win 32 API 형식으로 만들어 졌기 때문에 반드시 visual C++일 필요는 없다.

데이터수신 정보 전달시 사용자메시지전송방식과 키보드 이벤트 전송방식 2가지중 선택가능. - 일반 윈도우메시지 처리가 불가능한 개발툴에서는 키보드이벤트수신처리부로 구현가능.



특징

- Analog Input Channel : Max 64 ch (1,2,4,8,16,32,64 입력 아날로그신호 최대 채널 선택가능)
- Real Time Stream Data Acquisition
- Variable Sampling Frequency (Hz): 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384 가변가능.
- Variable voltage gain : 153 step (x 0 ~ x 1.5)
- 사용환경 : Win98se/Me/2000/XP
- 적용 가능한 장비 - USB 인터페이스 보드를 내장한 다음의 제품.
 - QEMG-8, WEMG-8, QEEG-16, QEEG-32, WEEG-32,

파일구성

파일명	설명
LXSM-D1WD8.DLL	DLL core, 프로젝트의 실행파일이 있는 폴더에 복사한다.
LXSM-D1WD8.LIB	프로젝트 소스파일 루트에 복사하고 프로젝트에 link 시킨다.
LXSM-D1WD8.H	프로젝트 소스파일 루트에 복사하고 메인 프로그램에서 include 시킨다.

LXSM-D1WD8

차 례

개요	1
특징	1
파일구성	1
D1WD8의 전체 작동 방식	6
스트림 데이터 처리방법 - 일반윈도우메시지 방식	8
스트림 메시지	8
스트림 데이터 처리방법 - 키보드 이벤트 전송 방식	9
키보드 이벤트 전송방식을 사용하기 위한 방법	9
스트림 메모리	11
<input type="checkbox"/> 데이터 저장 형식	11
<input type="checkbox"/> 마킹채널이란?	12
<input type="checkbox"/> 스트림 메모리에 저장된 값의 범위 및 단위	12
스트림 전송 메시지 처리 코딩 방법	15
<input type="checkbox"/> 단계1. message_map에 수동으로 메시지 처리부를 추가한다	15
<input type="checkbox"/> 단계 2. 메시지를 처리할 함수를 선언하고 정의한다	15
<input type="checkbox"/> 메시지 처리부의 주의사항	16
함수 리스트	18
<input type="checkbox"/> 전체 함수 리스트	18
함수 호출순서	19
함수별 상세설명	20
설명 규칙	20
함수명	20
<input type="checkbox"/> 함수선언	20
<input type="checkbox"/> 설명	20
<input type="checkbox"/> 인자	20
<input type="checkbox"/> 호출시점	20
<input type="checkbox"/> 리턴값 및 예러	20

LXSM-D1WD8

Init_Device	21
<input type="checkbox"/> 함수선언	21
<input type="checkbox"/> 설명	21
<input type="checkbox"/> 인자	21
<input type="checkbox"/> 호출시점	21
<input type="checkbox"/> 리턴값 및 에러	21
Start_Stream	23
<input type="checkbox"/> 함수선언	23
<input type="checkbox"/> 설명	23
<input type="checkbox"/> 호출시점	23
<input type="checkbox"/> 인자	23
<input type="checkbox"/> 리턴값 및 에러	23
Stop_Stream	24
<input type="checkbox"/> 함수선언	24
<input type="checkbox"/> 설명	24
<input type="checkbox"/> 호출시점	24
<input type="checkbox"/> 인자	24
<input type="checkbox"/> 리턴값 및 에러	24
Close_Device	25
<input type="checkbox"/> 함수선언	25
<input type="checkbox"/> 설명	25
<input type="checkbox"/> 호출시점	25
<input type="checkbox"/> 인자	25
<input type="checkbox"/> 리턴값 및 에러	25
Set_ADCMaxNumChannel	26
<input type="checkbox"/> 함수선언	26
<input type="checkbox"/> 설명	26
<input type="checkbox"/> 호출시점	26
<input type="checkbox"/> 인자	26
<input type="checkbox"/> 리턴값 및 에러	26
Set_SampleFreq	27
<input type="checkbox"/> 함수선언	27

LXSM-D1WD8

<input type="checkbox"/>	설명	27
<input type="checkbox"/>	호출시점	27
<input type="checkbox"/>	인자	27
<input type="checkbox"/>	리턴값 및 에러	27
Set_PGA		28
<input type="checkbox"/>	함수선언	28
<input type="checkbox"/>	설명	28
<input type="checkbox"/>	호출시점	28
<input type="checkbox"/>	인자	28
<input type="checkbox"/>	리턴값 및 에러	28
Set_ConfigChannel		30
<input type="checkbox"/>	함수선언	30
<input type="checkbox"/>	설명	30
<input type="checkbox"/>	호출시점	30
<input type="checkbox"/>	인자	30
<input type="checkbox"/>	리턴값 및 에러	30
Set_KeyBoardMarking		31
<input type="checkbox"/>	함수선언	31
<input type="checkbox"/>	설명	31
<input type="checkbox"/>	호출시점	31
<input type="checkbox"/>	인자	31
<input type="checkbox"/>	리턴값 및 에러	31
Get_StreamMemory		33
<input type="checkbox"/>	함수선언	33
<input type="checkbox"/>	설명	33
<input type="checkbox"/>	호출시점	33
<input type="checkbox"/>	리턴값	33
Get_StreamMemoryFP		34
<input type="checkbox"/>	함수선언	34
<input type="checkbox"/>	설명	34
<input type="checkbox"/>	호출시점	34
<input type="checkbox"/>	리턴값	34

LXSM-D1WD8

Set_MessageMode	35
<input type="checkbox"/> 함수선언	35
<input type="checkbox"/> 설명	35
<input type="checkbox"/> 호출시점	35
<input type="checkbox"/> 리턴값	35

그림 목차

그림- 1 . D1WD8의 전체 작동 방식.....	6
그림- 2. 스트림 데이터를 처리하는 스레드의 역할.....	8
그림- 3. 아날로그 신호 흐름 체계.....	13
그림- 4. gain_idx와 PGA Gain그래프.....	29

표 목차

표- 1. D1WD8의 기능별 설명 및 관련 DLL 함수.....	7
표- 2 최대 채널수에 대응하는 채널당 데이터 수 및 가능한 최대 샘플링 주파수	11
표- 3 전체 함수 리스트.....	18
표- 4 제품별 고유ID.....	22
표- 5 일부 gain_idx별 PGA의 전압이득표.	29

LXSM-D1WD8

D1WD8의 전체 작동 방식

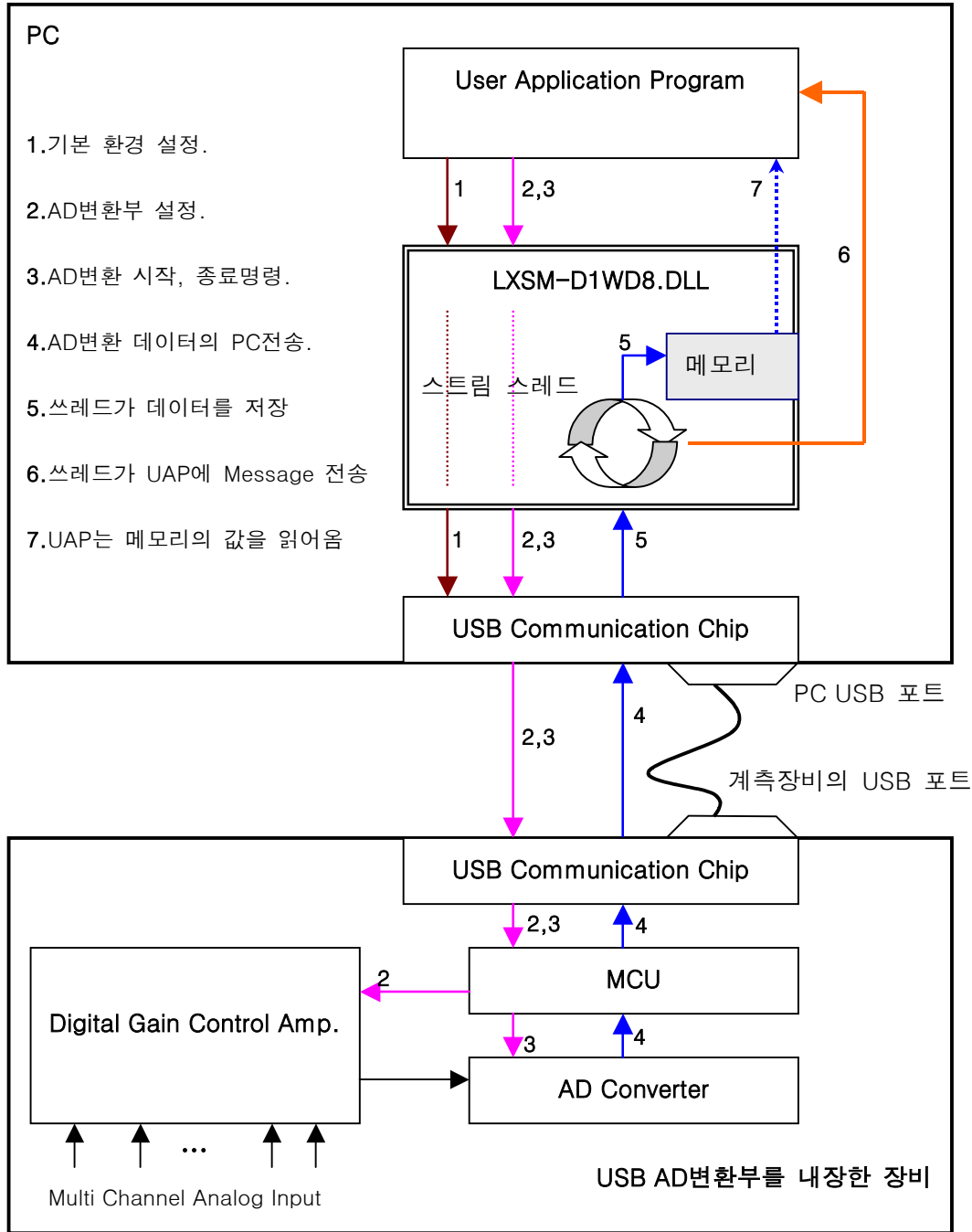


그림 - 1 . D1WD8의 전체 작동 방식.

LXSM-D1WD8

본 DLL을 사용하여 사용자의 메인 프로그램에서 장비를 제어하고 데이터를 수집하는 전체 작동상황을 그림-1 에 보이고 있으며, 아래 표-1 에 세부설명을 보이고 있다.

표- 1. D1WD8의 기능별 설명 및 관련 DLL 함수.

기능	설명	관련DLL함수
1. 기본환경설정	컴퓨터에 장착된 USB 장치를 찾고, 여러 개의 USB 장치 중에서 원하는 장치의 핸들을 잡든지 혹은 해제 하는 과정이다.	Init_Device Close_Device
2. AD변환부 설정	장비의 AD변환부의 최대 채널수 설정, 샘플링 주파수 설정, 프로그래머블 게인 앰프의 이득 설정 및 데이터를 전송 받을 채널을 선택한다..	Set_ADCMaxNumChannel Set_SampleFreq Set_PGA Set_ConfigChannel
3. AD변환시작, 종료	장비의 AD변환기의 작동을 On Off하는 것이다. AD변환을 ON 시키면 장비측의 AD변환기가 작동됨과 동시에 PC로 데이터를 전송하게 되며, DLL에서는 전송되어 오는 데이터를 메모리에 저장하고 메인 프로그램에 데이터를 전송하기 위한 스레드가 생성된다. 종료 명령에 의해 장비의 AD변환과정이 중지되고 동시에 스레드도 소멸된다.	Start_Stream Stop_Stream
4.AD변환데이터의 PC 전송	장비에서 변환된 데이터가 연속적으로 PC의 로레벨 버퍼로 저장되는 과정이다. 이 기능은 사용자가 개입될 수 없고 자동으로 진행된다.	함수 없음. 장비와 PC가 자동으로 처리함.
5.스레드가 데이터를 저장	AD변환이 시작되어 DLL에서 생성된 스레드는 연속적으로 PC로 전송되어 오는 AD변환데이터를 지정된 메모리로 저장하게 된다.	함수 없음. DLL내부에서 자동 처리됨.
6. 스레드가 UAP에 메시지 전송	단계5에서 지정된 메모리가 전부 채워지면 스레드 내에서 메인 프로그램으로 메시지를 전송한다. 사용자정의 메시지전송방식과 키보드 이벤트 전송방식 2종류가 지원되며 개발자는 2개중 1개방식을 지정해야함.	사용자정의 메시지 발생 방식. DLL에서 WM_AcqUnitData 가 WM_USER+1로 정의 되어 있다. 키보드 이벤트 발생방식. 데이터 전송해야할 상황에서 + 키 누름이벤트발생. Set_MessageMode 호출하여모드선택.
7. UAP는 메모리의 값을 읽음.	단계6에서 주기적으로 발생하는 메시지를 받은 메인프로그램은 메시지를 받은 즉시 해당 메모리에 접근하여 데이터를 가져가야 한다.	함수 없음. 개발자가 DLL에서 전송된 메시지를 처리할 수 있는 루틴을 만들어야 한다.

LXSM-D1WD8

스트림 데이터 처리방법. - 일반윈도우메시지 방식.

실시간 데이터 수집을 달성하기 위하여 본 DLL에서는 다음 그림- 2와 같은 방식으로 작동된다. DLL의 Start_Stream명령이 수행되면, 장치로 AD변환 및 데이터 전송하라는 명령이 전달되어 연속적으로 데이터가 PC로 전송되게 된다. 한편, DLL에서는 장치에서 연속적으로 PC로 수신되는 스트림 데이터를 감시하고 이후 처리하기 위하여 스트림 처리용 스레드가 생성되게 된다. 스레드의 역할은 입력되는 스트림 데이터를 스트림 메모리에 저장하고 스트림 메모리가 전부 채워지면 부모프로그램으로 메시지를 전송하게 된다. 이와 같은 동작이 무한히 반복되게 된다. 사용자 프로그램에서는 Start_Stream 명령을 내린 이후에는 메시지 처리 루틴에서 모든 데이터 처리를 수행하면 된다. 즉, 메인 프로그램에서는 스트림 메시지가 수수 될 때마다 스트림 메모리의 데이터를 가져와서 디스플레이 등의 사용자 처리를 하면 된다.

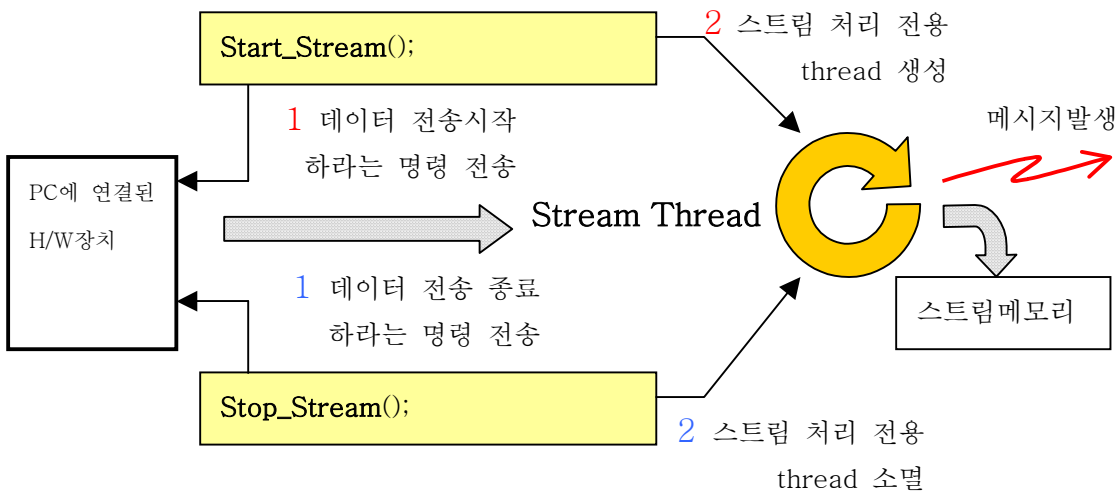


그림- 2. 스트림 데이터를 처리하는 스레드의 역할

스트림 메시지

DLL의 스트림 처리 루틴에서 발생하게 되는 스트림 메시지는 DLL내부적으로 다음과 같은 함수에 의하여 발생된다.

```
SendMessage(MSGTARGET_WINDOW, WM_AcqUnitData, 0, (LPARAM)AcqUnitData);
```

여기서 MSGTARGET_WINDOW 는 Init_Device함수의 인자로 전달된 메시지 타겟 윈도우의 핸들이며, WM_AcqUnitData 는 다음과 같이 정의되어 있다.

```
#define WM_AcqUnitData WM_USER+1
```

SendMessage의 마지막 인자인 (LPARAM)AcqUnitData 는 스트림 데이터를 저장하고 있는 스트림 메모리인 AcqUnitData 의 포인터를 부모프로그램으로 전달하기 위한 것이다. **팁!!!**: AcqUnitData 에 접근하는 방법으로 메시지로 전송된 인자를 이용하는 방법외에, Get_StreamMemory()의 반환값을 이용하는 방법도 가능함.

LXSM-D1WD8

스트림 데이터 처리방법. - 키보드 이벤트 전송 방식.

앞의 방식은 일반 윈도우 메시지(사용자 정의 메시지) 전송하는 방식이며, 이를 처리하기 위해서는 앞의 설명처럼 개발틀에서 사용자 정의 메시지처리가 가능해야한다. 만일 일반사용자정의 메시지처리가 지원되지 않고 키보드 이벤트 처리가가능한 개발도구인 경우에는 스트림전송방식을 키보드 이벤트 전송방식을 선택할 수있다.

키보드 이벤트 전송방식을 사용하기 위한 방법.

단계1: Set_MessageMode함수를 호출하면서 인자로 1을 전달하면(즉, Set_MessageMode(1);), DLL내부적으로 메시지전송시 사용자 정의메시지를 전송하는 것이나리라, 키보드를 누른효과를 주게된다. 지정된 키는 + 키이다.

만일, 이 함수가 호출되지 않은 상태의 DLL내부의 기본설정은 사용자정의메시지로 설정되어있다.

단계2: Get_StreamMemory(); 함수를 호출하여 DLL내의 스트림 메모리(실수형배열)의 주소를 받아온다.

상기 단계1, 단계2 의 해당함수를 호출한 상태에서 On_Stream() 함수가 호출되면 데이터가 전송되어야 하는 시점에 + 키를 누른 효과가 지정된 윈도우 핸들로 전송된다. 여기서 의미하는 지정된 윈도우 핸들이라함은 Init_Device() 함수 호출시 인자로 전달된 윈도우 핸들을 의미한다. 한편, 키보드 이벤트가 전송될 때 모 프로그램에서 처리해줘야 하는 코드는 이전의 사용자 정의 메시지 처리부가 아니라, 키보드 이벤트 처리부를 마련해야한다. .VC++6.0 인 경우 샘플코드는 아래와 같다.

```

/// +키 입력이벤트 핸들러.
/// !!!!! 사용자 정의 메시지로 +키를 누른 경우 오류 발생 하므로 수신처리중 키누르지 않도록 주의 !!!!!
///
void CTest_LXSM_D1WD8View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    // TODO: Add your message handler code here and/or call default
    switch(nChar)
    {
        | case VK_ADD: // + 키에 반응하게 하였다. DLL에서 + 키를 누른것처럼 키이벤트 발생함.

            /// 여기서 할 일은 데이터 가져와서 이용하는것이다.
            /// 본 루틴 실행 이전 단계에서 Get_StreamMemory 함수가 최소 한번 호출되어 pBuffer에 값을 받아와야한다.
            ACQPLOT_DLL_Array_Datain_Strip((float*)(pBuffer),DISPMAXCH+1,DISPDATANUM);

            break;
    }
    CView::OnKeyDown(nChar, nRepCnt, nFlags);
}

```

LXSM-D1WD8

앞의 이미지가 보이지는 않는 경우 아래 텍스트 참조.

```
/// +키 입력이벤트 핸들러.  
/// !!!!! 사용자가 별도로 +키를 누른 경우 오류 발생 하므로 수신처리중 키누르지 않도록 주의 !!!!!  
///  
void CTest_LXSM_D1WD8View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)  
{  
    // TODO: Add your message handler code here and/or call default  
    switch(nChar)  
    {  
        case VK_ADD: // + 키에 반응하게 하였다. DLL에서 + 키를 누른것처럼 키이벤트 발생함.  
  
            // 여기서 할 일은 데이터 가져와서 이용하는것이다.  
            // 본 루틴 실행 이전 단계에서 Get_StreamMemory 함수가 최소 한번 호출되어 pBuffer에 값을 받아둬야  
한다.  
            ACQPLOT_DLL_Array_Datain_Strip((float *) (pBuffer), DISPMAXCH+1, DISPDATANUM);  
  
            break;  
    }  
    CView::OnKeyDown(nChar, nRepCnt, nFlags);  
}
```

LXSM-D1WD8

스트림 메모리

□ 데이터 저장 형식

DLL에서 스트림 데이터를 임시 저장하는 스트림 메모리는 float형으로 선언되어 있고, 여기에 기록되어 있는 내용은 다음과 같다. 메모리의 앞부분부터 ch1,ch2,ch3,...,최대채널, 마킹용 채널 의 순서로 기록되어 있으며, 각각의 채널마다 동일한 개수의 데이터가 저장되어 있다.

스트림 메모리 : float 형으로 선언되어 있다.

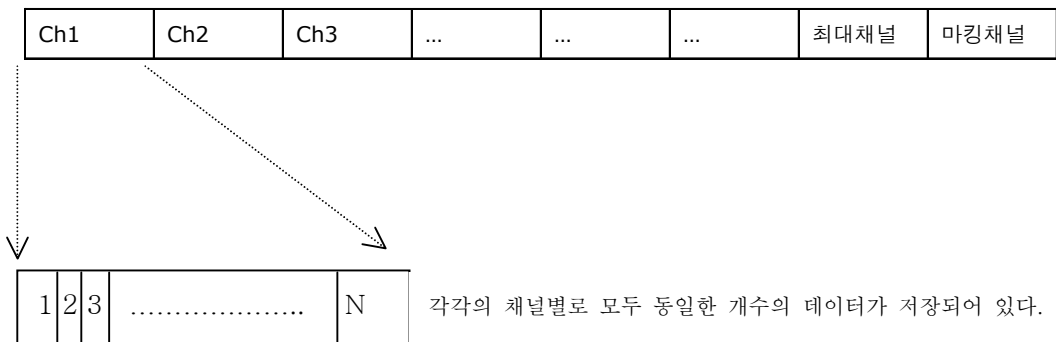


그림-3. 스트림 메모리에 데이터가 저장된 형식.

한 채널당 저장되어 있는 데이터의 수는 최대채널의 설정에 따라서 변경되는 값을 주의하여야 한다. **채널당 데이터수 = 512/ 최대채널수** 로 결정된다. 1채널부터 64채널까지 각각의 경우에 데이터 수는 다음 표-2 와 같다.

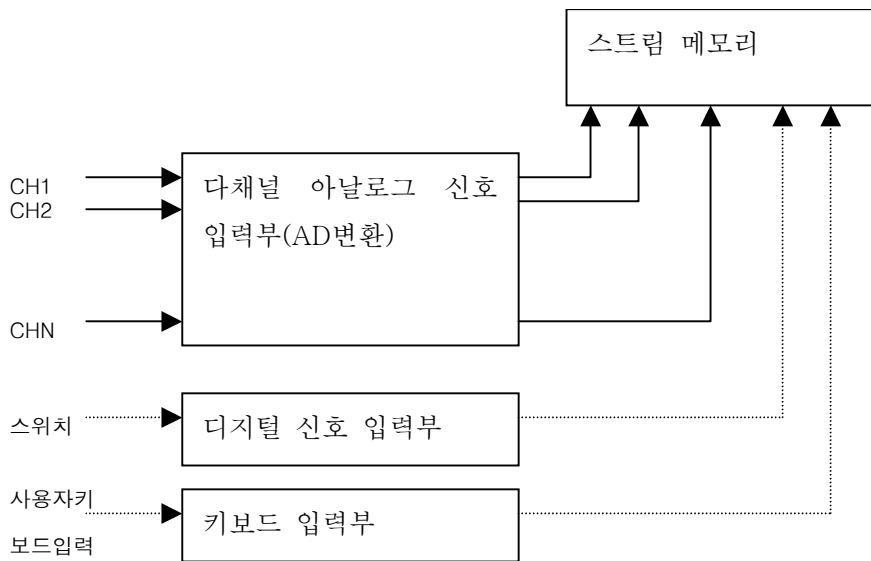
표- 2 최대 채널수에 대응하는 채널당 데이터 수 및 가능한 최대 샘플링 주파수

최대 채널수	스트림메모리의 한 채널당 데이터 수	가능한 최대 샘플링주파수
1	512	16384Hz
2	256	8192Hz
4	128	4096Hz
8	64	2048Hz
16	32	1024Hz
32	16	512Hz
64	8	256Hz

LXSM-D1WD8

□ 마킹채널이란?

스트림 메모리에 저장된 값을 보면 AD변환된 아날로그 신호 외에 마킹채널이 있음을 알 수 있다. 이것은 아날로그 신호가 아니며 장치에 하드웨어적으로 스위치를 연결하여 시점 정보를 같이 전송하거나, 키보드의 상하좌우 화살표 및 번호키패드를 사용자가 누른시점 정보를 전송하기 위한 용도로 사용된다. 그림으로 나타내면 다음과 같다. 입력되는 아날로그 신호를 AD변환하여 채널 순서대로 스트림 메모리에 기록하고 항상 마지막에는 스위치가 연결된 디지털 신호 입력부에서 값을 받아서 스트림 메모리의 마지막 부분에 스위치의 상태를 기록하게 된다.



□ 스트림 메모리에 저장된 값의 범위 및 단위.

아날로그 신호

스트림 메모리에 저장되어 있는 AD변환된 값은 항상 -1.25 에서 $+1.25$ 사이의 값을 갖는다. 단위는 볼트(V) 이다. -1.25 에서 $+1.25$ 는 PGA의 게인 값을 변경한 경우에도 값의 한계치는 변하지 않는다. 즉, 다음 그림-3 과 같은 입력 시스템에서 스트림 메모리에 저장되는 값은 ADC입력단의 값이다.

LXSM-D1WD8

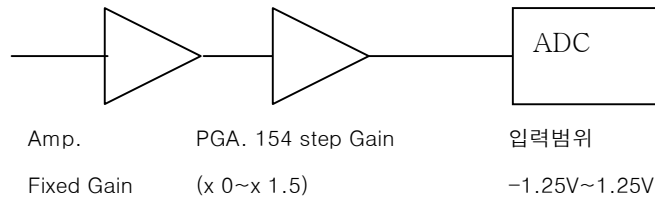


그림- 3. 아날로그 신호 흐름 체계

스위치 신호

장치외부에 연결된 스위치가 OFF된 상태를 1로 기록하며, ON상태는 0으로 기록된다. 혹은 장치의 BNC로 TTL 로직신호를 인가한 경우 인가한 로직값을 그대로 반영한다.

키보드 신호

사용자가 키보드를 누른경우 누른시점에 키의 종류에 따라 아래와 같은 값이 마킹채널에 기록된다.
지원되는 키 : 화살표 및 키패드의 번호키.

키	마킹채널에 기록되는 번호.
좌 화살표.	2
우 화살표.	3
위 화살표.	4
아래 화살표.	5
0	10
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19

!!! 키보드 신호마킹기능 이용시 주의할 사항.

본 DLL을 임포트 시켜서 사용하는 프로그램이 포커싱을 잃게되면 키입력을 마킹하지 못하게 되며, 한번 포커싱을 잃은경우 다시

LXSM-D1WD8

해당 프로그램창이 활성화 되어도 키보드 마킹기능이 작동되지 않는다. 이런상황이 벌어진 경우 반드시

Set_KeyBoardMarking (1); 함수를 호출하여 키보드 마킹기능을 활성화 시켜야만 키보드 입력을 마킹가능하다.

LXSM-D1WD8

스트림 전송 메시지 처리 코딩 방법.

앞에서 설명한 스트림 데이터를 처리하는 부분에 대하여 Visual C++ 6.0을 예로 view class에서 코딩하는 것으로 간주하여 설명한다.

□ 단계 1. message_map에 수동으로 메시지 처리부를 추가한다.

```

BEGIN_MESSAGE_MAP(CTest_LXSM_D1WD5Uview, CView)
    //{AFX_MSG_MAP(CTest_LXSM_D1WD5Uview)
    ON_COMMAND(ID_MENU_InitDevice, OnMENUInitDevice)
    ON_COMMAND(ID_MENU_CloseDevice, OnMENUCloseDevice)
    ON_COMMAND(ID_MENU_Sample128, OnMENSUSample128)
    ON_COMMAND(ID_MENU_Sample256, OnMENSUSample256)
    ON_COMMAND(ID_MENU_Sample512, OnMENSUSample512)
    ON_COMMAND(ID_MENU_Sample1024, OnMENSUSample1024)
    ON_COMMAND(ID_MENU_Sample2048, OnMENSUSample2048)
    ON_COMMAND(ID_MENU_Sample4096, OnMENSUSample4096)
    ON_COMMAND(ID_MENU_SetPga, OnMENUSetPga)
    ON_COMMAND(ID_MENU_StartStream, OnMENUStartStream)
    ON_COMMAND(ID_MENU_StopStream, OnMENUStopStream)
    ON_COMMAND(ID_MENU_SetMaxNumChannel, OnMENUSetMaxNumChannel)
    ON_COMMAND(ID_MENU_Sample8192, OnMENSUSample8192)
    ON_COMMAND(ID_MENU_Sample16384, OnMENSUSample16384)
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_AcqUnitData, OnStreamData) // !!!! STREAM데이터 메시지 처리.
END_MESSAGE_MAP()

```

위와 같이 메시지 처리부를 수동 설정하여, 프로그램으로 하여금 WM_AcqUnitData라는 메시지를 받으면 OnStreamData라는 함수를 실행하라는 설정을 여기서 하고 있는 것이다. WM_AcqUnitData는 LXSM-D1WD8.h에 이미 선언되어 있다. 한편, OnStreamData는 우리가 선언하고 정의해야 할 함수 있다. 다음 단계2를 보자.

□ 단계 2. 메시지를 처리할 함수를 선언하고 정의한다.

...View.h에 메시지 처리 함수를 선언하고 있다.

```

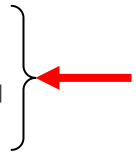
// Operations
public:
    CString y_text[65];
    long OnStreamData(WPARAM wParam, LPARAM lParam);

```

LXSM-D1WD8

이제 view.cpp에서 함수를 정의한다.

```
long CTest_LXSM_D1WD5View::OnStreamData(WPARAM wParam,LPARAM lParam)
{
    // 파형디스플레이 SWEEP 루틴임.
    ACQPLOT_DLL_Array_Datain_Strip((float *)lParam),MAXNUMBER_CHANNEL+1
    return 1;
}
```



이제 위 함수 내에서 원하는 코딩을 하면 된다. 본 예제에서는 파형 디스플레이 루틴이 실행되고 있는 상황이다. 여기서 스트림 메모리로 접근하는 방법을 주의해서 보아야 한다. DLL에서 메시지가 전송될 때 인자로써 스트림 메모리의 포인터가 전송되고 있으며, 이것을 사용하기 위해서는 lParam으로 받아야 한다. 위 예제에서 (float *) (lParam) 의 형태로 사용되고 있다.

□ 메시지 처리부의 주의사항.

위 메시지 처리 함수는 DLL에서 메시지가 발생할 때 마다(그림 -2참조) 자동으로 호출되는 부분이다. 실시간 처리를 위한 코드 작성과정에서 가장 주의 해야 하는 부분이다. 즉, 메시지 처리 루틴에는 처리속도가 느린 것을 수행시키면 데이터를 잃어버리는 사태가 벌어진다. 즉, 다음 메시지가 발생하기 전에 모든 처리가 이뤄져야 하는데, 메시지 발생 주기는 최대 채널수 및 샘플링 주파수 설정에 따라 다른 값을 갖는다. 다음 표는 최대 채널 8로 한 경우, 샘플링 주파수 별로 메시지 발생 시 간격을 보이고 있다.

샘플링 주파수	메시지 발생 시간격 (초)
32Hz	2
64Hz	1
128Hz	0.5
256Hz	0.25
512Hz	0.125
1024Hz	0.0625
2048Hz (최대)	0.03125

예로, 512Hz로 샘플링 한다면 다음 메시지가 도착하기 전까지는 0.125초의 시간적인 여유가 있으며, 이 시간동안 스트림 메모리에 접근하여 채널 당 64개(총 8 x 64 의 float 형 데이터개수)의 데이터를 확보하여 0.125초 이내에 파형 디스플레이라든지 혹은 별도의 사용자 메모리에 데이터 이전 작업을 끝내야만 한다. 이것은 실시간 처리가 필요한 상황에서는 피할 수 없는 문제이다.

LXSM-D1WD8

만일 시간적으로 긴 시간이 소요되는 계산이 필요한 경우에는 실시간으로 전송되어 오는 데이터를 일단 지정 메모리에 모아두고 나서 Stop_Stream 명령을 내려 DLL의 AD변환을 종료 시키고 나서 이후 계산을 해야만 한다. 계산 종료 후 다시 Start_Stream 명령을 내려 데이터 수집을 시작한다.

메시지 발생 시 간격에 영향을 주는 인자는 최대 채널수와 샘플링 주파수 2 가지이다. 이 2가지 인자에 의하여 메시지 발생 시간격을 계산하는 일반적인 식은 다음과 같다. 아래 식에서 샘플링 주파수는 Hz단위이다.

$$\text{메시지 발생시간격(초)} = (512/\text{최대채널수}) \times (1/\text{샘플링 주파수})$$

LXSM-D1WD8

함수 리스트

□ 전체 함수 리스트

D1WD8은 하드웨어 관련된 모든 복잡성을 DLL에서 처리하고 있으므로, 사용자는 다음 표-3 과 같이 단 8개의 함수 만으로 장치와 통신 가능하다.

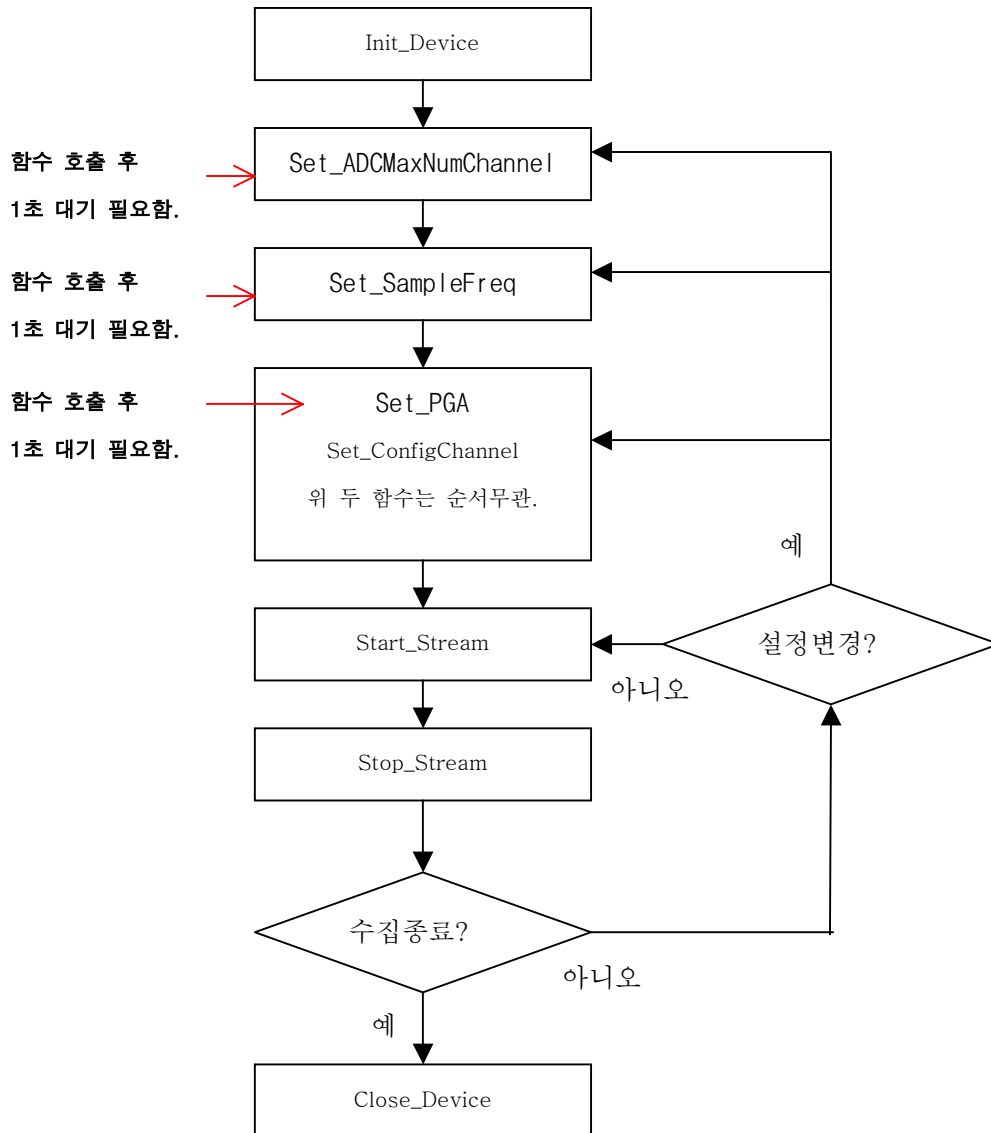
표- 3 전체 함수 리스트

Function	개요
short Init_Device(HWND msgtarget_window, int pid);	DLL 초기화 및 DLL에서 발생한 메시지를 받을 메인 프로그램의 핸들과 PC에 장착된 장치의 고유ID를 전달한다.
short Start_Stream();	장치의 AD변환부를 작동시키는 명령이다.
short Stop_Stream();	장치의 AD변환을 종료하라는 명령이다.
short Close_Device();	장치의 디바이스 핸들을 놓아주고, 설정하였던 메모리를 모두 해제 한다.
short Set_ADCMaxNumChannel (unsigned char maxnum_channel);	AD변환할 최대 채널 수를 지정한다. 1,2,4,8,16,32,64가능.
short Set_SampleFreq(unsigned char samplefreq_idx);	장치의 샘플링 주파수를 설정한다.
short Set_PGA(unsigned char gain_idx);	장치에 내장된 PGA(Programmable Gain amp)의 게인값을 변경한다.
short Set_ConfigChannel (unsigned char *ls_Select_Channel);	데이터를 받을 채널을 선택한다. 기본설정은 전부 선택이다.
short Set_KeyBoardMarking (unsigned char on_off);	키보드 마킹기능을 활성화/비활성화 설정함수. 인자로 1을 전달하면 활성화. 0을 전달하면 비활성화이다.

LXSM-D1WD8

함수 호출순서

전체적으로 함수는 호출 순서는 아래 그림과 같다. 가장 주의할 것은 Init_Device이후에 항상 Set_ADCMaxNumChannel을 호출하여 자신이 원하는 최대 채널을 설정하여야만 하고, 최대 채널 수 변경한 후에는 반드시 Set_SampleFreq를 호출하여 샘플링 주파수를 설정하여야 한다. 또 한가지 주의할 사항은 명령 들 중에서 장비에 내장된 MCU와 통신 완료하는데 0.5초 이상의 시간이 소요되는 함수들인 Set_ADCMaxNumChannel, Set_SampleFreq, Set_PGA 함수는 함수 호출이후 메인프로그램에서 1초 이내에는 다른 명령을 수행하지 않아야 한다. 한편, Set_KeyBoardMarking함수는 Init_Device이후이면 언제든지 임의로 호출가능한 함수다.



LXSM-D1WD8

함수별 상세설명

설명 규칙.

각각의 함수별로 모두 동일하게 다음의 설명방식을 따르고 있다.

함수명,

□ 함수선언

```
short Init_Device(HWND msgtarget_window);
```

□ 설명

함수사용을 위한 설명이 제시된다.

-

□ 인자

- 함수에서 인자가 있는 경우 각각의 입출력 값 전달 방법을 설명한다.

□ 호출시점

함수 호출시점을 설명한다.

□ 리턴값 및 에러

- 리턴값에 대한 설명이다. 모든 함수 공통적으로 음수가 반환 경우 모두 에러 상황이다.

LXSM-D1WD8

Init_Device

□ 함수 선언

```
short Init_Device(HWND msgtarget_window, int pid);
```

□ 설명

DLL 초기화 및 DLL에서 발생한 메시지를 받을 메인 프로그램의 핸들을 전달하고, 장치의 고유 Id를 인수로 전달한다.

DLL 초기화 과정에서 PC에 연결된 USB장치 중에서 인수로 전달된 장치의 고유 ID에 해당하는 장치가 장착되어 있는지 확인하고, 연결되어 있는 경우 이 장치를 사용하기 위하여 장치의 핸들을 확보한다. 이 함수는 장치를 사용하려고 할 때 한번 호출되고 Close_Device 전에는 중복 호출할 수 없다. 장치의 고유 ID는 제품별로 다르게 할당되어 있다. 제품별 고유 ID는 표-4에 제시되어 있다.

Init_Device호출하고 난 이후의 장치의 초기 상태는 최대 AD변환 채널 수 8, 샘플링 주파수 2048Hz 로 설정되어 있다.

□ 인자

HWND msgtarget_window

- I/O : 입력
- 설명: D1WD8에서 발생하는 스트림 메시지를 받을 윈도우 핸들을 전달한다.

int pid

- I/O : 입력
- 설명 : 장치의 고유 식별자를 입력한다. PC에 장착된 USB 장치 중에서 본 식별자를 이용하여 해당 장치를 찾는다. 다음 페이지의 표-4에 제품모델별 고유 ID를 보이고 있다.

□ 호출시점

메인 프로그램 실행 후 처음 한번만 호출하고 그 다음 호출은 Close_Device를 호출하고 난 후에 가능하다. Close_Device 호출 전 중복호출 불가능.

□ 리턴값 및 에러

- 1 : 초기화가 성공적으로 이뤄진 경우.
- 0보다 작은 경우는 에러 상황이며 다음과 같다.
 - -1 : PC에 장착된 장치를 발견하지 못한 경우.
 - -2 : 이미 1번 호출되어 성공적으로 초기화가 이뤄진 상태에서 Close_Device 호출 전에 중복 호출된 경우.
 - -3 : 초기화 함수 내부적으로 장치 핸들 확보 성공하면 장치로 명령을 전달하는데 이 과정에서 장치로의 명령전달 오류가 발생한 경우

LXSM-D1WD8

표- 4 제품별 고유ID.

제품명(모델명)	고유 ID
QEMG-8 (LXM3208)	1
WEMG-8 (LXM3208-RF)	2
QEEG-16 (LXM3216)	3
QEEG-32 (LXE3232)	4
WEEG-32 (LXE3232-RF)	5

LXSM-D1WD8

Start_Stream

□ 함수선언

```
short Start_Stream();
```

□ 설명

장치의 AD변환부를 작동시키는 명령이다. 이 명령이 수행되면 장치는 연속적으로 데이터를 PC로 전송하여 PC의 로레벨 버퍼로 데이터가 전송된다. 한편, 본 DLL에서는 연속적으로 PC로 들어오는 데이터를 감시하고 메인 프로그램으로 데이터를 전송하기 위하여 스레드가 생성된다. 스레드의 역할을 그림으로 나타낸 것이 그림-2이다. 본 함수가 호출되면 DLL 내부적으로 1개의 스레드가 생성되어 장비로부터 실시간으로 데이터를 스트림 메모리에 저장한다. 스트림 메모리가 전부 채워지면, 메인 프로그램은 스트림 메모리에 접근하여 데이터를 전부 가져가야 한다. 메인 프로그램이 언제 데이터를 가져가야 하는지를 알리기 위하여 스레드는 스트림 메모리가 채워진 시점마다 메시지를 메인 프로그램으로 전송한다. 즉, 메인 프로그램은 스레드에서 전송되어 오는 메시지를 처리하기 위한 루틴을 만들어 두어야 한다.

□ 호출시점

Init_Device - Set_ADCMaxNumChannel 호출이후, 중복 호출 불가능. Stop_Stream과는 교번식 호출은 제한 없음.

□ 인자

없음.

□ 리턴값 및 에러

- 1 : 정상 작동된 경우.
- 0 보다 작은 경우 에러
 - -1 : Init_Device호출되기 전에 이 함수 호출한 경우.
 - -2 : 이미 이 함수 호출되어 스트림 처리 스레드가 수행 중인 경우.
 - -3 : 장치로 스트림데이터 전송하라는 명령이 전달되지 못한 경우.
 - -4 : 현재 설정된 최대 채널 수에서 지원되지 않는 높은 샘플링 주파수로 데이터 수집을 하려고 할 때 발생. 표-2를 참조하여 낮은 값으로 재설정 해야한다.

LXSM-D1WD8

Stop_Stream

□ 함수 선언

```
short Stop_Stream();
```

□ 설명

장치의 AD변환을 종료하라는 명령이다.

DLL 내부적으로는 Start_Stream에서 만들어졌던 쓰레드를 종료시킨다. 더 이상 메시지가 부모프로그램으로 전송되지 않는다.

□ 호출시점

Init_Device 호출되고 난 후면 중복 호출되어도 안전하다.

□ 인자

없음.

□ 리턴값 및 에러

- 1 에러가 없는 경우.
- 0 보다 작은 경우는 에러 상황
 - -1: Init_Device 호출되기 전에 이 함수 호출한 경우.
 - -3: 장치로 스트림 데이터 전송 중지명령이 보내지지 않은 경우.

LXSM-D1WD8

Close_Device

□ 함수 선언

```
short Close_Device();
```

□ 설명

장치의 디바이스 핸들을 놓아주고, 설정하였던 메모리를 모두 해제 한다. 만일 Start_Stream이 호출되어 내부적으로 스트림 스레드 작동 중에 이 함수가 호출되면, Close_Device 함수 내부적으로 Stop_Stream 함수가 호출되어 스레드를 종료 시키고 장치를 해제 시킨다.

□ 호출시점

Init_Device 호출 성공 후.

□ 인자

없음

□ 리턴값 및 에러

- 1 : 정상 수행되었음.
- 0보다 작은 경우는 에러 상황
 - -1 : Init_Device호출되기 전에 이 함수 호출한 경우.
 - -3 : Start_Stream 함수가 정상 호출되어 DLL내부적으로 스레드 작동 중에 이 함수가 호출되면 내부적으로 Stop_Stream이 호출되는데 Stop_Stream수행 중 장치로 명령 전송이 실패한 경우.

LXSM-D1WD8

Set_ADCMaxNumChannel

□ 함수선언

```
short Set_ADCMaxNumChannel(unsigned_char maxnum_channel);
```

□ 설명

장치의 AD변환부에 인가하는 최대 아날로그 채널 수를 지정한다. 본 함수에서는 1,2,4,8,16,32, 64ch 까지 허용되지만, 실제 PC에 연결된 장치가 무엇이나에 따라서 값을 설정하여야 한다. 예로 제품명 QEMG-8인 경우 아날로그 8채널 장비이므로, 이 경우 최대 채널수는 8로 설정하여야 한다. 그러나, 본 함수에서 32를 설정하여도 오류는 발생하지 않는다. 단 32를 설정한 경우 앞부분 8채널은 의미 있는 데이터가 전송되어 오는 것이고, 나머지 32채널은 무의미한 데이터이다.

□ 호출시점

Init_Device가 호출되고 난 이후 항상 이 함수를 이용하여 채널 설정부터 해야한다. 중복 호출 가능. Start_Stream 수행 중에는 호출 불가. 이 함수가 호출된 다음에는 통상 샘플링 주파수를 재설정해야 한다. 예로 최대 채널 1, 샘플링 주파수 16384Hz로 설정된 상태에서 데이터 수집하다가 최대 채널 32로 변경하여 Start_Stream을 시도하면 샘플링 주파수 16384Hz로 시도하게 된다. 이것은 심각한 오류 상황이며, 최대 채널 32인 경우 최대 샘플링 주파수는 512Hz이므로 (표-2 참조) 이 값보다는 작게 설정 하여야 한다.

□ 인자

unsigned_char maxnum_channel

- I/O : 입력
- 설명 : 최대 채널수를 지정한다. 1,2,4,8,16,32,64 중의 한 값만 가능하다.

□ 리턴값 및 에러

- 0 보다 큰 경우 : 정상 작동된 경우에 0보다 큰값이 반환되면 반환되는 값은 현재 설정된 최대 채널수인 경우 각 채널 당 스트림메모리에 저장된 데이터 수가 반환된다. 예 : 32채널인 경우 16이 반환된다. 표-2 참조.
- 0 보다 작은 경우 에러
 - -1 : Init_Device호출되기 전에 이 함수 호출한 경우.
 - -2 : 스트림 처리 쓰레드가 수행 중인 경우.
 - -3 : 장치로 명령 전달 실패한 경우.
 - -10 : 전달된 최대 채널수가 1,2,4,8,16,32,64가 아닌 경우.

LXSM-D1WD8

Set_SampleFreq

□ 함수 선언

```
short Set_SampleFreq(unsigned char samplefreq_idx);
```

□ 설명

샘플링 주파수를 설정한다.

□ 호출시점

Init_Device 성공 후

□ 인자

unsigned char samplefreq_idx

- I/O - 입력
- 값설정방법 : 샘플링 주파수 = $2^{\text{samplefreq_idx}}$, 예: 512 Hz로 설정하고 싶은 경우 samplefreq_idx에는 9를 입력한다.
- 가능한 값 : 0과 14사이의 정수 예 : 7(128Hz 로 설정) ,8(256Hz로 설정) ,9(512Hz로 설정) , 10(1024Hz) , 14(16384Hz)

□ 리턴값 및 에러

- 1: 정상 수행되었음.
- 0보다 작은 경우 에러.
 - -1 : Init_Device가 정상적으로 수행되지 않은 상태에서 본 함수 호출되었다.
 - -3 : 장치로 명령전달 실패.
 - -4 : 현재 설정된 최대 채널 수에서 지원되지 않는 높은 샘플링 주파수로 데이터 수집을 하려고 할 때 발생. 표-2를 참조하여 낮은 값으로 재설정 해야 한다.
 - -10 : 인자로 전달된 값의 범위가 맞지 않다.

LXSM-D1WD8

Set_PGA

□ 함수 선언

```
short Set_PGA(unsigned char gain_idx)
```

□ 설명

장치에 내장된 PGA(Programmable Gain amp)의 게인값을 변경한다. gain_idx의 값으로 PGA의 전압이득을 설정한다.

전압이득 계산식은 gain_idx별로 아래와 같다.

$$\text{PGA_Gain} = +\text{gain_idx}/(255-\text{gain_idx})$$

한편, 위 식은 앞의 그림-3에서와 같이 ADC앞단에 놓여있는 PGA의 이득을 조절하는 것이며, 그 앞단의 전압이득이 얼마냐에 따라서 시스템의 전체 전압이득이 결정된다. 제품별로 고정전압이득은 해당제품의 매뉴얼에 명시되어 있다. 이값은 fixed_gain이라고 하면, 시스템의 전체 이득은 위 PGA_Gain과 fixed_gain의 곱으로 결정된다. 그림-4에 gain_idx별 PGA Gain의 그래프가 제시되어 있고, 표-5에 일부 gain_idx별 전압이득을 보이고 있다.

□ 호출시점

Init_Device 성공 후

□ 인자

unsigned char gain_idx

- I/O - 입력
- 값의 범위 : 0 ~ 153.

□ 리턴값 및 에러

- 1: 정상 수행되었음.
- 0보다 작은 경우 에러다. 에러의 내용은 다음.
 - -1 : Init_Device가 정상적으로 수행되지 않은 상태에서 본 함수 호출되었다.
 - -3 : 장치로 명령전달 실패다.
 - -10 : 인자로 전달된 값의 범위 오류.

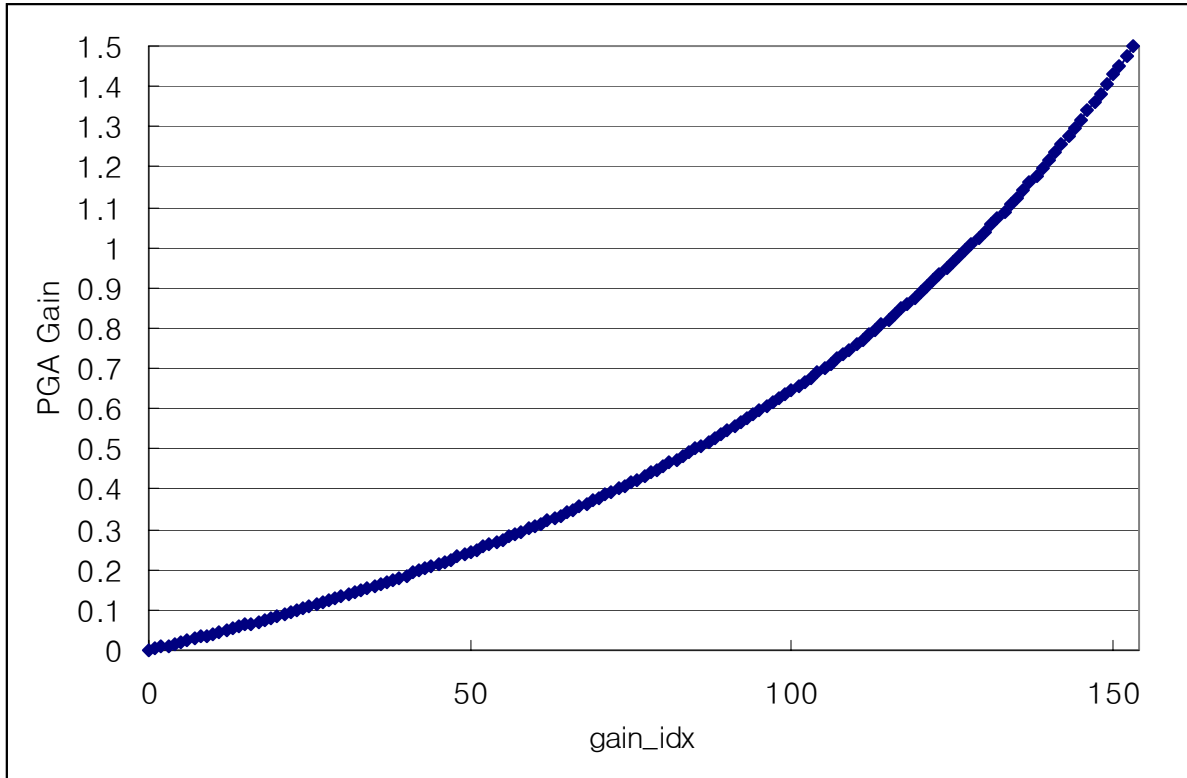
LXSM-D1WD8

그림- 4. gain_idx와 PGA Gain그래프.

표- 5 일부 gain_idx별 PGA의 전압이득표.

Gain_idx	PGA Gain
0	0
5	0.02
15	0.0625
51	0.25
55	0.275
85	0.5
105	0.7
127	0.99
128	1.00
136	1.125
153	1.5

LXSM-D1WD8

Set_ConfigChannel

□ 함수 선언

```
short Set_ConfigChannel(unsigned char * Is_Select_Channel);
```

□ 설명

데이터를 수집할 채널을 설정한다. 인자 Is_Select_Channel 로 전달하는 배열의 값이 1이면 해당채널 선택, 0이면 제외시킨다. 본 함수 호출하지 않으면 모든 채널이 선택된 것으로 설정된다.

배열의 인덱스 0 이 채널1을 의미함을 주의해야 한다. 예로 ch3, ch6만 데이터를 받고 나머지는 받고 싶지 않은 경우 Is_Select_Channel[2]=1; Is_Select_Channel[5]=1; Is_Select_Channel[32]=1; (마킹채널도 선택 가능하게 되어 있다.)로 설정하고 나머지 배열값은 모두 0으로 하여 이 함수를 호출하면 된다. 또한, 이 과정에서 주의할 점은 채널을 선택한 상황에 따라서 스트림 메모리(그림-3)에 저장되는 데이터의 순서가 변경된다는 것이다. Ch3, ch6만을 선택하였다면 스트림메모리에는 다음과 같이 데이터가 저장된다.

스트림 메모리 : 데이터 저장순서가 변경된다.

Ch3	Ch6	마킹채널	무의미	무의미	무의미	...	무의미
-----	-----	------	-----	-----	-----	-----	-----

□ 호출시점

Start_Stream 이 수행되지 않을 때.

□ 인자

unsigned char * Is_Select_Channel

- I/O : 입력

□ 리턴값 및 에러

- 1: 정상 수행되었음.
- 0보다 작은 경우 에러다. 에러의 내용은 다음.
 - -2 : 스트림 처리 쓰레드가 수행 중인 경우.

LXSM-D1WD8

Set_KeyBoardMarking

□ 함수 선언

```
short Set_KeyBoardMarking(unsigned char on_off);
```

□ 설명

사용자가 키보드 입력 정보를 마킹채널에 표시하는기능을 사용할지 말지를 설정한다.

인자로 전달되는 값이 1인 경우 키보드 입력이 마킹채널에 기록되며, 인자로 0을 전달하면 키보드 입력이 마킹채널에 기록되지 않는다. 키보드 입력은 다음 페이지의 지정된 키만 입력가능하다.

□ 호출시점

Init_Device 호출성공이후 언제든지 호출가능.

□ 인자

unsigned char on_off

- I/O : 입력
- 1: 키보드 입력 마킹기능을 활성화. 0: 키보드 입력마킹기능을 비활성화.

□ 리턴값 및 에러

- 1: 정상 수행되었음.
- 0보다 작은 경우 에러다. 에러의 내용은 다음.
 - -1 : Init_Device호출이전에 본 함수가 호출된 경우.
 - -10 : 인자로 전달된 값의 범위가 0또는 1이 아닌경우.

!!!! 키보드 신호마킹기능 이용시 주의할 사항.

본 DLL을 임포트 시켜서 사용하는 프로그램이 포커싱을 잃게되면 키입력을 마킹하지 못하게 되며, 한번 포커싱을 잃은경우 다시 해당 프로그램창이 활성화 되어도 키보드 마킹기능이 작동되지 않는다. 이런상황이 벌어진 경우 반드시

Set_KeyBoardMarking (1); 함수를 호출하여 키보드 마킹기능을 활성화 시켜야만 키보드 입력을 마킹가능하다.

LXSM-D1WD8

키	마킹채널에 기록되는 숫자 (float형 실수임)
좌 화살표.	2
우 화살표.	3
위 화살표.	4
아래 화살표.	5
0	10
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19

LXSM-D1WD8

Get_StreamMemory

□ 함수 선언

```
unsigned long Get_StreamMemory();
```

□ 설명

메시지로 전송되는 인자를 이용하여 DLL내의 float형 배열에 접근하는 방법외에, 이 함수를 호출하여 반환된 값을 이용하여 DLL 내의 배열에 접근이 가능하다.

□ 호출시점

언제든지 호출가능.

□ 리턴값

- DLL내의 float형 배열의 포인터가 반환됨.

사용예제. (DLL과 같이 배포된 Test_LXSM_D1WD8 소스코드에 사용예제 있음.)

```
unsigned long pBuffer; //변수 선언해두고.
```

```
pBuffer = Get_StreamMemory(); // 여기서 pBuffer로 DLL내의 float형 배열포인터를 받았다.
```

```
(float *)pBuffer ; //pBuffer를 사용하는 곳에서 float형 포인터임을 명시하면서 사용함.
```

LXSM-D1WD8

Get_StreamMemoryFP

□ 함수 선언

```
Float * Get_StreamMemory();
```

□ 설명

메시지로 전송되는 인자를 이용하여 DLL내의 float형 배열에 접근하는 방법외에, 이 함수를 호출하여 반환된 값을 이용하여 DLL 내의 배열에 접근이 가능하다.

□ 호출시점

언제든지 호출가능.

□ 리턴값

- DLL내의 float형 배열의 포인터 float* 타입으로 반환됨.

사용예제. (DLL과 같이 배포된 Test_LXSM_D1WD8 소스코드에 사용예제 있음.)

```
Float * fpBuffer; //변수 선언해두고.
```

```
fpBuffer = Get_StreamMemoryFP(); // 여기서 fpBuffer로 배열포인터를 받았다.
```

LXSM-D1WD8

Set_MessageMode

□ 함수 선언

```
short Set_MessageMode(unsigned char keybd);
```

□ 설명

스트림전송시 사용자 정의 이벤트 방식으로 할지, 키보드이벤트 방식으로 발생할지를 설정하는 함수. 이 함수 호출하기 전의 디폴터설정은 사용자정의이벤트 방식이다. 인자로 1을 전달하여 함수호출하면 키보드 이벤트 방식으로 작동되며, 0을 인자로 전달하면 사용자정의 이벤트방식으로 설정된다.

□ 호출시점

Init_Device이후. On_Stream중에는 호출불가.

□ 리턴값

- 1이면 정상수행. 음수이면 오류.
- -1: Init_Device호출이전에 본 함수가 호출된 경우.
- -2: 스트림전송중에 본함수가 호출된 경우.

LXSM-D1WD8

“Measuring is Believing”

락사는 정밀계측H/W, 신호처리 S/W 개발 전문 업체입니다.

락사는 전산화된 실시간 생체계측장비, 분석S/W를 제공합니다.

락사는 인체 대상의 연구, 임상분야에 최적 솔루션을 제공합니다.

LAXTHA Inc.

Advanced Scientific Instruments H/W & S/W

www.laxtha.com

general@laxtha.com

Copyright Notice

This product data sheet is the original work and copyrighted property of LAXTHA Inc. Reproduction in whole or in part must give clear acknowledgement to the copyright owner.