계측장비제어

(Communication)

A. 개 요

- 이번 장에서는 LabVIEW를 사용한 계측기 운용의 여러 가지 선택 사양을 소개한다.

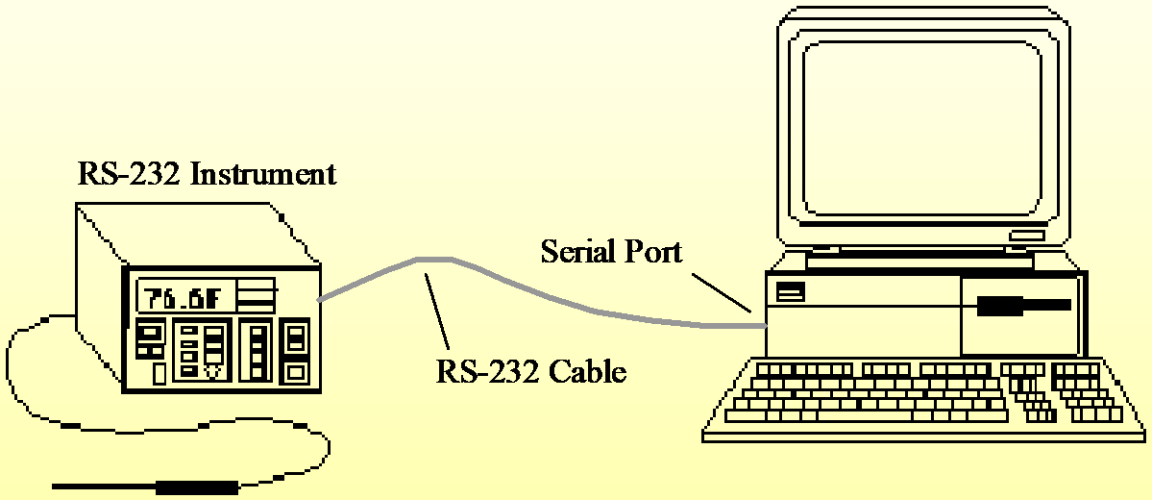
- LabVIEW로
  - Serial I/O,
  - GPIB I/O,
  - VISA I/O

  를 사용하는 방법을 배울 것이다.

- LabVIEW에서 제공하는 instrument 드라이버

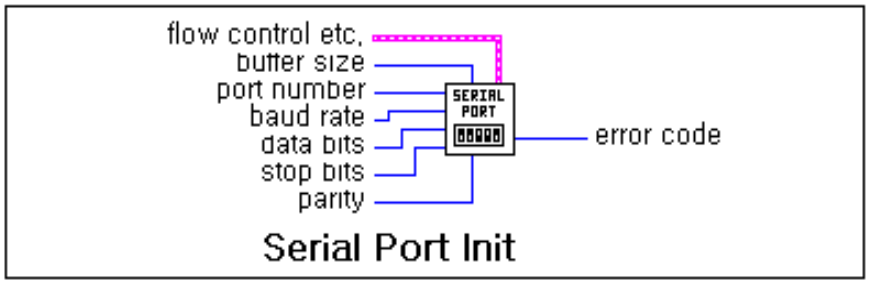
B. Serial 통신

- Serial 통신은 컴퓨터와 계측 장비 또는 컴퓨터 간의 중요한 데이터 전송 방법 중의 하나.
- serial 통신은 1회에 1bit의 데이터를 receiver로 전송하기 위해 transmitter를 사용한다. 이 방법은 데이터 전송 속도가 느리거나 데이터를 먼 거리에 전송하고자 할 때 사용.

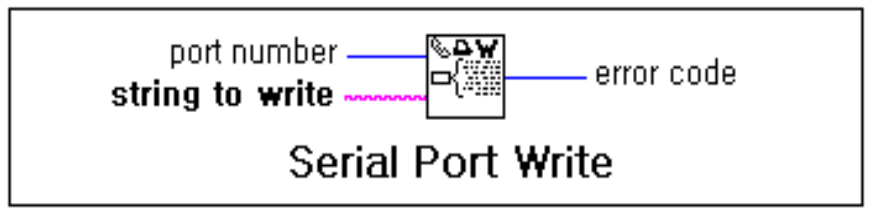


- Serial 통신의 단점은 1개의 serial포트가 1개의 device만을 제어.
  - 일부 주변 장비는 string데이터를 종료하기 위해 특수 문자열을 요구하기도 한다. 일반적인 종료 문자는 carriage return, line feed, semicolon 등. 종료 문자가 필요한지는 장비 책자를 참조.

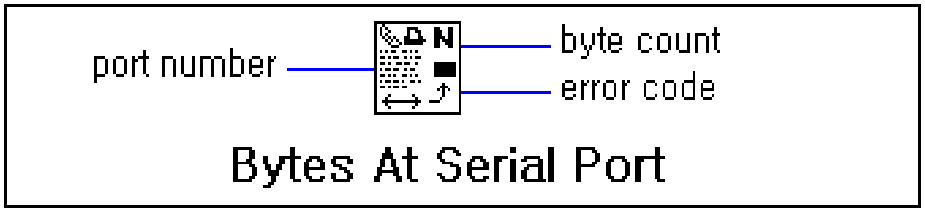
- LabVIEW의 Instrument I/O ▸ Serial 라이브러리는 serial 포트 운용에 필요한 함수들을 갖고있다



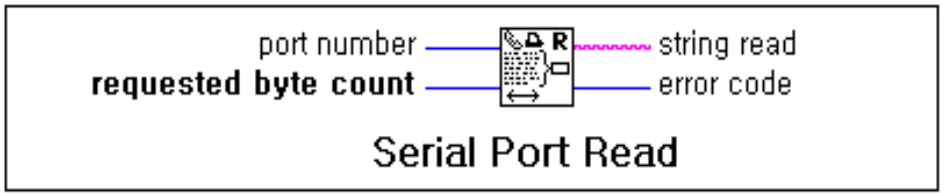
- 선택한 serial 포트를 지정된 값으로 초기화.
  - Flow control은 handshaking 변수들을 설정.
  - Buffer size는 VI가 할당하는 입력과 출력 버퍼 크기.
  - Port number는 통신에 사용되는 포트.
  - Baud rate, data bit, stop bit, parity는 지정된 포트의 변수들을 설정



- port number가 표시하는 serial 포트에 string(명령어 데이터)을 쓴다

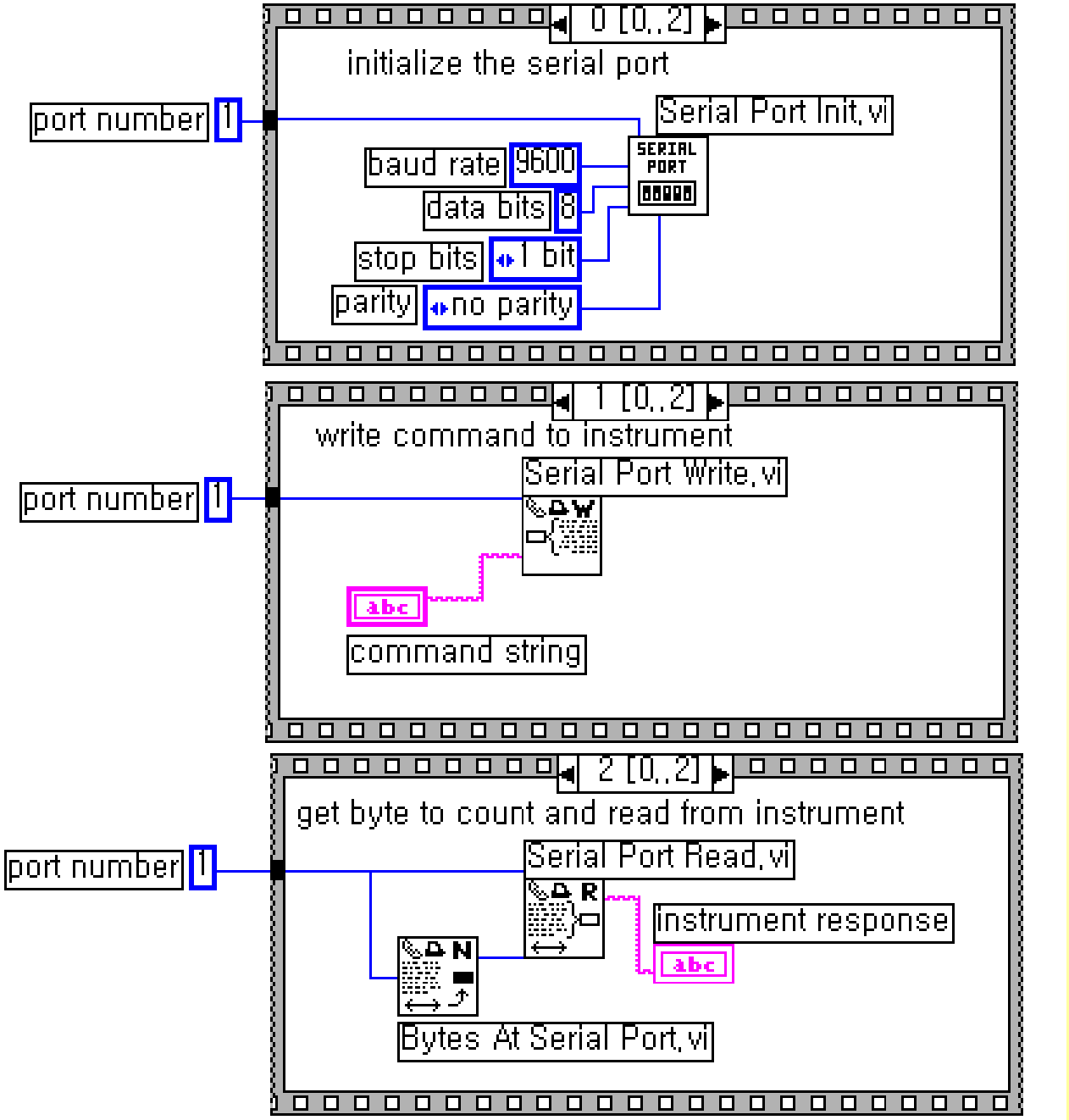


- port number가 표시하는 serial 포트의 입력 buffer의 byte수를 byte count에 출력한다.



- port number가 표시하는 serial 포트로부터 requested by count에 입력된 숫자만큼 문자열을 읽는다

- 예 : Serial 장비로부터 측정값을 읽는 경우.
  1. 먼저 Serial Port Init VI로 serial 포트를 초기화.
  2. Serial Port Write VI를 사용해 장비에 명령어를 보냄.
  3. Bytes at Serial Port VI를 사용해 serial 입력 버퍼에 사용 가능한 바이트 수를 결정.
  4. 최종적으로, Serial Port Read VI를 사용해서 장비로부터 데이터를 읽는다.


0 [0..2]
initialize the serial port
port number
1
Serial Port Init.vi
baud rate
9600
data bits
8
stop bits
1 bit
parity
no parity
SERIAL PORT
1 [0..2]
write command to instrument
port number
1
Serial Port Write.vi
abc
command string
2 [0..2]
get byte to count and read from instrument
port number
1
Serial Port Read.vi
R
instrument response
abc
N
Bytes At Serial Port.vi

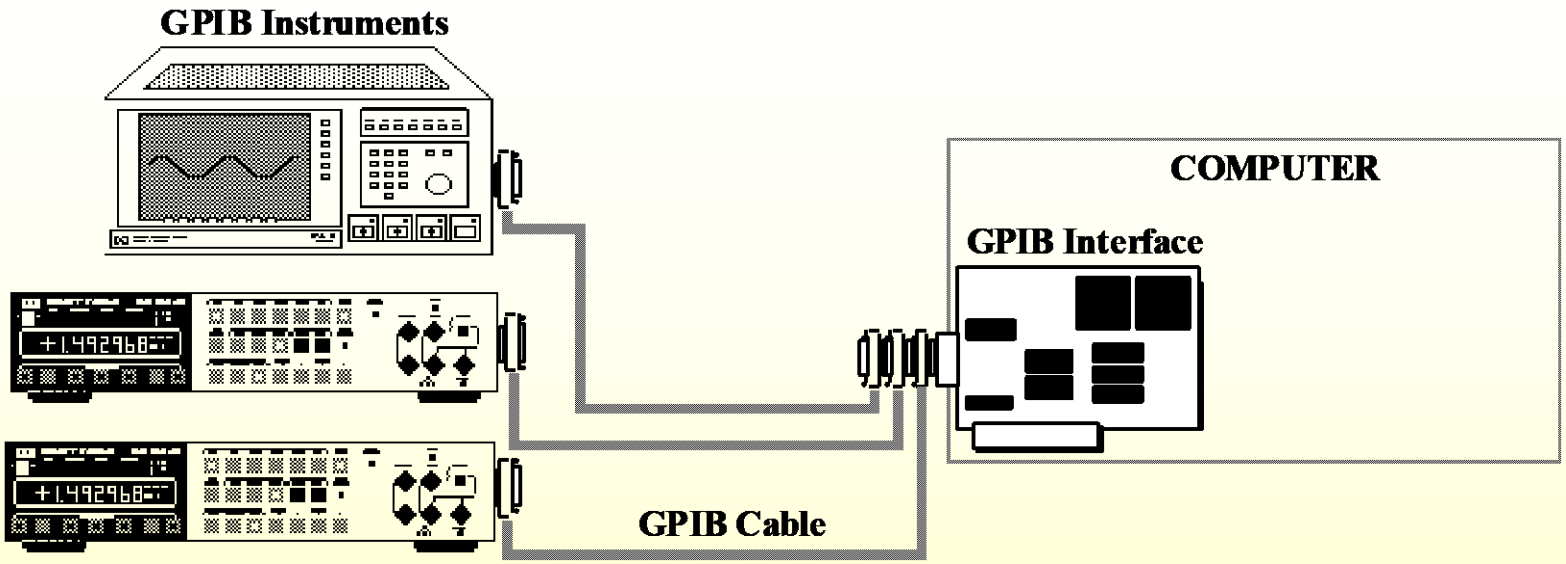
C. IEEE 488 (GPIB) 개요

- GPIB 라이브러리는 488.2 VI와 고전적 GPIB VI를 갖고있다.
  - GPIB 488.2 VI는 LabVIEW에 IEEE 488.2 호환성을 추가하였다.
  - 이들 VI는 National Instruments의 NI-488.2소프트웨어의 IEEE 488.2와 유사하다. 그러나, 여기서는 고전적 GPIB VI만 다룬다.
  - GPIB의 원래 목적은 시험 및 측정 장비를 동시에 컴퓨터로 제어. 현재 GPIB는 컴퓨터간 통신, 스캐너 제어, 필름 레코더 등 매우 다양하게 사용.
- 1970년대 초에 HP는 General Purpose Interface Bus(GPIB)를 개발, 1975년에 IEEE를 표준화, GPIB는 IEEE 488 표준으로 하였다.
- GPIB, HP-IB, IEEE 488라는 표현은 같은 뜻.
- GPIB는 24개의 디지털을 이용하는parallel bus. 8개의 데이터선, 5개의 버스 관리 선(ATN, EOI, IFC, REN, SRQ), 3개의 handshake선, 8개의 그라운드 선으로 구성.
- GPIB는 8비트 parallel, byte serial, asynchronous 데이터 전송 구조를 사용. 모든 바이트가 버스 선상에서 순차적으로 handshake함을 의미하며 버스 선상에 있는 가장 느린 장비가 전송속도를 좌우.
- GPIB에서 데이터의 단위는 byte(8 bit)이며, 메시지의 전송은 ASCII string으로 해석된다.

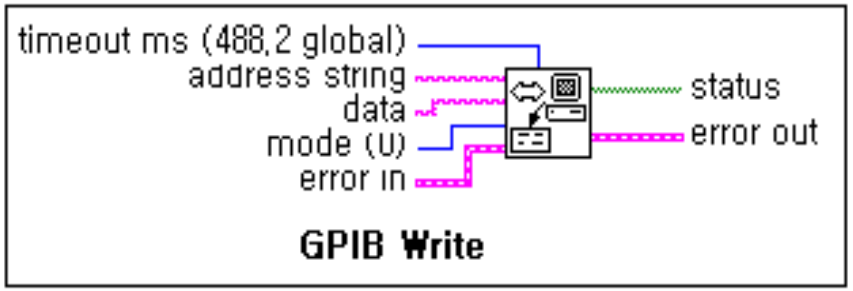
- 데이터 전송을 종료하는 방법은 3가지.

1. 가장 많이 사용하는 방법은, GPIB가 마지막 데이터를 전송하고 하드웨어 선(EOI)을 사용하는 것.
2. 특별한 문자 end-of-string(EOS)를 마지막 데이터에 추가하는 것. 일부 장비는 EOI 대신 EOS를 사용하거나, EOI와 EOS를 함께 사용.
3. Listener가 handshake되는 바이트를 지정된 수까지 카운트한 후 종료하는 것.

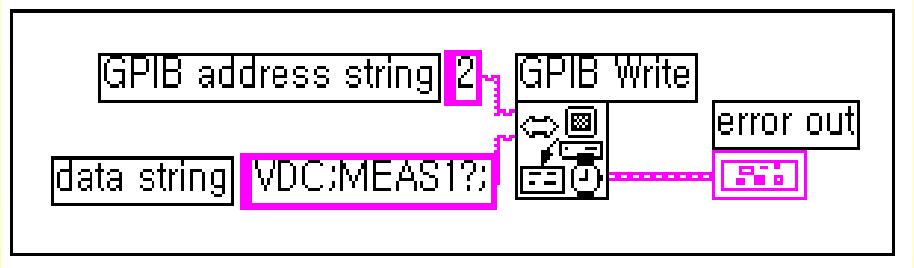
- 컴퓨터 인터페이스를 포함한 모든 장비는 0~30까지의 유일한 GPIB 어드레스를 가지고 있어야 한다.
  - **일반적으로 어드레스 0은 GPIB 인터페이스 보드가 사용**. GPIB선상에 있는 계측 장비는 어드레스 1~30을 사용.
- GPIB는 **버스를 제어하는 한 개의 *컨트롤러*(컴퓨터)를 갖고있다**.
  - 계측 장비 명령어와 데이터를 버스에 전송하기 위해서 **컨트롤러는 1개의 *Talker*와 1개 이상의 *Listener*를 지정한다**.
- **데이터 string은 버스를 통해 Talker로부터 Listener로 handshake된다**. LabVIEW의 GPIB VI는 자동적으로 어드레스를 부여하고 대부분의 버스 관리 함수를 처리한다.



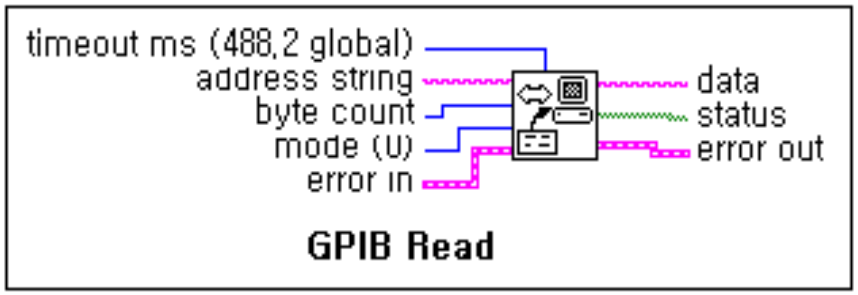
- GPIB VIs
  - LabVIEW는 GPIB통신을 하기 위한 여러 가지 VI를 Instrument 팔레트의 GPIB와 GPIB 488.2 보조 팔레트에 갖고있다. 이들 VI는 Low-level 488.2 드라이버로 부른다.
  - 대부분의 GPIB응용 프로그램은 계측 장비로부터 string을 출력하고 읽는 것으로 구성되어있다.
  - 다음은 고전적인 GPIB Write와 GPIB Read.



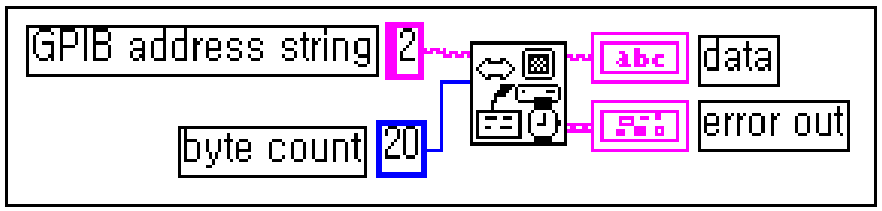
- Data string을 address string이 지시하는 GPIB장비에 출력.
  - Mode는 GPIB write를 종료하는 방법을 표시.
  - 지정된 timeout ms동안 완료되지 않으면 이 작업은 종료된다.
  - Error In과 Error Out cluster는 에러를 발견하기 위해 error handler VI와 함께 사용한다.
  - Status는 16-element Boolean array로 각 요소는 GPIB 컨트롤러의 상태를 표시.



- 예제 : GPIB Write VI는 string "VDC;MEAS1?"를 GPIB address가 2인 장비에 출력.
  - mode(0)와 timeout ms(25000)을 기본값으로 사용.



- 지정된 address string으로부터 byte count에 있는 바이트 수의 데이터를 읽는다. 데이터 읽는 것을 종료하기 위해 byte count와 함께 mode를 이용할 수 있다. 읽은 데이터는 data string으로 출력.



- 예 : GPIB Read VI가 어드레스 2인 계측 장비로부터 20바이트의 데이터를 읽음. mode(0)와 time-out ms(25000)은 기본값을 사용.

– GPIB Read VI는 다음과 같은 조건 중 한가지가 발생하면 종료.

(1) VI가 요청한 바이트 수를 모두 읽은 경우,

(2) VI가 에러를 발견한 경우,

(3) VI가 time limit를 초과한 경우,

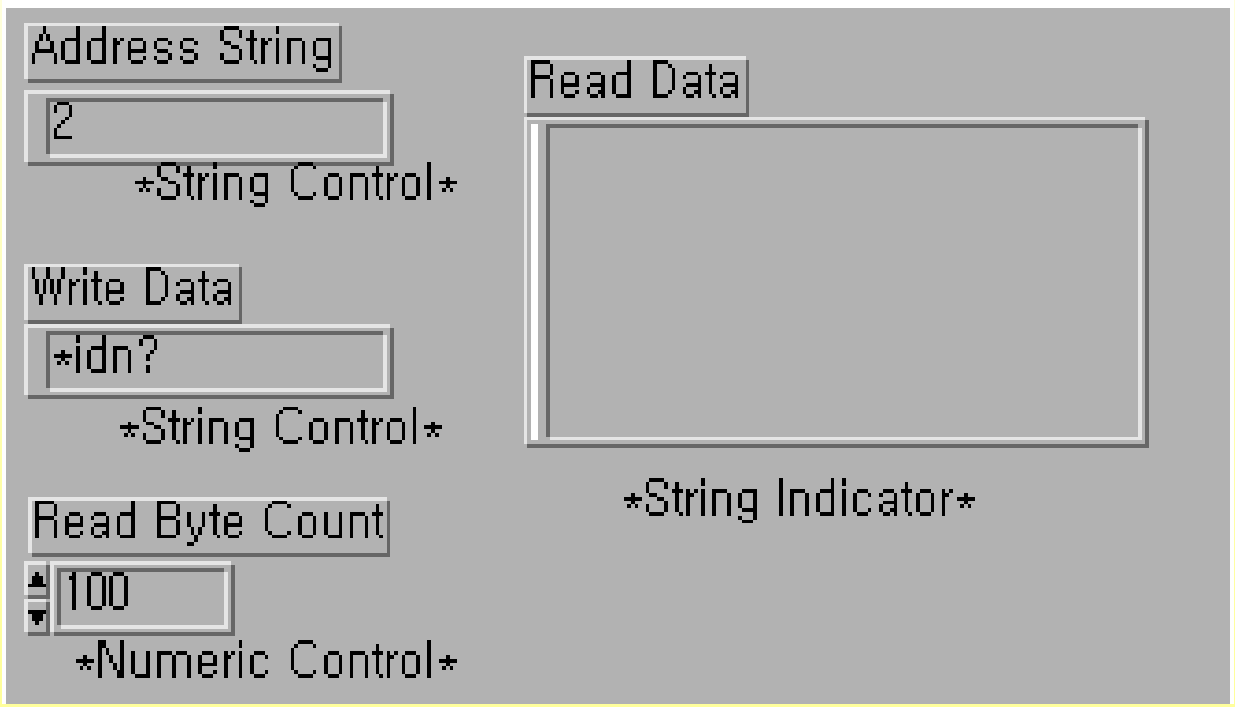
(4) VI가 END신호(EOI asserted)를 감지한 경우,

(5) VI가 end of string (EOS)를 감지한 경우

고전적인 GPIB VI를 사용해서 GPIB장비와 통신

- 예제 : GPIB 계측 장비와 통신하는 VI를 만들기.
- 고적인 GPIB VI 인 GPIB write와 GPIB read를 이용.
- 명령어 “*idn?”은 IEEE 488.2 호환 계측 장비에서 동작한다. 이 명령어로 계측기는 자신의 identification string을 출력한다.

프런트 패널



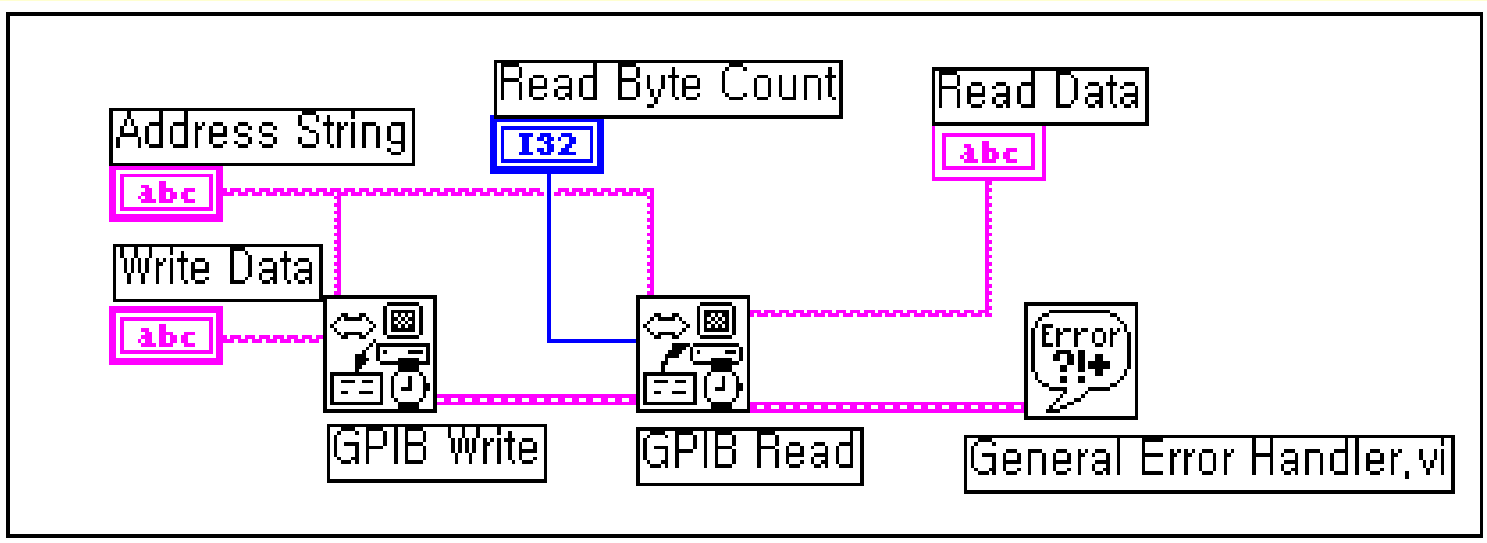
1.File메뉴에서 **New**를 선택하여 새로운 Panel을 연다.

2.앞의 예와 같이 **Panel의 Control과 indicator**를 만든다.

- 모든 Control과 indicator는 블럭 다이어그램에서 pop-up을 하면 Create Control 또는 Create Indicator 등 원하는 것을 만들 수 있다.

3.**Write Data string에 "*idn?"을 입력.** Read Byte Count numeric 컨트롤에는 100을 입력.

블럭 다이어그램



1.Diagram Window를 연다.

2.앞서와 같이 **블럭 다이어그램**을 완성한다.

GPIB Write함수(Instrument I/O :: GPIB 보조 팔레트): 데이터 string을 GPIB 계측 장비에 쓴다.

GPIB Read함수(Instrument I/O :: GPIB 보조 팔레트): GPIB 계측 장비로부터 데이터를 읽는다.

General Error Handler VI(Time & Dialog 보조 팔레트): 이 VI는 에러 cluster를 체크하고, 만약 에러가 발생했으면 dialog box를 표시한다.

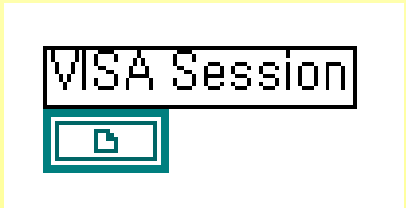
3.프런트 패널로 가서 VI를 실행.

- Identification string 값이 Read Data에 표시 되어야 한다.
- 만약 표시되는 값이 없다면 GPIB 에러, EABO 또는 error=6이 표시된다. 이는 계측 장비가 “*idn?”명령어를 이해하지 못한다는 의미이다.

*** 적절한 명령어를 주기 위해 계측 장비의 매뉴얼을 참조하라.**

D. VISA

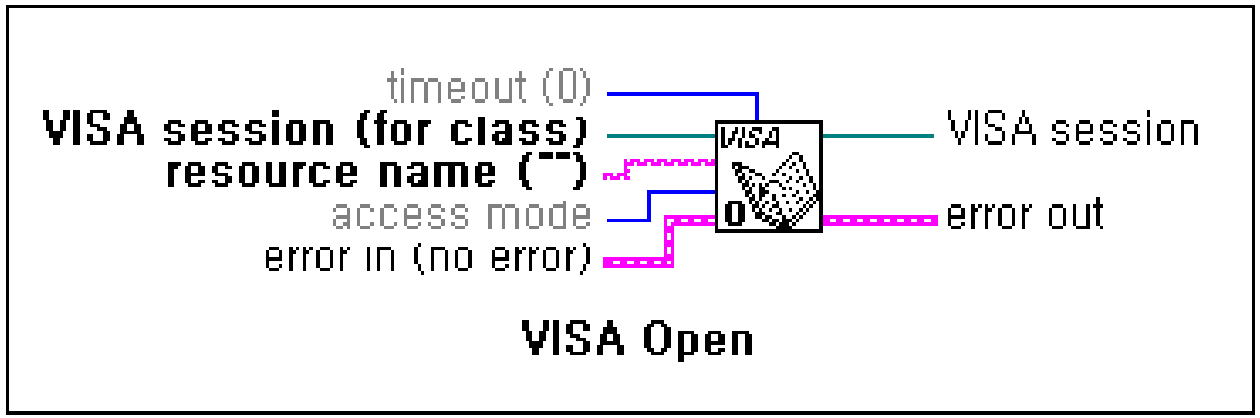
- VISA는 Virtual Instrumentation Software Architecture.
- VISA는 VXI, GPIB, RS-232, 기타 형태의 계측기 등을 LabVIEW로 제어하는 유일한 인터페이스 함수이다.
- VISA는 VXI*plug & play* System Alliance를 주축으로, 35개의 계측 장비 생산업체가 협력해서 만든 산업 표준이다.
  - VISA 표준은 시간과 instrument I/O 선택에 관계없이 소프트웨어를 상호간에 사용할 수 있고 재사용할 수 있도록 산업을 통일화 시킨다.
- 이들 함수들은 Function 팔레트의 Instrument I/O ▸ VISA 보조 팔레트에 있다.
- 대부분의 VISA 라이브러리 함수는 VISA session 변수를 사용. VISA session은 컨트롤 패널의 Path and Refnum 보조 팔레트에 있다.



- VISA session은 유일한 session의 논리 identifier이다.
  - 이것은 VI가 계측 장비와 통신하고 I/O를 하기 위해 필요한 모든 configuration 정보를 확인한다.

- VISA session은 VISA Open 함수로부터 만들어지고 VISA primitive 함수에 의해 사용된다.
- VISA Open 함수는 다음 VISA 함수에 session을 전달한다.
  - 이는 프로그램을 간략하게 만들고 file I/O 함수에서 제공하는 file refnum 과 유사하다.
- VISA session은 기본값으로 *Instr* class.
- VISA session을 pop-up하면 class를 *GPIB Instr, Serial Instr, VISA/GPIB-VXI RBD Instr, Generic Event, Trigger Event, VXI Signal Event, Resource Manager* 등으로 변경할 수 있다.

- 일반적인 VISA함수 VISA Open, VISA Write, VISA Read, VISA Close 는 다음과 같다.

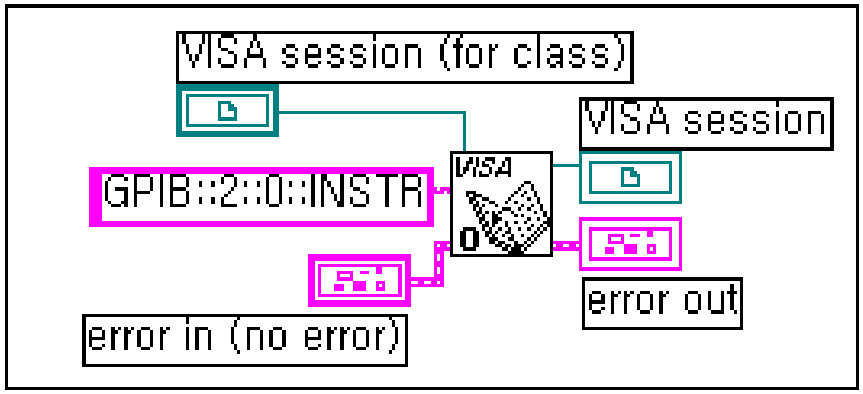


- VISA Open 은 Resource Name과 VISA session(for class)을 기본으로 지정된 device와 통신을 확립.
- 이 함수는 session identifier인 VISA session을 출력하며 이는 device가 다른 작업을 할 때 필요하다.
- Resource Name은 I/O 인터페이스의 형태와 장비 어드레스 정보를 포함하고 있다

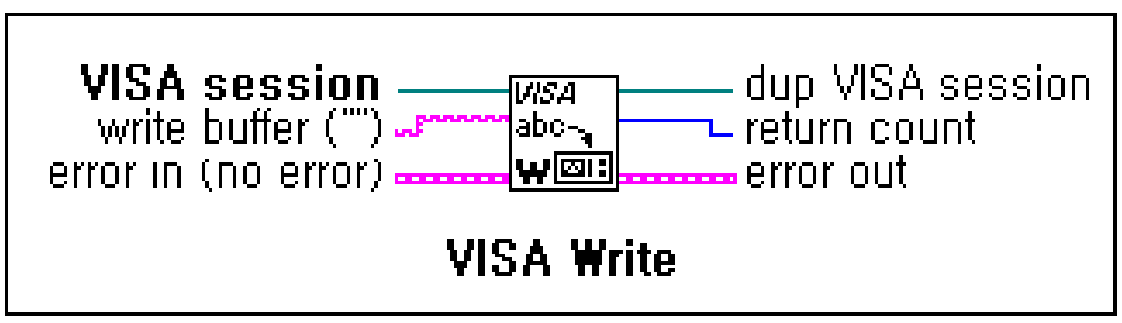
계측 장비를 표현하는 문법

Interface	문 법
SERIAL	ASRL[board][::INSTR]
GPIB	GPIB[*board*]::*primary address*[:: *secondary address*][::INSTR]
VXI	VXI[board]::*VXI logical address*[::INSTR]
GPIB-VXI	GPIB-VXI[board][::*GPIB-VXI primary address*]::*VXI logical address*[::INSTR]

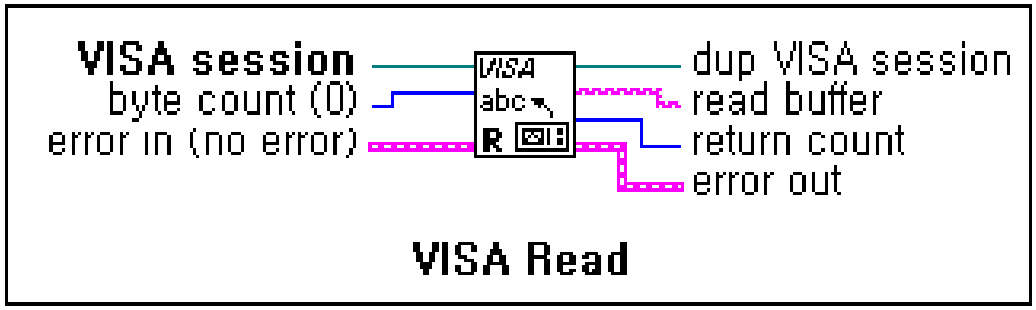
- GPIB 장비와 통신하려면 *GPIB* keyword를 사용.
- Embedded 또는 MXIbus 컨트롤러와 같은 VXI 장비는 *VXI* keyword를 사용.
- GPIB-VXI 컨트롤러는 *GPIB-VXI* keyword를 사용.
- Asynchronous serial과 통신하려면 *ASEL* keyword를 사용.



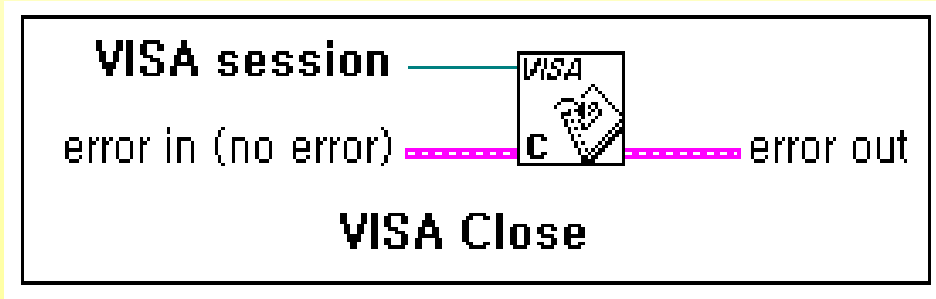
- VISA Open은 primary 어드레스 2인 계측 장비와 GPIB 통신을 하기 위해 "GPIB::2::0::INSTR"의 instrument descriptor string을 사용



- VISA Write는 write buffer string 값을 VISA session이 지시하는 장비에 쓴다. dup VISA session에는 동일한 handle 값이 출력.
  - UNIX에서 데이터는 synchronously하게 쓰여지고, 다른 종류의 컴퓨터는 데이터는 asynchronously하게 쓰여진다.
- return count는 실제 전송된 바이트 수.



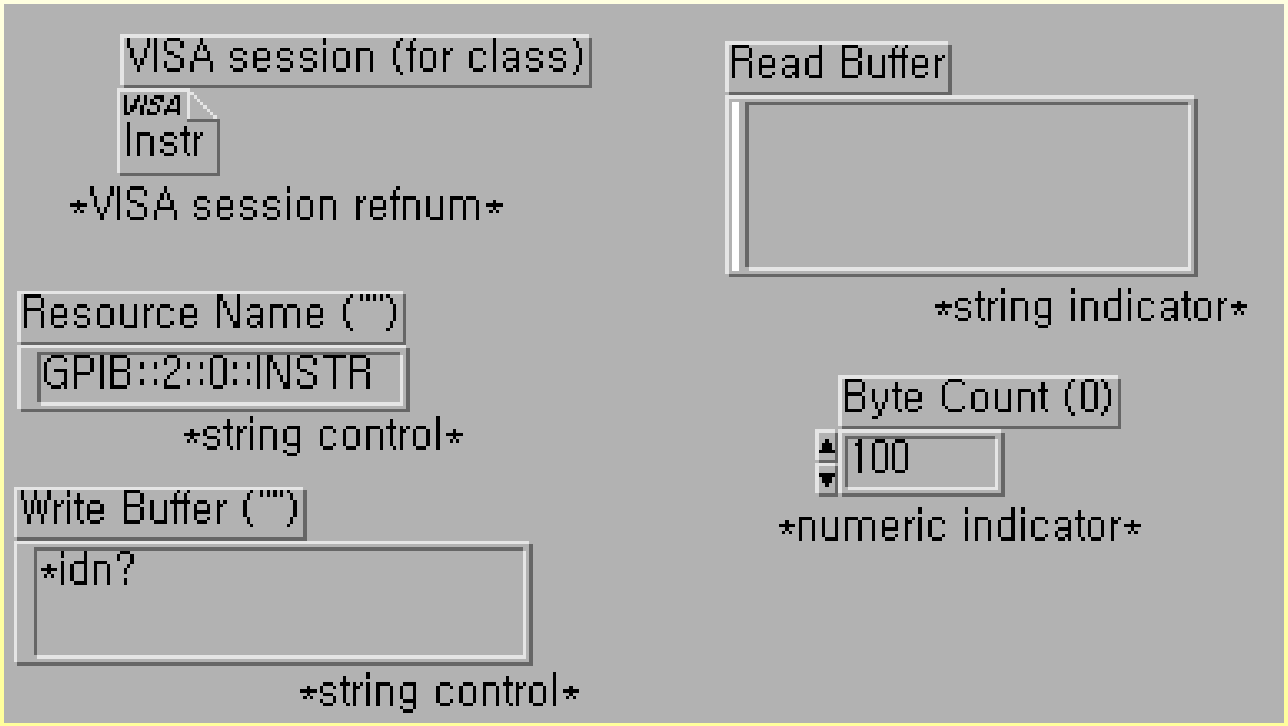
- VISA Read는 VISA session이 지시하는 장비로부터 data를 읽는다.
  - byte count 는 읽을 data의 바이트 수로 이 값은 read buffer로 출력.
  - dup VISA session에는 동일한 handle 값이 출력.
  - UNIX에서 데이터는 synchronously하게 쓰여지고, 다른 종류의 컴퓨터는 데이터는 asynchronously하게 쓰여진다.
  - return count는 실제 전송된 바이트 수



- VISA Close는 VISA session이 지시하는 장비를 close.

VISA VI를 이용해서 GPIB/Serial 장비와 통신

- 이 예는 VISA write와 VISA read 같은 VISA VI를 사용하여 계측 장비 (GPIB 또는 Serial)와 통신.
- 명령어 "*idn?"는 대부분의 GPIB 또는 serial 장비에 사용할 수 있고, 이 명령어로 장비는 identification string을 출력.
  - 아래의 VISA 예제는 Fluke 45를 사용해서 만들었다



1.새로운 Panel을 열고 위와 같이 만든다.

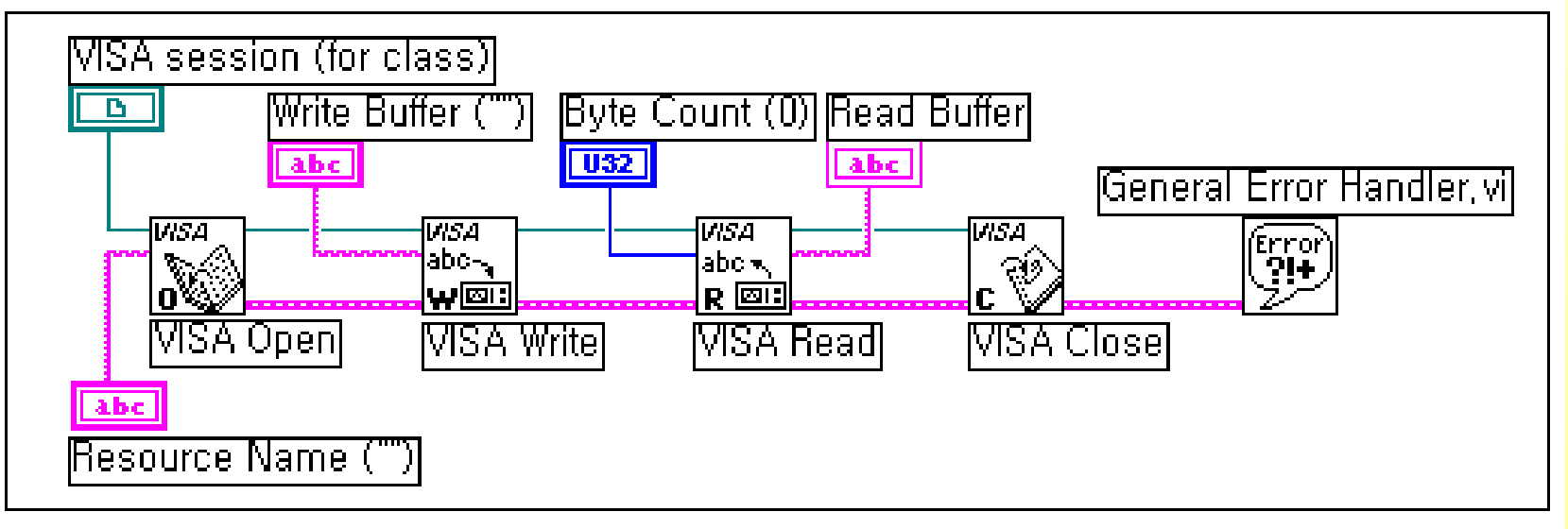
- VISA session 컨트롤은 컨트롤 팔레트의 Path & Refnum 보조 팔레트에 있다. 또한 다이어그램 윈도우에서 VISA Open 함수를 pop-up하면 위와 같이 만들 수 있다.

2.Resource Name 컨트롤은 다음과 같은 string 값으로 한다.

- 어드레스가 2인 GPIB 장비 : “GPIB::2::0::INSTR”
- COM1에 연결된 serial장비: “ASRL1::INSTR”

* Note : VISA Open 함수의 Online Help를 이용하면 상세한 Resource Name 컨트롤을 볼 수 있다.

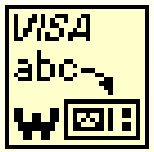
3.Write string Contrl에 “*idn?”를 입력. Byte Control에는 100을 입력.



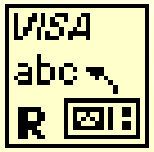
1.다이어그램 윈도우를 연다.

2.블럭 다이어그램에서 위와 같이 만든다.

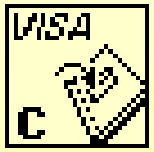
VISA Open 함수(Instrument I/O::VISA 보조팔레트) : communica-tion session을 열고 VISA session refnum을 발생.



VISA Write 함수(Instrument I/O::VISA 보조 팔레트) : 데이터 string을 지정된 장비에 쓴다.



VISA Open 함수(Instrument I/O::VISA 보조 팔레트) : 지정된 장비로부터 데이터를 읽는다.



VISA Open 함수(Instrument I/O::VISA 보조 팔레트) : 지정된 VISA session을 종료.

3.프런트 패널로 가서 VI를 실행.

- VISA session 컨트롤은 INSTR에 설정되어 있어야 한다. 이것을 변경하려면 VISA session 컨트롤을 pop-up한 후 VISA class를 선택.
- Resource name 컨트롤 값에 따라 serial 또는 GPIB 장비와 통신.

4.VI를 VISA. vi로 저장하고 종료.

E. LabVIEW로 Instrument 드라이버의 작성

- Instrument 드라이버란 특정한 계측 장비를 제어하는 S/W의 일부분.
- LabVIEW는 instrument 드라이버를 만드는데 이상적이다.
  - 프런트 패널은 계측 장비의 프런트 패널을 가상으로 운용할 수 있다.
  - 프런트 패널에는 단지 입력 값만 지정하면 된다.
  - 전체 시스템을 운용하기 위해서는 Instrument 드라이버를 다른 SubVI와 함께 Sub VI로 사용할 수 있다.

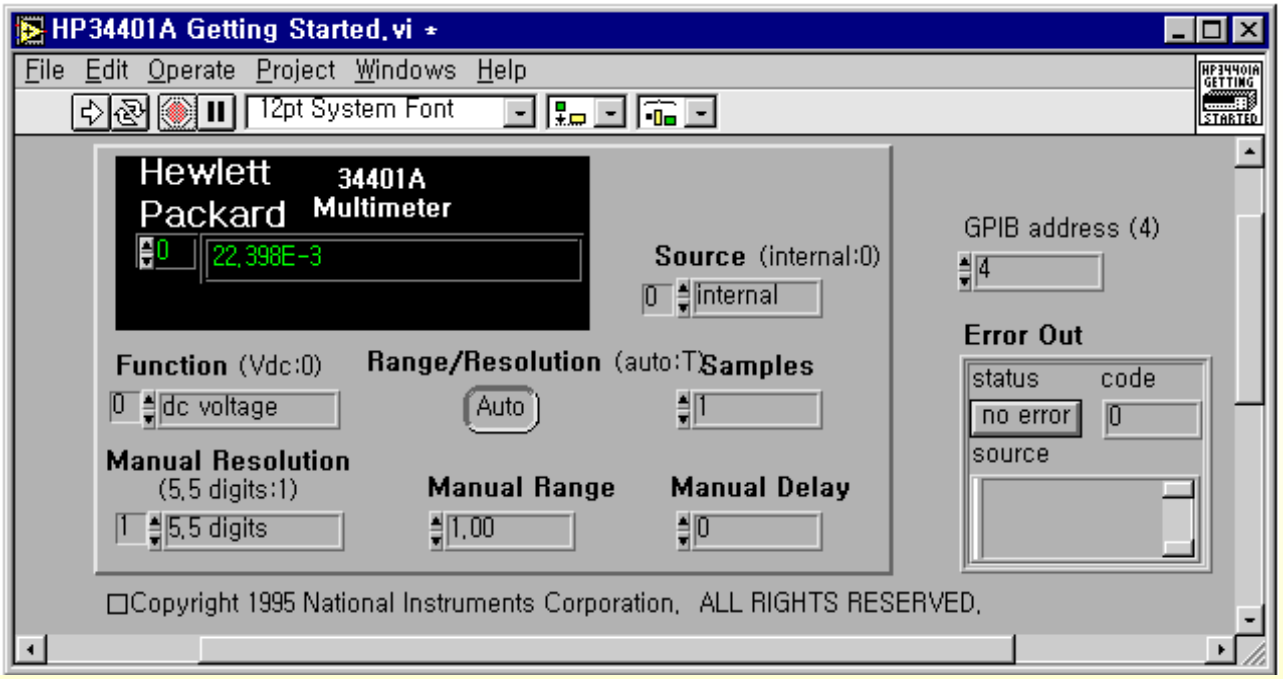
• Instrument 드라이버 Templates

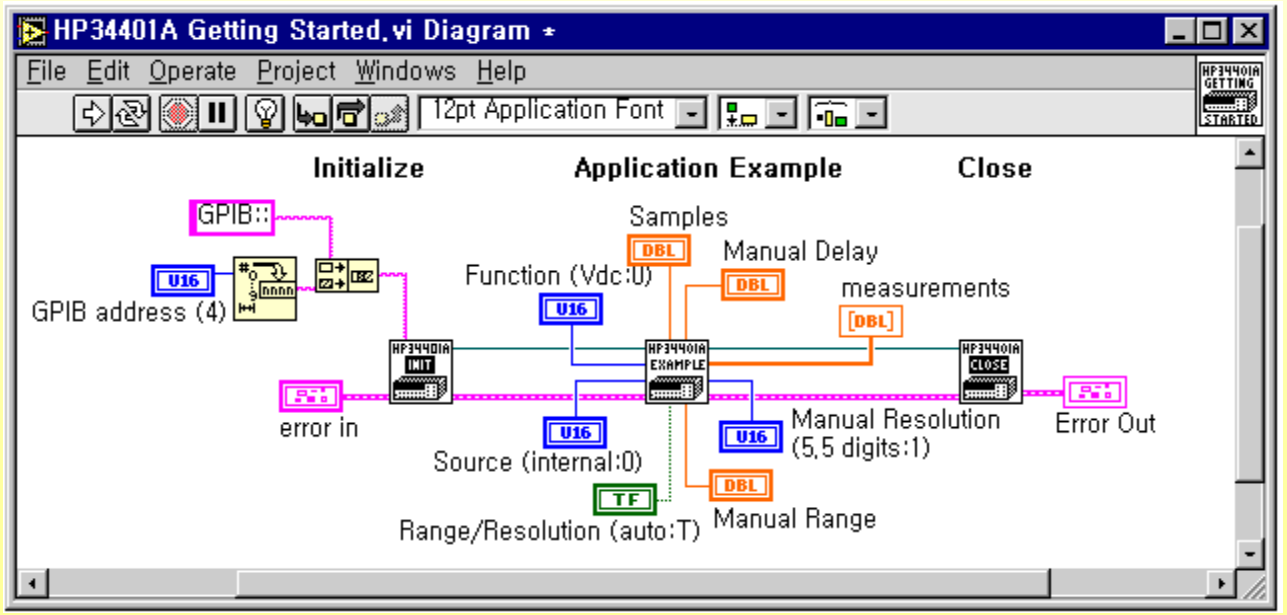
- LabVIEW example directory는 많은 VISA instrument 드라이버 template VI를 LabVIEW ▸ EXAMPLES ▸ INSTR ▸ INSTTMPL ▸ *.LLB 라이브러리에 갖고있다.
- 비교적 많이 사용되는 계측 장비들의 instrument 드라이버 VI이고, instrument 드라이버를 LabVIEW로 개발할 때 기본적을 사용된다.
- Instrument 드라이버를 개발하고자 할 때 LabVIEW instrument 드라이버 template VI를 보조로 사용해야 한다. Template VI에는 이미 instrument 드라이버 표준이 만들어져 있고, 각 VI는 명령어 세트를 수정하기 위한 지침서가 들어있다.

F. LabVIEW에 기본적으로 설치된 Instrument 드라이버

- HP34401A Getting Started instrument 드라이버.
  - 만약 HP34401A Multimeter를 갖고 있으면 이 VI를 실행할 수 있다.

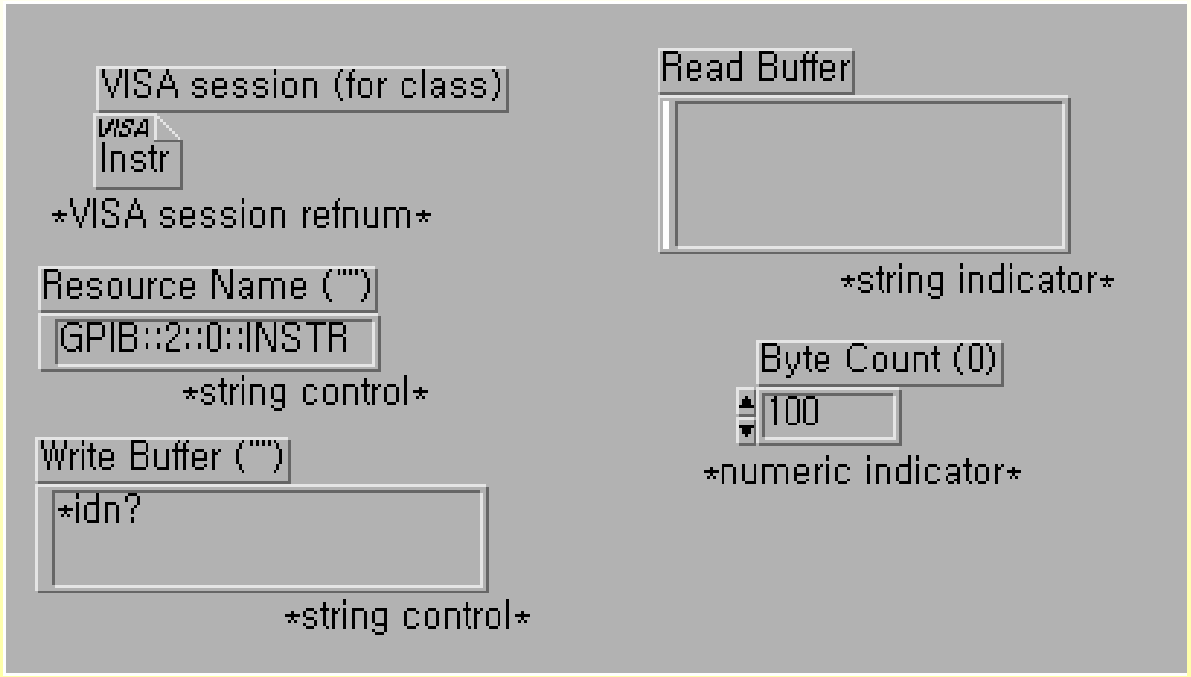
- LabVIEW ▸ EXAMPLES ▸ INSTR ▸ HP34401A.LLB에서 HP34401A Getting Started VI를 연다.
  - HP34401A Getting Started VI는 instrument 드라이버 응용 VI.
  - VI의 프런트 패널은 계측기의 프런트 패널과 유사하다.
- Help메뉴에서 Show Help를 선택해서 Help 윈도우를 표시.
  - HP34401A Getting Started VI의 프런트 패널에서 커서를 컨트롤과 indicator에 위치시키면 각 물체의 의미를 볼 수 있다.
  - instrument 드라이버를 작성하거나 사용할 때 documentation은 매우 중요하다.
- High-level VI는 계측 장비를 configure하거나 읽을 때 사용할 수 있다. 이들 VI는 HP34401A의 하위 레벨 instrument 드라이버 요소 VI들로 구성되어 있다. 하위 레벨 VI는 이전 장에서 배운 VISA를 사용했다.


HP34401A Getting Started.vi *
File Edit Operate Project Windows Help
12pt System Font
Hewlett Packard
34401A Multimeter
0
22,398E-3
Source (internal:0)
0 internal
GPIB address (4)
4
Error Out
status
code
no error
0
source
Function (Vdc:0)
0 dc voltage
Range/Resolution (auto:T)
Auto
Samples
1
Manual Resolution (5,5 digits:1)
1 5,5 digits
Manual Range
1,00
Manual Delay
0
Copyright 1995 National Instruments Corporation, ALL RIGHTS RESERVED.


HP34401A Getting Started.vi Diagram *
File Edit Operate Project Windows Help
12pt Application Font
Initialize
Application Example
Close
GPIB::
U16
GPIB address (4)
HP34401A INIT
error in
Samples
DBL
Manual Delay
Function (Vdc:0)
U16
measurements
[DBL]
HP34401A EXAMPLE
HP34401A CLOSE
Error Out
U16
Source (internal:0)
U16
Manual Resolution (5,5 digits:1)
TF
Range/Resolution (auto:T)
DBL
Manual Range

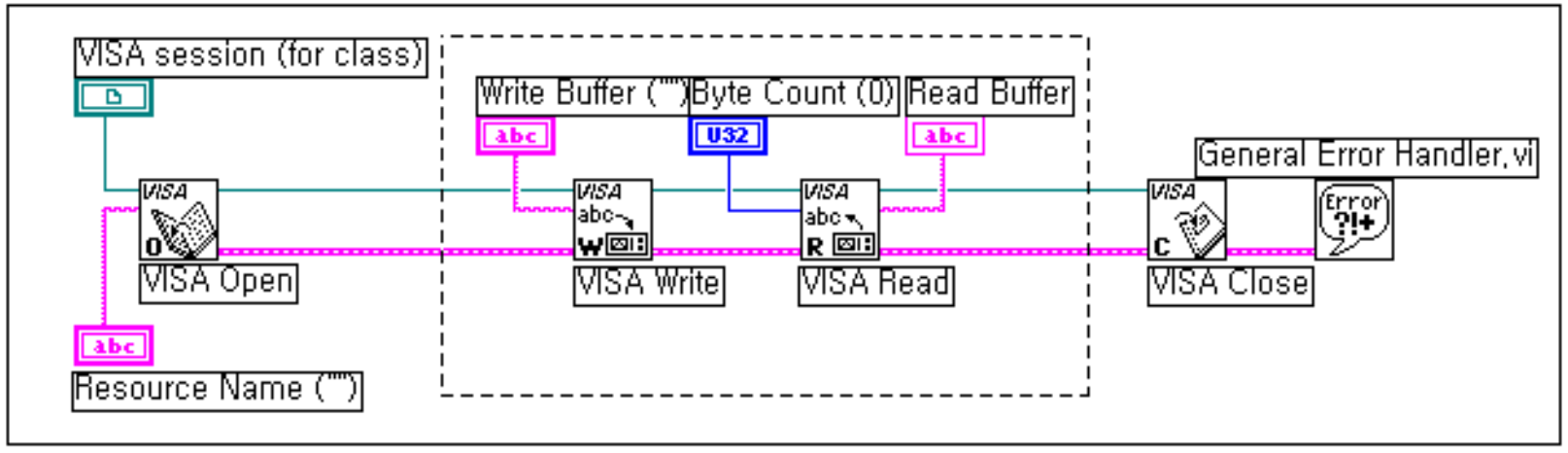
간단한 Instrument 드라이버를 작성

- 예제 3-2에서 만든 VISA.VI로 간단한 Instrument 드라이버를 작성.
- 이들 요소는 VISA.VI로 작성할 것이다.
- **프런트 패널**

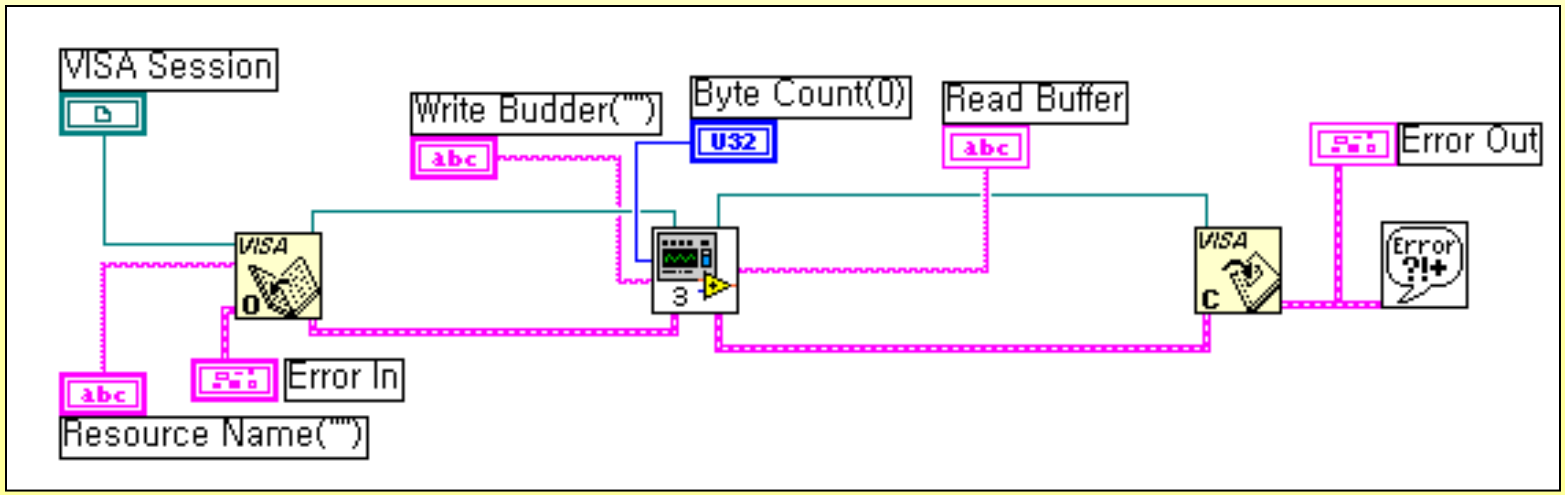


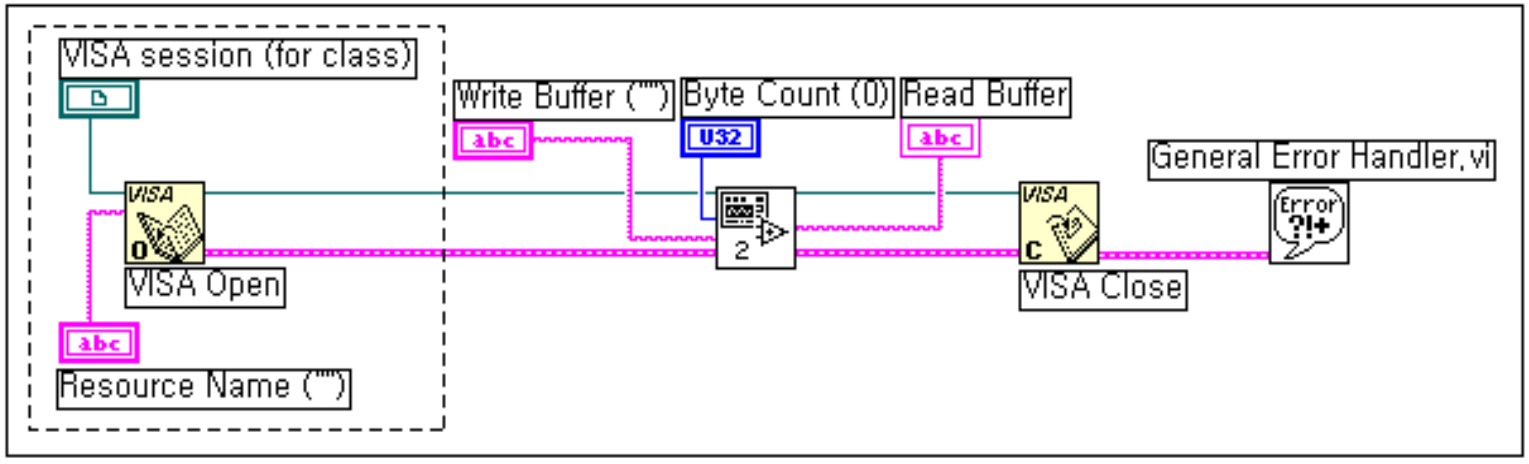
1. 이전 예제의 VISA.VI를 연다.
2. 프런트 패널은 수정하지 않는다.

-블럭 다이어그램

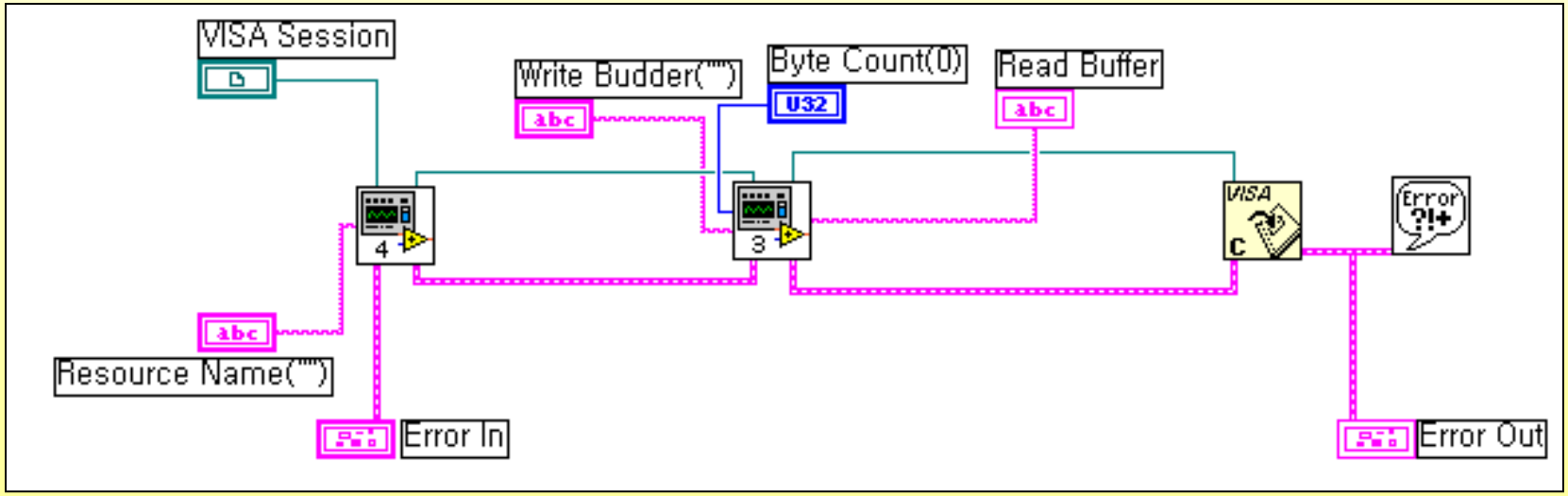


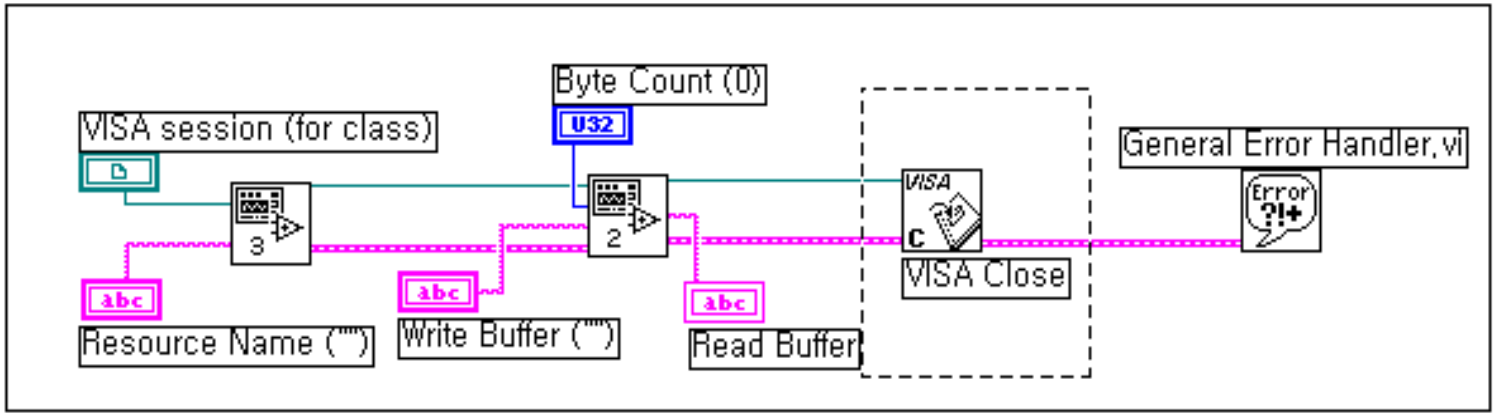
1. Diagram Window로 간다.
2. 위와 같이 Diagram Window에서 특정 영역을 선택.
   - Edit 메뉴에서 Create SubVI를 선택.→ SubVI를 생성.



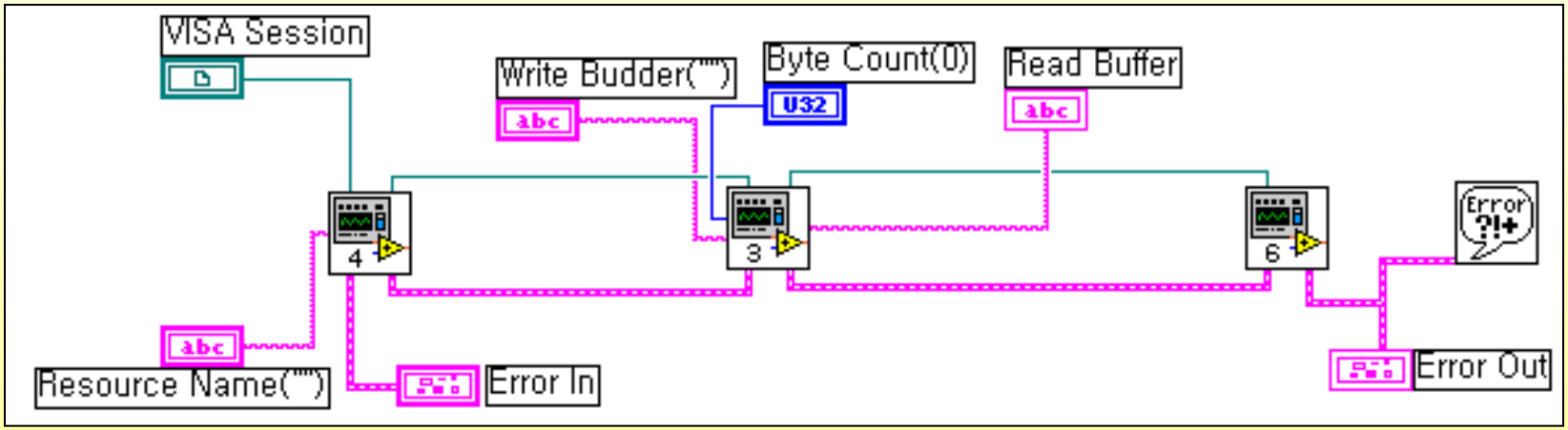


3. 다음으로, 위의 예와 같이 특정 영역을 지정하고 Edit 메뉴에서 Create SubVI를 선택.

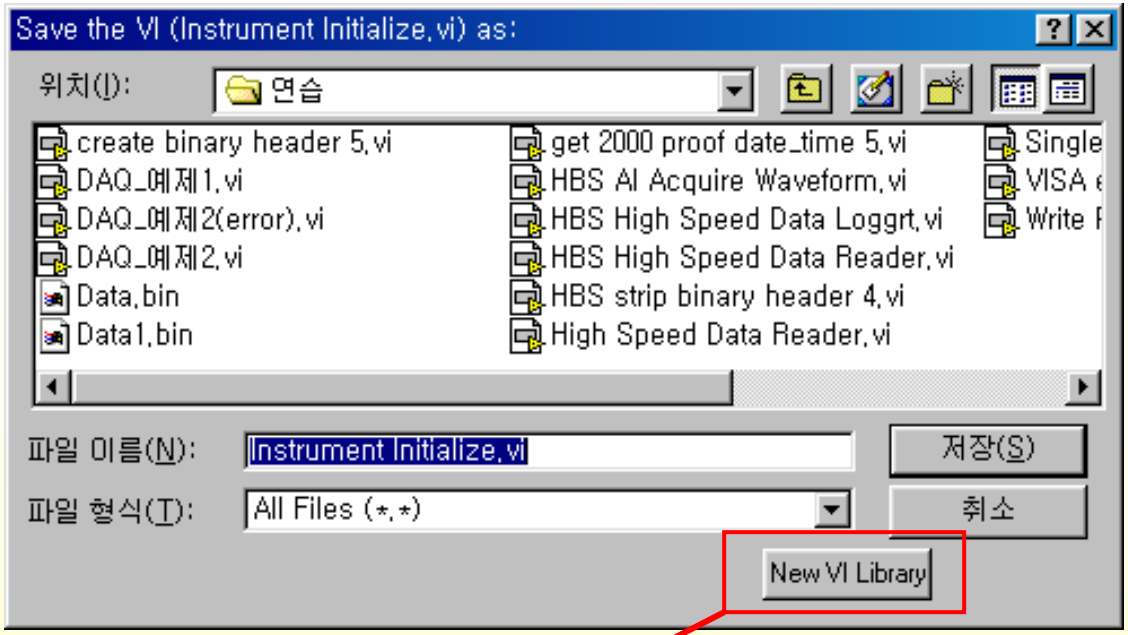


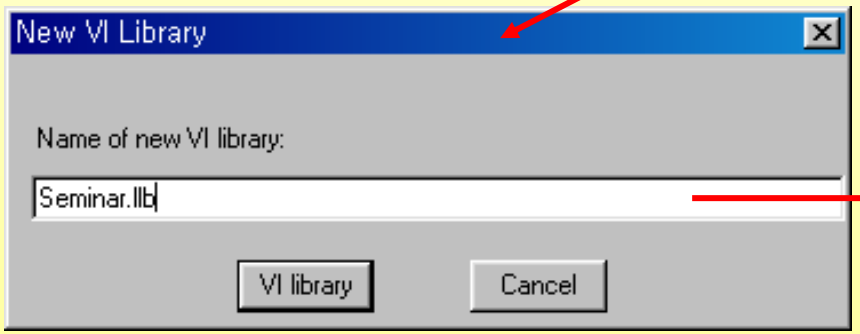


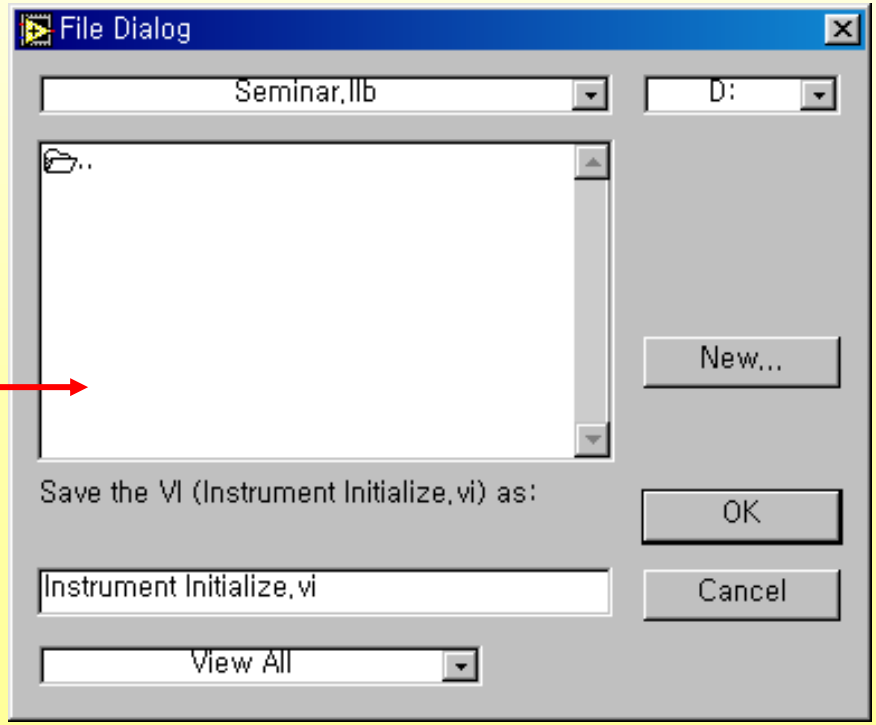
4. 마지막으로, 선택된 영역을 지정하고 Edit메뉴에서 Create SubVI를 선택.

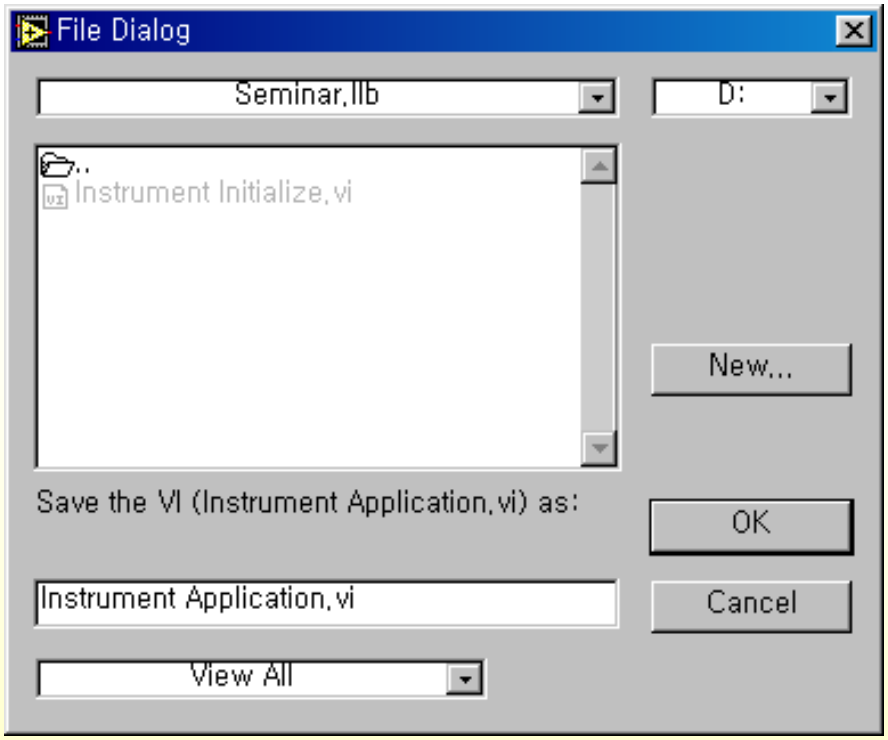


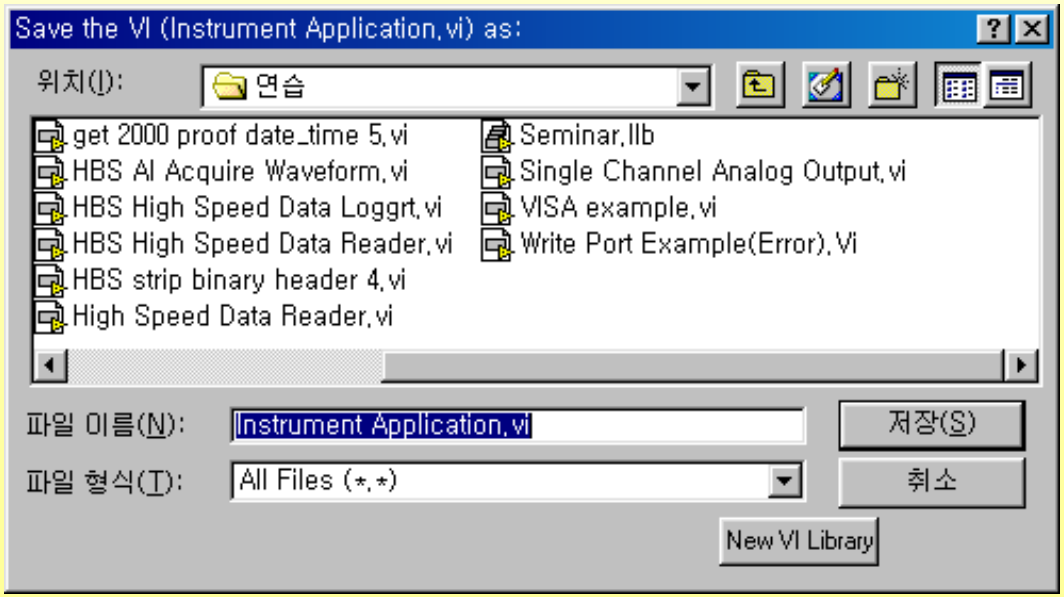
5.이것으로 instrument SubVI는 완성되었고, 각각의 SubVI 아이콘을 두번 선택해서 **Instrument Initialize .vi, Instrument Application. Vi, Instrument Close. vi**로 각각 **Seminar.llb**에 저장한다


Save the VI (Instrument Initialize.vi) as:
위치(I):
연습
create binary header 5.vi
DAQ_예제1.vi
DAQ_예제2(error).vi
DAQ_예제2.vi
Data.bin
Data1.bin
get 2000 proof date_time 5.vi
HBS AI Acquire Waveform.vi
HBS High Speed Data Loggrt.vi
HBS High Speed Data Reader.vi
HBS strip binary header 4.vi
High Speed Data Reader.vi
Single
VISA
Write
파일 이름(N):
Instrument Initialize.vi
저장(S)
파일 형식(T):
All Files (*.*)
취소
New VI Library


New VI Library
Name of new VI library:
Seminar.llb
VI library
Cancel

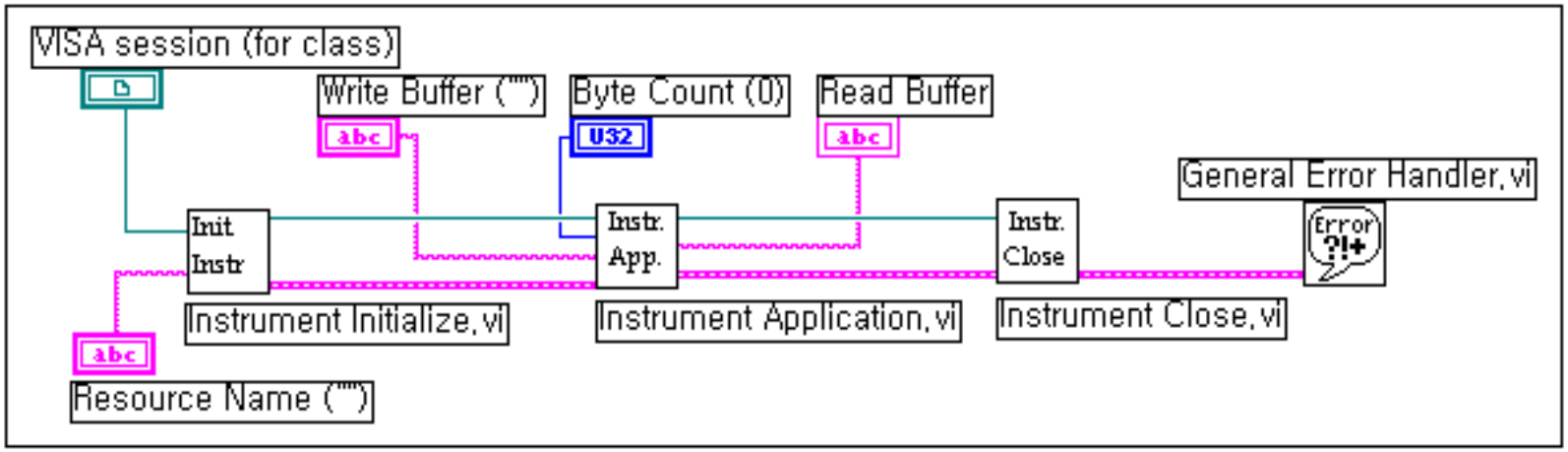

File Dialog
Seminar.llb
D:
..
New...
Save the VI (Instrument Initialize.vi) as:
OK
Instrument Initialize.vi
Cancel
View All





- **Note**: 이들 VI의 아이콘들을 바꾸고자 한다면, 프런트 패널 우측 상단의 **Icon Editor**로 간다. 우측 상단의 아이콘을 pop-up해서 **Icon Editor**를 선택해야 한다

6. Instrument 드라이버 SubVI를 만든 후 Seminar.llb에 저장하고, SubVI의 기본 아이콘을 수정해서 아래의 그림과 같게 한다.



7.프런트 패널로 되돌아간다. Resource Name string 컨트롤에 “GPIB::2::0::INSTR”을 입력한다. VI를 실행한다.

- 만약 2가 아닌 GPIB계측기를 갖고 있다면, 2를 적절한 GPIB어드레스로 바꾼다.
- 입력하고자 하는 명령어를 간단히 Write Buffer string 컨트롤에 입력.

8.VI를 Instrument driver.vi로 저장하고 종료한다.