

LabVIEW™ Connectivity 연습문제

교육과정 소프트웨어 버전 2011

2011 년 12 월판

325628A-0129

저작권

© 2004–2011 National Instruments Corporation. 판권 소유.

저작권법에 따라, 이 간행물은 National Instruments Corporation 의 사전 서면동의없이 간행물의 전부 또는 일부라도 사
진복사, 기록, 정보검색시스템으로 저장, 번역을 할 수 없음을 물론이거니와 전자 또는 기계방식의 여하한 형태로도 복제 또는
송신될 수 없습니다.

내쇼날인스트루먼트는 타인의 지적재산권을 존중하며 사용자에게도 그렇게 할 것을 요청하고 있습니다. NI 소프트웨어는 저
작권 및 기타 지적재산권법에 의해 보호받고 있습니다. NI 소프트웨어를 타인 소유의 소프트웨어 또는 기타 자료들을 복제하
는데 사용할 수 있는 경우, 여러분은 NI 소프트웨어를 라이센스 또는 기타 법적 제한조건에 따라 복제해도 되는 자료들을 복
제하는데에만 사용하여야 합니다.

For components used in USI (Xerces C++, ICU, HDF5, b64, Stingray, and STLport), the following copyright stipulations
apply. For a listing of the conditions and disclaimers, refer to either the `USICopyrights.chm` or the *Copyrights* topic
in your software.

Xerces C++. This product includes software that was developed by the Apache Software Foundation
(<http://www.apache.org/>).

Copyright 1999 The Apache Software Foundation. All rights reserved.

ICU. Copyright 1995–2009 International Business Machines Corporation and others. All rights reserved.

HDF5. NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright 1998, 1999, 2000, 2001, 2003 by the Board of Trustees of the University of Illinois. All rights reserved.

b64. Copyright © 2004–2006, Matthew Wilson and Synesis Software. All Rights Reserved.

Stingray. This software includes Stingray software developed by the Rogue Wave Software division of Quovadx, Inc.
Copyright 1995–2006, Quovadx, Inc. All Rights Reserved.

STLport. Copyright 1999–2003 Boris Fomitchev

상표

LabVIEW, National Instruments, NI, ni.com, National Instruments 회사 로고 및 이글 로고는 National Instruments
Corporation의 상표들입니다. National Instruments의 기타 상표는 ni.com/trademarks 의 *Trademark Information*을
참조하시기 바랍니다.

이 문서에서 언급된 다른 제품과 회사의 이름들은 각각 해당 회사들의 상표이거나 상호들입니다.

National Instruments Alliance Partner Program 의 멤버들은 National Instruments 와는 다른 독자적인 사업 기구들이며
National Instruments 와 어떠한 대리관계나 파트너십 또는 joint-venture 관계를 가지고 있지 않습니다.

특허권

National Instruments 제품 / 기술에 대한 특허권에 관하여는 귀하의 소프트웨어에 있는 **도움말** > **특허**, 귀하의 미디어에 있는
`patents.txt` 파일 또는 ni.com/patents 의 *National Instruments Patent Notice*를 참고하십시오.

전세계 기술 지원 및 제품 정보

ni.com/korea

전 세계 현지 사무소

ni.com/niglobal 을 방문하여 최신 연락 정보, 지원 전화번호, E-메일주소 및 이벤트 정보를 제공하는 각 사무소의 웹 사이트에 접속할 수 있습니다.

National Instruments 본사

11500 North Mopac Expressway Austin, Texas 78759-3504 USA 전화 : 512 683 0100

National Instruments 문서에 대한 문의사항은 National Instruments 웹 사이트의 ni.com/info 에서 정보 코드 `feedback` 을 입력하십시오.

목차

수강생 안내서

1 과

LabVIEW 에서 공유 라이브러리 호출하기 연습문제

연습문제 1-1	컴퓨터 이름	1-1
----------	--------------	-----

2 과

VI 서버 연습문제

연습문제 2-1	VI 서버 옵션	2-1
연습문제 2-2	VI 통계	2-2
연습문제 2-3	VI 원격 실행	2-9
연습문제 2-4	다이나믹하게 VI 호출	2-12

3 과

LabVIEW 에서 .NET 및 ActiveX 객체 사용하기 연습문제

연습문제 3-1	폰트 대화상자	3-1
연습문제 3-2	워드 프로세서	3-6
연습문제 3-3	자동 저장	3-20
연습문제 3-4	URL 탐색 및 VI 통계 리포트 디스플레이	3-27

4 과

데이터베이스에 연결하기 연습문제

연습문제 4-1	데이터베이스 보기	4-1
연습문제 4-2	LabVIEW 를 사용하여 Theatre Database 에 연결하기	4-3
연습문제 4-3	테이블에서 데이터 선택하기	4-5
연습문제 4-4	새로운 데이터 삽입하기	4-10
연습문제 4-5	SQL 쿼리	4-15

5 과

TCP/IP 및 UDP 연습문제

연습문제 5-1	Simple Data Client VI 및 Simple Data Server VI	5-1
연습문제 5-2	TCP 신호 데이터 전달	5-5

6 과

웹 서비스 연습문제

연습문제 6-1	LabVIEW 웹 서비스를 생성하여 두 숫자의 합 구하기	6-1
연습문제 6-2	HTML Form 으로부터 POST Data 받기	6-9
연습문제 6-3	웹 메소드로 이미지 생성하기	6-15
연습문제 6-4	LabVIEW 에서 HTTP 클라이언트 생성하기	6-19

LabVIEW 에서 공유 라이브러리 호출 하기 연습문제

연습문제 1-1 컴퓨터 이름

목표

Windows API 에서 DLL 함수를 호출합니다 .

시나리오

여러가지 경우에 Windows 컴퓨터 이름을 프로그램적으로 결정해야 합니다 . 예를 들어 , 중앙 서버에 데이터를 기록하는 컴퓨터가 여러 대 있는 경우 , 데이터를 추적하려면 각각의 데이터에서 컴퓨터 이름을 식별해야 합니다 . 컴퓨터 이름은 Windows 파일 공유를 통해 데이터를 저장하거나 불러올 때에도 유용하며 보고서를 생성할 때 포함시킬 유용한 정보이기도 합니다 .

LabVIEW 에는 컴퓨터 이름을 검색하는 고유 함수가 없습니다 . 그러나 , Windows 운영 시스템은 API 를 DLL 형식으로 제공하여 사용자가 운영 시스템과 상호작용할 수 있도록 합니다 . 이러한 API 의 일반적인 용도 중의 하나는 컴퓨터 이름과 같은 운영 시스템 또는 컴퓨터에 관한 유용한 정보를 검색하는 것입니다 .

Windows API DLL 함수를 호출하여 컴퓨터 이름을 알 수 있습니다 . 또한 함수가 반환하는 에러도 적절한 방법으로 처리해야 합니다 .



노트

Windows 컴퓨터 이름은 컴퓨터의 네트워크 주소와는 다릅니다 . TCP/IP 팔레트에 있는 (문자열을 IP 로) 및 (IP 를 문자열로) 함수를 사용하면 LabVIEW 에서 컴퓨터의 네트워크 주소를 결정할 수 있습니다 .

디자인

입력 및 출력

테이블 1-1. 컴퓨터 이름 입력과 출력

타입	이름	프로퍼티	기본값
숫자형 컨트롤	버퍼 크기	부호없는 32 비트 정수	256
문자열 인디케이터	컴퓨터 이름	—	비어있음

프로그램 구조

DLL 에서 함수를 호출할 때 , DLL 은 종종 에러를 보고하는 방법을 정의합니다 . Windows API 는 각 함수 호출에서 함수가 올바르게 실행되는지를 나타내는 숫자형 값을 반환합니다 . 올바르게 실행되지 않는 경우 , 다른 Windows API 함수를 호출하여 에러 코드를 판별하고 판별한 에러 코드를 에러를 설명하는 문자열로 변환합니다 . 에러를 올바르게 판별하려면 , 에러를 발생시키는 함수가 호출되면

곧바로 에러 핸들링 함수를 호출해야 합니다. 동일한 프로그램에서 다른 Windows API 가 호출되면 발생한 에러 정보가 손실되기 때문입니다. 이같은 에러 핸들링 메커니즘은 각 SubVI가 자체적으로 에러를 리포트하고 프로그램이나 섹션 끝에서 에러가 함께 연결되어 처리되는 LabVIEW 의 일반적인 에러 핸들링 메커니즘과는 다릅니다.

컴퓨터 이름 가져오기

Windows API 는 Kernel32.dll 에 GetComputerName 이라는 함수를 제공합니다. 함수가 반환하는 문자열을 저장하기 충분한 크기의 문자열 버퍼를 할당해야 합니다. 다행히 LabVIEW 8.20 또는 그 이후 버전에서 라이브러리 함수 호출 노드는 함수에 전달되는 크기 파라미터에 기반하여 자동으로 버퍼를 할당합니다. 이 함수는 버퍼가 너무 작아서 컴퓨터 이름을 저장할 수 없거나 다른 에러가 발생하면 0 을 반환합니다. 라이브러리 함수 호출 노드의 에러 출력 클러스터와 이 함수의 반환 값을 사용하여 상태 머신의 변환 로직을 컨트롤합니다.

에러 처리

이 프로그램에서는 일반적인 LabVIEW 에러 (예를 들어, 라이브러리 함수 호출 노드에서 DLL 파일 없음) 와 Windows API 에러 (예를 들어, GetComputerName 함수에서 컴퓨터 이름을 저장할 버퍼 크기가 충분하지 않음) 가 발생할 수 있습니다. 이 에러 처리 케이스는 두 가지 타입의 에러를 모두 처리해야 합니다.

(단순 에러 핸들러) VI 대신 (일반 에러 핸들러) VI 를 사용하여 Windows API 에러 리포트 호출이 사용자 정의 메시지를 생성할 수 있도록 하십시오.

Windows API 호출에서 의미를 전달할 수 있는 에러 메시지를 제공하려면, 다음 두 단계를 수행해야 합니다. 먼저, Windows API 함수 GetLastError 를 호출해야 합니다. 이 함수는 마지막 함수인 GetComputerName 호출 중에 발견된 모든 에러를 나타내는 숫자값을 반환합니다. 이 숫자 코드를 의미전달이 가능한 메시지로 변환하려면, Windows API 함수 FormatMessage 를 호출해야 합니다. 이 함수는 다양한 방법으로 사용할 수 있습니다. dwFlags 라는 파라미터에서 하나 이상의 플래그를 설정하여 함수의 동작을 제어합니다. 시스템 에러에 대한 메시지를 반환하려면, FORMAT_MESSAGE_FROM_SYSTEM 플래그를 설정하십시오. 또한, 메시지에 다른 파라미터를 포함시킬 필요가 없으므로 FORMAT_MESSAGE_IGNORE_INSERTS 플래그를 설정하십시오. 이 파라미터를 설정하면 함수의 arguments 파라미터를 무시하여 함수 호출을 간소화할 수 있습니다. 플래그 설정에 대한 자세한 내용은 *배경: Windows API 참조* 섹션을 참조하십시오.



팁

이 연습문제에서 호출하는 모든 함수는 kernel32.dll 에 있습니다. 그러나 다른 많은 DLL 에도 Windows API 함수가 있습니다. 함수가 어떤 DLL 에 위치하고 있는지 알려면 *Windows API 참조*를 참조하십시오.

추가 정보

다음 섹션에서는 솔루션을 구현하기 위해 해결해야 하는 Windows API 호출과 관련된 몇 가지 문제에 대해 설명합니다.

배경: Windows API 참조

이 연습문제에서 사용하는 각 함수에 대한 Windows API 정의는 <Exercises>\LabVIEW Connectivity\Computer Name 디렉토리에 있습니다. 이 디렉토리에서 각 PDF 파일을 열고 해당 내용을 참조하십시오. 연습문제를 진행하면서 함수에 전달하는 각 파라미터의 의미를 파악하려면 이 자료를 참조하십시오.

**노트**

이 매뉴얼의 함수 정보는 *MSDN 라이브러리*의 *Windows API 참조*에서 발췌한 내용입니다. 전체 *Windows API 참조*는 <http://msdn.microsoft.com/en-us/library/Aa383749> 에서 찾을 수 있습니다.

비트마스크 플래그 (Bitmasked Flag)

API 의 일부 함수에는 정수 비트를 마스크 (mask) 하여 하나 이상의 옵션을 설정할 수 있게 하는 플래그 파라미터가 있습니다. `FormatMessage` 함수의 `dwFlags` 파라미터가 이와 같은 예입니다.

LabVIEW 에서 플래그를 설정하려면, 적절한 크기와 값을 가진 정수 상수를 라이브러리 함수 호출 노드에 전달해야 합니다. 플래그 값을 더 쉽게 입력할 수 있도록 숫자형 상수의 표시 방법을 변경할 수 있음을 기억하십시오. 예를 들어, `FormatMessage` 함수에 대한 스펙은 플래그를 16 진수 포맷으로 제공합니다. 숫자형 상수의 포맷을 16 진수 포맷으로 설정한 후 값을 직접 입력할 수 있습니다.

플래그 파라미터에서 여러 플래그를 설정해야 할 경우, 비트 단위 OR (Bitwise Or) 함수를 사용하여 여러 플래그를 결합할 수 있습니다. LabVIEW 에서 OR 함수는 두 정수를 연결하면 자동으로 비트 단위 OR 이 되는 다형성 함수입니다.

데이터 타입

Windows API 및 다른 DLL 함수 스펙은 종종 LabVIEW 에서 이름이 다른 데이터 타입을 가리킵니다. 테이블 1-2 는 이 연습문제에서 사용하는 몇 가지 Windows 데이터 타입에 대한 LabVIEW 타입을 나타냅니다. 더 자세한 리스트는 NI 예제 탐색기에서 제공하는 DLL 호출 예제를 통해 각 데이터 타입별 예제와 전체 데이터 타입 변환 리스트를 참조하십시오. Windows 데이터 타입의 참조도 유용한 참조이며 <http://msdn.microsoft.com/en-us/library/Aa383751> 에서 찾을 수 있습니다.

테이블 1-2. 데이터 타입 변환

Windows 데이터 타입	LabVIEW 데이터 타입
LPTSTR	문자열 (C 문자열 포인터로 전달)
DWORD	U32
LPDWORD	U32 (포인터에 의한 전달)
BOOL	I32
va_list*	다양함 ((타입에 적용) 사용)
LPCVOID	다양함 ((타입에 적용) 사용)

문자열 타입

Windows API 함수는 대부분 두 가지 문자열 표시 방법을 지원합니다. 첫 번째 방법은 ASCII 로, 하나의 바이트가 문자열의 각 문자를 표시합니다. 전통적인 ASCII 는 128 문자 세트로 되어 있으며 영어의 모든 대문자, 소문자, 기타 공통적인 기호, 컨트롤 문자를 포함합니다. 일반적으로 LabVIEW 도 ASCII 를 사용하여 문자열을 표시합니다.

두 번째 방법은 유니코드 문자열 (UTF-16 인코딩 , 또는 "wide" 문자열) 표시를 사용합니다 . 유니코드에서는 2 바이트가 각 문자를 표시하므로 , 일반적인 ASCII 문자보다 훨씬 많은 문자 세트를 사용합니다 . LabVIEW 는 기본적으로 유니코드를 지원하지 않지만 , 애드온 라이브러리카 외부 함수의 호출을 통해 유니코드를 사용할 수 있습니다 .

문자열을 처리하는 Widows API 함수는 종종 DLL 에서 ASCII 에서 a, 유니코드에서는 w 라는 표시가 들어가는 두 가지 버전을 가집니다 . 예를 들어 , kernel32.dll 의 GetComputerName 함수는 GetComputerNameA 및 GetComputerNameW 버전이 있습니다 . 대부분의 경우 , LabVIEW 에서는 이 함수의 "A" 버전을 사용합니다 .

구현

1. 그림 1-1 과 같은 프런트패널을 만듭니다 .

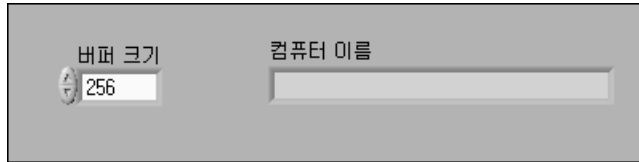


그림 1-1. 컴퓨터 이름 프런트패널

- 새 VI 를 만들고 <Exercises>\LabVIEW Connectivity\Computer Name 디렉토리에 Computer Name.vi 로 저장합니다 .
 - 테이블 1-1 에서 설명한 대로 **버퍼 크기** 컨트롤을 만듭니다 .
 - 테이블 1-1 에서 설명한 대로 **컴퓨터 이름** 인디케이터를 만듭니다 .
2. 테이블 1-3 과 같은 파라미터를 가지는 GetComputerName 함수를 호출합니다 .

테이블 1-3. GetComputerName 파라미터

파라미터 이름	타입	포맷 / 테이터 타입	그외
return type	숫자형	부호있는 32 비트 정수	
lpBuffer	문자열	C 문자열 포인터	최소 크기 에서 lpnSize 선택
lpnSize	숫자형	부호없는 32 비트 정수	전달 에서 값의 포인터 선택

- <Exercises>\LabVIEW Connectivity\Computer Name\getcomputername.pdf 에서 GetComputerName 함수 참조를 열고 함수 원형과 파라미터를 확인합니다 .
- 라이브러리 함수 호출 노드를 놓습니다 .
- 라이브러리 함수 호출 노드를 더블 클릭하여 **라이브러리 함수 호출** 대화 상자를 엽니다 .



□ 함수 탭을 클릭하고 그림 1-2 와 같이 셋팅을 설정합니다 .

- 탐색 버튼을 클릭하고 \Windows\System32\kernel32.dll 을 탐색하거나 kernel32.dll 를 입력합니다 .
- 함수 이름 풀다운 메뉴에서 **GetComputerNameA** 를 선택합니다 .
- 스레드 섹션에서 **모든 스레드에서 실행**을 선택합니다 .



노트

Windows API 함수는 재호출 (멀티 스레드형) 되므로 , 이 스레드에서 GetComputerNameA 함수를 호출하는 것은 제대로 동작합니다 . 단 , GetLastError 에서 에러가 접근하기에 적합한 메모리 위치에 저장되지 않으면 동작이 올바르지 않습니다 .

- 호출 형식 섹션에서 **표준 호출 (WINAPI)** 을 선택합니다 .

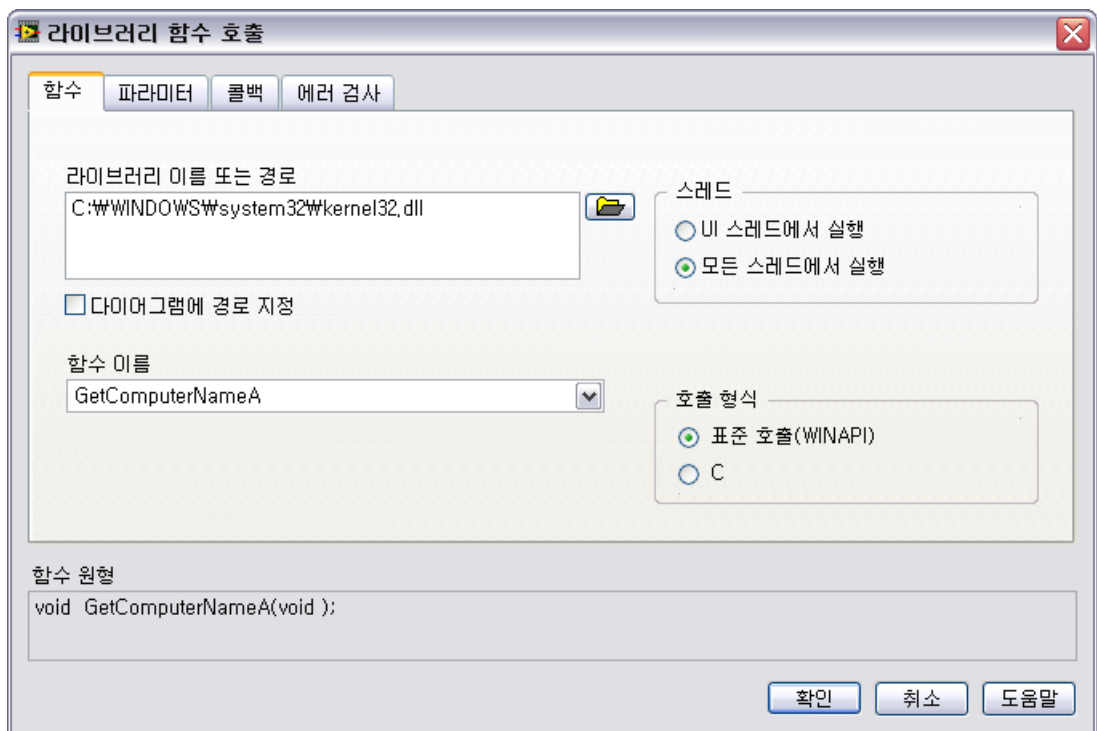


그림 1-2. GetComputerNameA 라이브러리 함수 호출

□ 파라미터 탭을 클릭합니다 .

- return type 파라미터가 선택되어 있는지 확인합니다 .
- 현재 파라미터 섹션의 타입 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- 데이터 타입 풀다운 메뉴가 **32 비트 부호 정수**로 설정되어 있는지 확인합니다 .



- + 버튼을 클릭하여 반환 타입 파라미터 다음에 파라미터를 추가합니다 .
- 이름 텍스트 박스에 lpBuffer 를 입력합니다 .

- **타입** 풀다운 메뉴에서 **문자열**을 선택합니다 .
- **문자열 포맷** 풀다운 메뉴가 **C 문자열 포인터**로 설정되어 있는지 확인합니다 .



노트

lpBuffer 파라미터의 최소 크기를 설정하기 전 파라미터를 추가적으로 선언해야 합니다 . **최소 크기**를 공란으로 둡니다 .

- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 lpnSize 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- **데이터 타입** 풀다운 메뉴에서 **부호없는 32 비트 정수**를 선택합니다 .
- **전달** 풀다운 메뉴에서 **값의 포인터**를 선택합니다 .
- 파라미터 리스트에서 **lpBuffer** 파라미터를 선택합니다 .
- **최소 크기** 풀다운 메뉴에서 **lpnSize** 를 선택합니다 .
- 함수 원형이 다음과 같은지 확인합니다 .

```
int32_t GetComputerNameA(CStr lpBuffer,
uint32_t *lpnSize);
```

확인 버튼을 클릭합니다 .

3. 테이블 1-4 와 같은 파라미터를 갖는 GetLastError 함수를 호출합니다 .

테이블 1-4. GetLastError 파라미터

파라미터 이름	타입	포맷 / 데이터 타입	그외
return type	숫자형	부호없는 32 비트 정수	—

- <Exercises>\LabVIEW Connectivity\Computer Name\getlasterror.pdf 에서 GetLastError 함수 참조를 열고 함수 원형과 파라미터를 확인합니다 .
- GetComputerName 호출의 다음에 또 하나의 라이브러리 함수 호출 노드를 놓습니다 .
- 라이브러리 함수 호출 노드를 더블 클릭하여 **라이브러리 함수 호출** 대화 상자를 엽니다 .
- 함수** 탭을 클릭합니다 .
 - **탐색** 버튼을 클릭하고 \windows\system32\kernel32.dll 을 탐색하거나 kernel32.dll 을 입력합니다 .
 - **함수 이름** 풀다운 메뉴에서 **GetLastError** 를 선택합니다 .
 - **스레드** 섹션에서 **모든 스레드에서 실행**을 선택합니다 .
 - **호출 형식** 섹션에서 **표준 호출 (WINAPI)** 을 선택합니다 .

파라미터 탭을 클릭합니다 .

- return type 파라미터가 선택되어 있는지 확인합니다 .
- **타입** 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- **타입** 풀다운 메뉴에서 **부호없는 32 비트 정수**를 선택합니다 .
- 함수 원형이 다음과 같은지 확인합니다 .

```
uint32_t GetLastError(void);
```

확인을 클릭합니다 .

4. 다음 아이템을 사용하여 블록다이어그램을 그림 1-3 과 같이 만듭니다 .

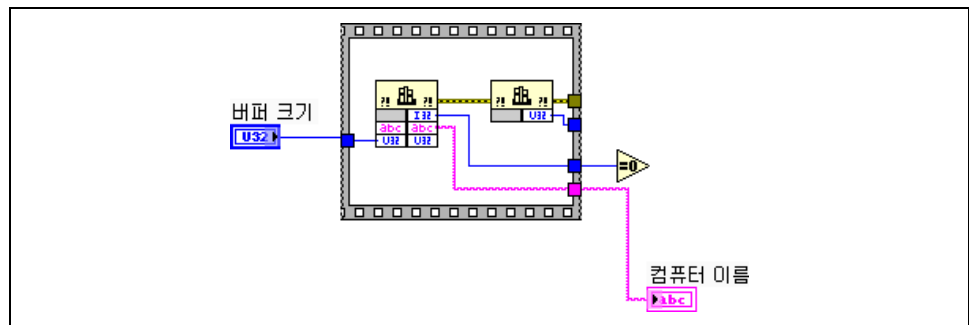


그림 1-3. 컴퓨터 이름 가져오기

플랫폼 시퀀스 구조 -2 개의 라이브러리 함수 호출 노드를 둘러쌉니다 .



노트

플랫폼 시퀀스 구조를 사용하면 VI 가 dll 에서 GetComputerNameA 함수를 호출한 후 즉시 에러를 확인할 수 있으며 , 같은 스레드에서는 동일하게 동작합니다 .

LabVIEW 실행 시스템의 작동 방식으로 인해 , 두 개의 DLL 노드 사이에 다른 동작이 실행될 수도 있습니다 . 이러한 경우 GetLastError 는 잘못된 결과를 반환할 수 있습니다 . 플랫폼 시퀀스 구조에 오직 DLL 노드 두 개만 놓게 되면 이러한 일이 일어나는 경우를 줄일 수 있습니다 .

- 0 과 같음 ?
- 컴퓨터 이름** 인디케이터
- 버퍼 크기** 컨트롤

5. 테이블 1-5 와 같은 파라미터를 갖는 FormatMessage 함수를 호출합니다 .

테이블 1-5. FormatMessage 파라미터

파라미터 이름	타입	포맷 / 테이터 타입	그외
return type	숫자형	부호없는 32 비트 정수	-
dwFlags	숫자형	부호없는 32 비트 정수	전달 에서 값 선택
lpSource	타입에 적용	핸들의 포인터	-
dwMessageId	숫자형	부호없는 32 비트 정수	전달 에서 값 선택

테이블 1-5. FormatMessage 파라미터 (계속됨)

파라미터 이름	타입	포맷 / 데이터 타입	그외
dwLanguageId	숫자형	부호없는 32 비트 정수	전달에서 값 선택
lpBuffer	문자열	C 문자열 포인터	최소 크기에서 nSize 선택
nSize	숫자형	부호없는 32 비트 정수	전달에서 값 선택
Arguments	타입에 적용	핸들의 포인터	-

- <Exercises>\LabVIEW Connectivity\Computer Name\formatmessage.pdf 에서 FormatMessage 함수 참조를 열고 함수 원형과 파라미터를 확인합니다 .
- 플랫폼 시퀀스 구조 뒤에 라이브러리 함수 호출 노드를 놓습니다 .
- 라이브러리 함수 호출 노드를 더블 클릭하여 **라이브러리 함수 호출** 대화 상자를 엽니다 .
- 함수** 탭을 선택합니다 .
 - **탐색** 버튼을 클릭하고 \Windows\System32\kernel32.dll 을 탐색하거나 kernel32.dll 를 입력합니다 .
 - **함수 이름** 풀다운 메뉴에서 **FormatMessageA** 를 선택합니다 .
 - **스레드** 섹션에서 **모든 스레드에서 실행** 을 선택합니다 .
 - **호출 형식** 섹션에서 **표준 호출 (WINAPI)** 을 선택합니다 .
- 파라미터** 탭을 클릭합니다 .
 - **return type** 파라미터가 선택되어 있는지 확인합니다 .
 - **타입** 풀다운 메뉴에서 **숫자형** 을 선택합니다 .
 - **타입** 풀다운 메뉴에서 **부호없는 32 비트 정수** 를 선택합니다 .
 - **+** 버튼을 클릭하여 파라미터를 추가합니다 .
 - **이름** 텍스트 박스에 dwFlags 를 입력합니다 .
 - **타입** 풀다운 메뉴에서 **숫자형** 을 선택합니다 .
 - **데이터 타입** 풀다운 메뉴에서 **부호없는 32 비트 정수** 를 선택합니다 .
 - **전달** 풀다운 메뉴가 **값** 으로 설정되어 있는지 확인합니다 .
 - **+** 버튼을 클릭하여 파라미터를 추가합니다 .
 - **이름** 텍스트 박스에 lpSource 를 입력합니다 .
 - **타입** 풀다운 메뉴에서 **타입에 적용** 을 선택합니다 .
 - **데이터 포맷** 풀다운 메뉴에서 **핸들의 포인터** 를 선택합니다 .

- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 `dwMessageId` 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- **데이터 타입** 풀다운 메뉴에서 **부호없는 32 비트 정수**를 선택합니다 .
- **전달** 풀다운 메뉴가 **값**으로 설정되어 있는지 확인합니다 .
- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 `dwLanguageId` 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- **데이터 타입** 풀다운 메뉴에서 **부호없는 32 비트 정수**를 선택합니다 .
- **전달** 풀다운 메뉴가 **값**으로 설정되어 있는지 확인합니다 .
- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 `lpBuffer` 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **문자열**을 선택합니다 .
- **문자열 포맷** 풀다운 메뉴가 **C 문자열 포인터**로 설정되어 있는지 확인합니다 .



노트

최소 크기 풀다운 메뉴를 설정하기 전에 파라미터를 추가적으로 선언해야 합니다 . 일단은 최소 크기를 **<None>** 으로 둡니다 .

- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 `nSize` 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **숫자형**을 선택합니다 .
- **데이터 타입** 풀다운 메뉴에서 **부호없는 32 비트 정수**를 선택합니다 .
- **전달** 풀다운 메뉴가 **값**으로 설정되어 있는지 확인합니다 .
- + 버튼을 클릭하여 파라미터를 추가합니다 .
- **이름** 텍스트 박스에 `Arguments` 를 입력합니다 .
- **타입** 풀다운 메뉴에서 **타입에 적용**을 선택합니다 .
- **데이터 포맷** 풀다운 메뉴에서 **핸들의 포인터**를 선택합니다 .
- 파라미터 리스트에서 **lpBuffer** 파라미터를 선택합니다 .
- **최소 크기** 풀다운 메뉴에서 **nSize** 를 선택합니다 .
- 함수 원형이 다음과 같은지 확인합니다 .

```

- uint32_t FormatMessageA(uint32_t dwFlags,
void *lpSource, uint32_t dwMessageId, uint32_t
dwLanguageId, CStr lpBuffer, uint32_t nSize,
void *Arguments);
    
```



노트

Void 파라미터는 임의의 데이터 타입을 허용하기 때문에 **Void** 타입의 인수에 해당하는 터미널은 연결될 때까지 비어 있습니다.

□ **확인** 버튼을 클릭합니다.

6. 다음 아이템을 사용하여 에러 핸들링 코드를 그림 1-4 와 같이 만듭니다.

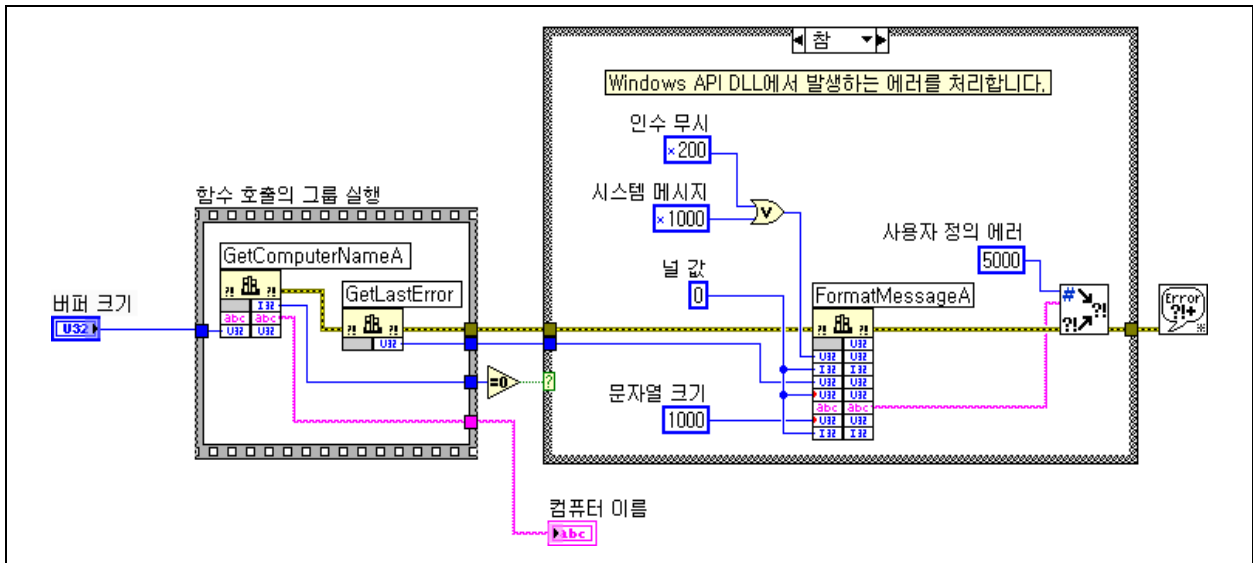


그림 1-4. 에러 핸들링 코드

□ 숫자형 상수

- 형을 U32 로 설정합니다 .
- 숫자형 상수에서 마우스 오른쪽 버튼을 클릭한 후 보이는 **아이템» 기수**를 선택합니다 .
- 기수를 클릭하고 **16 진수**를 선택합니다 .
- 상수의 값을 200 으로 설정합니다 .
- 상수에 **인수 무시**라는 라벨을 붙입니다 .

□ 숫자형 상수

- **인수 무시** 상수의 복사본을 만듭니다 .
- 새 상수에 **시스템 메시지**라는 라벨을 붙입니다 .
- **시스템 메시지** 상수의 값을 1000 으로 설정합니다 .

□ OR

- 숫자형 상수
 - 형을 I32 로 설정합니다 .
 - 상수에 **널 (Null) 값**이라는 라벨을 붙입니다 .
- 숫자형 상수
 - 형을 U32 로 설정합니다 .
 - 상수에 **문자열 크기**라는 라벨을 붙입니다 .
 - **문자열 크기** 상수의 값을 1000 으로 설정합니다 .
- (에러 코드를 에러 클러스터로) VI
 - (에러 코드를 에러 클러스터로) VI 의 **에러 코드** 입력 터미널에서 마우스 오른쪽 버튼을 클릭한 후 **생성»상수**를 선택합니다 .
 - 상수의 값을 5000 으로 설정합니다 .
 - 상수에 **사용자 정의 에러**라는 라벨을 붙입니다 .
- 케이스 구조를 에러 핸들링 코드를 돌려 싸도록 놓습니다 .
- (일반 에러 핸들러) VI



테스트

1. 적절한 버퍼 크기로 VI 를 테스트합니다 .
 - VI 를 기본 **버퍼 크기** (256) 로 실행합니다 .
 - 컴퓨터의 이름이 **컴퓨터 이름** 인디케이터에 표시되어야 합니다 .
2. 올바른 컴퓨터 이름이 표시되는지 확인합니다 .
 - 컴퓨터 바탕 화면 또는 Windows 탐색기에서 **내 컴퓨터**를 찾습니다 .
 - 내 컴퓨터**에서 마우스 오른쪽 버튼을 클릭하고 바로 가기 메뉴에서 **프로퍼티**를 선택합니다 .
 - 컴퓨터 이름** 탭을 선택하고 **전체 컴퓨터 이름**이 VI 가 반환하는 값과 같은지 확인합니다 .



노트

이름의 대소문자는 일치하지 않아도 상관없습니다 .

3. VI 에서 에러 처리를 테스트합니다 .
 - 버퍼 크기** 컨트롤을 1 로 설정하고 VI 를 실행합니다 .
 - VI 에 **파일 이름이 너무 길니다** . 라는 에러 메시지가 표시되는지 확인합니다 .



팁

이 VI 에 대해 표시되는 에러 메시지는 Windows 에러 메시지 ERROR_BUFFER_OVERFLOW (System Error 111) 에 대한 설명입니다 . 시스템 에러 코드와 해당 설명에 대해서는 http://msdn.microsoft.com/library/en-us/debug/base/system_error_codes.asp 를 참조하십시오 .

연습문제 1-1 끝

노트
