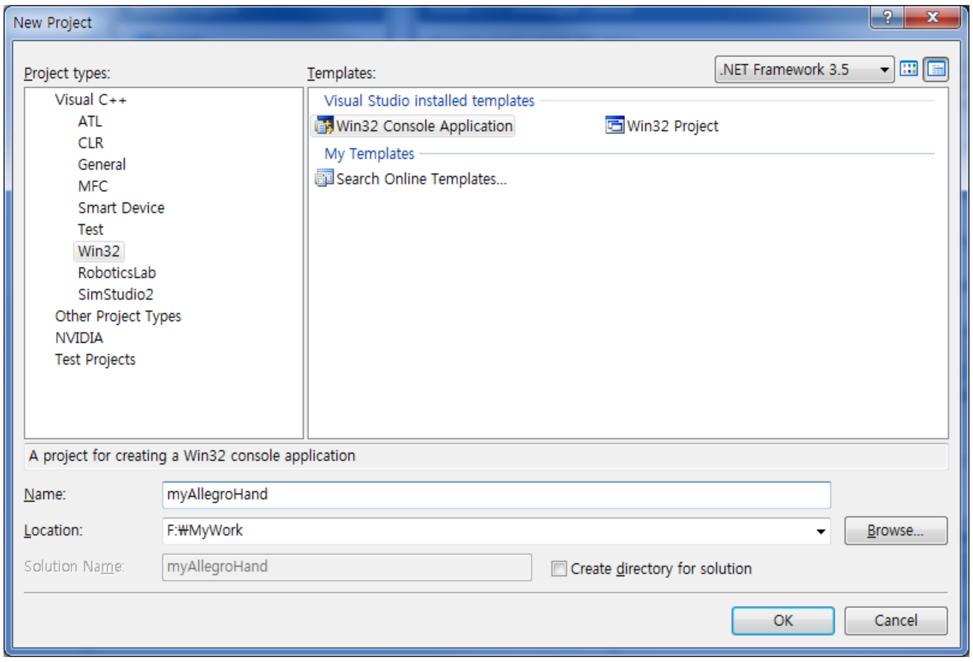
Allegro Hand Programming Tutorial

Allegro Hand Console Application

Allegro Hand 2.0

■ Creating a new project

1. Visual Studio 2008 실행한다.
2. “File > New > Project” 메뉴를 이용하여 새 프로젝트를 생성한다.
3. Win32 Console Application을 선택하고 프로젝트 이름에 myAllegroHand 라고 입력한다.



New Project

Project types: Visual C++ (ATL, CLR, General, MFC, Smart Device, Test, Win32, RoboticsLab, SimStudio2), Other Project Types, NVIDIA, Test Projects

Templates: .NET Framework 3.5

Visual Studio installed templates: Win32 Console Application, Win32 Project

My Templates: Search Online Templates...

A project for creating a Win32 console application

Name: myAllegroHand

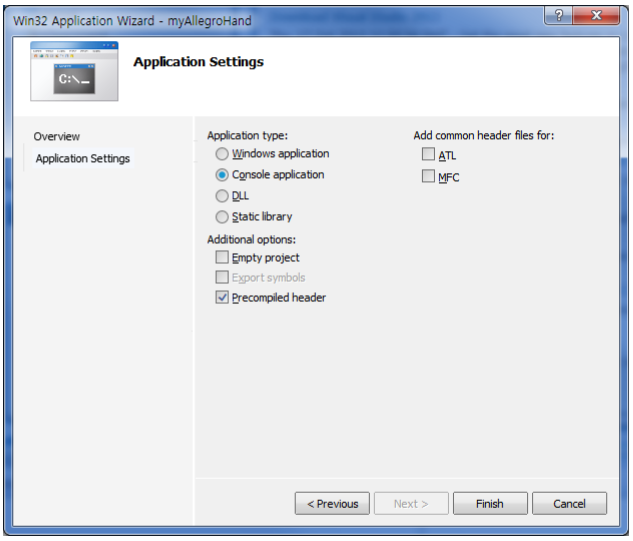
Location: F:₩MyWork

Solution Name: myAllegroHand

[ ] Create directory for solution

OK / Cancel

4. Application type에 “Console application”을 선택한 후 “Finish” 버튼을 눌러 완료한다.



Win32 Application Wizard - myAllegroHand

Application Settings

Overview / Application Settings

Application type:
- ○ Windows application
- ● Console application
- ○ DLL
- ○ Static library

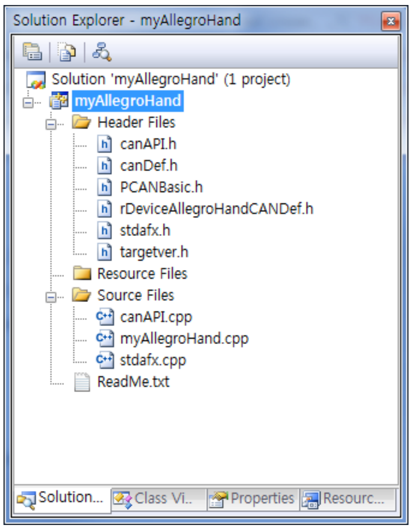
Additional options:
- [ ] Empty project
- [ ] Export symbols
- [✓] Precompiled header

Add common header files for:
- [ ] ATL
- [ ] MFC

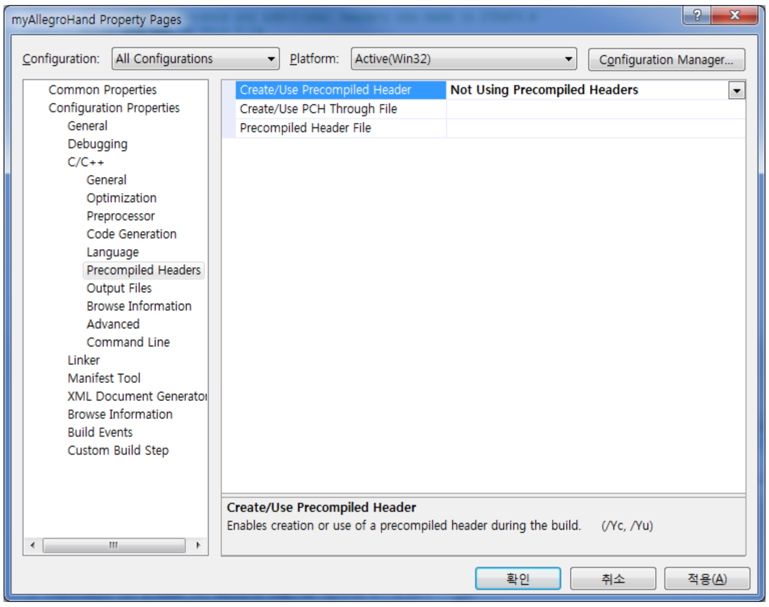
< Previous / Next > / Finish / Cancel

5. 프로젝트가 정상적으로 생성되었다면 CAN 통신 등을 위해 미리 작성된 파일들을 프로젝트가 생성된 폴더에 덮어쓴다. 그리고 프로젝트에 필요한 파일들을 추가한다. 다음은 필요 파일들을 추가한 후 솔루션 탐색기 모습이다. 프로젝트 생성시 자동 생성된 “stdafx.h”, “targetver.h”, “myAllegroHand.cpp”, “sdafx.cpp”, “ReadMe.txt” 외에 “canAPI.h”, “canDef.h”, PCANBasic.h”, “rDeviceAllegroHandCANDef.h”, “canAPI.cpp” 파일이 추가되어있다. 이 때 한가지 주의해야 할 점은 CAN 장치 제조사마다 “canAPI.cpp” 파일을 같은 이름으로 하나씩 제공하고 있기 때문에 사용하려고 하는 CAN 장치에 따라 알맞은 파일을 선택해야 한다는 것이다. 본 튜토리얼은 Peak

CAN 장치를 사용하는 것을 가정하고 있다. 때문에 “canAPI.cpp” 파일 추가시 “src₩Peak₩canAPI.cpp” 파일을 선택하여 프로젝트에 추가하였다.



6. 프로젝트 설정 다이얼로그를 열기 위해 솔루션 탐색기에서 프로젝트를 선택한 후 마우스 우클릭, 그리고 속성 메뉴를 선택한다. 프로젝트 설정 다이얼로그에서 다음과 같이 “Create/Use Precompiled Header” 속성을 “Not Using Precompiled Headers”로 변경한다.

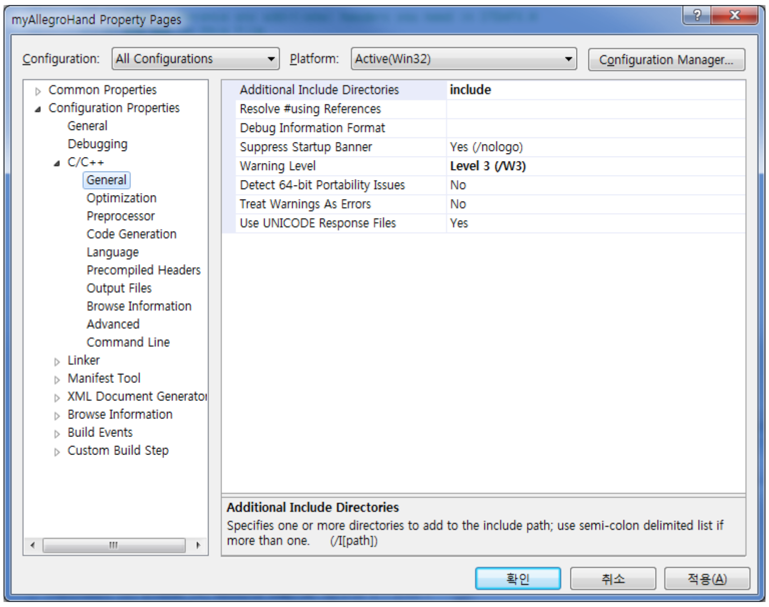


Configuration: All Configurations
Platform: Active(Win32)

Create/Use Precompiled Header	Not Using Precompiled Headers
Create/Use PCH Through File	
Precompiled Header File	

Create/Use Precompiled Header
Enables creation or use of a precompiled header during the build. (/Yc, /Yu)

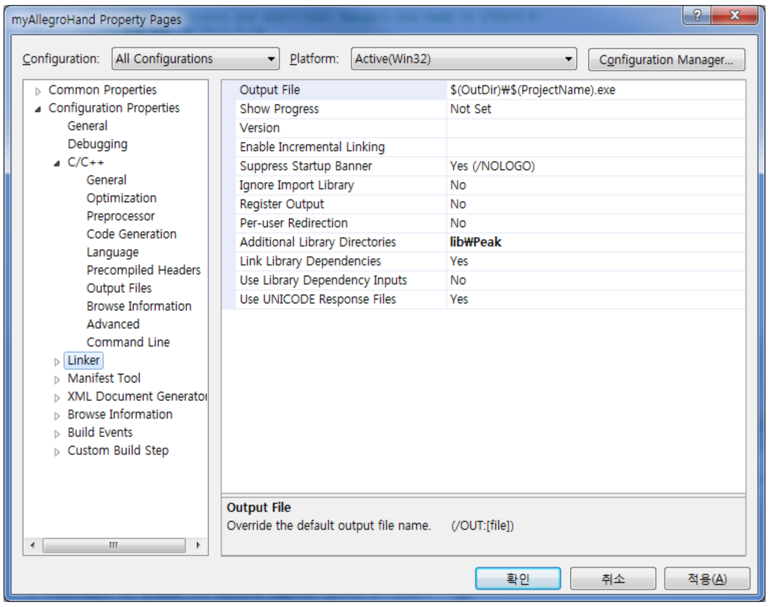
7. “Additional Include Directories”에 “include”를 추가한다.



Additional Include Directories	**include**
Resolve #using References	
Debug Information Format	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level 3 (/W3)**
Detect 64-bit Portability Issues	No
Treat Warnings As Errors	No
Use UNICODE Response Files	Yes

Additional Include Directories
Specifies one or more directories to add to the include path; use semi-colon delimited list if more than one. (/I[path])

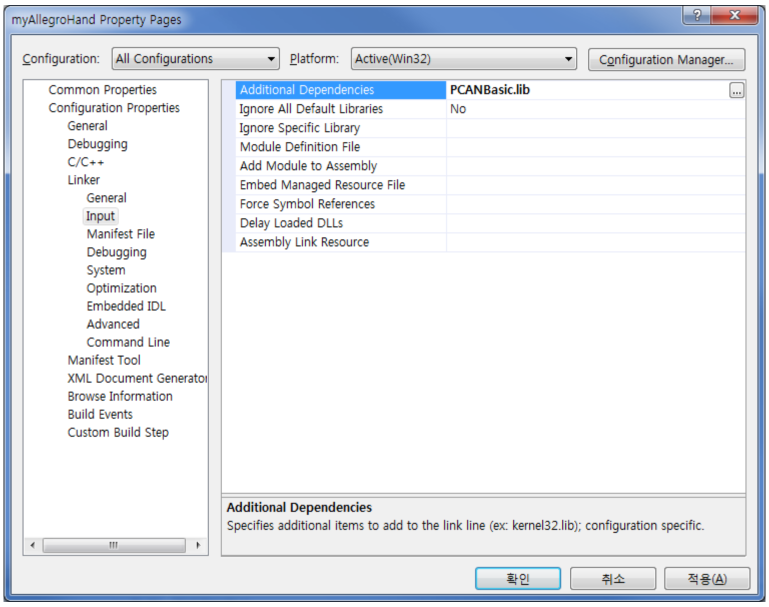
8. "Additional Library Directories"에 "lib₩Peak"를 추가한다. 본 튜토리얼은 Peak CAN 장치를 이용하는 것을 가정하고 있다. 만약 Peak CAN 이외의 다른 CAN 장치를 이용하는 경우 lib 폴더에서 해당 장치 제조사 이름의 폴더를 찾아 그 이름을 설정하도록 한다. 현재 Allegro Hand는 "ESD-CAN", "Kvaser", "NI-CAN", "Peak", "Softing" CAN 장치를 지원한다.



Output File	$(OutDir)₩$(ProjectName).exe
Show Progress	Not Set
Version	
Enable Incremental Linking	
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**lib₩Peak**
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Use UNICODE Response Files	Yes

Output File
Override the default output file name. (/OUT:[file])

9. "Additional Dependancies"에 "PCANBasic.lib"을 추가한다.

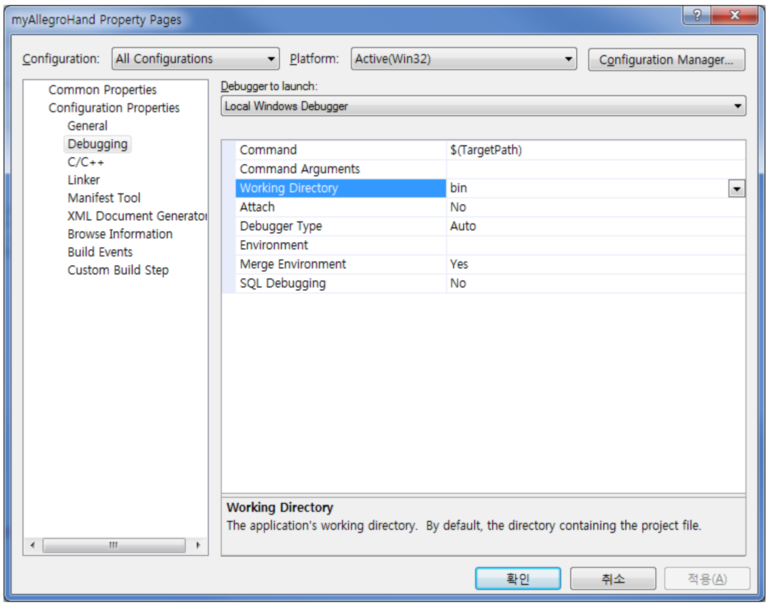


Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...

Additional Dependencies	PCANBasic.lib
Ignore All Default Libraries	No
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

10. 실행 경로를 "bin"으로 설정한다.



Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...

Debugger to launch:
Local Windows Debugger

Command	$(TargetPath)
Command Arguments	
Working Directory	bin
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No

Working Directory
The application's working directory. By default, the directory containing the project file.

■ CAN implementation

11. 이제 myAllegroHand.cpp 파일을 연다. 먼저 필요한 헤더 파일을 추가한다.

```
#include "stdafx.h"
#include "windows.h"
#include <conio.h>
#include <process.h>
#include <tchar.h>
#include "canAPI.h"
#include "rDeviceAllegroHandCANDef.h"
```

12. CAN 통신을 위한 전역변수를 선언한다. “CAN_Ch” 변수는 CAN 통신 채널을 위한 인덱스를 저장하기 위한 변수이다. “vars” 변수는 CAN 통신을 통해 수신된 각 관절 엔코더값 및 각 관절에 인가할 토크 명령을 저장한다.

```
int CAN_Ch = 0;
bool ioThreadRun = false;
uintptr_t ioThread = 0;
int recvNum = 0;
int sendNum = 0;
double statTime = -1.0;
double curTime = 0.0;
AllegroHand_DeviceMemory_t vars;
```

13. “main” 함수를 다음과 같이 편집한다. Main 함수는 크게 네가지 함수로 구성된다. 다음에 각 함수에 대해 설명한다.

```
int _tmain(int argc, _TCHAR* argv[])
{
		PrintInstruction();

		if (OpenCAN())
		{
				MainLoop();
				CloseCAN();
		}

		return 0;
};
```

14. “PrintInstruction()” 함수는 프로그램 정보 및 실행 방법 등을 콘솔에 출력하는 함수이다. 구현은 다음과 같다.

```
void PrintInstruction()
{
		printf("--------------------------------------------------\n");
		printf("myAllegroHand\n");
		printf("\n");
		printf("To quit this program, press 'q'.\n");
		printf("--------------------------------------------------\n");
}
```

15. “OpenCAN()” 함수는 CAN 통신을 초기화하고 CAN을 통해 데이터 수신 및 송신을 위한 별도의 쓰레드(thread)를 생성한다. 그리고 Allegro Hand 시스템을 초기화하고 제어를 위해 주기적 통신을 시작한다.

```
bool OpenCAN()
{
		int ret;

		CAN_Ch = GetCANChannelIndex(_T("USBBUS1"));

		printf(">CAN(%d): open\n", CAN_Ch);
		ret = command_can_open(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_canopen !!! \n");
				return false;
		}

		memset(&vars, 0, sizeof(vars));
		recvNum = 0;
		sendNum = 0;
		statTime = 0.0;

		ioThreadRun = true;
		ioThread = _beginthreadex(NULL, 0, ioThreadProc, NULL, 0, NULL);
		printf(">CAN: starts listening CAN frames\n");

		printf(">CAN: system init\n");
		ret = command_can_sys_init(CAN_Ch, 3/*msec*/);
		if(ret < 0)
		{
				printf("ERROR command_can_sys_init !!! \n");
				command_can_close(CAN_Ch);
				return false;
		}
		printf(">CAN: start periodic communication\n");
		ret = command_can_start(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_can_start !!! \n");
				command_can_stop(CAN_Ch);
				command_can_close(CAN_Ch);
				return false;
		}

		return true;
}
```

16. "CloseCAN()" 함수는 통신 쓰레드 및 CAN 통신을 종료한다.

```
void CloseCAN()
{
		int ret;

		printf(">CAN: stop periodic communication\n");
		ret = command_can_stop(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_can_stop !!! \n");
		}

		printf(">CAN: stoped listening CAN frames\n");
		ioThreadRun = false;
		WaitForSingleObject((HANDLE)ioThread, INFINITE);
		CloseHandle((HANDLE)ioThread);
		ioThread = 0;

		printf(">CAN(%d): close\n", CAN_Ch);
		ret = command_can_close(CAN_Ch);
		if(ret < 0) printf("ERROR command_can_close !!! \n");
}
```

17. 그 외 중요 함수로 "ioThreadProc" 함수가 있다. 이는 CAN 통신을 처리하는 쓰레드 핸들러이다. Allegro Hand는 3msec마다 모든 관절의 엔코더값을 4개의 CAN frame에 실어 PC로 전송하는데, 4개의 CAN frame을 모두 받으면 계산된 관절 토크를 Allegro Hand로 전송한다.

```
static unsigned int __stdcall ioThreadProc(void* inst)
{
		char id_des;
		char id_cmd;
		char id_src;
		int len;
		unsigned char data[8];
		unsigned char data_return = 0;


		while (ioThreadRun)
		{
				while (0 == get_message(CAN_Ch, &id_cmd, &id_src, &id_des, &len, data, FALSE))
				{
						switch (id_cmd)
						{
						case ID_CMD_QUERY_CONTROL_DATA:
								{
										if (id_src >= ID_DEVICE_SUB_01 && id_src <= ID_DEVICE_SUB_04)
										{
												vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 0] = (int)(data[0] | (data[1] << 8));
												vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 1] = (int)(data[2] | (data[3] << 8));
												vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 2] = (int)(data[4] | (data[5] << 8));
												vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 3] = (int)(data[6] | (data[7] << 8));
												data_return |= (0x01 << (id_src-ID_DEVICE_SUB_01));
												recvNum++;
										}
										if (data_return == (0x01 | 0x02 | 0x04 | 0x08))
										{
												// send torques
												for (int i=0; i<4;i++)
												{
														write_current(CAN_Ch, i, &vars.pwm_demand[4*i]);
														for(int k=0; k<100000; k++);
												}
												sendNum++;
												data_return = 0;
										}
								}
								break;
						}
				}
		}

		return 0;
}
```

18. "GetCANChannelIndex()" 함수는 Peak CAN에서 사용하기 위한 함수로 Peak CAN이 지원하는 포트 이름을 입력하여 그에 상응하는 인덱스를 리턴한다. "CAN_Ch" 변수를 초기화하기 위해 사용

된다.

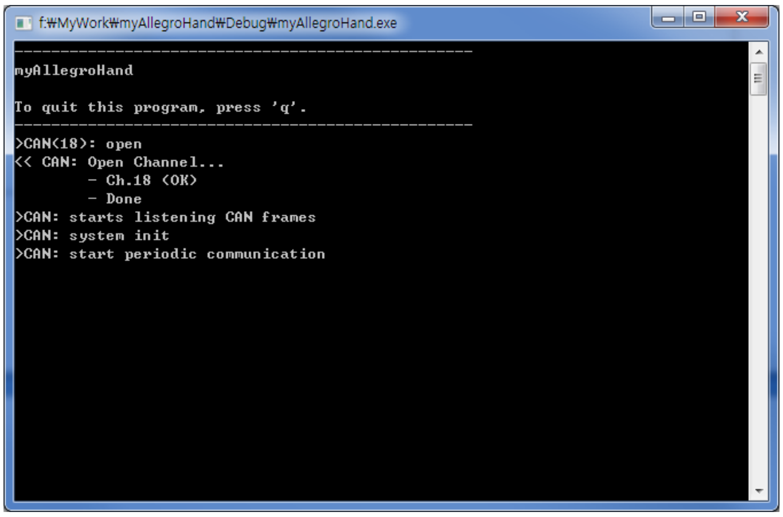
19. 다음은 전체 소스이다.

```
// myAllegroHand.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "windows.h"
#include <conio.h>
#include <process.h>
#include <tchar.h>
#include "canAPI.h"
#include "rDeviceAllegroHandCANDef.h"

int CAN_Ch = 0;
bool ioThreadRun = false;
uintptr_t ioThread = 0;
int recvNum = 0;
int sendNum = 0;
double statTime = -1.0;
AllegroHand_DeviceMemory_t vars;

void PrintInstruction();
void MainLoop();
bool OpenCAN();
void CloseCAN();
int GetCANChannelIndex(const TCHAR* cname);

static unsigned int __stdcall ioThreadProc(void* inst)
{
	char id_des;
	char id_cmd;
	char id_src;
	int len;
	unsigned char data[8];
	unsigned char data_return = 0;


	while (ioThreadRun)
	{
		while (0 == get_message(CAN_Ch, &id_cmd, &id_src, &id_des, &len, data, FALSE))
		{
			switch (id_cmd)
			{
			case ID_CMD_QUERY_CONTROL_DATA:
				{
					if (id_src >= ID_DEVICE_SUB_01 && id_src <= ID_DEVICE_SUB_04)
					{
						vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 0] = (int)(data[0] | (data[1] << 8));
						vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 1] = (int)(data[2] | (data[3] << 8));
						vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 2] = (int)(data[4] | (data[5] << 8));
						vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 3] = (int)(data[6] | (data[7] << 8));
						data_return |= (0x01 << (id_src-ID_DEVICE_SUB_01));
						recvNum++;
					}
					if (data_return == (0x01 | 0x02 | 0x04 | 0x08))
					{
						// send torques
						for (int i=0; i<4;i++)
						{
							write_current(CAN_Ch, i, &vars.pwm_demand[4*i]);
							for(int k=0; k<100000; k++);
						}
						sendNum++;
						data_return = 0;
					}
				}
				break;
			}
		}
	}

	return 0;
}

void MainLoop()
{
	bool bRun = true;
	int i;

	while (bRun)
	{
		if (!_kbhit())
		{
			Sleep(5);
		}
		else
		{
			int c = _getch();
			switch (c)
			{
			case 'q':
				bRun = false;
				break;
			}
		}
	}
```

```
}

bool OpenCAN()
{
		int ret;

		CAN_Ch = GetCANChannelIndex(_T("USBBUS1"));

		printf(">CAN(%d): open\n", CAN_Ch);
		ret = command_can_open(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_canopen !!! \n");
				return false;
		}

		memset(&vars, 0, sizeof(vars));
		recvNum = 0;
		sendNum = 0;
		statTime = 0.0;

		ioThreadRun = true;
		ioThread = _beginthreadex(NULL, 0, ioThreadProc, NULL, 0, NULL);
		printf(">CAN: starts listening CAN frames\n");

		printf(">CAN: system init\n");
		ret = command_can_sys_init(CAN_Ch, 3/*msec*/);
		if(ret < 0)
		{
				printf("ERROR command_can_sys_init !!! \n");
				command_can_close(CAN_Ch);
				return false;
		}

		printf(">CAN: start periodic communication\n");
		ret = command_can_start(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_can_start !!! \n");
				command_can_stop(CAN_Ch);
				command_can_close(CAN_Ch);
				return false;
		}

		return true;
}

void CloseCAN()
{
		int ret;

		printf(">CAN: stop periodic communication\n");
		ret = command_can_stop(CAN_Ch);
		if(ret < 0)
		{
				printf("ERROR command_can_stop !!! \n");
		}

		printf(">CAN: stoped listening CAN frames\n");
		ioThreadRun = false;
		WaitForSingleObject((HANDLE)ioThread, INFINITE);
		CloseHandle((HANDLE)ioThread);
		ioThread = 0;

		printf(">CAN(%d): close\n", CAN_Ch);
		ret = command_can_close(CAN_Ch);
		if(ret < 0) printf("ERROR command_can_close !!! \n");
}

void PrintInstruction()
{
		printf("--------------------------------------------------\n");
		printf("myAllegroHand\n");
		printf("\n");
		printf("To quit this program, press 'q'.\n");
		printf("--------------------------------------------------\n");
}

int GetCANChannelIndex(const TCHAR* cname)
{
		if (!cname) return 0;

		if (!_tcsicmp(cname, _T("0")) || !_tcsicmp(cname, _T("PCAN_NONEBUS")) || !_tcsicmp(cname, _T("NONEBUS")))
				return 0;
		else if (!_tcsicmp(cname, _T("1")) || !_tcsicmp(cname, _T("PCAN_ISABUS1")) || !_tcsicmp(cname, _T("ISABUS1")))
				return 1;
		else if (!_tcsicmp(cname, _T("2")) || !_tcsicmp(cname, _T("PCAN_ISABUS2")) || !_tcsicmp(cname, _T("ISABUS2")))
				return 2;
		else if (!_tcsicmp(cname, _T("3")) || !_tcsicmp(cname, _T("PCAN_ISABUS3")) || !_tcsicmp(cname, _T("ISABUS3")))
				return 3;
		else if (!_tcsicmp(cname, _T("4")) || !_tcsicmp(cname, _T("PCAN_ISABUS4")) || !_tcsicmp(cname, _T("ISABUS4")))
				return 4;
		else if (!_tcsicmp(cname, _T("5")) || !_tcsicmp(cname, _T("PCAN_ISABUS5")) || !_tcsicmp(cname, _T("ISABUS5")))
				return 5;
		else if (!_tcsicmp(cname, _T("7")) || !_tcsicmp(cname, _T("PCAN_ISABUS6")) || !_tcsicmp(cname, _T("ISABUS6")))
				return 6;
		else if (!_tcsicmp(cname, _T("8")) || !_tcsicmp(cname, _T("PCAN_ISABUS7")) || !_tcsicmp(cname, _T("ISABUS7")))
				return 7;
		else if (!_tcsicmp(cname, _T("8")) || !_tcsicmp(cname, _T("PCAN_ISABUS8")) || !_tcsicmp(cname, _T("ISABUS8")))
```

```
			return 8;
		else if (!_tcsicmp(cname, _T("9")) || !_tcsicmp(cname, _T("PCAN_DNGBUS1")) || !_tcsicmp(cname, _T("DNGBUS1")))
			return 9;
		else if (!_tcsicmp(cname, _T("10")) || !_tcsicmp(cname, _T("PCAN_PCIBUS1")) || !_tcsicmp(cname, _T("PCIBUS1")))
			return 10;
		else if (!_tcsicmp(cname, _T("11")) || !_tcsicmp(cname, _T("PCAN_PCIBUS2")) || !_tcsicmp(cname, _T("PCIBUS2")))
			return 11;
		else if (!_tcsicmp(cname, _T("12")) || !_tcsicmp(cname, _T("PCAN_PCIBUS3")) || !_tcsicmp(cname, _T("PCIBUS3")))
			return 12;
		else if (!_tcsicmp(cname, _T("13")) || !_tcsicmp(cname, _T("PCAN_PCIBUS4")) || !_tcsicmp(cname, _T("PCIBUS4")))
			return 13;
		else if (!_tcsicmp(cname, _T("14")) || !_tcsicmp(cname, _T("PCAN_PCIBUS5")) || !_tcsicmp(cname, _T("PCIBUS5")))
			return 14;
		else if (!_tcsicmp(cname, _T("15")) || !_tcsicmp(cname, _T("PCAN_PCIBUS6")) || !_tcsicmp(cname, _T("PCIBUS6")))
			return 15;
		else if (!_tcsicmp(cname, _T("16")) || !_tcsicmp(cname, _T("PCAN_PCIBUS7")) || !_tcsicmp(cname, _T("PCIBUS7")))
			return 16;
		else if (!_tcsicmp(cname, _T("17")) || !_tcsicmp(cname, _T("PCAN_PCIBUS8")) || !_tcsicmp(cname, _T("PCIBUS8")))
			return 17;
		else if (!_tcsicmp(cname, _T("18")) || !_tcsicmp(cname, _T("PCAN_USBBUS1")) || !_tcsicmp(cname, _T("USBBUS1")))
			return 18;
		else if (!_tcsicmp(cname, _T("19")) || !_tcsicmp(cname, _T("PCAN_USBBUS2")) || !_tcsicmp(cname, _T("USBBUS2")))
			return 19;
		else if (!_tcsicmp(cname, _T("20")) || !_tcsicmp(cname, _T("PCAN_USBBUS3")) || !_tcsicmp(cname, _T("USBBUS3")))
			return 20;
		else if (!_tcsicmp(cname, _T("21")) || !_tcsicmp(cname, _T("PCAN_USBBUS4")) || !_tcsicmp(cname, _T("USBBUS4")))
			return 21;
		else if (!_tcsicmp(cname, _T("22")) || !_tcsicmp(cname, _T("PCAN_USBBUS5")) || !_tcsicmp(cname, _T("USBBUS5")))
			return 22;
		else if (!_tcsicmp(cname, _T("23")) || !_tcsicmp(cname, _T("PCAN_USBBUS6")) || !_tcsicmp(cname, _T("USBBUS6")))
			return 23;
		else if (!_tcsicmp(cname, _T("24")) || !_tcsicmp(cname, _T("PCAN_USBBUS7")) || !_tcsicmp(cname, _T("USBBUS7")))
			return 24;
		else if (!_tcsicmp(cname, _T("25")) || !_tcsicmp(cname, _T("PCAN_USBBUS8")) || !_tcsicmp(cname, _T("USBBUS8")))
			return 25;
		else if (!_tcsicmp(cname, _T("26")) || !_tcsicmp(cname, _T("PCAN_PCCBUS1")) || !_tcsicmp(cname, _T("PCCBUS1")))
			return 26;
		else if (!_tcsicmp(cname, _T("27")) || !_tcsicmp(cname, _T("PCAN_PCCBUS2")) || !_tcsicmp(cname, _T("PCCBUS2")))
			return 271;
		else
			return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
		PrintInstruction();

		if (OpenCAN())
		{
			MainLoop();
			CloseCAN();
		}

		return 0;
}
```

20. 프로그램을 빌드한다.
21. Allegro Hand에 전원을 인가하고 프로그램을 실행한다. 다음과 같은 메시지가 보인다면 프로그램이 잘 실행되는 것이다.



f:₩MyWork₩myAllegroHand₩Debug₩myAllegroHand.exe

```
----------------------------------------------------
myAllegroHand

To quit this program, press 'q'.
----------------------------------------------------
>CAN(18): open
<< CAN: Open Channel...
        - Ch.18 (OK)
        - Done
>CAN: starts listening CAN frames
>CAN: system init
>CAN: start periodic communication
```

22. 'q' 버튼을 눌러 프로그램을 종료한다.

■ Monitoring joint angles using rPanelManipulator

23. 실제로 CAN 통신을 통해 엔코더값이 잘 수신되는지 확인하기 위하여 "rPanelManipulator"를 이용해보자. 이를 위하여 다음과 같이 추가적이 헤더 파일을 인크루드해야 한다.

```
#include "rPanelManipulatorCmdUtil.h"
```

24. 전역변수를 선언한다.

```
rPanelManipulatorData_t* pSHM = NULL;
```

25. "main" 함수에서 "pSHM" 변수를 초기화한다. 그리고 프로그램 종료전 사용을 해지한다. 다음은 수정된 "main" 함수이다.

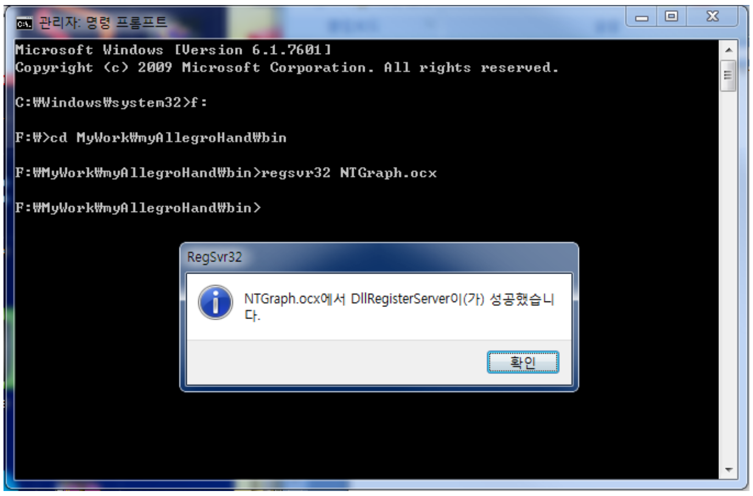
```
int _tmain(int argc, _TCHAR* argv[])
{
		PrintInstruction();

		pSHM = getrPanelManipulatorCmdMemory();

		if (OpenCAN())
		{
				MainLoop();
				CloseCAN();
		}

		closerPanelManipulatorCmdMemory();

		return 0;
}
```

26. "MainLoop" 함수에서 엔코더 값을 "rPanelManipulator" 메모리에 복사한다. "MainLoop" 함수를 다음과 같이 변경한다.

```
void MainLoop()
{
		bool bRun = true;
		int i;

		while (bRun)
		{
				if (!_kbhit())
				{
						Sleep(5);
						if (pSHM)
						{
							for (i=0; i<MAX_DOF; i++)
								pSHM->state.slave_state[i].position = (double)(vars.enc_actual[i]-32768)*(333.3/65536.0)*(3.141592/180.0);
						}
				}
				else
				{
						int c = _getch();
						switch (c)
						{
						case 'q':
								bRun = false;
								break;
						}
				}
		}
}
```

27. 프로그램을 빌드한 후 실행한다.

28. "rPanelManipulator" 실행을 위해서 우선 "NTGRAPH.ocx" 를 시스템에 등록해야 한다. 이를 위해서 "Windows Command Shell"을 반드시 관리자 권한으로 실행한다. "NTGRAPH.ocx"는 bin 폴더 아래에 있다. 해당 폴더로 이동 후 다음과 같이 "regsvr32 NTGraph.ocx" 명령을 실행하여 등록한다.

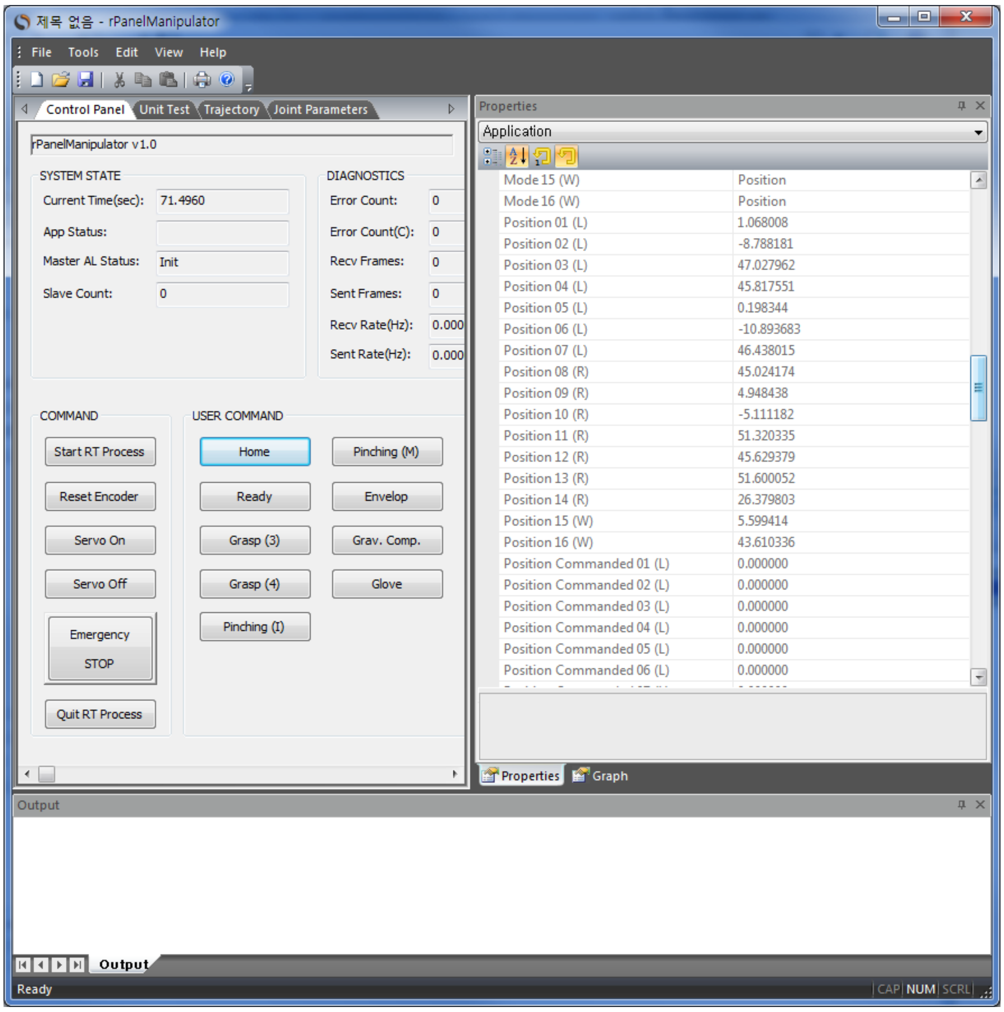


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:₩Windows₩system32>f:

F:₩>cd MyWork₩myAllegroHand₩bin

F:₩MyWork₩myAllegroHand₩bin>regsvr32 NTGraph.ocx

F:₩MyWork₩myAllegroHand₩bin>
```

RegSvr32

NTGraph.ocx에서 DllRegisterServer이(가) 성공했습니다.

29. “rPanelManipulator”을 실행한다. “rPanelManipulator”의 “Properties” 창에서 다음과 같이 엔코더 값을 확인한다. “Properties” 창의 refresh 버튼을 누르면 값이 갱신된다.



rPanelManipulator v1.0

SYSTEM STATE

Current Time(sec):	71.4960
App Status:	
Master AL Status:	Init
Slave Count:	0

DIAGNOSTICS

Error Count:	0
Error Count(C):	0
Recv Frames:	0
Sent Frames:	0
Recv Rate(Hz):	0.000
Sent Rate(Hz):	0.000

Properties

Application

Mode 15 (W)	Position
Mode 16 (W)	Position
Position 01 (L)	1.068008
Position 02 (L)	-8.788181
Position 03 (L)	47.027962
Position 04 (L)	45.817551
Position 05 (L)	0.198344
Position 06 (L)	-10.893683
Position 07 (L)	46.438015
Position 08 (R)	45.024174
Position 09 (R)	4.948438
Position 10 (R)	-5.111182
Position 11 (R)	51.320335
Position 12 (R)	45.629379
Position 13 (R)	51.600052
Position 14 (R)	26.379803
Position 15 (W)	5.599414
Position 16 (W)	43.610336
Position Commanded 01 (L)	0.000000
Position Commanded 02 (L)	0.000000
Position Commanded 03 (L)	0.000000
Position Commanded 04 (L)	0.000000
Position Commanded 05 (L)	0.000000
Position Commanded 06 (L)	0.000000

30. ‘q’ 버튼을 눌러 프로그램을 종료한다.

■ Control Allegro Hand using BHand library

31. "BHand" 라이브러리는 Allegro Hand를 위한 다양한 grasping 알고리즘을 제공한다. "BHand" 라이브러리를 이용하여 Allegro Hand를 제어해보자. 먼저 헤더파일을 인크루드한다.

```
#include "BHand/BHand.h"
```

32. 전역변수를 선언한다. "BHand" 인스턴스 포인터인 "pBHand", 관절각 및 토크 명령, 전류 명령을 저장하는 "q", "tau_des", "cur_des", 그리고 엔코더 오프셋 "enc_offset", 엔코더 방향 "enc_dir", 모터 방향 "motor_dir" 등을 선언한다. 엔코더 오프셋, 엔코더 방향, 모터 방향은 제품마다 다르므로 설정시 각 제품에 맞는 값을 설정한다. 이는 Allegro Hand Wiki 에서 얻을 수 있다.

```
BHand* pBHand = NULL;
double q[MAX_DOF];
double tau_des[MAX_DOF];
double cur_des[MAX_DOF];
const int enc_offset[MAX_DOF] = { // SAH020CR020
		-611, -66016, 1161, 1377,
		-342, -66033, -481, 303,
		30, -65620, 446, 387,
		-3942, -626, -65508, -66768
};
const double enc_dir[MAX_DOF] = { // SAH020CR020
		1.0, -1.0, 1.0, 1.0,
		1.0, -1.0, 1.0, 1.0,
		1.0, -1.0, 1.0, 1.0,
		1.0, 1.0, -1.0, -1.0
};
const double motor_dir[MAX_DOF] = { // SAH020CR020
		1.0, 1.0, 1.0, 1.0,
		1.0, -1.0, -1.0, 1.0,
		-1.0, 1.0, 1.0, 1.0,
		1.0, 1.0, 1.0, 1.0
};
```

33. 몇 개의 함수를 추가한다. 제일 먼저 "CreateBHandAlgorithm"는 BHand 객체를 생성한다.

```
bool CreateBHandAlgorithm()
{
		pBHand = bhCreateRightHand();
		if (!pBHand) return false;
		pBHand->SetTimeInterval(delT);
		return true;
}
```

34. "DestroyBHandAlgorithm"는 BHand 객체를 삭제한다.

```
void DestroyBHandAlgorithm()
{
		if (pBHand)
		{
				delete pBHand;
				pBHand = NULL;
		}
}
```

35. "ComputeTorque"는 "BHand" 라이브러리를 이용하여 관절 토크를 계산한다.

```
void ComputeTorque()
{
		if (!pBHand) return;
		pBHand->SetJointPosition(q); // tell BHand library the current joint positions
		pBHand->UpdateControl(0);
		pBHand->GetJointTorque(tau_des);
}
```

36. "ioThreadProc" 함수는 CAN 통신을 처리하는 쓰레드 핸들러이다. Allegro Hand는 3msec마다 모든 관절의 엔코더값을 4개의 CAN frame에 실어 PC로 전송하는데, 4개의 CAN frame을 모두 받으면 관절 토크를 계산하여 Allegro Hand로 전송한다. 따라서 "ioThreadProc" 함수를 다음과 같이 변경한다.

```
static unsigned int __stdcall ioThreadProc(void* inst)
{
		char id_des;
		char id_cmd;
		char id_src;
		int len;
		unsigned char data[8];
		unsigned char data_return = 0;
		int i;

		while (ioThreadRun)
		{
				while (0 == get_message(CAN_Ch, &id_cmd, &id_src, &id_des, &len, data, FALSE))
				{
```

```
				switch (id_cmd)
				{
				case ID_CMD_QUERY_CONTROL_DATA:
					{
						if (id_src >= ID_DEVICE_SUB_01 && id_src <= ID_DEVICE_SUB_04)
						{
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 0] = (int)(data[0] | (data[1] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 1] = (int)(data[2] | (data[3] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 2] = (int)(data[4] | (data[5] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 3] = (int)(data[6] | (data[7] << 8));
							data_return |= (0x01 << (id_src-ID_DEVICE_SUB_01));
							recvNum++;
						}
						if (data_return == (0x01 | 0x02 | 0x04 | 0x08))
						{
							// convert encoder count to joint angle
							for (i=0; i<MAX_DOF; i++)
								q[i] = (double)(vars.enc_actual[i]*enc_dir[i]-32768-
enc_offset[i])*(333.3/65536.0)*(3.141592/180.0);

							// compute joint torque
							ComputeTorque();

							// convert desired torque to desired current and PWM count
							for (i=0; i<MAX_DOF; i++)
							{
								cur_des[i] = tau_des[i] * motor_dir[i];
								if (cur_des[i] > 1.0) cur_des[i] = 1.0;
								else if (cur_des[i] < -1.0) cur_des[i] = -1.0;
							}

							// send torques
							for (int i=0; i<4;i++)
							{
								// the index order for motors is different from that of encoders
								vars.pwm_demand[i*4+3] = (short)(cur_des[i*4+0]*800.0);
								vars.pwm_demand[i*4+2] = (short)(cur_des[i*4+1]*800.0);
								vars.pwm_demand[i*4+1] = (short)(cur_des[i*4+2]*800.0);
								vars.pwm_demand[i*4+0] = (short)(cur_des[i*4+3]*800.0);

								write_current(CAN_Ch, i, &vars.pwm_demand[4*i]);
								for(int k=0; k<100000; k++);
							}
							sendNum++;
							curTime += delT;

							data_return = 0;
						}
					}
					break;
				}
			}
		}

		return 0;
}
```

37. “MainLoop” 함수에 “rPanelManipulator” 버튼 클릭을 처리하기 위한 코드를 추가한다.

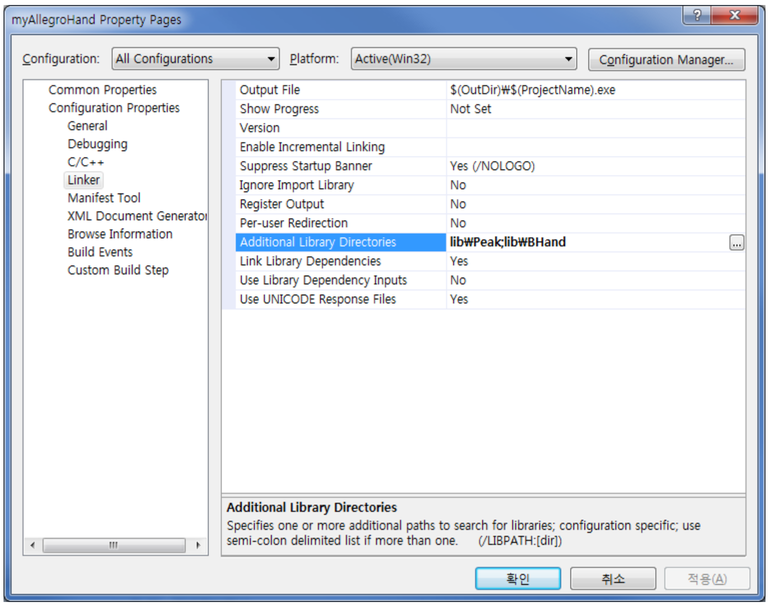
```
if (!_kbhit())
{
		Sleep(5);
		if (pSHM)
		{
			switch (pSHM->cmd.command)
			{
			case CMD_SERVO_ON:
				break;
			case CMD_SERVO_OFF:
				if (pBHand) pBHand->SetMotionType(eMotionType_NONE);
				break;
			case CMD_CMD_1:
				if (pBHand) pBHand->SetMotionType(eMotionType_HOME);
				break;
			case CMD_CMD_2:
				if (pBHand) pBHand->SetMotionType(eMotionType_READY);
				break;
			case CMD_CMD_3:
				if (pBHand) pBHand->SetMotionType(eMotionType_GRASP_3);
				break;
			case CMD_CMD_4:
				if (pBHand) pBHand->SetMotionType(eMotionType_GRASP_4);
				break;
			case CMD_CMD_5:
				if (pBHand) pBHand->SetMotionType(eMotionType_PINCH_IT);
				break;
			case CMD_CMD_6:
				if (pBHand) pBHand->SetMotionType(eMotionType_PINCH_MT);
				break;
			case CMD_CMD_7:
				if (pBHand) pBHand->SetMotionType(eMotionType_ENVELOP);
				break;
			case CMD_CMD_8:
				if (pBHand) pBHand->SetMotionType(eMotionType_GRAVITY_COMP);
				break;
			case CMD_EXIT:
				bRun = false;
				break;
			}
```

```
                        pSHM->cmd.command = CMD_NULL;
                        for (i=0; i<MAX_DOF; i++)
                        {
                                    pSHM->state.slave_state[i].position = q[i];
                                    pSHM->cmd.slave_command[i].torque = tau_des[i];
                        }
                        pSHM->state.time = curTime;
            }
}
```

38. 마지막으로 “main” 함수에서 변수 초기화 및 “BHand” 객체를 생성하고 삭제하기 위한 코드를 다음과 같이 추가한다.

```
int _tmain(int argc, _TCHAR* argv[])
{
            PrintInstruction();

            memset(&vars, 0, sizeof(vars));
            memset(q, 0, sizeof(q));
            memset(tau_des, 0, sizeof(tau_des));
            memset(cur_des, 0, sizeof(cur_des));
            curTime = 0.0;

            pSHM = getrPanelManipulatorCmdMemory();

            if (CreateBHandAlgorithm() && OpenCAN())
                        MainLoop();

            CloseCAN();
            DestroyBHandAlgorithm();
            closerPanelManipulatorCmdMemory();

            return 0;
}
```

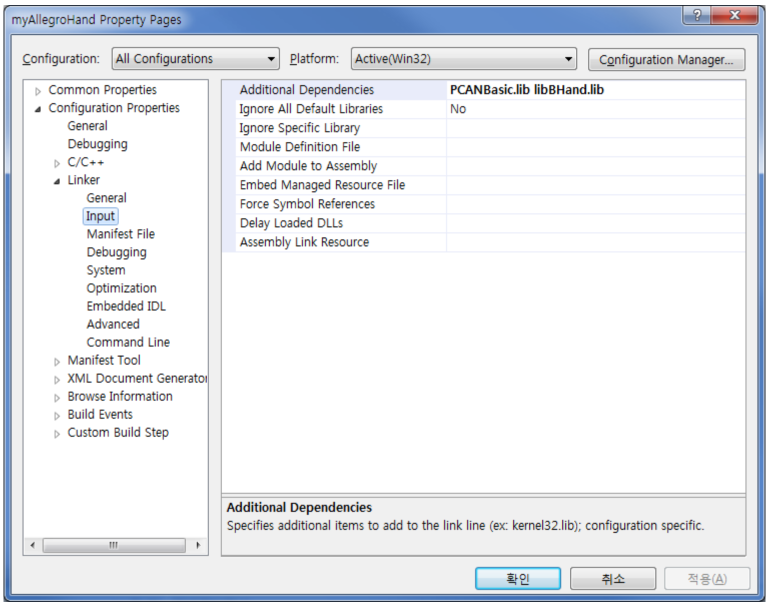
39. 프로젝트 속성에서 “Additional Library Directories”에 “BHand” 라이브러리 폴더를 추가한다.



Output File	$(OutDir)₩$(ProjectName).exe
Show Progress	Not Set
Version	
Enable Incremental Linking	
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	lib₩Peak;lib₩BHand
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Use UNICODE Response Files	Yes

Additional Library Directories
Specifies one or more additional paths to search for libraries; configuration specific; use semi-colon delimited list if more than one.   (/LIBPATH:[dir])

40. “Additional Dependancies”에 “libBHand.lib”을 추가한다.



Additional Dependencies	PCANBasic.lib libBHand.lib
Ignore All Default Libraries	No
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

41. 프로그램 빌드 후 실행한다.
42. "rPanelManipulator"의 "Control Panel" 탭에서 "Current Time(sec)"이 실제 시간과 같이 증가함을 우선 확인한다.
43. "rPanelManipulator"에서 "Home", "Ready", "Grasp (3)" 등의 버튼을 클릭하여 동작을 확인한다.

■ Source code

myAllegroHand.cpp

```
// myAllegroHand.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "windows.h"
#include <conio.h>
#include <process.h>
#include <tchar.h>
#include "canAPI.h"
#include "rDeviceAllegroHandCANDef.h"
#include "rPanelManipulatorCmdUtil.h"
#include "BHand/BHand.h"

/////////////////////////////////////
// for CAN communication
const double delT = 0.003;
int CAN_Ch = 0;
bool ioThreadRun = false;
uintptr_t ioThread = 0;
int recvNum = 0;
int sendNum = 0;
double statTime = -1.0;
AllegroHand_DeviceMemory_t vars;

/////////////////////////////////////
// for rPanelManipulator
rPanelManipulatorData_t* pSHM = NULL;
double curTime = 0.0;

/////////////////////////////////////
// for BHand library
BHand* pBHand = NULL;
double q[MAX_DOF];
double tau_des[MAX_DOF];
double cur_des[MAX_DOF];
const int enc_offset[MAX_DOF] = { // SAH020CR020
			-611, -66016, 1161, 1377,
			-342, -66033, -481, 303,
			30, -65620, 446, 387,
			-3942, -626, -65508, -66768
};
const double enc_dir[MAX_DOF] = { // SAH020CR020
			1.0, -1.0, 1.0, 1.0,
			1.0, -1.0, 1.0, 1.0,
			1.0, -1.0, 1.0, 1.0,
			1.0, 1.0, -1.0, -1.0
};
const double motor_dir[MAX_DOF] = { // SAH020CR020
			1.0, 1.0, 1.0, 1.0,
			1.0, -1.0, -1.0, 1.0,
			-1.0, 1.0, 1.0, 1.0,
			1.0, 1.0, 1.0, 1.0
};

/////////////////////////////////////
// functions
void PrintInstruction();
void MainLoop();
bool OpenCAN();
void CloseCAN();
int GetCANChannelIndex(const TCHAR* cname);
bool CreateBHandAlgorithm();
void DestroyBHandAlgorithm();
void ComputeTorque();

void ComputeTorque()
{
			/*memset(tau_des, 0, sizeof(tau_des));
			tau_des[0] = 200.0 / 800.0 * motor_dir[0];
			return;*/


			if (!pBHand) return;

			pBHand->SetJointPosition(q); // tell BHand library the current joint positions
			pBHand->UpdateControl(0);
			pBHand->GetJointTorque(tau_des);
}

static unsigned int __stdcall ioThreadProc(void* inst)
{
			char id_des;
			char id_cmd;
			char id_src;
			int len;
			unsigned char data[8];
			unsigned char data_return = 0;
			int i;

			while (ioThreadRun)
```

```
		{
			while (0 == get_message(CAN_Ch, &id_cmd, &id_src, &id_des, &len, data, FALSE))
			{
				switch (id_cmd)
				{
				case ID_CMD_QUERY_CONTROL_DATA:
					{
						if (id_src >= ID_DEVICE_SUB_01 && id_src <= ID_DEVICE_SUB_04)
						{
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 0] = (int)(data[0] | (data[1] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 1] = (int)(data[2] | (data[3] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 2] = (int)(data[4] | (data[5] << 8));
							vars.enc_actual[(id_src-ID_DEVICE_SUB_01)*4 + 3] = (int)(data[6] | (data[7] << 8));
							data_return |= (0x01 << (id_src-ID_DEVICE_SUB_01));
							recvNum++;
						}
						if (data_return == (0x01 | 0x02 | 0x04 | 0x08))
						{
							// convert encoder count to joint angle
							for (i=0; i<MAX_DOF; i++)
								q[i] = (double)(vars.enc_actual[i]*enc_dir[i]-32768-
enc_offset[i])*(333.3/65536.0)*(3.141592/180.0);

							// compute joint torque
							ComputeTorque();

							// convert desired torque to desired current and PWM count
							for (i=0; i<MAX_DOF; i++)
							{
								cur_des[i] = tau_des[i] * motor_dir[i];
								if (cur_des[i] > 1.0) cur_des[i] = 1.0;
								else if (cur_des[i] < -1.0) cur_des[i] = -1.0;
							}

							// send torques
							for (int i=0; i<4;i++)
							{
								// the index order for motors is different from that of encoders
								vars.pwm_demand[i*4+3] = (short)(cur_des[i*4+0]*800.0);
								vars.pwm_demand[i*4+2] = (short)(cur_des[i*4+1]*800.0);
								vars.pwm_demand[i*4+1] = (short)(cur_des[i*4+2]*800.0);
								vars.pwm_demand[i*4+0] = (short)(cur_des[i*4+3]*800.0);

								write_current(CAN_Ch, i, &vars.pwm_demand[4*i]);
								for(int k=0; k<100000; k++);
							}
							sendNum++;
							curTime += delT;

							data_return = 0;
						}
					}
					break;
				}
			}
		}

		return 0;
}

void MainLoop()
{
		bool bRun = true;
		int i;

		while (bRun)
		{
			if (!_kbhit())
			{
				Sleep(5);
				if (pSHM)
				{
					switch (pSHM->cmd.command)
					{
					case CMD_SERVO_ON:
						break;
					case CMD_SERVO_OFF:
						if (pBHand) pBHand->SetMotionType(eMotionType_NONE);
						break;
					case CMD_CMD_1:
						if (pBHand) pBHand->SetMotionType(eMotionType_HOME);
						break;
					case CMD_CMD_2:
						if (pBHand) pBHand->SetMotionType(eMotionType_READY);
						break;
					case CMD_CMD_3:
						if (pBHand) pBHand->SetMotionType(eMotionType_GRASP_3);
						break;
					case CMD_CMD_4:
						if (pBHand) pBHand->SetMotionType(eMotionType_GRASP_4);
						break;
					case CMD_CMD_5:
						if (pBHand) pBHand->SetMotionType(eMotionType_PINCH_IT);
						break;
					case CMD_CMD_6:
						if (pBHand) pBHand->SetMotionType(eMotionType_PINCH_MT);
						break;
					case CMD_CMD_7:
						if (pBHand) pBHand->SetMotionType(eMotionType_ENVELOP);
```

```
                                                break;
                                        case CMD_CMD_8:
                                                if (pBHand) pBHand->SetMotionType(eMotionType_GRAVITY_COMP);
                                                break;
                                        case CMD_EXIT:
                                                bRun = false;
                                                break;
                                        }
                                        pSHM->cmd.command = CMD_NULL;
                                        for (i=0; i<MAX_DOF; i++)
                                        {
                                                pSHM->state.slave_state[i].position = q[i];
                                                pSHM->cmd.slave_command[i].torque = tau_des[i];
                                        }
                                        pSHM->state.time = curTime;
                                }
                        }
                        else
                        {
                                int c = _getch();
                                switch (c)
                                {
                                case 'q':
                                        bRun = false;
                                        break;
                                }
                        }
        }
}

bool OpenCAN()
{
        int ret;

        CAN_Ch = GetCANChannelIndex(_T("USBBUS1"));

        printf(">CAN(%d): open\n", CAN_Ch);
        ret = command_can_open(CAN_Ch);
        if(ret < 0)
        {
                printf("ERROR command_canopen !!! \n");
                return false;
        }

        recvNum = 0;
        sendNum = 0;
        statTime = 0.0;

        ioThreadRun = true;
        ioThread = _beginthreadex(NULL, 0, ioThreadProc, NULL, 0, NULL);
        printf(">CAN: starts listening CAN frames\n");

        printf(">CAN: system init\n");
        ret = command_can_sys_init(CAN_Ch, 3/*msec*/);
        if(ret < 0)
        {
                printf("ERROR command_can_sys_init !!! \n");
                command_can_close(CAN_Ch);
                return false;
        }

        printf(">CAN: start periodic communication\n");
        ret = command_can_start(CAN_Ch);
        if(ret < 0)
        {
                printf("ERROR command_can_start !!! \n");
                command_can_stop(CAN_Ch);
                command_can_close(CAN_Ch);
                return false;
        }

        return true;
}

void CloseCAN()
{
        int ret;

        printf(">CAN: stop periodic communication\n");
        ret = command_can_stop(CAN_Ch);
        if(ret < 0)
        {
                printf("ERROR command_can_stop !!! \n");
        }

        if (ioThreadRun)
        {
                printf(">CAN: stoped listening CAN frames\n");
                ioThreadRun = false;
                WaitForSingleObject((HANDLE)ioThread, INFINITE);
                CloseHandle((HANDLE)ioThread);
                ioThread = 0;
        }

        printf(">CAN(%d): close\n", CAN_Ch);
        ret = command_can_close(CAN_Ch);
        if(ret < 0) printf("ERROR command_can_close !!! \n");
}
```

```
void PrintInstruction()
{
	printf("--------------------------------------------------\n");
	printf("myAllegroHand\n");
	printf("\n");
	printf("To quit this program, press 'q'.\n");
	printf("--------------------------------------------------\n");
}

int GetCANChannelIndex(const TCHAR* cname)
{
	if (!cname) return 0;

	if (!_tcsicmp(cname, _T("0")) || !_tcsicmp(cname, _T("PCAN_NONEBUS")) || !_tcsicmp(cname, _T("NONEBUS")))
		return 0;
	else if (!_tcsicmp(cname, _T("1")) || !_tcsicmp(cname, _T("PCAN_ISABUS1")) || !_tcsicmp(cname, _T("ISABUS1")))
		return 1;
	else if (!_tcsicmp(cname, _T("2")) || !_tcsicmp(cname, _T("PCAN_ISABUS2")) || !_tcsicmp(cname, _T("ISABUS2")))
		return 2;
	else if (!_tcsicmp(cname, _T("3")) || !_tcsicmp(cname, _T("PCAN_ISABUS3")) || !_tcsicmp(cname, _T("ISABUS3")))
		return 3;
	else if (!_tcsicmp(cname, _T("4")) || !_tcsicmp(cname, _T("PCAN_ISABUS4")) || !_tcsicmp(cname, _T("ISABUS4")))
		return 4;
	else if (!_tcsicmp(cname, _T("5")) || !_tcsicmp(cname, _T("PCAN_ISABUS5")) || !_tcsicmp(cname, _T("ISABUS5")))
		return 5;
	else if (!_tcsicmp(cname, _T("7")) || !_tcsicmp(cname, _T("PCAN_ISABUS6")) || !_tcsicmp(cname, _T("ISABUS6")))
		return 6;
	else if (!_tcsicmp(cname, _T("8")) || !_tcsicmp(cname, _T("PCAN_ISABUS7")) || !_tcsicmp(cname, _T("ISABUS7")))
		return 7;
	else if (!_tcsicmp(cname, _T("8")) || !_tcsicmp(cname, _T("PCAN_ISABUS8")) || !_tcsicmp(cname, _T("ISABUS8")))
		return 8;
	else if (!_tcsicmp(cname, _T("9")) || !_tcsicmp(cname, _T("PCAN_DNGBUS1")) || !_tcsicmp(cname, _T("DNGBUS1")))
		return 9;
	else if (!_tcsicmp(cname, _T("10")) || !_tcsicmp(cname, _T("PCAN_PCIBUS1")) || !_tcsicmp(cname, _T("PCIBUS1")))
		return 10;
	else if (!_tcsicmp(cname, _T("11")) || !_tcsicmp(cname, _T("PCAN_PCIBUS2")) || !_tcsicmp(cname, _T("PCIBUS2")))
		return 11;
	else if (!_tcsicmp(cname, _T("12")) || !_tcsicmp(cname, _T("PCAN_PCIBUS3")) || !_tcsicmp(cname, _T("PCIBUS3")))
		return 12;
	else if (!_tcsicmp(cname, _T("13")) || !_tcsicmp(cname, _T("PCAN_PCIBUS4")) || !_tcsicmp(cname, _T("PCIBUS4")))
		return 13;
	else if (!_tcsicmp(cname, _T("14")) || !_tcsicmp(cname, _T("PCAN_PCIBUS5")) || !_tcsicmp(cname, _T("PCIBUS5")))
		return 14;
	else if (!_tcsicmp(cname, _T("15")) || !_tcsicmp(cname, _T("PCAN_PCIBUS6")) || !_tcsicmp(cname, _T("PCIBUS6")))
		return 15;
	else if (!_tcsicmp(cname, _T("16")) || !_tcsicmp(cname, _T("PCAN_PCIBUS7")) || !_tcsicmp(cname, _T("PCIBUS7")))
		return 16;
	else if (!_tcsicmp(cname, _T("17")) || !_tcsicmp(cname, _T("PCAN_PCIBUS8")) || !_tcsicmp(cname, _T("PCIBUS8")))
		return 17;
	else if (!_tcsicmp(cname, _T("18")) || !_tcsicmp(cname, _T("PCAN_USBBUS1")) || !_tcsicmp(cname, _T("USBBUS1")))
		return 18;
	else if (!_tcsicmp(cname, _T("19")) || !_tcsicmp(cname, _T("PCAN_USBBUS2")) || !_tcsicmp(cname, _T("USBBUS2")))
		return 19;
	else if (!_tcsicmp(cname, _T("20")) || !_tcsicmp(cname, _T("PCAN_USBBUS3")) || !_tcsicmp(cname, _T("USBBUS3")))
		return 20;
	else if (!_tcsicmp(cname, _T("21")) || !_tcsicmp(cname, _T("PCAN_USBBUS4")) || !_tcsicmp(cname, _T("USBBUS4")))
		return 21;
	else if (!_tcsicmp(cname, _T("22")) || !_tcsicmp(cname, _T("PCAN_USBBUS5")) || !_tcsicmp(cname, _T("USBBUS5")))
		return 22;
	else if (!_tcsicmp(cname, _T("23")) || !_tcsicmp(cname, _T("PCAN_USBBUS6")) || !_tcsicmp(cname, _T("USBBUS6")))
		return 23;
	else if (!_tcsicmp(cname, _T("24")) || !_tcsicmp(cname, _T("PCAN_USBBUS7")) || !_tcsicmp(cname, _T("USBBUS7")))
		return 24;
	else if (!_tcsicmp(cname, _T("25")) || !_tcsicmp(cname, _T("PCAN_USBBUS8")) || !_tcsicmp(cname, _T("USBBUS8")))
		return 25;
	else if (!_tcsicmp(cname, _T("26")) || !_tcsicmp(cname, _T("PCAN_PCCBUS1")) || !_tcsicmp(cname, _T("PCCBUS1")))
		return 26;
	else if (!_tcsicmp(cname, _T("27")) || !_tcsicmp(cname, _T("PCAN_PCCBUS2")) || !_tcsicmp(cname, _T("PCCBUS2")))
		return 271;
	else
		return 0;
}

bool CreateBHandAlgorithm()
{
	pBHand = bhCreateRightHand();
	if (!pBHand) return false;
	pBHand->SetTimeInterval(delT);
	return true;
}

void DestroyBHandAlgorithm()
{
	if (pBHand)
	{
		delete pBHand;
		pBHand = NULL;
	}
}

int _tmain(int argc, _TCHAR* argv[])
{
	PrintInstruction();

	memset(&vars, 0, sizeof(vars));
	memset(q, 0, sizeof(q));
	memset(tau_des, 0, sizeof(tau_des));
	memset(cur_des, 0, sizeof(cur_des));
	curTime = 0.0;
```

```
			pSHM = getrPanelManipulatorCmdMemory();

			if (CreateBHandAlgorithm() && OpenCAN())
					MainLoop();

			CloseCAN();
			DestroyBHandAlgorithm();
			closerPanelManipulatorCmdMemory();

			return 0;
}
```

canAPI.h

```
/*
  *\brief API for communication over CAN bus
  *\detailed The API for communicating with the various motor controllers
  *              over the CAN bus interface on the robot hand
  *
  *$Author: Sangyup Yi $
  *$Date: 2012/5/11 23:34:00 $
  *$Revision: 1.0 $
  */

#ifndef _CANDAPI_H
#define _CANDAPI_H

#include "canDef.h"

CANAPI_BEGIN

#ifndef FALSE
#define FALSE (0)
#endif
#ifndef TRUE
#define TRUE (1)
#endif

/*=====================*/
/*          Defines          */
/*=====================*/
//constants
#define TX_QUEUE_SIZE           (32)
#define RX_QUEUE_SIZE           (32)
#define TX_TIMEOUT              (5)
#define RX_TIMEOUT              (5)
#define mbxID                   (0)
#define BASE_ID                 (0)
#define MAX_BUS                 (256)

/******************/
/* CAN device API */
/******************/
int command_can_open(int ch);
int command_can_reset(int ch);
int command_can_close(int ch);

int command_can_sys_init(int ch, int period_msec);
int command_can_start(int ch);
int command_can_stop(int ch);

int write_current(int ch, int findex, short* pwm);

int get_message(int ch, char* cmd, char* src, char* des, int* len, unsigned char* data, int blocking);

CANAPI_END

#endif
```

canDef.h

```
/*
  *\brief Constants and enums for CAN communication
  *
  *$Author: Sangyup Yi $
  *$Date: 2012/5/11 23:34:00 $
  *$Revision: 1.0 $
  */

#ifndef _CANDEF_H
#define _CANDEF_H


#ifdef USING_NAMESPACE_CANAPI
#     define CANAPI_BEGIN namespace CANAPI {
#     define CANAPI_END };
#else
#     define CANAPI_BEGIN
#     define CANAPI_END
#endif



CANAPI_BEGIN


typedef struct{
                    unsigned char STD_EXT;
                    unsigned long msg_id;            // message identifier
                    unsigned char data_length;       //
                    char                                          data[8];          // data array
} can_msg;

#define                       STD                           (bool)0
#define                       EXT                           (bool)1

////////////////////////////////////////////////
//    Define Control Board ID number
#define ID_COMMON                                           0x01
#define ID_DEVICE_MAIN                              0x02
#define ID_DEVICE_SUB_01 0x03
#define ID_DEVICE_SUB_02 0x04
#define ID_DEVICE_SUB_03 0x05
#define ID_DEVICE_SUB_04 0x06
#define ID_DEVICE_SUB_05 0x07
#define ID_DEVICE_SUB_06 0x08
#define ID_DEVICE_SUB_07 0x09
#define ID_DEVICE_SUB_08 0x0a
#define ID_DEVICE_SUB_09 0x0b
#define ID_DEVICE_SUB_10 0x0c

////////////////////////////////////////////////
//    Define CAN Command
#define ID_CMD_SET_SYSTEM_ON                                          0x01
#define ID_CMD_SET_SYSTEM_OFF                                         0x02
#define ID_CMD_SET_PERIOD                                                     0x03
#define ID_CMD_SET_MODE_JOINT                                         0x04
#define ID_CMD_SET_MODE_TASK                                          0x05
#define ID_CMD_SET_TORQUE_1                                                   0x06
#define ID_CMD_SET_TORQUE_2                                                   0x07
#define ID_CMD_SET_TORQUE_3                                                   0x08
#define ID_CMD_SET_TORQUE_4                                                   0x09
#define ID_CMD_SET_POSITION_1                                         0x0a
#define ID_CMD_SET_POSITION_2                                         0x0b
#define ID_CMD_SET_POSITION_3                                         0x0c
#define ID_CMD_SET_POSITION_4                                         0x0d
#define ID_CMD_QUERY_STATE_DATA                                       0x0e
#define ID_CMD_QUERY_CONTROL_DATA                                     0x0f

////////////////////////////////////////////////
//    Define Control Channel in Device
#define MOTOR_CH_1                                          0x01
#define MOTOR_CH_2                                          0x02
#define MOTOR_CH_3                                          0x03
#define MOTOR_CH_4                                          0x04
#define MOTOR_CH_ALL                                0x05
#define DEVICE_SYSTEM                               0x06

////////////////////////////////////////////////
// Define State
#define STATE_CH1                                           0x01
#define STATE_CH2                                           0x02
#define STATE_CH3                                           0x04
#define STATE_CH4                                           0x08
#define STATE_ALL                                           0x10
#define STATE_SYSTEM                                0x80

////////////////////////////////////////////////
//      Define Motor State

#define SYSTEM_ON                                   0x01
#define SYSTEM_OFF                                  0x02
#define MOTOR_RUN                                   0x04
#define MOTOR_STOP                                  0x08

#define MOTOR_DIR_CW      0x01
#define MOTOR_DIR_CCW     0x02

#define ON      1
#define OFF 0
////////////////////////////////////////////////



CANAPI_END

#endif
```

rDeviceAllegroHandCANDef.h

```
#ifndef __RDEVICEALLEGROHANDCANDEF_H__
#define __RDEVICEALLEGROHANDCANDEF_H__
```

```
#define MAX_DOF 16

typedef struct tagDeviceMemory_AllegroHand
{
			int enc_actual[MAX_DOF];
			short pwm_actual[MAX_DOF];
			short pwm_demand[MAX_DOF];
} AllegroHand_DeviceMemory_t;

#endif // __RDEVICEALLEGROHANDCANDEF_H__
```

canAPI.cpp

```
/*======================*/
/*		Includes		*/
/*======================*/
//system headers
#include <stdio.h>
#include <errno.h>
#ifndef _WIN32
#include <inttypes.h>
#include <pthread.h>
#include <syslog.h>
#include <unistd.h>
#else
#include <windows.h>
#endif
#include <malloc.h>
#include <assert.h>
//project headers
extern "C" {
#include "Peak/PCANBasic.h"
}
#include "canDef.h"
#include "canAPI.h"


CANAPI_BEGIN

/*=====================*/
/*		Defines		*/
/*=====================*/
//macros
#define isAlpha(c) ( ((c >= 'A') && (c <= 'Z')) ? 1 : 0 )
#define isSpace(c) ( (c == ' ') ? 1 : 0 )
#define isDigit(c) ( ((c >= '0') && (c <= '9')) ? 1 : 0 )
#define ADDR2NODE(x) ((((x) >> 5) & 0x001F) - BASE_ID)
#define NODE2ADDR(x) (((mbxID + BASE_ID) << 5) | ((x) + BASE_ID))
#define GROUPID(n)	(((mbxID + BASE_ID) << 5) | (0x0400 + (n)))
#define BROADCAST	(GROUPID(0))
#define Border(Value,Min,Max)	(Value<Min)?Min:((Value>Max)?Max:Value)
//typedefs & structs
typedef unsigned long DWORD;

/*========================================*/
/*		Global file-scope variables		*/
/*========================================*/

TPCANHandle canDev[MAX_BUS] = {
			PCAN_NONEBUS, // Undefined/default value for a PCAN bus

			PCAN_ISABUS1, // PCAN-ISA interface, channel 1
			PCAN_ISABUS2, // PCAN-ISA interface, channel 2
			PCAN_ISABUS3, // PCAN-ISA interface, channel 3
			PCAN_ISABUS4, // PCAN-ISA interface, channel 4
			PCAN_ISABUS5, // PCAN-ISA interface, channel 5
			PCAN_ISABUS6, // PCAN-ISA interface, channel 6
			PCAN_ISABUS7, // PCAN-ISA interface, channel 7
			PCAN_ISABUS8, // PCAN-ISA interface, channel 8

			PCAN_DNGBUS1, // PCAN-Dongle/LPT interface, channel 1

			PCAN_PCIBUS1, // PCAN-PCI interface, channel 1
			PCAN_PCIBUS2, // PCAN-PCI interface, channel 2
			PCAN_PCIBUS3, // PCAN-PCI interface, channel 3
			PCAN_PCIBUS4, // PCAN-PCI interface, channel 4
			PCAN_PCIBUS5, // PCAN-PCI interface, channel 5
			PCAN_PCIBUS6, // PCAN-PCI interface, channel 6
			PCAN_PCIBUS7, // PCAN-PCI interface, channel 7
			PCAN_PCIBUS8, // PCAN-PCI interface, channel 8

			PCAN_USBBUS1, // PCAN-USB interface, channel 1
			PCAN_USBBUS2, // PCAN-USB interface, channel 2
			PCAN_USBBUS3, // PCAN-USB interface, channel 3
			PCAN_USBBUS4, // PCAN-USB interface, channel 4
			PCAN_USBBUS5, // PCAN-USB interface, channel 5
			PCAN_USBBUS6, // PCAN-USB interface, channel 6
			PCAN_USBBUS7, // PCAN-USB interface, channel 7
			PCAN_USBBUS8, // PCAN-USB interface, channel 8

			PCAN_PCCBUS1, // PCAN-PC Card interface, channel 1
			PCAN_PCCBUS2, // PCAN-PC Card interface, channel 2
};
```

```c
/*==========================================*/
/*          Private functions prototypes          */
/*==========================================*/
int canReadMsg(int bus, int *id, int *len, unsigned char *data, int blocking);
int canSendMsg(int bus, int id, char len, unsigned char *data, int blocking);

/*========================================*/
/*          Public functions (CAN API)          */
/*========================================*/
int initCAN(int bus){
		TPCANStatus Status = PCAN_ERROR_OK;
		char strMsg[256];
		TPCANBaudrate Baudrate = PCAN_BAUD_1M;
		TPCANType HwType = 0;
		DWORD IOPort = 0;
		WORD Interrupt = 0;

		Status = CAN_Initialize(canDev[bus], Baudrate, HwType, IOPort, Interrupt);
		if (Status != PCAN_ERROR_OK)
		{
				CAN_GetErrorText(Status, 0, strMsg);
				printf("initCAN(): CAN_Initialize() failed with error %ld\n", Status);
				printf("%s\n", strMsg);
				return Status;
		}

		Status = CAN_FilterMessages(
				canDev[bus],
				((unsigned long)(ID_CMD_QUERY_CONTROL_DATA) <<6) | ((unsigned long)ID_DEVICE_MAIN <<3) | ((unsigned long)ID_DEVICE_SUB_01),
				((unsigned long)(ID_CMD_QUERY_CONTROL_DATA) <<6) | ((unsigned long)ID_DEVICE_MAIN <<3) | ((unsigned long)ID_DEVICE_SUB_04),
				PCAN_MESSAGE_STANDARD);
		if (Status != PCAN_ERROR_OK)
		{
				CAN_GetErrorText(Status, 0, strMsg);
				printf("initCAN(): CAN_FilterMessages() failed with error %ld\n", Status);
				printf("%s\n", strMsg);
				//return Status;
		}

		Status = CAN_Reset(canDev[bus]);
		if (Status != PCAN_ERROR_OK)
		{
				CAN_GetErrorText(Status, 0, strMsg);
				printf("initCAN(): CAN_Reset() failed with error %ld\n", Status);
				printf("%s\n", strMsg);
				//return Status;
		}

		return 0; // PCAN_ERROR_OK
}

int freeCAN(int bus){
		TPCANStatus Status = PCAN_ERROR_OK;
		char strMsg[256];

		Status = CAN_Uninitialize(canDev[bus]);
		if (Status != PCAN_ERROR_OK)
		{
				CAN_GetErrorText(Status, 0, strMsg);
				printf("freeCAN(): CAN_Uninitialize() failed with error %ld\n", Status);
				printf("%s\n", strMsg);
				return Status;
		}

		return 0; // PCAN_ERROR_OK
}

int canReadMsg(int bus, int *id, int *len, unsigned char *data, int blocking){
		TPCANMsg CANMsg;
		TPCANTimestamp CANTimeStamp;
		TPCANStatus Status = PCAN_ERROR_OK;
		char strMsg[256];
		int i;

		// We execute the "Read" function of the PCANBasic
		//
		Status = CAN_Read(canDev[bus], &CANMsg, &CANTimeStamp);

		if (Status != PCAN_ERROR_OK)
		{
				if (Status != PCAN_ERROR_QRCVEMPTY)
				{
						CAN_GetErrorText(Status, 0, strMsg);
						printf("canReadMsg(): CAN_Read() failed with error %ld\n", Status);
						printf("%s\n", strMsg);
				}
				return Status;
		}

		*id = CANMsg.ID;
		*len = CANMsg.LEN;
		for(i = 0; i < CANMsg.LEN; i++)
				data[i] = CANMsg.DATA[i];

		return 0;
}

int canSendMsg(int bus, int id, char len, unsigned char *data, int blocking){
		TPCANMsg CANMsg;
```

```
			TPCANStatus Status = PCAN_ERROR_OK;
			char strMsg[256];
			int i;

			CANMsg.ID = id;
			CANMsg.LEN = len & 0x0F;
			for(i = 0; i < len; i++)
		CANMsg.DATA[i] = data[i];
			CANMsg.MSGTYPE = PCAN_MESSAGE_STANDARD;

			Status = CAN_Write(canDev[bus], &CANMsg);
			if (Status != PCAN_ERROR_OK)
			{
					CAN_GetErrorText(Status, 0, strMsg);
					printf("canSendMsg(): CAN_Write() failed with error %ld\n", Status);
					printf("%s\n", strMsg);
					return Status;
			}

			return 0;
}

/*========================================*/
/*		CAN API						*/
/*========================================*/
int command_can_open(int ch)
{
			assert(ch >= 0 && ch < MAX_BUS);

			DWORD ret;

			printf("<< CAN: Open Channel...\n");
			ret = initCAN(ch);
			if (ret != 0) return ret;
			printf("\t- Ch.%2d (OK)\n", ch);
			printf("\t- Done\n");

			return 0;
}

int command_can_reset(int ch)
{
			return -1;
}

int command_can_close(int ch)
{
			assert(ch >= 0 && ch < MAX_BUS);

			TPCANStatus Status = PCAN_ERROR_OK;
			char strMsg[256];
			printf("<< CAN: Close...\n");

			Status = CAN_Uninitialize(canDev[ch]);
			if (Status != PCAN_ERROR_OK)
			{
					CAN_GetErrorText(Status, 0, strMsg);
					printf("freeCAN(): CAN_Uninitialize() failed with error %ld\n", Status);
					printf("%s\n", strMsg);
					return Status;
			}

			printf("\t- Done\n");
			return 0; // PCAN_ERROR_OK
}

int command_can_sys_init(int ch, int period_msec)
{
			assert(ch >= 0 && ch < MAX_BUS);

			long Txid;
			unsigned char data[8];
			int ret;

			Txid = ((unsigned long)ID_CMD_SET_PERIOD<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
			data[0] = (unsigned char)period_msec;
			ret = canSendMsg(ch, Txid, 1, data, TRUE);

			Sleep(10);

			Txid = ((unsigned long)ID_CMD_SET_MODE_TASK<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
			ret = canSendMsg(ch, Txid, 0, data, TRUE);

			Sleep(10);

			Txid = ((unsigned long)ID_CMD_QUERY_STATE_DATA<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
			ret = canSendMsg(ch, Txid, 0, data, TRUE);

			return 0;
}

int command_can_start(int ch)
{
			assert(ch >= 0 && ch < MAX_BUS);

			long Txid;
			unsigned char data[8];
			int ret;
		Txid = ((unsigned long)ID_CMD_QUERY_STATE_DATA<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
		ret = canSendMsg(ch, Txid, 0, data, TRUE);

		Sleep(10);

		Txid = ((unsigned long)ID_CMD_SET_SYSTEM_ON<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
		ret = canSendMsg(ch, Txid, 0, data, TRUE);

		return 0;
}

int command_can_stop(int ch)
{
		assert(ch >= 0 && ch < MAX_BUS);

		long Txid;
		unsigned char data[8];
		int ret;

		Txid = ((unsigned long)ID_CMD_SET_SYSTEM_OFF<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
		ret = canSendMsg(ch, Txid, 0, data, TRUE);

		return 0;
}

int write_current(int ch, int findex, short* pwm)
{
		assert(ch >= 0 && ch < MAX_BUS);

		long Txid;
		unsigned char data[8];
		int ret;

		if (findex >= 0 && findex < 4)
		{
				data[0] = (unsigned char)( (pwm[0] >> 8) & 0x00ff);
				data[1] = (unsigned char)(pwm[0] & 0x00ff);

				data[2] = (unsigned char)( (pwm[1] >> 8) & 0x00ff);
				data[3] = (unsigned char)(pwm[1] & 0x00ff);

				data[4] = (unsigned char)( (pwm[2] >> 8) & 0x00ff);
				data[5] = (unsigned char)(pwm[2] & 0x00ff);

				data[6] = (unsigned char)( (pwm[3] >> 8) & 0x00ff);
				data[7] = (unsigned char)(pwm[3] & 0x00ff);

				Txid = ((unsigned long)(ID_CMD_SET_TORQUE_1 + findex)<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
				ret = canSendMsg(ch, Txid, 8, data, TRUE);
		}
		else
				return -1;

		return 0;
}

int get_message(int ch, char* cmd, char* src, char* des, int* len, unsigned char* data, int blocking)
{
		int err;
		unsigned long Rxid;

		err = canReadMsg(ch, (int*)&Rxid, len, data, blocking);
		if (!err)
		{
				*cmd = (char)( (Rxid >> 6) & 0x1f );
				*des = (char)( (Rxid >> 3) & 0x07 );
				*src = (char)( Rxid & 0x07);
		}
		else
		{
				return err;
		}
		return 0;
}



CANAPI_END
```