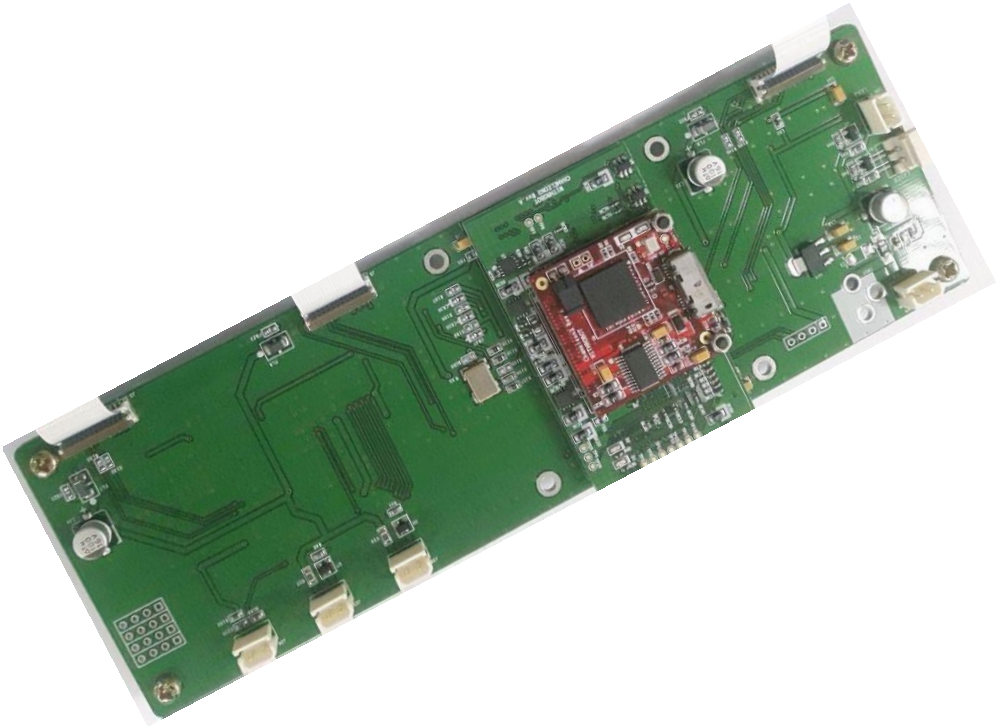
OjOcam-3Ch Board



10/19/2016

OjOcam-3Ch Board 3 채널 영상을 전송하는 보드

OjOcam-3Ch Board 는 3 채널의 CMOS 카메라 센서의 영상을 USB 포트를 통하여 PC 로 전송하는 보드로, 센서 영상을 확인할 수 있는 보드입니다.

OjOcam-3Ch Board

제품 구성

제품 구성

제품의 구성은 아래와 같습니다.

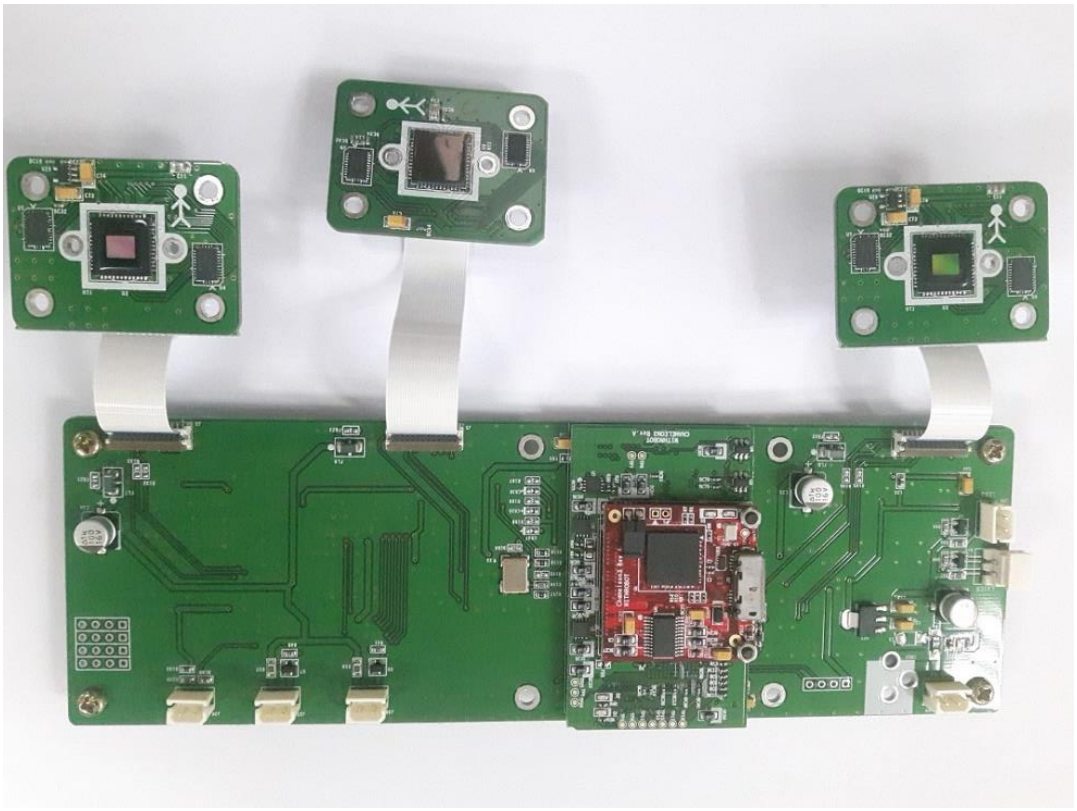


그림 1. OjOcam-3Ch Board

- OjOcam-3Ch Board 구성
  - ✓ Base Board – 1ea
  - ✓ CMOS Sensor Board – 3ea
  - ✓ Chameleon Board – 1ea

제품 소개

들어가며

OjOcam-3Ch Board 는 3 채널의 카메라 영상을 PC 에서 확인할 수 있도록 하는 보드입니다. 센서 보드는 보드 형태로 제공되기 때문에 사용자가 다양한 렌즈를 장착하여 실험해 볼 수 있으며 필요에 따라서는 센서 보드를 자체적으로 제작하여 테스트에 이용할 수도 있습니다. 또한 제품 자체가 보드 형태로 제공되기 때문에 다른 제품의 부속 제품으로 사용하는 것도 가능합니다.

CMOS 센서 보드는 1.3M 급의 Global Shutter CMOS 센서 (MT9M031)을 사용하며 초당 45 frame 으로 영상이 출력됩니다. 출력되는 영상은 베이스 보드를 거쳐 카멜레온 보드의 FPGA 에 입력되며, FPGA 에서는 영상 데이터를 USB 전송에 적합하도록 가공하여 USB 전송칩인 FX3 칩에 전달합니다. FX3 칩은 USB 3.0 전송 칩으로 영상 데이터를 USB 3.0 인터페이스를 이용하여 PC 로 전송하는 역할을 합니다.

제품 특징

- 3 채널 CMOS 센서 : 1280x960, 45FPS
- FPGA 를 통한 I/O 제공
- USB 3.0 super speed 인터페이스
- 관련 자료 제공

보드의 크기

외형 치수

OjOcam-3Ch Board 는 그림과 같은 크기를 갖고 있습니다. PCB 에는 센서 보드와 베이스 보드를 연결할 수 있는 Cable 커넥터가 있으며 베이스 보드에는 USB 3.0 전송 보드인 카멜레온 보드를 결합할 수 있는 커넥터가 있습니다.

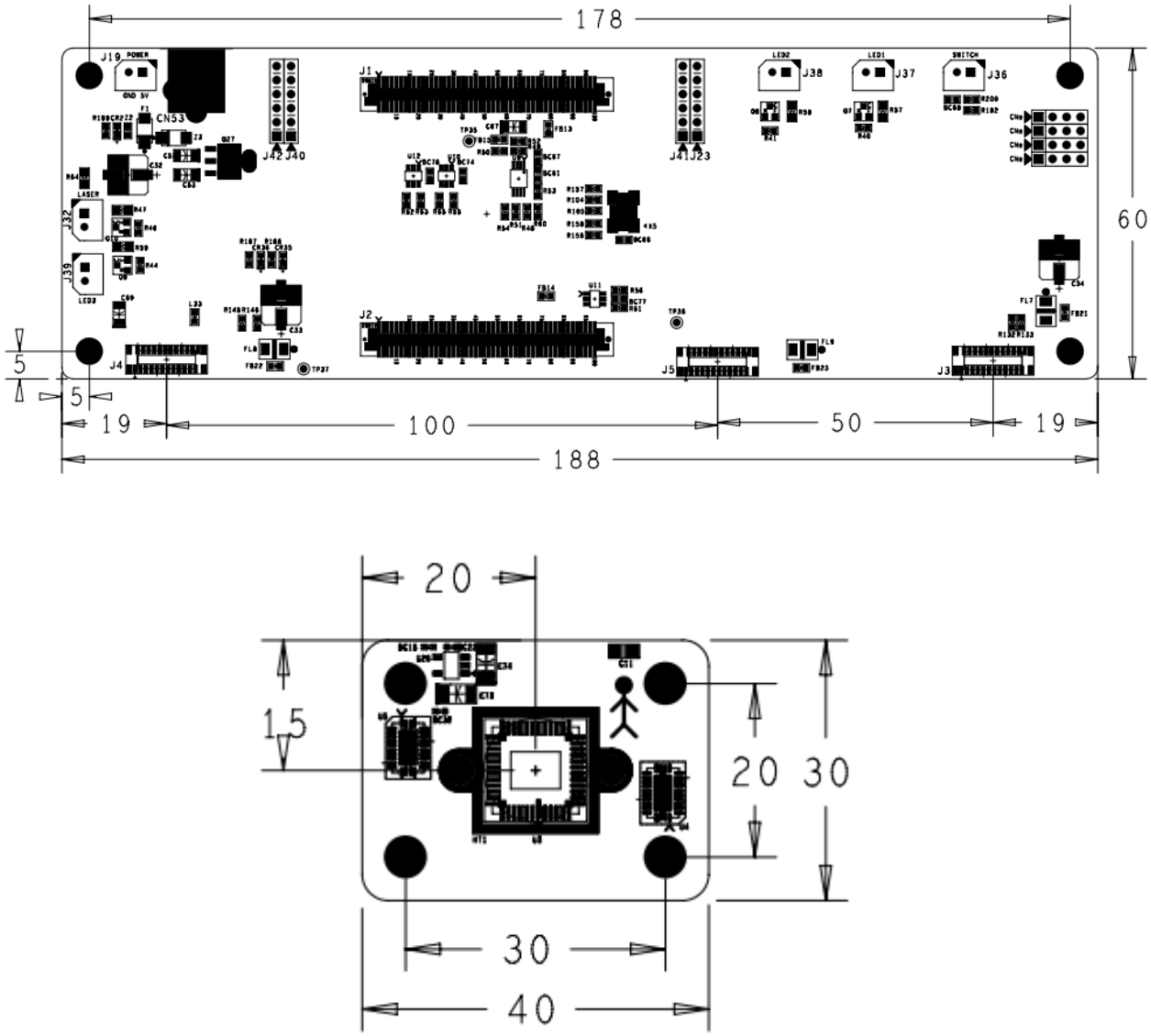


그림 2. 보드 치수

프로그램 실행

드라이버 설치

OjOcam-3Ch Board 를 동작시키기 위해서는 먼저 드라이버를 설치하여야 합니다. 드라이버 디렉토리를 보면 아래와 같이 win7, wlh, wxp 로 나누어져 있는데, win7 은 윈도우 7 용(윈도우 8

호환)이며 wlh 는 윈도우 비스타, wxp 는 윈도우 XP 용입니다. 64 비트 버전과 32 비트 버전이 나누어져 있으므로 사용하시는 시스템에 맞는 드라이버를 설치하시면 됩니다.

그림 3. 드라이버 디렉토리

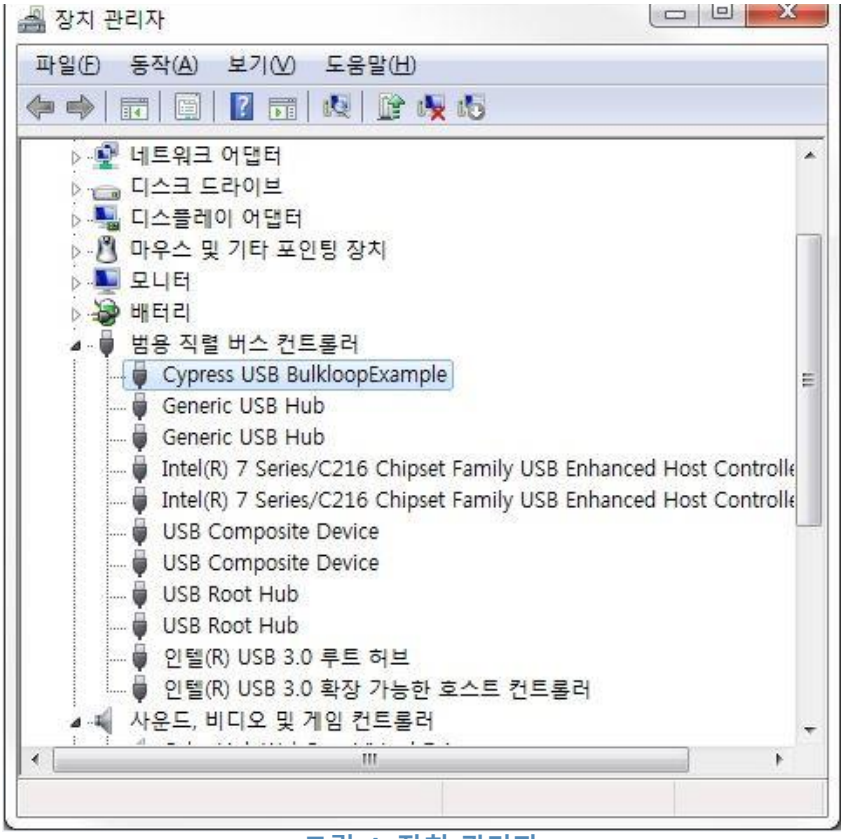


그림 4. 장치 관리자

드라이버가 무사히 설치되었으면 그림과 같이 장치 관리자에 장치가 나타납니다.

프로그램 실행

영상을 확인하기 위해서는 OjOcamMulti.exe 를 실행하시면 됩니다. OjOcamMulti.exe 는 위드로봇에서 제작되는 멀티 카메라를 위한 예제 프로그램으로 OjOcam-3Ch 을 비롯해 추후

출시 예정인 멀티카메라를 지원합니다. OjOcamMulti.exe 를 실행하시면 그림과 같은 프로그램이 실행됩니다. "Cam On" 버튼을 누르면 영상이 나오기 시작하고 "Cam Off" 버튼을 누르면 영상이 정지됩니다. "확인" 버튼을 누르면 프로그램이 종료합니다.

OjOcamMulti.exe 는 Visual C++ 2010 을 이용하여 제작되었습니다. 만일 컴퓨터에 Visual Studio 가 설치되어 있지 않아 프로그램이 실행되지 않을 경우 아래의 재배포 가능 패키지를 설치해 주시면 됩니다.

- Microsoft Visual C++ 2010 재배포 가능 패키지(x86)
  http://www.microsoft.com/ko-kr/download/details.aspx?id=5555
- Microsoft Visual C++ 2010 재배포 가능 패키지(x64)
  http://www.microsoft.com/ko-kr/download/details.aspx?id=14632

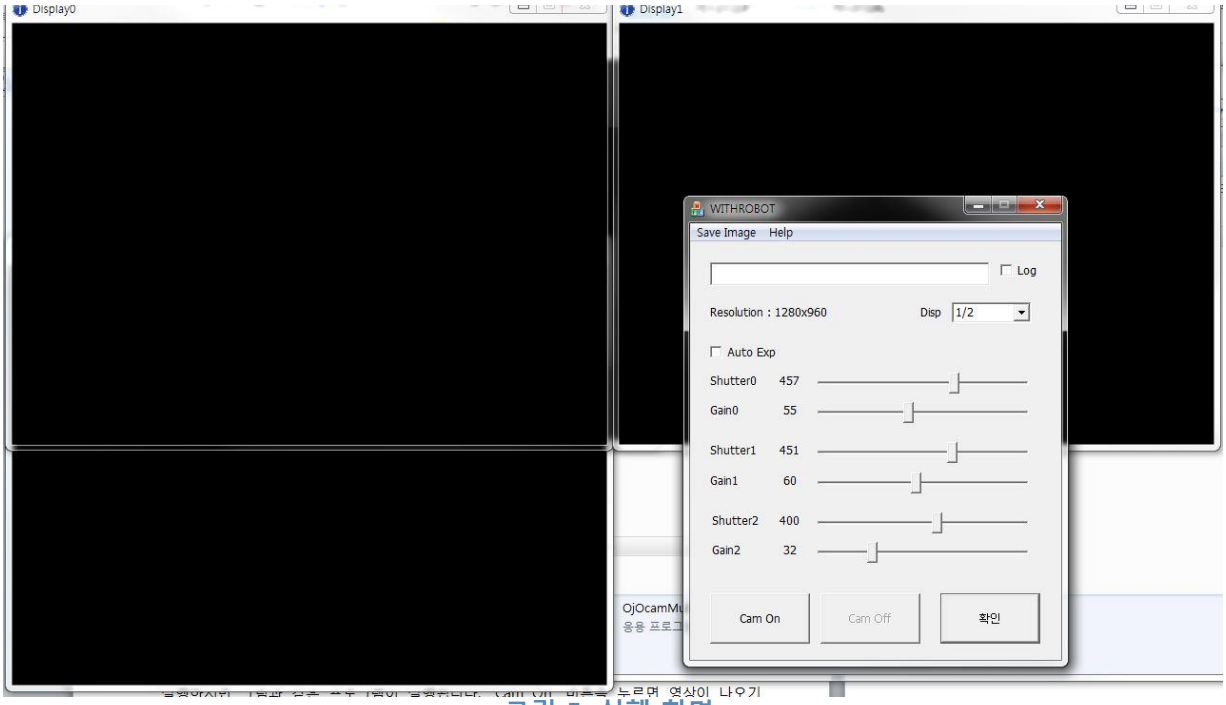


그림 5. 실행 화면.

프로그램 포팅

OjOcamMulti 를 다른 프로그램에서 사용하기 위하여는 간단한 포팅 과정을 거치면 됩니다. 비주얼 스튜디오를 이용하여 예제 프로그램을 제작하여 OjOcamMulti 를 포팅하는 법을 설명드리도록 하겠습니다. (비주얼 스튜디오 2010 을 기준으로 하였습니다.)

MFC 기반 프로그램

비주얼 스튜디오에서 MFC 프로그램을 하나 생성합니다. 프로젝트 이름은 TestMFC 로 하고 다이얼로그 기반으로 만듭니다.

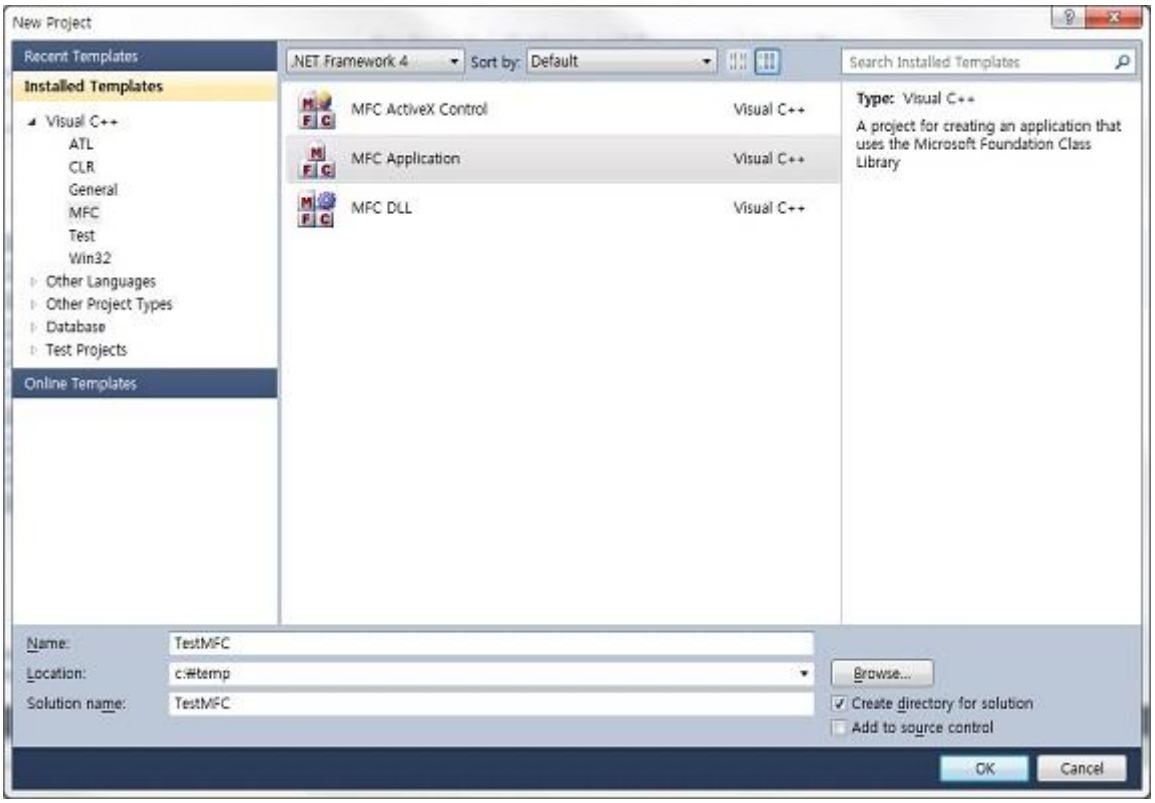


그림 6. MFC 기반 프로젝트 생성

TestMFC 프로젝트가 생성되었으면 프로젝트 디렉토리에 포팅을 위하여 아래와 같은 파일들을 복사해 넣습니다.

CyAPI.h, CyAPI.lib, CyUSB30_def.h

Camera.h, Camera.cpp

wDisplay.h, wDisplay.cpp

wImage.h, wImage.cpp

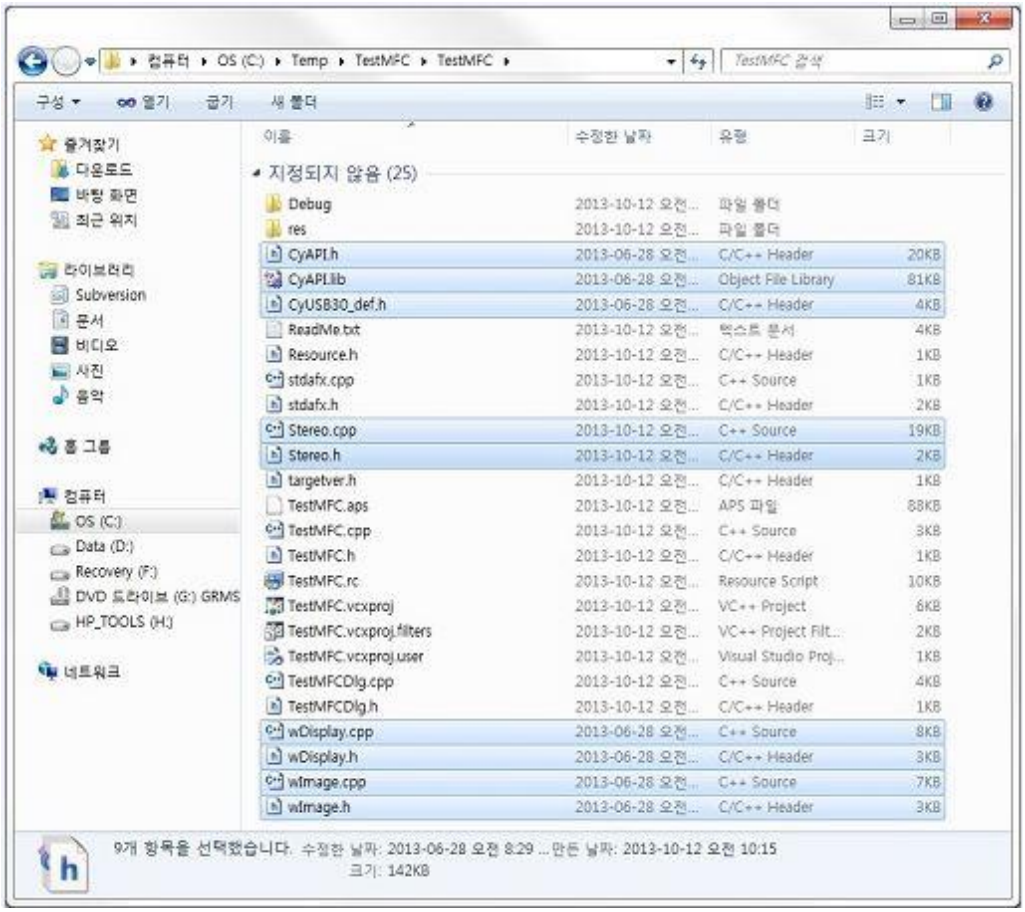


그림 7. 필요한 파일 복사

CyAPI 관련 파일의 경우 FX3 를 구동하기 위한 파일들이고 Camera.h, Camera.cpp 파일은 OjOcamMulti 를 동작시키기 위한 파일입니다. wDisplay 와 wImage 관련 파일은 캡쳐된 영상을 화면에 보여주기 위한 파일로 OpenCV 등의 다른 함수를 사용하셔도 됩니다.

위의 파일을 디렉토리에 복사하였으면 프로젝트에 포함시킵니다. 이때 CyAPI 관련 파일들은 Camera.cpp 에서 자체적으로 불러 쓰기 때문에 굳이 추가할 필요는 없습니다. Stereo, wImage, wDisplay 관련 파일만 추가하면 됩니다.

위의 파일들은 유니코드로 작성되어 있지 않으므로 프로젝트 PropertyPages 에 들어가서 Character Set 을 유니코드에서 Multi-Byte 로 변경합니다.

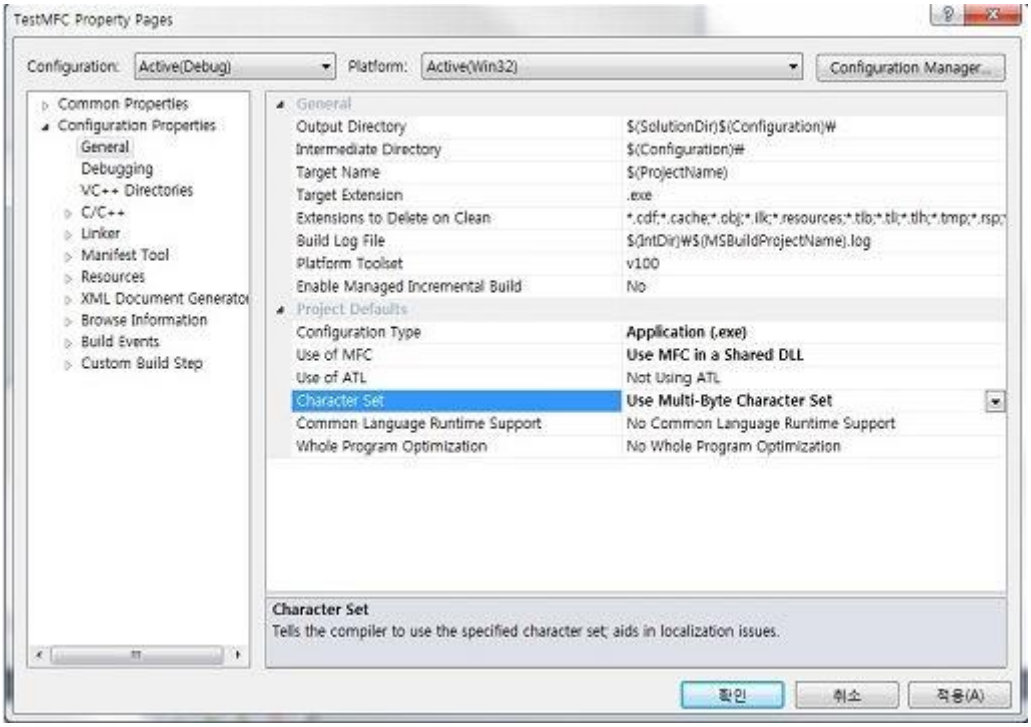


그림 8. Character Set 변경

또한 CyAPI.lib 라이브러리와 충돌이 나는 부분이 있으므로 Linker->Input->Ignore Specific Default Libraries 에 LIBCMT 를 추가합니다.

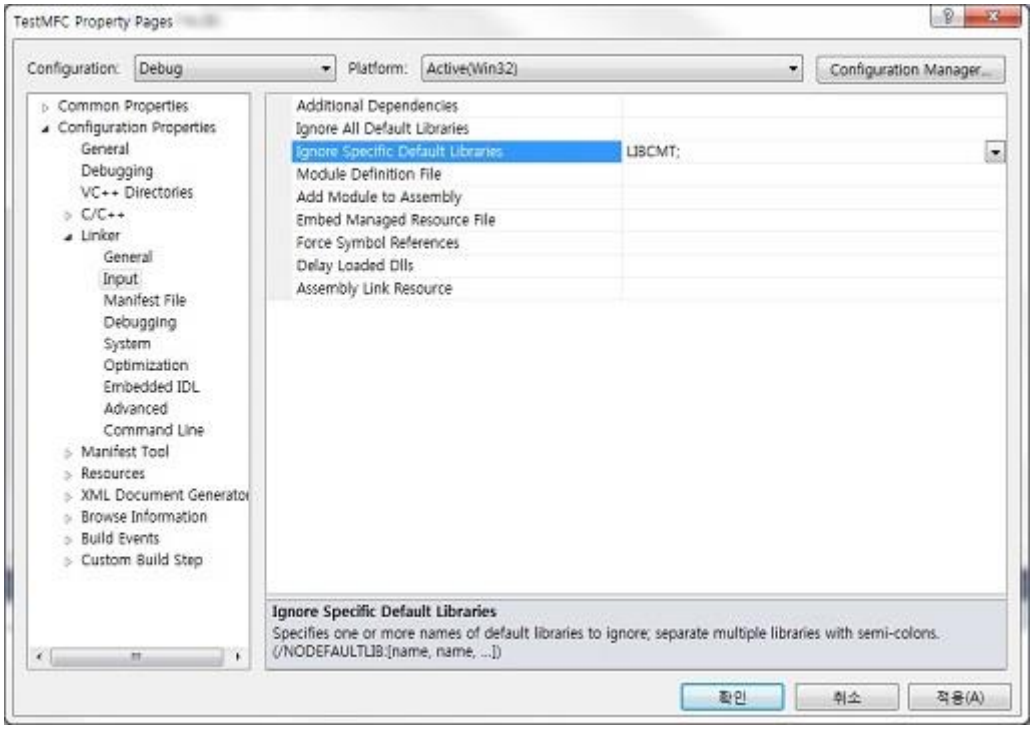


그림 9. LIBCMT 라이브러리 무시

위와 같이 설정하면 기본적인 설정은 끝났습니다. 이제 관련 코드를 작성합니다. 먼저 TestMFCDlg.h 에 아래와 같이 헤더 파일과 관련 코드를 추가합니다.

```
#pragma once

#include "wImage.h"
#include "wDisplay.h"
#include "Camera.h"
.
.
	Camera		m_Camera;
	wImage		m_Image[NUM_CAM];
	wDisplay	m_Display[NUM_CAM];
.
.
.
public:
	afx_msg void OnBnClickedButtonCamOn();
	afx_msg void OnBnClickedButtonCamOff();
	afx_msg void OnTimer(UINT_PTR nIDEvent);
```

리소스 에디터에서 아래와 같이 Cam On, Cam Off 버튼을 만들고 관련 함수와 연결해 줍니다.

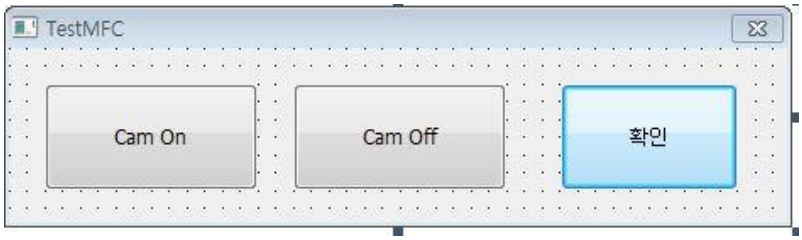


그림 10. 버튼 생성

관련된 코드를 아래와 같이 작성해 줍니다. 코드가 단순하므로 이해하는데 어려움은 없으리라 생각합니다.

```
void CTestMFCDlg::OnBnClickedButtonCamOn()
{
	// TODO: Add your control notification handler code here
	m_Display[0].ShowWindow(SW_SHOW);
```

```
	m_Display[1].ShowWindow(SW_SHOW);

	m_Image[0].Alloc(1280, 960, MV_Y8);
	m_Image[1].Alloc(1280, 960, MV_Y8);

	m_Camera.Open(0, 1280, 960);
	m_Camera.Run();

	SetTimer(1, 50, NULL);
}

void CTestMFCDlg::OnBnClickedButtonCamOff()
{
	// TODO: Add your control notification handler code here
	KillTimer(1);

	m_Camera.Stop();
	m_Camera.Close();

	m_Image[0].Free();
	m_Image[1].Free();
}

void CTestMFCDlg::OnTimer(UINT_PTR nIDEvent)
{
	// TODO: Add your message handler code here and/or call default
	BYTE *buf0 = (BYTE *)m_Image[0].GetPtr1D();
	BYTE *buf1 = (BYTE *)m_Image[1].GetPtr1D();

	if (m_Camera.GetImages(buf0, buf1))
	{
		m_Display[0].Display(m_Image[0]);
		m_Display[1].Display(m_Image[1]);
	}

	CDialogEx::OnTimer(nIDEvent);
}
```

이제 Cam On 버튼을 눌러 프로그램을 동작시키면 아래와 같이 무사히 동작함을 확인하실 수 있으실 것입니다.

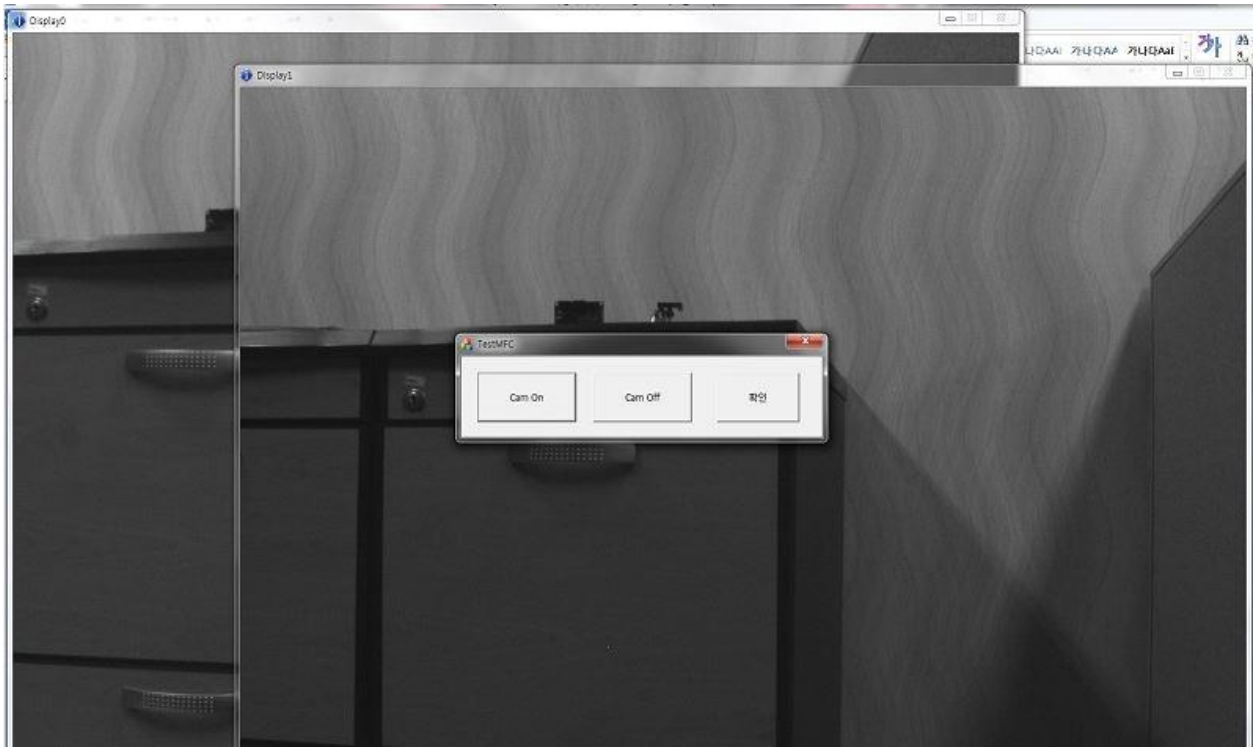


그림 11. TestMFC 의 동작

콘솔 기반 프로그램

콘솔 기반 프로그램도 MFC 기반 프로그램과 유사합니다. 먼저 콘솔 기반 프로젝트를 하나 생성합니다 프로젝트의 이름은 TestConsole 로 합니다.

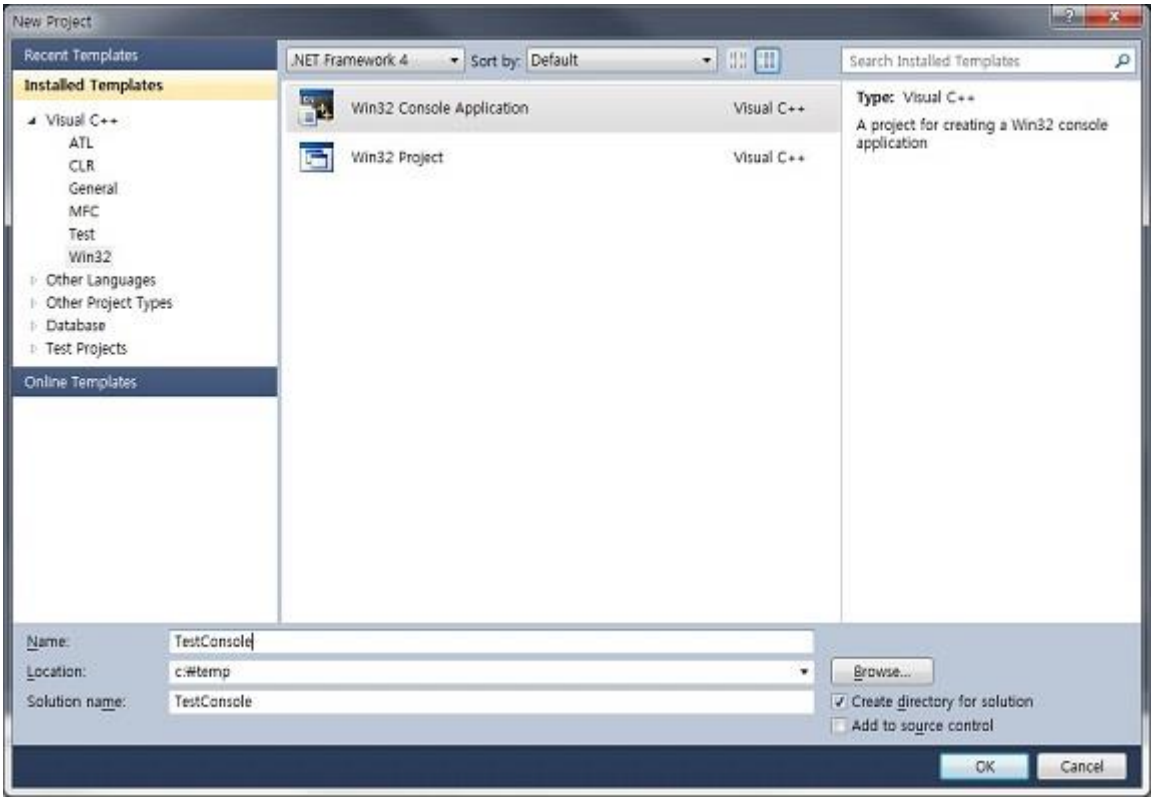


그림 12. 콘솔 기반 프로젝트 생성

프로젝트를 생성할 때는 반드시 MFC 를 체크해 줍니다.

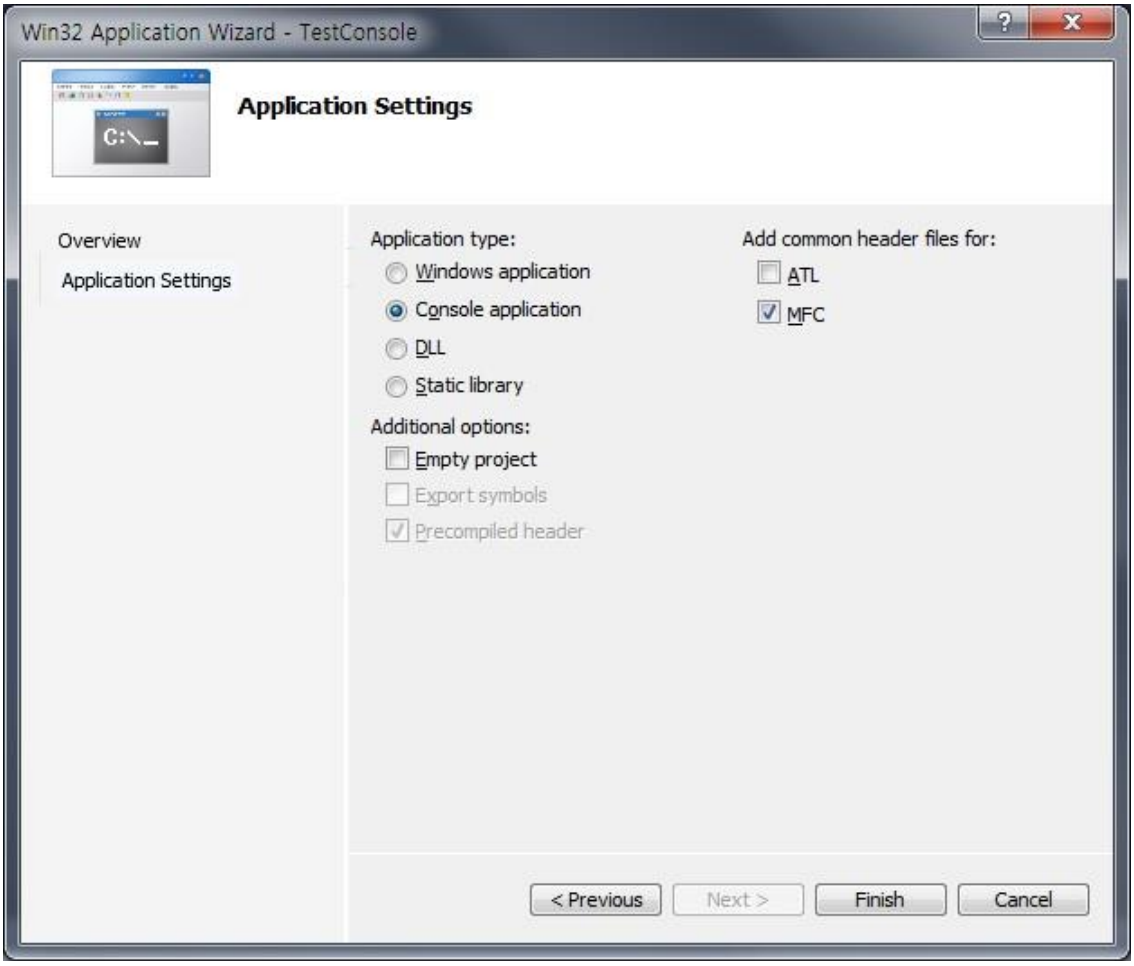


그림 13. MFC 체크박스 체크

TestMFC 에서와 마찬가지로 프로젝트에 아래의 파일들을 추가합니다.

CyAPI.h, CyAPI.lib, CyUSB30_def.h

Camera.h, Camera.cpp

wDisplay.h, wDisplay.cpp

wImage.h, wImage.cpp

TestMFC 에서와 마찬가지로 Character Set 을 유니코드에서 Multi-Byte 로 변경하고 Linker->Input->Ignore Specific Default Libraries 에 LIBCMT 도 추가합니다.

코드 상에서 아래와 같이 코드를 추가합니다.

```
#include "wImage.h"
#include "wDisplay.h"
#include "Camera.h"
.
.
	Camera		m_Camera;
	wImage		m_Image[NUM_CAM];
	wDisplay	m_Display[NUM_CAM];

	m_Display[0].ShowWindow(SW_SHOW);
	m_Display[1].ShowWindow(SW_SHOW);

	m_Image[0].Alloc(1280, 960, MV_Y8);
	m_Image[1].Alloc(1280, 960, MV_Y8);

	m_Camera.Open(0, 1280, 960);
	m_Camera.Run();

	BYTE *buf0 = (BYTE *)m_Image[0].GetPtr1D();
	BYTE *buf1 = (BYTE *)m_Image[1].GetPtr1D();

	while (!_kbhit())
	{
		if (m_Camera.GetImages(buf0, buf1))
		{
			m_Display[0].Display(m_Image[0]);
			m_Display[1].Display(m_Image[1]);
		}

		MSG  msg;
		if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
		{
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}

		Sleep(10);
	}

	m_Camera.Stop();
```

```
m_Camera.Close();

m_Image[0].Free();
m_Image[1].Free();
```

프로그램을 동작시키면 아래와 같이 프로그램이 동작하는 것을 확인할 수 있습니다.

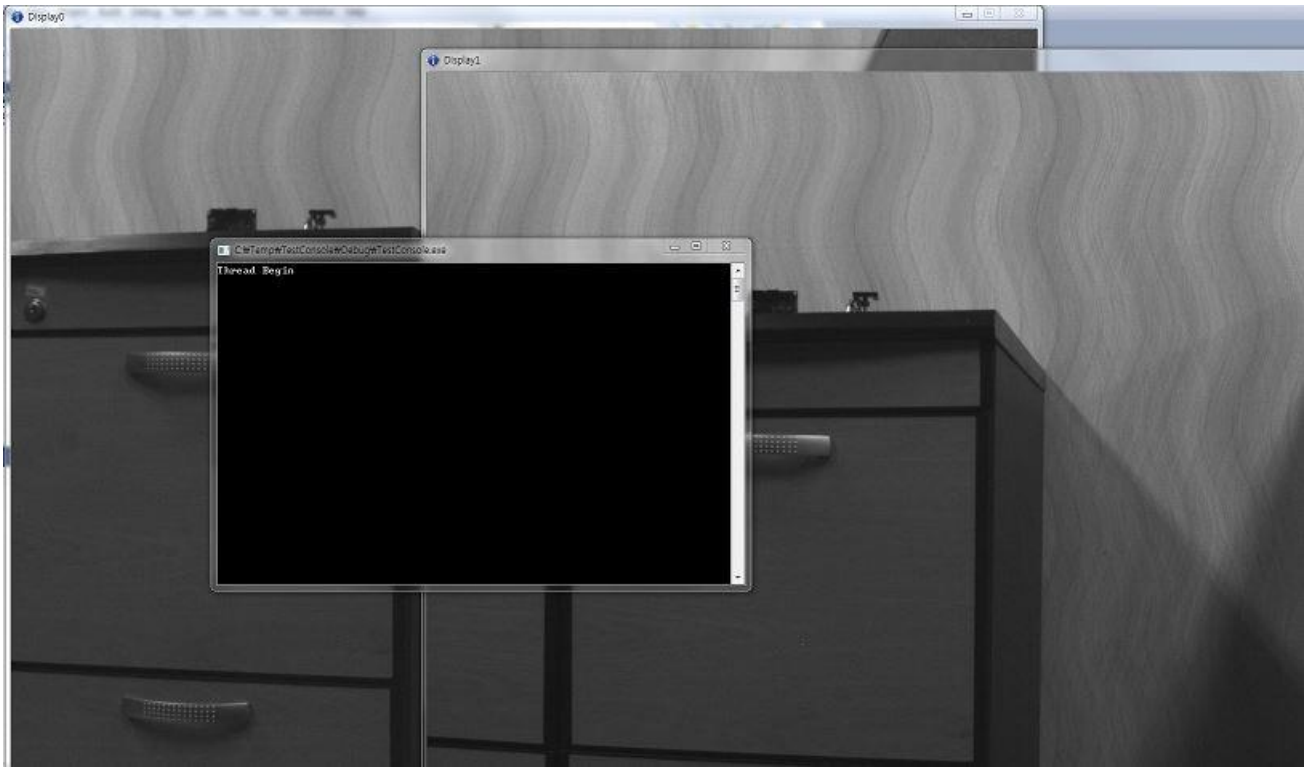


그림 14. 콘솔 기반 프로그램

사용시 주의 사항

- FFC Cable 연결 시 무리하게 힘을 가하면, 커넥터나 Cable 에 손상이 생길 수 있으며, 이는 화면 깨짐 등의 문제를 일으킬 수 있습니다.
- 개조, 변형에 의한 파손은 제조사에서 책임지지 않습니다.

Release Information

The following changes have been made in this document.

Change history

Date	Issue	변동 사항
2016.10	A	The first draft