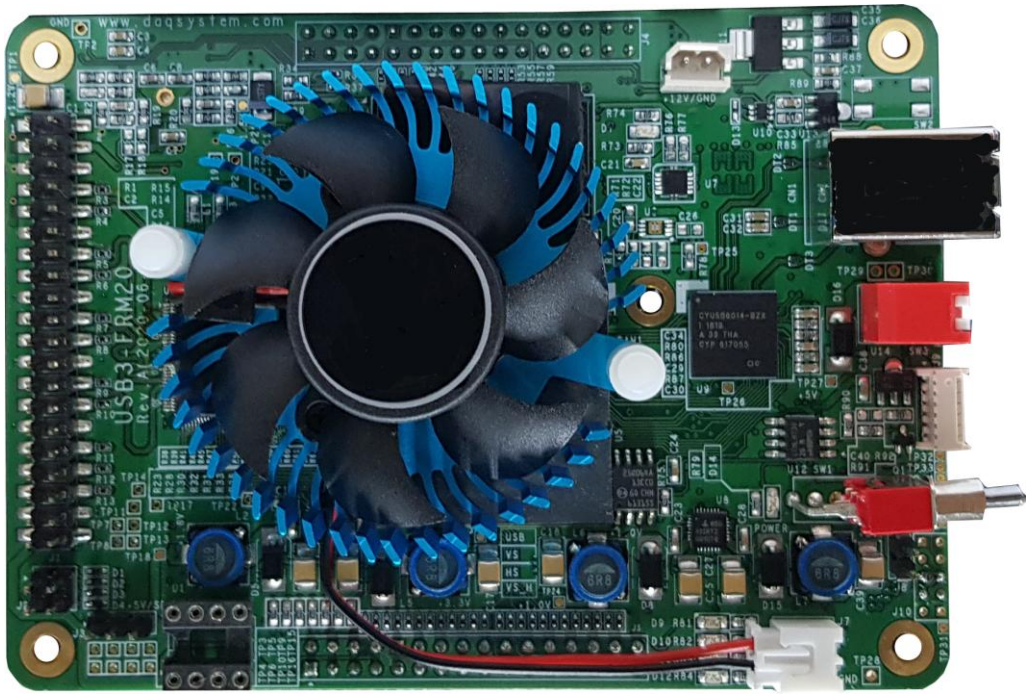
# USB3-FRM20

## API 매뉴얼

**버전 1.1**






DAQ SYSTEM
Measurement & Automation

# 목 차

## Board Level API Functions


OpenDAQDevice ---------- 4
ResetBoard ---------- 4
CloseDAQDevice ---------- 5
GetBoardNum ---------- 5
GetDllVersion ---------- 5
SendUsrMsg ---------- 6
IsConnected ---------- 6


## LVDS API Functions


LVDS_Init ---------- 7
LVDS_Start ---------- 8
LVDS_GetFrame ---------- 8
LVDS_GetError ---------- 9
LVDS_Close ---------- 9
LVDS_GetResolution ---------- 9
LVDS_SetResolutionDelay ---------- 10
LVDS_Stop ---------- 10
LVDS_SetDataMode ---------- 10
LVDS_VirtualMode ---------- 11
LVDS_GetVersion ---------- 11
LVDS_GetMipiID ---------- 11
LVDS_BufferFlush ---------- 12
LVDS_CheckSum ---------- 12
LVDS_SetRolling ---------- 12


## Clcock API Functions


CLK_Select ---------- 13
CLK_Set ---------- 13
CLK_Off ---------- 14

## Parallel GPIO Input/Output API Functions

DIO_SetDirection ---------- 15
DIO_GetDirection ---------- 15
DIO_Read ---------- 16
DIO_Write ---------- 16
DIO_GetWrite ---------- 16

## Sensor GPIO Input/Output API Functions

SDIO_SetDirection ---------- 17
SDIO_GetDirection ---------- 17
SDIO_Read ---------- 18
SDIO_Write ---------- 18
SDIO_GetWrite ---------- 18

## User GPIO(3.3V Fix) Input/Output API Functions

DIO33_SetDirection ---------- 19
DIO33_GetDirection ---------- 19
DIO33_Read ---------- 20
DIO33_Write ---------- 20
DIO33_GetWrite ---------- 20

## Power Boatd SP(Sensor Power) API Functions

SP_SetDirection ---------- 21
SP_GetDirection ---------- 21
SP_Write ---------- 22

## I2C API Functions

I2C_PWR_Init ---------- 24
I2C_PWR_Read ---------- 24
I2C_PWR_Write ---------- 25
I2C_PWR_Close ---------- 25
I2C_ADC_Init ---------- 26
I2C_ADC_Read ---------- 26
I2C_ADC_Write ---------- 27
I2C_SDC_Close ---------- 27

I2C_OS_Init ---------- 28
I2C_OS_Read ---------- 28
I2C_OS_Write ---------- 29
I2C_OS_Close ---------- 29
I2C_SYS_Reset ---------- 29
I2C_SYS_Set_Clock ---------- 30
I2C_SYS_Read ---------- 30
I2C_SYS_Write ---------- 31
I2C_SYS_Read2 ---------- 31
I2C_SYS_Write2 ---------- 32
I2C_SYS_Read_Ex ---------- 32
I2C_SYS_Read2_Ex ---------- 33
I2C_SYS_Write_Ex ---------- 33
I2C_SYS_Write2_Ex ---------- 34
SEN_Reset ---------- 34
SEN_Enable ---------- 35
SEN_Power ---------- 35

## Power Control API Functions

PWR_IO_Voltage ---------- 36
PWR_IO_ON ---------- 36
PWR_IO_Off ---------- 37
I2C_SEN_On ---------- 37
I2C_SEN_Off ---------- 37
I2C_BOOT_Addr_Sel ---------- 38

## SPI(Serial Peripheral Interface) API Functions

SPI_Send ---------- 39
FAN_Conrol ---------- 39

# Board Level APIs

***Overview***

**Int** **OpenDAQDevice (void)**
**BOOL** **ResetBoard (int nBoard)**
**BOOL** **CloseDAQDevice (void)**
**Int** **GetBoardNum (void)**
**char*** **GetDllVersion (void)**
**BOOL** **SendUserMsg (int nBoard, unsigned char* chBuf, BOOL bReply)**
**BOOL** **IsConnected (int nBoard)**

## OpenDAQDevice

디바이스를 Open한다.
프로그램에서 초기에 반드시 한번 함수를 호출하여 디바이스를 Open 하여야 한다.

**Int** **OpenDAQDevice (void)**

**Parameters**: 없음.

**Return Value**:

함수 호출에 성공한 경우, 설치된 보드의 번호를 리턴한다.
함수 호출에 실패한 경우, "-1"을 리턴한다. 이것의 의미는 시스템에 장치가 없다는 의미이다.

## ResetBoard

현재 시스템(PC)에 장착된 디바이스를 초기화 한다.

**BOOL** **ResetBoard (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 성공일 경우 "TRUE"을 리턴함.
Open된 보드가 없을 경우 "FALSE" 를 리턴한다.

## CloseDAQDevice

Open된 디바이스를 Close한다. 장치의 사용이 끝나게 되면, 반드시 장치를 Close 하여 다른 프로그램에서 사용할 수 있도록 한다.

**BOOL** **CloseDAQDevice (void)**

**Parameters**: 없음.

**Return Value**:

디바이스 Open에 성공할 경우 TRUE를 실패할 경우 FALSE를 리턴한다.

## GetBoardNum

시스템에서 설치된 보드 개수를 알려준다.

**int** **GetBoardNum (void)**

**Parameters**: 없음.

**Return Value**:

설치된 보드의 개수를 리턴한다.

보드 넘버는 장착되어있는 DIP 스위치(SW1)의 설정 값이 넘어온다.

## GetDllVersion

DLL 버젼을 알려준다.

**Char*** **GetDllVersion (void)**

**Parameters**: 없음.

**Return Value**:

설치되어 있는 DLL의 날짜가 넘어 온다.

## SendUserMsg

메시지 전송을 시작한다.

**BOOL SendUserMsg(int nBoard, unsigned char *chBuf, BOOL bReply)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

chBuf : 전송할 메시지가 있는 번지

bReply : 메시지 전송이 성공이면 "1"
메시지 전송이 실패면 "0"

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## IsConnected

USB에 연결이 되어 있는지를 알려준다.

**BOOL IsConnected (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

# LVDS API Functions

## *Overview*

**BOOL LVDS_Init (int nBoard)**
**BOOL LVDS_Start (int nBoard)**
**BOOL LVDS_GetFrame (int nBoard, DWORD* nCnt, unsigned char* buf)**
**BOOL LVDS_GetError ((int nBoard, DWORD *dwStatus)**
**BOOL LVDS_Close (int nBoard)**
**BOOL LVDS_GetResolutuion (int nBoard, DWORD *xRes, DWORD *yRes)**
**BOOL LVDS_SetResolutuionDelay (int nBoard, DWORD dwDelay)**
**BOOL LVDS_Stop (int nBoard)**
**BOOL LVDS_SetDataMode (int nBoard, int nMode)**
**BOOL LVDS_VirtualMode (int nBoard, BOOL bSet)**
**BOOL LVDS_GetVersion (int nBoard, int *nFpgaVer, int nFirmVer)**
**BOOL LVDS_SelectInput (int nBoard, Int nInput)**
**BOOL LVDS_GetMipiID (int nBoard, DWORD *dwID)**
**BOOL LVDS_BufferFlush (int nBoard)**
**BOOL LVDS_CheckSum (int nBoard, BOOL bOn)**
**BOOL LVDS_SetRolling (int nBoard, BOOL bRolling)**

## LVDS_Init

LVDS sub-system의 자원, 예를 들어 interrupt와 LVDS control register을 초기화한다.

**BOOL LVDS_Init (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## LVDS_Start

LVDS_Start 함수 호출이 되면 Board에서 DLL로 Data를 전송 시작한다.
(USB Board의 경우 Thread를 통하여 DLL의 Buffer에 Data를 쌓고, LVDS_GetFrame함수로 DLL -> APP 로 copy 한다.)

**BOOL LVDS_Start (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## LVDS_GetFrame

프레임 데이터가 완성이 되었는가를 검사하고 완성이 되었으면 프레임 데이터를 가져온다. 이때 데이터를 받아올 버퍼 크기를 반드시 알려주어야 한다.

**BOOL LVDS_GetFrame (int nBoard, DWORD* nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

nCnt : 프레임 데이터를 받아 올 버퍼의 크기가 저장 되어있는 변수의 번지이다.
함수를 호출할때 버퍼크기를 지정하고 호출한 후에는 변수 값을 읽어서 실제로 읽어 온 개수를 확인한다. 데이터 크기는 바이트 단위이다.

buf : 프레임 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.
호출이 성공하면 nCnt 값을 확인하여 원하는 크기 만큼 읽어 졌는가를 확인한다.

㈜ Open이 되지 않았거나, DLL Thread가 돌지 않는 경우, USER가 입력한 nCnt 값이 Buffer Size보다 큰 경우 nCnt를 "0"으로 Return 하고,
아직 프레임 데이터가 덜 쌓인 경우에도 return "FALSE"만 합니다.

## LVDS_GetError

프레임(이미지) 에러를 가져온다.

**DWORD LVDS_GetError (int nBoard, DWORD *dwStatus)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwStatus : 에러 상태 값을 가져온다.
"1" 이면 Overflow error, "2" 이면 Read error
"4" 이면 Size error

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_Close

LVDS 함수에서 사용된 모든 자원을 되돌려준다. 어플리케이션 프로그램은 프로그램 종료 시 이 함수를 부른다.

**BOOL LVDS_Close (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## LVDS_GetResolution

Video 입력의 해상도를 가져온다.

**BOOL LVDS_GetResolutuion (int nBoard, DWORD *xRes, DWORD *yRes)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

*nXres : 수평해상도 값을 얻어올 번지.

*nYres : 수직해상도 값을 얻어올 번지.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_SetResolutionDelay

Video 입력의 해상도 delay(지연) 시간를 설정한다.

**BOOL LVDS_SetResolutuionDelay (int nBoard, DWORD dwDelay)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwDelay : milisecond unit (Max. 10sec)

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_Stop

프레임 데이터 Capture를 중지한다.

**BOOL LVDS_Stop (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_SetDataMode

입력되는 프레임(이미지) 데이터 모드를 설정한다.

**BOOL LVDS_SetDataMode (int nBoard, int nMode)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

nMode : "0" 이면 8bit Mode이고, "1" 이면 16bit Mode
"2" 이면 24bit Mode이고, "3" 이면 32bit Mode
"4" 이면 16bit YUV Mode 이다.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_VirtualMode

Virtual 데이터 모드를 설정한다.

**BOOL LVDS_VirtualMode (int nBoard, BOOL bSet)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

bSet : "1" 이면 Virtual mode 이다.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_GetVersion

FPGA 와 Firmware version을 가져온다.

**BOOL LVDS_GetVersion (int nBoard, int *nFpgaVer, int *nFirmVer)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.

*nFpgaVer : FPGA 버전. *nFirmVer : Firmware 버전

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_GetMipiID

MIPI 이미지 ID를 읽어온다.

**BOOL LVDS_GetMipiID (int nBoard, DWORD : *dwID)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwID: MIPI Image ID 번지.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_BufferFlush

버퍼를 초기화 한다.

**BOOL LVDS_BufferFlush (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_CheckSum

CRC Check Sum 실행 유무를 결정한다.

**BOOL LVDS_SelectInput (int nBoard, BOOL bOn)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP스위치를 이용하여 설정한다.
bOn : "0" 이면 CheckSum Data를 추가하지 않는다.
"1" 이면 CheckSum 2Bytes Data를 Frame의 Line 끝마다 추가한다.

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## LVDS_SetRolling

LVDS 프레임 데이터를 GetFrame 함수 사용 없이 DDR 메모리를 업데이트 한다.

**BOOL LVDS_SetRolling (int nBoard, BOOL bRolling)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.
bRolling : "1" 이면 Getframe 함수에 관계없이 DDR 메모리를 업데이트

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# Clock API Functions

## *Overview*

**BOOL CLK_Select (int nBoard, int nSelect)**

**BOOL CLK_Set (int nBoard, DWORD val)**

**BOOL CLK_Off (int nBoard, BOOL bOff)**

## CLK_Select

센서에 제공하는 MCLK(Master Clock 의 종류을 선정한다.

**BOOL CLK_Select (int nBoard, int nSelect)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nSelect : "0" 이면 Fixed Clock, "Others" 이면 Program Clock

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## CLK_Set

MCLK(Master Clock 의 주파수를 설정한다. (1039Hz ~ 68Mhz까지 세팅할 수 있다.)

**BOOL CLK_Set (int nBoard, DWORD val)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

val : 1039Hz ~ 68Mhz.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## CLK_Off

MCLK(Master Clock) 을 동작을 On/Off 시킨다.

**BOOL CLK_Off (int nBoard, BOOL bOff)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

bOff : "0" 이면 : Clock On , '1" 이면 Clock Off

**Return Value** :

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# Parallel GPIO Input/Output API Functions

## *Overview*

**BOOL DIO_SetDirection (int nBoard, DWORD dwVal)**
**BOOL DIO_GetDirection (int nBoard, DWORD *dwVal)**
**DWORD DIO_Read (int nBoard)**
**BOOL DIO_Write (int nBoard, DWORD dwVal)**
**BOOL DIO_GetWrite (int nBoard, DWORD *dwVal)**

## DIO_SetDirection

각각의 포트를 입력으로 사용할지 출력으로 사용할지 설정한다.
(Parallel GPIO setting 하위 8비트)

**BOOL DIO_SetDirection (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 입/출력 direction 설정 값.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO_GetDirection

현재 설정된 direction 값을 읽어 온다.

**BOOL DIO_GetDirection (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 입/출력 direction을 읽어올 변수.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO_Read

입력 값을 읽어 온다.

**DWORD DIO_Read (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO_Write

원하는 값을 출력포트에 출력한다.

**BOOL DIO_Write (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 출력 포트에 기록할 값.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO_GetWrite

출력된 현재 값을 적는다.

**BOOL DIO_GetWrite (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 출력 포트의 현재 값을 쓸 변수이다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# Sensor GPIO Input/Output API Functions

## *Overview*

**BOOL SDIO_SetDirection (int nBoard, DWORD dwVal)**
**BOOL SDIO_GetDirection (int nBoard, DWORD *dwVal)**
**DWORD SDIO_Read (int nBoard)**
**BOOL SDIO_Write (int nBoard, DWORD dwVal)**
**BOOL SDIO_GetWrite (int nBoard, DWORD *dwVal)**

## SDIO_SetDirection

센서 각각의 포트를 입력으로 사용할지 출력으로 사용할지 설정한다.
(Parallel GPIO setting 하위 8비트)

**BOOL SDIO_SetDirection (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 입/출력 direction 설정 값.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## SDIO_GetDirection

센서의 현재 설정된 direction 값을 읽어 온다.

**BOOL SDIO_GetDirection (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 입/출력 direction을 읽어올 변수.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## SDIO_Read

센서 입력 값을 읽어 온다.

**DWORD SDIO_Read (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## SDIO_Write

센서에 원하는 값을 출력포트에 출력한다.

**BOOL SDIO_Write (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 출력 포트에 기록할 값.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## SDIO_GetWrite

센서에 출력된 현재 값을 적는다.

**BOOL SDIO_GetWrite (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 출력 포트의 현재 값을 쓸 변수이다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# User GPIO(3.3V Fix) Input/Output API Functions

## *Overview*

**BOOL DIO33_SetDirection (int nBoard, DWORD dwVal)**
**BOOL DIO33_GetDirection (int nBoard, DWORD *dwVal)**
**DWORD DIO33_Read (int nBoard)**
**BOOL DIO33_Write (int nBoard, DWORD dwVal)**
**BOOL DIO33_GetWrite (int nBoard, DWORD *dwVal)**

## DIO33_SetDirection

사용자 GPIO 각각의 포트를 입력으로 사용할지 출력으로 사용할지 설정한다.
(Parallel GPIO setting 하위 8비트)

**BOOL DIO33_SetDirection (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 입/출력 direction 설정 값.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO33_GetDirection

사용자 GPIO 설정된 direction 값을 읽어 온다.

**BOOL DIO33_GetDirection (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 입/출력 direction을 읽어올 변수.
각 포트에 '1'이면 출력 / '0'이면 입력

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO33_Read

사용자 GPIO 입력 값을 읽어 온다.

**DWORD DIO33_Read (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO33_Write

사용자 GPIO 원하는 값을 출력포트에 출력한다.

**BOOL DIO33_Write (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : 출력 포트에 기록할 값.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

## DIO33_GetWrite

사용자 GPIO 출력된 현재 값을 적는다.

**BOOL DIO33_GetWrite (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : 출력 포트의 현재 값을 쓸 변수이다.

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# Power Board SP(Sensor Power) API Functions

## *Overview*

**BOOL SP_SetDirection (int nBoard, DWORD dwVal)**

**BOOL SP_GetDirection (int nBoard, DWORD *dwVal)**

**BOOL SP_Write (int nBoard, int nCh, float fVolt)**

## SP_SetDirection

SP(Sensor Power) 중에 어떤 채널을 사용할지 설정한다. 이 함수는 PowerBoard를 사용하는 보드에서만 사용된다.

**BOOL SP_SetDirection (int nBoard, DWORD dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dwVal : Channel 값.
0x00 => ALL Off,
0x01 => Ch1(SP0) Off, 0x02 => Ch2(SP1) Off
0x04 => Ch3(SP2) Off, 0x08 => Ch4(SP3) Off
0x10 => Ch5(SP4) Off, 0x20 => Ch6(SP5) Off
0x40 => Reserve, 0x80 => Reserve

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.
(PowerBoard를 사용하지 않는 로직에서는 -1을 return 한다. )

## SP_GetDirection

SP(Sensor Power) 중에 어떤 채널을 가져올지 설정한다. 이 함수는 PowerBoard를 사용하는 보드에서만 사용된다.

**BOOL SP_GetDirection (int nBoard, DWORD *dwVal)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

*dwVal : Channel 값.

0x00 => ALL Off,

0x01 => Ch1(SP0) On, 0x02 => Ch2(SP1) On

0x04 => Ch3(SP2) On, 0x08 => Ch4(SP3) On

0x10 => Ch5(SP4) On, 0x20 => Ch6(SP5) On

0x40 => Reserve, 0x80 => Reserve

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

(PowerBoard를 사용하지 않는 로직에서는 -1을 return 한다. )

## SP_Write

SP(Sensor Power) 의 각 채널에 전압 값을 설정한다. 이 함수는 PowerBoard를 사용하는 보드에서만 사용된다.

**BOOL SP_Write (int nBoard, int nCh, float fVolt)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.

보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nCh : Channel 값.

0x01 : SP0

0x02 : SP1

0x04 : SP2

0x08 : SP3

0x10 : SP4

0x20 : SP5

fVolt : 전압 값 (1.0 ~ 4.0V)

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

fVolt 값이 0.89보다 작거나 4.1 보다 큰 경우는 -2를 return한다.

(PowerBoard를 사용하지 않는 로직에서는 -1을 return 한다. )

# I2C API Functions

## *Overview*

**BOOL I2C_PWR_Init (int nBoard, DWORD nKHz)**

**BOOL I2C_PWR_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_PWR_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_PWR_Close (int nBoard)**

**BOOL I2C_ADC_Init (int nBoard, DWORD nKHz)**

**BOOL I2C_ADC_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_ADC_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_ADC_Close (int nBoard)**

**BOOL I2C_OS_Init (int nBoard, DWORD nKHz)**

**BOOL I2C_OS_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_OS_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_OS_Closel (int nBoard)**

**BOOL I2C_SYS_Reset (int nBoard)**

**BOOL I2C_SYS_Set_Clock (int nBoard, int nClock)**

**BOOL I2C_SYS_Read (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Write (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Read2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Write2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Read_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Read2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Write_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)**

**BOOL I2C_SYS_Write2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)**

**BOOL SEN_Reset (int nBoard, BOOL bReset)**

**BOOL SEN_Enable (int nBoard, BOOL bEnable)**

**BOOL SEN_Power (int nBoard, BOOL bOn)**

## I2C_PWR_Init

Power 모듈의 I2C 기능을 정해진 클럭 속도에 따라 초기화한다. 프로그램 기동 시 한 번 호출한다. 다른 I2C PWR 함수는 이 함수를 호출한 이후에 사용한다.

**BOOL I2C_PWR_Init (int nBoard, DWORD nKHz)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nKHz : I2C 통신 속도(Clock 속도를 설정한다.)
1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_PWR_Read

I2C를 통하여 Power 모듈의 특정 어드레스 번지 값을 정해진 개수만큼 읽어 온다.

**BOOL I2C_PWR_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_PWR_Write

I2C를 통하여 Power 모듈의 특정 어드레스 번지 값을 정해준 개수만큼 기록한다.

**BOOL I2C_PWR_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_PWR_Close

I2C 기능 사용이 완전히 끝나고 리소스를 해제할 때 호출한다. 프로그램 종료 시 한번 호출하면 된다.

**BOOL I2C_PWR_Close (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_ADC_Init

AD Converter 모듈의 I2C 기능을 정해진 클럭 속도에 따라 초기화한다. 프로그램 기동 시 한 번 호출한다. 다른 I2C ADC 함수는 이 함수를 호출한 이후에 사용한다.

**BOOL I2C_ADC_Init (int nBoard, DWORD nKHz)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nKHz : 1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_ADC_Read

I2C를 통하여 AD Converter 모듈의 특정 어드레스 번지 값을 정해진 개수만큼 읽어 온다.

**BOOL I2C_ADC_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_ADC_Write

I2C를 통하여 AD Converter 모듈의 특정 어드레스 번지 값을 정해준 개수만큼 기록 한다.

**BOOL I2C_ADC_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_ADC_Close

I2C 기능 사용이 완전히 끝나고 리소스를 해제할 때 호출한다. 프로그램 종료 시 한번 호출하면 된다.

**BOOL I2C_ADC_Close (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_OS_Init

Open/Short 모듈의 I2C 기능을 정해진 클럭 속도에 따라 초기화한다. 프로그램 기동 시 한 번 호출한다. 다른 I2C OS 함수는 이 함수를 호출한 이후에 사용한다.

**BOOL I2C_OS_Init (int nBoard, DWORD nKHz)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nKHz : I2C 통신 속도(Clock 속도를 설정한다.)
1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_OS_Read

I2C를 통하여 Open/Short 모듈의 특정 어드레스 번지 값을 정해진 개수만큼 읽어 온다.

**BOOL I2C_OS_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_OS_Write

I2C를 통하여 Open/Short 모듈의 특정 어드레스 번지 값을 정해준 개수만큼 기록한다.

**BOOL I2C_OS_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_OS_Close

I2C 기능 사용이 완전히 끝나고 리소스를 해제할 때 호출한다. 프로그램 종료 시 한번 호출하면 된다.

**BOOL I2C_OS_Close (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Reset

시스템 모듈의 I2C system의 자원을 초기화 한다.

**BOOL I2C_SYS_Reset (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Set_Clock

시스템 모듈의 I2C system의 baud rate을 설정한다.

**BOOL I2C_SYS_Set_Clcok (int nBoard, int nClock)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nClock : 100KHz, 200KHz, 300KHz (default :100000)

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Read

시스템 모듈의 I2C를 통하여 특정 어드레스 번지 값을 정해진 개수만큼 읽어 온다.

**BOOL I2C_SYS_Read (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Write

시스템 모듈의 I2C를 통하여 특정 어드레스 번지 값을 정해진 개수만큼 읽어 온다.

**BOOL I2C_SYS_Write (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Read2

I2C를 통하여 데이터를 받는다. Repeated start 가능

**BOOL I2C_SYS_Read2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : 문자를 나타내는 어드레스를 바이트 사이즈로 받는다.
현재 최대 받을 수 있는 문자는 1024byte로 제한되어 있다.

buf : 버퍼 어드레스.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Write2

I2C를 통하여 데이터를 보낸다. Read Back Mark 가능

**BOOL I2C_SYS_Write2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : 문자를 나타내는 어드레스를 바이트 사이즈로 받는다.
현재 최대 전송할 수 있는 문자는 1Kbyte(1024)로 제한되어 있다.

buf : 버퍼 어드레스.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Read_Ex

시스템 모듈의 I2C를 통하여 지정된 어드레스 길이만큼 어드레스 버퍼에서 count 개수만큼 읽어 온다.

**BOOL I2C_SYS_Read_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, Unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Read2_Ex

시스템 모듈의 I2C를 통하여 지정된 어드레스 길이만큼 어드레스 버퍼에서 count 개수만큼 읽어 온다. Repeated start 가능

**BOOL I2C_SYS_Read2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, Unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : 읽어 올 데이터의 바이트 개수.

buf : 읽어 올 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Write_Ex

시스템 모듈의 I2C를 통하여 지정된 어드레스 길이만큼 어드레스 버퍼에서 count 개수만큼 읽어 온다.

**BOOL I2C_SYS_Write_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, Unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SYS_Write2_Ex

시스템 모듈의 I2C를 통하여 지정된 어드레스 길이만큼 어드레스 버퍼에서 count 개수만큼 읽어 온다. Read Back Mark 가능

**BOOL I2C_SYS_Write2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, Unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

slAddr : I2C address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : 기록할 데이터의 바이트 개수.

buf : 기록할 데이터의 버퍼 포인터.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## SEN_Reset

MIPI 센서 reset 신호를 제어한다.

**BOOL SEN_Reset (int nBoard, BOOL bReset)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

bReset : "0" 이면 Disable
"1" 이면 Enable

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## SEN_Enable

MIPI 센서 Enable 신호를 제어한다.

**BOOL SEN_Enable (int nBoard, BOOL bEnable)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

bEnable : "0" 이면 Disable
"1" 이면 Enable

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## SEN_Power

5V 전원을 On/Off 시킨다.

**BOOL SEN_Power (int nBoard, BOOL bOn)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

addr : "0" 이면 +5V Power Off
"1" 이면 +5V Power On

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

# Power Control API Functions

## *Overview*

**BOOL** **PWR_IO_Voltage (int nBoard, float fVoltage)**

**BOOL** **PWR_IO_On (int nBoard)**

**BOOL** **PWR_IO_Off (int nBoard)**

**BOOL** **I2C_SEN_On (int nBoard)**

**BOOL** **I2C_SEN_Off (int nBoard)**

**BOOL** **I2C_BOOT_Addr_Sel (int nBoard, BOOL addr)**

## PWR_IO_Voltage

센서의 전압 값을 SendUserMsg 함수에 실어 보낸다.
(최소값은 1.25V).

**BOOL** **PWR_IO_Voltage (int nBoard, float fVoltage)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

fVoltage : 전압 값.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## PWR_IO_On

센서의 IO 전원을 enable 시킨다.

**BOOL** **PWR_IO_On (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## PWR_IO_Off

센서의 IO 전원을 disable 시킨다.

**BOOL** **PWR_IO_Off (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SEN_On

센서의 I2C를 enable 시킨다.

**BOOL** **I2C_SEN_On (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_SEN_Off

센서의 I2C를 disable 시킨다.

**BOOL** **I2C_SEN_Off (int nBoard)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## I2C_BOOT_Addr_Sel

EPROM I2C를 enable or disable 시킨다.

**BOOL I2C_BOOT_Addr_Sel (int nBoard, BOOL addr)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

addr : "0" 이면 EPROM I2C Disable , "1" 이면 EPROM I2C Enable.

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

# SPI(Serial Peripheral Interface) API Functions

## *Overview*

**BOOL SPI_Send (int nBoard, DWORD dwLen, DWORD dwData0, DWORD dwData1)**

**BOOL FAN_Control (int nBoard, BOOL bOn, BYTE bySpeed)**

## SPI_Send

SPI(Serial Peripheral Interface) 에 데이터를 전송한다.

**BOOL SPI_Send (int nBoard, DWORD dWlen, DWORD dwData0, DWORD dwData1)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

dWlen : 데이터의 길이(8바이트 이내). 0이거나 8보다 크면 FALSE.

dwData0 : 전송할 데이터 0

dwData1 : 전송할 데이터 1

**Return Value**:

함수 호출에 실패할 경우 "FALSE" 성공일 경우 "TRUE"을 리턴함.

## FAN_Control

FAN 을 On/OFF하고 속도를 제어한다.

**BOOL FAN_Control (int nBoard, BOOL bOn, BYTE bySpeed)**

**Parameters**:

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

bEn : 참이면 Fan On , 거짓이면 Fan Off

bySpeed: FAN Speed 값

**Return Value**:

함수 호출에 실패한 경우 'FALSE', 성공일 경우 'TRUE'를 return한다.

# Memo

## Contact Point

Web sit : https://www.daqsystem.com

Email : postmaster@daqsystem.com

DAQ SYSTEM
Measurement & Automation