🕮 pv$CS

🗁 pv$CS

plantVIEW 2.0에서 사용하는 데이터베이스를 억세스할 수 있는 애플리케이션을 작성하려는 사용자를 위하여 다음의 API를 제공한다. API를 사용하기 위해서는 애플리케이션에 "pv$cs.h" 헤더 파일을 include하고, "pv$cs.lib" 라이브러리 파일을 링크해야 한다. 사용자는 애플리케이션 작성시 아래의 함수를 호출함으로써 데이터베이스의 억세스가 가능하다. 제공하는 억세스 방식에는 GET, PUT, 그리고 SET이 있다. GET은 데이터베이스에서 데이터를 READ하는 것이고, PUT은 데이터베이스에 데이터를 WRITE하는 것이다. 그리고 SET은 SERVER로 하여금 제어를 행하게 하는 것이다.

CALLING CONVENTION

함수의 type과 각 파라미터들에 대한 설명이다. 각 설정 정보에 따라 액세스를 처리하고 필요하다면 frame 버퍼에 데이터를 담아서 넘겨준다. 그러면 애플리케이션에서 frame 버퍼에 있는 데이터를 원하는 곳에 사용하면 되는 것이다.

---

pv$cs(command, address, band, frame)

command ; 코맨드를 설정하는 파라미터

address ; [Node↑ Table↑ Signal↑ Tag↑ Field]

Node ; SERVER가 있는 노드 이름

Table ; Access하려는 DB Table 이름

-> DB의 구체적인 내용은 매뉴얼의 "OLMP" 부분을 참고

Signal ; 신호의 종류 (ANALOG / DISCRETE)

Tag_cnt; 신호의 개수

Tag[] ; 신호의 이름 배열

Field ; 신호에 대해 원하는 데이터 이름 (필드 이름)

➔ 필드의 구체적인 내용은 매뉴얼의 "태그정의 필드" 부분을 참고

band ; 데이터 검색 시간의 범위를 설정하는 파라미터(long형 배열)

band[0] ; 검색 시작 시간 (초단위)

band[1] ; 검색 끝 시간 (초단위)

frame ; 데이터 버퍼

data count ; 데이터의 개수

data type ; 데이터의 종류

('F' : float, 'S' : string, 'T' : time and value)

data length ; 데이터의 바이트 수

data [] ; 데이터의 배열

처리 결과 ; 아래의 값들을 리턴해주고 에러 발생시 에러 메시지를 frame 버퍼에 넘겨준다.

[message]	[value]	[comment]
NORMAL	= 0	Data Base Access Success
NO NODE	= -1	Invalid node name
INIT FAIL	= -2	Initializing error
NO COMMAND	= -3	Invalid command
SET FAIL	= -4	Invalid control info
REQUEST FAIL	= -5	Request fail
RECEIVE FAIL	= -6	Receive fail

APPLICATION INCLUDE FILE

함수 호출 시 파라미터로 설정 정보를 넘겨줄 때 아래의 헤더 파일을 참고하여 설정한다. 이 헤더 파일은 엔지니어링 작업이 끝난 프로젝트의 ON-LINE DIRECTORY에 존재하게 된다.

---

pv$cs.h file contents

```
#include "optional_db.h"
// 1. command define
#define cs$put         =  221
#define cs$get         =  222
```

```
#define cs$set         =  231

// 2. static db table define
#define TDF$ANALOG     =    1
#define TDF$DISCRETE   =    2
#define ADF$ALARM      =  500
#define CDF$TRANSITION = 1000

// 3. signal define
#define ANALOG   = 1
#define DISCRETE = 2

// 4. 신호 이름 define : from PV-OLMP and PV-LINK
#define PMP_PRESSURE = 1

// 5. field define
#define ANA$ENG_UNIT =  4
#define ANA$PV       = 83
#define STS$PV       = 63
#define DUMP         =  1

typedef structure Address_Stru
    char *node;
    int   table;
    int   signal;
    int   tag_cnt;
    int   tag[MX_TAG_CNT];
    int   field;
 ADDRESS_STRU

 ADDRESS_STRU address;
```

```
        long        band[2];
        char        frame[MAX_BUFFER];


optional_db.h file contents : from PV-OLMP and PV-LINK
      #define UDF$SEC_001 =  11
      #define UDF$MIN_001 =  12
      #define UDM$SEC_001 = 911
```

Example

다음은 실제 함수를 사용하여 데이터베이스를 액세스하는 예제 프로그램의 소스이다. Analog Table에 대해서 PMP_PRESSURE 신호의 현재 값을 READ하고(1), 해당 신호의 단위를 WRITE한다(2). Historical DB에 대해서는 PMP_PRESSURE 신호에 대해 현재 시간으로부터 10초전까지 1초 간격으로 저장되어져 있는 데이터를 READ한다.

```
#include "pv$cs.h"

main()

    int   i, ids, data_count, data_len, data_type;

// 1. Read Data From Analog Table
    // 1.1 setting data information
    address.node = "YOON-W.J"; address.table = TDF$ANALOG;
    address.signal = ANALOG;   address.tag_cnt = 1;
    address.tag[0] = PMP_PRESSURE;  address.field = ANA$PV;

    // 1.2 communication
    ids = pv$cs ( cs$get, address, NULL, frame );

    // 1.3 application
    if( ids == 0 )
```

```
        data_count = *(int*)&frame[0];  data_type = *(int*)&frame[4];
        data_len   = *(int*)&frame[8];
        switch( data_type )
            case 'F':
                for( i=0; i<data_count; i++ )
                    printf( "Requested Data is [%f] ",
                            *(float*)&frame[12 + data_len*i] );
                break;
            case 'S':
                for( i=0; i<data_count; i++ )
                    printf( "Requested Data is [%s] ", &frame[12 +
                            data_len*i] );
                break;

    else
        printf( "Data Read Failed..." );

// 2. Write Data From Analog Table
    // 2.1 setting data information
    address.node = "YOON-W.J";    address.table = TDF$ANALOG;
    address.signal = ANALOG;   address.tag_cnt = 1;
    address.tag[0] = PMP_PRESSURE;  address.field = ANA$ENG_UNIT;

    // 2.2 setting write information
    *(int*) &frame[0] = 1;  *(int*) &frame[4] = 'S';  strcpy( &frame[12],
             "kg/cm2" );
    *(int*) &frame[8] = strlen( &frame[12] );

    // 2.3 communication
    ids = pv$cs( cs$put, address, NULL, frame );
```

```
    // 2.4 application
    if( ids == 0 )
        printf( "Data Write Success..." );

    else
        printf( "Data Write Failed..." );

// 3. Read Data From SEC-001.UDF
    // 3.1 setting data information
    address.node = "YOON-W.J";    address.table = UDF$SEC_001;
    address.signal = ANALOG;   address.tag_cnt = 1;
    address.tag[0] = PMP_PRESSURE;  address.field = DUMP;
    time( &band[1] );    band[0] = band[1] - 10;

    // 3.2 communication
    ids = pv$cs( cs$get, address, band, frame );

    // 3.3 application
    if( ids == 0 )
        data_count = *(int*)&frame[0];  data_type = *(int*)&frame[4];
        data_len   = *(int*)&frame[8];
        switch( data_type )
            case 'F':
                for( i=0; i<data_count; i++ )
                    printf( "Requested Data is [%d -> %f] ",
                            band[0] + i, *(float*)&frame[12 +
                            data_len*i] );
                break;
            case 'S':
                for( i=0; i<data_count; i++ )
                    printf( "Requested Data is [%s] ", &frame[12 +
```

```
                                    data_len*i] );
                        break;

    else
printf( "Data Read Failed..." );
```