Allegro Hand CAN Protocol Specification

SIMLAB CO., LTD.

Version 1.0.0

Allegro Hand CAN Protocol Specification

Hyobong Bldg 2nd Fl, 1425-9 Seocho-Dong, Seocho-Gu,
Seoul 137-864, Korea
Tel +82-2-3471-2014 • Fax +82-2-6280-9931

Copyright & Trademark Notice

Table of Contents

List of Figures

그림 목차 항목을 찾을 수 없습니다.

List of Tables

그림 목차 항목을 찾을 수 없습니다.

List of Code

그림 목차 항목을 찾을 수 없습니다.

1. CAN communication

1.1. Baud-rate

Baud-rate is 1Mbps.

1.2. Non-periodic communication

CAN 통신을 초기화하거나 주기통신을 시작 혹은 중지하기 위한 CAN 메시지

1.3. Periodic communication

Allegro Hand 제어를 위해 제어 프로그램은 주기적으로 통신을 시도한다. 계산된 토크 입력을 매 3 millisecond 마다 전송하며 이에 대한 응답으로 각 관절의 각도가 갱신된다.

2. CAN frames

Standard CAN 패킷(데이터 8bytes)을 사용한다.

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


2.1. ID(Message identifier)

아이디는 4bytes integer 인데 효율적인 통신을 위하여 명령어(26 bits), 송신측 아이디(3 bits), 수신측 아이디(3 bits)로 분할하여 사용한다.

1 8 16 24 26	27 29	30 32
Command id	destination id	sorce id

2.1.1. Command id

Variable name	Value	Description	Sorce	Destination
ID_CMD_SET_SYSTEM_ON	0x01	주기 통신 시작 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_SYSTEM_OFF	0x02	주기 통신 종료 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_PERIOD	0x03	통신 주기 설정 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_MODE_JOINT	0x04	제어 모드 설정 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_MODE_TASK	0x05	제어 모드 설정 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_TORQUE_1	0x06	첫번째 손가락(index finger)에 토크 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_TORQUE_2	0x07	두번째 손가락(middle finger)에 토크 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_TORQUE_3	0x08	세번째 손가락(little finger)에 토크 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_TORQUE_4	0x09	엄지(thumb)에 토크 명령 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_SET_POSITION_1	0x0a	*(unused)*		
ID_CMD_SET_POSITION_2	0x0b	*(unused)*		
ID_CMD_SET_POSITION_3	0x0c	*(unused)*		
ID_CMD_SET_POSITION_4	0x0d	*(unused)*		
ID_CMD_QUERY_STATE_DATA	0x0e	관절각 요청 송신	ID_DEVICE_MAIN	ID_COMMON
ID_CMD_QUERY_STATE_DATA	0x0e	관절각 응답 수신 (관절각 요청 송신에 대한 응답)	ID_DEVICE_SUB_01 ID_DEVICE_SUB_02 ID_DEVICE_SUB_03 ID_DEVICE_SUB_04	ID_DEVICE_MAIN
ID_CMD_QUERY_CONTROL_DATA	0x0f	관절각 응답 수신	ID_DEVICE_SUB_01 ID_DEVICE_SUB_02 ID_DEVICE_SUB_03 ID_DEVICE_SUB_04	ID_DEVICE_MAIN

관절 토크 전송, 관절각 수신 등 각 명령어의 사용에 관한 실예는 3 장 Case-study: Softing CAN 부분을 참고하시기 바란다.

2.1.2. Sorce/Destination id

Variable name	Value	Description
ID_COMMON	0x01	Allegro Hand 공통(Allegro Hand)
ID_DEVICE_MAIN	0x02	제어 PC(control PC)
ID_DEVICE_SUB_01	0x03	첫번째 손가락(index finger)
ID_DEVICE_SUB_02	0x04	두번째 손가락(middle finger)
ID_DEVICE_SUB_03	0x05	세번째 손가락(little finger)
ID_DEVICE_SUB_04	0x06	엄지(thumb)

3. Case-study: Softing CAN

In this chapter, sample code which implements CAN communication foundation for Softing PCI CAN interface is represented.

3.1. Open CAN communication channel

```
char ch_name[256];
sprintf_s(ch_name, 256, "CAN-ACx-PCI_%d", ch);
INIL2_initialize_channel(&hCAN[ch-1], ch_name);

L2CONFIG L2Config;
L2Config.fBaudrate = 1000.0;
L2Config.bEnableAck = 0;
L2Config.bEnableErrorframe = 0;
L2Config.s32AccCodeStd = 0;
L2Config.s32AccMaskStd = 0;
L2Config.s32AccCodeXtd = 0;
L2Config.s32AccMaskXtd = 0;
L2Config.s32OutputCtrl = GET_FROM_SCIM;
L2Config.s32Prescaler = 1;
L2Config.s32Sam = 0;
L2Config.s32Sjw = 1;
L2Config.s32Tseg1 = 4;
L2Config.s32Tseg2 = 3;
L2Config.hEvent = (void*)-1;
```



```
CANL2_initialize_fifo_mode(hCAN[ch-1], &L2Config);
```

3.2. Initialization

```
long Txid;
unsigned char data[8];

Txid = ((unsigned long)ID_CMD_SET_PERIOD<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
data[0] = (unsigned char)period_msec;
canWrite(hCAN, Txid, data, 1, STD);

Sleep(10);

Txid = ((unsigned long)ID_CMD_SET_MODE_TASK<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
canWrite(hCAN, Txid, data, 0, STD);

Sleep(10);

Txid = ((unsigned long)ID_CMD_QUERY_STATE_DATA<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
canWrite(hCAN, Txid, data, 0, STD);
```

3.3. Start periodic CAN communication

Periodic CAN communication 을 시작하면 제어 토크 입력에 대한 응답으로 갱신된 관절각이 자동으로 전송된다.

```
long Txid;
unsigned char data[8];

Txid = ((unsigned long)ID_CMD_QUERY_STATE_DATA<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
canWrite(hCAN[ch-1], Txid, data, 0, STD);

Sleep(10);

Txid = ((unsigned long)ID_CMD_SET_SYSTEM_ON<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
canWrite(hCAN[ch-1], Txid, data, 0, STD);
```

3.4. Stop periodic CAN communication

```
long Txid;
```



```
unsigned char data[8];

Txid = ((unsigned long)ID_CMD_SET_SYSTEM_OFF<<6) | ((unsigned long)ID_COMMON <<3) | ((unsigned long)ID_DEVICE_MAIN);
canWrite(hCAN[ch-1], Txid, data, 0, STD);
```

3.5. Transmit control torques

Control inputs for four joints in the same finger should be packed in a CAN frame. The sample code below shows how to encode four pwm inputs into 8 bytes data buffers and how to set the CAN frame id properly.

```
long Txid;
unsigned char data[8];
float torque2pwm = 800.0f
short pwm[4] = {
        0.1*torque2pwm,
        0.1*torque2pwm,
        0.1*torque2pwm,
        0.1*torque2pwm
};

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);
        canWrite(hCAN, Txid, data, 8, STD);
}
```

3.6. Receive joint angles

A finger consists of four joints and joint angles for that four joints in a finger can be received through only one CAN packet. The sample code below shows how to decode data buffer to get joint angles.

The sample code assumes that when all fingers are in their zero positions, the joint angles from CAN packet are 32768. But in practice users need to introduce offsets by experiments.

```
char cmd;
char src;
char des;
int len;
unsigned char data[8];
int ret;
can_msg msg;
PARAM_STRUCT param;

ret = CANL2_read_ac(hCAN, &param);

switch (ret)
{
case CANL2_RA_DATAFRAME:
        msg.msg_id = param.Ident;
        msg.STD_EXT = STD;
        msg.data_length = param.DataLength;

        msg.data[0] = param.RCV_data[0];
        msg.data[1] = param.RCV_data[1];
        msg.data[2] = param.RCV_data[2];
        msg.data[3] = param.RCV_data[3];
        msg.data[4] = param.RCV_data[4];
        msg.data[5] = param.RCV_data[5];
        msg.data[6] = param.RCV_data[6];
        msg.data[7] = param.RCV_data[7];

        break;
}

cmd = (char)( (msg.msg_id >> 6) & 0x1f );
des = (char)( (msg.msg_id >> 3) & 0x07 );
src = (char)( msg.msg_id & 0x07 );
len = (int)( msg.data_length );
for(int nd=0; nd<len; nd++)
        data[nd] = msg.data[nd];

switch (cmd)
{
case ID_CMD_QUERY_CONTROL_DATA:
        {
                if (id_src >= ID_DEVICE_SUB_01 && id_src <= ID_DEVICE_SUB_04)
                {
                        int temp_pos[4];  // raw angle data
                        float ang[4];     // degree
                        float q[4];       // radian
```

```
                        temp_pos[0] = (int)(data[0] | (data[1] << 8));
                        temp_pos[1] = (int)(data[2] | (data[3] << 8));
                        temp_pos[2] = (int)(data[4] | (data[5] << 8));
                        temp_pos[3] = (int)(data[6] | (data[7] << 8));

                        ang[0] = ((float)(temp_pos[0]-32768)*(333.3f/65536.0f))*(1);
                        ang[1] = ((float)(temp_pos[1]-32768)*(333.3f/65536.0f))*(1);
                        ang[2] = ((float)(temp_pos[2]-32768)*(333.3f/65536.0f))*(1);
                        ang[3] = ((float)(temp_pos[3]-32768)*(333.3f/65536.0f))*(1);

                        q[0] = (3.141592f/180.0f) * ang[0];
                        q[1] = (3.141592f/180.0f) * ang[1];
                        q[2] = (3.141592f/180.0f) * ang[2];
                        q[3] = (3.141592f/180.0f) * ang[3];
                }
        }
}
```

