Kelly KLS Controller RS232 Commands List

Version 1.1

Data Structure definitions

```
#define ETS_MAX_DATA_LEN            16   //max length,can not be changed

typedef struct _T_Sync_Comm_Buff
{
    unsigned char   command;
    unsigned char   no_bytes;
    unsigned char   data[ETS_MAX_DATA_LEN + 1];
}T_Sync_Comm_Buff;
```

Description: Data structure T_Sync_Comm_Buff are used by the controller with whole data length 19 bytes. A complete data sent to the controller or received from the controller include the command field, no_bytes field and the data field.Serial baud rate is 19200.

The command field indicates the operation. The controller response returns the same command value it received.

The no_bytes field indicates the number of bytes to be sent or received of the data buffer field, excluding the checksum byte.

The data buffer field includes two parts: valid data and the checksum. Before sending T_Sync_Comm_Buff type data, the ETS_TxMsg method make a checksum and place it at the valid data heels. After receiving a complete data from the controller, the ETS_RxMsg method will make a checksum and compare it with the checksum part in the receiving data buffer field.

Refer to implementation of ETS_TxMsg, ETS_RxMsg ,Please See **Appendix B Programming Reference**.

Commands definitions

Command	**ETS_A2D_BATCH_READ**
no_bytes	0
Controller response	
no_bytes	16
data[0]	Brake A/D
data[1]	TPS A/D
data[2]	motor temperature A/D
data[3]	Control power A/D
data[4]	Vs A/D
data[5]	B+ A/D
data[6]	Controller's temperature A/D
data[7]	Ia A/D
data[8]	Ib A/D
data[9]	Ic A/D
data[10]	PCB_Temp
data[11]	Vb A/D
data[12]	Vc A/D
data[13]	H_Temperature
data[14]	V+ A/D
data[15]	L_Temperature

Description: Data batch reading.

1) For control power, B+, Va, Vb, Vc, A/D value and voltage mapping relation is:
   V = Vad / 1.84. (For 120V controller).

V = Vad / 1.39. (For 144V controller).

2) Vs is defined as the 5V power supply for Hall sensor, control panel,ect. A/D value and voltage mapping relation is: 0 ~ 127 mapping to 0 ~ 5V.
3) Brake and TPS are defined as the Brake and the Throttle analog input. A/D value and voltage mapping relation is: 0 ~ 255 mapping to 0 ~ 5V.
4) V+ is defined as the internal voltage of the controller. A/D value and voltage mapping relation is: 153~250 mapping to 9~15 V.
5) motor temperature A/D is defined as the temperature of motor A/D value.
6) data[6],data[13],data[14] are defined as controller inside temperature A/D value.

Command	**ETS_USER_MONITOR1**
no_bytes	0
Controller response	
no_bytes	16
data[0]	TPS A/D
data[1]	Brake A/D
data[2]	BRK_SW
data[3]	FOOT_SW
data[4]	FWD_SW
data[5]	REV_SW
data[6]	HALL_SA
data[7]	HALL_SB
data[8]	HALL_SC
data[8]	B_Voltage
data[10]	Motor_Temp
data[11]	Controller's temperature
data[12]	Setting direction
data[13]	Actual direction
data[14]	Break_SW2
data[15]	

Command	**ETS_USER_MONITOR2**
no_bytes	0
Controller response	
no_bytes	16
data[0]	MSB of controller's error state
data[1]	LSB of controller's error state
data[2]	MSB of mechanical speed in RPM
data[3]	LSB of mechanical speed in RPM
data[4]	
data[5]	
data[6]	
data[7]	
data[8]	
data[8]	
data[10]	
data[11]	
data[12]	
data[13]	
data[14]	
data[15]	

Appendix A: Caldata.cpp

ETS_FLASH_OPEN	0xF1
ETS_FLASH_READ	0xF2
ETS_A2D_BATCH_READ	0x1b
ETS_USER_MONITOR1	0x3A
ETS_USER_MONITOR2	0x3B

Appendix B : Programming Reference

1) RS232 command sending and receiving samples

```
CommClass   m_Comm;     //Object of CommClass
```

a) Connection

```
//Connect to the controller
errorcode = m_Comm.ComOpen(pBuffer);
if(errorcode != COMM_ERROR_OK)
{
    //error dealing
}
else
{
    //operation on connection success
}
```

b) Data sending/receiving sample – ETS_FLASH_OPEN

```
//data sending/receiving with retry-on-fail
    while(errorcode != COMM_ERROR_OK && readloopcount-- > 0)
    {
        m_Comm.m_SyncOutputBuff.command = ETS_FLASH_OPEN;
        m_Comm.m_SyncOutputBuff.no_bytes = 0;
        errorcode = m_Comm.ComSyncSend();    //send data using internal buffer
    }
    if(errorcode != COMM_ERROR_OK)
    {
            //error dealing
    }
    else
    {
            //operation on connection success
    }
```

c) Data sending/receiving sample – ETS_FLASH_READ

```
    m_Comm.m_SyncOutputBuff.command = ETS_FLASH_READ;
    m_Comm.m_SyncOutputBuff.no_bytes = 3;
    m_Comm.m_SyncOutputBuff.data[0] = INFO_SOFTWARE_VER;
    m_Comm.m_SyncOutputBuff.data[1] = 2;
    m_Comm.m_SyncOutputBuff.data[2] = 0;
    errorcode = m_Comm.ComSyncSend();
    if(errorcode != COMM_ERROR_OK)
    {
        //error dealing
    }
    else
    {
        //operation on connection success
    }
```

2) Notice on time delay setting

Controller’s Flash reading/writing operations need some time to finish, so it is necessary to set longer receiving wait time using ComSetDelayTime method in CommClass.

3) ETS_TxMsg 和 ETS_RxMsg

```
T_Sync_Comm_Buff  Rx_Buff,TxBuff;   //define send and receive data buffer field
//transmit message
int ETS_TxMsg(void)      //calculate checksum and transmitter data
{
    char i,check_sum;
    check_sum = 0;
    for (i=0; i<Tx_buff.no_bytes; i++)
      check_sum += Tx_buff.data[i];

    check_sum += Tx_buff.no_bytes;
    check_sum += Tx_buff.command;
    Tx_buff.data[Tx_buff.no_bytes] = check_sum;  //load checksum

    WriteFile();//send
    return 1;
}

int ETS_RxMsg(void)
{
    unsigned char check_sum,k=0,i=0,len=0;
    for(int j=0;j<10;j++)//receive 10 times
    {
        Sleep(5);//wait 5ms
        len=0; //Number of bytes
        ReadFile();//Read the serial port, the data storage into Rx_Buff,
                   Rx_Buff.no_bytes given len
        for (i = 0; i < len; i++)
        {
            check_sum += Rx_Buff.data[i];
        }
        check_sum += Rx_buff.no_bytes;
        check_sum += Rx_buff.command;

        if(check_sum !=Rx_Buff.data[Rx_buff.no_bytes])//checksum error
            return 0;
        else
            return 1;   //OK
    }
    return 2;//don't receive data
}
```