



## 三菱 FX-PLC 的通讯协议参考

说明: 由三菱工控网收集整理, 仅供参考。如有更好的建议, 请向本站提交。 <http://www.5130cn.com>

### 三菱 FX 系列 PLC 专用协议通信指令一览

#### FX 系列 PLC 专用协议通信指令一览

以下将详细列出 PLC 专用协议通信的指令:

指令 注释

BR 以 1 点为单位, 读出位元件的状态

WR 以 16 点为单位, 读出位元件的状态, 或以 1 字为单位, 读出字元件的值

BW 以 1 点为单位, 写入位元件的状态

WW 以 16 点为单位, 写入位元件的状态, 或以 1 字为单位, 写入值到字元件

BT 以 1 点为单位, SET/RESET 位元件

WT 以 16 点为单位, SET/RESET 位元件, 或写入值到字元件

RR 控制 PLC 运行 RUN

RS 控制 PLC 停止 STOP

PC 读出 PLC 设备类型

TT 连接测试

注: 位元件包括 X,Y,M,S 以及 T,C 的线圈等;

字元件包括 D,T,C,KnX,KnY,KnM 等。

### 三菱 FX 系列 PLC 编程口通信协议总览

#### 三菱 FX 系列 PLC 编程口通信协议总览

该协议实际上适用于 PLC 编程端口以及 FX-232AW 模块的通信。感谢网友 visualboy 提供。

通讯格式:

命令 命令码 目标设备

DEVICE READ CMD "0" X,Y,M,S,T,C,D

DEVICE WRITE CMD "1" X,Y,M,S,T,C,D

FORCE ON CMD "7" X,Y,M,S,T,C

FORCE OFF CMD "8" X,Y,M,S,T,C

传输格式: RS232C

波特率: 9600bps



奇偶: even

校验: 累加方式 (和校验)

字符: ASCII

16 进制代码:

ENQ 05H 请求

ACK 06H PLC 正确响应

NAK 15H PLC 错误响应

STX 02H 报文开始

ETX 03H 报文结束

帧格式:

STX CMD DATA ..... DATA ETX SUM(upper) SUM(lower)

例子:

STX ,CMD ,ADDRESS, BYTES, ETX, SUM

02H, 30H, 31H,30H,46H,36H, 30H,34H, 03H, 37H,34H

SUM=CMD+.....+ETX;

30h+31h+30h+46h+36h+30h+34h+03h=74h;

累加和超过两位取低两位

三菱 FX 系列 PLC 编程口通信协议举例

### 三菱 FX 系列 PLC 编程口通信源代码

```
fx_comm.h
```

```
#define DELAY_TIMES 30000L
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
#define FORCE_ON 0x37
```



```
#define FORCE_OFF 0x38
void init_plc(void);
int check_plc(void);
int _read_data_register(unsigned int uAddress,unsigned int number);
int _read_mdata_register(unsigned int uAddress,unsigned int number);
int _write_data_register(unsigned int uAddress,unsigned int number);
int _force_m_contact(unsigned int uAddress,unsigned char ucOn_off);
int read_data_register(unsigned int uAddress,unsigned int number);
int read_mdata_register(unsigned int uAddress,unsigned int number);
int write_data_register(unsigned int uAddress,unsigned int number);
int force_m_contact(unsigned int uAddress,unsigned char ucOn_off);
int _read_m_register(unsigned int uAddress,unsigned int number);
int read_m_register(unsigned int uAddress,unsigned int number);
int TESTING=0;
unsigned int uRead_value[25];
unsigned int uWrite_value[25];
unsigned int COMM_PORT=1;
unsigned int STATS_PORT=0x2fd;
unsigned int DATA_PORT=0x2f8;

void init_plc(void)
{
    _AX=0xfa;
    _DX=COMM_PORT;
    geninterrupt(0x14);
    while((inportb(STATS_PORT)&1)!=0) inportb(DATA_PORT);
}
//返回顶部

int check_plc(void)
{
    long lTmp;
    if(TESTING==1)return TRUE;
    init_plc();
    for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
    {
        if((inportb(STATS_PORT)&0x20)!=0)
            break;
    }
    if(lTmp>=DELAY_TIMES)
        return(FALSE);
    outportb(DATA_PORT,5);
    disable();
    for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
```



```
{ if((inportb(STATS_PORT)&1)!=0)
    break;
}
if(ITmp>=DELAY_TIMES)
{
    enable();
    return(FALSE);
}
if((ITmp=inportb(DATA_PORT))==6)
{
    enable();
    return(TRUE);
}
else
{
    enable();
    return(FALSE);
}
}
//返回顶部

int read_data_register(unsigned int uAddress,unsigned int number)
{
    int i;
    for(i=0;i<3;i++)
        if(_read_data_register(uAddress,number)==TRUE)
            return TRUE;
    return FALSE;
}
//返回顶部

int _read_data_register(unsigned int uAddress,unsigned int number)
{ unsigned char uSend[]={2,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x32,3,0x30,0x30};
    unsigned char uReceive[104];
    unsigned int uTmp;
    unsigned int uSum;
    unsigned int num;
    long lTmp;
    int i,j;
    if(TESTING==1)
    { for(i=0;i<number;i++)uRead_value[i]=0;
        return TRUE;
    }
    init_plc();
    num=number*2;
    if((num/16)>=10)
```



```
        uSend[6]=(unsigned char)(num/16+0x41-10);
    else
        uSend[6]=(unsigned char)(num/16+0x30);
    if((num%16)>=10)
        uSend[7]=(unsigned char)((num%16)+0x41-10);
    else
        uSend[7]=(unsigned char)((num%16)+0x30);
    uAddress=uAddress*2+0x1000;
    uTmp=uAddress & 0x000f;
    uSend[5]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>4) & 0x000f;
    uSend[4]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>8) & 0x000f;
    uSend[3]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>12)&0x000f;
    uSend[2]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uSum=0;
    for(i=1;i<9;i++)
        uSum=uSum+(unsigned char)uSend[i];
    uTmp=uSum&0x000f;
    uSend[10]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uSum>>4)&0x000f;
    uSend[9]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

    for(i=0;i<11;i++)
    { for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
        { if((inportb(STATS_PORT)&0x20)!=0)
            break;
        }
        if(lTmp>=DELAY_TIMES)
        {

            return(FALSE);
        }
        outportb(DATA_PORT,uSend[i]);
    }
    disable();

    for(lTmp=0;lTmp<DELAY_TIMES;lTmp++)
    { if((inportb(STATS_PORT)&1)!=0)
        break;
    }
    if(lTmp>=DELAY_TIMES)
    {
```



```
        enable();
        return(FALSE);
    }
    uReceive[0]=inportb(DATA_PORT);
    if(uReceive[0]!=2)
    {
        enable();
        return(FALSE);
    }
    for(i=1;i<number*4+4;i++)
    { for(!Tmp=0L;!Tmp<DELAY_TIMES;!Tmp++)
        { if((inportb(STATS_PORT)&1)!=0)
            break;
        }
        if(!Tmp>=DELAY_TIMES)
        {
            enable();
            return(FALSE);
        }
        uReceive[i]=inportb(DATA_PORT);
    }
    enable();
    uSum=0;
    for(i=1;i<number*4+2;i++)
        uSum=uSum+(unsigned int)uReceive[i];
    uTmp=uSum&0xf;
    uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

    if((unsigned char)uTmp!=uReceive[number*4+3]) return(FALSE);
    uTmp=(uSum>>4)&0xf;
    uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

    if((unsigned char)uTmp!=uReceive[number*4+2]) return(FALSE);
    for(j=0;j<number;j++)
    {
        for(i=j*4+1;i<j*4+5;i++)
            uReceive[i]=(uReceive[i]>0x39)?uReceive[i]-0x41+0xa:uReceive[i]-0x30;
    }
    uRead_value[j]=((((uReceive[j*4+3]<<4)+uReceive[j*4+4])<<4)+uReceive[j*4+1])<<4)+uReceive[j*4+2];
    }
    return TRUE;
}
//返回顶部
```



```
int read_mdata_register(unsigned int uAddress,unsigned int number)
{
    int i;
    for(i=0;i<3;i++)
        if(_read_mdata_register(uAddress,number)==TRUE)
            return TRUE;
    return FALSE;
}
//返回顶部

int _read_mdata_register(unsigned int uAddress,unsigned int number)
{ unsigned char uSend[]={2,0x30,0x30,0x30,0x30,0x30,0x30,0x32,3,0x30,0x30};
  unsigned char uReceive[104];
  unsigned int uTmp;
  unsigned int uSum;
  unsigned int num;
  long lTmp;
  int i,j;
  if(TESTING==1)
  { for(i=0;i<number;i++)uRead_value[i]=0;
    return TRUE;
  }
  init_plc();
  num=number*2;
  if((num/16)>=10)
      uSend[6]=(unsigned char)(num/16+0x41-10);
  else
      uSend[6]=(unsigned char)(num/16+0x30);
  if((num%16)>=10)
      uSend[7]=(unsigned char)((num%16)+0x41-10);
  else
      uSend[7]=(unsigned char)((num%16)+0x30);
  /*uAddress=uAddress*2+0x1000;*/
  uTmp=uAddress & 0x000f;
  uSend[5]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
  uTmp=(uAddress>>4) & 0x000f;
  uSend[4]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
  uTmp=(uAddress>>8) & 0x000f;
  uSend[3]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
  uTmp=(uAddress>>12)&0x000f;
  uSend[2]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
  uSum=0;
```



```
for(i=1;i<9;i++)
    uSum=uSum+(unsigned char)uSend[i];
uTmp=uSum&0x000f;
uSend[10]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uSum>>4)&0x000f;
uSend[9]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
for(i=0;i<11;i++)
{ for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
    { if((inportb(STATS_PORT)&0x20)!=0)
        break;
    }
    if(lTmp>=DELAY_TIMES)
    {
        return(FALSE);
    }
    outportb(DATA_PORT,uSend[i]);
}
disable();

for(lTmp=0;lTmp<DELAY_TIMES;lTmp++)
{ if((inportb(STATS_PORT)&1)!=0)
    break;
}
if(lTmp>=DELAY_TIMES)
{
    enable();
    return(FALSE);
}
uReceive[0]=inportb(DATA_PORT);
if(uReceive[0]!=2)
{
    enable();
    return(FALSE);
}
for(i=1;i<number*4+4;i++)
{ for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
    { if((inportb(STATS_PORT)&1)!=0)
        break;
    }
    if(lTmp>=DELAY_TIMES)
    {
        enable();
        return(FALSE);
    }
}
```





```
    }
    uReceive[i]=inportb(DATA_PORT);
}
enable();
uSum=0;
for(i=1;i<number*4+2;i++)
    uSum=uSum+(unsigned int)uReceive[i];
uTmp=uSum&0xf;
uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

if((unsigned char)uTmp!=uReceive[number*4+3])return(FALSE);
uTmp=(uSum>>4)&0xf;
uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

if((unsigned char)uTmp!=uReceive[number*4+2])return(FALSE);
for(j=0;j<number;j++)
{
    for(i=j*4+1;i<j*4+5;i++)
        uReceive[i]=(uReceive[i]>0x39)?uReceive[i]-0x41+0xa:uReceive[i]-0x30;
uRead_value[j]=((((uReceive[j*4+3]<<4)+uReceive[j*4+4])<<4)+uReceive[j*4+1])<<4)+uReceive[j*4+2];
}
return TRUE;
}
//返回顶部

int write_data_register(unsigned int uAddress,unsigned int number)
{
    int i;
    for(i=0;i<3;i++)
        if(_write_data_register(uAddress,number)==TRUE)
            return TRUE;
    return FALSE;
}
//返回顶部

int _write_data_register(unsigned int uAddress,unsigned int number)
{
    unsigned char uSend[111];
    unsigned int uTmp,uSum,num;
    long lTmp;
    int i;
    if(TESTING==1)return TRUE;
```



```
init_plc();
uSend[0]=2;
uSend[1]=0x31;
uSend[number*4+8]=3;
num=(number*2)/16;
if(num>=10)uSend[6]=num+0x41-10;
else uSend[6]=num+0x30;
num=(number*2)%16;
if(num>=10)uSend[7]=num+0x41-10;
else uSend[7]=num+0x30;
uAddress=0x1000+2*uAddress;
uTmp=uAddress&0x000f;
uSend[5]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>4)&0x000f;
uSend[4]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>8)&0x000f;
uSend[3]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>12)&0x000f;
uSend[2]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
for(i=0;i<number;i++)
{
    uTmp=uWrite_value[i]&0x000f;
    uSend[i*4+9]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uWrite_value[i]>>4)&0x000f;
    uSend[i*4+8]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uWrite_value[i]>>8)&0x000f;
    uSend[i*4+11]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uWrite_value[i]>>12)&0x000f;
    uSend[i*4+10]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
}
uSum=0;
for(i=1;i<9+number*4;i++)
    uSum+=uSend[i];
uTmp=uSum&0x000f;
uSend[number*4+10]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uSum>>4)&0x000f;
uSend[number*4+9]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
for(i=0;i<11+number*4;i++)
{ for(!Tmp=0L;!Tmp<DELAY_TIMES;!Tmp++)
    { if((inportb(STATS_PORT)&0x20)!=0)
        break;
    }
    if(!Tmp>=DELAY_TIMES)
```



```
        {
            /*enable()*/
            return(FALSE);
        }
        outportb(DATA_PORT,uSend[i]);
    }
    disable();
    for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
    { if((inportb(STATS_PORT)&1)!=0)
        break;
    }
    if(lTmp>=DELAY_TIMES)
    {
        enable();
        return(FALSE);
    }
    if(inportb(DATA_PORT)!=6)
    {
        enable();
        return(FALSE);
    }
    else
    {
        enable();
        return(TRUE);
    }
}

int force_m_contact(unsigned uAddress,unsigned char ucOn_off)
{
    int i;
    for(i=0;i<3;i++)
        if(_force_m_contact(uAddress,ucOn_off)==TRUE)
            return TRUE;
    return FALSE;
}

//返回顶部

int _force_m_contact(unsigned uAddress,unsigned char ucOn_off)
{ unsigned uSend[]={2,0x37,0x30,0x30,0x30,0x30,3,0x30,0x30};
  unsigned uTmp,uSum,i;
```



```
long lTmp;
if(TESTING==1)return TRUE;
init_plc();
uAddress=uAddress+0x800;
uSend[1]=ucOn_off;
uTmp=uAddress&0x000f;
uSend[3]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>4)&0x000f;
uSend[2]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>8)&0x000f;
uSend[5]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uAddress>>12)&0x000f;
uSend[4]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uSum=0;
for(i=1;i<7;i++)
    uSum+=uSend[i];
uTmp=uSum&0x000f;
uSend[8]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
uTmp=(uSum>>4)&0x000f;
uSend[7]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
for(i=0;i<9;i++)
{ for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
    { if((inportb(STATS_PORT)&0x20)!=0)
        break;
    }
    if(lTmp>=DELAY_TIMES)
    {
        enable();
        return(FALSE);
    }
    outportb(DATA_PORT,uSend[i]);
}
disable();
for(lTmp=0L;lTmp<DELAY_TIMES;lTmp++)
{ if((inportb(STATS_PORT)&1)!=0)
    break;
}
if(lTmp>=DELAY_TIMES)
{
    enable();
    return(FALSE);
}
if(inportb(DATA_PORT)!=6)
{
```



```
        enable();
        return(FALSE);
    }
    else
    { enable();
      return(TRUE);
    }
}
//返回顶部

int read_m_register(unsigned int uAddress,unsigned int number)
{
    int i;
    for(i=0;i<3;i++)
        if(_read_m_register(uAddress,number)==TRUE)
            return TRUE;
    return FALSE;
}
//返回顶部

int _read_m_register(unsigned int uAddress,unsigned int number)
{ unsigned char uSend[]={2,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x32,3,0x30,0x30};
  unsigned char uReceive[54];
  unsigned int uTmp;
  unsigned int uSum;
  unsigned int num;
  long lTmp;
  int i,j;
  if(TESTING==1)
  { for(i=0;i<number;i++)uRead_value[i]=0;
    return TRUE;
  }
  init_plc();
  num=number;
  if((num/16)>=10)
      uSend[6]=(unsigned char)(num/16+0x41-10);
  else
      uSend[6]=(unsigned char)(num/16+0x30);
  if((num%16)>=10)
      uSend[7]=(unsigned char)((num%16)+0x41-10);
  else
```



```
    uSend[7]=(unsigned char)((num%16)+0x30);
    uAddress=uAddress/8+0x100;
    uTmp=uAddress & 0x000f;
    uSend[5]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>4) & 0x000f;
    uSend[4]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>8) & 0x000f;
    uSend[3]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uAddress>>12)&0x000f;
    uSend[2]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uSum=0;
    for(i=1;i<9;i++)
        uSum=uSum+(unsigned char)uSend[i];
    uTmp=uSum&0x000f;
    uSend[10]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);
    uTmp=(uSum>>4)&0x000f;
    uSend[9]=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

    for(i=0;i<11;i++)
    { for(!Tmp=0L;!Tmp<DELAY_TIMES;!Tmp++)
        { if((inportb(STATS_PORT)&0x20)!=0)
            break;

            }
        if(!Tmp>=DELAY_TIMES)
            {
                /*enable()*/
                return(FALSE);
            }
        outportb(DATA_PORT,uSend[i]);
    }
    disable();

    for(!Tmp=0;!Tmp<DELAY_TIMES;!Tmp++)
    { if((inportb(STATS_PORT)&1)!=0)
        break;
    }
    if(!Tmp>=DELAY_TIMES)
    {
        enable();
        return(FALSE);
    }
    uReceive[0]=inportb(DATA_PORT);
    if(uReceive[0]!=2)
```



```
{
    enable();
    return(FALSE);
}
for(i=1;i<number*2+4;i++)
{ for(!Tmp=0L;!Tmp<DELAY_TIMES;!Tmp++)
    { if((inportb(STATS_PORT)&1)!=0)
        break;
    }
    if(!Tmp>=DELAY_TIMES)
    {
        enable();
        return(FALSE);
    }
    uReceive[i]=inportb(DATA_PORT);
}
enable();
uSum=0;
for(i=1;i<number*2+2;i++)
    uSum=uSum+(unsigned int)uReceive[i];
uTmp=uSum&0xf;
uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

if((unsigned char)uTmp!=uReceive[number*2+3]) return(FALSE);
uTmp=(uSum>>4)&0xf;
uTmp=(uTmp<10)?(uTmp+0x30):(uTmp+0x41-0xa);

if((unsigned char)uTmp!=uReceive[number*2+2]) return(FALSE);
for(j=0;j<number;j++)
{
    for(i=j*2+1;i<j*2+3;i++)
        uReceive[i]=(uReceive[i]>0x39)?uReceive[i]-0x41+0xa:uReceive[i]-0x30;
    uRead_value[j]=((uReceive[j*2+1]<<4)+uReceive[j*2+2]);
}
return TRUE;
}
//返回顶部
```

## FX 系列 PLC 编程口通信协议 II



## 1、DEVICE READ (读出软设备状态值)

计算机向 PLC 发送:

始 命令 首地址 位数 终 和校验

STX CMD GROUP ADDRESS BYTES ETX SUM

例子: 从 D123 开始读取 4 个字节数据

02h 30h 31h,30h,46h,36h 30h,34h 03h 37h,34h

地址算法:  $address = address * 2 + 1000h$

再转换成 ASCII

31h,30h,46h,36h

PLC 返回

STX 1ST DATA 2ND DATA ..... LAST DATA ETX SUM

注: 最多可以读取 64 个字节的数据

例子: 从指定的存储器单元读到 3584 这个数据

02h 33h 35h 38h 34h 03h 44h,36h

## 2、DEVICE WRITE (向 PLC 软设备写入值)

始 命令 首地址 位数 数据 终 和校验

STX CMD GROUP ADDRESS BYTES 1ST DATA 2ND DATA ..... LAST DATA ETX SUM

例子: 向 D123 开始的两个存储器中写入 1234,ABCD

02h 31h 31h,30h,46h,36h 30h,34h 33h,34h,31h,32h,43h,44h,41h,42h 03h 34h,39h

PLC 返回

ACK (06H) 接受正确

NAK (15H) 接受错误





3、位设备强制置位/复位

FORCE ON 置位

始命令 地址 终 和校验

STX CMD ADDRESS ETX SUM

02h 37h address 03h sum

FORCE OFF 复位

始命令 地址 终 和校验

STX CMD ADDRESS ETX SUM

02h 38h address 03h sum

PLC 返回

ACK(06H) 接受正确

NAK(15H) 接受错误

设备强制中的地址公式:  $Address = Address / 8 + 100h$

说明:

1. 帧中的 BYTES 表示需要读取或者写入的字节数。
2. 地址算法上有说明。
3. 累加和是从 STX 后面一个字节开始累加到 ETX 的和。