GNetPlus® Communication Protocol

TM970013 REV.D

July 22, 2009

Basic Package (BINARY VERSION)

Master Query Package (HOST)

Field	Header	Address	Query Function	Data length	DATA BYTES	Error Check	
Desc	SOH	0~255	0~255	0~255		CRC16_Low	CRC16_Hi
Size	1 BYTE	1 BYTE	1 BYTE	1 BYTE	0~255 BYTES	1 BYTE	1 BYTE

Note:

SOH = 01h.

Address = Device Address (Slave Machine ID)

Slave Response Package (DEVICE)

Field	Header	Address	Response Function	Data length	DATA BYTES	Error Check	
Desc	SOH	0~255	ACK / NAK / EVN	0~255		CRC16_Low	CRC16_High
Size	1 BYTE	1 BYTE	1 BYTE	1 BYTE	0~255 BYTES	1 BYTE	1 BYTE

Note:

SOH = 01h.

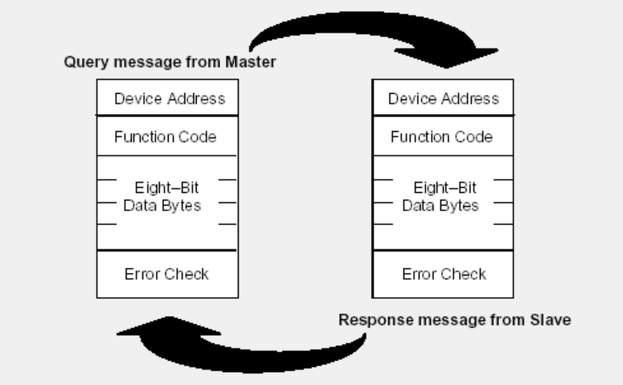
Address = Device Address (Slave Machine ID)

ACK = 06h, Acknowledge (Passive, in response to Master message)

NAK = 15h, Negative Acknowledge (Passive, in response to Master message)

EVN = 12h, Event Message(Active, For One Host to One Device Connection)

The Query - Response Cycle (Passive)



The Query

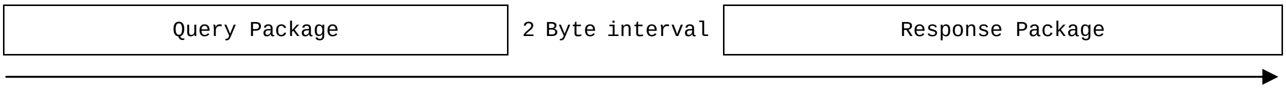
The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain all additional information that the slave will need to perform the function. The error check field provides a method for the slave to validate the integrity of the message contents.

The Response

If the slave makes a normal response, the function code in the response is an ACK of the function code in the response. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is NAK to indicate that the response is an error response, and the data bytes contain a code that describes the error. The error check field allows the master to confirm that the message contents are valid.

Avoiding Collision And Synchronizing (optional)

1. Send Package: Must have a 2 byte interval (by baudrate) after end of receiving.
2. Receive Package: If package is cut off (unfinished) and exceeds the interval, receiver can discard and reset the unfinished package buffer.



Byte sequence for data type INTEGER or LONG in "DATA BYTES" field

Example for data type integer 261Bh (2 BYTES)

SOH	Address	Function	Data Length	(MSB) Data Bytes (LSB)	
			2	26h	1Bh

Example for data type long 261B3C27h (4 BYTES)

SOH	Address	Function	Data Length	(MSB) Data Bytes (LSB)			
			4	26h	1Bh	3Ch	27h

Example for data structure:

Structure _example

Byte	bCode = 2Ch;
Int	nValue = 261Bh;
Long	lValue = 261B3C27h;

SOH	Address	Function	Data Length	(MSB) Data Bytes (LSB)						
				bCode	nValue		lValue			
			7	2Ch	26h	1Bh	26h	1Bh	3Ch	27h

Generating a CRC

Step 1 Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.

Step 2 Exclusive OR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.

Step 3 Shift the CRC register one bit to the right (toward the LSB), zero filling the MSB. Extract and examine the LSB.

Step 4 If the LSB is 0, repeat Step 3 (another shift). If the LSB is 1, Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).

Step 5 Repeat Steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.

Step 6 Repeat Steps 2 ... 5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.

Result The final contents of the CRC register is the CRC value.

Step 7 When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

Note: GNetPlus CRC16 is for fields from "Address" to "Data Bytes".

Placing the CRC into the communication package

When the 16-bit CRC (two eight-bit bytes) is transmitted in the message, **the low order byte will be transmitted first**, followed by the high order byte:

CRC16 Generation Function (C Code)

```
unsigned short GNetPlusCRC16(unsigned char *pBuffer, int iDataLen)
{
    const CRC_PRESET=0xFFFF;
    const CRC_POLYNOM=0xA001;
    unsigned short nCRC16=CRC_PRESET;
    char i;
    while( iDataLen-- )
    {
        nCRC16 ^= *pBuffer;
        pBuffer++;
        for( i=0; i<8; i++)
        {
            if( (nCRC16 & 1)==1 )
                nCRC16 = (nCRC16>>1) ^ CRC_POLYNOM;
            else
                nCRC16 = (nCRC16>>1);
        }
    }
    return nCRC16;
}
```

Common Query Function Code Table (00h~1Fh)

	Query (**Master**)			Response (**Slave**)		
Desc	**Func**	**Len**	**Data Bytes**	**Func**	**Len**	**Data Bytes**
Polling	00h	0		ACK	n	Return OEM Status
Get Version	01h	0		ACK	n	Return OEM Version String
Set Slave Addr	02h	1	New Address (1~255)	ACK	0	
		5	Addr (byte) + SN (long)	ACK	1	Return New Addr
Logon	03h	n	OEM Password	ACK	0	
Logoff	04h	0		ACK	0	
Set Password	05h	n	OEM New Password	ACK	0	
Class Name	**06h**	**0**		**ACK**	**n**	**Return Class Name**
Set Date/Time	07h	7	GNET_DATETIME Structure[1]	ACK	0	
Get Date/Time	08h	0		ACK	7	GNET_DATETIME Structure
Get Register	09h	3	Reg.Address[2] + Reg.Len[3]	ACK	n	Reg.Block
Set Register	0Ah	n	Reg.Address + Reg.Buffer	ACK	0	
Record Count	0Bh	0		ACK	2	Record Count (Integer)
Get First Record	0Ch	0		ACK	n	First Record
Get Next Record	0Dh	0		ACK	n	Next Record
Erase All Records	0Eh	0		ACK	0	
Add Record	0Fh	n	Record	ACK	0	
Recover All Records	10h	0		ACK	0	
DO	11h	2	DO# + STATUS(0 or 1)	ACK	0	
DI	12h	1	DI#	ACK	1	DI STATUS
Analog Input	**13h**	**1**	**Channel#**	**ACK**	**2**	**Value (Integer)**
Thermometer	**14h**	**0**		**ACK**	**2**	**Value (Integer)**
Get Node	15h	0		ACK	n	Get Record with Remove
Get S/N	16h	0		ACK	4	SN
Silent Mode	17h	1	ON(1)/OFF(0)	ACK	1	ON(1) / OFF(0)
		2	ON(1)/OFF(0)+except Addr.	ACK	1	ON(1) / OFF(0)
Reserve	18h					
Enable Auto Mode	**19h**	**1**	**Boolean**	**ACK**	**1**	**Result**
Get Time Adjust^{OEM}	1Ah	0		ACK	4	Time Adjust Value (long)
Echo	1Bh	N	Any Data	ACK	n	Master Data
Set Time Adjust^{OEM}	1Ch	4	Time Adjust Value (long)	ACK	0	
Debug	1Dh	0		ACK	n	OEM Debug Message
Reset	1Eh	0		ACK	0	
Go To ISP	1Fh	0		ACK	0	

Note:

1. GNET_DATETIME Structure (7 BYTES)

```
BYTE Second;
BYTE Minute;
BYTE Hour;
BYTE DayOfWeek;
BYTE Day;
BYTE Month;
BYTE Year;
```

2. Reg.Address: Integer Type(2 Bytes), Range 0000h~FFFFh.
3. Reg.Len : Register Block Size (1 Byte), Max.255.

Response NAK Code Table (Common)

Func	Len	Data Bytes	Description
NAK	1	E0h	Access Denied
NAK	1	E4h	Illegal Query Code
NAK	1	E6h	Overrun, Out of record count
NAK	1	E7h	CRC Error
NAK	1	ECh	Query Number no support
NAK	1	EDh	Out Of Memory Range
NAK	1	EEh	Address Number out of range
NAK	1	EFh	Unknown

Response Event (For Active Slave)

	Active Response (**Slave**)		
Desc	**Func**	**Len**	**Data Bytes**
Event	12h	n	Customer Event Code or Data