UM4001 GUTTA 通讯协议

2008/11/15

概述

GUTTA 通讯协议分为 3 大部分内容：PLC 控制协议、PLC 传输协议、PLC 状态协议。控制协议用于 PLC 的基本控制；例如登陆、登出、复位、清除等等。传输协议用于程序的上传和下载；状态协议用于 PLC 程序的在线监控和调试。

由于 PLC 的实现方式不同（编译和解释）、CPU 性能和 FLASH 容量的不同、通讯方式的不同（RS243、RS485、USB、CAN 等）、功能的不同（是否支持调试），特定的 PLC 系统可以部分实现上述协议。对于特定的 PLC 类型，GUTTA Ladder Editor 软件需要知道目标 PLC 支持那些通讯协议，从而判断当前的通讯任务是否能够完成。GUTTA 软件在启动的时候，在载入 *ManagerFun.xml* 和 *ManagerVar.xml* 之前，先要载入 *PlcType.xml* 文件。这个文件是对当前 PLC 类型的一个总体描述。这个文件位于 PLC 类型文件夹下（可参考《UM4002 变量描述文件规范》、《UM4003 指令描述文件规范》）。一个典型的 *PlcType.xml* 文件看起来是这个样子的：

```
<?xml version="1.0" encoding="utf-16"?>
<PlcType
  Name="CPU-EC20"
  Information="CPU-EC20"
  SystemBlockDll="SystemBlockDll.dll"
  SystemBlockBinarySize="50"
  DataBlockPageSize="16"
  DataBlockPageItemSize="16"
  ProgramBlockPageIntSize="8"
  ProgramBlockPageSbrSize="8"
  ProgramBlockPageArgIntSize="32"
  ProgramBlockPageArgSbrSize="32"
  ProgramBlockConstBinarySize="128"
  ProgramBlockInstructionBinarySize="10752"
  CommunicationDll="CommunicationDll.dll"
  ExchPackSize="64"
  ExchSupport="0000|0000"
  >
</PlcType>
```

<PlcType>: Name 属性

数据类型为 STRING，长度必须大于 0 且小于或等于 16。

节点 *PlcType* 的 *Name* 属性描述当前 PLC 类型的名称。这个名称是 GUTTA 软件识别当前 PLC 类型的唯一依据。（不论是打开文件还是上传程序，GUTTA 软件都要先确定这个属性的值，然后根据此值载入对应的 PLC 配置）

<PlcType>: Information 属性

数据类型为 STRING，长度必须大于 0 且小于或等于 64。

节点 *PlcType* 的 *Information* 属性是对当前 PLC 类型名称的一个更详细描述。这个描述信息仅仅只是对名称的一个补充说明，不参与 PLC 类型的识别。

<PlcType>: SystemBlockDll 属性

数据类型为 STRING，不能为空。

节点 *PlcType* 的 *SystemBlockDll* 属性告诉 GUTTA 软件如果需要编辑 PLC 系统块，应该载入动态连接库的文件名。

<PlcType>: SystemBlockBinarySize 属性

数据类型为 INT。

节点 *PlcType* 的 *SystemBlockBinarySize* 属性描述 PLC 目标硬件允许的最大系统块大小（字节为单位）。

<PlcType>: DataBlockPageSize 属性

数据类型为 INT。

节点 *PlcType* 的 *DataBlockPageSize* 属性描述 PLC 目标硬件允许的最大数**据块页**数量。

<PlcType>: DataBlockPageItemSize 属性

数据类型为 INT。

节点 *PlcType* 的 *DataBlockPageItemSize* 属性描述 PLC 目标硬件允许的最大数据块单页**数据项**数量。

由于一个数据项是 8 个字节，故数据块的总大小为（字节为单位）：

$$DataBlockBinarySize = DataBlockPageSize \times DataBlockPageItemSize \times 8$$

<PlcType>: ProgramBlockPageIntSize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockPageIntSize* 属性描述 PLC 目标硬件允许的最大中断调用数量。

<PlcType>: ProgramBlockPageSbrSize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockPageSbrSize* 属性描述 PLC 目标硬件允许的最大函数调用数量。

<PlcType>: ProgramBlockPageArgIntSize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockPageArgIntSize* 属性描述 PLC 目标硬件允许的最大中断调用参数个数。

<PlcType>: ProgramBlockPageArgSbrSize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockPageArgSbrSize* 属性描述 PLC 目标硬件允许的最大函数调用参数个数。

<PlcType>: ProgramBlockConstBinarySize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockConstBinarySize* 属性描述 PLC 目标硬件允许的最大常数块大小（字节为单位）。

<PlcType>: ProgramBlockInstructionBinarySize 属性

数据类型为 INT。

节点 *PlcType* 的 *ProgramBlockInstructionBinarySize* 属性描述 PLC 目标硬件允许的最大指令块大小（字节为单位）。

<PlcType>: CommunicationDll 属性

数据类型为 STRING，不能为空。

节点 *PlcType* 的 *CommunicationDll* 属性告诉 GUTTA 软件如果需要进行上传或下载，应该载入动态连接库的文件名。

<PlcType>: ExchPackSize 属性

数据类型为 INT，必须大于或等于 64 且小于 1024。

节点 *PlcType* 的 *ExchPackSize* 属性描述一个数据包的最大有效数据长度。由于这个长度并不包含 GUTTA 封包附加数据和 MODBUS 封包附加数据，实际的硬件缓冲应该比这个长度略大。在传送比这个值要长的

<PlcType>: ExchSupport 属性

数据类型为 STRING。格式为“*M1|V1|M2|V2|...|Mn|Vn*”

节点 *PlcType* 的 *ExchSupport* 属性描述当前 PLC 类型支持的通讯指令。*Mn* 和 *Vn* 都用 16 进制的数表达。某条通讯指令是否被当前 PLC 支持，可以用下面的公式表示：

ExchSupport=((ExchID&M1)==V1)||((ExchID&M2)==V2)||...||((ExchID&Mn)==Vn)

由于任何数与上 16#0000 都等于 16#0000，故上面给出的例子表示 CPU-EC20 支持全部的通讯指令。

封包结构

GUTTA 通讯协议由 MODBUS 协议封包。在解析本协议前，必须先解析 MODBUS 协议。在进行 MODBUS 通讯时，PLC 编程软件 GUTTA Ladder Editor 控制的主机端口为 MODBUS 主站；PLC 硬件端口是 MODBUS 从站。通讯由主站发起，子站在接受到主站数据后，必须在指定的时间内向主站发送响应数据。至此，一个通讯来回完成。

GUTTA 通讯数据由 MODBUS 的 13 号功能封装：

发送数据：

站号	功能码	GUTTA 通讯数据	CRC 校验	
N	13			

返回数据：

站号	功能码	GUTTA 通讯数据	CRC 校验	
N	13			

GUTTA 通讯数据长度是可变的，其格式如下：

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		通讯数据

返回数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		通讯数据

数据长度、**功能码**、**内部包号**都是两字节变量，范围是 0H ~ FFFFH。在通讯的时候，这三个变量都是先传送高位字节，然后传送低位字节。数据长度是功能码、内部包号、通讯数据三者长度之和，以字节计。功能码在后面有详细说明。当通讯数据比较大时，需要分包传送。为了统一，不论是单包数据还是多包数据，都有内部包号，包号从 0 开始，若是最后一包数据，需要将内部包号的最高置为 1。

若 PLC 执行成功，返回功能码为发送数据的功能码，否则，返回数据功能码为发送数据的功能码最高位置 1 后的值。

功能码参考

PLC 控制协议

<0100H> PLC_CLEAR 清除 PLC

PLC_CLEAR 指令用于清除 PLC 用户程序（包括密码）。一旦执行 PLC_CLEAR 指令，所有的 PLC 用户程序将被丢失，PLC 被停止。由于 FLASH 的用户程序被清除，这个时候 PLC 不能直接被运行(PLC 某些状态和 FLASH 数据不匹配)，必须通过执行 STATUS_RESET 使 PLC 回到出厂状态。通讯端口也恢复成 19200 波特率、偶校验、1 位停止位。

PLC 一般通过 PLC_CLEAR 解除用户 FLASH 的锁定。通过 STATUS_RESET 恢复用户 FLASH 的锁定。故程序的下载必须发生在 PLC_CLEAR 和 STATUS_RESET 这两条指令之间。考虑到用户忘记 PLC 密码的情况，PLC_CLEAR 指令的执行不用登陆到 PLC。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	00H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	81H	00H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	00H	80H	00H

返回数据（成功）的条件：

- 发送数据长度为 0。

返回数据（成功），PLC 执行的操作：

- 解除 PLC 用户 FLASH 锁定。
- 擦除用户 FLASH。
- 停止 PLC。

<0110H> PLC_ATTACH 连接 PLC

PLC_ATTACH 指令用于连接到 PLC。连接到 PLC 需要提供一个 16 位的登陆密码，PLC 根据登陆密码决定是否接受当前连接（登陆密码由上一次下载程序的系统页来设置，出厂值为空密码，即 16 个连续的 16#FF）。若密码正确，PLC 进入被登陆状态。绝大部分通讯指令只有在 PLC 被登陆后才能被执行。PLC 记录当前的登陆状态直到下列任一情况发生：PLC_DETACH 指令被执行、PLC 被断电、PLC 被复位。

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		登陆密码（16 字节）
00H	14H	01H	10H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	81H	10H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	10H	80H	00H

返回数据（成功）的条件：

- 发送数据长度为 16。
- 发送数据和系统页的登陆密码一致。

返回数据（成功），PLC 执行的操作：

- 设置 PLC 状态的 ATTACH 位。

<0111H> PLC_DETACH 断开 PLC

PLC_DETACH 指令用于断开当前 PLC 的连接。PLC 清除当前的登陆状态。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	11H	80H	00H

返回数据（失败）

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	81H	11H	80H	00H

返回数据（成功）的条件：

- 发送数据长度为 0。

返回数据（成功）的操作：

- 复位 PLC 状态的 ATTACH 位。

<0120H> GET_PLC_NAME 获得 PLC 名称

GET_PLC_NAME 指令获得目标硬件的 PLC 类型名称。这个名称和*<PlcType>:Name* 属性对应。若 PLC 类型名称不足 16 个字节，以字节 0 标志结束。

GUTTA 软件在上载程序前必须通过通讯确认 PLC 类型名称。然后在 PLC 类型库中载入对应的 *ManagerVar.xml*、*ManagerFun.xml*、*SystemBlockDll.dll*、*CommunicationDll.dll* 这四个文件。其中后面的两个动态连接库文件可以被 *PlcType.xml* 另外指定。若 GUTTA 软件找不到对应的 PLC 类型，将会终止当前上载操作。

GUTTA 软件在下载程序前必须通过通讯确认 PLC 类型名称。如果硬件的 PLC 类型名称和当前用户程序的 PLC 类型名一致，则开始下载，否则会终止当前下载操作。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	20H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	81H	20H	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		PLC 名称（16 字节）
00H	14H	01H	20H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0121H> GET_PLC_INFOR 获得 PLC 信息

GET_PLC_INFOR 指令获得目标硬件的 PLC 类型信息。这个信息和*<PlcType>:Information* 属性对应。这个描述信息仅仅只是对名称的一个补充说明，不参与 PLC 类型的识别。若 PLC 类型信息不足 64 个字节，以字节 0 标志结束。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	01H	21H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	81H	21H	80H	00H

返回数据（成功）：

GUTTA 通讯数据

数据长度		功能码		内部包号		PLC 信息（64 字节）
00H	44H	01H	21H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

PLC 传输协议

<0210H> DATA_ASK 查询 PLC 有效数据页

DATA_ASK 指令查询目标有效数据页的页号。每个数据页页号为一个字节，返回数据的实际长度根据有效数据页的数量决定。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	02H	10H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	82H	10H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	页号序列（n 字节）
n+4		02H	10H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<022XH> DATA_PAGE_ASK 查询 PLC 数据页长度

DATA_PAGE_ASK 指令读取指定数据页的长度。数据页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0220H~022FH，即最多指定 16 个数据页。返回数据为指定数据页的长度（字变量），传送时高位在前，低位在后。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	02H	2XH	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	

00H	04H	82H	2XH	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		数据页 X 的长度（2 字节）
00H	06H	02H	2XH	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<023XH> DATA_PAGE_READ 读 PLC 指定数据页

DATA_PAGE_READ 指令读取数据页数据。数据页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0230H~023FH，即最多指定 16 个数据页。返回数据为数据页的数据。GUTTA 软件先通过 DATA_PAGE_ASK 指令获得数据页的数据长度。若数据页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次读取部分数据页数据的方式来完成整个数据页数据的传输。每次读取数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。PLC 返回从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	02H	3XH	i

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	82H	3XH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号	数据页数据（n 字节）
n+4		02H	3XH	i	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<024XH> DATA_PAGE_WRITE 写 PLC 指定数据页

DATA_PAGE_WRITE 指令写入数据页数据。数据页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0240H~024FH，即最多指定 16 个数据页。发送数据为数据页的数据。若需要发送数据页的数据长度大于最大通讯数据长度，GUTTA 软件会

采取多次写入部分数据页数据的方式来完成整个数据页数据的传输。每次写入数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。软件写入从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度	功能码		内部包号	数据页数据（n 字节）
n+4	02H	4XH	i	

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	82H	4XH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	02H	4XH	i

返回数据（成功）的条件：

- PLC 被登陆。
- PLC 不运行。

返回数据（成功），PLC 执行的操作：

- 写入数据页数据。

<0260H> SYSTEM_ASK 查询 PLC 有效系统页

SYSTEM_ASK 指令查询目标有效系统页的页号。每个系统页页号为一个字节，返回数据的实际长度根据有效系统页的数量决定。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	02H	60H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	82H	60H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度	功能码		内部包号		页号序列（n 字节）
n+4	02H	60H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<027XH> SYSTEM_PAGE_ASK 查询 PLC 系统页长度

SYSTEM_PAGE_ASK 指令读取指定系统页的长度。系统页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0270H~027FH，即最多指定 16 个系统页。返回数据为指定系统页的长度（字变量），传送时高位在前，低位在后。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	02H	7XH	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	82H	7XH	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		系统页 X 的长度（2 字节）
00H	06H	02H	7XH	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<028XH> SYSTEM_PAGE_READ 读 PLC 指定系统页

SYSTEM_PAGE_READ 指令读取系统页数据。系统页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0280H~028FH，即最多指定 16 个系统页。返回数据为系统页的数据。GUTTA 软件先通过 SYSTEM_PAGE_ASK 指令获得系统页的数据长度。若系统页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次读取部分系统页数据的方式来完成整个系统页数据的传输。每次读取数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。PLC 返回从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	02H	8XH	i

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	82H	8XH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度	功能码		内部包号	系统页数据 （n 字节）
n+4	02H	8XH	i	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<029XH> SYSTEM_PAGE_WRITE 写 PLC 指定系统页

SYSTEM_PAGE_WRITE 指令写入系统页数据。系统页页号包含在功能码中，由功能码的最低 4 位确定页号。因此实际的功能码为 0290H~029FH，即最多指定 16 个系统页。发送数据为系统页的数据。若需要发送系统页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次写入部分系统页数据的方式来完成整个系统页数据的传输。每次写入数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。软件写入从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度	功能码		内部包号	系统页数据 （n 字节）
n+4	02H	9XH	i	

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	82H	9XH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	02H	9XH	i

返回数据（成功）的条件：

- PLC 被登陆。
- PLC 不运行。

返回数据（成功），PLC 执行的操作：

- 写入系统页数据。

<0300H> PROGRAM_ASK 查询 PLC 有效程序页

PROGRAM_ASK 指令查询目标有效程序页的页号。每个程序页页号为一个字节，返回数据的实际长度根据有效程序页的数量决定。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	03H	00H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	83H	00H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	页号序列（n 字节）
n+4		03H	00H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0310H> PROGRAM_CONST_ASK 查询 PLC 常数页长度

PROGRAM_CONST_ASK 指令读取常数页的长度。返回数据为常数页的长度(字变量)，传送时高位在前，低位在后。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	03H	10H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	83H	10H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	常数页的长度 （2 字节）
00H	06H	03H	10H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0311H> PROGRAM_CONST_READ 读 PLC 指定常数页

PROGRAM_CONST_READ 指令读取常数页数据。GUTTA 软件先通过 PROGRAM_CONST_ASK 指令获得常数页的数据长度。若常数页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次读取部分常数页数据的方式来完成整个常数页数据的传

输。每次读取数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。PLC 返回从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	03H	11H	i

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	83H	11H	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度	功能码		内部包号	常数页数据 （n 字节）
n+4	03H	11H	i	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0312H> PROGRAM_CONST_WRITE 写 PLC 指定常数页

PROGRAM_CONST_WRITE 指令写入常数页数据。若需要发送常数页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次写入部分常数页数据的方式来完成整个常数页数据的传输。每次写入数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。软件写入从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度	功能码		内部包号	常数页数据 （n 字节）
n+4	03H	12H	i	

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	83H	12H	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	03H	12H	i

返回数据（成功）的条件：

- PLC 被登陆。

- PLC 不运行。

返回数据（成功），PLC 执行的操作：

- 写入常数页数据。

<04XXH> PROGRAM_ARGUMENT_ASK 查询 PLC 参数页长度

PROGRAM_ARGUMENT_ASK 指令读取指定参数页的长度。参数页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0400H~04FFH，即最多指定 256 个参数页。返回数据为指定参数页的长度（字变量），传送时高位在前，低位在后。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	04H	XXH	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	84H	XXH	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		参数页 XX 的长度（2 字节）
00H	06H	04H	XXH	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<05XXH> PROGRAM_ARGUMENT_READ 读 PLC 指定参数页

PROGRAM_ARGUMENT_READ 指令读取参数页数据。参数页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0500H~05FFH，即最多指定 256 个参数页。返回数据为参数页的数据。GUTTA 软件先通过 PROGRAM_ARGUMENT_ASK 指令获得参数页的数据长度。若参数页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次读取部分参数页数据的方式来完成整个参数页数据的传输。每次读取数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。PLC 返回从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	05H	XXH	i

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	85H	XXH	i

返回数据（成功）：

GUTTA 通讯数据			
数据长度	功能码		内部包号	参数页数据（n 字节）
n+4	05H	XXH	i	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<06XXH> PROGRAM_ARGUMENT_WRITE 写 PLC 指定参数页

PROGRAM_ARGUMENT_WRITE 指令写入参数页数据。参数页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0600H~06FFH，即最多指定 256 个参数页。发送数据为参数页的数据。若需要发送参数页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次写入部分参数页数据的方式来完成整个参数页数据的传输。每次写入数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。软件写入从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度	功能码		内部包号	参数页数据（n 字节）
n+4	06H	XXH	i	

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	86H	XXH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	06H	XXH	i

返回数据（成功）的条件：

- PLC 被登陆。
- PLC 不运行。

返回数据（成功），PLC 执行的操作：

- 写入参数页数据。

<07XXH> PROGRAM_INSTRUCTION_ASK 查询 PLC 程序页长度

PROGRAM_INSTRUCTION_ASK 指令读取指定程序页的长度。程序页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0700H~07FFH，即最多指定 256 个程序页。返回数据为指定程序页的长度（字变量），传送时高位在前，低位在后。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	07H	XXH	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	87H	XXH	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		程序页 XX 的长度（2 字节）
00H	06H	07H	XXH	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<08XXH> PROGRAM_INSTRUCTION_READ 读 PLC 指定程序页

PROGRAM_INSTRUCTION_READ 指令读取程序页数据。程序页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0800H~08FFH，即最多指定 256 个程序页。返回数据为程序页的数据。GUTTA 软件先通过 PROGRAM_INSTRUCTION_ASK 指令获得程序页的数据长度。若程序页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次读取部分程序页数据的方式来完成整个程序页数据的传输。每次读取数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。PLC 返回从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	08H	XXH	i

返回数据（失败）：

GUTTA 通讯数据		
数据长度	功能码	内部包号

00H	04H	88H	XXH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码	内部包号	程序页数据 （n 字节）
n+4		08H XXH	i	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<09XXH> PROGRAM_INSTRUCTION_WRITE 写 PLC 指定程序页

PROGRAM_INSTRUCTION_WRITE 指令写入程序页数据。程序页页号包含在功能码中，由功能码的最低 8 位确定页号。因此实际的功能码为 0900H~09FFH，即最多指定 256 个程序页。发送数据为程序页的数据。若需要发送程序页的数据长度大于最大通讯数据长度，GUTTA 软件会采取多次写入部分程序页数据的方式来完成整个程序页数据的传输。每次写入数据的起始地址由内部包号决定，即内部包号乘以最大通讯数据长度。软件写入从起始地址开始的对应数据，直到数据结束或者达到最大通讯数据长度。若为最后一包数据，GUTTA 软件会置位发送数据内部包号的最高位。

发送数据：

GUTTA 通讯数据				
数据长度		功能码	内部包号	程序页数据 （n 字节）
n+4		09H XXH	i	

返回数据（失败）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	89H	XXH	i

返回数据（成功）：

GUTTA 通讯数据				
数据长度		功能码		内部包号
00H	04H	09H	XXH	i

返回数据（成功）的条件：

- PLC 被登陆。
- PLC 不运行。

返回数据（成功），PLC 执行的操作：

- 写入程序页数据。

PLC 状态协议

<0A00H> STATUS_READ 读 PLC 状态

STATUS_READ 指令读取当前 PLC 的状态。PLC 的状态为一个 8 位的字节：

SYS_STATE								
8	7	6	5	4	3	2	1	0
					ERROR	ATTACH	RESET	RUN

- RUN 表示 PLC 是否在运行状态。
- RESET 表示 PLC 是否存在复位请求。
- ATTACH 表示 PLC 是否被登陆。
- ERROR 表示 PLC 是否发生错误。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	00H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	00H	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		**SYS_STATE**（1 字节）
00H	05H	0AH	00H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0A01H> STATUS_WRITE 写 PLC 状态

STATUS_WRITE 指令修改当前 PLC 的状态的 RUN 位。STATUS_WRITE 指令只发送一个字节的数据。若 RUN 为 0，强制 PLC 进入停止状态。若 RUN 不为 0，强制 PLC 进入运行状态。

发送数据：

GUTTA 通讯数据			
数据长度	功能码	内部包号	**RUN**（1 字节）

00H	05H	0AH	01H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	01H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	01H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 1。

返回数据（成功），PLC 执行的操作：

- 若 RUN 为 0，强制 PLC 进入停止状态。若 RUN 不为 0，强制 PLC 进入运行状态。

<0A02H> STATUS_SCAN 强制扫描

STATUS_SCAN 指令使处于停止状态的 PLC 进行指定次数的主循环扫描。STATUS_SCAN 指令只发送一个字节的数据。这个数据代表主循环扫描的次数。

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		扫描次数（1 字节）
00H	05H	0AH	02H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	02H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	02H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 1。

返回数据（成功），PLC 执行的操作：

- 记录主循环扫描值（在扫描完成前返回通讯数据）。

<0A03H> STATUS_RESET 强制复位

STATUS_RESET 指令使 PLC 记录复位请求。当 PLC 发现有复位请求且可以复位时，执行复位操作。这个复位并不对 CPU 外设进行硬件复位，而是根据用户 FLASH 程序对外设

硬件进行重新配置（例如通讯口），并且从新初始化 PLC 需要使用的内存。STATUS_RESET 指令一般和 PLC_CLEAR 指令成对使用（参考 PLC_CLEAR 指令）。考虑到用户忘记 PLC 密码的情况，STATUS_RESET 指令的执行不用登陆到 PLC。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	03H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	03H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	03H	80H	00H

返回数据（成功）的条件：

- 发送数据长度为 0。

返回数据（成功），PLC 执行的操作：

- 恢复 PLC 用户 FLASH 锁定。
- 设置 PLC 状态的 RESET 位（在复位完成前返回通讯数据）。

<0A10H> STATUS_VAR_READ 读变量值

STATUS_VAR_READ 指令读取 PLC 变量的值，这个值用于 GUTTA 编程软件在线调试的数据显示。发送数据为需要读变量的地址，返回数据为对应地址上变量的值。为了减少通讯次数，STATUS_VAR_READ 一次可以读多个变量的值，发送数据为读变量地址的序列。返回数据为对应地址上变量值的序列。由于变量地址为 32 位，返回的变量值也是 32 位，故返回数据的长度总是等于发送数据长度。

发送数据：

GUTTA 通讯数据					
数据长度	功能码		内部包号		地址序列（4n 字节）
4n+4	0AH	10H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	10H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度	功能码		内部包号		值序列（4n 字节）
4n+4	0AH	10H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。

- 发送数据能够被 4 整除。

地址的格式：

地址（高 16 位）															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET															

地址（低 16 位）															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIT_OFFSET				WIDTH				SLOT				USE			

- USE 表示变量的用法。
- SLOT 表示变量的区域。
- WIDTH 表示变量的宽度。
- BIT_OFFSET 表示变量的位偏移。
- OFFSET 表示变量的偏移。

发送时 4 字节的地址数据按照**低字节到高字节**的顺序发送。其中 USE、WIDTH、BIT_OFFSET 字段被忽略。PLC 硬件根据 SLOT 和 OFFSET 确定变量所在的第一个字节，然后将第一个字节开始的 4 个字节作为值返回。

返回时 4 字节的值按照低字节到高字节的顺序返回。

<0A11H> STATUS_VAR_WRITE 写变量值

STATUS_VAR_WRITE 指令写入 PLC 变量的值。发送数据为地址和值的数据对。STATUS_VAR_WRITE 可以一次写入多个变量的值，发送数据为地址和值的序列。

发送数据：

GUTTA 通讯数据					
数据长度	功能码		内部包号		地址和值的序列（8n 字节）
8n+4	0AH	11H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	11H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	11H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据能够被 8 整除。

返回数据（成功），PLC 执行的操作：

- 修改指定变量的值。（对于强制的变量，还要修改它的强制为值）

地址和值的序列：

地址和值序列							
地址	值	地址	值	地址	值	地址	值

地址的格式：

地址（高 16 位）															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET															

地址（低 16 位）															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIT_OFFSET				WIDTH				SLOT				USE			

- USE 表示变量的用法。
- SLOT 表示变量的区域。
- WIDTH 表示变量的宽度。
- BIT_OFFSET 表示变量的位偏移。
- OFFSET 表示变量的偏移。

发送数据时，4 字节的地址数据和 4 字节的值数据都是按照**低字节到高字节**的顺序发送。其中 USE 字段被忽略。PLC 根据 SLOT、WIDTH、BIT_OFFSET、OFFSET 确定需要修改 PLC 内存的位置和长度，然后根据要求将值中的数据复制到 PLC 内存中。

<0A20H> STATUS_VAR_FORCE_READ 读变量强制值

STATUS_VAR_FORCE_READ 指令读取 PLC 变量的强制值，这个强制值用于 GUTTA 编程软件在线调试的数据显示。发送数据为需要读变量强制值的地址，返回数据为对应地址上变量的强制值。为了减少通讯次数，STATUS_VAR_FORCE_READ 一次可以读多个变量的强制值，发送数据为读变量地址的序列。返回数据为对应地址上变量强制值的序列。由于变量地址为 32 位，返回的变量强制值也是 32 位，故返回数据的长度总是等于发送数据长度。

发送数据：

GUTTA 通讯数据					
数据长度	功能码		内部包号		地址序列（4n 字节）
4n+4	0AH	20H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	20H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度	功能码		内部包号		强制值序列（4n 字节）
4n+4	0AH	20H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据能够被 4 整除。

地址和强制值的格式参照 STATUS_VAR_WRITE 指令。

<0A21H> STATUS_VAR_FORCE_WRITE 写变量强制值

STATUS_VAR_FORCE_WRITE 指令写入 PLC 变量的强制值。发送数据为地址和强制值的数据对。STATUS_VAR_FORCE_WRITE 可以一次写入多个变量的强制值，发送数据为地址和强制值的序列。

发送数据：

GUTTA 通讯数据					
数据长度	功能码		内部包号		地址和强制值的序列（8n 字节）
8n+4	0AH	21H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	21H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	21H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据能够被 8 整除。

返回数据（成功），PLC 执行的操作：

- 修改指定变量的强制值。

地址和强制值的格式参照 STATUS_VAR_WRITE 指令。

<0A30H> STATUS_DEBUG_READ 读 PLC 调试状态

STATUS_DEBUG_READ 指令读取当前 PLC 的调试状态。PLC 的调试状态为一个 8 位的字节：

DEBUG_STATE								
8	7	6	5	4	3	2	1	0
							RESET	RUN

- RUN 表示 PLC 是否在调试运行状态。
- RESET 表示 PLC 是否存在调试复位请求。

发送数据：

GUTTA 通讯数据

数据长度		功能码		内部包号	
00H	04H	0AH	30H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	30H	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		**DEBUG_STATE**（1 字节）
00H	05H	0AH	30H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

<0A31H> STATUS_DEBUG_WRITE 写 PLC 调试状态

STATUS_DEBUG_WRITE 指令修改当前 PLC 的调试状态的 RUN 位。STATUS_DEBUG_WRITE 指令只发送一个字节的数据。若 RUN 为 0，强制 PLC 进入单步调试的停止状态。若 RUN 不为 0，强制 PLC 进入单步调试的连续运行状态（可以被断点停止）。

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		**DEBUG_RUN**（1 字节）
00H	05H	0AH	31H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	31H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	31H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 1。

返回数据（成功），PLC 执行的操作：

- 若 RUN 为 0，强制 PLC 进入单步调试的停止状态。若 RUN 不为 0，强制 PLC 进入单步调试的连续运行状态。

<0A32H> STATUS_DEBUG_SCAN 调试模式扫描

STATUS_DEBUG_SCAN 指令使处于单步调试状态的 PLC 进行单步调试操作。STATUS_DEBUG_SCAN 指令只发送一个字节的数据。这个数据可以取下面的值。

- 16#01 STEP_IN：单步进入。
- 16#02 STEP_OVER：单步跳过。
- 16#03 STEP_OUT：单步跳出。

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		STEP（1 字节）
00H	05H	0AH	32H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	32H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	32H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 1。

返回数据（成功），PLC 执行的操作：

- 进行 STEP 对应的单步调试操作。

<0A33H> STATUS_DEBUG_RESET 调试模式复位

STATUS_DEBUG_RESET 指令使 PLC 记录调试复位请求。当 PLC 发现有调试复位请求且可以调试复位时，执行调试复位操作。相对于普通复位，调试复位不改变当前的调试状态。即复位后 PLC 不进入运行状态，而是进入调试单步停止状态。

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	33H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	33H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	33H	80H	00H

返回数据（成功）的条件：

- 发送数据长度为 0。

返回数据（成功），PLC 执行的操作：

- 设置 PLC 调试状态的 RESET 位（在调试复位完成前返回通讯数据）。

<0A34H> STATUS_DEBUG_INFOR_READ 读调试信息

STATUS_DEBUG_INFOR_READ 指令读取目标 PLC 的调试信息。

发送数据：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	34H	80H	00H

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	34H	80H	00H

返回数据（成功）：

GUTTA 通讯数据						
数据长度		功能码		内部包号		调试信息（16 字节）
00H	14H	0AH	34H	80H	00H	

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 0。

返回调试信息的格式：

0	BYTE	DEBUG_INDEX	调试断点的函数号
1 2	WORD	DEBUG_NETWORK	调试断点的网络号
3 4	WORD	DEBUG_STEP	调试断点的步号
5	BYTE	STATE_INDEX	当前的函数号
6 7	WORD	STATE_NETWORK	当前的网络号
8 9	WORD	STATE_STEP	当前的步号
10 11	WORD	STACK_DATA	数据栈值
12 13	WORD	STACK_LOGIC	辅助栈值
14	BYTE	SP	调用嵌套层次
15	BYTE	ERROR	错误代码

所有的调试信息字数据都按照高字节到低字节的顺序发送。

<0A35H> STATUS_DEBUG_INFOR_WRITE 写调试信息

STATUS_DEBUG_INFOR_WRITE 指令写入目标 PLC 的调试信息。

发送数据：

GUTTA 通讯数据						
数据长度		功能码		内部包号		调试信息（5 字节）
00H	09H	0AH	35H	80H	00H	

返回数据（失败）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	8AH	35H	80H	00H

返回数据（成功）：

GUTTA 通讯数据					
数据长度		功能码		内部包号	
00H	04H	0AH	35H	80H	00H

返回数据（成功）的条件：

- PLC 被登陆。
- 发送数据长度为 5。

返回数据（成功），PLC 执行的操作：

- 修改对应的 PLC 调试数据。

发送调试信息的格式：

0	BYTE	DEBUG_INDEX	调试断点的函数号
1	WORD	DEBUG_NETWORK	调试断点的网络号
2			
3	WORD	DEBUG_STEP	调试断点的步号
4			

所有的调试信息数据都按照高字节到低字节的顺序发送。