UM4002 变量描述文件规范

2008/11/15

概述

在 GUTTA 平台中，PLC 对内存的访问是按“区域”来进行的。例如：**I0.0** 表示离散量输入区域第 0 个字节的第 0 个位；**M30.2** 表示中间变量区域第 30 个字节的第 2 个位；**MW20** 表示中间变量区域第 20 个字节开始的字（由 MB20 和 MB21 这两个字节构成）。在不同的 PLC 实现中，根据目标硬件的资源的不同、需要处理的外部数字量以及模拟量数量的不同，往往分配给最终用户使用的每个区域的大小是不一样的。在某些情况下甚至区域的分布也有所不同。因此，对于任何一个 PLC 系统，开发者必须提供一个变量描叙文件。这个文件可以被 GUTTA 编程软件识别并记忆。一方面，在使用 GUTTA 软件编辑梯形图或指令表程序时，软件可以根据描述文件判断用户输入的变量是否合法。另一方面，在使用 GUTTA 软件编译梯形图或指令表时，软件可以根据描述文件给出错误信息并阻止错误的变量访问下载到目标系统中去。因此，编辑或修改变量描述文件是构建任何一个 PLC 系统的必须过程。阅读变量描述文件是确认一个 PLC 系统变量分布的最直接途径。

MODBUS 变量区间

GUTTA 变量描述文件的变量区间划分是在 MODBUS 变量区间划分的基础上进行的一种二次划分。并且 PLC 系统在进行 MODBUS 通讯时，依然依据 MODBUS 变量系统对内存进行访问。因此在这里有必要先明确 MODBUS 的变量区间。在标准的 MODBUS PLC 中，一般认为有下面 4 种变量区间。

输入线圈（**Input Coil**），从地址 10001 开始。

保持线圈（**Holding Coil**），从地址 00001 开始。

输入寄存器（**Input Register**），从地址 30001 开始。

保持寄存器（**Holding Register**），从地址 40001 开始。

其中，输入线圈可以使用 02 通讯指令操作；输出线圈可以使用 01，05，15 通讯指令操作；输入寄存器可以使用 04 通讯指令操作；保持寄存器可以使用 03、06、16 通讯指令操作。通讯指令对输入线圈和输入寄存器只能进行读操作。通讯指令对保持线圈和保持寄存器除了可以进行读操作之外，还能进行写操作。输入线圈和输入寄存器顾名思义，一般被做为离散量输入和模拟量输入的映像。保持线圈和保持寄存器除了被做为离散量输出和模拟量输出的映像之外，还可用来存储 PLC 程序中的中间变量（又称中间线圈或中间寄存器）。在作为输入输出映像的时候，可以通过一个叫做 IO 映射表的部件来配置（又称 IOMAP）。

（在 GUTTA 平台中，IO 映射表是固定的，用户不能够任意配置。在需要经常进行 IO 重映射的时候，建议使用符号表来进行 IO 的替换。但这样的替换是静态的，程序重新下载到 PLC 后，配置才生效。因此即使使用了符号表，也不能实现动态的 IO 重映设）

MODBUS 通讯协议中，输入线圈、保持线圈以位为单位操作。以位为单位意味着最小的操作单元是位，并且通讯中的偏移字段代表的也是位偏移。输入寄存器、保持寄存器以字为单位操作（2 字节）。以字为单位操作意味着最小的操作单元是字，并且通讯中的偏移字段代表的也是字偏移。

GUTTA 需要扩展的变量区间

完整的 GUTTA 系统中，在 MODBUS 提供的 4 类变量空间的基础上，还需要拓展两类变量空间。他们是**常数区域**和**临时区域**。由于这两个区域在 MODBUS 定义之外，因此这些区域的值不能被标准的 MODBUS 通讯协议直接访问。但是这些区域的值可以由 GUTTA 通讯协议直接访问。详情可参考文档：《UM4001 GUTTA 通讯协议》

常数区域用来存放 PLC 程序中使用的所有常数（指令中的立即数）；临时区域用来存放调用单元（函数调用或中断调用）中需要使用的临时变量。临时变量除了存放调用单元需要使用的临时变量，也可用于函数调用的参数传递。

典型的变量描述文件一览

变量描述文件以 XML（*Extensible Markup Language*）格式保存，一般保存在 GUTTA Ladder 软件安装文件夹中。在软件的安装文件夹中，有一个名为 *GuttaLad* 的子文件夹。在 *GuttaLad* 子文件夹中，又存在若干子文件夹，其中每一个子文件夹代表一种 CPU 配置。每个 CPU 配置下面分别有 *ManagerEnu* 和 *ManagerChs* 这两个子文件夹。这两个文件夹中的文件内容基本一致。只不过对应的语言选项不同。*ManagerEnu* 对应的语言选项为英文，*ManagerChs* 对应的语言选项为中文。如果不考虑多语言的支持，PLC 的构建者可以让两个文件夹中的文件保持一致（直接拷贝）。软件 GUTTA Ladder 在启动时根据当前的语言选项，载入这两个文件夹中的其中一个。

变量描述文件就在这两个目录中，文件名为 *ManagerVar.xml*。一个最简单的 *ManagerVar.xml* 看起来应该是这个样子的：

```
<?xml version="1.0" encoding="utf-16"?>
<ManagerVar Base="0"
            BitBase="0"
            Length="0"
            BitLength="0"
            Marker="0">
  <Region Slot="0"
          Name="I"
          Area="Di"
          AreaBegin="0"
          AreaEnd="16"
          Use="Value"
          Comment="Discrete inputs">
    <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
    <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
    <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
    <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</ManagerVar>
```

在这个 XML 文件中，根节点为 *ManagerVar*，这个节点中包含若干个 *Region* 节点。每一个 *Region* 节点对应了实际在 PLC 中能够使用的一个变量域。上面这段 XML 代码的大致

意思就是：在这个 PLC 变量系统中，只有一个名称为 **I**（*Name="I"*）的变量域。这个域分配在输入线圈中（*Area="Di"*）的前 16 个字节（MODBUS 的 10001 到 10128）。对这个域的访问只能使用直接取值的方式（*Use="Value"*）。取值有 4 种方式：按位取值（例如 **I1.2**）、按字节取值（例如 **IB10**）、按字取值（例如 **IW8**）、按双字取值（例如 **ID6**）。

变量描述文件节点属性说明

*<ManagerVar>*节点

ManagerVar 是变量描述文件的根节点。同时每个变量描述文件的根节点必须是 *ManagerVar*。

<ManagerVar>:Base 属性

数据类型为 INT（保留属性，目前 GUTTA Ladder 1.0 版本不能识别）。

节点 *ManagerVar* 的 *Base* 属性表示变量系统中变量偏移数值的开始值。在 MODBUS 中，**40001** 表示保持寄存器的首个字，偏移是从 1 开始，那么 *Base* 的值就应该是 1。在 CPU-EC20 中，**MW0** 表示中间变量的第首个字。偏移是从 0 开始，那么 *Base* 的值就应该是 0。*Base* 的值也可以取 0 和 1 以外的值，不过不建议这么做。除非有特殊含义，其它的值很可能对 PLC 的使用者造成困惑和不便。

<ManagerVar>:BitBase 属性

数据类型为 INT（保留属性，目前 GUTTA Ladder 1.0 版本不能识别）。

节点 *ManagerVar* 的 *BitBase* 属性和 *Base* 属性类似，用来说明位偏移的数值的开始值。在 CPU-EC20 中，**M100.0** 表示第 100 个字节的最低位，那么 *BitBase* 的值就应该是 0。若 *BitBase* 的值是 1，那么就必须用 **M100.1** 表示第 100 个字节最低位。*BitBase* 的值也可以取 0 和 1 以外的值，不过不建议这么做。除非有特殊含义，其它的值很可能对 PLC 的使用者造成困惑和不便。

<ManagerVar>:Length 属性

数据类型为 INT（保留属性，目前 GUTTA Ladder 1.0 版本不能识别）。

节点 *ManagerVar* 的 *Length* 属性定义了变量偏移的字符串长度。在 MODBUS 中，变量 **40001** 给变量偏移保留了 4 个 0 字符，故变量偏移的字符串长度为 5。Length 为 0 表示不添加占位的字符 0。

<ManagerVar>:BitLength 属性

数据类型为 INT（保留属性，目前 GUTTA Ladder 1.0 版本不能识别）。

节点 *ManagerVar* 的 *BitLength* 属性和 *Length* 属性类似，定义了位变量偏移的字符串长度。*BitLength* 为 0 表示不添加占位的 0。

<ManagerVar>:Marker 属性

数据类型为 INT（保留属性，目前 GUTTA Ladder 1.0 版本不能识别）。

在 IEC61131-6 的标准中，所有直接地址前需要加入标识符“%”，以区别于符号地址。由于在小型 PLC 系统中，直接地址比符号地址更加常用，因此在很多软件中“%”可以省略。Marker 为 0 表示系统在显示直接变量时前面省略“%” 标识符。Marker 为非 0 的值表示系统在显示直接变量时前面必须加上“%” 标识符。

*<ManagerVar>:<Region>*节点

一个 *ManagerVar* 根节点应该包含多个 *Region* 节点。每一个 *Region* 节点代表 PLC 的变量域。例如在 CPU-EC20 中，中间变量 **M** 和定时器变量 **T** 分别用两个 *Region* 来描述。

<ManagerVar><Region>:Slot 属性

数据类型为 INT，范围为 0~15（16#00~16#0F）。

Region 的 *Slot* 属性定义当前变量域的识别码。由于效率的原因，GUTTA PLC 固件不能识别字符串形式的变量。梯形图或指令表中的变量必须以二进制的形式下载到 PLC 中（编译型的 PLC 需要根据此二进制继续生成特定的语言）。*Slot* 的二进制值是 PLC 固件识别本变量域的唯一标识。

<ManagerVar><Region>:Name 属性

数据类型为 STRING，可以为字母或数字，至少一个字符长度。

Region 的 *Name* 属性定义当前变量域的名称。名称 *Name* 和识别码 *Slot* 一一对应。在 CPU-EC20 中，中间变量的名称是“**M**”（例如 **MW20**），系统变量的名称是“**SM**”（例如 **SMB20**）。

<ManagerVar><Region>:Area 属性

数据类型为 STRING，可能的值有：“*Di*”“*Do*”“*Ri*”“*Ro*”“*Const*”“*Local*”。

Region 的 *Area* 属性定义当前变量域所在的分区：

“*Di*”表示 MODUBS 的输入线圈区（**Input Coil**）。

“*Do*”表示 MODUBS 的保持线圈区（**Holding Coil**）。

“*Ri*”表示 MODUBS 的输入寄存器区（**Input Register**）。

“*Ro*”表示 MODUBS 的保持寄存器区（**Holding Register**）。

“*Const*”表示扩展变量区间的常数区。

“*Local*”表示扩展变量区间的临时区。

需要注意的是，*ManagerVar* 的所有 *Region* 中，必须有一个且只能有一个 *Area* 为“*Const*”的 *Region*；必须有一个且只能有一个 *Area* 为“*Local*”的 *Region*。否则 GUTTA Ladder Editor 在启动的时候会返回错误而终止程序的启动。

<ManagerVar><Region>:AreaBegin 属性

数据类型为 INT。

Region 的 *AreaBegin* 属性表示本区域在 *Area* 中的开始地址。

<ManagerVar><Region>:AreaEnd 属性

数据类型为 INT。

Region 的 *AreaEnd* 属性表示本区域在 *Area* 中的结束地址。

<ManagerVar><Region>:Use 属性

数据类型为 STRING，可能的值有：“*Address*”“*Value*”“*Pointer*”。*Use* 的值也可以是这些可能的值的复合形式，例如“*Address*| *Value*”表示即支持 *Address* 用法，也支持 *Value* 用法。

Region 的 *Use* 属性表示本区域可能的用法。

“*Address*”表示本区域的变量可以写成地址的形式，例如**&MB20**。

“*Value*”表示本区域的变量可以写成变量的形式，例如 **MW20**。

“*Pointer*”表示本区域的变量可以写成变量的形式，例如***MD20**。

对于不支持地址变量的精简 GUTTA PLC 系统，所有的区域都必须只有 *Value* 用法。

<ManagerVar><Region>:Comment 属性

数据类型为 STRING。

对本变量区域的一个描述。对读者来说是一个注释，以方便理解本区域的作用。简单的处理可以直接设为空字符串。设置时需要考虑当前变量描述文件所属的语言。

*<ManagerVar><Region><Access>*节点

在 *Region* 节点中应该至少包含一个 *Access* 节点。*Acess* 节点定义了对本变量区域的一种具体引用类型。

<ManagerVar><Region><Access>:Name 属性

数据类型为 STRING。

Access 的 *Name* 属性定义当前引用类型的**后缀**。GUTTA PLC 系统通过此后缀确定当前的引用类型。例如在 CPU-EC20 中，**MB20** 的“**M**”表示中间变量区域（中间变量区域的名称是“**M**”）。同时，“**B**”表示在中间变量区域中的引用类型后缀（由于后缀名为“**B**”的引用类型的数据宽度是字节，故 **MB20** 表示引用一个字节）。

<ManagerVar><Region><Access>:Width 属性

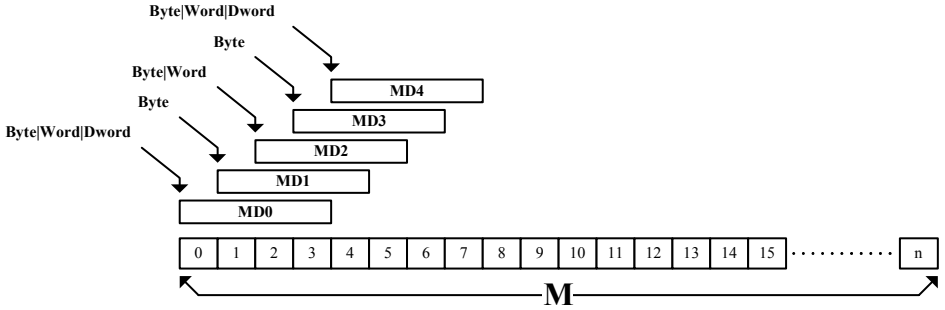
数据类型为 STRING。可能的值有“*Bit*”、“*Byte*”、“*Word*”、“*Dword*”。

Access 的 *Width* 属性定义当前引用类型的的**数据宽度**。“*Bit*”、“*Byte*”、“*Word*”、“*Dword*”代表的数据宽度分别为位、字节（8bit）、字（8bit）、双字（32bit）。

<ManagerVar><Region><Access>:Step 属性

数据类型为 STRING。可能的值有“*Bit*”、“*Byte*”、“*Word*”、“*Dword*”。

Access 的 *Step* 属性定义当前引用类型的**对齐宽度**。“*Bit*”、“*Byte*”、“*Word*”、“*Dword*”代表的数据宽度分别为位（1bit）、字节（8bit）、字（8bit）、双字（32bit）。例如在 CPU-EC20 系统中，由于 **M** 区域中 **D** 引用类型的对齐宽度为“*Byte*”，因此 **MD0**、**MD1**、**MD2**、**MD3**、**MD4** 都是合法的变量，因为他们都是字节（8bit）对齐的。若将对齐宽度改为“*Word*”，那么以上变量中就只有 **MD0**、**MD2**、**MD4** 是合法变量了，因为他们是字（16bit）对齐的。若将此值改为“*Dword*”，那么以上变量中就只有 **MD0**、**MD4** 是合法变量了，因为他们是双字（32bit）对齐的。



在一些 CPU 架构中，CPU 不支持非对齐数据的访问。例如 ARM7 就没有非字对齐的字载入指令。在种 CPU 架构中，可以通过限制非对齐变量的访问来提高执行效率。

<ManagerVar><Region><Access>:Offset 属性

数据类型为 STRING。可能的值有“*Bit*”、“*Byte*”、“*Word*”、“*Dword*”。

Access 的 *Offset* 属性定义当前引用类型的**偏移单位宽度**。例如在 CPU-EC20 中，**MB20** 表示 **M** 区域开始的第 20 个字节。而 **T20** 表示 **T** 区域开始的第 20 个字（第 40 个字节位置

开始的字）。因为 **MB20** 对应的引用类型偏移单位宽度为“*Byte*”，而 **T20** 对应的引用类型偏移单位宽度为“*Word*”。

附录

CPU-EC20 变量描述文件范例

```
<?xml version="1.0" encoding="utf-16"?>
<ManagerVar Base="0"
          BitBase="0"
          Length="0"
          BitLength="0"
          Marker="0">
  <Region Slot="0"
        Name="I"
        Area="Di"
        AreaBegin="0"
        AreaEnd="16"
        Use="Value"
        Comment="Discrete inputs">
    <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
    <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
    <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
    <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
  </Region>
  <Region Slot="1"
        Name="Q"
        Area="Do"
        AreaBegin="0"
        AreaEnd="16"
        Use="Value"
        Comment="Discrete outputs">
    <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
    <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
    <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
    <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
  </Region>
  <Region Slot="2"
        Name="AI"
        Area="Ri"
        AreaBegin="0"
        AreaEnd="16"
```

```
        Use="Value"
        Comment="Analog inputs">
  <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
  <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
  <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="3"
        Name="AQ"
        Area="Ro"
        AreaBegin="0"
        AreaEnd="16"
        Use="Value"
        Comment="Analog outputs">
  <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
  <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
  <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="4"
        Name="M"
        Area="Ro"
        AreaBegin="16"
        AreaEnd="416"
        Use="Address|Value|Pointer"
        Comment="Internal memory">
  <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
  <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
  <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
  <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="5"
        Name="T"
        Area="Ro"
        AreaBegin="416"
        AreaEnd="480"
        Use="Value"
        Comment="Timer currents">
  <Access Name="" Width="Word" Step="Word" Offset="Word"/>
</Region>
<Region Slot="6"
        Name="C"
        Area="Ro"
        AreaBegin="480"
        AreaEnd="512"
        Use="Value"
```

```
        Comment="Counter currents">
  <Access Name="" Width="Word" Step="Word" Offset="Word"/>
</Region>
<Region Slot="7"
        Name="SM"
        Area="Ro"
        AreaBegin="512"
        AreaEnd="544"
        Use="Address|Value"
        Comment="Special memory">
  <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
  <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
  <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
  <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="8"
        Name="J"
        Area="Ro"
        AreaBegin="544"
        AreaEnd="560"
        Use="Value"
        Comment="Jumper memory">
  <Access Name="" Width="Byte" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="9"
        Name="K"
        Area="Const"
        AreaBegin="0"
        AreaEnd="128"
        Use="Value"
        Comment="Const memory">
  <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
  <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
  <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
  <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
</Region>
<Region Slot="10"
        Name="L"
        Area="Local"
        AreaBegin="0"
        AreaEnd="32"
        Use="Address|Value|Pointer"
        Comment="Local variable memory">
  <Access Name="" Width="Bit" Step="Byte" Offset="Byte"/>
```

```
    <Access Name="B" Width="Byte" Step="Byte" Offset="Byte"/>
    <Access Name="W" Width="Word" Step="Byte" Offset="Byte"/>
    <Access Name="D" Width="Dword" Step="Byte" Offset="Byte"/>
  </Region>
</ManagerVar>
```