**LvData🡨🡪MemoryData**

**数据解析规范**

**编制：xxx**

**2020年05月15日**

协议修改历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **修改内容** | **修改人** | **日期** | **协议版本** |
| 1 | 编制 |  |  | V1.00 |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 说明：  1：协议版本号采用三个数据位标识。例：V1.00  2：每修改一次协议，需要将协议文件的版本号加1，同时要在协议修改历史中写出修改的内容。 | | | | |

# 概述

此函数库基于APT Demo的应用开发，实现了LabVIEW数据与内存数据的互相转换，从而保证LabVIEW与其他运行环境进行数据交互时具有统一的接口类型。既支持本地化的数据交互也支持基于RPC协议的远程数据交互。本地化的数据交互可以通过内存地址直接访问内存数据，即PTR模式，远程数据交互必须将数据进行二进制序列化之后才能实现数据的传输，即NO\_PTR模式。

# LvData🡨🡪MemoryData数据类型

针对LabVIEW在测试测量过程中常用的数据类型设计数据解析规范，主要支持的数据类型如下表所示：

|  |  |  |
| --- | --- | --- |
| 变量类型 | 变量长度 | 说明 |
| Boolean | 1个字节(8位) | 0表示False，1表示True |
| I8 | 1个字节(8位) | 有符号字节型 |
| U8 | 1个字节(8位) | 无符号字节型 |
| I16 | 2个字节(16位) | 有符号短整型 |
| U16 | 2个字节(16位) | 无符号短整型 |
| I32 | 4个字节(32位) | 有符号整型 |
| U32 | 4个字节(32位) | 无符号整型 |
| I64 | 8个字节(64位) | 有符号长整型 |
| U64 | 8个字节(64位) | 无符号长整型 |
| SGL | 4个字节(32位) | 单精度浮点型 |
| DBL | 8个字节(64位) | 双精度浮点型 |
| Timestamp | 8个字节(64位) | 时间戳。使用1904.1.1 00:00:00 作为基准，DBL的值表示当前时间举例基准时间的差值，单位为秒(s). |
| Array | **PTR Mode:**  12个字节（96位）  **NOPTR Mode:**  根据元素和数据长度变化 | **PTR Mode:**  长度=数据长度(U32)+数据首地址(U64)  **NOPTR Mode:**  长度=数据长度(U32)+数组长度(n)\*元素变量长度 |
| Waveform | **PTR Mode:**  12个字节（96位）  **NOPTR Mode:**  根据数据长度变化,可以当作DBL 解析，第一位为T0，第二位为deltaT，以后的数据为 Data。 | Waveform = Array(T0(DBL) + deltaT(DBL) +Data(DBL Array))  **PTR Mode:**  长度=数据长度(U32)+数据首地址(U64)  **NOPTR Mode:**  长度=数据长度(U32)+数组长度(n)\*元素变量长度 |
| Cluster | 根据组成元素变化 | Cluster 的组成元素按照顺序排列，长度为组成元素长度之和。 |
| String | 1024个字节，当作U8Array解析 | 不包含长度信息，长度小于1024时，空位补0 |
| Path | 1024个字节，当作U8Array解析 | 不包含长度信息，长度小于1024时，空位补0 |

举例：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据 |  | Binary数据格式 |  |
|  | Cluster | **PTR Mode:**  DD040000  8D976E1283C0F33F  54657374537472696F67000…  04000000  C03B3F1500000000  0600000000  10034C1500000000  **NOPTR Mode:**  DD040000  8D976E1283C0F33F  54657374537472696F67000…  04000000  0000000000000000  000000000000F03F  0000000000000040  0000000000000840  06000000  0000008864BBEB41  000000000000F03F  000000000000F03F  0000000000000040  0000000000000840  0000000000001040 | **PTR Mode:**  I32,  DBL  String(1024 Byte)  Array Length(U32)  Array Addr(U64)  Waveform Length(U32)  Waveform Addr(U64)  **NOPTR Mode:**  I32,  DBL  String(1024 Byte)  Array Length(U32) 4  DBL Array[0] 0.00  DBL Array[1] 1.00  DBL Array[2] 2.00  DBL Array[3] 3.00  Waveform Length(U32) 6  TimeStamp: 12:00:00 PM 12/12/2021  Dt：1.00  Waveform[0]:1.00  Waveform[0]:2.00  Waveform[0]:3.00  Waveform[0]:4.00 |
|  | DBL Array | **PTR Mode:**  04000000  80043F1500000000  **NOPTR Mode:**  04000000  0000000000001440  000000000000F03F  0000000000000040  0000000000000840 | **PTR Mode:**  U32 Array(Length) 4  U64 Array Addr  **NOPTR Mode:**  U32 Array(Length) 4  DBL Array[0] 5.00  DBL Array[1] 1.00  DBL Array[2] 2.00  DBL Array[2] 3.00 |

## 2.2 解析规范

为了便于转码，将LabVIEW 数据转换为Byte Array的内存格式进行传输。

### 2.2.1 简单数据类型

简单数据类型主要包括：I8, I16, I32, ​I64, U8, U16,​U32, U64, SGL, ​DBL, Boolean

简单数据类型直接转换为Byte Array, 通过ByteArray的内存数据格式进行传输，要特别注意不同环境下的**大小字节序(LabVIEW高位在前)**，简单数据类型与ByteArray的对应关系如下：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type | I8 | I16 | I32 | I64 | U8 | U16 | U32 | U64 | SGL | DBL | BOOL |
| Byte | 1 | 2 | 4 | 8 | 1 | 2 | 4 | 8 | 4 | 8 | 1 |

## 2.2.2 string&path数据类型（1024Byte）

**path数据类型**：

* 将其视为特殊的string数据类型进行处理，详细查看string数据类型。

**string 数据类型**：

* LabVIEW将字符串转换为Byte Array时，默认前4个Byte包含长度信息，而在众多文本语言中，长度信息需要进行额外的计算。因此LvData🡪MemoryData转换时不包含长度信息。
* 为了便于字符串的处理，以及内存空间的管理，默认字符串的最大长度为1024，字符串按照1024长度进行传输。如果字符串长度不足，则补NULL字符(ASCII = 0).
* MemoryData🡪LvData转换时，通过ASCII=0字符获取末尾元素，解析字符串。

## 2.2.3 TimeStamp数据类型（8Byte）

* LvData🡪MemoryData将TimeStamp转换为DBL类型，然后将DBL数据转换为内存格式进行传输。
* TimeStamp转换为DBL时，以01-01-1904 00:00:00为0时刻，计算相对时间。
* MemoryData🡪LvData接收到的是DBL数据类型，以0时刻为基准，转换为真实的TimeStamp数据类型。

## 2.2.4 Waveform数据类型（PTR mode: 12Byte）

* LvData🡪MemoryData: Waveform包含TimeStamp，dt以及DBL 1D array 三种信息。为了便于数据转换及传输，将TimeStamp转换为DBL, dt转换为DBL，然后将TimeStamp和dt信息插入到DBL 1D Array的前面，从而将waveform转换为1维DBL数据进行存储。
* MemoryData🡪LvData: 按照1维数据获取Byte Array内存格式数据，1~4字节表数据总长度, 9~16字节表示TimeStamp信息, 17~24字节表示dt信息，后续数据长度根据总长度进行计算解析。
* 关于1维DBL数据数据转换详见下小节

**PTR Mode:**

* 为了保证1维数据地址有效，必须在数据传输前申请内存地址，数据交互结束后释放内存地址，以防止内存数据的丢失。

## 2.2.5 1D Array数据类型(PTR mode: 12Byte)

**PTR Mode:**

* 只支持2.2.1~2.2.4小节中描述的数据类型。
* 所有的1维数据都可以使用长度及首地址信息进行描述：

NI\_ARRAY{

U32: Length；

U64: Address;

}

* LvData🡪MemoryData: 首先计算1维数组长度，然后获取数组首地址。将长度及首地址转换为内存格式进行传输, 所有的1维数组都可以利用12 Byte内存数据在本地进行传输。
* MemoryData🡪LvData: 通过12Byte内存数据解析数据长度，以及首地址。根据首地址访问所有的数组元素。
* 为了保证1维数据地址有效，必须在数据传输前申请内存地址，数据交互结束后释放内存地址，以防止内存数据的丢失。

**NOPTR Mode:**

* 只支持2.2.1~2.2.4小节中描述的数据类型。
* LvData🡪MemoryData: 将1维数组的长度信息插入数据头部，占用4Byte. 1维数组的数据逐一进行内存格式转换，与头部4Byte拼接完整的1维数据内存数据格式。
* MemoryData🡪LvData: 通过头部4Byte内存数据解析数据长度，然后根据1维数组元素的数据类型，获取1维数组元素的内存格式数据，然后进行数据转换。

## 2.2.6 Cluster数据类型

* 只支持2.2.1~2.2.5小节描述的数据类型组成的簇结构。
* 将Cluster元素逐一按照上述的数据类型进行解析。

**PTR Mode:**

* 为了保证1维数据地址有效，必须在数据传输前申请内存地址，数据交互结束后释放内存地址，以防止内存数据的丢失。

# API使用说明

## LvData🡪MemoryData

Graphical user interface, diagram

Description automatically generatedDiagram

Description automatically generated

PTR Mode

Graphical user interface, diagram

Description automatically generated

NOPTR Mode

**PTR Mode:** PTR模式为了传输内存数据地址，需要进行内存管理。传输数据前需要申请内存，数据使用后需要释放内存。因此，PTR模式需要和Mem\_ObtainPTRQueue.vi以及Mem\_ReleasePTRQueue,vi配合使用。

Diagram, text

Description automatically generated

**NOPTR Mode:** 直接对数据进行转换，LabVIEW可以自行管理内存，可直接使用。

## MemoryData🡪LvData

A picture containing text, indoor, screenshot

Description automatically generated

A picture containing text, indoor, computer, screenshot

Description automatically generatedA picture containing text, indoor, screenshot

Description automatically generated

PTR Mode

NOPTR Mode

**PTR Mode:** 内存管理由数据提供方负责，Mem\_MemoryToLvData.vi只负责对数据的解析。按照接口要求输入即可。

**NOPTR Mode:** 直接对数据进行转换，按照接口要求输入即可