Acute LAVISA

Version 1.0.9 rev. 3

Acute Technology Inc.

Copyright © 2025 Acute Technology Inc.

Table of contents

1. Introduction 4
1.1 Scope 4
1.2 Core Concepts 5
1.3 Supported Platforms 6
1.3.1 Windows 6
1.3.2 Linux 6
1.4 Interfaces 7
2. Basic Control 8
2.1 Flow Chart 8
2.2 Installation 9
2.2.1 概述 9
2.2.2 安裝步驟說明 9
2.3 DLL Functions 10
2.3.1 viSelectAppType 10
2.3.2 viOpenRM 10
2.3.3 viWrite 10
2.3.4 viRead 11
2.3.5 viGetCommandResult 11
2.3.6 viCloseRM 11
2.3.7 viErrCode 11
3. Remote Control 12
3.1 Overview 12
3.2 Quick Start 13
3.2.1 啟動 gRPC 伺服器 13
3.2.2 Python 範例客戶端 14
3.3 Protobuf Definition 16
3.3.1 遠端程序呼叫 (RPCs) 16
3.3.2 Message 16
3.3.3 Enum 17
4. Supported Commands 18
4.1 Common Commands 18
4.1.1 指令格式 18
4.1.2 指令定義 19
4.2 File Subsystem 21
4.2.1 指令格式 21

Copyright © 2025 Acute Technology Inc.

4.2.2 指令定義 21
4.3 Window Subsystem 28
4.3.1 指令格式 28
4.3.2 指令定義 28
4.4 Logic Analyzer Mode Subsystem 30
4.4.1 什麼是邏輯分析儀模式？ 30
4.4.2 指令格式 30
4.4.3 指令定義 33
4.5 Protocol Analyzer Mode Subsystem 73
4.5.1 什麼是協定分析儀模式？ 73
4.5.2 指令格式 73
4.5.3 指令定義 74
5. Error Codes 80
5.1 狀態碼表 80
6. Release Notes 83

Copyright © 2025 Acute Technology Inc.

1. Introduction

1.1 Scope

這是 Acute LAVISA（簡稱 AqLAVISA）的完整文件。

本文件涵蓋以下主題：

- AqLAVISA 基本概念和功能
- AqLAVISA 的介面及其使用方式
- 支援的指令及其使用方式

本文件不涵蓋以下主題：

- 第三方應用程式的系統設計細節
- 特定協定的解碼和分析細節

AqLAVISA 提供了一個直覺的介面，用於無縫管理 Acute 邏輯分析儀的控制。開發人員可以使用此介面來擷取、分析和儲存來自邏輯分析儀的擷取資料。

Copyright © 2025 Acute Technology Inc.

1.2 Core Concepts

AqLAVISA 提供 API 介面，讓使用者程式能透過文字指令與 Acute 邏輯分析儀軟體進行通訊。使用者可以透過此介面存取協定分析頁面中的協定分析報告、解碼報告，或邏輯分析儀頁面中的時序分析報告。

對於自動化測試，開發人員可以透過載入範例波形檔案來設定擷取和環境設定，然後開始擷取以記錄新資料，並讀取解碼或分析資訊來檢查和驗證目標裝置的行為。

AqLAVISA 介面允許開發人員透過發送純文字指令來自動化軟體。開發人員能夠在擷取程序後收集這些資料，如解碼資訊、時序測量等。

最初，此介面是透過共用函式庫（.dll）提供，用於處理所有接收到的指令並與軟體進行通訊。然而，這種方式僅限於儀器實體連接的單一機器上使用，這產生了一個問題。因此，遠端控制成為了一種使其更加通用和靈活的方式。

Copyright © 2025 Acute Technology Inc.

1.3 Supported Platforms

1.3.1 Windows

軟體	版本
MSO	1.7.82 或以上
MSO (Full)	1.7.82 或以上
BusFinder and Logic Analyzer	1.7.74 或以上
TravelLogic	1.7.47 或以上
TravelBus	1.7.34 或以上

1.3.2 Linux

Note

Linux 平台上的應用程式仍標記為測試版本。如果您在執行程式時遇到任何錯誤，請建立問題回報。

軟體	版本
Mixed_Signal_Oscilloscope	2.0.0 或以上
Digital_Storage_Oscilloscope	2.0.0 或以上
BusFinder_Logic_Analyzer	2.0.0 或以上
TravelLogic_Analyzer	2.0.0 或以上
TravelBus_Analyzer	2.0.0 或以上

Copyright © 2025 Acute Technology Inc.

1.4 Interfaces

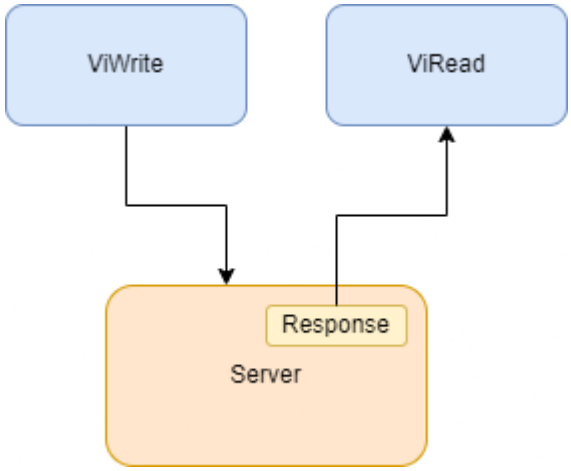
AqLAVISA 提供三種不同的介面來與邏輯分析儀互動。每個介面都有連結到以下的說明、教學和 API 參考。支援的介面如下：

- 本地 Socket 行程間通訊: 僅限 Windows 平台
- TCP Socket 連線: 僅限 Windows 平台
- gRPC

Copyright © 2025 Acute Technology Inc.

2. Basic Control

2.1 Flow Chart



只需透過 viWrite 呼叫來發送指令。伺服器會在指令完全執行後產生回應。如果有需要回傳的查詢值，您可以使用 viRead 呼叫來取得該值。

Copyright © 2025 Acute Technology Inc.

2.2 Installation

2.2.1 概述

AqVISA 包含以下幾個部分：

- 動態連結函式庫（DLL）檔案
- 64 位元 Windows 系統使用 `AqVISA64.dll`
- 函式庫相依檔案

元件	檔案名稱
Qt 5 核心函式庫	`Qt5Core.dll`
Qt 5 網路函式庫	`Qt5Network.dll`
ICU 函式庫	`icudt.dll`、`icuin.dll`、`icuuc.dll`
C++ 執行階段函式庫	`msvcp120.dll`、`msvcr120.dll`

- 使用者手冊
- `AqVISA_UserManual_cht.pdf`
- 提供Python 使用者的 Python 套件

  PyPI 套件：aqvisa GitHub 儲存庫：aqvisa-python

2.2.2 安裝步驟說明

- 執行安裝程式

  請參考安裝頁面下載安裝程式。

  預設情況下，安裝程式會將動態連結函式庫和相依函式庫安裝到以下目錄：

  ```
  C:\Users\<使用者名稱>\AppData\Local\Acute\SDK\AqVISA
  ```

  安裝程式提供將安裝目錄加入系統 PATH 的選項。 建議大多數使用者選擇此選項，以便輕鬆找到動態連結函式庫。

- （選擇性）安裝 Python 套件

  您可以使用名為 `pip` 的套件管理器從 Python 套件索引（PyPI）安裝 aqvisa 套件。

  ```
  pip install aqvisa
  ```

- 範例使用

  請參考 GitHub 儲存庫中的範例程式碼。

  Python

Copyright © 2025 Acute Technology Inc.

2.3 DLL Functions

2.3.1 viSelectAppType

```
int viSelectAppType(int iAppType)
```

設定目標連接的 Acute 邏輯分析儀軟體。

參數

- `iAppType [In]`:

  類型：***int***

  可用的參數如下列表所示

參數	值	目標軟體	建議軟體版本
APP_TYPE_TL	0	TL (x64).exe / TL.exe	1.7.47 or higher
APP_TYPE_BF	1	BF_LA3K(x64).exe / BF_LA3K.exe	1.7.74 or higher
APP_TYPE_TB	2	TBA (x64).exe / TBA.exe	1.7.34 or higher
APP_TYPE_MS	3	MSO (x64).exe / MSO.exe	1.7.82 or higher

回傳值

成功時回傳 1；否則回傳 0。

2.3.2 viOpenRM

```
int viOpenRM()
```

連接到邏輯分析儀軟體。

回傳值

成功時回傳 1；否則回傳 0。

2.3.3 viWrite

```
int viWrite( char* szCommand )
```

向邏輯分析儀軟體發送 AqLAVISA 指令。

參數

- `szCommand [In]`:

  類型：***char ****

  包含 AqLAVISA 指令，詳細資訊請參考 Acute LAVISA 指令列表。

回傳值

成功時回傳 1；否則回傳 0。

Copyright © 2025 Acute Technology Inc.

2.3.4 viRead

```
int viRead(char *szReturnBuffer, int iBufferLength )
```

從邏輯分析儀軟體讀取 AqLAVISA 指令執行結果。

參數

- `szReturnBuffer [Out]`:

  Type: ***char ****

  用於 AqLAVISA 指令回傳值的資料緩衝區。

- `iBufferLength [In]`:

  Type: ***int***

  用於 szReturnBuffer 的緩衝區大小。

回傳值

回傳讀取到的字數長度。

2.3.5 viGetCommandResult

```
int viGetCommandResult()
```

讀取前一個 AqLAVISA viWrite 函式呼叫的指令處理結果。

回傳值

回傳值	值	描述
PROC_OK_WOUT_RETURN	0	處理成功且無回傳資料。
PROC_OK_WITH_RETURN	1	處理成功且有回傳資料。
PROC_FAIL	2	處理失敗。
PROC_PENDING	3	處理中。

2.3.6 viCloseRM

```
int viCloseRM()
```

停用 AqLAVISA 功能並斷開與邏輯分析儀軟體的連接。

回傳值

成功時回傳 1；否則回傳 0。

2.3.7 viErrCode

```
int viErrCode()
```

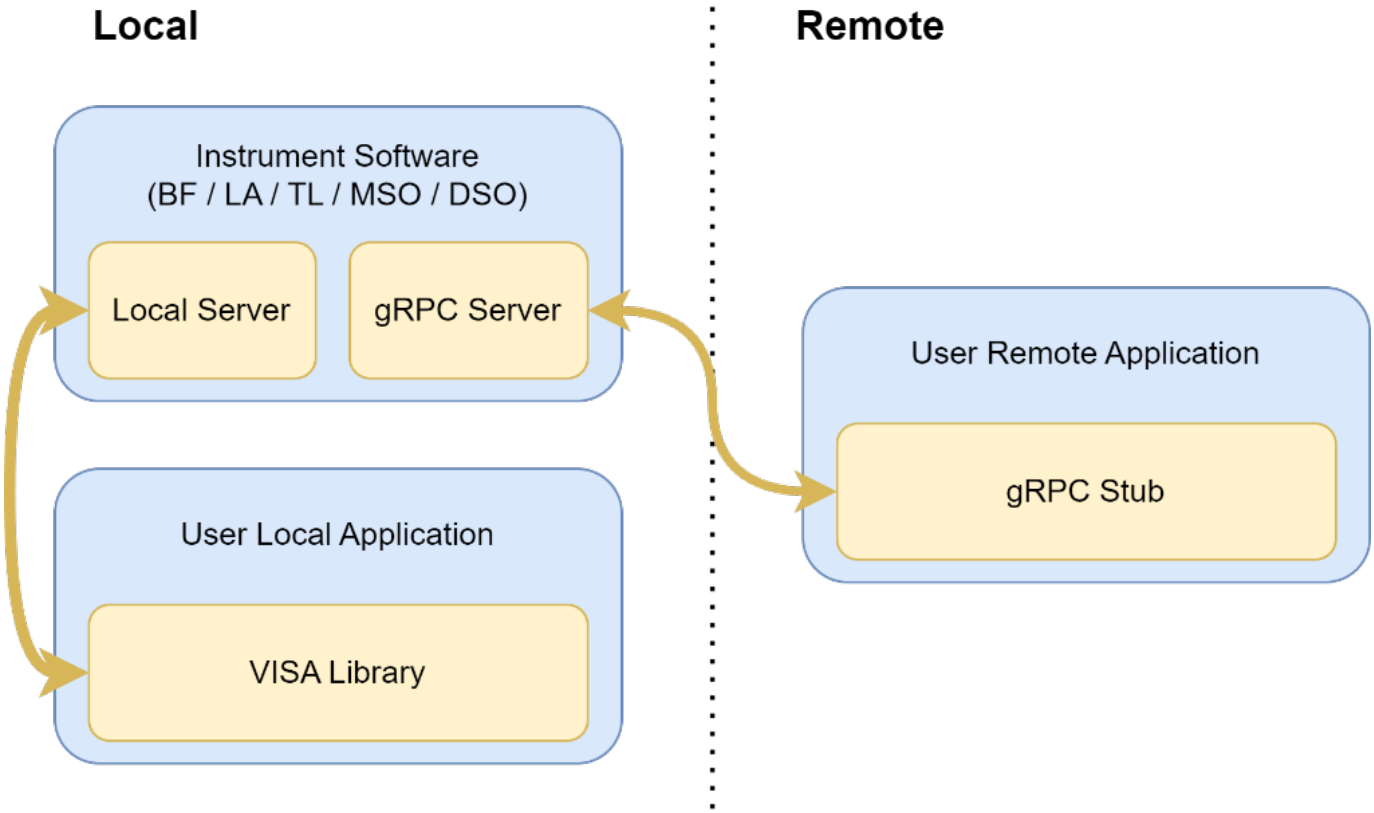
取得 AqLAVISA 錯誤代碼。詳細資訊請參考 錯誤碼。

Copyright © 2025 Acute Technology Inc.

3. Remote Control

3.1 Overview

如同許多自動化系統，AqVISA 透過允許新型態的文字指令來保持介面的一致性並擴充其功能。整體架構圍繞著客戶端-伺服器模型的概念。在伺服器端，儀器軟體實作了 AqVISA 介面並運行伺服器以處理傳入的指令。從下方的方塊圖中可以看到，軟體同時運行本地伺服器和 gRPC 伺服器，以處理來自本地應用程式或遠端應用程式的請求。



在客戶端，由於 gRPC 支援多語言這項基本功能，開發人員可以使用他們偏好的任何程式語言來建立應用程式。例如，您可以輕鬆地使用 Python、C# 等語言建立客戶端應用程式。

在本節中，我們將僅涵蓋透過 gRPC 進行遠端控制的設定。

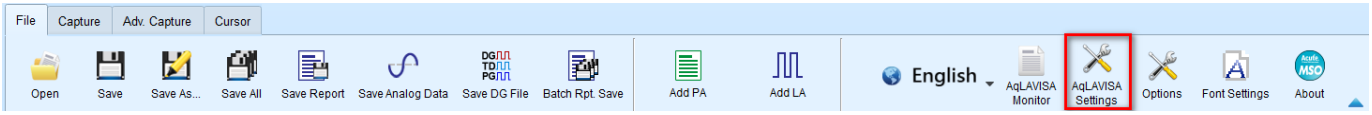
Copyright © 2025 Acute Technology Inc.

3.2 Quick Start

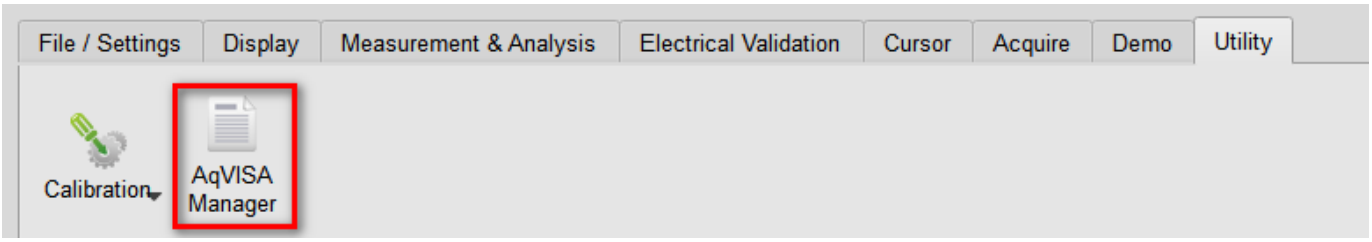
3.2.1 啟動 gRPC 伺服器

gRPC 伺服器可以透過軟體使用者介面或命令列來啟用

對於 BusFinder / Logic Analyzer / TravelLogic / MSO 軟體，選擇 **檔案 > AqLAVISA 管理器**。

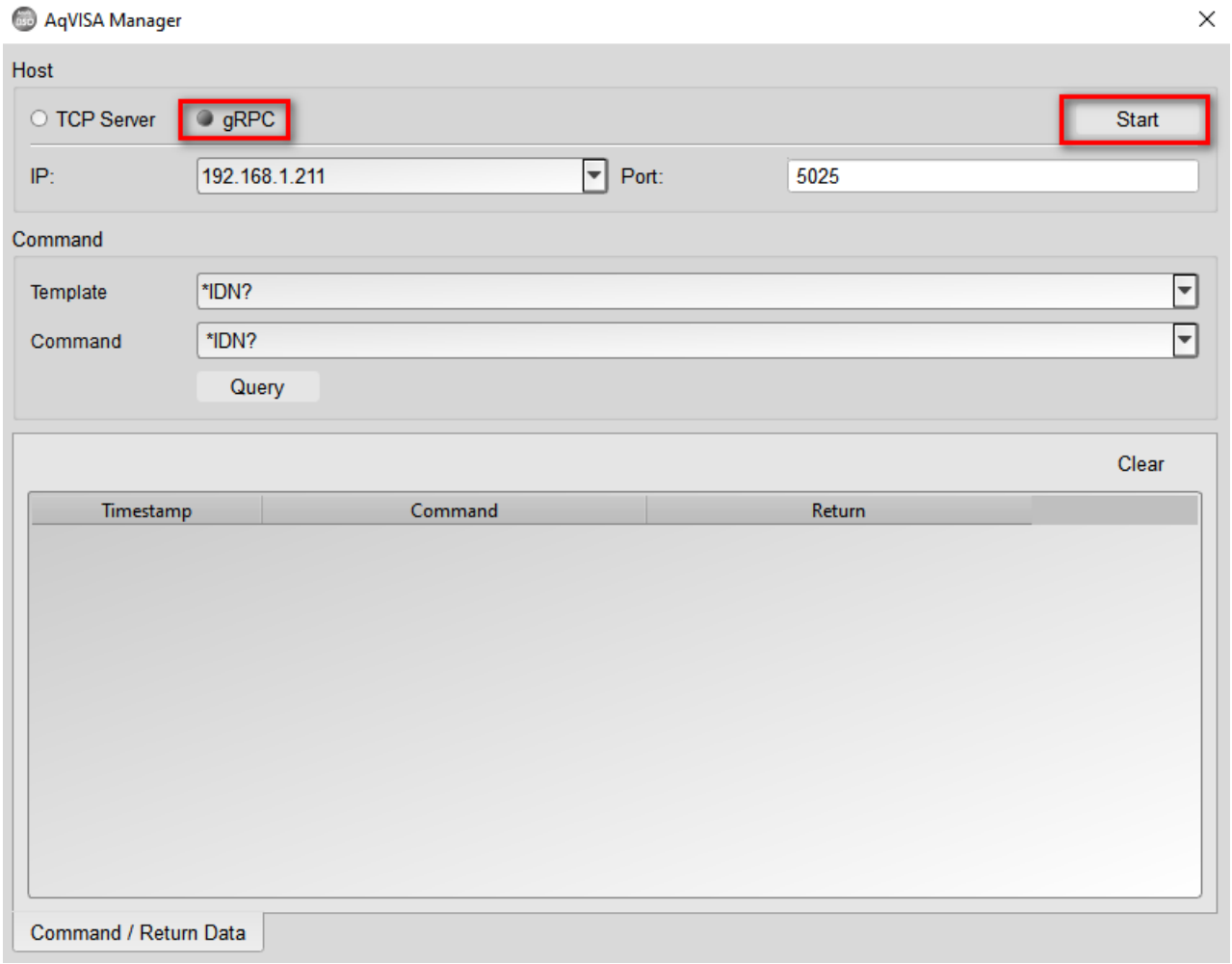


對於 DSO 軟體，選擇 **工具 > AqVISA 管理器**。



當主機設定為 **gRPC** 並按下 **開始** 後，gRPC 伺服器將在連接埠 5025 上開始運行。

Copyright © 2025 Acute Technology Inc.



或者，您可以在啟動軟體時透過命令列啟動伺服器。

以 MSO 為例

Windows 命令提示字元

```
MSO(x64).exe --host grpc --port 5025
```

Windows PowerShell

```
.\MSO(x64).exe --host grpc --port 5025
```

Linux

```
./Mixed_Signal_Oscilloscope-x86_64.AppImage --host grpc --port 5025
```

3.2.2 Python 範例客戶端

我們發布了一個 Python 範例客戶端程式碼。請按照以下步驟執行範例客戶端。另請參閱 GitHub 儲存庫 以獲取更多詳細資訊。

Copyright © 2025 Acute Technology Inc.

需求

我們目前使用 gRPC v1.56.0。

```
pip install -r requirements.txt
```

Protobuf 編譯

請在 `proto/` 目錄中找到 `aqvisa.proto`。執行以下命令進行編譯

```
python -m grpc_tools.protoc -Iproto
                            --python_out=.
                            --pyi_out=.
                            --grpc_python_out=.
                            proto/aqvisa.proto
```

有關 protobuf 的更多資訊，請訪問 Protocol Buffers 官方網站。

範例程式碼

請先將 `address` 變數替換為您自己的 IP 和連接埠。您可以從 DSO 或 LA 應用程式中獲取這些值。

VIWRITE 和 VIREAD

只需執行程式碼即可向應用程式發出單一請求

```
python write_read.py
```

在 `command_list` 中加入任何自訂指令來嘗試您偏好的任何指令。

VIWRITEFROMFILE 和 VIREADTOFILE

另外將 `json_file` 變數替換為指定的路徑。然後，只需執行程式碼即可向應用程式發送 JSON 內容。

```
python write_file.py
```

此外，您也可以從應用程式請求 JSON 設定。

```
python read_file.py
```


Copyright © 2025 Acute Technology Inc.

3.3 Protobuf Definition

3.3.1 遠端程序呼叫 (RPCs)

ViRead

查詢由 ViWrite 發送的前一個請求指令的回應值（如果有的話）。

另請參閱 ViReadRequest 和 ViReadResponse 訊息。

ViWrite

使用參數 *command* 向應用程式發送指令。成功執行時回傳 AQVI_NO_ERROR。否則，伺服器回應在 AQVI_STATUS 中定義的錯誤代碼。

另請參閱 ViWriteRequest 和 ViWriteResponse 訊息。

ViWriteFromFile

此 RPC 用於發送設定檔案，而不是向應用程式發送指令。設定檔案採用 *JSON* 格式。它會被載入並且可以透過 gRPC 傳輸其內容。

另請參閱 ViWriteFromFileRequest 和 ViWriteFromFileResponse 訊息。

關於 *JSON* 設定，請參閱 schema 以獲取更多詳細資訊。

3.3.2 Message

ViReadRequest

- 參數
- bytes ***jobid*** [已棄用]: 不使用。
- int64 ***count***: 表示用於從應用程式接收資料的資料緩衝區大小。

ViReadResponse

- 參數
- AQVI_STATUS ***status_code***: 回傳狀態。
- bytes ***command_response***: 從前一個 ViWrite 指令查詢的回應。
- int64 ***ret_count***: 表示回應的長度。

ViWriteRequest

- 參數
- bytes ***command***: 準備發送到應用程式的位元組陣列指令。

ViWriteResponse

- 參數
- bytes ***jobid***: 表示 viwrite 呼叫的唯一識別碼。
- AQVI_STATUS ***status_code***: 回傳狀態。

Copyright © 2025 Acute Technology Inc.

ViWriteFromFileRequest

- 參數
- bytes ***payload***: 從 JSON 設定檔案載入的位元組陣列內容。 目前支援的結構描述如下：

結構描述名稱	結構描述
電氣驗證	結構描述

ViWriteFromFileResponse

- 參數
- bytes ***jobid***: 表示 ViWriteFromFile 呼叫的唯一識別碼。
- AQVI_STATUS ***status_code***: 回傳狀態。

ViReadToFileRequest

- 參數
- AQVI_JSON_SCHEMA ***schema_type***: 結構描述類型的邏輯索引
- int64 ***count***: 用於接受請求內容的輸入緩衝區大小 目前支援的結構描述如下：

結構描述名稱	結構描述
電氣驗證	結構描述

ViReadToFileResponse

- 參數
- AQVI_STATUS ***status_code***: 回傳狀態。
- bytes ***payload***: 請求的 JSON 內容
- int64 ***ret_count***: 請求的 JSON 內容大小

3.3.3 Enum

AQVI_JSON_SCHEMA

值	列舉	說明
0	AQVI_JSON_SCHEMA_DUMMY	不使用
1	AQVI_JSON_SCHEMA_ELECTRICAL_VALIDATION	電氣驗證設定

AQVI_STATUS

與狀態碼表中的狀態碼用法相同。

Copyright © 2025 Acute Technology Inc.

4. Supported Commands

4.1 Common Commands

4.1.1 指令格式

下表列出常用指令清單。

指令	說明
*IDN?	查詢裝置資訊
*STB?	查詢目前軟體狀態

Copyright © 2025 Acute Technology Inc.

4.1.2 指令定義

***IDN?**

查詢裝置資訊。

- **語法**:

```
*IDN?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回應格式如下：

  Model: <型號名稱>, Serial No.: <裝置序號>;

  如果未連接，則回應固定為：

  Model: Demo, Serial No.: Demo;

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*IDN?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("裝置資訊 = %s\n", buffer);
```

***STB?**

查詢目前軟體狀態。必須在特定指令後使用以檢查處理完成狀態。

Copyright © 2025 Acute Technology Inc.

需要在以下指令後使用：

- *LA:CAPTURE:STOP
- *LA:DECODEREPORT:SAVE
- *LA:TIMINGREPORT:LOADSETTING
- *LA:TIMINGREPORT:SAVE
- *PA:CAPTURE:STOP
- *PA:REPORT:SAVE
- **語法**:

```
*STB?
```

- **參數**:

  無

- **viRead() 回傳值**:

  `0`：軟體忙碌中 `1`：軟體等待指令中

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*STB?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("軟體狀態 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

4.2 File Subsystem

4.2.1 指令格式

以下表格為 FILE 子系統的指令樹狀結構。

指令	說明
*FILE	-
:LOAD	載入波形檔案
:LOADLOG	載入波形記錄檔案的相關指令集
:CONVERT	開始轉換 .LOG 檔案
:ENDTIME?	回傳 .LOG 檔案的結束時間
:SPLITCNT?	回傳 .LOG 檔案的分割數量
:STARTTIME?	回傳 .LOG 檔案的開始時間
:SAVE	儲存波形檔案
:SAVEANALOG	儲存類比波形資料（僅限 MSO 系列）
:SAVEIMAGE	儲存軟體畫面截圖
:SELECTPAGE	選擇子視窗頁面
:CLOSE	關閉子視窗頁面

4.2.2 指令定義

***FILE:LOAD**

載入波形檔案。

- **語法**:

```
*FILE:LOAD <filename>
```

- **參數**:

  <filename>: 要載入的波形檔案路徑。

應用程式	支援的檔案副檔名
MSO	.MSW
BusFinder and Logic Analyzer	.BFW
TravelLogic	.TLW
TravelBus	.TBW

- **範例**:

```
*FILE:LOAD "SampleWaveform.MSW"
```


Copyright © 2025 Acute Technology Inc.

***FILE:LOADLOG:CONVERT**

將波形記錄檔案轉換為波形檔案。

- **語法**:

```
*FILE:LOADLOG:CONVERT <filename> <output_filename> {-INDEX <index>}
```

- **參數**:

  <filename>: 要載入的波形記錄檔案路徑。

  <output_filename>: 輸出波形檔案的路徑。

  -INDEX <index> (選擇性): 要載入的波形索引。若未指定，將轉換完整範圍的波形。

- **viRead() 回傳值**:

  應用程式視窗的寬度和高度。

- **範例**:

  假設有一個記錄檔案（.LOG）準備分割成 3 個較小的波形檔案：

```
// 轉換完整範圍的記錄檔案（在此例中包含 0 - 2）
*FILE:LOADLOG:CONVERT "SampleWaveform.LOG" "SampleWaveform.MSW"
```

```
// 轉換索引為 1 的記錄檔案
*FILE:LOADLOG:CONVERT "SampleWaveform.LOG" "SampleWaveform.MSW" -INDEX 1
```

***FILE:LOADLOG:ENDTIME?**

回傳 .LOG 檔案的結束時間。

- **語法**:

```
*FILE:LOADLOG:ENDTIME? <filename>
```

- **參數**:

  <filename>: 要載入的波形記錄檔案路徑。

- **viRead() 回傳值**:

  .LOG 檔案的結束時間。

- **範例**:

```
*FILE:LOADLOG:ENDTIME? "SampleWaveform.LOG"
```


Copyright © 2025 Acute Technology Inc.

*FILE:LOADLOG:SPLITCNT?

回傳 .LOG 檔案的分割數量。

- **語法**:

```
*FILE:LOADLOG:SPLITCNT? <filename>
```

- **參數**:

  <filename>: 要載入的波形記錄檔案路徑。

- **viRead() 回傳值**:

  .LOG 檔案的分割數量。

- **範例**:

```
*FILE:LOADLOG:SPLITCNT? "SampleWaveform.LOG"
```

*FILE:LOADLOG:STARTTIME?

回傳 .LOG 檔案的開始時間。

- **語法**:

```
*FILE:LOADLOG:STARTTIME? <filename>
```

- **參數**:

  <filename>: 要載入的波形記錄檔案路徑。

- **viRead() 回傳值**:

  .LOG 檔案的開始時間。

- **範例**:

```
*FILE:LOADLOG:STARTTIME? "SampleWaveform.LOG"
```


Copyright © 2025 Acute Technology Inc.

***FILE:SAVE**

儲存波形檔案。

- **語法**:

```
// 儲存波形檔案
*FILE:SAVE -WAVEFILE <filename>
```

```
// 儲存轉換文字檔案
*FILE:SAVE -TRANSTEXT <filename> [-NOTS] [-NOHEADER] [-TIMESEC]
```

```
// 儲存文字檔案，檔案大小可能比 -TRANSTEXT 大很多
*FILE:SAVE -SAMPLETEXT <filename> [-NOTS] [-NOHEADER] [-TIMESEC]
```

- **參數**:

  -WAVEFILE <filename>: 要儲存的波形檔案路徑。

  -TRANSTEXT <filename>: 要儲存的轉換文字檔案路徑。

  -SAMPLETEXT <filename>: 要儲存的文字檔案路徑。

  -NOTS (選擇性): 不儲存時間戳記。

  -NOHEADER (選擇性): 不儲存標頭。

  -TIMESEC (選擇性): 以秒為單位儲存時間戳記。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將波形檔案儲存為 .MSW
*FILE:SAVE -WAVEFILE "SampleWaveform.MSW"
```

```
// 將轉換文字檔案儲存為 .TXT
*FILE:SAVE -TRANSTEXT "SampleWaveform.TXT"
```

```
// 將文字檔案儲存為 .TXT，不包含時間戳記
*FILE:SAVE -SAMPLETEXT "SampleWaveform.TXT" -NOTS
```


Copyright © 2025 Acute Technology Inc.

***FILE:SAVEANALOG**

儲存類比波形資料（僅限 MSO 系列）。

- **語法**:

```
*FILE:SAVEANALOG <filename> -SAVEONFILE -CHSEL <channel_bits> [-NOTS]
```

```
*FILE:SAVEANALOG <filename> -SAVEEACHFILE -CHSEL <channel_bits> [-NOTS]
```

- **參數**:

  <filename>: 要儲存的波形檔案路徑。

  -SAVEONFILE: 將波形資料儲存到單一檔案。

  -SAVEEACHFILE: 為每個通道儲存一個波形資料檔案。

  -CHSEL <channel_bits>: 輸入 -CHSEL 並以十進位位元 OR 通道選擇旗標來選擇目標輸出通道。例如：輸入 -CHSEL 31 以輸出 CH0 到 CH4 的類比資料。

  -NOTS (選擇性): 不儲存時間戳記。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將 CH0 和 CH1 的資料與時間戳記資訊儲存到一個檔案中。
// 輸出檔案：File.csv
viWrite("*FILE:SAVEANALOG File.csv -SAVEONEFILE -CHSEL 3");
```

```
// 將 CH1 和 CH2 的資料不含時間戳記資訊儲存到 2 個檔案中。
// 輸出檔案：File_DSO CH1.csv 和 File_DSO CH2.csv
viWrite("*FILE:SAVEANALOG File.csv -SAVEEACHFILE -CHSEL 6 -NOTS");
```


Copyright © 2025 Acute Technology Inc.

*FILE:SAVEIMAGE

儲存軟體畫面截圖。

- **語法**:

```
*FILE:SAVEIMAGE <filename>
```

- **參數**:

  <filename>: 要儲存的圖片檔案路徑。

  支援的檔案副檔名：.png、.jpg、.bmp

- **viRead() 回傳值**:

  無

- **範例**:

```
// 儲存為 BMP 檔案
*FILE:SAVEIMAGE "SampleImage.bmp"
```

```
// 儲存為 JPG 檔案
*FILE:SAVEIMAGE "SampleImage.jpg"
```

```
// 儲存為 PNG 檔案
*FILE:SAVEIMAGE "SampleImage.png"
```

*FILE:SELECTPAGE

選擇波形檔案的頁面。

- **語法**:

```
*FILE:SELECTPAGE <page_number>
```

- **參數**:

  <page_number>: 要選擇的子視窗頁面編號。第一頁的索引從 0 開始。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 選擇第一個子視窗頁面
viWrite("*FILE:SELECTPAGE 0");
```

```
// 選擇第三個子視窗頁面
viWrite("*FILE:SELECTPAGE 2");
```


Copyright © 2025 Acute Technology Inc.

***FILE:CLOSE**

關閉子視窗頁面。

- **語法**:

```
// 關閉所有子視窗頁面
*FILE:CLOSE -ALL
```

```
// 關閉目前的子視窗頁面
*FILE:CLOSE -CURRENT
```

- **參數**:

  無

- **範例**:

```
// 關閉目前的子視窗頁面
viWrite("*FILE:CLOSE -CURRENT");
```

```
// 關閉所有子視窗頁面
viWrite("*FILE:CLOSE -ALL");
```


Copyright © 2025 Acute Technology Inc.

4.3 Window Subsystem

4.3.1 指令格式

以下表格為 WINDOW 子系統的指令樹狀結構。

指令	說明
*WINDOW	-
:SIZE	設定應用程式視窗大小
:SIZE?	查詢應用程式視窗大小
:SCREEN	設定應用程式視窗在指定螢幕上的索引
:SCREEN?	查詢應用程式視窗在指定螢幕上的索引

4.3.2 指令定義

***WINDOW:SIZE**

設定應用程式視窗大小。

- **語法**:

```
*WINDOW:SIZE <width> <height>
```

- **參數**:

  <width>: 設定應用程式視窗寬度，例如：1000

  <height>: 設定應用程式視窗高度，例如：800

- **範例**:

```
*WINDOW:SIZE 1000 800
```

***WINDOW:SIZE?**

查詢應用程式視窗大小。

- **語法**:

```
*WINDOW:SIZE?
```

- **參數**:

  無

- **viRead() 回傳值**:

  應用程式視窗的寬度和高度。

- **範例**:

```
*WINDOW:SIZE?
```


Copyright © 2025 Acute Technology Inc.

*WINDOW:SCREEN

設定應用程式視窗在指定螢幕上的索引。

- **語法**:

```
*WINDOW:SCREEN <screenIndex>
```

- **參數**:

  <screenIndex>: 設定應用程式視窗在指定螢幕上的索引，例如：1

- **viRead() 回傳值**:

  無

- **範例**:

```
*WINDOW:SCREEN 1
```

*WINDOW:SCREEN?

查詢應用程式視窗在指定螢幕上的索引。

- **語法**:

```
*WINDOW:SCREEN?
```

- **參數**:

  無

- **viRead() 回傳值**:

  應用程式視窗在指定螢幕上的索引。

- **範例**:

```
*WINDOW:SCREEN?
```


Copyright © 2025 Acute Technology Inc.

4.4 Logic Analyzer Mode Subsystem

4.4.1 什麼是邏輯分析儀模式？

邏輯分析儀模式允許您擷取和分析數位訊號。 它為各種數位協定（如 I2C、SPI、UART 等）提供專門的解碼和分析功能。 邏輯分析儀模式通過將原始訊號解碼為特定協定的交易內容，並以易於理解的格式呈現，幫助工程師除錯和驗證數位通訊。

4.4.2 指令格式

本節描述邏輯分析儀子系統的指令格式。每個邏輯分析儀子系統的指令都以 ***LA:** 開頭。

Copyright © 2025 Acute Technology Inc.

以下表格為邏輯分析儀子系統的指令樹狀結構。

指令	說明
*LA:CAPTURE	用於擷取和分析協定資料的指令集
:START	開始波形擷取
:STOP	停止波形擷取
:PROGRESS?	讀取擷取進度
:TRIGGERED?	讀取擷取觸發狀態
:STATUS?	讀取擷取狀態
*LA:CURSOR	游標相關的指令集
:DELETE	刪除選定的游標
:SEARCH	游標搜尋的相關指令集
:SET	設定游標移動/搜尋條件的相關指令集
:BYEDGE	設定游標移動/搜尋條件為邊緣轉換
:BYTIME	設定游標移動/搜尋條件為時間週期
:BYVALUEMATCH	設定游標移動/搜尋條件為數值或文字匹配
:SEARCHPULSEWIDTH	設定游標移動/搜尋條件為脈衝寬度
:START	開始游標移動/搜尋
*LA:WAVEFORM	波形相關的指令集
:SCALE	設定波形水平時間刻度
:MOVETO	移動波形中心到指定游標位置
:STORE	儲存波形位置或時間刻度
:RESTORE	還原波形位置或時間刻度
*LA:WINDOWSIZE	視窗大小相關的指令集
:WAVEFORMSIZE?	查詢波形區域大小
:REPORTSIZE?	查詢報表區域大小

Copyright © 2025 Acute Technology Inc.

指令	說明
*LA:DECODEREPORT	解碼報表資料的相關指令集
:COLUMNCOUNT?	查詢目前選定解碼報表的欄數
:COUNT?	查詢報表總數
:DATA?	查詢選定索引的報表資料
:NAME?	查詢目前選定解碼報表的名稱
:ROWCOUNT?	查詢目前選定解碼報表的列數
:SAVE	儲存目前選定的解碼報表到檔案
:SELECT	選擇解碼報表清單中的索引
*LA:MEASUREMENTREPORT	量測報表資料的相關指令集
:COLUMNCOUNT?	查詢目前選定量測報表的欄數
:COLUMNHEADER?	查詢指定索引欄的名稱
:DATA?	查詢指定列和欄索引的資料
:REFRESH	重新計算量測報表
:ROWCOUNT?	查詢目前選定量測報表的列數
:SAVE	儲存目前選定的量測報表到檔案
*LA:TIMINGREPORT	時序報表資料的相關指令集
:COLUMNCOUNT?	查詢目前選定時序報表的欄數
:COLUMNHEADER?	查詢指定索引欄的名稱
:DATA?	查詢指定列和欄索引的資料
:LOADRESULT?	載入時序報表處理的結果
:LOADSETTING	載入時序報表的設定
:ROWCOUNT?	查詢目前選定時序報表的列數
:SAVE	儲存目前選定的時序報表到檔案
:TESTRESULT?	測試處理的結果
*LA:TRANSITIONREPORT	轉換報表資料的相關指令集
:COLUMNCOUNT?	查詢目前選定轉換報表的欄數
:COLUMNHEADER?	查詢指定索引欄的名稱
:CURSORPOSITION?	查詢游標位置
:DATA?	查詢指定列和欄索引的資料
:ROWCOUNT?	查詢目前選定轉換報表的列數
:SAVE	儲存目前選定的轉換報表到檔案
:SELECT	選擇轉換報表清單中的索引

Copyright © 2025 Acute Technology Inc.

4.4.3 指令定義

***LA:CAPTURE:START**

開始波形擷取。

- **語法**:

```
*LA:CAPTURE:START
```

- **參數**:

  無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_CAPTURE_ALREADY_RUNNING	表示擷取程序已在執行中。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 開始邏輯分析儀擷取
viWrite("*LA:CAPTURE:START");
```

***LA:CAPTURE:STOP**

停止波形擷取。

- **語法**:

```
*LA:CAPTURE:STOP
```

- **參數**: 無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_CAPTURE_NOT_RUNNING	表示應用程式尚未開始擷取。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 停止邏輯分析儀擷取
viWrite("*LA:CAPTURE:STOP");
```


Copyright © 2025 Acute Technology Inc.

***LA:CAPTURE:PROGRESS?**

讀取擷取進度。

- **語法**:

```
*LA:CAPTURE:PROGRESS?
```

- **參數**:

  無

- **viRead() 回傳值**:

  表示目前擷取進度的值，範圍從 0.00 到 1.00。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:CAPTURE:PROGRESS?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("擷取進度 = %f\n", atof(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:CAPTURE:TRIGGERED?**

讀取擷取觸發狀態。

- **語法**:

```
*LA:CAPTURE:TRIGGERED?
```

- **參數**:

  無

- **viRead() 回傳值**:

  `1`: 已發生觸發事件。

  `0`: 尚未發生觸發事件。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:CAPTURE:TRIGGERED?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("擷取觸發狀態 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:CAPTURE:STATUS?**

讀取擷取狀態。

Copyright © 2025 Acute Technology Inc.

- **語法:**

```
*LA:CAPTURE:STATUS?
```

- **參數:**

  無

- **viRead() 回傳值:**

  表示目前擷取狀態的字串。

狀態	說明
Data Transferring	資料傳輸中。
Data Processing	資料處理中。
Capture Finished	邏輯分析儀已完成資料擷取。
Analyzing	邏輯分析儀正在分析資料。
Analysis Finished	邏輯分析儀已完成資料分析。
Pre-Trigger	邏輯分析儀進入預觸發狀態。
Wait For Trigger	邏輯分析儀正在等待觸發條件。
Post-Trigger	邏輯分析儀進入後觸發狀態。
Wait For DSO Capture	邏輯分析儀正在等待 DSO 擷取。
Capturing	邏輯分析儀正在擷取資料。
Saving Report	邏輯分析儀正在儲存報表。
Report Save Finished	邏輯分析儀已完成儲存報表。
File Load Finished	邏輯分析儀已完成載入檔案。

- **範例:**

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:CAPTURE:STATUS?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}
```


Copyright © 2025 Acute Technology Inc.

```
// 顯示結果
printf("擷取狀態 = %s\n", buffer);
```

*LA:CURSOR:DELETE

刪除選定的游標。

- **語法**:

```
// 刪除所有游標
*LA:CURSOR:DELETE -ALL
```

```
// 刪除指定游標
*LA:CURSOR:DELETE <cursorN>
```

- **參數**:

  -ALL: 刪除所有游標。

  <cursorN>: 刪除指定游標。

- **viRead() 回傳值**: 無
- **範例**:

```
// 刪除所有游標
viWrite("*LA:CURSOR:DELETE -ALL");
```

```
// 刪除游標 B
viWrite("*LA:CURSOR:DELETE CursorB");
```


Copyright © 2025 Acute Technology Inc.

*LA:CURSOR:SEARCH:SET:BYEDGE

設定游標移動/搜尋條件為邊緣轉換。

- **語法**:

```
*LA:CURSOR:SEARCH:SET:BYEDGE <label> <edgeCount> <edgeType>
```

- **參數**:

  <label>: 設定目標標籤，例如：CH-04

  <edgeCount>: 設定條件邊緣計數。

  <edgeType>: 設定條件邊緣類型。

- **viRead() 回傳值**: 無
- **範例**:

```
// 設定游標移動/搜尋條件為 CH-05 上升邊緣 x8
viWrite("*LA:CURSOR:SEARCH:SET:BYEDGE CH-05 8 RISE");
```

```
// 設定游標移動/搜尋條件為 CH-15 下降邊緣 x10
viWrite("*LA:CURSOR:SEARCH:SET:BYEDGE CH-15 10 FALL");
```

*LA:CURSOR:SEARCH:SET:BYTIME

設定游標移動/搜尋條件為時間週期。

- **語法**:

```
*LA:CURSOR:SEARCH:SET:BYTIME <time>
```

- **參數**:

  <time>: 設定時間條件，例如：10ns、50ms

- **viRead() 回傳值**: 無
- **範例**:

```
// 設定游標移動/搜尋條件為 50.5us
viWrite("*LA:CURSOR:SEARCH:SET:BYTIME 50.5us");
```


Copyright © 2025 Acute Technology Inc.

*LA:CURSOR:SEARCH:SET:BYVALUEMATCH

設定游標移動/搜尋條件為數值或文字匹配。

- **語法**:

```
*LA:CURSOR:SEARCH:SET:BYVALUEMATCH <label> <value>
```

- **參數**:

  <label>: 設定目標標籤，例如：CH-04

  <value>: 設定數值/文字匹配條件，例如：05h 或 "Data Read"

- **viRead() 回傳值**: 無
- **範例**:

```
// 設定游標移動/搜尋條件為數值 50h（十六進制）
viWrite("*LA:CURSOR:SEARCH:SET:BYVALUEMATCH 50h");
```

```
// 設定游標移動/搜尋條件為文字匹配 "Data Read"
viWrite("*LA:CURSOR:SEARCH:SET:BYVALUEMATCH \"Data Read\"");
```


Copyright © 2025 Acute Technology Inc.

*LA:CURSOR:SEARCH:SET:SEARCHPULSEWIDTH

設定游標移動/搜尋條件為脈衝寬度。

- **語法**:

```
*LA:CURSOR:SEARCH:SET:SEARCHPULSEWIDTH <label> <pulseType> G|L|GE|LE|E|NE <time>

*LA:CURSOR:SEARCH:SET:SEARCHPULSEWIDTH <label> <pulseType> INRANGE|OUTRANGE <time1> <time2>
```

- **參數**:

  <label>: 設定目標標籤，例如：CH-04

  <pulseType>: 設定脈衝類型為高電位、低電位或任一。

  <conditionType>:

```
- G: 大於
- L: 小於
- GE: 大於或等於
- LE: 小於或等於
- E: 等於
- NE: 不等於
- INRANGE: 在範圍內，需要指定兩個時間條件。
- OUTRANGE: 在範圍外，需要指定兩個時間條件。
```

  <time>: 設定時間條件，例如：10ns、50ms

- **viRead() 回傳值**: 無

- **範例**:

```
// 設定游標移動/搜尋條件為 CH-03 正脈衝寬度為 10 us
viWrite("*LA:CURSOR:SEARCH:SET:SEARCHPULSEWIDTH CH-03 HIGH L 10us");
```

```
// 設定游標移動/搜尋條件為 CH-13 任意脈衝寬度在 1 ms 到 10 ms 之間
viWrite("*LA:CURSOR:SEARCH:SET:SEARCHPULSEWIDTH CH-13 ALL INRANGE 1ms 10ms");
```


Copyright © 2025 Acute Technology Inc.

*LA:CURSOR:SEARCH:START

開始游標移動/搜尋。

- **語法**:

```
*LA:CURSOR:SEARCH:START <direction> <cursor> [MULTIPLE <count>]
```

- **參數**:

  <direction>: 設定搜尋方向為左或右。

  <cursor>: 設定目標游標，例如：CursorA、CursorB、...、CursorZ

  MULTIPLE <count>: 設定移動多個游標，需要指定額外的游標移動數量，例如：5。

- **viRead() 回傳值**: 無
- **範例**:

```
// 從游標 C 開始，向右搜尋並放置游標 C 到 E。
viWrite("*LA:CURSOR:SEARCH:START RIGHT CursorC MULTIPLE 3");
```

*LA:WAVEFORM:SCALE

設定波形水平時間刻度。

- **語法**:

```
*LA:WAVEFORM:SCALE <time>
```

- **參數**:

  time: 水平波形時間刻度，單位：ps、ns、us、ms、s。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 設定波形時間刻度為 10 us。
viWrite("*LA:WAVEFORM:SCALE 10us");
```


Copyright © 2025 Acute Technology Inc.

*LA:WAVEFORM:MOVETO

移動波形中心到指定游標位置。

- **語法**:

```
*LA:WAVEFORM:MOVETO <cursor>
```

- **參數**:

  cursor: 游標名稱，例如：CursorA、CursorB、...、CursorZ

- **viRead() 回傳值**: 無
- **範例**:

```
// 移動波形中心點到游標 C 的位置。
viWrite("*LA:WAVEFORM:MOVETO CURSORC");
```

*LA:WAVEFORM:STORE

儲存目前波形位置或時間刻度。

- **語法**:

```
*LA:WAVEFORM:STORE [-SCALE] [-POSITION]
```

- **參數**:

  -SCALE: 儲存目前波形時間刻度。

  -POSITION: 儲存目前波形位置。

- **viRead() 回傳值**: 無
- **範例**:

```
// 儲存目前波形位置
viWrite("*LA:WAVEFORM:STORE -POSITION");
```

```
// 儲存目前波形時間刻度
viWrite("*LA:WAVEFORM:STORE -SCALE");
```

```
// 同時儲存目前波形位置和時間刻度
viWrite("*LA:WAVEFORM:STORE -POSITION -SCALE");
```


Copyright © 2025 Acute Technology Inc.

*LA:WAVEFORM:RESTORE

還原先前的波形位置或時間刻度。

- **語法**:

```
*LA:WAVEFORM:RESTORE [-SCALE] [-POSITION]
```

- **參數**:

  -SCALE: 還原先前的波形時間刻度。

  -POSITION: 還原先前的波形位置。

- **viRead() 回傳值**: 無
- **範例**:

```
// 還原先前的波形位置
viWrite("*LA:WAVEFORM:RESTORE -POSITION");
```

```
// 還原先前的波形時間刻度
viWrite("*LA:WAVEFORM:RESTORE -SCALE");
```

```
// 同時還原先前的波形位置和時間刻度
viWrite("*LA:WAVEFORM:RESTORE -POSITION -SCALE");
```

*LA:WINDOWSIZE:REPORTSIZE

設定報表區域大小，同時也會改變波形區域大小。

- **語法**:

```
*LA:WINDOWSIZE:REPORTSIZE <size>
```

- **參數**:

  SIZE: 指定報表區域大小，範圍從 0 到 100（單位：百分比）。

- **viRead() 回傳值**: 無
- **範例**:

```
// 設定報表區域大小為螢幕的 50%
viWrite("*LA:WINDOWSIZE:REPORTSIZE 50");
```


Copyright © 2025 Acute Technology Inc.

*LA:WINDOWSIZE:WAVEFORMSIZE

設定波形區域大小，同時也會改變報表區域大小。

- **語法**:

```
*LA:WINDOWSIZE:WAVEFORMSIZE <size>
```

- **參數**:

  SIZE: 指定波形區域大小，範圍從 0 到 100（單位：百分比）。

- **viRead() 回傳值**: 無
- **範例**:

```
// 設定波形區域大小為螢幕的 50%
viWrite("*LA:WINDOWSIZE:WAVEFORMSIZE 50");
```


Copyright © 2025 Acute Technology Inc.

***LA:DECODEREPORT:COLUMNCOUNT?**

取得目前解碼報表的欄數。

- **語法**:

```
*LA:DECODEREPORT:COLUMNCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  目前解碼報表的欄數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:DECODEREPORT:COLUMNCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("解碼報表欄數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:DECODEREPORT:COUNT?**

取得解碼報表數量。

- **語法**:

```
*LA:DECODEREPORT:COUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  目前解碼報表數量。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:DECODEREPORT:COUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("解碼報表數量 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:DECODEREPORT:DATA?**

取得指定索引的解碼報表資料。

- **語法**:

```
*LA:DECODEREPORT:DATA? <row> <column>
```

- **參數**:

  row: 解碼報表列索引。

  column: 解碼報表欄索引。

- **viRead() 回傳值**:

  選定索引的解碼報表資料。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:DECODEREPORT:DATA? 2 1")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

printf("索引 (2, 1) 的解碼報表資料 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

***LA:DECODEREPORT:NAME?**

取得目前選定的解碼報表名稱。

- **語法**:

```
*LA:DECODEREPORT:NAME?
```

- **參數**:

  無

- **viRead() 回傳值**:

  目前選定的解碼報表名稱。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:DECODEREPORT:NAME?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("解碼報表名稱 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

***LA:DECODEREPORT:ROWCOUNT?**

取得目前解碼報表的列數。

- **語法**:

```
*LA:DECODEREPORT:ROWCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  目前解碼報表的列數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:DECODEREPORT:ROWCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("解碼報表列數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*LA:DECODEREPORT:SAVE

儲存解碼報表。

- **語法**:

```
*LA:DECODEREPORT:SAVE <filename> [-BEGIN <row>] [-END <row>] [-NOTS] [-USESEC] [-NOHEADER]
```

- **參數**:

  filename: 目標檔案名稱。

  -BEGIN <row> (選擇性): 輸入報表儲存的起始列，預設從報表開頭開始。

  -END <row> (選擇性): 輸入報表儲存的結束列，預設到報表結尾結束。

  -NOTS (選擇性): 報表儲存功能會在儲存報表的第一欄儲存時間戳記資訊，加入此參數可指定不要將時間戳記資訊儲存到報表中。

  -USESEC (選擇性): 報表儲存功能會以奈秒為單位儲存時間戳記資訊，加入此參數可指定使用秒來儲存時間戳記資訊。

  -NOHEADER (選擇性): 指定不要在儲存的報表中包含標頭資訊。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將報表儲存為 MyDecodeReport.csv
viWrite("*LA:DECODEREPORT:SAVE MyDecodeReport.csv");
```

> **Note**
>
> 請確保您有權限寫入目標檔案目錄。

*LA:DECODEREPORT:SELECT

選擇解碼報表。

- **語法**:

```
*LA:DECODEREPORT:SELECT <index>
```

- **參數**:

  index: 解碼報表索引。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 選擇第一個解碼報表
viWrite("*LA:DECODEREPORT:SELECT 0");
```


Copyright © 2025 Acute Technology Inc.

***LA:MEASUREMENTREPORT:COLUMNCOUNT?**

取得量測報表欄數。

- **語法**:

```
*LA:MEASUREMENTREPORT:COLUMNCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳目前量測報表欄數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:MEASUREMENTREPORT:COLUMNCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("量測報表欄數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*LA:MEASUREMENTREPORT:COLUMNHEADER?

取得指定欄索引的量測報表欄標題名稱。

- **語法**:

```
*LA:MEASUREMENTREPORT:COLUMNHEADER? <column>
```

- **參數**:

  column: 指定欄索引。

- **viRead() 回傳值**:

  回傳目前量測報表指定欄索引的欄標題名稱。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:MEASUREMENTREPORT:COLUMNHEADER? 1")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 1 的量測報表欄標題 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

*LA:MEASUREMENTREPORT:DATA?

取得指定列和欄索引的量測報表資料。

- **語法**:

```
*LA:MEASUREMENTREPORT:DATA? <row> <column>
```

- **參數**:

  row: 指定列索引。

  column: 指定欄索引。

- **viRead() 回傳值**:

  指定列和欄索引的量測報表資料。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:MEASUREMENTREPORT:DATA? 1 2")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 [1, 2] 的量測報表資料 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

*LA:MEASUREMENTREPORT:REFRESH

重新計算量測報表。

- **語法**:

```
*LA:MEASUREMENTREPORT:REFRESH
```

- **參數**:

  無

- **viRead() 回傳值**:

  無

- **範例**:

```
// 重新計算量測報表
viWrite("*LA:MEASUREMENTREPORT:REFRESH");
```


Copyright © 2025 Acute Technology Inc.

*LA:MEASUREMENTREPORT:ROWCOUNT?

取得量測報表列數。

- **語法**:

```
*LA:MEASUREMENTREPORT:ROWCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳目前量測報表列數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:MEASUREMENTREPORT:ROWCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("量測報表列數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*LA:MEASUREMENTREPORT:SAVE

將量測報表儲存到檔案。

- **語法**:

```
*LA:MEASUREMENTREPORT:SAVE <filename> [-BEGIN <row>] [-END <row>] [-NOTS] [-USESEC] [-NOHEADER]
```

- **參數**:

  filename: 目標檔案名稱。

  -BEGIN <row> (選擇性): 輸入報表儲存的起始列，預設從報表開頭開始。

  -END <row> (選擇性): 輸入報表儲存的結束列，預設到報表結尾結束。

  -NOTS (選擇性): 報表儲存功能會在儲存報表的第一欄儲存時間戳記資訊，加入此參數可指定不要將時間戳記資訊儲存到報表中。

  -USESEC (選擇性): 報表儲存功能會以奈秒為單位儲存時間戳記資訊，加入此參數可指定使用秒來儲存時間戳記資訊。

  -NOHEADER (選擇性): 指定不要在儲存的報表中包含標頭資訊。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將量測報表儲存為 .csv 檔案到 MyMeasurementReport.csv
viWrite("*LA:MEASUREMENTREPORT:SAVE MyMeasurementReport.csv");
```

```
// 將量測報表儲存為 .html 檔案到 MyMeasurementReport.html
viWrite("*LA:MEASUREMENTREPORT:SAVE MyMeasurementReport.html");
```

> **Note**
>
> 請確保您有權限寫入目標檔案目錄。

Copyright © 2025 Acute Technology Inc.

***LA:TIMINGREPORT:COLUMNCOUNT?**

取得時序報表欄數。

- **語法**:

```
*LA:TIMINGREPORT:COLUMNCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  目前時序報表欄數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:COLUMNCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("時序報表欄數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:TIMINGREPORT:COLUMNHEADER?**

取得時序報表欄標題。

- **語法**:

```
*LA:TIMINGREPORT:COLUMNHEADER? <column>
```

- **參數**:

  column: 時序報表欄索引。

- **viRead() 回傳值**:

  目前時序報表欄標題。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:COLUMNHEADER? 1")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 1 的時序報表欄標題 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

*LA:TIMINGREPORT:DATA?

取得指定列和欄索引的時序報表資料。

- **語法**:

```
*LA:TIMINGREPORT:DATA? <row> <column>
```

- **參數**:

  row: 時序報表列索引。

  column: 時序報表欄索引。

- **viRead() 回傳值**:

  指定列和欄索引的時序報表資料。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:DATA? 1 2")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 [1, 2] 的時序報表資料 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

***LA:TIMINGREPORT:LOADRESULT?**

載入時序報表結果。

- **語法**:

```
*LA:TIMINGREPORT:LOADRESULT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳 *LA:TIMINGREPORT:LOADSETTING 的處理結果， 如果沒有錯誤則回傳 "OK"，否則回傳錯誤資訊。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:LOADRESULT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("時序報表載入結果 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

*LA:TIMINGREPORT:LOADSETTING

載入時序報表設定。

- **語法**:

```
*LA:TIMINGREPORT:LOADSETTING <filename>
```

- **參數**:

  filename: 時序報表設定檔案名稱。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 載入時序報表設定
viWrite("*LA:TIMINGREPORT:LOADSETTING MyTimingSettings.csv");
```

> **Note**
>
> 請確保您有權限寫入目標檔案目錄。

Copyright © 2025 Acute Technology Inc.

*LA:TIMINGREPORT:ROWCOUNT?

取得時序報表列數。

- **語法**:

```
*LA:TIMINGREPORT:ROWCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳目前時序報表列數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:ROWCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("時序報表列數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*LA:TIMINGREPORT:SAVE

將時序報表儲存到檔案。

- **語法**:

```
// 將時序報表儲存為 .csv 檔案
*LA:TIMINGREPORT:SAVE -CSV <filename>
```

```
// 將時序報表儲存為 .txt 檔案
*LA:TIMINGREPORT:SAVE -TXT <filename>
```

```
// 將時序報表儲存為 .html 檔案
*LA:TIMINGREPORT:SAVE -HTML <filename> [-TITLE <title>] [-USERINPUT <userInput>]
```

- **參數**:

  filename: 目標檔案名稱。

  -CSV: 將報表儲存為 .csv 檔案。

  -TXT: 將報表儲存為 .txt 檔案。

  -HTML: 將報表儲存為 .html 檔案。

  -TITLE <title> (選擇性): 指定 HTML 報表標題。

  -USERINPUT <userInput> (選擇性): 指定使用者輸入資訊。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將時序報表儲存為 .csv 檔案到 MyTimingReport.csv
viWrite("*LA:TIMINGREPORT:SAVE –CSV MyTimingReport.csv");
```

```
// 將時序報表儲存為 .html 檔案到 MyTimingReport.html 並加入標題
viWrite("*LA:TIMINGREPORT:SAVE –HTML MyTimingReport.html –TITLE My Example Title");
```

> **Note**
>
> 請確保您有權限寫入目標檔案目錄。

Copyright © 2025 Acute Technology Inc.

*LA:TIMINGREPORT:TESTRESULT?

從時序報表取得整體通過/失敗結果。

- **語法**:

```
*LA:TIMINGREPORT:TESTRESULT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  如果所有項目都測試通過則回傳 `Pass`，否則回傳 `Fail`。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TIMINGREPORT:TESTRESULT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("時序報表測試結果 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

***LA:TRANSITIONREPORT:COLUMNCOUNT?**

取得轉換報表欄數。

- **語法**:

```
*LA:TRANSITIONREPORT:COLUMNCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳目前轉換報表欄數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TRANSITIONREPORT:COLUMNCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("轉換報表欄數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

***LA:TRANSITIONREPORT:COLUMNHEADER?**

取得指定欄索引的轉換報表欄標題名稱。

- **語法**:

```
*LA:TRANSITIONREPORT:COLUMNHEADER? <column>
```

- **參數**:

  <column>: 指定欄索引。

- **viRead() 回傳值**:

  回傳目前轉換報表指定欄索引的欄標題名稱。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TRANSITIONREPORT:COLUMNHEADER? 1")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 [1] 的轉換報表欄標題名稱 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

*LA:TRANSITIONREPORT:DATA?

取得指定列和欄索引的轉換報表資料。

- **語法**:

```
*LA:TRANSITIONREPORT:DATA? <row> <column>
```

- **參數**:

  row: 指定列索引。

  column: 指定欄索引。

- **viRead() 回傳值**:

  指定列和欄索引的轉換報表資料。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TRANSITIONREPORT:DATA? 1 2")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 [1, 2] 的轉換報表資料 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

***LA:TRANSITIONREPORT:ROWCOUNT?**

取得轉換報表列數。

- **語法**:

```
*LA:TRANSITIONREPORT:ROWCOUNT?
```

- **參數**:

  無

- **viRead() 回傳值**:

  回傳目前轉換報表列數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TRANSITIONREPORT:ROWCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("轉換報表列數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*LA:TRANSITIONREPORT:SAVE

將轉換報表儲存到檔案。

- **語法**:

```
*LA:TRANSITIONREPORT:SAVE <filename> [-BEGIN <row>] [-END <row>] [-NOTS] [-USESEC] [-NOHEADER]
```

- **參數**:

  filename: 目標檔案名稱。

  -BEGIN <row> (選擇性): 輸入報表儲存的起始列，預設從報表開頭開始。

  -END <row> (選擇性): 輸入報表儲存的結束列，預設到報表結尾結束。

  -NOTS (選擇性): 報表儲存功能會在儲存報表的第一欄儲存時間戳記資訊，加入此參數可指定不要將時間戳記資訊儲存到報表中。

  -USESEC (選擇性): 報表儲存功能會以奈秒為單位儲存時間戳記資訊，加入此參數可指定使用秒來儲存時間戳記資訊。

  -NOHEADER (選擇性): 指定不要在儲存的報表中包含標頭資訊。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將轉換報表儲存為 .csv 檔案到 MyTransitionReport.csv
viWrite("*LA:TRANSITIONREPORT:SAVE MyTransitionReport.csv");
```

```
// 將轉換報表儲存為 .html 檔案到 MyTransitionReport.html
viWrite("*LA:TRANSITIONREPORT:SAVE MyTransitionReport.html");
```

> **Note**
>
> 請確保您有權限寫入目標檔案目錄。

Copyright © 2025 Acute Technology Inc.

*LA:TRANSITIONREPORT:SELECT

選擇標籤並重新整理轉換報表。

- **語法**:

```
// 選擇所有標籤
*LA:TRANSITIONREPORT:SELECT -ALL
```

```
// 選擇特定標籤
*LA:TRANSITIONREPORT:SELECT <label1> <label2> ... <labelN>
```

- **參數**:

-ALL: 選擇所有標籤。

<labelN>: 選擇特定標籤。

- **viRead() 回傳值**:

無

- **範例**:

```
// 選擇標籤並重新整理轉換報表
viWrite("*LA:TRANSITIONREPORT:SELECT");
```

```
// 選擇所有標籤以建立轉換報表
viWrite("*LA:TRANSITIONREPORT:SELECT -ALL");
```

```
// 選擇標籤 CH-00、Bus[7:0] 和 CH50 以建立轉換報表
viWrite("*LA:TRANSITIONREPORT:SELECT CH-00 Bus[7:0] CH50");
```


Copyright © 2025 Acute Technology Inc.

*LA:TRANSITIONREPORT:CURSORPOSITION?

取得指定游標所在的轉換報表列號。

- **語法**:

```
*LA:TRANSITIONREPORT:CURSORPOSITION? <cursor>
```

- **參數**:

  <cursor>: 指定游標名稱，例如：CursorA、CursorB、...、CursorZ

- **viRead() 回傳值**:

  回傳目前指定游標所在的轉換報表列號。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*LA:TRANSITIONREPORT:CURSORPOSITION? CursorA")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("轉換報表游標位置 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

4.5 Protocol Analyzer Mode Subsystem

4.5.1 什麼是協定分析儀模式？

協定分析儀模式允許您擷取和分析序列協定資料。 它為各種序列協定（如 I2C、SPI、UART 等）提供專門的解碼和分析功能。 協定分析儀模式通過將原始訊號解碼為特定協定的交易內容，並以易於理解的格式呈現，幫助工程師除錯和驗證序列通訊。

主要功能包括：

- 即時協定訊號解碼
- 詳細的交易檢視與統計
- 資料匯出和報表

4.5.2 指令格式

本節描述協定分析儀子系統的指令格式。 協定分析儀子系統的每個指令都以 ***PA:** 開頭。

以下表格為協定分析儀子系統的指令樹狀結構。

指令	說明
*PA:CAPTURE	用於擷取和分析協定資料的指令集
:START	開始波形擷取
:STOP	停止波形擷取
*PA:REPORT	用於查詢和儲存報表資料的指令集
:ROWCOUNT?	取得報表列數
:COLUMNCOUNT?	取得報表欄數
:SAVE	儲存報表至檔案
:DATA?	取得指定索引的報表資料

Copyright © 2025 Acute Technology Inc.

4.5.3 指令定義

***PA:CAPTURE:START**

開始協定分析儀擷取。

- **語法**:

```
*PA:CAPTURE:START
```

- **參數**:

  無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_CAPTURE_ALREADY_RUNNING	表示擷取程序已在執行中。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 開始協定分析儀擷取
viWrite("*PA:CAPTURE:START");
```

***PA:CAPTURE:STOP**

停止協定分析儀擷取。

- **語法**:

```
*PA:CAPTURE:STOP
```

- **參數**:

  無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_CAPTURE_NOT_RUNNING	表示應用程式尚未開始擷取。

- **查詢值**:

  無

- **範例**:

```
// 停止協定分析儀擷取
viWrite("*PA:CAPTURE:STOP");
```


Copyright © 2025 Acute Technology Inc.

*PA:REPORT:ROWCOUNT?

取得報表的列數。

- **語法**:

```
*PA:REPORT:ROWCOUNT?
```

- **參數**:

  無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。

- **viRead() 回傳值**:

  報表的列數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*PA:REPORT:ROWCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("列數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*PA:REPORT:COLUMNCOUNT?

取得報表的欄數。

- **語法**:

```
*PA:REPORT:COLUMNCOUNT?
```

- **參數**:

  無

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。

- **viRead() 回傳值**:

  報表的欄數。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*PA:REPORT:COLUMNCOUNT?")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("欄數 = %d\n", atoi(buffer));
```


Copyright © 2025 Acute Technology Inc.

*PA:REPORT:SAVE

使用指定類型將報表儲存到指定檔名的檔案中。

- **語法**:

```
*PA:REPORT:SAVE <filename> [-SAVEALL] [-ADVRPT] [-SPLITDUR]
```

```
*PA:REPORT:SAVE <filename> [-SAVESPLIT <row>] [-ADVRPT] [-SPLITDUR]
```

```
*PA:REPORT:SAVE <filename> [-SAVERANGE <start row> <end row> <start column> <end column>] [-ADVRPT]
[-SPLITDUR]
```

- **參數**:

  filename: 目標檔案名稱。

  -SAVEALL (選擇性): 將整個報表儲存到一個檔案中。

  -ADVRPT (選擇性): 包含進階報表資料。

  -SPLITDUR (選擇性): 將報表時間戳記和持續時間資料分成兩欄。

  -SAVESPLIT <row> (選擇性): 根據列數將報表分割成多個檔案。

  -SAVERANGE (選擇性): 儲存選定列和欄範圍內的報表。

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_INPUT_FILE_DIR_INVALID	表示輸入的檔案名稱無效。

- **viRead() 回傳值**:

  無

- **範例**:

```
// 將整個報表儲存到一個檔案中
viWrite("\*PA:REPORT:SAVE C:/Temp/AllReport.csv -SAVEALL");
```

```
// 將整個報表連同進階報表資料儲存到一個檔案中
viWrite("\*PA:REPORT:SAVE C:/Temp/AdvReport.csv -SAVEALL -ADVRPT");
```

```
// 根據列數 = 2000 將報表分割成多個檔案
viWrite("\*PA:REPORT:SAVE C:/Temp/LimitRpt.csv -SAVESPLIT 2000");
```

```
// 儲存選定列和欄範圍內的報表
viWrite("\*PA:REPORT:SAVE C:/Temp/RangedRpt.csv -SAVERANGE 200 2000 1 5");
```


Copyright © 2025 Acute Technology Inc.

*PA:REPORT:DATA?

從選定索引取得報表資料。

- **語法**:

```
*PA:REPORT:DATA? <row> <column>
```

- **參數**:

  row: 列索引。

  column: 欄索引。

- **狀態碼**:

狀態碼	說明
AQVI_NO_ERROR	指令執行成功。
AQVI_ROW_COL_INDEX_INVALID	表示欄索引無效。

- **viRead() 回傳值**:

  選定索引的報表資料。

- **範例**:

```
// 使用 viWrite() 發送指令
if (!viWrite("*PA:REPORT:DATA? 105 2")) {
    return; // viWrite 失敗
}

// 等待軟體處理指令
while (viGetCommandResult() == PROC_PENDING) {
    ProcessWaitAction();
}

// 檢查指令是否執行成功
if (viGetCommandResult() != PROC_OK_WITH_RETURN) {
    // 無回傳資料或處理指令失敗
    return;
}

// 使用 viRead() 從應用程式緩衝區取得回應
char buffer[32] = {0};
if (!viRead(buffer, sizeof(buffer))) {
    return; // viRead 失敗
}

// 顯示結果
printf("索引 [105, 2] 的報表資料 = %s\n", buffer);
```


Copyright © 2025 Acute Technology Inc.

Copyright © 2025 Acute Technology Inc.

5. Error Codes

5.1 狀態碼表

Status Code	Value	Description
AQVI_NO_ERROR	0	Operation succeeded without error
AQVI_ALREADY_OPEN	1	OPENRM should only be called once to establish the software connection
AQVI_APPLICATION_NOT_STARTED	2	The main software application was not started before calling the viOPENRM
AQVI_FAILED_TO_CONNECT	3	Failed to establish the connection to software application
AQVI_CONNECTION_NOT_VALID	4	The software connection is not valid yet, call viOpenRM first to establish the connection
AQVI_CONNECTION_LOST	5	The software application has been closed or re-started, need to call viCloseRM and viOpenRM to establish the connection again
AQVI_FAILED_TO_COMMUNICATE	6	Unable to send data to software application
AQVI_NO_RETURN_DATA	7	No data was returned from the software application
AQVI_INPUT_NULL_POINTER	8	Input data buffer is NULL pointer and not valid
AQVI_DATA_BUFFER_TOO_SMALL	9	Input data buffer size is too small
AQVI_SEL_APP_AFTER_VM_OPENED	10	Select application type should be done before OpenVM
AQVI_SEL_APP_TYPE_INVALID	11	Selected invalid application type
AQVI_PREVIOUS_CMD_PROCESSING	12	Previous input command still processing
AQVI_INPUT_PARAMETER_UNKNOWN	13	Input parameter unknown or not supported
AQVI_INPUT_PARAMETER_INCOMPLETED	14	Input parameter is incomplete
AQVI_TIMEOUT	15	Input request timeout

Copyright © 2025 Acute Technology Inc.

Status Code	Value	Description
AQVI_NOT_SUPPORTED	1000	Not supported in current software
AQVI_INCOMPLETE_COMMAND	1001	Input command incomplete
AQVI_SUBWND_INVALID	1002	Input command requires an existing SubWindow
AQVI_SUBWND_CNT_EXCEED	1003	Input command attempts to add new SubWindow page, but software has already exceeded maximum page count
AQVI_SW_BUSY	1004	Input command ignored while software busy
AQVI_LASUBWND_INVALID	1005	Input command requires exist Logic Analyzer SubWindow
AQVI_PASUBWND_INVALID	1006	Input command requires exist Protocol Analyzer SubWindow
AQVI_DECODE_REPORT_INVALID	1007	Input command requires exist Decode Report
AQVI_TIMING_REPORT_INVALID	1008	Input command requires exist Timing Report
AQVI_INPUT_COMMAND_FORMAT_ERROR	1009	Input command format error
AQVI_INPUT_FILE_DIR_INVALID	1010	Input file directory invalid
AQVI_CAPTURE_ALREADY_RUNNING	1011	Sending Capture Start command while capture already running
AQVI_CAPTURE_NOT_RUNNING	1012	Sending Capture Stop command while capture is not running
AQVI_ROW_COL_INDEX_INVALID	1013	Input Row or Column index invalid
AQVI_SELECT_INDEX_INVALID	1014	Input index selection invalid
AQVI_INPUT_PARAMETER_INVALID	1015	Missing input parameters or not enough input parameter
AQVI_INPUT_SETTING_FILE_FORMAT_ERROR	1016	Input setting file format error
AQVI_FILE_ACCESS_ERROR	1017	Unable to access File, unable to reach select file directory or selected unsupported file type
AQVI_TRANSITION_REPORT_INVALID	1018	Input command requires an existing Transition Report
AQVI_MEASUREMENT_REPORT_INVALID	1019	Input command requires exist Measurement Report
AQVI_SELECT_LABEL_INVALID	1020	Selected Label Name not valid
AQVI_SELECT_RANGE_ERROR	1021	Selected Range Condition Error, can't mix Line and Cursor Range Condition
AQVI_SELECT_CHANNEL_NAME_INVALID	1022	Selected Channel Name in Cursor Search is not valid
AQVI_CURSOR_SEARCH_FAILED	1023	Cursor Search Failed, unable to find more valid match
AQVI_WAVEFORM_NOT_READY	1024	Waveform Data Not Ready
AQVI_NOT_IN_EV_MODE	1025	Software is not configured in EV mode
AQVI_NO_EV_ANALYSIS_RESULT	1026	Software doesn't have any valid EV analysis result
AQVI_SELECT_DEV_NOT_EXIST	1027	Selected device S/N not present in device list
AQVI_UNSUPPORT_FEATURE	9995	Unsupported feature, feature is not supported in current model or demo mode
AQVI_UNFINISHED_FEATURE	9996	Unfinished Feature

Copyright © 2025 Acute Technology Inc.

Status Code	Value	Description
AQVI_UNKNOWN_COMMAND_ERROR	9997	Input command is not supported
AQVI_UNKNOWN_ERROR	9998	Detected unknown error
AQVI_DLL_NOT_READY	9999	DLL initialization not finished, please retry again later

Copyright © 2025 Acute Technology Inc.

6. Release Notes

Version	Date	Description
1.0.9	2025-03-09	Update the documentation
1.0.9 rev. 2	2025-03-20	Fix a discrepancy in the documentation with the saving file commands
1.0.9 rev. 3	2025-03-21	Add installation instructions

Copyright © 2025 Acute Technology Inc.