

LJ-X8000A通信库

参考手册

请在使用前阅读此手册。
请妥善保管此手册，以便随时查阅。

目录

1	软件使用许可协议	4
2	前言	5
3	运行环境	5
4	文件构成	5
5	嵌入方法	6
5.1	文件构成	6
5.2	链接	6
5.2.1	C++	6
5.2.2	C# / VB.NET	6
6	类型	6
7	轮廓数据结构、CallBack 函数	7
7.1	轮廓数据构成	7
7.2	轮廓数据中包含的高度数据数	8
7.3	轮廓数据结构	9
7.4	返回函数 I/F 定义	10
8	函数	17
8.1	函数一览	17
8.1.1	对 DLL 的操作	17
8.1.2	建立 / 断开与控制器之间的通信路径	17
8.1.3	系统控制	17
8.1.4	测量控制	18
8.1.5	变更 / 读取设定相关	18
8.1.6	测量结果的获取	18
8.1.7	高速数据通信相关	19
8.1.8	实用函数相关	19
8.2	函数参考	19
8.2.1	对 DLL 的操作	19
8.2.2	建立 / 断开与控制器之间的通信路径	20
8.2.3	系统控制	21
8.2.4	测量控制	25
8.2.5	变更 / 读取设定相关	27
8.2.6	测量结果的获取	33
8.2.7	高速数据通信相关	41
8.2.8	实用函数相关	45
8.2.9	补充	55
9	通用返回代码	59
9.1	通信库反馈的返回代码	59
9.2	控制器反馈的返回代码	59
10	示例程序	60
10.1	示例程序	60
10.1.1	用户界面规格	60
10.1.2	文件保存格式	64
10.2	图像获取示例	65
11	附录	66
11.1	发送 / 接收设定	66
11.2	批处理发送 / 接收	67
11.3	发送 / 接收设定项目详情	68
	修订履历	84

1 软件使用许可协议

使用本软件前，客户需同意下述软件使用许可协议（以下简称“本协议”）的内容。

客户使用或复制 LJ-X8000A 通信库（以下简称“本软件”）的部分或全部功能时，将视为客户已同意本协议中规定的所有内容，且本协议成立。

第 1 条（使用权的授予）

1. 在客户遵守本协议内容的前提下，基恩士株式会社（以下简称“本公司”）将对客户授予本软件的非独占性使用权。

第 2 条（禁止事项）

客户不可对本软件实施以下操作或改动。

- a. 更改本软件现有功能或向本软件添加新功能。
- b. 对本软件实施的所有逆向工程行为，如：反编译、反汇编等。
- c. 向第三方二次销售、转让、二次分配、授予、租赁本软件及本公司提供的本软件授权密钥等。但允许客户将本软件与使用本软件编写的应用程序共同进行二次分配。

第 3 条（著作权等）

与本软件及本软件手册相关的所有知识产权（如著作权），归本公司所有。

第 4 条（免责）

客户或第三方因使用本软件而遭受的所有损害，本公司概不负责！

第 5 条（支持）

本公司将基于本协议，根据客户针对本软件提出的问题，提供技术支持。但并不保证本公司提供的技术支持服务可使客户达成期望目的。

第 6 条（协议终止）

1. 当客户进行废弃本软件及其复制品等以停止使用本软件时，本协议自动终止。
2. 当客户违反本协议中规定的任一条款时，本公司可单方面解除本协议。
同时，客户应立即废弃本软件及其复制品，或将之返还至本公司。
3. 因客户违反本协议，而使本公司蒙受损失时，客户应向本公司赔偿相关损失。

第 7 条（准据法）

本协议遵从日本国法律。

2 前言

LJ-X8000A 通信库，为客户提供通过用户应用程序控制 LJ-X8000A 的通信接口。具体使用方法，请参照示例程序。

3 运行环境

操作系统	Windows 10 (Home/Pro/Enterprise) Windows 7 (SP1 或更高版本) (Home Premium/Professional/Ultimate)
CPU	intel®Core™ i3 处理器同级规格
内存容量	8 GB 以上
硬盘可用空间	10 GB 以上
接口	配备下列任意一项。 Ethernet 1000BASE-T/100BASE-TX *1

*1 不保证连接到 LAN 和使用路由器连接时的操作状况。

3.1 运行环境

以下内容是在使用 LJ-X8000A 通信库执行应用程序时的基本环境。

3.1.1 Microsoft C Runtime Library

Microsoft C Runtime Library。这是执行 DLL 所必需的运行环境。

请执行 vcredist_x86.exe/vcredist_x64.exe 进行安装。

3.1.2 Microsoft .NET Framework

这是执行示例程序所必需的运行环境。

请执行 dotnetfx45_full_x86_x64.exe 进行安装。

4 文件构成

整套通信库存储如下。

C:\Program Files\KEYENCE\LJ-X Navigator\lib\Sample

\src\lib\include	存储 • 定义错误代码的头文件 • 定义 API 的头文件 (LJX8_IF.h)。
\src\lib\x64 \src\lib\x86	存储 • DLL 本体 (LJX8_IF.dll) • 引入库 (LJX8_IF.lib)。 提供 32 位 (x86)、64 位 (x64) 专用文件。
\src\CPP \src\CS \src\VBNET	• 存储示例程序的源文件。 支持 C++、C#、VB.NET。
\exe\x64 \exe\x86	• 存储示例程序的执行文件 (exe)。 通过 C++、C#、VB.NET 制作的 32 位 (x86)、64 位 (x64) 专用文件。

C:\Program Files\KEYENCE\LJ-X Navigator\lib\Sample_ImageAcquisition 也同样如此。

5 嵌入方法

5.1 文件构成

执行时必需的文件如下。

请将下列文件夹及文件与执行文件放置在同一文件夹中。

- LJX8_IF.dll

* “LJX8_IF.dll” 由 Visual C++ 2015 Update3 创建。

5.2 链接

5.2.1 C++

5.2.1.1 链接

明链接、暗链接均可使用。

进行暗链接时，请链接 “LJX8_IF.lib”。

* “LJX8_IF.lib” 由 Visual C++ 2015 Update3 创建。

5.2.1.2 包含文件

请务必将下列头文件放置在源文件中。

- LJX8_IF.h
- LJX8_ErrorCode.h

5.2.2 C# / VB.NET

利用 DllImport 属性，调用各 IF。

通过 IF 引数传输结构时，需指定结构布局属性，传输给与 DLL 存储结构相同的结构。

详情请参照示例的 NativeMethods 等级 (NativeMethods.cs)。

安装了调用各函数的处理。

6 类型

本文中的变量类型定义如下。

CHAR	有符号的 8 bit 整数
BYTE	无符号的 8 bit 整数
SHORT	有符号的 16 bit 整数
WORD	无符号的 16 bit 整数
LONG	有符号的 32 bit 整数
DWORD	无符号的 32 bit 整数
FLOAT	单精度浮动小数点数 (32bit)
DOUBLE	双精度浮动小数点数 (64bit)

7 轮廓数据结构、CallBack 函数

7.1 轮廓数据构成

基本的轮廓数据结构如下所示。各轮廓数据在标题和页脚之间。
详情请参照“7.3 轮廓数据结构”。（有关带 SimpleArray 的函数，请参照各函数的说明）

例：获取 10 条 LJ-X8000 系列轮廓数据（高度数据（有符号的 32 bit）× 3200 点）时，pdwProfileData 或 pdwBatchData、pBuffer 中将存储如下数据。连接 LJ-X 传感头的情况下，请通过 LJ-X Navigator 设定亮度无/有。连接 LJ-V 传感头的情况下，型号末尾带有 B 的类型的传感头有亮度，其他传感头无亮度。

• **LJ-X 无亮度时**

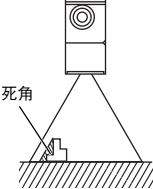
轮廓 1 ⋮	LJX8IF_PROFILE_HEADER	32bit×6
	轮廓（3200 点）	32bit×3200
	LJX8IF_PROFILE_FOOTER	32bit×1
轮廓 10	LJX8IF_PROFILE_HEADER	32bit×6
	轮廓（3200 点）	32bit×3200
	LJX8IF_PROFILE_FOOTER	32bit×1

• **LJ-X 有亮度时**

轮廓 1 ⋮	LJX8IF_PROFILE_HEADER	32bit×6
	轮廓（3200 点）	32bit×3200
	亮度轮廓（3200 点）	32bit×3200
	LJX8IF_PROFILE_FOOTER	32bit×1
轮廓 10	LJX8IF_PROFILE_HEADER	32bit×6
	轮廓（3200 点）	32bit×3200
	亮度轮廓（3200 点）	32bit×3200
	LJX8IF_PROFILE_FOOTER	32bit×1

以 0.01 μm 为单位存储轮廓数据。
例：为 1.98750 mm 时，将存储为 198750。
亮度数据存储 0 至 1023 范围内的值。

轮廓数据的输出格式是有符号的 32 bit 数据，无法检测出正确轮廓的点时将输出下列值。（有关带 SimpleArray 的函数，请参照各函数的说明）

值（16 进制表示）	名称	理由
-2147483648 (0x80000000) -2147483647 (0x80000001)	无效数据	在无法检测来自目标物的反射光量或设定了屏蔽等设定无效的情况下输出。
-2147483646 (0x80000002)	死角数据	位于死角时输出。 
-2147483645 (0x80000003)	判断待机数据	进行平均处理所必需的轮廓不足时输出。

7.2 轮廓数据中包含的高度数据数

7.2.1 轮廓数据中包含的高度数据数计算方法（LJ-X 传感头）

在 LJ-X 传感头上，将 3200 点作为 DLL 获取的轮廓数据数的基数，乘上由下列设定决定的补正系数，即得出实际获取的有效轮廓数据数。

设定			补正系数	备注
主项	分项	设定值		
拍摄设定	测量范围 X 方向	FULL	1.00	初始值
		3/4	0.75	
		1/2	0.50	
		1/4	0.25	
轮廓设定	间隔（X 轴）	OFF	1.00	初始值
		1/2	0.50	
		1/4	0.25	
采样周期	16 kHz（无亮度输出） 8 kHz（有亮度输出）		0.50	
	上述以外		1.00	

例如，以下设定中的轮廓数据数为 400 ($= 3200 \times 0.25 \times 0.50 \times 1.00$) 点。

测量范围 X 方向：1/4

间隔（X 轴）：1/2

4kHz（有亮度输出）

但是，测量范围 X 方向 1/4 间隔（X 轴）1/4、采样周期 8 kHz 或 16 kHz 时，将间隔（X 轴）设定为 1/2，轮廓数据将自动变为“200”。

7.2.2 轮廓数据中包含的高度数据数计算方法（LJ-V 传感头）

在 LJ-V 传感头上，将 800 点作为 DLL 获取的轮廓数据数的基数，乘上由下列设定决定的补正系数，即得出实际获取的有效轮廓数据数。

设定			补正系数	备注
主项	分项	设定值		
拍摄设定	测量范围 X 方向	FULL	1.00	初始值
		MIDDLE	0.75	
		SMALL	0.50	
	Binning	OFF	1.00	初始值
		ON	0.50	
轮廓设定	间隔（X 轴）	OFF	1.00	初始值
		1/2	0.50	
		1/4	0.25	

例如，以下设定中的轮廓数据数为 300 ($= 800 \times 0.75 \times 1.00 \times 0.50$) 点。

测量范围 X 方向：MIDDLE

Binning：OFF

间隔（X 轴）：1/2

但如果上述计算结果求得的轮廓数据数小于 200 间隔（X 轴）的设定将被调整，确保轮廓数据数大于 200。

例如，下列情况中，轮廓数据数为 300 点。

测量范围 X 方向：MIDDLE

Binning：OFF

间隔（X 轴）：1/4

- 具体计算过程如下所示。
- (1) $800 \times 0.75 \times 1.00 \times 0.25 = 150$
 - (2) 由于计算结果小于 200，将间隔（X 轴）的设定值由 4 调整为 2
 - (3) $800 \times 0.75 \times 1.00 \times 0.5 = 300$
 - (4) 由于计算结果为 200 以上，确定为上述轮廓数据数

7.3 轮廓数据结构

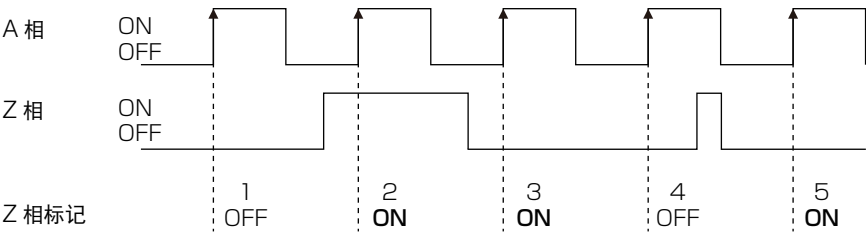
各轮廓数据在标题和页脚之间。标题中存储有编码器 Z 相的 ON/OFF 信息、触发计数值、编码器计数值。
以下表示不同的结构。

名称	轮廓标题信息结构
定义	<pre>Typedef struct { DWORD reserve; DWORD dwTriggerCount; LONG IEncoderCount; DWORD reserve2[3]; } LJX8IF_PROFILE_HEADER;</pre>
描述	<p>附加在轮廓中的标题信息。</p> <p>reserve 第 7 bit: Z 相标记</p> <p>表示是否已输入编码器的 Z 相。（*）</p> <div><div>MSB</div><div>31</div><div>...</div><div>7</div><div>6</div><div>5</div><div>4</div><div>3</div><div>2</div><div>1</div><div>0</div><div>LSB</div></div> <p>dwTriggerCount 表示轮廓是由测量开始后的第几次触发获得的。（触发计数）从“0”开始计数。</p> <p>IEncoderCount 触发发送时的编码器计数。（编码器计数）从“0”开始计数。</p> <p>reserve2[0] 发行触发时的计时器计数值。 在超出上限 (0 ~ 4,294,967,295) 时，计数器将运行一周后返回到 0。 计数的刻度为 0.1ms。</p>
备注	<p>除了变更设定及切换程序时，触发计数和编码器计数还将在下列情况下被复位。</p> <ul style="list-style-type: none">• 执行内存清除时• LASER_ON 端子停止发射激光后再次开启时 <p>计时器计数，在启动控制器后，在自由运行下累加。 计时器计数的当前值获取 / 设置到任意值，可通过指令做到。</p>

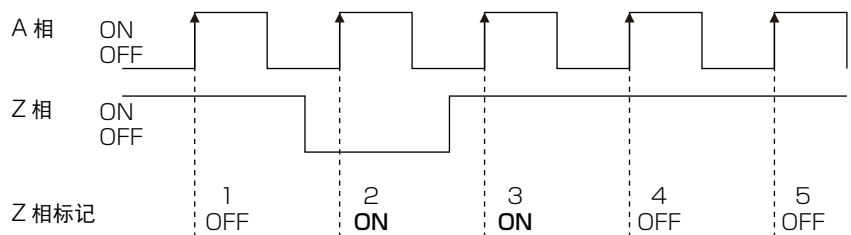
(*) Z 相标记

前次触发输入（没有前次触发时，则以当前测量开始为准）到本次触发输入之间，如果有 Z 相的 ON 输入，则标记开启。

例) 编码器触发 1 相 1 倍增、无间隔



[注] 使用负逻辑编码器进行 Z 相输入时, 请将测量通用设定的 TRG 最短输入时间设定为 “7 μ s”。负逻辑时, 如下所示, Z 相标记开启。



名称	轮廓页脚信息结构
定义	<pre> typedef struct { DWORD reserve; } LJX8IF_PROFILE_FOOTER; </pre>
描述	附加在轮廓中的页脚信息。无（仅限预约）。
备注	—

7.4 返回函数 I/F 定义

要高速获取轮廓数据时, 使用高速数据通信。(参照 “8.1.7 高速数据通信相关”)

开始高速数据通信后, 将从控制器连续向 PC 传送轮廓。传送指定数量的轮廓后, 将调用返回函数。返回函数有两种类型, 一种将获取的轮廓数据直接存储在缓存中, 另一种将轮廓数据中包含的高度数据和亮度数据以容易转换为 Bitmap 等格式存储在缓存中 (Simple Array)。

高速数据通信用返回函数的详情如下所示。支持 InitializeHighSpeedDataCommunication 指令。

格式	<pre> void (*pCallback) (BYTE* pBuffer, DWORD dwSize, DWORD dwCount, DWORD dwNotify, DWORD dwUser); </pre>
参数	<p>pBuffer(in) 指向存储轮廓数据缓存的指针。如 “7.1 轮廓数据构成” 中所示, 轮廓数据夹在标题和页脚之间被输出。该参数作为 1 个单位, 只重复存储实际获取的轮廓数 (dwCount)。</p> <p>dwSize(in) 将 pBuffer 中的标题 (32 bit x 6) + 轮廓数据 (32 bit x 轮廓数据数量) + 页脚 (32 bit) 作为 1 个单位时 BYTE 单位的大小。</p> <p>dwCount(in) 存储在 pBuffer 中的轮廓数。</p> <p>dwNotify(in) 通知高速数据通信的中断及批处理测量的间隔。详情请参照 “7.4.1 补充”。</p> <p>dwUser(in) 高速通信初始化时设置的用户信息。(dwThreadId)</p>
返回值	无
说明	<p>使用高速数据通信功能时, 根据数据接收及通信状态的变化进行调用。</p> <p>该返回函数由主线程以外的线程调用。</p> <p>在返回函数中, 请仅将轮廓数据存储在线程安全的缓存中。由于调用返回函数的线程与接收数据的线程相同, 返回函数的处理时间会影响数据的接收速度, 根据环境条件, 可能导致无法正常通信。详情请参照示例程序。</p> <p>以 0.01 μm 为单位存储轮廓数据。</p>

高速数据通信（SimpleArray）用返回函数的详情如下所示。支持 InitializeHighSpeedDataCommunicationSimpleArray 指令。

格式	<div>void (*pCallBackSimpleArray) (LJX8IF_PROFILE_HEADER* pProfileHeaderArray, WORD* pHeightProfileArray, WORD* pLuminanceProfileArray, DWORD dwLuminanceEnable, DWORD dwProfileDataCount, DWORD dwCount, DWORD dwNotify, DWORD dwUser);</div>																						
参数	<div><div>pProfileHeaderArray(in)</div><div>指向存储轮廓标题信息（排列）缓存的指针。只重复存储实际获取的轮廓数（dwCount）。</div></div> <div><div>pHeightProfileArray(in)</div><div>指向存储轮廓数据中所包含“高度数据”（排列）缓存的指针。“高度数据”的 dwProfileDataCount 部分的数据包含在 1 个轮廓中，并以此作为 1 个单位，只重复存储实际获取的轮廓数（dwCount）。如需将存储的值（0 至 65535）转换为高度数据（μm），请使用以下公式。</div><div>高度（μm） = （ 存储的值 - 32768 ）* 换算系数</div><div>LJ-X 传感头的换算系数</div><table><tr><td>LJ-X8020</td><td>LJ-X8060</td><td>LJ-X8080</td><td>LJ-X8200</td><td>LJ-X8400</td><td>LJ-X8900</td></tr><tr><td>0.4</td><td>0.8</td><td>1.6</td><td>4</td><td>8</td><td>16</td></tr></table><div>LJ-V 传感头的换算系数</div><table><tr><td>LJ-V7020(K)(B)</td><td>LJ-V7060(K)(B)</td><td>LJ-V7080(B)</td><td>LJ-V7200(B)</td><td>LJ-V7300(B)</td></tr><tr><td>0.4</td><td>0.8</td><td>1.6</td><td>4</td><td>8</td></tr></table></div> <div><div>例：使用 LJ-X8080 传感头、存储的值为 34232 时， （34232 - 32768 ）*1.6 = 2342.4 [μm] 使用 LJ-X8080 传感头、存储的值为 31575 时， （31575 - 32768 ）*1.6 = -1908.8 [μm]</div><div>无效数据、死角数据和判断待机数据存储 0。</div></div> <div><div>pLuminanceProfileArray(in)</div><div>指向存储轮廓数据中所包含“亮度数据”（排列）缓存的指针。只重复存储实际获取的轮廓数（dwCount）。存储 0 至 1023 范围内的值。</div><div>dwLuminanceEnable(in)</div><div>亮度数据的有无 有：1、 无亮度：0</div><div>dwProfileDataCount(in)</div><div>轮廓数据 X 方向的数据点数（LJ-X 传感头的初始值为 3200，LJ-V 传感头的初始值为 800）</div><div>dwCount(in)</div><div>每个排列中存储的轮廓数。</div><div>dwNotify(in)</div><div>通知高速数据通信的中断及批处理测量的间隔。详情请参照“7.4.1 补充”。</div><div>dwUser(in)</div><div>高速通信初始化时设置的用户信息。（dwThreadId）</div></div>	LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900	0.4	0.8	1.6	4	8	16	LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)	0.4	0.8	1.6	4	8
	LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900																	
	0.4	0.8	1.6	4	8	16																	
	LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)																		
	0.4	0.8	1.6	4	8																		
返回值	无																						
说明	<div>使用高速数据通信功能时，根据数据接收及通信状态的变化进行调用。</div> <div>该返回函数由主线程以外的线程调用。</div> <div>在返回函数中，请仅将轮廓数据存储在线程安全的缓存中。由于调用返回函数的线程与接收数据的线程相同，返回函数的处理时间会影响数据的接收速度，根据环境条件，可能导致无法正常通信。详情请参照示例程序。</div>																						

7.4.1 补充

dwNotify 参数

高速数据通信模式下，除了在接收轮廓数据时，返回函数还会在其他几种情况发生时被调用。所发生的情况可以通过 dwNotify 参数进行确认。dwNotify = 0: 通信正常进行。出现 0 以外的情况，请参照下列说明。

dwNotify 的 bit		状况	批处理 OFF	批处理 ON
LSB	0	连续发送被中止（因指令中止）	○	○
	1	连续发送被中止（自动中止） * 由于设定被变更。	○	○
	2	连续发送被中止（自动中止） * 由于程序被切换。	○	○
	3	连续发送被强制中止。 例：由于 LJ-X Navigator 使用了高速通信。	○	○
	4 至 7	预约		
	8	内存清除造成的发送中断	○	○
	9 至 15	预约		
	16	批处理测量的数据已发送完毕。 * 但进行 LASER OFF ON 操作后，传送可能会截止到该瞬间正在发送的轮廓，而无法传送批处理测量的全部数据。	—	○
MSB	17 至 31	预约		

0 至 3 bit、8 bit 表示连续发送被中止。

重新开始连续发送时，请按照“结束高速数据通信”→“中断 Ethernet 通信”→“开始 Ethernet 通信”→“初始化 Ethernet 高速通信”→“请求开始准备 Ethernet 高速数据通信”的流程，开始高速数据通信。

第 16 bit 仅在批处理测量 ON 时有效。

批处理测量 ON 时，即使未达到所设定的批处理点数，也可结束批处理测量。因此，为了对批处理数据进行分段，将进行本位起始通知。

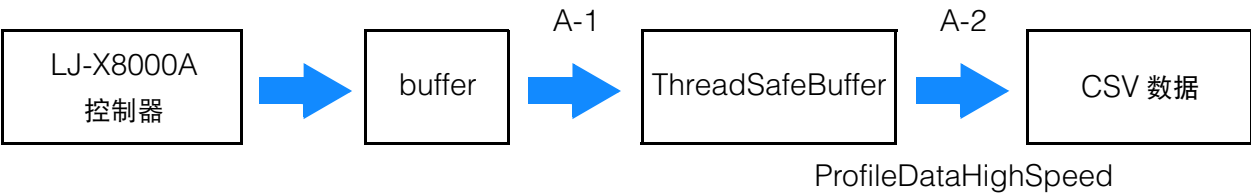
7.4.2 返回函数详情

开始高速数据通信后，将通过 LJ-X8000A 向 PC 的内部缓存（buffer）传送轮廓数据。此时，当内部缓存中的数据达到开始高速数据通信时指定的（Ethernet 高速数据通信初始化指令的 dwProfileCount）轮廓数时，将调用返回函数。

返回函数有两种类型，一种将获取的轮廓数据直接存储在缓存中（支持高速数据通信的 InitializeHighSpeedDataCommunication 指令），另一种将轮廓数据中包含的高度数据和亮度数据以容易转换为 Bitmap 等格式存储在缓存中（支持高速数据通信的 InitializeHighSpeedDataCommunicationSimpleArray 指令）。

7.4.2.1 返回函数（支持 InitializeHighSpeedDataCommunication 指令）

在示例程序中，将 PC 内部缓存中存储的轮廓转移到 ThreadSafeBuffer。此外还备有将轮廓数据以 CSV 数据格式保存到指定文件夹 / 文件中的示例程序，请进行参考。



注意点

- 在返回函数中，请仅将轮廓数据存储在线程安全的缓存中。如果放入各种处理，可能会无法正常接收轮廓数据。

A-1 <从 buffer 向 ThreadSafeBuffer 转移轮廓数据的具体示例（依据 C# 示例程序）>

调用返回函数后，将运行以下程序。此处会将存储的轮廓数据（buffer 内）最终转移到 ThreadSafeBuffer。请勿在此处添加处理。如果此处的处理变多，可能会无法正常接收轮廓数据。

```
/// <summary>
/// Method that is called from the DLL as a callback function
/// </summary>
/// <param name="buffer">Leading pointer of the received data</param>
/// <param name="size">Size in units of bytes of one profile</param>
/// <param name="count">Number of profiles</param>
/// <param name="notify">Completion flag</param>
/// <param name="user">Thread ID (value passed during initialization)</param>
private static void ReceiveHighSpeedData(IntPtr buffer, uint size, uint count, uint notify, uint user)
{
    // @Point
    // Take care to only implement storing profile data in a thread save buffer in the callback function.
    // As the thread used to call the callback function is the same as the thread used to receive data,
    // the processing time of the callback function affects the speed at which data is received,
    // and may stop communication from being performed properly in some environments.

    uint profileSize = (uint)(size / Marshal.SizeOf(typeof(int)));
    List<int[]> receiveBuffer = new List<int[]>();
    int[] bufferArray = new int[(int)(profileSize * count)];
    Marshal.Copy(buffer, bufferArray, 0, (int)(profileSize * count));

    // Profile data retention
    for (int i = 0; i < (int)count; i++)
    {
        int[] oneProfile = new int[(int)profileSize];
        Array.Copy(bufferArray, i * profileSize, oneProfile, 0, profileSize);
        receiveBuffer.Add(oneProfile);
    }

    if (ThreadSafeBuffer.GetBufferDataCount((int)user) + receiveBuffer.Count < Define.WriteDataSize)
    {
        ThreadSafeBuffer.Add((int)user, receiveBuffer, notify);
    }
    else
    {
        _isBufferFull[(int)user] = true;
    }
}
```

A-2 <以 CSV 格式保存由返回函数调出的轮廓数据（C# 示例程序）>

如上述说明，调出返回函数后，示例程序会将数据转移到 ThreadSafeBuffer。点击“Save the profile”后，将分别以 CSV 格式保存“高度数据”和“亮度数据”。具体来说，示例程序说明了每条信息的存储位置。

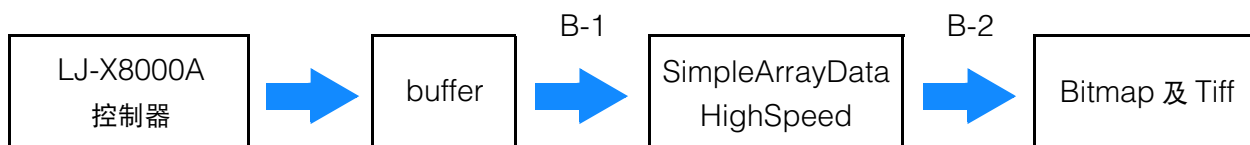
如 P.7 中轮廓数据结构说明所述，“高度数据”和“亮度数据”存储在以下 profile.ProfData 中。（有亮度时）例如轮廓数据点数为 3200 点时，profile.ProfData[0] 至 profile.ProfData[3199] 中存储“高度数据”，profile.ProfData[3200] 至 profile.ProfData[6399] 中存储“亮度数据”。使用 startIndex 将这些分别存储为“高度数据”和“亮度数据”。

```
/// <summary>
/// Export processing
/// </summary>
/// <param name="profileDataList">Profile data list</param>
/// <param name="savePath">Save file path</param>
/// <param name="startIndex">Data start index for saving in one profile</param>
private static void ExportMultipleData(List<ProfileData> profileDataList, string savePath, int startIndex)
{
    // Save the profile
    using (StreamWriter streamWriter = new StreamWriter(savePath, false, Encoding.GetEncoding("utf-16")))
    {
        // Output the data of each profile
        foreach (ProfileData profile in profileDataList)
        {
            StringBuilder stringBuilder = new StringBuilder();
            short profileDataCount = profile.ProfInfo.nProfileDataCount;

            for (int i = 0; i < profileDataCount; i++)
            {
                stringBuilder.AppendFormat("{0}\t", profile.ProfData[startIndex + i]);
            }
            streamWriter.WriteLine(stringBuilder);
        }
    }
}
```

7.4.2.2 返回函数（支持 InitializeHighSpeedDataCommunicationSimpleArray 指令）

在示例程序中，将 PC 内部缓存中存储的轮廓转移到 SimpleArrayDataHighSpeed。此外还备有将轮廓数据以 Bitmap 格式及 Tiff 格式保存到指定文件夹 / 文件中的示例程序，请进行参考。



注意点

- 在返回函数中，请仅将轮廓数据存储在在线程安全的缓存中。如果放入各种处理，可能会无法正常接收轮廓数据。

B-1 <从 buffer 向 SimpleArrayDataHighSpeed 转移轮廓数据的具体示例（依据 C# 示例程序）>

调用返回函数后，将运行以下程序。此处会将存储的轮廓数据（buffer 内）最终转移到 SimpleArrayDataHighSpeed。请勿在此处添加处理。如果此处的处理变多，可能会无法正常接收轮廓数据。

以下 profileBuffer 对应高度数据，luminanceBuffer 对应亮度数据。分别以 16 bit 存储。如需转换为高度数据，请参照“P.11 高速数据通信（SimpleArray）用返回函数”。

```
/// <summary>
/// Method that is called from the DLL as a callback function
/// </summary>
/// <param name="headBuffer">A pointer to the buffer that stores the header data array</param>
/// <param name="profileBuffer">A pointer to the buffer that stores the profile data array</param>
/// <param name="luminanceBuffer">A pointer to the buffer that stores the luminance profile data array</param>
/// <param name="isLuminanceEnable">The value indicating whether luminance data output is enable or not</param>
/// <param name="profileSize">The data count of one profile</param>
/// <param name="count">Number of profiles</param>
/// <param name="notify">Completion flag</param>
/// <param name="user">Thread ID (value passed during initialization)</param>
private void ReceiveHighSpeedSimpleArray(IntPtr headBuffer, IntPtr profileBuffer, IntPtr luminanceBuffer, uint
isLuminanceEnable, uint profileSize, uint count, uint notify, uint user)
{
    // @Point
    // Take care to only implement storing profile data in a thread save buffer in the callback function.
    // As the thread used to call the callback function is the same as the thread used to receive data,
    // the processing time of the callback function affects the speed at which data is received,
    // and may stop communication from being performed properly in some environments.

    _deviceData[(int)user].SimpleArrayDataHighSpeed.AddReceivedData(profileBuffer, luminanceBuffer, count);
    _deviceData[(int)user].SimpleArrayDataHighSpeed.Notify = notify;
}
```

B-2 <以 Bitmap 格式及 Tiff 格式保存轮廓数据的示例（C# 示例程序）>

如上所述调用返回函数后，轮廓数据会存储到 SimpleArrayDataHighSpeed。示例程序中已将此数据的以下部分转换为 Bitmap 格式及 Tiff 格式。详情请参照源代码。

```
/// <summary>
/// "Save As Image File" button clicked.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void _buttonHighSpeedSaveAsBitmapFile_Click(object sender, EventArgs e)
{
    int width = _deviceData[_currentDeviceId].SimpleArrayDataHighSpeed.DataWidth;
    if (width == 0)
    {
        AddLog("No simple array data.");
        return;
    }

    if (string.IsNullOrEmpty(_textBoxHighSpeedProfileFilePath.Text))
    {
        AddLog("Failed to save image. (File path is empty.)");
        return;
    }

    Cursor.Current = Cursors.WaitCursor;

    int startIndex = (int)_numericUpDownProfileNo.Value;
    int dataCount = (int)_numericUpDownProfileSaveCount.Value;
    bool result = _deviceData[_currentDeviceId].SimpleArrayDataHighSpeed.SaveDataAs
        Images(_textBoxHighSpeedProfileFilePath.Text, startIndex, dataCount);

    AddLog(result ? "Succeed to save image." : "Failed to save image.");
}
```


8 函数

8.1 函数一览

8.1.1 对 DLL 的操作

即使控制器发生系统错误，也会被正常处理。

函数名	概要
LJX8IF_Initialize	初始化 DLL
LJX8IF_Finalize	执行 DLL 的结束处理
LJX8IF_GetVersion	获取 DLL 的版本

8.1.2 建立 / 断开与控制器之间的通信路径

即使控制器发生系统错误，也会被正常处理。

函数名	概要
LJX8IF_EthernetOpen	通过 Ethernet 建立连接
LJX8IF_CommunicationClose	断开连接

8.1.3 系统控制

附有 *1 的指令发生系统错误时，可能无法正常处理。

函数名	概要
LJX8IF_RebootController	重新启动控制器
LJX8IF_ReturnToFactorySetting	将控制器还原到出厂状态 *1
LJX8IF_ControlLaser	进行激光控制 *1
LJX8IF_GetError	获取控制器的系统错误信息
LJX8IF_ClearError	解除控制器的系统错误
LJX8IF_TriggerErrorReset	TRG_ERROR_RESET *1
LJX8IF_GetTriggerAndPulseCount	获取触发和脉冲计数 *1
LJX8IF_SetTimerCount	变更计时器计数值 *1
LJX8IF_GetTimerCount	获取计时器计数值 *1
LJX8IF_GetHeadTemperature	获取传感头温度 *1
LJX8IF_GetHeadModel	获取传感头型号 *1
LJX8IF_GetSerialNumber	获取序列号 *1
LJX8IF_GetAttentionStatus	获取 TRG_ERROR/MEM_FULL/TRG_PASS 的状态
LJX8IF_GetLedLightImage	获取 LED 光源图像

8.1.4 测量控制

在控制器系统错误状态下，处理失败。

函数名	概要
LJX8IF_Trigger	发送触发
LJX8IF_StartMeasure	开始批处理测量 / 开始多台控制器同步
LJX8IF_StopMeasure	停止批处理测量 / 停止多台控制器同步
LJX8IF_ClearMemory	清除内存

8.1.5 变更 / 读取设定相关

在控制器系统错误状态下，处理失败。

函数名	概要
LJX8IF_SetSetting	向控制器发送设定值
LJX8IF_GetSetting	从控制器获取设定值
LJX8IF_InitializeSetting	初始化控制器的设定值
LJX8IF_ReflectSetting	将设定写入区域的内容反映到运行设定区域及保存区域中
LJX8IF_RewriteTemporarySetting	用运行设定区域及保存区域中的设定覆盖设定写入区域的内容
LJX8IF_CheckMemoryAccess	确认是否正在对保存区域进行保存处理
LJX8IF_ChangeActiveProgram	切换活动程序编号
LJX8IF_GetActiveProgram	获取活动程序编号
LJX8IF_SetXpitch	变更轮廓的 X 间隔
LJX8IF_GetXpitch	获取轮廓的 X 间隔

8.1.6 测量结果的获取

在控制器系统错误状态下，处理失败。

函数名	概要
LJX8IF_GetProfile	获取轮廓
LJX8IF_GetBatchProfile	获取批处理轮廓
LJX8IF_GetBatchSimpleArray	获取批处理轮廓（SimpleArray）

* 为了获取轮廓数据，每次都会发送 / 接收这些指令，因此无法连续且高速地获取轮廓数据。如需连续且高速地获取轮廓数据，请使用“8.1.7 高速数据通信相关”中记载的指令。

* 由于 GetBatchSimpleArray 从轮廓数据中分别单独输出“标题信息”、“高度数据”和“亮度数据”，因此在需要转换为位图文件时非常有用。

8.1.7 高速数据通信相关

要高速且连续地获取轮廓数据时，使用以下指令。

在控制器系统错误状态下，处理失败。

函数名	概要
LJX8IF_InitializeHighSpeedDataCommunication	进行高速数据通信必需的初始化
LJX8IF_InitializeHighSpeedDataCommunicationSimpleArray	进行高速数据通信（SimpleArray）必需的初始化
LJX8IF_PreStartHighSpeedDataCommunication	请求执行高速数据通信开始前的准备
LJX8IF_StartHighSpeedDataCommunication	开始高速数据通信
LJX8IF_StopHighSpeedDataCommunication	停止高速数据通信
LJX8IF_FinalizeHighSpeedDataCommunication	执行高速数据通信的结束处理
LJX8IF_GetZUnitSimpleArray	获取 SimpleArray 用的换算系数

由于 SimpleArray 从轮廓数据中分别单独输出“标题信息”、“高度数据”和“亮度数据”，因此在需要转换为 Bitmap 时非常有用。

8.1.8 实用函数相关

提供对已获取的图像数据进行各种转换的函数。

函数名	概要
LJX8IF_CombineImage	返回多个图像数据拼接后的图像数据
LJX8IF_CircularImage	将互相转换到圆形的图像
LJX8IF_CircularImagePointCloud	将图像转换到圆形的点群数据

8.2 函数参考

所有可能返回错误信息的函数返回值均为 LONG 型。正常时返回 0（ERR_NONE），返回代码表示为下位 2BYTE（上位的 2BYTE 为预约）。

有关函数通用的返回代码，请参照“9 通用返回代码”。有关函数个别的返回代码，请参照本章各函数的说明。用 16 进制记录返回代码下位 2BYTE 的数据量（例：0x0100 等）。

8.2.1 对 DLL 的操作

■ DLL 初始化

格式	LONG LJX8IF_Initialize(void);
参数	-
返回值	无个别返回代码
说明	初始化 DLL。（请务必实施）

■ DLL 结束处理

格式	LONG LJX8IF_Finalize(void);
参数	-
返回值	无个别返回代码
说明	执行 DLL 的结束处理。（请务必实施）

■ 获取 DLL 版本

格式	LJX8IF_VERSION_INFO LJX8IF_GetVersion(void);
参数	-
返回值	DLL 的版本
说明	<p>获取 DLL 的版本。版本由 4 个值构成。</p> <pre>Typedef struct { INT nMajorNumber; INT nMinorNumber; INT nRevisionNumber; INT nBuildNumber; } LJX8IF_VERSION_INFO;</pre>

8.2.2 建立 / 断开与控制器之间的通信路径

有关“通信设备”的详情，请参照“8.2.9.1 通信设备”。

■ Ethernet 通信连接

格式	LONG LJX8IF_EthernetOpen (LONG IDDeviceId, LJX8IF_ETHERNET_CONFIG* pEthernetConfig);
参数	<p>IDDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>pEthernetConfig(in) Typedef struct { BYTE abyIpAddress[4]; 连接控制器的 IP 地址 WORD wPortNo; 连接控制器的端口编号。 BYTE reserve[2]; } LJX8IF_ETHERNET_CONFIG;</p> <p>IP 地址为 192.168.0.1 时，设定为 abyIpAddress[0]=192、 abyIpAddress[1]=168。</p>
返回值	无个别返回代码
说明	建立连接，使设备能够与通过 Ethernet 连接的控制器进行通信。

■ 断开通信路径

格式	LONG LJX8IF_CommunicationClose(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	断开 Ethernet 的连接。在未建立连接的状态下，即使进行调用也不会被作为错误进行处理。

8.2.3 系统控制

有关“通信设备”的详情，请参照“8.2.9.1 通信设备”。

■ 重新启动控制器

格式	LONG LJX8IF_RebootController(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	0x80A0：正在对保存区域进行访问
说明	重新启动控制器与各个连接的设备。正在对保存区域进行访问将作为错误处理。

■ 还原到出厂状态

格式	LONG LJX8IF_RetrurnToFactorySetting(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	将控制器的所有设定值还原到出厂状态。 处理从本 I/F 返回后，还将在控制器内部对保存区域进行写入处理。在关闭电源之前，请通过 LJX8IF_CheckMemoryAccess 函数（参照“8.1.5 变更 / 读取设定相关”）确认对保存区域的访问情况。

■ 激光控制

格式	LONG LJX8IF_ControlLaser(LONG IDeviceId, BYTE byState);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 BYTE byState(In) 状态指定（0：禁止点亮、0 以外：允许点亮）
返回值	无个别返回代码
说明	控制激光的允许点亮、禁止点亮。

■ 获取系统错误信息

格式	LONG LJX8IF_GetError (LONG IDeviceId, BYTE byReceivedMax, BYTE* pbyErrCount, WORD* pwErrCode);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byReceivedMax(in) 指定接收系统错误信息的数量上限。 (由 pwErrCode 传输的缓存大小) pbyErrCount(out) 用于接收系统错误信息数量的缓存。 pwErrCode(out) 用于接收系统错误信息的缓存。根据由新到旧的顺序, 存储 *pbyErrCount 个 (最多 byReceivedMax 个) 数量的系统错误信息。
返回值	无个别返回代码
说明	获取控制器的系统错误信息。 有关所返回的错误代码含义, 请参照 《LJ-X8000A 用户手册》。

■ 解除系统错误

格式	LONG LJX8IF_ClearError(LONG IDeviceId, WORD wErrCode);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 wErrCode(in) 需要解除的错误的错误代码。
返回值	无个别返回代码
说明	解除 “0x0085: 与上次启动时连接的传感头种类不同。” 系统错误。在发生的所有系统错误被成功解除后, 控制器才能正常开始测量。

■ TRG_ERROR_RESET

格式	LONG LJX8IF_TrgErrorReset(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	清除 TRG_ERROR 的 ON 状态。

■ 获取触发和脉冲计数

格式	LONG LJX8IF_GetTriggerAndPulseCount(LONG IDeviceId, DWORD* pdwTriggerCount, LONG* plEncoderCount);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pdwTriggerCount(out) 触发计数 (0 至 4,294,967,295) 超过上限后归 0。 plEncoderCount(out) 脉冲计数 (-2,147,483,648 至 2,147,483,647) 超过正最大值时, 变为负最大值。超过负最大值时, 则变为正最大值。
返回值	无个别返回代码
说明	获取触发计数和脉冲计数的当前值。

■ 获取传感头温度

格式	LONG LJX8IF_GetHeadTemperature(LONG IDeviceId, SHORT* pnSensorTemperature, SHORT* pnProcessorTemperature, SHORT* pnCaseTemperature);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pnSensorTemperature(out) 传感器 (CMOS) 温度 pnProcessorTemperature(out) 处理器温度 pnCaseTemperature(out) 外壳 (壳体) 温度
返回值	无个别返回代码
说明	读取传感头温度。 温度格式: 有符号。数值为 xxxxx (10 进制) 时, 表示摄氏温度 xxx . xx 度。 * LJ-V 传感头无法执行处理器、外壳的温度获取。返回 (0xFFFF)。

■ 获取传感头型号

格式	LONG LJX8IF_GetHeadModel(LONG IDeviceId, CHAR* pHeadModel);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pHeadModel(out) 传感头型号 (32byte)
返回值	无个别返回代码
说明	获取传感头型号。 1 个字符 1 byte, 以 ASCII 码存储。 没有连接传感头时, 传感头型号以空白方式返回。
备注	支持 DLL 1.3.0.0 或更高版本。

■ 获取序列号

格式	LONG LJX8IF_GetSerialNumber (LONG IDeviceId, CHAR *pControllerSerialNo, CHAR *pHeadSerialNo);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pControllerSerialNo(out) 控制器 序列号 (16byte) pHeadSerialNo(out) 传感头 序列号 (16byte)
返回值	无个别返回代码
说明	获取序列号。1 个字符 1 byte, 以 ASCII 码存储。

■ TRG_ERROR/ MEM_FULL / TRG_PASS 状态获取

格式	LONG LJX8IF_GetAttentionStatus (LONG IDeviceId, WORD* pwAttentionStatus);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pwAttentionStatus (out) ON: 1、OFF:0 0bit (LSB): TRG_ERROR 6bit: MEM_FULL 7bit: TRG_PASS
返回值	无个别返回代码
说明	获取 TRG_ERROR/ MEM_FULL / TRG_PASS 的状态。

■ 获取 LED 光源图像

格式	LONG LJX8IF_GetLedLightImage(LONG IDeviceId, BYTE byLedBrightness, BYTE byLaserBrightness, BYTE* pbyImageData)
参数	IDeviceId(in) 指定通过哪台通信设备进行通信 (1 至 100) byLedBrightness(in) 指定 LED 亮度 (1 至 100) byLaserBrightness(in) 指定激光亮度 (1 至 100) pbyImageData(out) 用于获取 LED 光源图像数据的缓存。 每像素 8 bit, 分辨率固定为 VGA (640×480)。
返回值	0x80A0: 由于 LASER_ON 端子为 OFF 等原因, 未成功进行拍摄处理
说明	获取 LED 光源图像。
备注	支持 DLL 1.2.0.0 或更高版本。 支持控制器 1.02.00 或更高版本。 使用本函数的示例应用程序可通过基恩士用户支持页面下载。

■ 更改定时器计数值

格式	LONG LJX8IF_SetTimerCount(LONG IDeviceId, DWORD dwTimerCount)
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 dwTimerCount(in) 指定向计时器计数复位的值 (0.1ms 单位)
返回值	无个别返回代码
说明	发行触发时的计时器计数值，存储到轮廓标题部中。 计时器计数，在电源启动之后通过自由运行累加， 可通过本指令将当前值改写为任意值。 例如，在使计时器计数值从零开始重新开始时，可以使用这项功能。
备注	支持 DLL 1.3.0.0 或更高版本。 支持控制器 1.02.00 或更高版本。

■ 获取定时器计数值

格式	LONG LJX8IF_GetTimerCount(LONG IDeviceId, DWORD * pdwTimerCount)
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pdwTimerCount(out) 获取计时器计数的当前值。(0.1ms 单位)
返回值	无个别返回代码
说明	获取计时器计数的当前值。
备注	支持 DLL 1.3.0.0 或更高版本。 支持控制器 1.02.00 或更高版本。

8.2.4 测量控制

有关“通信设备”的详情，请参照“8.2.9.1 通信设备”。

■ 触发

格式	LONG LJX8IF_Trigger(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	0x8080: 触发模式不是“外部触发”
说明	发送触发。

■ 开始批处理测量

格式	LONG LJX8IF_StartMeasure(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	0x8080: 批处理测量的设定为 OFF。或者, 虽然在使用多台控制器同步功能, 但在“同步主站”中未设定所指定的设备。 0x80A0: 由于 LASER_ON 端子为 OFF 等原因, 未成功进行批处理测量开始处理。
说明	批处理为 ON 时, 本指令开始批处理测量。(若批处理测量已经开始, 将不做任何处理, 也不会返回错误信息) 批处理为 OFF 且多台控制器同步功能使用中时, 本指令开始同步。同步开始后将开始测量。(接收触发)

■ 结束批处理测量

格式	LONG LJX8IF_StopMeasure(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	0x8080: 批处理测量的设定为 OFF。或者, 虽然在使用多台控制器同步功能, 但在“同步主站”中未设定所指定的设备。 0x80A0: 由于 LASER_ON 端子为 OFF 等原因, 未成功进行批处理测量结束处理
说明	批处理为 ON 时, 本指令停止批处理测量。(若批处理测量未开始, 将不做任何处理, 也不会返回错误信息) 批处理为 OFF 且多台控制器同步功能使用中时, 本指令停止同步。同步停止后将停止测量。(停止接收触发)

■ 清除内存

格式	LONG LJX8IF_ClearMemory(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	内存中保留的轮廓数据将被清除。

8.2.5 变更 / 读取设定相关

有关“通信设备”的详情，请参照“8.2.9.1 通信设备”。

■ 发送设定

该指令的使用方法请参照“11.1 发送 / 接收设定”。

格式	LONG LJX8IF_SetSetting (LONG IDeviceId, BYTE byDepth, LJX8IF_TARGET_SETTING TargetSetting, void* pData, DWORD dwDataSize, DWORD* pdwError);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byDepth(in) 指定将发送的设定值反映到哪一级别。详情请参照“8.2.9.3 设定的写入处理”。 Typedef enum { LJX8IF_SETTING_DEPTH_WRITE = 0x00, // 设定写入区域 LJX8IF_SETTING_DEPTH_RUNNING = 0x01, // 运行设定区域 LJX8IF_SETTING_DEPTH_SAVE = 0x02, // 保存区域 } LJX8IF_SETTING_DEPTH; TargetSetting(in) 作为特定发送对象的项目。各参数请参照“11.3 发送 / 接收设定项目详情”。 Typedef struct { BYTE byType; BYTE byCategory; BYTE byItem; BYTE reserve; BYTE byTarget1; BYTE byTarget2; BYTE byTarget3; BYTE byTarget4; } LJX8IF_TARGET_SETTING; pData(in) 指定用于存储所发送设定数据的缓存。 dwDataSize(in) 发送设定数据的 BYTE 单位大小。 pdwError(out) 用于接收设定错误详情的缓存（参照“8.2.9.4 设定错误详情”）。
返回值	0x8032: 指令长度错误（设定数据未达到规定大小时等）
说明	将已指定的项目设定发送到控制器。

■ 获取设定

该指令的使用方法请参照“11.1 发送 / 接收设定”。

格式	LONG LJX8IF_GetSetting (LONG IDeviceId, BYTE byDepth, LJX8IF_TARGET_SETTING TargetSetting, void* pData, DWORD dwDataSize);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byDepth(in) 指定获取哪一级别的设定值。详情请参照“8.2.9.3 设定的写入处理”。 Typedef enum { LJX8IF_SETTING_DEPTH_WRITE = 0x00, // 设定写入区域 LJX8IF_SETTING_DEPTH_RUNNING = 0x01, // 运行设定区域 LJX8IF_SETTING_DEPTH_SAVE = 0x02, // 保存区域 } LJX8IF_SETTING_DEPTH; TargetSetting(in) 作为特定获取对象的项目。各参数请参照“11.3 发送 / 接收设定项目详情”。 Typedef struct { BYTE byType; BYTE byCategory; BYTE byItem; BYTE reserve; BYTE byTarget1; BYTE byTarget2; BYTE byTarget3; BYTE byTarget4; } LJX8IF_TARGET_SETTING; pData(out) 指定用于接收所获取设定数据的缓存。 dwDataSize(in) 获取数据接收缓存的 BYTE 单位大小。
返回值	无个别返回代码
说明	从控制器获取已指定的项目设定。

■ 初始化设定

格式	LONG LJX8IF_InitializeSetting(LONG IDeviceId, BYTE byDepth, BYTE byTarget);
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>byDepth(in) 指定将初始化的设定值反映到哪一级别。详情请参照“8.2.9.3 设定的写入处理”。指定运行设定区域后，将反映到运行设定区域和设定写入区域中。指定保存区域后，将反映到保存区域、运行设定区域和设定写入区域中。</p> <pre> Typedef enum { LJX8IF_SETTING_DEPTH_WRITE = 0x00, // 设定写入区域 LJX8IF_SETTING_DEPTH_RUNNING = 0x01, // 运行设定区域 LJX8IF_SETTING_DEPTH_SAVE = 0x02, // 保存区域 } LJX8IF_SETTING_DEPTH; </pre> <p>byTarget(in) 指定将哪一项设定作为初始化对象。</p> <pre> Typedef enum { LJX8IF_INIT_SETTING_TARGET_PRG0 = 0x00, // 程序 0 LJX8IF_INIT_SETTING_TARGET_PRG1 = 0x01, // 程序 1 LJX8IF_INIT_SETTING_TARGET_PRG2 = 0x02, // 程序 2 LJX8IF_INIT_SETTING_TARGET_PRG3 = 0x03, // 程序 3 LJX8IF_INIT_SETTING_TARGET_PRG4 = 0x04, // 程序 4 LJX8IF_INIT_SETTING_TARGET_PRG5 = 0x05, // 程序 5 LJX8IF_INIT_SETTING_TARGET_PRG6 = 0x06, // 程序 6 LJX8IF_INIT_SETTING_TARGET_PRG7 = 0x07, // 程序 7 LJX8IF_INIT_SETTING_TARGET_PRG8 = 0x08, // 程序 8 LJX8IF_INIT_SETTING_TARGET_PRG9 = 0x09, // 程序 9 LJX8IF_INIT_SETTING_TARGET_PRG10 = 0x0A, // 程序 10 LJX8IF_INIT_SETTING_TARGET_PRG11 = 0x0B, // 程序 11 LJX8IF_INIT_SETTING_TARGET_PRG12 = 0x0C, // 程序 12 LJX8IF_INIT_SETTING_TARGET_PRG13 = 0x0D, // 程序 13 LJX8IF_INIT_SETTING_TARGET_PRG14 = 0x0E, // 程序 14 LJX8IF_INIT_SETTING_TARGET_PRG15 = 0x0F, // 程序 15 } LJX8IF_INIT_SETTING_TARGET; </pre>
返回值	无个别返回代码
说明	初始化被指定为初始化对象的区域设定。

■ 设定写入区域的生效请求

格式	LONG LJX8IF_ReflectSetting (LONG IDeviceId, BYTE byDepth, DWORD*pdwError);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byDepth(in) 指定将设定写入区域中写入的设定值反映到哪一级别。详情请参照“8.2.9.3 设定的写入处理”。 Typedef enum { LJX8IF_SETTING_DEPTH_RUNNING = 0x01, // 运行设定区域 LJX8IF_SETTING_DEPTH_SAVE = 0x02, // 保存区域 } LJX8IF_SETTING_DEPTH; pdwError(out) 用于接收设定错误详情的缓存（参照“8.2.9.4 设定错误详情”）。
返回值	无个别返回代码
说明	命令用本函数将设定值反映到保存区域时，请在关闭电源前使用 LJX8IF_CheckMemoryAccess 函数确认对保存区域的访问情况。

■ 更新设定写入区域

格式	LONG LJX8IF_RewriteTemporarySetting (LONG IDeviceId, BYTE byDepth);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byDepth(in) 指定用哪一级别的设定值更新设定写入区域。 Typedef enum { LJX8IF_SETTING_DEPTH_RUNNING = 0x01, // 运行设定区域 LJX8IF_SETTING_DEPTH_SAVE = 0x02, // 保存区域 } LJX8IF_SETTING_DEPTH;
返回值	无个别返回代码
说明	使用保存在运行设定区域或保存区域的设定之一更新设定写入区域的内容。

■ 确认对保存区域的保存处理情况

格式	LONG LJX8IF_CheckMemoryAccess(LONG IDeviceId, BYTE* pbyBusy);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pbyBusy(out) 用于接收是否正在访问保存区域的反馈的缓存。 0 以外：正在访问保存区域、 0：未在访问保存区域。
返回值	无个别返回代码
说明	确认控制器是否正在通过设定值保存处理等访问保存区域。 命令使用 LJX8IF_ReturnToFactorySetting 函数 / LJX8IF_SetSetting 函数 / LJX8IF_InitializeSetting 函数 / LJX8IF_ReflectSetting 函数将设定值保存到保存区域时，请在确认本函数是否已完成对保存区域的访问后再关闭电源。

■ 程序切换

格式	LONG LJX8IF_ChangeActiveProgram (LONG IDeviceId, BYTE byProgramNo);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 byProgramNo(in) 用切换后的程序编号 0 至 15 进行指定 (0：程序 0、 1：程序 1、 ...)
返回值	无个别返回代码
说明	切换活动程序编号。 对 byProgramNo 指定与活动程序编号相同的编号，或指定了无效的程序编号时，虽然会执行程序切换动作（内存清除等），但不会变更活动程序编号。

■ 获取活动程序编号

格式	LONG LJX8IF_GetActiveProgram (LONG IDeviceId, BYTE* pbyProgramNo);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pbyProgramNo(out) 存储活动程序编号 0 至 15 (0：程序 0、 1：程序 1、 ...)
返回值	无个别返回代码
说明	获取活动程序编号。

■ 变更轮廓的 X 间隔

格式	LONG LJX8IF_SetXpitch(LONG IDeviceId, DWORD dwXpitch)																								
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>dwXpitch(in) 指定轮廓数据的 X 间隔 （以 0.1 μm 为单位） 例：12.5 μm = 125</p> <table><tr><th>型号</th><th>设定范围（μm）</th><th>初始值</th></tr><tr><td>LJ-X8020</td><td>2.0 至 3.0</td><td>2.5</td></tr><tr><td>LJ-X8060</td><td>4.0 至 6.0</td><td>5.0</td></tr><tr><td>LJ-X8080</td><td>10.0 至 15.0</td><td>12.5</td></tr><tr><td>LJ-X8200*</td><td>20.0 至 30.0</td><td>25.0</td></tr><tr><td>LJ-X8400（扩展功能 OFF 时）*</td><td>50.0 至 120.0</td><td>75.0</td></tr><tr><td>LJ-X8400（扩展功能 ON 时）*</td><td>50.0 至 120.0</td><td>100.0</td></tr><tr><td>LJ-X8900*</td><td>100.0 至 250.0</td><td>225.0</td></tr></table> <p>* 以 0.5 μm 为单位进行设定。若设定单位不是 0.5 μm，则返回参数错误（0x8042）。</p>	型号	设定范围（μm）	初始值	LJ-X8020	2.0 至 3.0	2.5	LJ-X8060	4.0 至 6.0	5.0	LJ-X8080	10.0 至 15.0	12.5	LJ-X8200*	20.0 至 30.0	25.0	LJ-X8400（扩展功能 OFF 时）*	50.0 至 120.0	75.0	LJ-X8400（扩展功能 ON 时）*	50.0 至 120.0	100.0	LJ-X8900*	100.0 至 250.0	225.0
型号	设定范围（μm）	初始值																							
LJ-X8020	2.0 至 3.0	2.5																							
LJ-X8060	4.0 至 6.0	5.0																							
LJ-X8080	10.0 至 15.0	12.5																							
LJ-X8200*	20.0 至 30.0	25.0																							
LJ-X8400（扩展功能 OFF 时）*	50.0 至 120.0	75.0																							
LJ-X8400（扩展功能 ON 时）*	50.0 至 120.0	100.0																							
LJ-X8900*	100.0 至 250.0	225.0																							
返回值	无个别返回代码																								
说明	变更轮廓的 X 间隔。 不同于其他设定值，该参数无法被保存至非易失性存储器。该参数将在电源重新启动时返回初始值。																								
备注	支持 DLL 1.1.0.0 或更高版本。 支持控制器 1.01.00 或更高版本。																								

■ 获取轮廓的 X 间隔

格式	LONG LJX8IF_GetXpitch (LONG IDeviceId, DWORD * pdwXpitch)
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>pdwXpitch(out) 获取轮廓数据的 X 间隔（以 0.1 μm 为单位） 例：125 = 12.5 μm</p>
返回值	无个别返回代码
说明	获取轮廓的 X 间隔。
备注	支持 DLL 1.1.0.0 或更高版本。 支持控制器 1.01.00 或更高版本。

8.2.6 测量结果的获取

■ 获取轮廓

格式	LONG LJX8IF_GetProfile (LONG IDeviceId, LJX8IF_GET_PROFILE_REQUEST* pReq, LJX8IF_GET_PROFILE_RESPONSE* pRsp, LJX8IF_PROFILE_INFO* pProfileInfo, DWORD* pdwProfileData, DWORD dwDataSize);
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>pReq(in) 指定获取轮廓的位置等。</p> <pre>Typedef struct { BYTE byTargetBank; BYTE byPositionMode; BYTE reserve[2]; DWORD dwGetProfileNo; BYTE byGetProfileCount; BYTE byErase; BYTE reserve2[2]; } LJX8IF_GET_PROFILE_REQUEST;</pre> <p>byTargetBank 内存分配为“双缓存”时，指定是从活动区域（当前程序）还是从非活动区域（前一个程序）获取。详情请参照“8.2.9.2 内存”。</p> <pre>Typedef enum { LJX8IF_PROFILE_BANK_ACTIVE = 0x00, // 活动区域 LJX8IF_PROFILE_BANK_INACTIVE = 0x01, // 非活动区域 } LJX8IF_PROFILE_BANK;</pre> <p>byPositionMode 指定轮廓获取位置的指定方法。</p> <pre>Typedef enum { LJX8IF_PROFILE_POSITION_CURRENT = 0x00, // 从当前开始 LJX8IF_PROFILE_POSITION_OLDEST = 0x01, // 从最早的开始 LJX8IF_PROFILE_POSITION_SPEC = 0x02, // 指定位置 } LJX8IF_PROFILE_POSITION;</pre> <p>在轮廓获取指令中，表示从控制器内部保存的轮廓数据中获取哪一个轮廓的指定方法。按照先后顺序存储获取的轮廓。</p> <p><< 从当前开始 >> 获取当前轮廓。获取的最后一项轮廓将成为当前轮廓。</p> <p><< 从最早的开始 >> 获取最早的轮廓。获取的首个轮廓将成为最早的轮廓。</p> <p><< 指定位置 >> 从指定的位置开始获取指定数量的轮廓。获取的首个轮廓将成为指定位置的轮廓。</p> <p>dwGetProfileNo byPositionMode 为 LJX8IF_PROFILE_POSITION_SPEC 时，指定获取对象的轮廓编号。轮廓编号的首位从“0”开始。</p>

参数	<p>byGetProfileCount 读取的轮廓数。 根据设定不同，可读取的轮廓数有所限制。（轮廓数据点数为 3200 点且有亮度时，最多 128 个轮廓，部分设定下最多可达 255 个轮廓。）因此，可能无法获取指定数量的轮廓。此时，将返回能够获取的最大轮廓数。请通过 pRsp 的 byGetProfileCount 确认已获取的轮廓数。</p> <p>byErase 指定是否删除已读取的轮廓数据以及当前读取轮廓之前的轮廓数据。 0：不删除、1：删除。</p> <p>pRsp(out) 表示实际获取的轮廓位置等。 <pre> Typedef struct { DWORD dwCurrentProfileNo; // 当前最新的轮廓编号。 DWORD dwOldestProfileNo; DWORD dwGetTopProfileNo; BYTE byGetProfileCount; BYTE reserve[3]; } LJX8IF_GET_PROFILE_RESPONSE; </pre> </p> <p>dwOldestProfileNo 控制器保存的最早轮廓的轮廓编号。</p> <p>dwGetTopProfileNo 本次读取的内容中，最早轮廓的轮廓编号。</p> <p>byGetProfileCount 本次读取的轮廓数。</p> <p>pProfileInfo(out) 已获取轮廓的轮廓信息。 <pre> Typedef struct { BYTE byProfileCount; // 固定为 1 BYTE reserve1; BYTE byLuminanceOutput; // 有、无亮度（1：有亮度、0：无亮度） BYTE reserve2; WORD wProfileDataCount; // 轮廓的数据数 （初始设定分别为 LJ-X：3200、 LJ-V：800） BYTE reserve3[2]; LONG IXStart; // 第 1 点的 X 坐标。 LONG IXPitch; // 轮廓数据的 X 方向间隔。 } LJX8IF_PROFILE_INFO; </pre> 以 0.01 μm 为单位存储 IXStart、IXPitch。 </p> <p>pdwProfileData (out) 获取轮廓数据的缓存。 将“LJX8IF_PROFILE_HEADER- 有符号的 32 bit 轮廓数据 - LJX8IF_PROFILE_FOOTER”作为 1 个单位，只重复存储实际获取的轮廓数。</p> <p>dwDataSize(in) pdwProfileData 的 BYTE 单位大小</p>
返回值	<p>0x8081：“批处理测量为 ON”</p> <p>0x80A0：没有任何 1 个轮廓数据</p>

说明	<p>获取轮廓数据。 以 0.01 μm 为单位存储轮廓数据。 有关轮廓数据中存储的数据内容（存储顺序及大小），请参照“7 轮廓数据结构、CallBack 函数”。</p> <p>根据测量设定情况，可一次性读取的轮廓数存在上限。 请参照 pRsp 的 byGetProfileCount（本次读取的轮廓数），确认是否获取了需要获取的所有数据。未成功全部获取时，请使用本函数，指定为 pReq 的 byPositionMode=LJX8IF_PROFILE_POSITION_SPEC pReq 的 dwGetProfileNo = pRsp 的 dwGetTopProfileNo + pRsp 的 byGetProfileCount， 获取剩余的数据。（请指定从本次读取轮廓数据的下一个轮廓数据开始读取）</p>
----	---

■ 获取批处理轮廓

格式	<p>LONG LJX8IF_GetBatchProfile (LONG IDeviceId, LJX8IF_GET_BATCH_PROFILE_REQUEST* pReq, LJX8IF_GET_BATCH_PROFILE_RESPONSE* pRsp, LJX8IF_PROFILE_INFO* pProfileInfo, DWORD* pdwBatchData, DWORD dwDataSize);</p>
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>pReq(in) 指定获取轮廓的位置等。</p> <pre> Typedef struct { BYTE byTargetBank; BYTE byPositionMode; BYTE reserve[2]; DWORD dwGetBatchNo; DWORD dwGetProfileNo; BYTE byGetProfileCount; BYTE byErase; BYTE reserve2[2]; } LJX8IF_GET_BATCH_PROFILE_REQUEST; </pre> <p>byTargetBank 内存分配为“双缓存”时，指定是从活动区域还是从非活动区域获取。“活动区域”是指正在写入（运行中）轮廓数据的缓存区域。详情请参照“8.2.9.2 内存”。</p> <pre> Typedef enum { LJX8IF_PROFILE_BANK_ACTIVE = 0x00, // 活动区域 LJX8IF_PROFILE_BANK_INACTIVE = 0x01, // 非活动区域 } LJX8IF_PROFILE_BANK; </pre> <p>byPositionMode</p> <pre> Typedef enum { LJX8IF_BATCH_POSITION_CURRENT = 0x00, // 从当前开始 LJX8IF_BATCH_POSITION_SPEC = 0x02, // 指定位置 LJX8IF_BATCH_POSITION_COMMITTED = 0x03, // 从确定批次后的 // 最新开始 LJX8IF_BATCH_POSITION_CURRENT_ONLY = 0x04, // 单个最新 } LJX8IF_BATCH_POSITION; </pre> <p>在批处理轮廓获取指令中，表示从控制器内部保存的批处理数据中获取哪一批次中轮廓的指定方法。按照先后顺序存储获取的轮廓。</p>

参数	<p><< 从当前开始 >> 获取最新批处理数据中的轮廓。</p> <p><< 指定位置 >> 获取指定编号批处理数据中的轮廓。</p> <p><< 从确定批次后的最新开始 >> 获取指定批次之后最新批处理数据中的轮廓。仅获取最新批处理数据中的最后一个轮廓。</p> <p>dwGetBatchNo byPositionMode 为 LJX8IF_BATCH_POSITION_SPEC 时，指定要获取的轮廓是第几批。批次编号的首位从“0”开始。</p> <p>dwGetProfileNo 指定批次编号中的获取起始轮廓编号。轮廓编号的首位从“0”开始。</p> <p>byGetProfileCount 读取的轮廓数。</p> <p>byErase 指定是否删除已读取的批处理数据以及当前读取批次之前的批处理数据。 0：不删除、1：删除。</p> <p>pRsp(out) 表示实际获取的轮廓位置等。 Typedef struct { DWORD dwCurrentBatchNo; DWORD dwCurrentBatchProfileCount; DWORD dwOldestBatchNo; DWORD dwOldestBatchProfileCount; DWORD dwGetBatchNo; DWORD dwGetBatchProfileCount; DWORD dwGetBatchTopProfileNo; BYTE byGetProfileCount; BYTE byCurrentBatchCommitted; BYTE reserve[2]; } LJX8IF_GET_BATCH_PROFILE_RESPONSE;</p> <p>批处理测量 ON 时，对轮廓获取指令返回的轮廓信息。</p> <p>dwCurrentBatchNo 当前最新的批次编号。</p> <p>dwCurrentBatchProfileCount 最新批次中的轮廓数。</p> <p>dwOldestBatchNo 控制器保存的最早批次的批次编号。</p> <p>dwOldestBatchProfileCount 控制器保存的最早批次中的轮廓数。</p> <p>dwGetBatchNo 本次读取的批次编号。</p> <p>dwGetBatchProfileCount 本次读取批次中的轮廓数。</p> <p>dwGetBatchTopProfileNo 表示本次读取的内容中，最早的轮廓是批次内的第几项轮廓。</p>
----	---

参数	<p>byGetProfileCount 本次读取的轮廓数。（轮廓数据点数为 3200 点且有亮度时，最多 128 个轮廓，部分设定下最多可达 255 个轮廓。）</p> <p>byCurrentBatchCommitted 表示最新批次编号的批处理测量是否完毕。 0：未完毕、1：完毕</p> <p>pProfileInfo(out) 已获取轮廓的轮廓信息。 Typedef struct { BYTE byProfileCount; // 固定为 1 BYTE reserve1; BYTE byLuminanceOutput; // 有、无亮度（1：有亮度、0：无亮度） BYTE reserve2; WORD wProfileDataCount; // 轮廓的数据数 （初始设定分别为 LJ-V：800、LJ-X：3200） BYTE reserve3[2]; LONG IXStart; // 第 1 点的 X 坐标。 LONG IXPitch; // 轮廓数据的 X 方向间隔。 } LJX8IF_PROFILE_INFO; 以 0.01 μm 为单位存储 IXStart、IXPitch。</p> <p>pdwBatchData(out) 获取轮廓数据的缓存。 将“LJX8IF_PROFILE_HEADER- 有符号的 32 bit 轮廓数据 - LJX8IF_PROFILE_FOOTER”作为 1 个单位，只重复存储实际获取的轮廓数。</p> <p>dwDataSize(in) pdwProfileData 的 BYTE 单位大小</p>
返回值	<p>0x8081：“批处理测量不为 ON” 0x80A0：没有任何 1 个批处理数据（未进行过批处理测量）</p>
说明	<p>获取轮廓数据。 以 0.01 μm 为单位存储轮廓数据。 有关轮廓数据中存储的数据内容（存储顺序及大小），请参照“7 轮廓数据结构、CallBack 函数”。</p> <p>读取 1 个批次内的所有轮廓时，请通过下列步骤进行读取。 (1) 对 pReq 的 byPositionMode 指定 LJX8IF_BATCH_POSITION_CURRENT，调用本函数。保存读取的轮廓起始位置及数量、读取的批次编号。 (2) 对 pReq 的设定进行如下更新，再次调用本函数。 byPositionMode = LJX8IF_BATCH_POSITION_SPEC dwGetBatchNo = 读取的批次编号 byGetProfileNo = 批次内仍未读取的轮廓数据中第一个轮廓的编号 (3)更新 (2) 的 dwGetProfileNo，调用本函数，直到读取出批次内的所有轮廓。</p>

■ 获取批处理轮廓（Simple Array）

格式	<p>LONG LJX8IF_GetBatchSimpleArray(LONG IDeviceld, LJX8IF_GET_BATCH_PROFILE_REQUEST* pReq, LJX8IF_GET_BATCH_PROFILE_RESPONSE* pRsp, LJX8IF_PROFILE_INFO* pProfileInfo, LJX8IF_PROFILE_HEADER* pProfileHeaderArray, WORD* pHeightProfileArray, WORD* pLuminanceProfileArray);</p>
参数	<p>IDeviceld(in) 指定通过哪台通信设备进行通信。</p> <p>pReq(in) 指定获取轮廓的位置等。</p> <pre> Typedef struct { BYTE byTargetBank; BYTE byPositionMode; BYTE reserve[2]; DWORD dwGetBatchNo; DWORD dwGetProfileNo; BYTE byGetProfileCount; BYTE byErase; BYTE reserve2[2]; } LJX8IF_GET_BATCH_PROFILE_REQUEST; </pre> <p>byTargetBank 内存分配为“双缓存”时，指定是从活动区域还是从非活动区域获取。“活动区域”是指正在写入（运行中）轮廓数据的缓存区域。详情请参照“8.2.9.2 内存”。</p> <pre> Typedef enum { LJX8IF_PROFILE_BANK_ACTIVE = 0x00, // 活动区域 LJX8IF_PROFILE_BANK_INACTIVE = 0x01, // 非活动区域 } LJX8IF_PROFILE_BANK; </pre> <p>byPositionMode Typedef enum { LJX8IF_BATCH_POSITION_CURRENT = 0x00, // 从当前开始 LJX8IF_BATCH_POSITION_SPEC = 0x02, // 指定位置 LJX8IF_BATCH_POSITION_COMMITTED = 0x03, // 从确定批次后的 最新开始 LJX8IF_BATCH_POSITION_CURRENT_ONLY = 0x04, // 单个最新 } LJX8IF_BATCH_POSITION; <p>在批处理轮廓获取指令中，表示从控制器内部保存的批处理数据中获取哪一批次中轮廓的指定方法。按照先后顺序存储获取的轮廓。</p> <p><< 从当前开始 >> 获取最新批处理数据中的轮廓。</p> <p><< 指定位置 >> 获取指定编号批处理数据中的轮廓。</p> <p><< 从确定批次后的最新开始 >> 获取指定批次之后最新批处理数据中的轮廓。仅获取最新批处理数据中的最后一个轮廓。</p> <p>dwGetBatchNo byPositionMode 为 LJX8IF_BATCH_POSITION_SPEC 时，指定要获取的轮廓是第几批。批次编号的首位从“0”开始。</p> <p>dwGetProfileNo 指定批次编号中的获取起始轮廓编号。轮廓编号的首位从“0”开始。</p> </p>

参数	<p>byGetProfileCount 读取的轮廓数。（轮廓数据点数为 3200 点且有亮度时，最多 128 个轮廓，部分设定下最多可达 255 个轮廓。） 因此，可能无法获取指定数量的轮廓。此时，将返回能够获取的最大轮廓数。请通过 pRsp 的 byGetProfileCount 确认已获取的轮廓数。</p> <p>byErase 指定是否删除已读取的批处理数据以及当前读取批次之前的批处理数据。 0：不删除、1：删除。</p> <p>pRsp(out) 表示实际获取的轮廓位置等。 Typedef struct { DWORD dwCurrentBatchNo; DWORD dwCurrentBatchProfileCount; DWORD dwOldestBatchNo; DWORD dwOldestBatchProfileCount; DWORD dwGetBatchNo; DWORD dwGetBatchProfileCount; DWORD dwGetBatchTopProfileNo; BYTE byGetProfileCount; BYTE byCurrentBatchCommitted; BYTE reserve[2]; } LJX8IF_GET_BATCH_PROFILE_RESPONSE;</p> <p>批处理测量 ON 时，对轮廓获取指令返回的轮廓信息。</p> <p>dwCurrentBatchNo 当前最新的批次编号。</p> <p>dwCurrentBatchProfileCount 最新批次中的轮廓数。</p> <p>dwOldestBatchNo 控制器保存的最早批次的批次编号。</p> <p>dwOldestBatchProfileCount 控制器保存的最早批次中的轮廓数。</p> <p>dwGetBatchNo 本次读取的批次编号。</p> <p>dwGetBatchProfileCount 本次读取批次中的轮廓数。</p> <p>dwGetBatchTopProfileNo 表示本次读取的内容中，最早的轮廓是批次内的第几项轮廓。</p> <p>byGetProfileCount 本次读取的轮廓数。</p> <p>byCurrentBatchCommitted 表示最新批次编号的批处理测量是否完毕。 0：未完毕、1：完毕</p>
----	---

参数

```
typedef struct {  
    BYTE    byProfileCount;           // 固定为 1  
    BYTE    reserve1;  
    BYTE    BYTE byLuminanceOutput;   // 有、无亮度  
                                         (1: 有亮度、0: 无亮度)  
  
    BYTE    reserve2;  
    WORD    wProfileDataCount;        // 轮廓的数据数（初始设定分别为  
                                         LJ-V: 800、LJ-X: 3200）  
  
    BYTE    reserve3[2];  
    LONG    IXStart;                  // 第 1 点的 X 坐标。  
    LONG    IXPitch;                 // 轮廓数据的 X 方向间隔。  
}  
LJX8IF_PROFILE_INFO;
```

pProfileHeaderArray(out)

pHeightProfileArray(out)

$$\text{高度}(\mu\text{m}) = (\text{存储的值} - 32768) * \text{换算系数}$$

LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900
0.4	0.8	1.6	4	8	16

LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)
0.4	0.6	1.6	4	8

无效数据、死角数据和判断待机数据存储 0。

指向存储轮廓数据中所包含“亮度数据”（排列）缓存的指针。只重复存储实际获取的轮廓数（byGetProfileCount）。存储 0 至 1023 范围内的值。

0x8081: “批处理测量不为 ON”
0x80A0: 没有任何 1 个批处理数据 (未进行过批处理测量)

说明	<p>获取轮廓数据。pProfileHeaderArray、pHeightProfileArray、pLuminanceProfileArray 中分别存储排列数量的数据。</p> <p>读取 1 个批次内的所有轮廓时，请通过下列步骤进行读取。</p> <p>(1)对 pReq 的 byPositionMode 指定 LJX8IF_BATCH_POSITION_CURRENT，调用本函数。保存读取的轮廓起始位置及数量、读取的批次编号。</p> <p>(2)对 pReq 的设定进行如下更新，再次调用本函数。</p> <p style="padding-left: 20px;">byPositionMode = LJX8IF_BATCH_POSITION_SPEC</p> <p style="padding-left: 20px;">dwGetBatchNo = 读取的批次编号</p> <p style="padding-left: 20px;">byGetProfileNo = 批次内仍未读取的轮廓数据中第一个轮廓的编号</p> <p>(3)更新 (2) 的 dwGetProfileNo，调用本函数，直到读取出批次内的所有轮廓。</p>
----	---

8.2.7 高速数据通信相关

用于高速且连续地获取轮廓数据的指令。

有关通过高速数据通信获取的轮廓数据的存储位置，请参照“7.4 返回函数 I/F 定义”。

有关“通信设备”的详情，请参照“8.2.9.1 通信设备”。

■ 初始化 Ethernet 高速数据通信

格式	LONG LJX8IF_InitializeHighSpeedDataCommunication (LONG IDeviceId, LJX8IF_ETHERNET_CONFIG* pEthernetConfig, WORD wHighSpeedPortNo, void (*pCallback)(BYTE*, DWORD, DWORD, DWORD, DWORD), DWORD dwProfileCount, DWORD dwThreadId);
参数	<p>IDeviceId(in) 指定通过哪台通信设备进行通信。</p> <p>pEthernetConfig(in) Typedef struct { BYTE abyIpAddress[4]; 连接控制器的 IP 地址 WORD wPortNo; 连接控制器的端口编号。 BYTE reserve[2]; } LJX8IF_ETHERNET_CONFIG;</p> <p>IP 地址为 192.168.0.1 时，设定为 abyIpAddress[0]=192、abyIpAddress[1]=168。</p> <p>wHighSpeedPortNo(in) 指定用于高速通信的端口编号。</p> <p>pCallback(in) 指定于通过高速通信接收数据时调用的返回函数。</p> <p>dwProfileCount(in) 指定调用返回函数的频率。接收指定个数的轮廓时，调用返回函数。</p> <p>dwThreadId(in) (含义同 dwUser) 存储 ID。</p>
返回值	无个别返回代码
说明	<p>对通过 Ethernet 连接的控制器的，进行高速数据通信的初始化。与常规的指令发送 / 接收不同，高速通信需要确保专用的通信路径。本函数开设该高速通信专用的通信路径。常规的指令发送 / 接收和高速通信需要设定不同的 TCP/IP 端口编号（有关端口编号的设定方法，请参照 LJ-X8000A 系列用户手册）。</p>

■ 初始化 Ethernet 高速数据通信（SimpleArray）

格式	LONG LJX8IF_InitializeHighSpeedDataCommunicationSimpleArray (LONG IDeviceId, LJX8IF_ETHERNET_CONFIG* pEthernetConfig, WORD wHighSpeedPortNo, LJX8IF_CALLBACK_SIMPLE_ARRAY pCallBackSimpleArray, DWORD dwProfileCount, DWORD dwThreadId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pEthernetConfig(in) Typedef struct { BYTE abyIpAddress[4]; 连接控制器的 IP 地址 WORD wPortNo; 连接控制器的端口编号。 BYTE reserve[2]; } LJX8IF_ETHERNET_CONFIG; IP 地址为 192.168.0.1 时, 设定为 abyIpAddress[0]=192、 abyIpAddress[1]=168。 wHighSpeedPortNo(in) 指定用于高速通信的端口编号。 pCallBackSimpleArray(in) 指定于通过高速通信接收数据时调用的返回函数（SimpleArray）。 dwProfileCount(in) 指定调用返回函数的频率。接收指定个数的轮廓时, 调用返回函数。 dwThreadId(in)（含义同 dwUser） 存储 ID。
返回值	无个别返回代码
说明	对通过 Ethernet 连接的控制器, 进行高速数据通信（SimpleArray）的初始化。 与常规的指令发送 / 接收不同, 高速通信需要确保专用的通信路径。本函数开设该高速通信专用的通信路径。 常规的指令发送 / 接收和高速通信需要设定不同的 TCP/IP 端口编号 （有关端口编号的设定方法, 请参照 LJ-X8000A 系列用户手册）。

■ 高速数据通信开始前准备请求

格式	LONG LJX8IF_PreStartHighSpeedDataCommunication (LONG IDeviceId, LJX8IF_HIGH_SPEED_PRE_START_REQ* pReq, LJX8IF_PROFILE_INFO*pProfileInfo);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pReq(in) 指定高速通信从哪项数据开始发送。 Typedef struct { BYTE bySendPosition; BYTE reserve[3]; } LJX8IF_HIGH_SPEED_PRE_START_REQ; bySendPosition 发送起始位置。 0: 从上次发送完毕的位置开始（首次执行就从最早的数据开始）、 1: 从最早的数据开始（重新操作）、 2: 从下一数据开始 pProfileInfo(out) 存储轮廓信息。
返回值	0x8081: 被指定为发送起始位置的数据不存在 0x80A1: 已在进行高速数据通信
说明	进行高速数据通信的开始准备。高速数据通信时，控制器会在发送获取轮廓后，从其内存中删除已发送的轮廓。如需通过控制器连续获取轮廓，请确保 PC 读取轮廓的速度比控制器内存存储数据的速度更快。

■ 开始高速数据通信

格式	LONG LJX8IF_StartHighSpeedDataCommunication(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	0x80A0: 未建立高速数据通信用的连接 0x80A1: 已在进行高速数据通信 0x80A2、0x80A3: 未进行高速数据通信开始前准备 0x80A4: 发送对象数据被清除
说明	开始高速数据通信。 高速数据通信未正常运行时，请参照“12.4 高速数据通信的故障排除”。

■ 停止高速数据通信

格式	LONG LJX8IF_StopHighSpeedDataCommunication(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	停止高速数据通信。

■ 结束高速数据通信

格式	LONG LJX8IF_FinalizeHighSpeedDataCommunication(LONG IDeviceId);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。
返回值	无个别返回代码
说明	结束高速数据通信。

■ 获取换算系数

格式	LONG LJX8IF_GetZUnitSimpleArray(LONG IDeviceId, WORD* pwZUnit);
参数	IDeviceId(in) 指定通过哪台通信设备进行通信。 pwZUnit(out) 以整数表现的换算系数 * 将位于 P.40 的记载表中的值扩大为 100 倍后，以整数方式显示的值。 (例) 在 LJ-X8020 时，将向 pwZUnit 返回 40 这个值。 将存储在 SimpleArray 中的值 (0 至 65535) 转换到高度数据 (μm) 时，请使用以下的计算公式。 高度 (μm) = (存储的值 - 32768) * pwZUnit / 100
返回值	0x80A0: 还未执行过高速数据通信
说明	在将 SimpleArray 格式的数据转换到高度数据 (μm) 之际，使用这个函数。 以通过最后进行的高速数据通信获取到的信息为基础，计算换算系数。 如果从未进行高速数据通信，将返回错误。
备注	支持 DLL 1.3.0.0 或更高版本。

8.2.8 实用函数相关

■ 多个图像拼接

格式	LONG LJX8IF_CombineImage(BYTE bPixel, LONG INumX, LONG INumY, LONG ISizeX, LONG ISizeY, LONG* pOpSort, BYTE* pbOpInX, BYTE* pbOpInY, LONG* pOpCutX, LONG* pOpGeoX, LONG* pOpGeoY, void* pSourceImage, void* pDestImage);																													
参数	bPixel(in) 每个像素的字节数 (1、2、4) 使用通过 SimpleArray 格式获取的图像数据时, 每个像素的字节数为 2。																													
	INumX(in)、INumY(in) X 方向 /Y 方向的图像张数。正的整数。 输入图像的张数为 INumX×INumY。																													
	ISizeX(in)、ISizeY(in) 每个输入图像的 X 方向 /Y 方向的像素数。正的整数。																													
	pOpSort(in) 顺序信息。正的整数型排列, 对象数与输入图像张数相同。 输出图像根据输入图像的张数, 向区域分配编号。 通过这个编号的排列指定输入图像的配置处, 便可自由配置。 (例) INumX=3、INumY=2 时, 输入图像为 3 × 2 = 6 张。 在输出图像中, 存在与 6 张图像相应的 6 处区域 (#0 ~ #5) 如下所示, 编号按照从左上向右下逐渐变大的方式分配。																													
	<div><table><tr><td>#0</td><td>#1</td><td>#2</td></tr><tr><td>#3</td><td>#4</td><td>#5</td></tr></table><div><div>→ X</div><div>↓ Y</div></div><p>向顺序信息指定区域的编号, 便可改变输出图像的配置。</p><div><div><p>输入图像</p><table><tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr></table><p>pOpSort={ 0, 1, 2, 3, 4, 5, }</p></div><div><p>输出图像</p><table><tr><td>A</td><td>B</td><td>C</td></tr><tr><td>D</td><td>E</td><td>F</td></tr></table></div></div><div><div><p>输入图像</p><table><tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr></table><p>pOpSort={ 0, 3, 1, 4, 2, 5, }</p></div><div><p>输出图像</p><table><tr><td>A</td><td>C</td><td>E</td></tr><tr><td>B</td><td>D</td><td>F</td></tr></table></div></div></div>	#0	#1	#2	#3	#4	#5	A	B	C	D	E	F	A	B	C	D	E	F	A	B	C	D	E	F	A	C	E	B	D
#0	#1	#2																												
#3	#4	#5																												
A	B	C	D	E	F																									
A	B	C																												
D	E	F																												
A	B	C	D	E	F																									
A	C	E																												
B	D	F																												
pbOpInX(in)、pbOpInY(in) X 方向 /Y 方向的反转信息。在整数型的排列下, 对象数与输入图像张数相同。 使指定部位的图像数据反转。0= 不反转、0 以外 (1)= 反转。																														
(例) 使第 2 张的输入图像向 X 方向反转。																														
<div><div><p>输入图像</p><table><tr><td>←</td><td>←</td><td>←</td><td>←</td><td>←</td><td>←</td></tr></table><p>pOpSort={ 0, 1, 0, 0, 0, 0, }</p></div><div><p>输出图像</p><table><tr><td>←</td><td>→</td><td>←</td></tr><tr><td>←</td><td>←</td><td>←</td></tr></table></div></div>	←	←	←	←	←	←	←	→	←	←	←	←																		
←	←	←	←	←	←																									
←	→	←																												
←	←	←																												

pOpCutX(in)

切割信息。在整数型的排列下，对象数与输入张数相同。
针对指定位置的图像数据，忽视从两端起的指定像素数（不执行数据复制）。

在 X 方向的测量范围外，当发生类似输入图像两端为无效数据的情形时，切割无效数据，便可消除相邻图像之间的缝隙而紧密排列有效数据。

（例）在输入图像张数 =6 的情形下，切割第 3 张的图像两端 400 像素时，
指定为 pOpCutX = { 0, 0, 400, 0, 0, 0 }。

pOpGeoX(in)、pOpGeoY(in)

X 方向 /Y 方向的偏移量信息。在整数型的排列下，要素数与输入张数相同。
使指定部位的图像数据相应指定像素数执行偏移。




- 通过针对相邻图像的相对值指定。
- 数据发生重复的部位，将优先序号较大的图像的数据。
- 通过偏移，输出图像大小将忽视突出的部分（不执行数据复制）。

（例）没有偏移（将偏移信息全部都视为 0 时）
如以下所示，在与邻接图像没有缝隙 / 重复的情形下配置。

将 A 至 F 的 6 张图像配置为 X x Y = 3 x 2 的示例

A	A	A	B	B	B	C	C	C
A	A	A	B	B	B	C	C	C
A	A	A	B	B	B	C	C	C
D	D	D	E	E	E	F	F	F
D	D	D	E	E	E	F	F	F
D	D	D	E	E	E	F	F	F

- （例）使第 2 张图像向 X 方向的左侧（负侧）偏移 1 像素
- 指定为 pOpGeoX={0,-1,0,0,0,0}
 - 在第 1 张 / 第 2 张图像数据发生重复的部位（斜线），将优先指数较大的第 2 张图像。
 - 第 3 张图像不存在与第 2 张图像相对的偏移，因此与第 2 张图像一同向左方偏移。

A	A		B	B	C	C	C	
A	A		B	B	C	C	C	
A	A		B	B	C	C	C	
D	D	D	E	E	E	F	F	F
D	D	D	E	E	E	F	F	F
D	D	D	E	E	E	F	F	F

参数	<p>(例) 使第 2 张图像向 Y 方向的下方 (正侧) 偏移 2 像素</p> <ul style="list-style-type: none">指定为 pOpGeoY={0,2,0,0,0}第 5 张图像不存在与第 2 张图像相对的偏移, 因此与第 2 张图像一同向下方偏移。超过的图像大小的部分, 突出部分 (斜线部) 不会包含在输出图像中。																																																																								
	<table><tr><td>A</td><td>A</td><td>A</td><td></td><td></td><td></td><td>C</td><td>C</td><td>C</td></tr><tr><td>A</td><td>A</td><td>A</td><td></td><td></td><td></td><td>C</td><td>C</td><td>C</td></tr><tr><td>A</td><td>A</td><td>A</td><td>B</td><td>B</td><td>B</td><td>C</td><td>C</td><td>C</td></tr><tr><td>D</td><td>D</td><td>D</td><td>B</td><td>B</td><td>B</td><td>F</td><td>F</td><td>F</td></tr><tr><td>D</td><td>D</td><td>D</td><td>B</td><td>B</td><td>B</td><td>F</td><td>F</td><td>F</td></tr><tr><td>D</td><td>D</td><td>D</td><td>E</td><td>E</td><td>E</td><td>F</td><td>F</td><td>F</td></tr><tr><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr></table>	A	A	A				C	C	C	A	A	A				C	C	C	A	A	A	B	B	B	C	C	C	D	D	D	B	B	B	F	F	F	D	D	D	B	B	B	F	F	F	D	D	D	E	E	E	F	F	F				X	X	X							X	X	X			
	A	A	A				C	C	C																																																																
	A	A	A				C	C	C																																																																
A	A	A	B	B	B	C	C	C																																																																	
D	D	D	B	B	B	F	F	F																																																																	
D	D	D	B	B	B	F	F	F																																																																	
D	D	D	E	E	E	F	F	F																																																																	
			X	X	X																																																																				
			X	X	X																																																																				
<p>pSourceImage(in)</p> <p>输入图像数据指针。每张的 byPixel x ISizeX x ISizeY (byte) 排列数据, 相应图像张数 (INumX x INumY) 而保存的内存区域。</p> <p>pDestImage(out)</p> <p>输出图像数据指针。与输入图像为相同大小的内存区域。</p> <p>(注意) 需要在函数调用前确保内存。</p> <p>在偏移之后, 不针对超出输出图像大小的突出部位执行数据复制。</p>																																																																									
返回值	<p>0X1006: 参数传输错误</p> <p>1 个像素的字节数: 指定了 1、2、4 以外的值</p> <p>图像张数: 指定了负值</p> <p>顺序信息: 指定了负值, 或者超出了图像张数的序号</p>																																																																								
说明	编写连接了多个图像数据的图像数据。																																																																								
备注	<p>支持 DLL 1.3.0.0 或更高版本。</p> <p>使用本函数的示例应用程序 (仅限执行文件) 可通过基恩士用户支持页面下载。</p>																																																																								

■ 圆形转换

格式	<p>Long LJX8IF_CircularImage(LONG IXPointNum, LONG IProfileNum, FLOAT fCircleCenterXUm, FLOAT fThetaPitchDeg, FLOAT fTiltCorrectDeg, FLOAT fXPitchUm, FLOAT fZUnit, BYTE byImageType, LONG IDestXPointNum, LONG IDestYPointNumP, IDestYPointNumM, void* pSourceImage, void* pDestImage);</p>
参数	<p>将旋转工件扫描获取到的长方形图像数据, 转换为对应实际形状的圆形图像。由此能够获取到不会发生失真的图像, 可以在字符识别等的预处理上使用。</p> <div data-bbox="454 1736 1356 2027"> </div>

IXPointNum(in)

输入图像的 X 方向点数。正的整数。

IProfileNum(in)

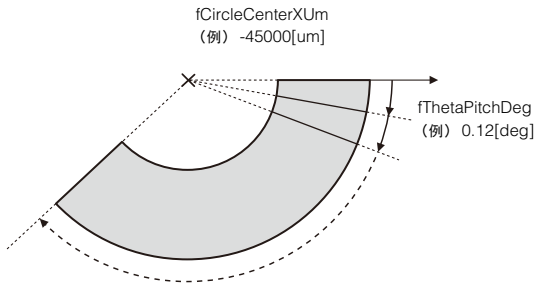
输入图像的轮廓数量。正的整数。

fCircleCenterXUm(in)

圆的中心。单位： μm 。

将输入轮廓的 X 方向中心像素视为 0.0，将向右方向视为正、将向左方向视为负。

当 IXPointNum = 3200，将第 1600 点视为 0.0。



fThetaPitchDeg(in)

旋转节距。单位：度。

将顺时针视为正、将逆时针视为负。

fTiltCorrectDeg(in)

倾斜补正角度。单位：度。

将顺时针视为正、将逆时针视为负。

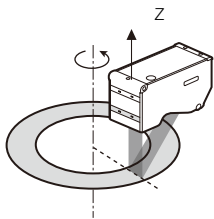
* 只在 byImageType = 0(高度图像)时使用。

旋转轴不与 Z 轴平行时，原本应该为平坦形状的数据，现在变形为盆状或圆锥状。

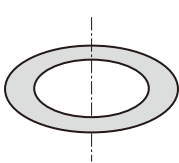
指定补正角度，便可补正为平坦的形状数据。

不采用旋转计算，而是减去 $\tan(fTiltCorrect)$ 的值，由此去除倾斜。

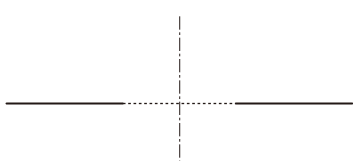
无倾斜时



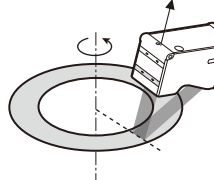
形状数据



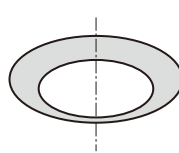
截面



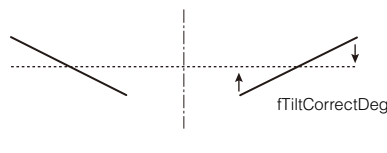
倾斜时



形状数据



截面



fXPitchUm(in)

输入图像的 X 方向间隔。单位：μm。正值。

通过通信获取高度图像时，可参照以下：

- 参照 LJX8IF_PROFILE_INFO 结构的 IXPitch 变量 (单位：0.01μm)
- 通过 LJX8IF_GetXpitch 指令获取 (单位：0.1μm)

在离线状态下转换数据时，请参考下表。

在以下的情形下，初始值有可能会与值有所不同。

- 通过 LJX8IF_SetXpitch 指令变更了 X 节距时 (只限 LJ-X 传感头)
- 使用“间隔 (X 轴)”以及“Binning”功能变更了数据间隔时

型号	初始值 (μm)
LJ-X8020	2.5
LJ-X8060	5.0
LJ-X8080	12.5
LJ-X8200	25.0
LJ-X8400 (扩展功能 OFF)	75.0
LJ-X8400 (扩展功能 ON)	100.0
LJ-X8900	225.0
LJ-V7020(K)(B)	10.0
LJ-V7060(K)(B)	20.0
LJ-V7080(B)	50.0
LJ-V7200(B)	100.0
LJ-V7300(B)	300.0

fZUnit(in)

输入图像的 Z 方向间隔 (换算系数)。单位：μm。正值。

* 只在 byImageType = 0 (高度图像) 时使用。

通过通信获取高度图像时，能够参照以下。

- 通过 LJX8IF_GetZUnitSimpleArray 指令获取 (单位：0.01μm)

在离线状态下转换数据时

LJ-X 传感头的换算系数

LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900
0.4	0.8	1.6	4	8	16

LJ-X 传感头的换算系数

LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)
0.4	0.8	1.6	4	8

byImageType(in)

图像的种类。0: 高度图像 1: 亮度图像

* 指定了亮度图像时，在函数内不使用以下的参数。

- fTiltCorrectDeg: 倾斜校正角度
- Zunit: 输入图像的 Z 方向间隔

IDestXPointNum(in)、IDestYPointNumP(in)、IDestYPointNumM(in)

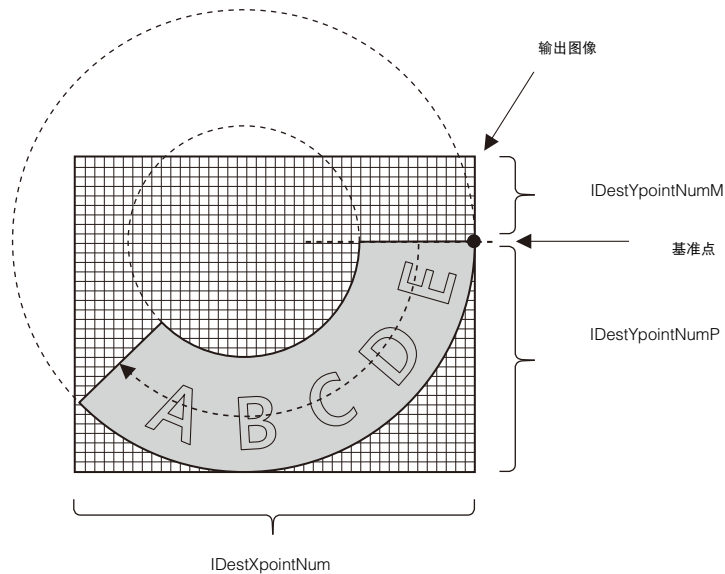
输出图像的 X 方向像素数、Y 方向像素数 (P 正方向)、Y 方向像素数 (M 负方向) 通过正的整数指定。

输出图像，为通过 XY 的数据间隔 fXPitchUm 施加了重新取样的图像。

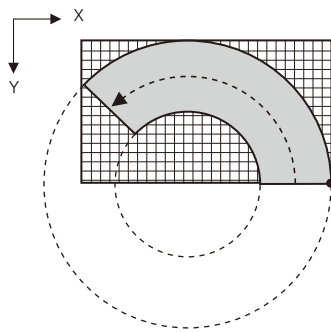
如例 1 的图片那样，能够通过从基准点起的像素数指定输出图像范围。

* 基准点：在旋转开始位置下，轮廓数据的最外侧点。

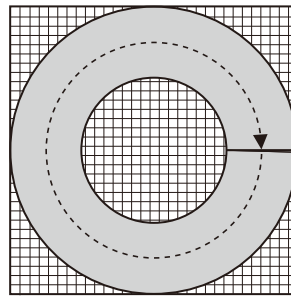
(例 1)



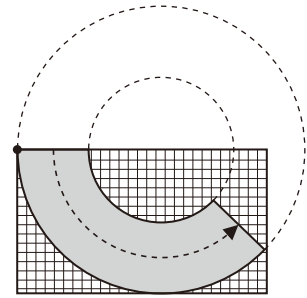
(例 2)



(例 3)



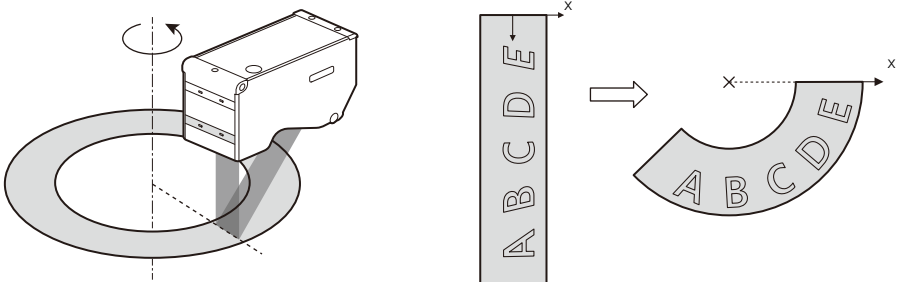
(例 4)



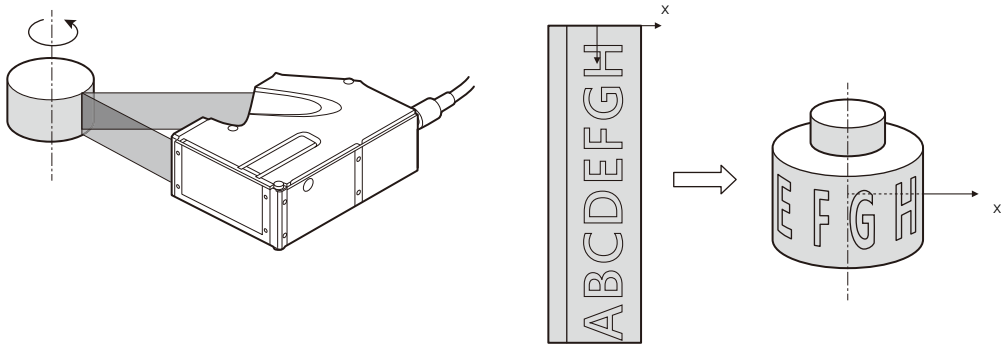
(例 2) 对 fThetaPitchDeg(旋转节距) 指定了负值时。
逆时针旋转，因此转换后的数据偏向于 Y 方向的负侧。
使用 IDistYpointNumM，便可限定存在数据的领域范围后执行切出。

(例 3) 存在 360 度全范围的数据时。
转换后的数据，在 Y 方向的正侧 / 负侧的双方上都存在。
使用 IDistYpointNumP、IDistYpointNumM 这双方切出数据。

(例 4) 对 fCircleCenterXUm(圆的中心) 指定了正值时。
在旋转开始位置上，轮廓的最外侧点将会转为 X 方向的负侧，因此基准点将发生变化。
请注意，即使在这种情形下，也通过正值指定 IDistXpointNum。

参数	<p>pSourceImage(in) 输入图像。IXPointNum×IProfileNum 的 16bit 排列数据。 使用通过 SimpleArray 格式获取的高度数据或者亮度数据。</p> <p>pDestImage(out) 输出图像。IDestXPointNum×(IDestYPointNumP + IDestYPointNumM) 的 16bit 的排列数据。 (注意) 需要在函数调用侧确保内存。 X、Y 方向的数据间隔, 与输入图像的 X 方向数据间隔 (fXPitchUm) 相同。 Z 方向的数据间隔, 与输入图像的 Z 方向数据间隔 (fZUnit) 相同。</p>
返回值	<p>0X1006: 参数传输错误 输入图像的轮廓数量: 指定了负值 输入图像的 X 方向间隔: 指定了负值 输入图像的 Z 方向间隔: 指定了负值 输入图像的 X 方向像素数: 指定了负值 输出图像的 Y 方向像素数 (P/M): 指定了负值</p>
说明	<p>将旋转工件扫描获取到的长方形图像数据, 转换为对应实际形状的圆形图像。 由此能够获取到不会发生失真的图像, 可以在字符识别等的预处理上使用。</p> 
备注	<p>支持 DLL 1.3.0.0 或更高版本。 使用本函数的示例应用程序 (仅限执行文件) 可通过基恩士用户支持页面下载。</p>

■ 圆形转换 (点云)

格式	<p>Long LJX8IF_CircularImagePointCloud(LONG IXPointNum, LONG IProfileNum, FLOAT fCircleCenterXUm, FLOAT fThetaPitchDeg, FLOAT fTiltCorrectDeg, FLOAT fViewAngleDeg, FLOAT fXPitchUm, FLOAT fZUnit, void* pSourceImage, FLOAT* pfDestData);</p>
参数	<p>在对旋转工件进行扫描后获取到的长方形图像数据, 能够转换到与实际形状相符的点云数据。 能够作为执行 3D 显示以及 3D 之际的预处理而使用。</p> 

IXPointNum(in)

输入图像的 X 方向点数。正的整数。

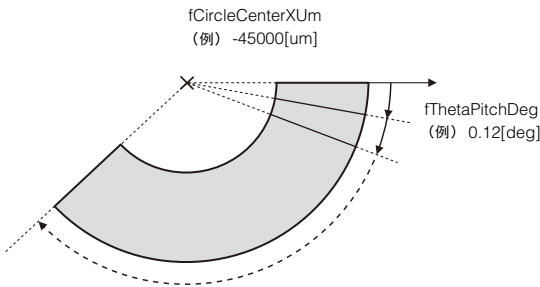
IProfileNum(in)

输入图像的轮廓数量。正的整数。

fCircleCenterXUm(in)

圆的中心。单位： μm 。

将输入轮廓的 X 方向中心像素视为 0.0，将向右方向视为正、将向左方向视为负。
当 IXPointNum = 3200，将第 1600 点视为 0.0。



(注意) 使用了 ViewAngle 时，通过视点转换后的坐标系统指定。
例如，在使传感头横倒 0 度的状态下使用时，视点转换后的 X 方向距离相当于转换视点之前的 Z 方向距离。
详细内容，请参照本说明栏末尾的“关于函数内部的处理顺序”的记载。

fThetaPitchDeg(in)

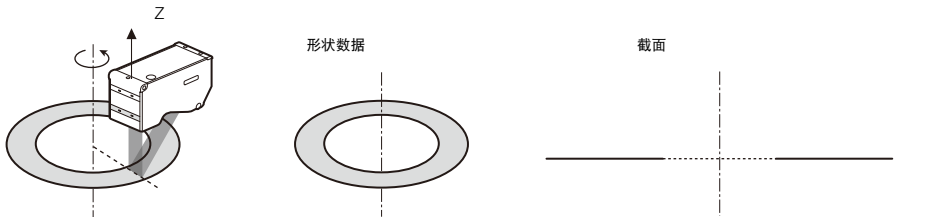
旋转节距。单位：度。
将顺时针视为正、将逆时针视为负。

fTiltCorrectDeg(in)

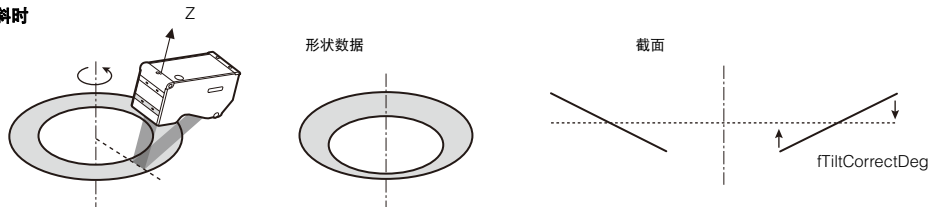
倾斜补正角度。单位：度。
将顺时针视为正、将逆时针视为负。

旋转轴不与 Z 轴平行时，原本应该为平坦形状的数据，现在变形为擂钵状以及圆锥状。
指定补正角度，便可补正为平坦的形状数据。
不采用旋转计算，而是减去 $\tan(fTiltCorrect)$ 的值，由此去除倾斜。

无倾斜时



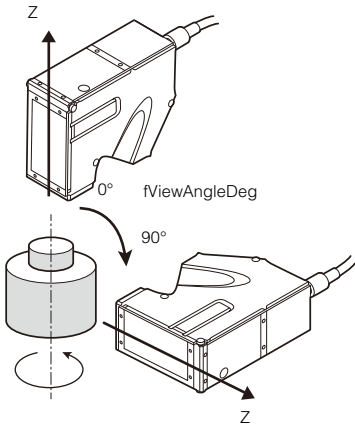
倾斜时



fViewAngleDeg(in)

视点角度。单位：度。
将顺时针视为正、将逆时针视为负。

在针对旋转轴，从工件侧面开始进行测量等情形下，使用这个项目。



* 因设置时的安装误差导致的轻微倾斜，请通过“fTiltCorrectDeg”执行补正。

fXPitchUm(in)

输入图像的 X 方向间隔。单位：μm。正值。

通过通信获取高度图像时，能够参照以下

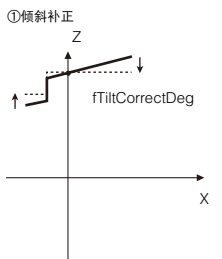
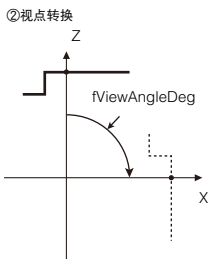
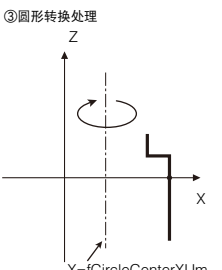
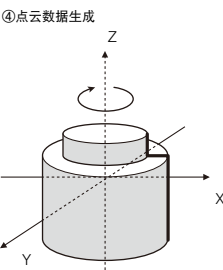
- 参照 LJX8IF_PROFILE_INFO 结构的 IXPitch 变量 (单位：0.01μm)
- 通过 LJX8IF_GetXpitch 指令获取 (单位：0.1μm)

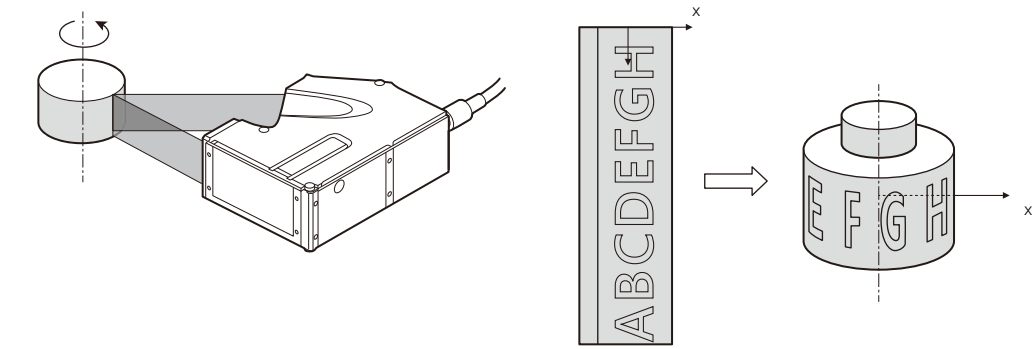
在离线状态下转换数据时，请参考下表。

在以下的情形下，初始值有可能会与值有所不同。

- 通过 LJX8IF_SetXpitch 指令变更了 X 节距时 (只限 LJ-X 传感头)
- 使用“间隔 (X 轴)”以及“Binning”功能变更了数据间隔时

型号	初始值 (μm)
LJ-X8020	2.5
LJ-X8060	5.0
LJ-X8080	12.5
LJ-X8200	25.0
LJ-X8400 (扩展功能 OFF)	75.0
LJ-X8400 (扩展功能 ON)	100.0
LJ-X8900	225.0
LJ-V7020(K)(B)	10.0
LJ-V7060(K)(B)	20.0
LJ-V7080(B)	50.0
LJ-V7200(B)	100.0
LJ-V7300(B)	300.0

参数	<p>fZUnit(in)</p> <p>输入图像的 Z 方向间隔 (换算系数)。单位：μm。正值。</p> <p>通过通信获取高度图像时，能够参照以下。</p> <ul style="list-style-type: none">通过 LJX8IF_GetZUnitSimpleArray 指令获取 (单位：0.01μm) <p>在离线状态下转换数据时</p> <p>LJ-X 传感头的换算系数</p> <table><tr><td>LJ-X8020</td><td>LJ-X8060</td><td>LJ-X8080</td><td>LJ-X8200</td><td>LJ-X8400</td><td>LJ-X8900</td></tr><tr><td>0.4</td><td>0.8</td><td>1.6</td><td>4</td><td>8</td><td>16</td></tr></table> <p>LJ-V 传感头的换算系数</p> <table><tr><td>LJ-V7020(K)(B)</td><td>LJ-V7060(K)(B)</td><td>LJ-V7080(B)</td><td>LJ-V7200(B)</td><td>LJ-V7300(B)</td></tr><tr><td>0.4</td><td>0.8</td><td>1.6</td><td>4</td><td>8</td></tr></table> <p>pSourceImage(in)</p> <p>输入图像。IXPointNum×IProfileNum 的 16bit 排列数据。</p> <p>使用通过 SimpleArray 格式获取到的高度数据。</p> <p>pfDestData(out)</p> <p>输出的点群数据。单位：μm。</p> <p>每个点 (X、Y、Z) 的三个值，是与 IXPointNum×IProfileNum 点数量相应的 FLOAT 型排列数据。</p> <p>按照 X[0]、Y[0]、Z[0]、X[1]、Y[1]、Z[1]、...、的顺序保存到内存中。</p> <p>(注意) 需要在函数调用侧确保内存。</p> <p>* 在以无效值 (属于输入图像，值为 0) 为基础转换的点 Z 中，保存 FLOAT 型的最小值 (-3.40282347E+38)。</p> <ul style="list-style-type: none">C#、VB.NET: float.MinValueC++: std::numeric_limits<float>::lowest() <p>* 在旋转轴下，获取到会转为 X=0.0、Y=0.0 的坐标原点。</p> <p>内部处理顺序</p> <p>按照以下顺序进行处理。</p> <p>圆形转换的旋转轴的指定，请注意通过视点转换后坐标指定的点。</p> <div><div><p>①倾斜修正</p></div><div><p>②视点转换</p></div><div><p>③圆形转换处理</p></div><div><p>④点云数据生成</p></div></div>	LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900	0.4	0.8	1.6	4	8	16	LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)	0.4	0.8	1.6	4	8
	LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900																	
	0.4	0.8	1.6	4	8	16																	
	LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)																		
	0.4	0.8	1.6	4	8																		
	<p>0X1006: 参数传输错误</p> <p>输入图像的 X 方向点数：指定了负值</p> <p>输入图像的轮廓数量：指定了负值</p> <p>输入图像的 X 方向间隔：指定了负值</p> <p>输入图像的 Z 方向间隔：指定了负值</p>																						

说明	<p>在对旋转工件进行扫描后获取到的长方形图像数据，能够转换到与实际形状相符的点群数据。 能够作为执行 3D 显示以及 3D 测量之际的预处理而使用。</p> 
备注	<p>支持 DLL 1.3.0.0 或更高版本。 使用本函数的示例应用程序（仅限执行文件）可通过基恩士用户支持页面下载。</p>

8.2.9 补充

8.2.9.1 通信设备

在“通信设备”中，确定与 PC 进行通信的控制器。

在 LJX8IF_EthernetOpen 中指定 IDeviceId 和 IP 地址。例如，与某个控制器 A 建立 Ethernet 通信时，IDeviceId=0、IP 地址 192.168.1.1，与某个控制器 B 建立 Ethernet 通信时，IDeviceId = 1、IP 地址 192.168.1.5。需要从控制器 A 获取轮廓数据时，如果将发送指令的 IDeviceId 设为 0，将从控制器 A 获取轮廓数据。如上所述，在发送指令时，用该指令指定 IDeviceId，即可选择向已连接的哪个控制器发送指令。

可实现同时通信的控制器台数上限为 6 台。（连接超过 6 台时，无法确认或保证动作。）

伴随通信的 I/F 中，可通过 IDeviceId 指定通信相应的控制器。可在 0 至 127 中指定 IDeviceId。

- 1 台控制器与多台 PC 通信时，最多可与 3 台 PC 进行 Ethernet 通信。要连接第 4 台 PC，需要将已连接的 3 台中最终通信日期时间距今最久的路径断开。

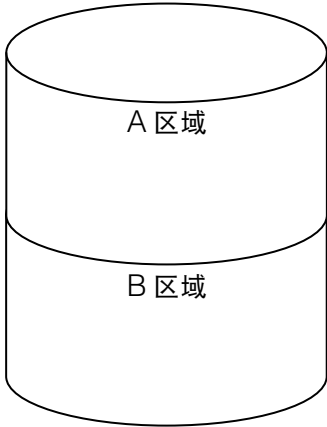
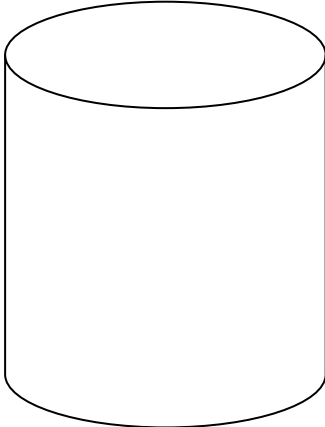
* 但是，1 台控制器只能与 1 台 PC 进行高速数据通信。

8.2.9.2 内存

根据内存分配设定及动作模式的不同，测量数据的可保存大小及保存时间不同。使用下列函数时，请加以注意。

- LJX8IF_GetProfile（获取轮廓）
- LJX8IF_GetBatchProfile（获取批处理轮廓）

各内存分配设定的内存使用区域如下所示。

双缓存	全部区域
将内存分为 A、B 的两个区域，每一次切换程序时，将会交替使用 A 区域、B 区域。 *1	使用内存的全部区域。
	

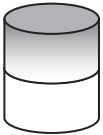
*1 活动程序使用中的内存区域被称为“活动区域”、未使用的区域被称为“非活动区域”。

内存分配设定为“双缓存”时

程序切换操作和各内存的使用状态迁移如下所示。可获取活动程序和之前 1 个程序中保存的数据。

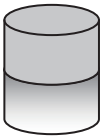
（*：活动区域）

程序编号	A 区域	B 区域
1（测量中）	保存中 *	无数据



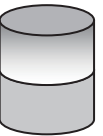
↓ 程序切换（保存对象区域为 B 区域）

程序编号	A 区域	B 区域
2（测量中）	程序编号 1 数据	保存中 *



↓ 程序切换（保存对象区域为 A 区域。删除 A 区域的数据）

程序编号	A 区域	B 区域
3（测量中）	保存中 *	程序编号 2 数据



内存的保存状态（白色：无数据 ⇔ 灰色：已保存） ↑

内存分配设定为“全部区域”时

执行程序切换操作时，将删除内存，但可以最大限度地使用内存区域。

在读取轮廓的操作中，指定 LJX8IF_GET_PROFILE_REQUEST 及

LJX8IF_GET_BATCH_PROFILE_REQUEST 的 byErase 为 1（删除读取的轮廓）并读取轮廓时，将从已读取的轮廓（批次）以及该轮廓（批次）之前的内存开始删除。此外，高速数据通信时，控制器会在发送获取轮廓后，从内存中删除已发送的轮廓。如需通过控制器连续获取轮廓，请确保 PC 读取轮廓的速度比控制器保存数据的保存速度更快。

8.2.9.3 设定的写入处理

在设定的写入处理中用到的函数有如下 4 种。

- LJX8IF_SetSetting（发送设定）
- LJX8IF_ReflectSetting（设定写入区域的生效请求）
- LJX8IF_RewriteTemporarySetting（更新设定写入区域）
- LJX8IF_CheckMemoryAccess（确认对保存区域的保存处理情况）

发送设定时，需要通过 LJX8IF_SetSetting 指定 Depth。Depth 的各选项及对应作用如下所示。

Depth	作用
LJX8IF_SETTING_DEPTH_WRITE (设定写入区域)	对动作不产生影响的设定区域。变更多项设定时，为了避免让暂时的设定不匹配导致控制器运行错误，可在本区域改写设定后，将设定从本区域反映到运行设定区域或保存区域，在不返回错误信息的情况下变更控制器的动作。
LJX8IF_SETTING_DEPTH_RUNNING (运行设定区域)	控制器运行中使用的设定值。在控制器启动时，被初始化为保存区域的设定值。 但是，该设定区域的设定值在关闭电源后就会丢失。 (重新启动时，将反映保存区域的设定)
LJX8IF_SETTING_DEPTH_SAVE (保存区域)	该区域中写入的设定在关闭电源后仍会保存在控制器中。但写入该区域时需要耗费时间。（可通过 LJX8IF_CheckMemoryAccess 确认是否正在写入该区域。请通过该函数确认写入完成后再关闭电源）

<使用示例①> … 集中变更多项设定时

1: 变更设定 LJX8IF_SetSetting (WRITE)
2: 变更设定 LJX8IF_SetSetting (WRITE)
⋮
最后: 变更设定 LJX8IF_SetSetting (WRITE)
更新设定写入区域 LJX8IF_ReflectSetting (RUNNING)

在写入 RUNNING 或 SAVE 时，会检查设定的匹配性，如果存在不匹配则会返回错误。（有关错误的详情，请参照“8.2.9.4 设定错误详情”。）因此，变更多项设定时，如果逐项将设定写入 RUNNING (SAVE)，设定项目引发不匹配，设定将不被反映到控制器。将多项设定写入 WRITE，在编写匹配的设定后，最后将集中反映到控制器。如需将不匹配的 WRITE 设定还原为控制器内的设定，请使用 LJX8IF_RewriteTemporarySetting。

注意

- 设定写入 RUNNING (SAVE) 时，测量将被中断。
- 在设定写入 SAVE 的过程中，请勿关闭控制器的电源。可通过 LJX8IF_CheckMemoryAccess 确认设定是否已写入。
- 将最后的 LJX8IF_SetSetting (WRITE) 作为 LJX8IF_SetSetting (RUNNING)，也会获得同样的结果。（无需更新设定写入区域。）

<使用示例②> … 只变更 1 项设定时

- 不将设定保存到控制器中时
变更设定 LJX8IF_SetSetting (RUNNING)
- 将设定保存到控制器中时
变更设定 LJX8IF_SetSetting (SAVE)

注意

- 设定写入 RUNNING (SAVE) 时，测量将被中断。
- 在设定写入 SAVE 的过程中，请勿关闭控制器的电源。可通过 LJX8IF_CheckMemoryAccess 确认设定是否已写入。

8.2.9.4 设定错误详情

各类设定中，存在必须满足的匹配性要求。以不满足匹配性要求的设定发送、设定写入区域的反映请求指令所反馈的设定错误详情 (dwError) 如下所示。

dwError 的值	错误内容
0x1X000000 (*1)	采样周期超出可设定范围
0x1X000002 (*1)	批处理点数超出可设定范围
0x1X010000 (*1)	光量控制上限值 < 下限值
0x1X010001 (*1)	测量范围 Z 方向超出可设定范围
0x1X010002 (*1)	测量范围中心 X 超出可设定范围
0x1X010003 (*1)	测量范围中心 Z 超出可设定范围

*1: X 代表程序编号，存储 0 至 F

9 通用返回代码

9.1 通信库反馈的返回代码

下列返回代码，将在通信库内部被判断并返回。

具体来说，返回代码会在与控制器的通信失败，或因通信库内部的状态依存导致未执行处理时返回。

定义名称	数据 (下位 2BYTE)	原因
LJX8IF_RC_OK	0x0000	正常结束
LJX8IF_RC_ERR_OPEN	0x1000	开启通信路径失败。
LJX8IF_RC_ERR_NOT_OPEN	0x1001	未成功建立通信路径。
LJX8IF_RC_ERR_SEND	0x1002	发送指令失败。
LJX8IF_RC_ERR_RECEIVE	0x1003	接收反馈失败。
LJX8IF_RC_ERR_TIMEOUT	0x1004	接收反馈超时。
LJX8IF_RC_ERR_NOMEMORY	0x1005	保护内存失败。
LJX8IF_RC_ERR_PARAMETER	0x1006	参数传输错误。
LJX8IF_RC_ERR_RECV_FMT	0x1007	非法的反馈数据。
LJX8IF_RC_ERR_HISPEED_NO_DEVICE	0x1009	高速通信初始化失败。
LJX8IF_RC_ERR_HISPEED_OPEN_YET	0x100A	高速通信初始化完毕。
LJX8IF_RC_ERR_HISPEED_RECV_YET	0x100B	已发生通信错误（高速通信用）
LJX8IF_RC_ERR_BUFFER_SHORT	0x100C	引数传输的缓存大小不足。

9.2 控制器反馈的返回代码

下列返回代码，将在与控制器通信成功的基础上，因控制器无法处理等原因而返回。

数据 (下位 2BYTE)	原因
0x8031	未定义指令错误（发送到不支持指令的控制器时等）
0x8041	状态错误（发生系统错误时等）
0x8042	参数错误（参数设定错误时等）

10 示例程序

本通信库附带以下示例程序。

- Sample.....具有 GUI 的示例应用程序。可以确认各函数的详细使用方法。
- Sample.....简洁的控制台应用程序，专为图像获取而打造。

10.1 示例程序

[存储在 C:\Program Files\KEYENCE\LJ-X Navigator\lib\Sample 中。

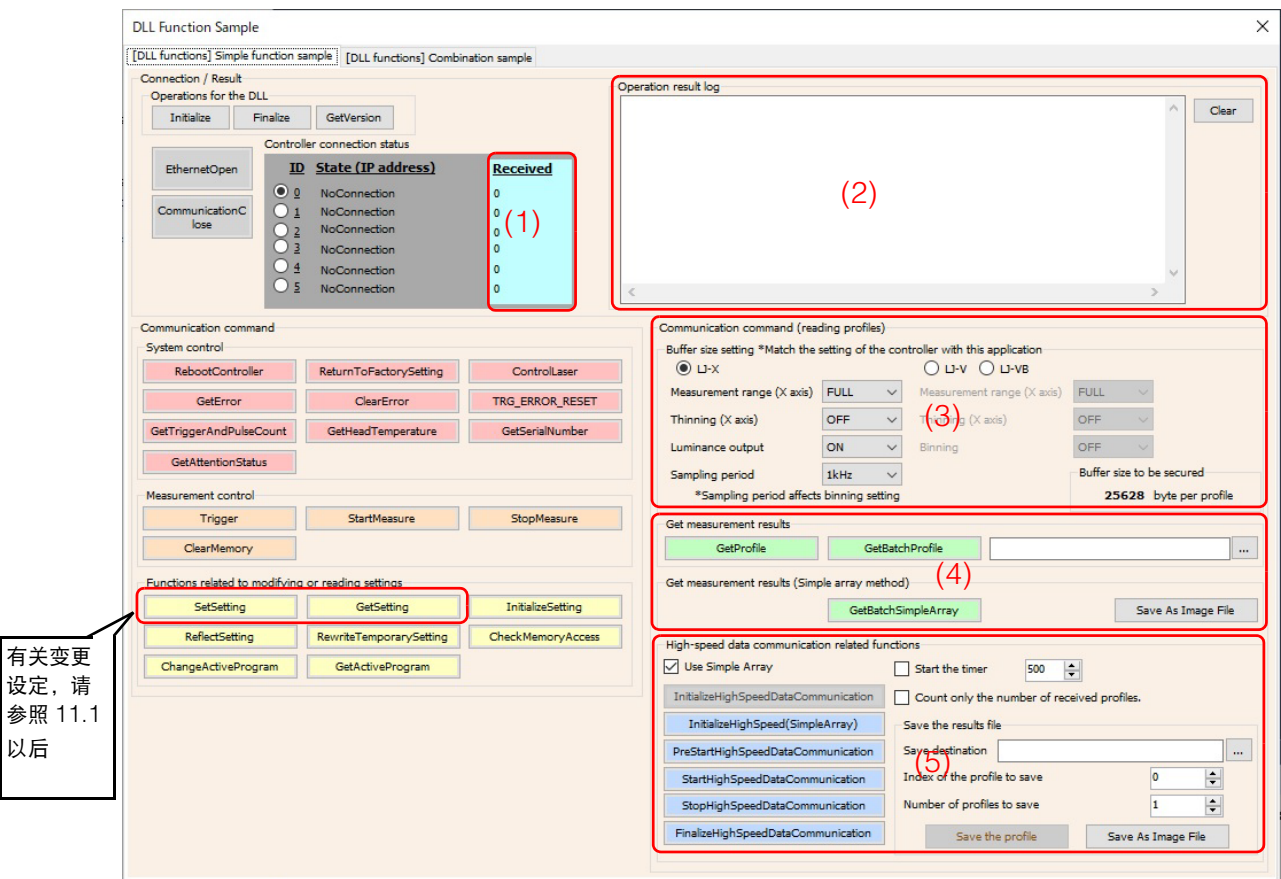
下面对作为使用通信库的应用程序创建示例而附带的示例程序进行说明。

与 C#、VB.NET、C++ 基本相同。下面以 C# 为例进行说明。

* C#、VB.NET、C++ 由 Visual Studio 2015 Update3 创建。

10.1.1 用户界面规格

[DLL functions] Simple function sample 标签



各按钮均记载有函数名称。点击该按钮，即执行相应函数。

(1) 显示通过高速数据通信接收的轮廓数。最多可显示 6 台控制器。

* 非高速数据通信的常规轮廓接收将不被计入。

(2) 显示已执行的指令及其结果。若有错误则显示错误代码。有关错误代码的详情，请参照各函数的返回值（参照“8.2 函数参考”）或“9 通用返回代码”的返回代码一览。

(3) 在准备通过该示例程序获取轮廓时，用于决定排列大小。

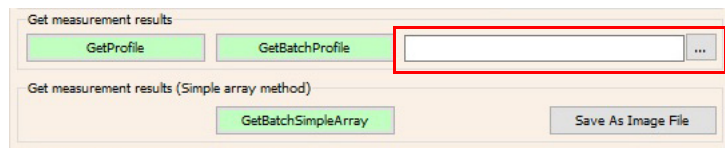
用于 GetProfile、GetBatchProfile、GetBatchSimpleArray。请确保传感头种类及各参数符合 LJ-X8000A 本体的设定。用于准备的排列大小如果过小，将无法正确读取轮廓。此外，接收轮廓数据所需的缓冲大小显示在右下方。

- (4) 可保存使用 GetProfile、GetBatchProfile、GetBatchSimpleArray 获取的轮廓数据（高度数据和亮度数据）。这些指令不适合高速连续地输出轮廓，但与高速数据通信相比，可以更简单地获取轮廓数据。1 次指令发送可获取的轮廓数有所限制。（轮廓数据点数为 3200 点且有亮度时，最多 128 个轮廓，其他情况则为最多 255 个轮廓。）在指定文件名中以 _height 命名并保存高度数据。在指定文件名中以 _luminance 命名并保存亮度数据。

GetProfile、GetBatchProfile 事先指定以下文件夹及文件，即可以 CSV 格式保存轮廓数据。

对于 GetBatchSimpleArray，执行“GetBatchSimpleArray”后点击“Save As Image”指定保存位置 / 文件名，即可以 Bitmap 格式及 Tiff 格式保存轮廓数据。有关将图像内 Pixel 数据转换为高度 [μm] 的方法，请参照第 8 章的 GetBatchSimpleArray 指令说明。

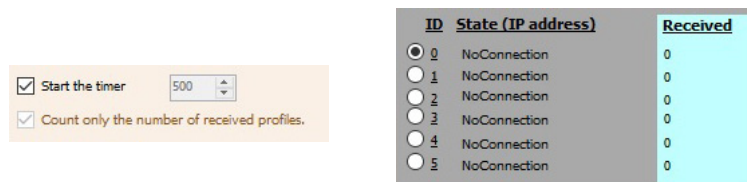
GetBatchProfile、GetBatchSimpleArray 在批处理测量为 ON 时有效。



- (5) 用于进行高速数据通信的动作确认。
* 请在开始高速数据通信前进行设定。

<需要确认通信速度时>

请勾选“Start the timer”和“Count only the number of profiles.”，开始高速数据通信。每间隔右侧数值（单位 ms：初始值 500 ms）的周期，就对示例程序所接收的轮廓进行检查，将获取的轮廓数更新到①。（Callback 函数中，将只对被调用的次数进行计数动作。）



如果该获取数与预计的速度不符，可能是如下原因。

- 使用编码器触发时，如果采样周期比编码器的触发输入周期大，将无法确认编码器的所有输入。→请变更设定，确保采样周期比编码器输入周期短。
- 可能是网络环境、电缆的类别 / 设备构成等因素。

如果该示例程序的程序获取数已达到预计的速度，客户编写的程序中的获取数与预计不符，则可能是程序有问题。可能是由于调用 Callback 函数后的处理负荷过重。请参照“7.4 返回函数 I/F 定义”。

<需要保存通过高速数据通信获取的轮廓时>

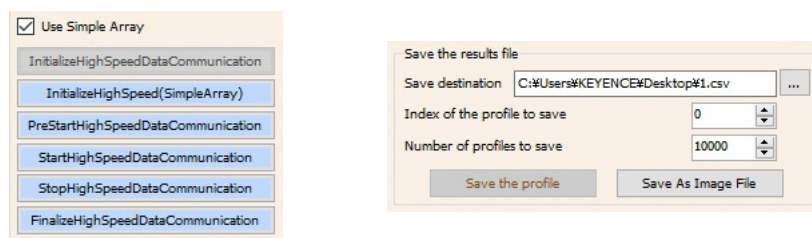
请仅勾选“Start the timer”，开始高速数据通信。每间隔右侧数值（单位 ms：初始值 500 ms）的周期，就对示例程序所接收的轮廓进行检查，将获取的轮廓数更新到①。（将轮廓存储到 PC 中。）



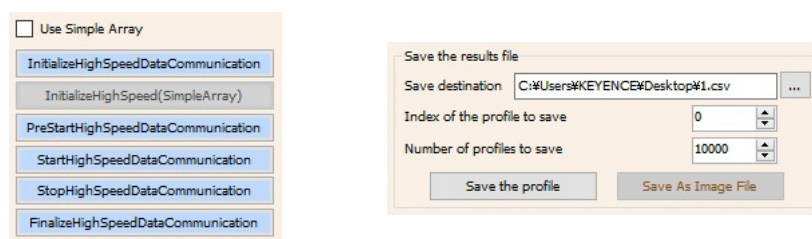
对“Save Destination”指定保存轮廓数据的位置。还可通过“Index of the profile to save”指定获取已存储轮廓中的第几项轮廓。（0 至 60000）

此外，可通过“Number of profiles to save”指定要从中保存几个轮廓数据。（1 至 60000）

需要以 Bitmap 格式保存通过高速数据通信获取的轮廓时，请勾选“Use Simple Array”并使用以下 InitializeHighSpeed(SimpleArray) 开始高速数据通信。点击“Save As Image File”，将在指定的文件夹中保存 Bitmap 及 Tiff 文件。



需要以 CSV 格式保存轮廓数据时，请取消勾选“Use Simple Array”并使用以下 InitializeHighSpeedDataCommunication 开始高速数据通信。点击“Save the profile”，将在指定的文件夹中保存 CSV 文件。

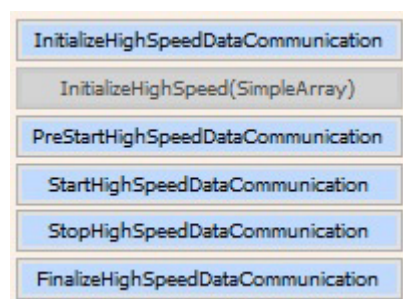


[DLL functions] Combination sample 标签

左上角的 Communication establishment（finalization）是初始化 DLL + 建立通信路径（断开通信路径）的组合示例。

如果是右上角的 Get profiles、Get batch profiles，将用 Save path 中指定的文件夹和文件名，自动以 txt 格式保存 10 个轮廓。需要任意指定获取数时，请参照 Simple function sample 标签的“GetProfile / GetBatchProfiles”指令。

右下角 High-speed data communication 的“Start”、“Finalize”是下列函数的组合示例。



左下角的 Sending or receiving settings for each program number 是“11.2 批处理发送 / 接收”的示例。可将环境设定、测量通用设定、各程序的设定保存到指定文件中，或从指定文件进行发送。

10.1.2 文件保存格式

- CSV 格式

在指定文件名中添加 _height 并保存的数据即为轮廓数据（高度数据）。以 0.01 μm 为单位存储。

例：为 1.98750 mm 时，将存储为 198750。

在指定文件名中添加 _luminance 并保存的数据即为亮度数据。存储 0 至 1023 范围内的值。

- Bitmap 格式

16bit rgb565^{*1}

- Tiff 格式

16 位无压缩灰阶^{*1}

^{*1} 如需将存储的值（0 至 65535）转换为高度数据（μm），请使用以下公式。

高度（μm）=（存储的值 - 32768）* 换算系数

LJ-X 传感头的换算系数

LJ-X8020	LJ-X8060	LJ-X8080	LJ-X8200	LJ-X8400	LJ-X8900
0.4	0.8	1.6	4	8	16

LJ-V 传感头的换算系数

LJ-V7020(K)(B)	LJ-V7060(K)(B)	LJ-V7080(B)	LJ-V7200(B)	LJ-V7300(B)
0.4	0.8	1.6	4	8

例：使用 LJ-X8080 传感头、存储的值为 34232 时，

$(34232 - 32768) * 1.6 = 2342.4 \text{ [}\mu\text{m]}$

使用 LJ-X8080 传感头、存储的值为 31575 时，

$(31575 - 32768) * 1.6 = -1908.8 \text{ [}\mu\text{m]}$

无效数据、死角数据和判断待机数据存储 0。

10.2 图像获取示例

存储在 C:\Program Files\KEYENCE\LJ-X Navigator\lib\Sample\Sample_ImageAcquisition 中。
进行高速通信，从控制器获取高度 / 亮度图像并将图像保存为 CSV/TIFF 格式。

以下源代码记述了程序的主例程。

- C++..... main.cpp
- C# Program.cs
- VB.NET..... Module1.vb

* C++、C#、VB.NET 由 Visual Studio 2015 Update3 创建。

可以通过改写以下注释包围的范围来变更连接目标控制器等。

(摘自 C++ main.cpp)

```
//-----  
// CHANGE THIS BLOCK TO MATCH YOUR SENSOR SETTINGS (FROM HERE)  
//-----  
int deviceId = 0;           // Set "0" if you use only 1 head.  
int xImageSize = 3200;      // Number of X points.  
int yImageSize = 1000;      // Number of Y lines.  
float y_pitch_um = 20.0;    // Data pitch of Y data. (e.g. your encoder setting)  
int timeout_ms = 5000;      // Timeout value for the acquiring image (in milisecond).  
int use_external_batchStart = 0; // Set "1" if you controll the batch start timing externally. (e.g. terminal input)  
string outputFilePath1 = "sample_height.csv";  
string outputFilePath2 = "sample_height.tif";  
string outputFilePath3 = "sample_luminance.tif";  
  
LJX8IF_ETHERNET_CONFIG EthernetConfig =  
{  
    { 192, 168, 0, 1 },      // IP address  
    24691                    // Port number  
};  
  
int HighSpeedPortNo = 24692; // Port number for high-speed communication  
//-----  
// CHANGE THIS BLOCK TO MATCH YOUR SENSOR SETTINGS (TO HERE)  
//-----
```

- CSV 格式

以 mm 为单位存储高度数据。

- Tiff 格式

分别存储高度数据、亮度数据。文件格式如 10.1.2 文件保存格式中所述。

11 附录

11.1 发送 / 接收设定

LJ-X8000 系列可通过发送设定（LJX8IF_SetSetting）、接收设定（LJX8IF_GetSetting），分别对各项目发送 / 接收设定。（各项目指的是环境设定 / 测量通用设定 / 程序 0 至 15 的各项设定参数）如需批处理进行环境设定、测量通用设定或各程序设定的发送 / 接收，而非各设定参数的设定发送 / 接收，请参照“11.2 批处理发送 / 接收”。

下面就输入 / 输出到设定发送 / 接收指令的 Target Setting、pData 进行说明。（有关 byDepth，请参照“8.2.9.3 设定的写入处理”）

Target Setting：指定发送 / 接收设定的项目。成员如下所示。各成员的参数详情，请参照下一页的「发送 / 接收设定项目详情」。

Type	指定发送 / 接收环境设定（01h）、测量通用设定（02h）、程序 0 至程序 15（10h 至 1Fh）中的哪一项设定。
Category	如需发送 / 接收程序 0 至 15 的设定，请像触发设定、拍摄设定一样，指定发送 / 接收哪一项设定。发送 / 接收环境设定、测量通用设定时，指定为 0。
Item	指定发送 / 接收 Category 指定项目中的哪一项设定。
Target1	根据发送 / 接收的设定，可能需要进行相应的指定。无需设定时，指定为 0。
Target2	
Target3	
Target4	

pData：指定发送 / 接收设定的设定数据。详情请参照“11.3 发送 / 接收设定项目详情”。

<补充：通过示例软件使用 Set setting 指令的示例>
使用 LJX8IF_SetSetting 指令，将程序编号 3 的轮廓数据向 Z 方向移动 +1 mm。
（介绍将移位（Z）值设为 +1.0 mm 的方法。）

如 P.51 所示，指定参数如下所示。
且此时指定的 pData 为 4 byte。

< 移位（X） / 移位（Z） >
Type: 10h 至 1Fh（10h: 程序编号 0、11h: 程序编号 1、…、1F: 程序编号 15）
Category: 02h、Item: 08h（移位 X）、09h（移位 Z）
Target1 至 4: 00h

byte	pData
0	移位量：测量范围内的任意数值（单位 0.01 μm、有符号的 32 bit 整数 例：1 mm=100000、2 mm = 200000）
1	
2	
3	

Settings

For details on the items that follow the category, see the tables in the communication command specifications.

Ox 2

Get target area and setting depth

0: Write settings area, 1: Running settings area, 2: Save area

Ox 13

Setting type

0x01: Environment settings, 0x02: Common measurement settings, 0x10: Program 0, 0x11: Program 1, ..., 0x1F: Program 15

Ox 2

Category

Ox 9

Setting item

Ox 0

Setting target 1

Ox 0

Setting target 2

Ox 0

Setting target 3

Ox 0

Setting target 4

Amount of data

4

BYTE

Writing parameters (comma-separated hexadecimal values)

A0,86,01,00

OK

Cancel

需要指定 +1.0 mm 时，必须以 0.01 μm 为单位输入数值，请输入数值 1.0 mm= 100000。转换到 16 进制，即为 00 01 86 A0。示例程序输入需要有小尾数及逗号分隔，因此应输入 A0,86,01,00。

有关最上位参数的 0: Write settings area、1: Running settings area、2: Save area, 请参照“8.2.9.3 设定的写入处理”。

有关最上位参数的 0: Write settings area、1: Running settings area、2: Save area, 请参照“8.2.9.3 设定的写入处理”。

11.2 批处理发送 / 接收

上述 Category 指定为 FFh, 即可批处理发送 / 接收环境设定 / 测量通用设定 / 各程序的设定。

例 1: 批处理发送 / 接收测量通用设定数据

Type: 02h、Category: FFh、Item: 00h、Target1 至 4: 00h

例 2: 批处理发送 / 接收程序编号 5 的设定

Type: 15h、Category: FFh、Item: 00h、Target1 至 4: 00h

详情请参照示例程序。

* 批处理发送 / 接收环境设定时为 60 byte、测量通用设定为 20 byte、程序为 10980 byte。

11.3 发送 / 接收设定项目详情

■ 变更环境设定时

< 设备名称 >

Type: 01h、Category: 00h、Item: 00h
Target1 至 4: 00h

byte	pData
0	设备名称、第 1 byte
1	设备名称、第 2 byte
2	设备名称、第 3 byte
:	:
31	设备名称、第 32 byte

* 最多 32 个字符。末尾不带 0。

* SHIFT-JIS

< 下次接通电源时动作 >

Type: 01h、Category: 00h、Item: 01h
Target1 至 4: 00h

byte	pData
0	下次接通电源时动作 0: BOOTP → IP 地址固定 1: IP 地址固定、 2: BOOTP
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 高速通信带宽限制 >

Type: 01h、Category: 00h、Item: 02h
Target1 至 4: 00h

byte	pData
0	高速通信带宽限制 0: OFF、 1: 500 Mbps、 2: 200 Mbps、 3: 100 Mbps
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 高速通信时 MTU >

Type: 01h、Category: 00h、Item: 03h
Target1 至 4: 00h

byte	pData
0	MTU 设定: 1500 至 9216
1	
2	预约 (固定为 0)
3	预约 (固定为 0)

< IP 地址 / 子网掩码 / 默认网关 >

Type: 01h、Category: 00h
Item: 04h (IP 地址) / 05h (子网掩码)
/ 06h (默认网关)
Target1 至 4: 00h

byte	pData
0	第 1 byte
1	第 2 byte
2	第 3 byte
3	第 4 byte

* 下列 IP 地址将被视作不正确的 IP 地址。

0.0.0.0 / 224.0.0.0 至 255.255.255.255

* 下列地址将被视作不正确的子网掩码。

0.0.0.0 / 255.255.255.255 / 从首位起, 位“1”不连续
(例: 255.255.255.64 =
11111111.11111111.11111111.01000000 为错误)

* 下列地址将被视作不正确的默认网关。

224.0.0.0 至 255.255.255.255

< TCP 指令端口编号 / TCP 高速端口编号 >

Type: 01h、Category: 00h
Item: 07h (TCP 指令端口编号) /
08h (TCP 高速端口编号)
Target1 至 4: 00h

byte	pData
0	端口编号 (1 至 65535)
1	
2	预约 (固定为 0)
3	预约 (固定为 0)

请勿设定同样的指令端口编号和高速端口编号。

■ 变更通用设定时

< 内存分配 >

Type: 02h、Category: 00h、Item: 01h

Target1 至 4: 00h

byte	pData
0	内存分配设定: 0: 双缓存、 1: 全部区域
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 内存 FULL 时的动作 >

Type: 02h、Category: 00h、Item: 02h

Target1 至 4: 00h

byte	pData
0	内存 FULL 时的动作: 0: 覆盖、 1: 停止
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 并列拍摄 >

* 使用 LJ-X 传感头时无效。

Type: 02h、Category: 00h、Item: 03h

Target1 至 4: 00h

byte	pData
0	并列拍摄: 0: 无效、1: 有效
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< TRG 最短输入时间 >

Type: 02h、Category: 00h、Item: 06h

Target1 至 4: 00h

byte	pData
0	TRG 输入端子时间常数: 0: 7 μ s、1: 10 μ s、2: 20 μ s、 3: 50 μ s、4: 100 μ s、5: 200 μ s、 6: 500 μ s、7: 1 ms
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< ENCODER 最短输入时间 >

Type: 02h、Category: 00h、Item: 07h

Target1 至 4: 00h

byte	pData
0	ENCODER 输入端子时间常数: 0: 120 ns、1: 150 ns、2: 250 ns、 3: 500 ns、4: 1 μ s、5: 2 μ s、 6: 5 μ s、7: 10 μ s、8: 20 μ s
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 控制端子最短输入时间 >

Type: 02h、Category: 00h、Item: 08h

Target1 至 4: 00h

byte	pData
0	控制端子最短输入时间: 0: 250 μ s、 1: 1 ms
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 亮度输出 >

* LJ-V 传感头无法进行亮度输出。

LJ-VB 传感头的亮度输出始终有效。

Type: 02h、Category: 00h、Item: 0Bh

Target1 至 4: 00h

byte	pData
0	亮度输出: 0: 仅高度数据、 1: 高度 + 亮度数据
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

■ < 变更各程序内的设定时 >

● 触发设定

< 触发模式 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 01h、

Target1 至 4: 00h

byte	pData
0	触发模式: 0: 连续触发、 1: 外部触发、 2: 编码器触发
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 采样周期 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 02h、

Target1 至 4: 00h

byte	pData
0	采样周期 (LJ-X): 0: 10 Hz、1: 20 Hz、2: 50 Hz、 3: 100 Hz、4: 200 Hz、5: 500 Hz、 6: 1 KHz、7: 2 KHz、8: 4 KHz、 9: 8 KHz、10: 16 KHz 采样周期 (LJ-V): 0: 10 Hz、1: 20 Hz、2: 50 Hz、 3: 100 Hz、4: 200 Hz、5: 500 Hz、 6: 1 KHz、7: 2 KHz、8: 4 KHz、 9: 4.13 KHz、10: 8 KHz、11: 16 KHz、 12: 32 KHz、13: 64 KHz 采样周期 (LJ-V***B): 0: 10 Hz、1: 20 Hz、2: 50 Hz、 3: 100 Hz、4: 200 Hz、5: 500 Hz、 6: 1 KHz、7: 2 KHz、8: 4 KHz、 9: 4.13 KHz、10: 8 KHz
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 批处理测量 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 03h、

Target1 至 4: 00h

byte	pData
0	批处理测量: 0: 批处理 OFF、 1: 批处理 ON
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 批处理点数 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 0Ah、

Target1 至 4: 00h

byte	pData
0	批处理点数: 50 至 60000
1	
2	预约 (固定为 0)
3	预约 (固定为 0)

< 触发间隔 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 04h、

Target1 至 4: 00h

byte	pData
0	触发间隔: 0: 间隔 OFF、 1: 间隔 ON
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 触发间隔 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 05h、

Target1 至 4: 00h

byte	pData
0 至 3	间隔数: 1 至 500000 (单位 0.0001 mm、 0.0001 至 50.0000 mm)

< 多台控制器同步 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 0Bh、

Target1 至 4: 00h

byte	pData
0	多台控制器同步: 0: 同步 OFF、 1: 同步主站、 2: 同步从站
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 防止相互干扰 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 06h、

Target1 至 4: 00h

byte	pData
0	防止相互干扰: 0: 相互干扰 OFF、 1: 相互干扰 AB-ON、 2: 相互干扰 ABC-ON*
1	从站发光组: 0: 与主站同时发光 (A)、 1: 与主站交替发光 (B)、 2: 与主机交替发光 (C) *
2	预约 (固定为 0)
3	预约 (固定为 0)

* 支持控制器 1.02.00 或更高版本。

< 输入模式 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 07h、

Target1 至 4: 00h

byte	pData
0	编码器触发输入模式: 0: 1 相 1 倍增 (无方向)、 1: 2 相 1 倍增、 2: 2 相 2 倍增、 3: 2 相 4 倍增
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 间隔 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 08h、

Target1 至 4: 00h

byte	pData
0	编码器触发间隔: 0: 间隔 OFF、 1: 间隔 ON
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 间隔点数 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 00h、Item: 09h、

Target1 至 4: 00h

byte	pData
0	编码器触发间隔点数: 2 至 1000
1	
2	预约 (固定为 0)
3	预约 (固定为 0)

● 拍摄设定

<Binning>

* 使用 LJ-X 传感头时无效。

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 01h、

Target1 至 4: 00h

byte	pData
0	Binning: 0: Binning OFF、 1: Binning ON
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 测量范围 X 方向 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 02h、

Target1 至 4: 00h

byte	pData
0	测量范围 X 方向: LJ-X 0: FULL、1: (3/4)、2: (1/2)、 3: (1/4) LJ-V 0: FULL、1: MIDDLE、 2: SMALL
1	预约 (固定为 0)
2	测量范围中心 X: LJ-X: 0 至 1000
3	(以 0.1% 为单位、0.0 至 100.0%) LJ-V: 预约 (固定为 0)

< 测量范围 Z 方向 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 03h、

Target1 至 4: 00h

byte	pData
0	测量范围 Z 方向: LJ-X 0: FULL、1: (1/2)、2: (1/4)、 3: (1/8)、4: (1/16)、5: (1/32) LJ-V 0: FULL、1: MIDDLE、 2: SMALL
1	预约 (固定为 0)
2	测量范围中心 Z: LJ-X: 0 至 1000
3	(以 0.1% 为单位、0.0 至 100.0%) LJ-V: 预约 (固定为 0)
4	0: Z 范围扩展 OFF、1: Z 范围扩展 ON
5	Binning (Z) * ON 时请指定 0, OFF 时请指定 1。(ON/ OFF 逻辑颠倒, 敬请注意) * 仅在使用 LJ-X8080/8200 传感头时有效的 设定。
6	预约 (固定为 0)
7	预约 (固定为 0)

< 动态量程 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 05h、

Target1 至 4: 00h

byte	pData
0	动态量程 (LJ-X): 1 至 9 动态量程 (LJ-V): 0: 高精度、1: 高动态量程 1、 2: 高动态量程 2、3: 高动态量程 3
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 曝光时间 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 06h、

Target1 至 4: 00h

byte	pData
0	曝光时间 (LJ-X): 0: 15 μ s、1: 30 μ s、2: 60 μ s、 3: 120 μ s、4: 240 μ s、5: 480 μ s、 6: 960 μ s、7: 1700 μ s、8: 4.8 ms、 9: 9.6ms 曝光时间 (LJ-V): 0: 15 μ s、1: 30 μ s、2: 60 μ s、 3: 120 μ s、4: 240 μ s、5: 480 μ s、 6: 960 μ s、7: 1920 μ s、8: 5 ms、 9: 10 ms
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 曝光模式 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 07h、

Target1 至 4: 00h

byte	pData
0	曝光模式: 0: 标准、 1: 多重发光 (合成)、 2: 多重发光 (光量最佳化)
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 多重发光 (光量最佳化) 详情 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 08h、

Target1 至 4: 00h

byte	pData
0	发光次数: 0: 2 次、 1: 4 次
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 多重发光 (合成) 详情 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 09h、

Target1 至 4: 00h

byte	pData
0	发光次数: 0: 3 次、 1: 5 次
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 屏蔽设定 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 0Ah、

Target1: 00h (上侧屏蔽 1)、

01h (上侧屏蔽 2)、02h (上侧屏蔽 3)、

03h (下侧屏蔽 1)、04h (下侧屏蔽 2)、

05h (下侧屏蔽 3)、Target2 至 4: 00h

byte	pData
0	有效 / 无效: 0: 无效、 1: 矩形、 2: 三角形
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)
4	X 坐标 1: 0 至 1000
5	(以 0.1% 为单位、0.0 至 100.0%)
6	Z 坐标 1: 0 至 1000
7	(以 0.1% 为单位、0.0 至 100.0%)
8	X 坐标 2: 0 至 1000
9	(以 0.1% 为单位、0.0 至 100.0%)
10	Z 坐标 2: 0 至 1000
11	(以 0.1% 为单位、0.0 至 100.0%)
12	X 坐标 3: 0 至 1000
13	(以 0.1% 为单位、0.0 至 100.0%) * 矩形时无需注意
14	Z 坐标 3: 0 至 1000
15	(以 0.1% 为单位、0.0 至 100.0%) * 矩形时无需注意

< 光量控制 控制模式 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 0Bh

Target1 至 4: 00h

byte	pData
0	控制模式: 0: AUTO、1: MANUAL
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 光量控制 上限值 / 下限值 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、

Item: 0Ch (上限值)、0Dh (下限值)

Target1 至 4: 00h

byte	pData
0	上限值: 1 至 99 下限值: 1 至 99
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 光量控制 对象区域 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 0Eh、

Target1 至 4: 00h

byte	pData
0	光量控制对象区域起始: 0 至 1000 (以 0.1% 为单位、0.0 至 100.0 %)
1	
2	光量控制对象区域结束: 0 至 1000 (以 0.1% 为单位、0.0 至 100.0 %)
3	

< 峰值检测灵敏度 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 0Fh、

Target1 至 4: 00h

byte	pData
0	峰值检测等级: 1 至 5
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 无效数据插补点数 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 10h、

Target1 至 4: 00h

byte	pData
0	无效数据插补点数: 0 至 255
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 峰值选择 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 01h、Item: 11h、

Target1 至 4: 00h

byte	pData
0	LJ-X: 0: 标准、1: NEAR、2: FAR、 3 至 5: 作为无效数据 LJ-V: 0: 标准、1: NEAR、2: FAR、 3: 消除 X 多重反射、 4: 消除 Y 多重反射、 5: 作为无效数据
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 峰值宽度过滤器 >

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 01h、Item: 12h、

Target1 至 4: 00h

byte	pData
0	峰值宽度过滤器: 0: 过滤器 OFF、 1: 过滤器 ON
1	LJ-X: 过滤器强度: 1 (弱) 至 5 (强) LJ-V: 预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 干扰光抑制功能 >

* 使用 LJ-V 传感头时无效。

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 01h、Item: 14h、

Target1 至 4: 00h

byte	pData
0	干扰光抑制功能: 0: 过滤器 OFF、 1: 过滤器 ON
1	过滤器强度 1 (弱) 至 3 (强)
2	预约 (固定为 0)
3	预约 (固定为 0)

● 轮廓

< 间隔 (X 轴) >

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 02h、Item: 02h、

Target1 至 4: 00h

byte	pData
0	间隔 (X 轴): 0: 间隔 OFF、 1: 1/2、2: 1/4
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 死角数据插补 >

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 02h、Item: 05h、

Target1 至 4: 00h

byte	pData
0	死角数据插补方法: 0: 不进行插补、 1: 直线插补、 2: 垂直插补
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 反转 (X) / 反转 (Z) >

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 02h、

Item: 06h (反转 X)、07h (反转 Z)

Target1 至 4: 00h

byte	pData
0	反转: 0: 反转 OFF、1: 反转 ON
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 移位 (X) / 移位 (Z) >

Type: 10h 至 1Fh

(10h: 程序编号 0、
11h: 程序编号 1、…、
1F: 程序编号 15)

Category: 02h、

Item: 08h (移位 X)、09h (移位 Z)

Target1 至 4: 00h

byte	pData
0	移位量: 测量范围内的任意数值 (单位 0.01 μ m、有符号的 32 bit 整数 例: 1 mm=100000、2 mm = 200000)
1	
2	
3	

< 中位数 (X 轴) / 中位数 (时间轴) >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、

Item: 0Ah (中位数 (X 轴))

0Ch (中位数 (时间轴))、

Target1 至 4: 00h

byte	pData
0	中位数点数 (X 轴): 0: OFF、1: 3 点、2: 5 点、3: 7 点、 4: 9 点、5: 17 点、6: 25 点、7: 33 点 中位数点数 (时间轴): 0: OFF、1: 3 点、2: 5 点、3: 7 点、 4: 9 点
1	预约 (固定为 0)
2	预约 (固定为 0)
3	预约 (固定为 0)

< 平滑 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、Item: 0Bh、

Target1 至 4: 00h

byte	pData
0	平滑次数: 0: 1 次、1: 2 次、2: 4 次、3: 8 次、 4: 16 次、5: 32 次、6: 64 次
1	形状保持: 0: OFF、1: ON
2	预约 (固定为 0)
3	预约 (固定为 0)

< 平均 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、Item: 0Dh、

Target1 至 4: 00h

byte	pData
0	平均次数: 0: 1 回、1: 2 回、2: 4 回、3: 8 回、 4: 16 回、5: 32 回、6: 64 回、 7: 128 回、8: 256 回
1	预约 (0 固定)
2	预约 (0 固定)
3	预约 (0 固定)

< 无效数据处理 (时间轴) >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、Item: 0Eh、

Target1 至 4: 00h

byte	pData
0	处理次数: 0 至 255
1	恢复次数: 0 至 255
2	预约 (固定为 0)
3	预约 (固定为 0)

< 倾斜补正 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、Item: 0Fh

Target1 至 4: 00h

byte	pData
0	ON/OFF: 0: 补正无效、1: 补正有效
1 至 3	预约 (固定为 0)
4	直线计算区域数: 0: 区域 2 无效、1: 区域 2 有效
5 至 7	预约 (固定为 0)
8	区域起始位置 1: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
9	
10	
11	
12	区域结束位置 1: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
13	
14	
15	
16	区域起始位置 2: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
17	
18	
19	
20	区域起始位置 2: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
21	
22	
23	
24	补正后角度 (- 45.00 至 +45.00deg): - 4500 至 + 4500
25	
26	补正角度 (- 45.00 至 +45.00deg): - 4500 至 + 4500
27	

< 高度补正 >

Type: 10h 至 1Fh

(10h: 程序编号 0、

11h: 程序编号 1、…、

1F: 程序编号 15)

Category: 02h、Item: 10h、

Target1 至 4: 00h

byte	pData
0	ON/OFF: 0: 补正无效、1: 补正有效
1 至 3	预约 (固定为 0)
4	区域起始位置 1: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
5	
6	
7	
8	区域结束位置 1: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
9	
10	
11	
12	区域起始位置 2: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
13	
14	
15	
16	区域结束位置 2: 测量范围内的任意数值 (单位 0.01 μm 、有符号的 32 bit 整数 例: 5 mm = 500000)
17	
18	
19	
20	补正后高度 0 至 999.99900 mm: 0 至 99999900
21	
22	
23	
24	补正跨度: 0 至 131072 * 将设定值除以 65536, 即得到 补正跨度值。 (条件: $0 < \text{补正跨度值} < 2$) 例: 设定为 98304, 则 $98304 \div 65536 = 1.5$ 补正跨度值为 1.5。
25	
26	
27	

12 高速数据通信指令使用示例

使用高速数据通信指令时，可将当前测得的轮廓数据通过控制器高速输出到计算机中。可在使用 LJ-X 系列测量的同时，对传送到计算机中的数据进行处理。

12.1 高速数据通信准备

为进行高速数据通信，请执行以下设定。

[设定设备]

- 请使用支持 Gigabit 通信的网卡
- 请使用支持 Gigabit 通信的集线器
- 请使用 7 类以上或支持 10 GBASE-T 的 Ethernet 电缆

[设定]

- 关于高速数据通信开始前准备请求中指定的参数 Req 的 SendPosition

■ Req

SendPosition: 发送起始位置。

0: 从上次发送完毕的位置开始（首次执行就从最早的数据开始）

1: 从最早的数据开始

2: 从下一数据开始

* 开始高速通信后读取轮廓时，请指定 2。如果指定 0 或 1，将读取高速通信开始前存储在控制器内部的轮廓。

* 所读取的轮廓将从控制器的内存中删除。

* 采样周期快于读取速度时，会造成内存 FULL。如果在测量通用设定中指定的“内存 FULL 时动作”为“停止”，则轮廓存储中止；如果为“覆盖”，则从最早的数据开始覆盖。

* 执行一次高速数据通信后，隔一段时间执行下一次高速数据通信时，内存被覆盖（内存 FULL 时动作为“覆盖”时），上次发送完毕位置的数据将被覆盖。此时如果指定 0，将返回错误信息。

[执行高速数据通信]

根据目标物的形状及测量内容，高速数据通信有以下 2 种步骤。

- 不使用批处理设定时
- 使用批处理设定时

批处理设定是将指定个数的轮廓整合为 1 个整体进行处理的功能。可通过 LJ-X Navigator 设定是否使用。

[高速数据通信种类]

高速数据通信有两种类型，一种将获取的轮廓数据直接存储在缓存中，另一种将轮廓数据中包含的高度数据和亮度数据以容易转换为 Bitmap 等格式存储在缓存中（Simple Array）。

[确认读取的轮廓]

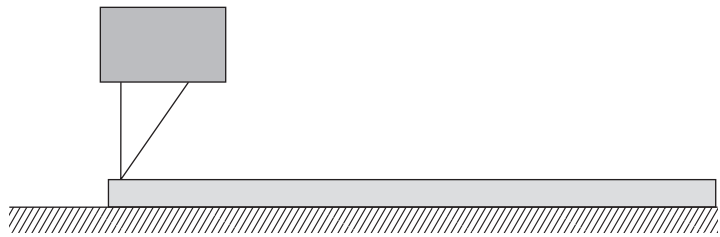
通过 Callback 函数读取的轮廓分别含有标题信息，标题中会记载下列信息。

- 编码器的 Z 相是否已输入（参照“7.3 轮廓数据结构”的“轮廓标题信息结构”）
- 触发计数 表示轮廓是由测量开始后的第几次触发获得的。
- 编码器计数 编码器计数值。编码器的输入周期快于设定的采样周期时，该编码器输入将不被视作触发，而是作为编码器计数值进行计数。（输入脉冲超出了编码器最短输入时间时）

请根据用途，利用这些信息确定轮廓的位置。

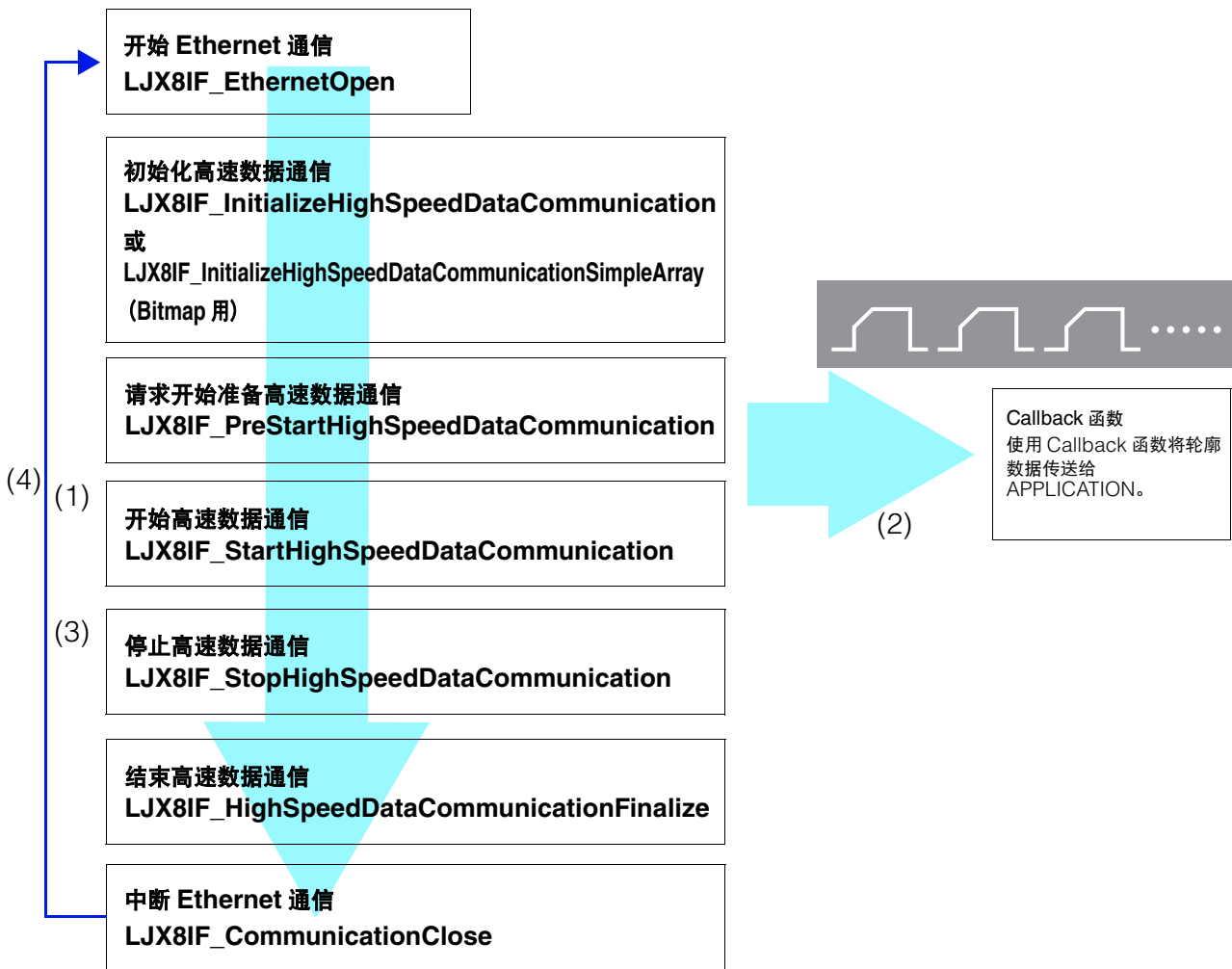
12.2 不使用批处理功能的高速数据通信

如下图所示，用于测量连续目标物的轮廓时。用于需要测量、输出无数数据点数上限的轮廓时。



[指令的步骤]

按照以下步骤发送指令。



(1)开始高速数据通信

向控制器发送高速数据通信开始指令后，在 LJ-X 中测量的轮廓数据将开始输出。每次向 LJ-X 输入触发时，都将进行测量。

(2)Callback 函数

使用 Callback 函数时，每当高速数据通信初始化时所指定个数的轮廓数据被传送到计算机，应用程序就会接收到这些轮廓数据。

*1 Callback 函数将在下列条件下被调用。

- 控制器发送出指定点数的轮廓
- 高速通信停止（设定被变更，高速数据通信停止指令、内存清除指令被发出）
- 程序被切换

*2 将批处理设定中指定点数的轮廓传送到计算机后，8.3.1.1 dwNotify 参数的第 16 bit 开启。

*3 如果 Callback 函数中指定的轮廓个数较少，传输的频率会增高，计算机负荷由此加大，可能导致计算机处理速度变慢。请在确认应用程序的负荷状态后，再决定传输的个数。

(3)停止高速数据通信 停止从控制器向计算机输出轮廓。高速通信停止后，8.3.1.1 dwNotify 参数的第 1 bit 开启。

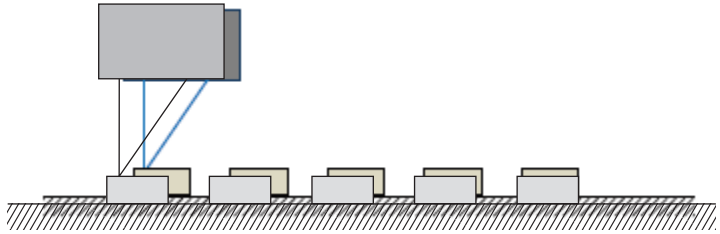
(4)重新开始高速数据通信

停止 1 次高速通信后重新开始高速数据通信时，请按照“结束高速数据通信”→“中断 Ethernet 通信”→“开始 Ethernet 通信”→“初始化 Ethernet 高速通信”→“请求开始准备 Ethernet 高速数据通信”的流程，开始高速数据通信。

12.3 使用批处理功能的高速数据通信

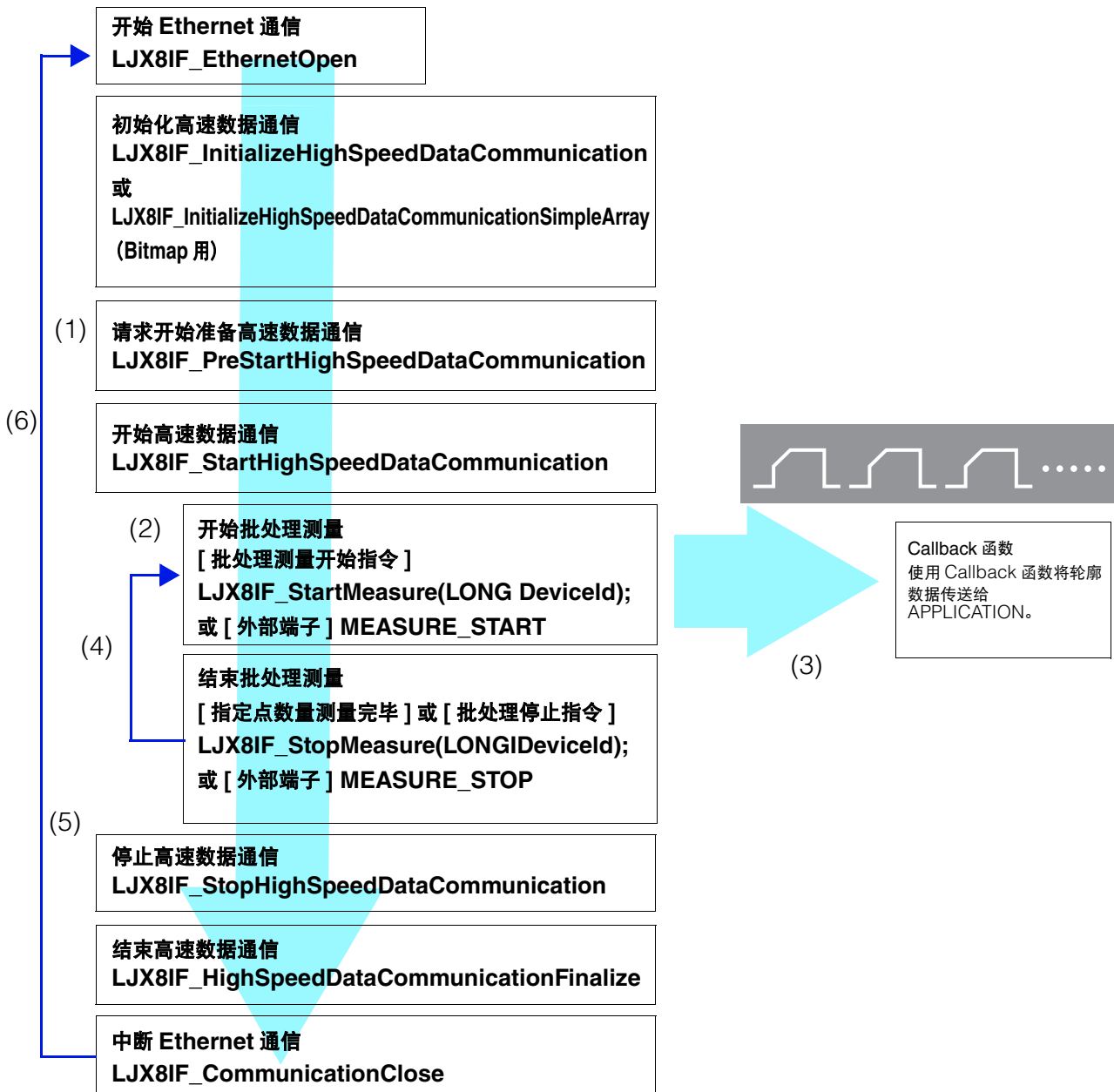
如下图所示，用于批处理流转目标物的逐一测量。使用批处理功能，具有以下优势。

- 由于可将最多 60000 项轮廓整合为 1 整块数据加以管理，计算机对数据的处理更便捷。
- 对事先设定好的数据点数进行测量后，将自动结束数据测量。
- 一旦开始高速数据通信，即可通过控制控制器的批处理开始时间，控制开始向计算机发送轮廓的时间。



[指令的步骤]

按照以下步骤发送指令。请确认控制器的批处理设定是否为 ON。



(1)开始高速数据通信

向控制器发送高速数据通信开始指令后，在 LJ-X 中测量的轮廓数据将开始输出。

* 批处理设定： ON 时，只要不开始批处理测量，就不会接收触发输入。因此，即使在步骤①发送高速数据通信开始指令，也不会输出轮廓。

(2)开始批处理测量

开始批处理测量。开始批处理测量的方法有如下 2 种。

- 端子台： 开启 MEASURE_START 端子后开始批处理测量。
- 指令： 批处理测量开始指令 （8.1.4 测量控制 LJX8IF_StartMeasure(LONG IDeviceId);)
开始批处理测量。

(3) Callback 函数

使用 Callback 函数时，每当高速数据通信初始化时所指定个数的轮廓数据被传送到计算机，应用程序就会接收到这些轮廓数据。

*1 Callback 函数将在下列条件下被调用。

- 控制器发送出指定点数的轮廓
- 高速通信停止（设定被变更，高速数据通信停止指令、内存清除指令被发出）
- 批处理点数量数据获取完毕
- 程序被切换

*2 将批处理设定中指定点数的轮廓传送到计算机后，8.3.1.1 dwNotify 参数的第 16 bit 开启。

*3 如果 Callback 函数中指定的轮廓个数较少，传输的频率会增高，计算机负荷由此加大，可能导致计算机处理速度变慢。请在确认应用程序的负荷状态后，再决定传输的个数。

(4)循环

通过循环操作批处理开始，可连续输出轮廓。批处理测量将在下列条件下停止。

- 已获取指定点数的轮廓
- 输入了批处理测量停止信号

(5)高速数据通信

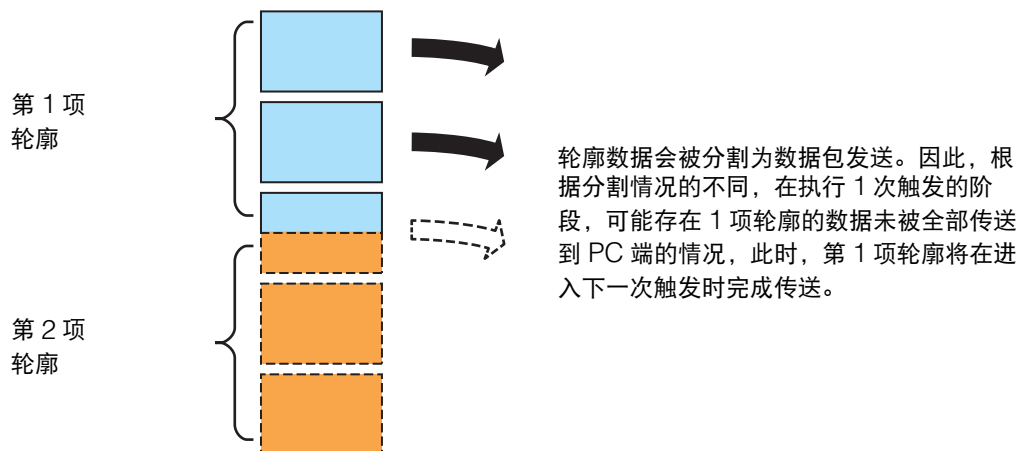
停止从控制器向计算机输出轮廓。高速通信停止后，8.3.1.1 dwNotify 参数的第 1 bit 开启。

(6)重新开始高速数据通信

停止 1 次高速通信后重新开始高速数据通信时，请按照“结束高速数据通信”→“中断 Ethernet 通信”→“开始 Ethernet 通信”→“初始化 Ethernet 高速通信”→“请求开始准备 Ethernet 高速数据通信”的流程，开始高速数据通信。

12.4 高速数据通信的故障排除

症状	确认内容	处理方法
开始高速数据通信后，程序异常终止。	是否指定了正确的返回函数调用协议。	请确保返回函数的调用协议与头文件的内容匹配。
获取的轮廓数据不定期出现异常。	主线程所使用的数据是否未经过函数内排他处理。	请对通用数据进行排他处理。
获取的轮廓数据定期出现异常。	返回函数内的（文件保存等）处理负荷是否过重。	请变更设定，缩短返回函数的处理时间。
	所使用通信路径的通信速度是否足够。	请变更为 1000BASE-T 等高速通信路径。
高速通信被中断。	返回函数内的（文件保存等）处理负荷是否过重。	请变更设定，缩短返回函数的处理时间。
	所使用通信路径的通信速度是否足够。	请变更为 1000BASE-T 等高速通信路径。
	传感头电缆连接器是否松动？或电缆是否断线？（控制器本体 LED 是否点亮红灯？）	请紧固连接器，关闭 / 开启电源。 请更换电缆。
未成功获取预计数量的轮廓	请确认示例程序。（详情请参照“10 示例程序”。）	请确认网络环境、LAN 电缆类别等。
		可能是由于应用程序方面的处理速度太慢，请变更处理，或减慢获取速度。
通过返回函数输入指定次数的触发后，返回函数未被调用。	轮廓被从控制器传输到 PC 时，会被分割为数据包发送，可能存在 1 触发 ≠ 1 轮廓传送的情况。 * 见下文	请输入更多的触发。



修订履历

印刷日期	版 本	修订内容
2019 年 9 月	初 版	
2019 年 12 月	2 版	
2020 年 4 月	3 版	
2021 年 1 月	4 版	

产品保证书

KEYENCE 的产品经过严格的出厂检验。如出现故障，请与就近的 KEYENCE 办事处联系，并提供故障详细情况。

1. 保质期

保质期为一年，从产品发送到购方指定地点之日算起

2. 保修范围

- (1) 如果在上述保质期内出现 KEYENCE 公司造成的故障，我们将免费修理产品。但是以下情况不属于保修范围。
- 未按照操作手册、用户手册或购方与 KEYENCE 公司专门达成的技术要求中规定的条件、环境下的不正确的操作，或不正确使用造成的故障。
 - 故障不是由于产品缺陷，而是购方设备或购方软件设计造成的。
 - 由非 KEYENCE 公司人员进行的修改或修理而造成的故障。
 - 按照操作手册或用户手册正确维修或更换易损件等规定可以完全避免的故障。
 - 在产品从 KEYENCE 公司发货后，因无法预料的科学技术水平变化等因素而造成的故障。
 - 由于火灾、地震和洪水等自然灾害，或异常电压等外部因素造成的故障，我公司不负责保修。
- (2) 保修范围只限于第 (1) 条规定的情况，KEYENCE 公司对其设备造成的购方间接损失（设备损坏、机会丧失、利润损失等）或其它损失不承担任何责任。

3. 产品适用性

KEYENCE 公司的产品是针对一般行业的通用产品而设计生产的。因此，我公司产品不得用于下列应用且不适合其使用。但是，如果购方以对自己负责的态度提前就产品的使用向我方进行了咨询并了解产品的技术规范，等级和性能，并采取必要的安全措施，则产品可以使用。在这种情况下，产品保修范围和上述相同。

- 对生命和财产有严重影响的设施，如核发电厂、机场、铁路、轮船、机动装置及医疗设备
- 公共事业如电力、气体及供水服务
- 相似条件或环境的户外使用

有关规格等的变化不再另行通知。

KEYENCE CORPORATION

1-3-14, Higashi-Nakajima, Higashi-Yodogawa-ku, Osaka, 533-8555, Japan 电话: +81-6-6379-2211

www.keyence.com/glb

全球网络 -国家和地区-

奥地利 电话: +43 (0)2236 378266 0	法国 电话: +33 1 56 37 78 00	意大利 电话: +39-02-6688220	波兰 电话: +48 71 368 61 60	台湾 电话: +886-2-2721-8080
比利时 电话: +32 (0)15 281 222	德国 电话: +49-6102-3689-0	韩国 电话: +82-31-789-4300	罗马尼亚 电话: +40 (0)269 232 808	泰国 电话: +66-2-369-2777
巴西 电话: +55-11-3045-4011	香港 电话: +852-3104-1010	马来西亚 电话: +60-3-7883-2211	新加坡 电话: +65-6392-1011	英国及爱尔兰 电话: +44 (0)1908-696-900
加拿大 电话: +1-905-366-7655	匈牙利 电话: +36 1 802 7360	墨西哥 电话: +52-55-8850-0100	斯洛伐克 电话: +421 (0)2 5939 6461	美国 电话: +1-201-930-0100
中国 电话: +86-21-3357-1001	印度 电话: +91-44-4963-0900	荷兰 电话: +31 (0)40 206 6100	斯洛文尼亚 电话: +386 (0)1 4701 666	越南 电话: +84-24-3772-5555
捷克共和国 电话: +420 220 184 700	印度尼西亚 电话: +62-21-2966-0120	菲律宾 电话: +63-(0)2-8981-5000	瑞士 电话: +41 (0)43 455 77 30	

