RICOH

RICOH SC-20

套接字通信功能使用说明书

本说明书的阅读方式

关于标记

本说明书所使用的标记含义如下。

重要

说明了操作时的注意事项和限制事项等。请务必阅读。

补充

说明了便利的信息以及补充的操作方法。

参考 /（→P. ##）

表示参考来源。

[ ]

表示画面上的项目和按钮的名称。

RICOH

目录

1. 概要

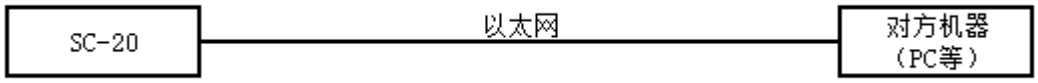
SC-20 可使用 TCP/IP 的套接字通信功能连接外部设备。
本说明书介绍套接字通信的连接步骤和套接字通信时设定的数据格式。

连接配置

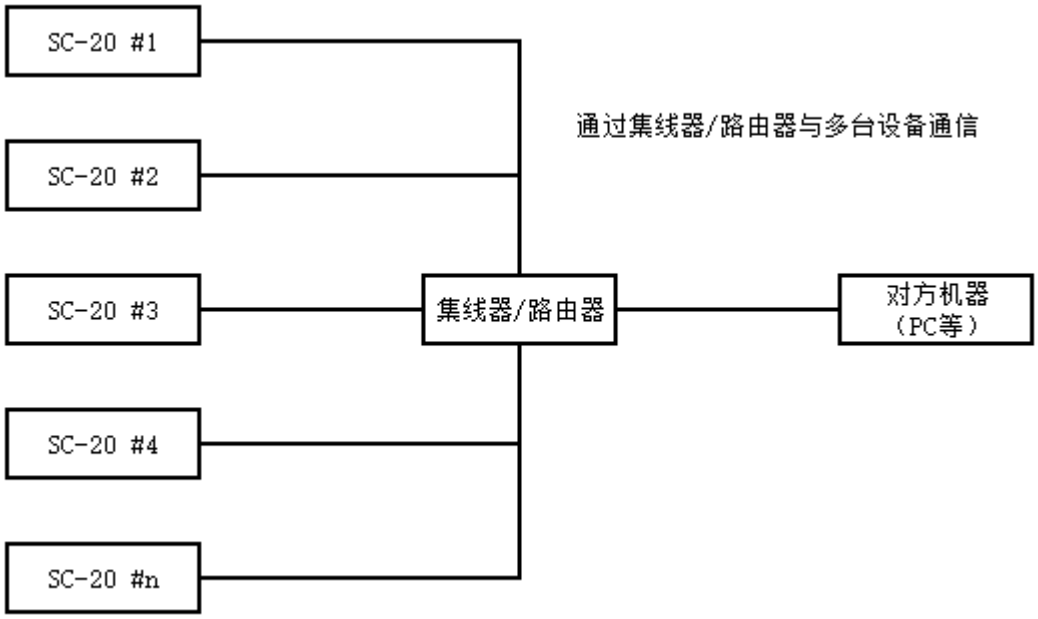
连接配置

使用以太网连接时，如下图所示，可连接多个作业支持相机系统。
根据 SC-20 的连接方式，需要搭建用于对方机器操作的应用软件。

例 1：



例 2：



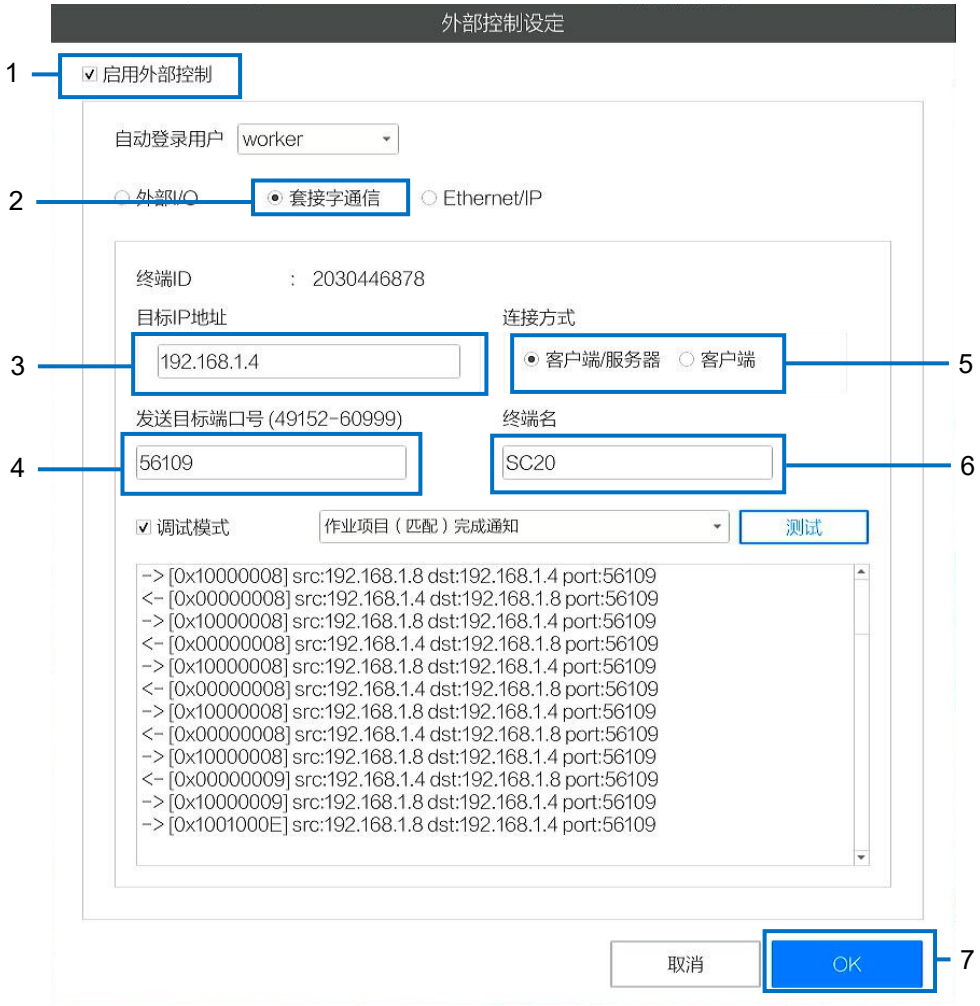
2. 套接字通信

启用套接字通信控制功能

以［管理员模式］登录 SC-20，选择［系统设定］菜单中的［外部控制设定...］，显示以下画面。

参考

- 关于 SC-20 的操作详情，请参阅 SC-20 系列使用说明书。



1. **勾选［启用外部控制］。**
2. **选择［套接字通信］。**
3. **在［目标 IP 地址］中设定对方机器（套接字通信的对象）的 IP 地址。**
4. **在［发送目标端口号］中，设定对方机器的端口号。**
5. **在［连接方式］中设定［客户端/服务器］或［客户端］。**
6. **在［终端名］中输入设备名称。**
   请使用 1 至 50 个半角英文字母和数字输入所需名称。
7. **点击［OK］。**
   保存设置，功能将在重新启动后生效。

补充

- ［终端 ID］由系统自动设定。

执行调试模式

可设定调试模式。进行对方机器的应用程序开发时，可使用调试模式进行通畅确认和操作验证。

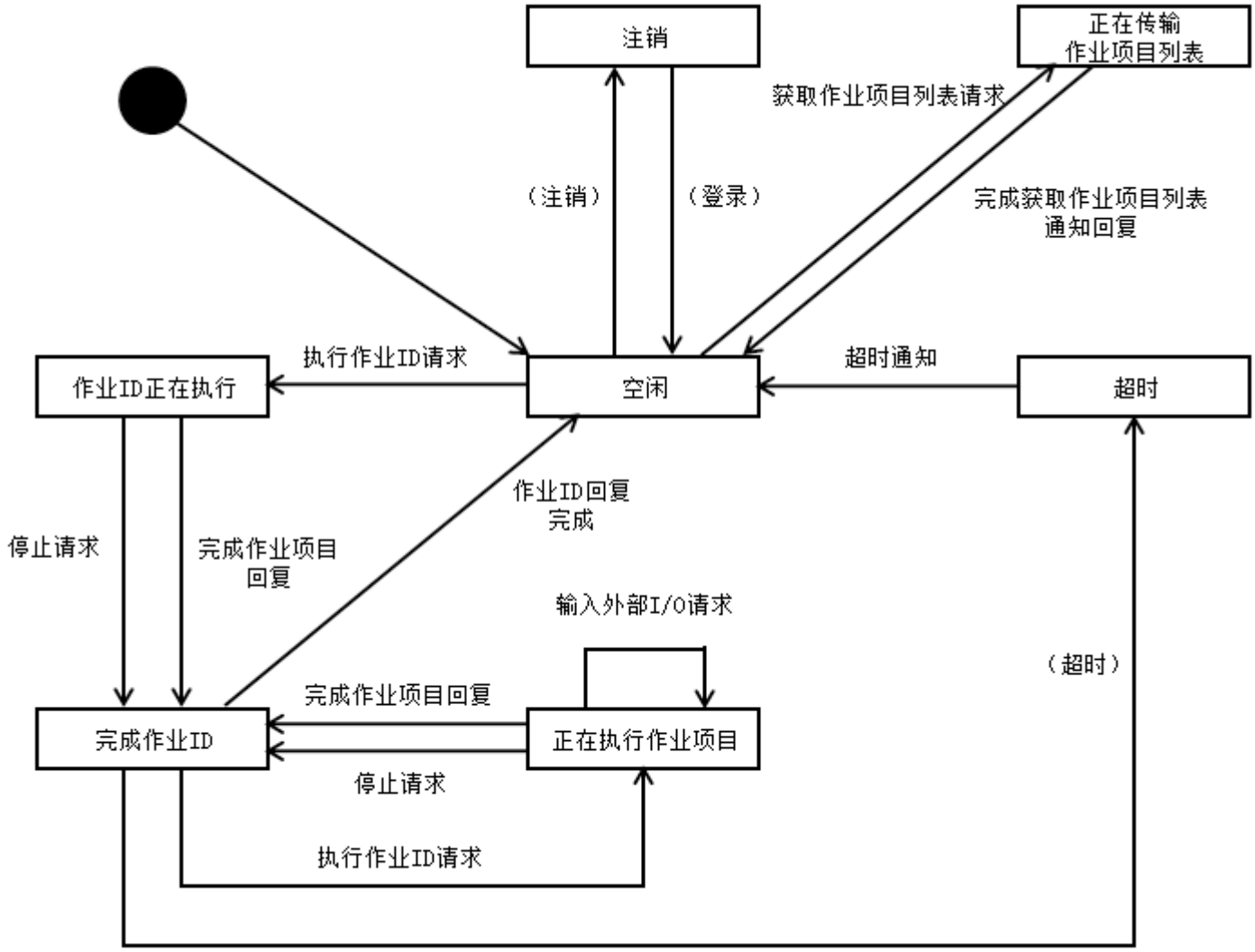
补充

- 要设定调试模式时，请预先启用套接字通信功能（→P. 6）。

1. **勾选［调试模式］。**
2. **从左侧的下拉菜单中选择通知消息，然后点击［测试］。**
   可以根据下拉菜单发送包含虚拟数据的通知消息。
3. **显示 SC-20 和对方机器的消息日志。**

指令和系统的状态

在套接字通信中交换的指令以及系统的状态变化如下所示。
括号中的处理是用户的手动操作，或系统内自动执行的操作。



序列

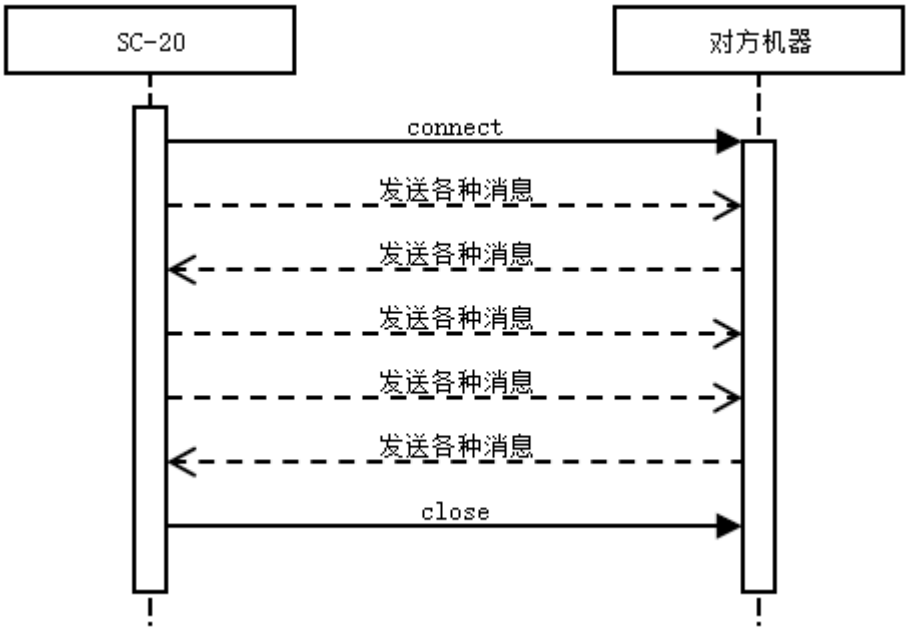
说明在套接字通信控制功能中设想的序列。

重要

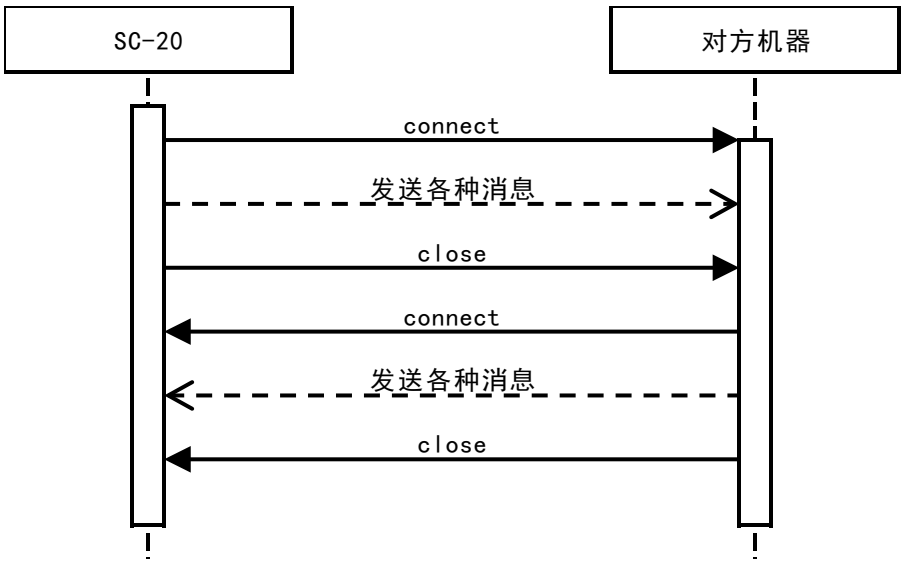
・如果进行了以下所示序列以外的通信，则不能进行正确的处理。
・请不要在其他命令的序列中发送请求命令。

连接方式

【客户端模式】
这是执行 connect 并一直保持到系统停止结束为止的连接方式。

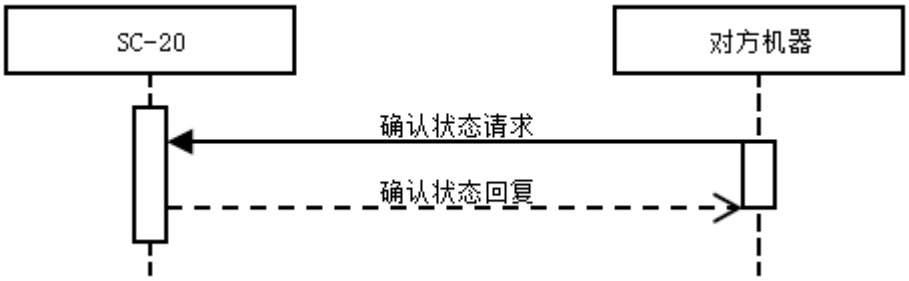


【客户端/服务器模式】
该方式在发送侧执行 connect 并在发送消息后执行 close。
*从对方机器向 SC-20 方向的 IP 端口固定为 56109。



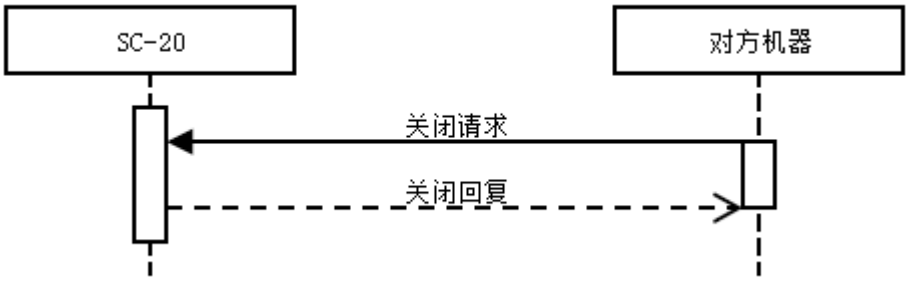
确认状态

在对方机器确认 SC-20 的状态时，对方机器将发送“确认状态请求”。
相机状态在任意时机作为“确认状态回复”发送。



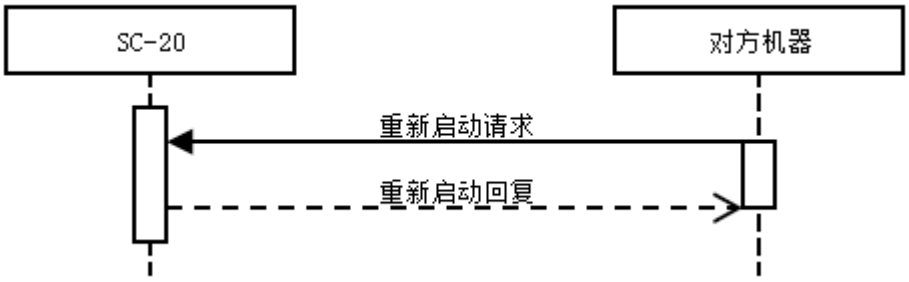
关闭

在对方机器关闭 SC-20 时，对向机器将发送“关闭请求”。
相机状态在任意时机作为“关闭回复”发送。



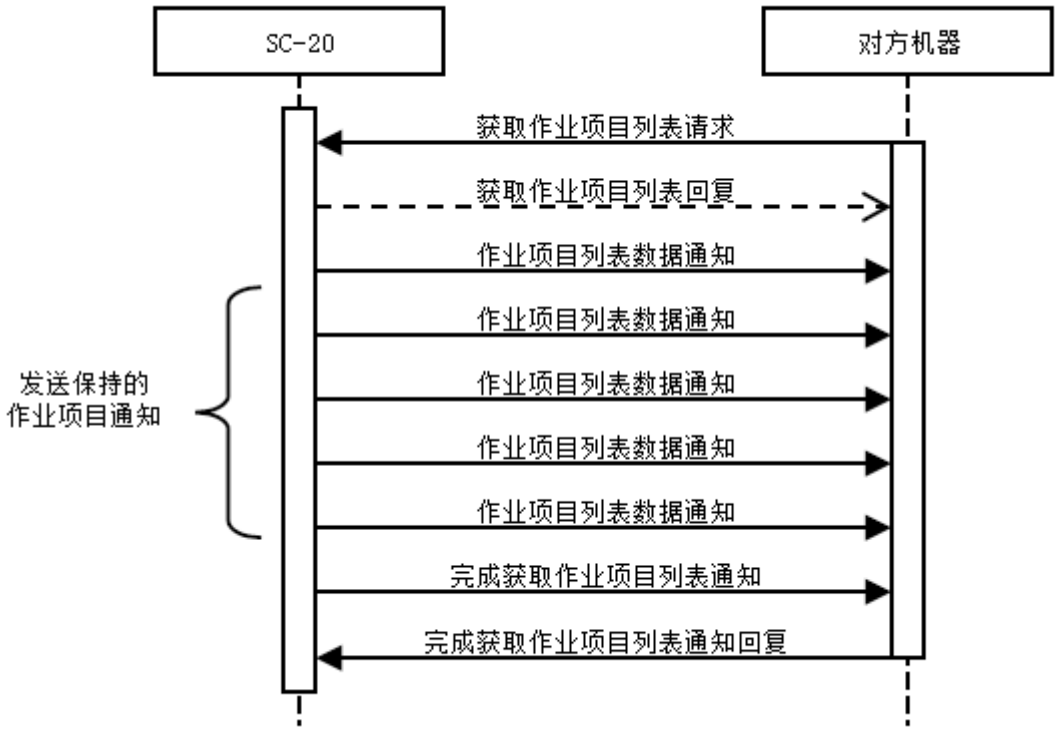
重新启动

在对方机器重新启动 SC-20 时，对方机器将发送“重新启动请求”。
相机状态在任意时机作为“重新启动回复”发送。



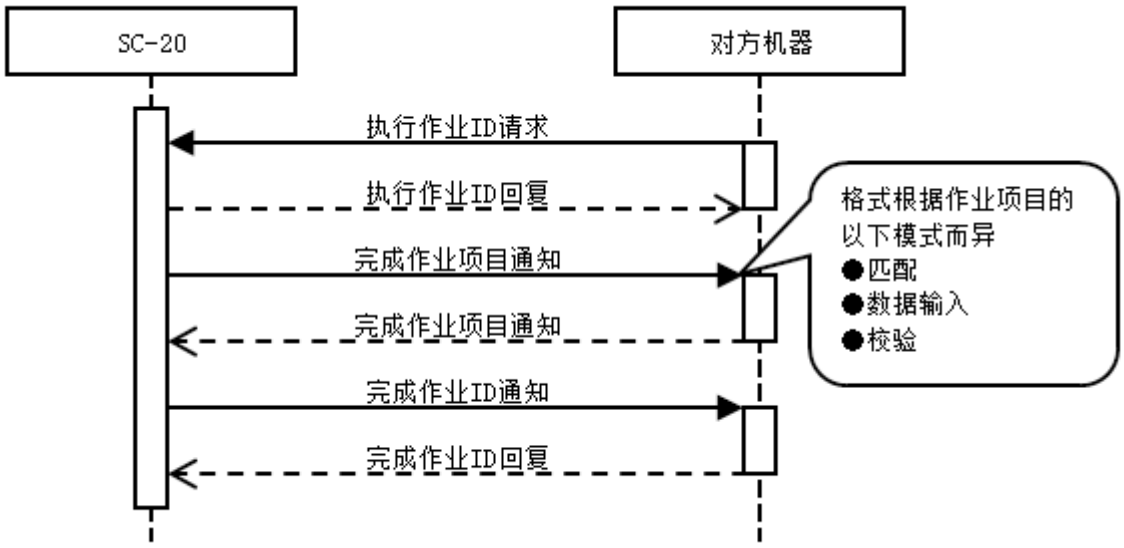
获取作业项目列表

要获取 SC-20 中登记的作业项目时，对方机器将发送“获取作业项目列表请求”。SC-20 在发送“作业项目列表回复”后，发送“作业项目列表数据通知”。
对于在 SC-20 中登记的每 1 个作业项目，发送 1 次“作业项目列表数据通知”。所有作业项目的“作业项目列表数据通知”发送完成后，将发送“完成获取作业项目列表通知”。对方机器发送“完成获取作业项目列表回复”，序列结束。



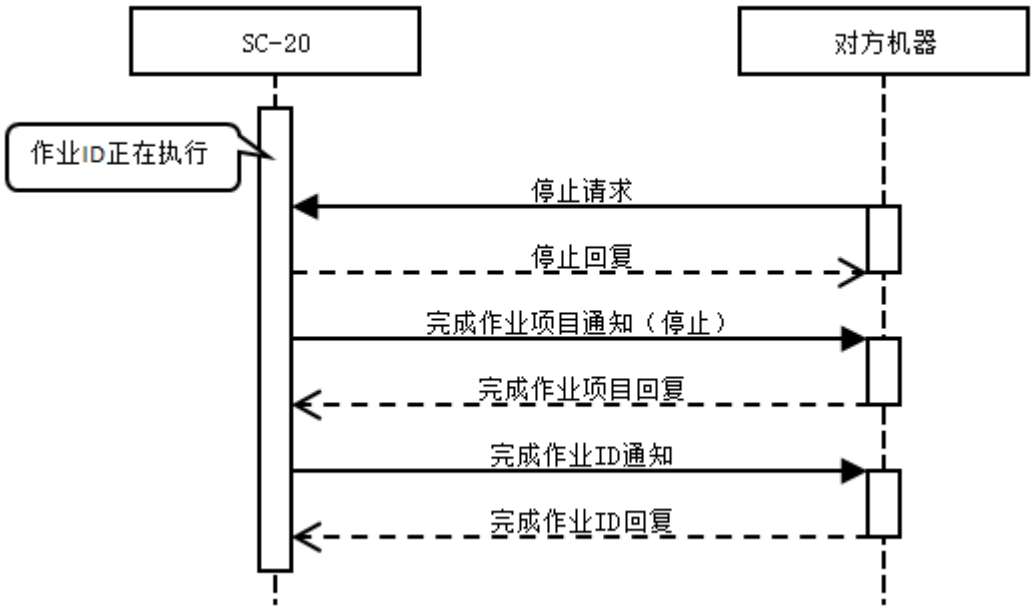
执行作业 ID 处理

若要从对方机器切换 SC-20 的作业 ID，并从作业 ID 中登记的首个作业项目开始执行序列，将从对方机器发送“执行作业 ID 请求”。从作业 ID 中登记的首个作业项目开始依次执行。执行结果将从 SC-20 通过“完成作业项目通知”发送。
完成作业 ID 中登记的所有作业项目执行后，SC-20 将发送“完成作业 ID 通知”。对方机器向 SC-20 发送“完成作业 ID 回复”，序列结束。



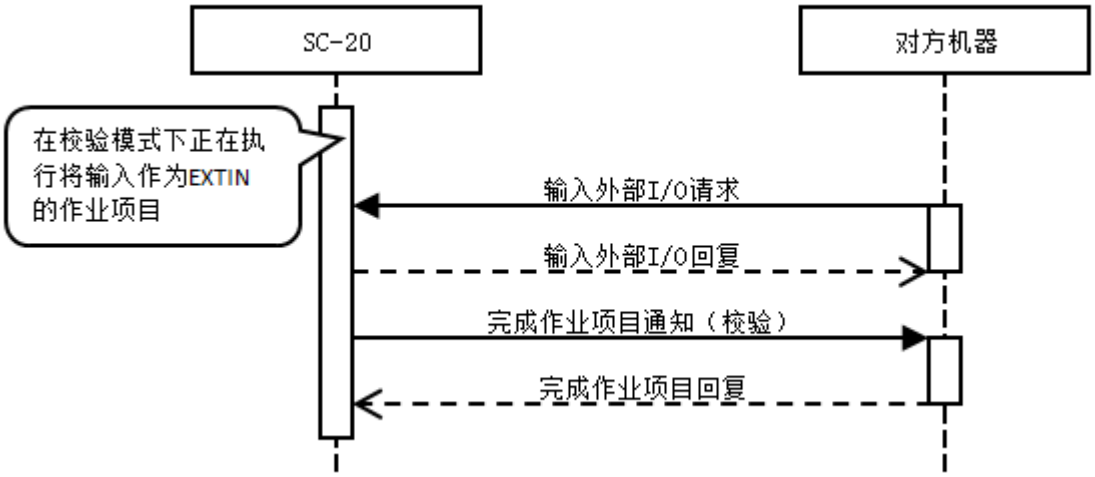
停止作业项目

若要在“正在执行作业项目”期间停止处理，向 SC-20 发送“停止请求”即可停止作业。



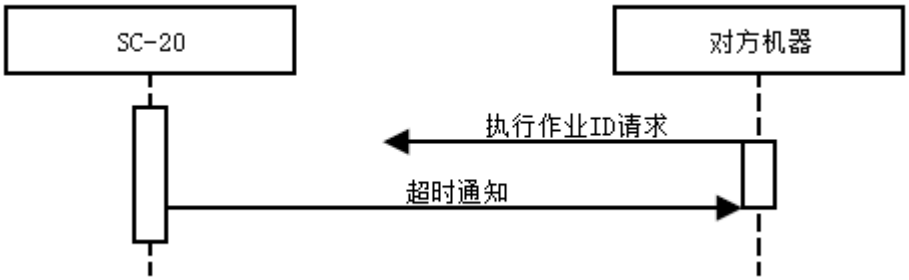
输入外部 I/O

如果在执行预先设定校验模式 EXTIN 的作业项目期间，发送“输入外部 I/O 请求”，则执行命令参数对应的处理，然后 SC-20 依次发送“输入外部 I/O 回复”、“完成作业项目通知”。



超时

若来自对方机器的“执行作业 ID 请求”因某种原因未送至 SC-20，当 SC-20 内部设定的计时器期满时，系统内部将发送“超时通知”，然后向对方机器发送“超时通知”。由于网络路径问题，“超时通知”可能无法到达，所以请在对方机器侧确认网络情况后，采取应对措施。



3. 消息 ID

本章节说明套接字通信中使用的消息 ID。

套接字通信消息 ID

请求类

消息名		消息 ID	
		对方机器→SC-20	SC-20→对方机器
		请求	回复
	停止	0x00000003	0x10000003
	获取作业项目列表	0x00000004	0x10000004
	执行作业 ID	0x00000005	0x10000005
	输入外部 I/O	0x00000007	0x10000007
	确认状态	0x00000008	0x10000008
	执行关闭	0x00000009	0x10000009
	执行重新启动	0x0000000A	0x1000000A

通知类

消息名		消息 ID	
		SC-20→对方机器	对方机器→SC-20
		通知	通知回复
	完成作业项目（匹配）	0x10010002	0x00010007
	完成作业项目（数据输入）	0x10010003	0x00010007
	完成作业项目（校验模式）	0x10010004	0x00010007
	完成作业项目（停止）	0x10010005	0x00010007
	完成作业 ID	0x10010008	0x00010008
	作业项目列表数据	0x10010009	-
	完成获取作业项目列表	0x1001000B	0x0001000B
	系统停止	0x1001000E	-
	超时	0x1001000F	-

消息标题

以下为套接字通信的数据区域中设定的通用消息标题配置图。
对于终端 ID 及终端名，请使用 SC-20 启动时发送的“启动通知”内的数据。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

消息 ID（请求消息）

停止请求

消息 ID	消息名	说明
0x00000003	停止请求	若在“完成作业项目通知”前发送“停止请求”，将中断正在识别的处理。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

停止回复

消息 ID	消息名	说明
0x10000003	停止回复	回复从对方机器发送的“停止请求”的消息。 回复消息中将输入结果和错误代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

获取作业项目列表请求

消息 ID	消息名	说明
0x00000004	获取作业项目列表请求	获取 SC-20 中登记的作业项目列表。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

获取作业项目列表回复

消息 ID	消息名	说明
0x10000004	获取作业项目列表回复	回复从对方机器发送的“获取作业项目列表请求”的消息。 结果中将插入 SC-20 中登记的作业项目数量（最多 32767 个）。 NG 时将返回-1，并在错误代码中给出表示事由的代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	1-32767：总项目数 -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

执行作业 ID 请求

消息 ID	消息名	说明
0x00000005	执行作业 ID 请求	从对方机器开始执行作业 ID。 外部 I/O 的 OUT0 中设定为“RUN”时，转换至 ON 状态。 补充 • ON/OFF 取决于用户定义。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	作业 ID			
0x0088	作业指示			
0x00c8	作业项目			
0x0108	操作员 ID			
0x0148	作业编号			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048 - 0x0087	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0088 - 0x00c7	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00c8 - 0x0107	64 字节	char	作业项目	最多 50 个半角英文字母和数字
0x0108 - 0x0147	64 字节	char	操作员 ID	最多 50 个半角英文字母和数字
0x0148 - 0x0187	64 字节	char	作业编号	最多 50 个半角英文字母和数字

执行作业 ID 回复

消息 ID	消息名	说明
0x10000005	执行作业 ID 回复	回复从对方机器发送的“执行作业 ID 请求”的消息。 回复消息中将输入结果和错误代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

输入外部 I/O 请求

消息 ID	消息名	说明
0x00000007	输入外部 I/O 请求	作业项目在校验模式下待机时向 SC-20 发送的消息。 执行等同于外部 I/O 的 EXTIN 的操作。

消息格式

地址	位										
	31						16	15			0
0x0000	消息 ID										
0x0004	终端 ID										
0x0008	终端名										
0x0048	作业 ID										
0x0088	预留							EXTIN9	EXTIN8	EXTIN7	EXTIN6	EXTIN5	EXTIN4	EXTIN3	EXTIN2	EXTIN1	EXTIN0

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048 - 0x0087	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0088	4 字节	uint32	外部 I/O	在位字段中设置外部 I/O 的输入逻辑值。 [0]EXTIN0：1/0 [1]EXTIN1：1/0 [2]EXTIN2：1/0 [3]EXTIN3：1/0 [4]EXTIN4：1/0 [5]EXTIN5：1/0 [6]EXTIN6：1/0 [7]EXTIN7：1/0 [8]EXTIN8：1/0 [9]EXTIN9：1/0 [10-31]：预留

输入外部 I/O 回复

消息 ID	消息名	说明
0x10000007	输入外部 I/O 回复	回复“输入外部 I/O 请求”的消息。 回复消息中将输入结果和错误代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

确认状态请求

消息 ID	消息名	说明
0x00000008	确认状态请求	用于确认套接字通信当前状态的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

确认状态回复

消息 ID	消息名	说明
0x10000008	确认状态回复	回复“确认状态请求”的消息。 回复消息中将输入结果和错误代码。 根据结果的值表示当前的相机状态。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	使用以下编号显示指令和系统的状态 （→P. 8）的各状态。 1：注销 2：空闲 3、4：作业项目正在传输 5、6：开始作业 ID 7、13：正在执行作业项目 8、9、14：作业 ID 正在执行 10、11、12：完成作业 ID 15：超时 -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

执行关闭请求

消息 ID	消息名	说明
0x00000009	执行关闭请求	要关闭 SC-20 时使用的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

补充

・执行关闭请求在注销时不起作用。

执行关闭回复

消息 ID	消息名	说明
0x10000009	执行关闭回复	回复“执行关闭请求”的消息。 回复消息中将输入结果和错误代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

执行重新启动请求

消息 ID	消息名	说明
0x0000000A	执行重新启动请求	要重新启动 SC-20 时使用的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

补充

・执行重新启动请求在注销时不起作用。

执行重新启动回复

消息 ID	消息名	说明
0x1000000A	执行重新启动回复	回复“执行重新启动请求”的消息。 回复消息中将输入结果和错误代码。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

消息 ID（通知消息）

完成作业项目通知（匹配）

消息 ID	消息名	说明
0x10010002	完成作业项目通知（匹配）	退出匹配处理时向对方机器侧发送的消息。“校验点 ID_X”后的数据将根据“校验点数”的设定数量进行数据输入。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			
0x0090	作业指示			
0x00d0	作业项目			
0x0110	操作员 ID			
0x01d8	作业编号			
0x02a0	经过时间		作业项目最终结果	
0x02a4	基准点类似度			
0x02ac	校检点数		基准点旋转角度	
0x02b0	预留	判定结果	模式（匹配）	校检点 ID_1
0x02b4	匹配时间[毫秒]		旋转角度	
0x02b8	类似度			
0x02c0	预留	判定结果	模式（匹配）	校检点 ID_2
0x02c4	匹配时间[毫秒]		旋转角度	
0x02c8	类似度			
≀	≀			
0x04a0	预留	判定结果	模式（匹配）	校检点 ID_20
0x04a4	匹配时间[毫秒]		旋转角度	
0x04a8	类似度			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0090 - 0x00cf	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00d0 - 0x010f	64 字节	char	作业项目	最多 50 个半角英文字母和数字
0x0110 - 0x01d7	200 字节	char	操作员 ID	最多 198 个半角英文字母和数字
0x01d8 - 0x029f	200 字节	char	作业编号	最多 198 个半角英文字母和数字
0x02a0	2 字节	int16	作业项目 最终结果	0：OK -1：NG -2：基准点 NG
0x02a2	2 字节	uint16	经过时间（秒）	以秒为单位设定作业经过时间
0x02a4	8 字节	double	基准点 类似度	在 0.00000～1.00000 的范围内设定
0x02ac	2 字节	int16	基准点 旋转角度	在 180～-180 的范围内设定。
0x02ae	2 字节	uint16	校检点数	在 0～20 的范围内设定校检点数
0x02b0	1 字节	uchar	校检点 ID_1	在 1～20 的范围内设定校验点 ID
0x02b1	1 字节	uchar	模式	0：形状 1：颜色识别 2：纹理
0x02b2	1 字节	char	判定结果	0：OK -1：NG
0x02b3	1 字节	uchar	预留	未使用区域
0x02b4	2 字节	int16	旋转角度	在 180～-180 的范围内设定 *颜色识别和纹理固定为 0
0x02b6	2 字节	uint16	匹配时间[毫秒]	在 0～65535 的范围内设定
0x02b8	8 字节	double	类似度	在 0.00000～1.00000 的范围内设定

完成作业项目通知（数据输入）

消息 ID	消息名	说明
0x10010003	完成作业项目通知（数据输入）	退出数据输入时向对方机器侧发送的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			
0x0090	作业指示			
0x00d0	作业项目			
0x0110	操作员 ID			
0x01d8	作业编号			
0x02a0	经过时间（秒）		作业项目最终结果	
0x02a4	零件编号（已设定的编号）			
0x0324	输入的数据（用户输入的值）			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0090 - 0x00cf	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00d0 - 0x010f	64 字节	char	作业项目	最多 50 个半角英文字母和数字
0x0110 - 0x01d7	200 字节	char	操作员 ID	最多 198 个半角英文字母和数字
0x01d8 - 0x029f	200 字节	char	作业编号	最多 198 个半角英文字母和数字
0x02a0	2 字节	int16	作业项目 最终结果	0：OK -1：NG
0x02a2	2 字节	uint16	经过时间（秒）	以秒为单位设定作业经过时间
0x02a4 - 0x0323	128 字节	char	零件编号	半角英文字母和数字
0x0324 - 0x0523	512 字节	char	输入的数据	半角英文字母和数字

完成作业项目通知（校验模式）

消息 ID	消息名	说明
0x10010004	完成作业项目通知（校验模式）	退出校验模式时向对方机器侧发送的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			
0x0090	作业指示			
0x00d0	作业项目			
0x0110	操作员 ID			
0x01d8	作业编号			
0x02a0	经过时间（秒）		作业项目最终结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0090 - 0x00cf	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00d0 - 0x010f	64 字节	char	作业项目	最多 50 个半角英文字母和数字
0x0110 - 0x01d7	200 字节	char	操作员 ID	最多 198 个半角英文字母和数字
0x01d8 - 0x029f	200 字节	char	作业编号	最多 198 个半角英文字母和数字
0x02a0	2 字节	int16	作业项目最终结果	0：OK -1：NG
0x02a2	2 字节	uint16	经过时间（秒）	以秒为单位设定作业经过时间

完成作业项目通知（停止）

消息 ID	消息名	说明
0x10010005	完成作业项目通知（停止）	执行来自 UI、外部 I/O 及套接字通信的停止后从 SC-20 发送的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			
0x0090	作业指示			
0x00d0	作业项目			
0x0110	经过时间（秒）		停止原因	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0090 - 0x00cf	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00d0 - 0x010f	64 字节	char	作业项目	最多 50 个半角英文字母和数字
0x0110	2 字节	int16	停止原因	0：来自 UI 的停止 1：来自外部 I/O 的停止 2：来自套接字通信的停止
0x0112	2 字节	uint16	经过时间（秒）	以秒为单位设定作业经过时间

完成作业项目通知回复

消息 ID	消息名	说明
0x00010007	完成作业项目通知回复	回复从 SC-20 发送的“完成作业项目通知”的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	预留			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	4 字节	Uint32	预留	未使用区域

完成作业 ID 通知

消息 ID	消息名	说明
0x10010008	完成作业 ID 通知	作业 ID 中管理的作业项目全部执行完成后从 SC-20 发送的消息。 “完成作业项目通知”的“结果”字段中设定为“2”时，也发送“完成作业 ID 通知”。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字

完成作业 ID 通知回复

消息 ID	消息名	说明
0x00010008	完成作业 ID 通知回复	回复从 SC-20 发送的“完成作业 ID 通知”的消息。 通过接收“完成作业 ID 通知回复”完成一系列操作。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

作业项目列表数据通知

消息 ID	消息名	说明
0x10010009	作业项目列表数据通知	对“获取作业项目列表请求”通知作业 ID、作业指示、作业项目信息的消息。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	作业 ID			
0x0090	作业指示			
0x00d0	作业项目			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050 - 0x008f	64 字节	char	作业 ID	最多 50 个半角英文字母和数字
0x0090 - 0x00cf	64 字节	char	作业指示	最多 50 个半角英文字母和数字
0x00d0 - 0x010f	64 字节	char	作业项目	最多 50 个半角英文字母和数字

完成获取作业项目列表通知

消息 ID	消息名	说明
0x1001000B	完成获取作业项目列表通知	完成所有“作业项目列表数据通知”发送后向对方机器发送的消息。 在传输个数字段中输入已发送的项目数量。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		传输个数	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	传输个数	1-32767：总项目数 -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

完成获取作业项目列表通知回复

消息 ID	消息名	说明
0x0001000B	完成获取作业项目列表通知回复	回复从 SC-20 发送的“完成获取作业项目列表通知”的消息。 通过“完成获取作业项目列表通知回复”完成一系列操作。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字

系统停止通知

消息 ID	消息名	说明
0x1001000E	系统停止通知	关闭或重新启动 SC-20 后向对方机器侧发送的消息。 没有对该消息进行回复。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	停止模式			

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	4 字节	uint32	停止模式	0：关闭 1：重新启动

超时通知

消息 ID	消息名	说明
0x1001000F	超时通知	SC-20 内部由于某种原因在消息处理不完全时向对方机器侧发送的消息。 没有对该消息进行回复。

消息格式

地址	位			
	31	16	15	0
0x0000	消息 ID			
0x0004	终端 ID			
0x0008	终端名			
0x0048	日	月	年	
0x004c	预留	秒	分	时
0x0050	错误代码		结果	

地址	大小	属性	字段名称	说明
0x0000	4 字节	uint32	消息 ID	消息固有的 ID
0x0004	4 字节	uint32	终端 ID	机身固有的 ID
0x0008 - 0x0047	64 字节	char	终端名	最多 50 个半角英文字母和数字
0x0048	2 字节	uint16	年	设定 SC-20 的时间
0x004a	1 字节	uchar	月	在 1～12 的范围内设定 SC-20 的时间
0x004b	1 字节	uchar	日	在 1～31 的范围内设定 SC-20 的时间
0x004c	1 字节	uchar	时	在 0～23 的范围内设定 SC-20 的时间
0x004d	1 字节	uchar	分	在 0～59 的范围内设定 SC-20 的时间
0x004e	1 字节	uchar	秒	在 0～59 的范围内设定 SC-20 的时间
0x004f	1 字节	uchar	预留	未使用区域
0x0050	2 字节	int16	结果	0：OK -1：NG
0x0052	2 字节	uint16	错误代码	参照 4.错误代码（P.43）

4. 错误代码

错误代码	错误名	原因	对象消息	解决方法
1	未知终端 ID	发送至 SC-20 的终端 ID 与设定的终端 ID 不一致	全体请求消息	请在 SC-20 上确认是否与发送消息的终端 ID、终端名一致。 请确认终端名的大写小写字符是否错误。
2	未知终端名	发送至 SC-20 的终端名与设定的终端名不一致	全体请求消息	
101	状态转换失败	待机以外的状态下收到开始作业 ID 请求	开始作业 ID 请求	“开始作业 ID 请求”和“执行作业 ID 请求”仅在登录后或完成作业 ID 后受理。 可能在匹配过程中或“完成作业 ID 回复”前发送。
102		待机以外的状态下收到执行作业 ID 请求	执行作业 ID 请求	
103	状态转换失败	准备以外的状态下收到开始请求	开始请求	开始请求仅在“开始作业 ID 请求”后或“完成作业项目回复”后受理。作业 ID 正在执行时不受理。
104	状态转换失败	执行以外的状态下收到停止请求	停止请求	停止请求仅在匹配过程中受理。若在停止请求前完成匹配，也将显示该错误代码。
105	状态转换失败	待机以外的状态下收到获取项目请求	作业项目 获取请求	获取项目请求仅在登录后或完成作业 ID 后受理。 可能在匹配过程中或“完成作业 ID 回复”前发送。
106	状态转换失败	操作员模式下收到获取项目请求	作业项目 获取请求	获取项目请求仅可在管理员模式下发出请求。请以管理员模式登录。
107	状态转换失败	待机以外的状态下收到更改作业 ID 请求	更改作业 ID 请求	获取项目请求仅在登录后或完成作业 ID 后受理。 可能在匹配过程中或“完成作业 ID 回复”前发送。
108	输入 EXTIN	在匹配过程中以外的状态下收到输入 EXTIN 请求	输入 EXTIN 请求	输入 EXTIN 请求仅在匹配过程中受理。
109	正在注销	在用户注销时收到请求消息	全体请求消息	请登录。
201	作业 ID 名不一致	指定的作业 ID 不存在	开始请求 执行作业 ID 请求	请求中输入的作业 ID、指示列表、项目不一致。 请在 SC-20 上确认作业名是否被更改。 请确认开头或末尾是否输入了多余的空格等。
202	作业指示 列表名不一致	指定的作业指示不存在	开始请求 执行作业 ID 请求	
203	作业项目名称不一致	指定的作业项目不存在	开始请求 执行作业 ID 请求	

错误代码	错误名	原因	对象消息	解决方法
204	作业 ID 名空白	作业 ID 名未指定	开始请求 执行作业 ID 请求	请在指定格式中输入作业 ID 名称。
207	忙碌状态	相机正在处理，开始失败	开始作业 ID 请求	请稍后重新请求。
208	忙碌状态	相机正在处理，更改失败	更改作业 ID 请求	请稍后重新请求。
209	忙碌状态	针对连续请求，相机发生超时	开始作业 ID 请求、 执行作业 ID 请求	请稍后重新请求。
210	Extin Input	输入的 I/O 不正确或匹配停止	输入 Extin 请求	请在匹配画面中输入正常范围内的值。
301	匹配结果生成失败	匹配结果的生成失败	完成作业项目通知	匹配数据的生成失败时显示。
401	超时	无法接收对通知的回复	超时通知	发送“开始作业 ID 回复”、“完成作业项目通知”、“完成作业 ID 通知”后，若 3 秒内无回复时显示。
550	连接错误 NomachIPError	接收到来自未知 IP 地址的数据。	全体消息	请从设定的 IP 地址发送。

5. 示例代码

基于 C 语言的示例代码如下所示，该代码假定作业 ID“Default”、列表“Work_1”、项目“Item_1”已在 SC-20 主机中登记为匹配模式，并记载了执行作业 ID 1 路径的示例（P.12 执行作业 ID 处理）。
此外，由于此代码基于 Windows 使用了 winsock2，所以需要链接“ws2_32.lib”库。通过安装 Visual Studio 等工具，可以获得该库。

客户端模式

```
#include <stdio.h>
#include <winsock2.h>       // need ws2_32.lib
#include <ws2tcpip.h>

// char* -> short
short char2short(char* c)
{
    short ret = 0;
    ret |= ((short)c[0] & 0x00FF);
    ret |= (((short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> unsigned short
unsigned short char2ushort(char* c)
{
    unsigned short ret = 0;
    ret |= ((unsigned short)c[0] & 0x00FF);
    ret |= (((unsigned short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> double
double char2double(char* c)
{
    double ret = 0.0;
    memcpy(&ret, c, 8);
    return ret;
}

// main
int main()
{
    int ret = 0;
    unsigned short port = 56109; // IP Port Number (Default=56109)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[]= "SC20"; // Terminal Name (Default="SC20")
    SOCKET srcSocket;  // My PC Socket
    SOCKET dstSocket;  // SC20 Socket

    // sockaddr_in
    struct sockaddr_in srcAddr;
    struct sockaddr_in dstAddr;
    int dstAddrSize = sizeof(dstAddr);
```

```
// buffer
char rcvBuffer[4096];
char tmpBuffer[128];
char sndBuffer1[0x0188];
char sndBuffer2[0x004c];

// Windows Only Process
WSADATA data;
WSAStartup(MAKEWORD(2,0), &data);

// sockaddr_in Setting
memset(&srcAddr, 0, sizeof(srcAddr));
srcAddr.sin_port = htons(port);
srcAddr.sin_family = AF_INET;
srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);

// My PC Socket Create
srcSocket = socket(AF_INET, SOCK_STREAM, 0);

// My PC Socket Bind
bind(srcSocket, (struct sockaddr *) &srcAddr, sizeof(srcAddr));

// Connect Listen
listen(srcSocket, 1);

// Connect Accept <- SC20
printf("Waiting for connection ...¥n");
dstSocket = accept(srcSocket, (struct sockaddr *) &dstAddr, &dstAddrSize);
printf("Connected from %s¥n¥n", inet_ntoa(dstAddr.sin_addr));

////////////////////////////////////////
// Work ID Execute Command
////////////////////////////////////////

// 0x00000005 Request Event Buffer
memset(sndBuffer1, 0, 0x0188);
sndBuffer1[0x0000] = 0x05;
sndBuffer1[0x0001] = 0x00;
sndBuffer1[0x0002] = 0x00;
sndBuffer1[0x0003] = 0x00;
sndBuffer1[0x0004] = (char)((termID & 0x000000FF) >>  0);
sndBuffer1[0x0005] = (char)((termID & 0x0000FF00) >>  8);
sndBuffer1[0x0006] = (char)((termID & 0x00FF0000) >> 16);
sndBuffer1[0x0007] = (char)((termID & 0xFF000000) >> 24);
sprintf(&sndBuffer1[0x0008], termName);
sprintf(&sndBuffer1[0x0048], "Default");
sprintf(&sndBuffer1[0x0088], "Work_1");
sprintf(&sndBuffer1[0x00C8], "Item_1");
sprintf(&sndBuffer1[0x0108], "User");
sprintf(&sndBuffer1[0x0148], "1234567890");

// 0x00000005 Request Send -> SC-20
ret = send(dstSocket, sndBuffer1, 0x0188, 0);
if(0 > ret){
    printf("Socket Send Error¥n");
    return -1;
}
```

```
    // 0x10000005 Response Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
    }

    // Response  Buffur Output
    printf("¥n-------------------------------------------------------------¥n");
    printf("@ Work ID Execute Command Response¥n");
    printf("-------------------------------------------------------------¥n¥n");
    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName  : %s¥n", tmpBuffer);
    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                        , rcvBuffer[0x4A]
                                                        , rcvBuffer[0x4B]
                                                        , rcvBuffer[0x4C]
                                                        , rcvBuffer[0x4D]
                                                        , rcvBuffer[0x4E]
    );
    short result = ((short)rcvBuffer[0x51] << 8) | (short)rcvBuffer[0x50];
    printf("Result    : %d¥n", result);
    printf("ErrorCode : %d¥n", (((unsigned short)rcvBuffer[0x53] & 0xFF) << 8) | ((unsigned short)rcvBuffer[0x52] &
0xFF));
    if(0 > result){
        printf("0x00000005 Command Error¥n");
        return -1;
    }

    ////////////////////////////////////
    // Work Item Done (Matching) Notify
    ////////////////////////////////////

    // 0x10010002 Notify Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
    }

    // Work Item Done (Matching) Buffer Output
    printf("¥n-------------------------------------------------------------¥n");
    printf("@ Work Item Done (Matching) Notify¥n");
    printf("-------------------------------------------------------------¥n¥n");
    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName  : %s¥n", tmpBuffer);
    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                        , rcvBuffer[0x4A]
                                                        , rcvBuffer[0x4B]
                                                        , rcvBuffer[0x4C]
```

```
                                                    , rcvBuffer[0x4D]
                                                    , rcvBuffer[0x4E]
    );
    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
    printf("WorkID    : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x0090]);
    printf("WorkList  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x00D0]);
    printf("WorkItem  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x0110]);
    printf("WorkerID  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x01D8]);
    printf("WorkNum   : %s¥n", tmpBuffer);
    printf("Result    : %d¥n", char2short(&rcvBuffer[0x02A0]));
    printf("Time      : %d¥n", char2ushort(&rcvBuffer[0x02A2]));
    printf("BaseScore : %f¥n", char2double(&rcvBuffer[0x02A4]));
    printf("BaseAngle : %d¥n", char2short(&rcvBuffer[0x02AC]));
    int checkNum = (int)char2short(&rcvBuffer[0x02AE]);
    printf("CheckNum  : %d¥n", checkNum);
    for(int i = 0; i < checkNum; i++){
        int offset = i * 16;
        printf("CheckIndex[%d]¥n", rcvBuffer[0x02B0 + offset]);

        printf("  Mode    : %d¥n", rcvBuffer[0x02B1 + offset]);
        printf("  Result  : %d¥n", char2short(&rcvBuffer[0x02B2]));
        printf("  Angle   : %d¥n", char2short(&rcvBuffer[0x02B4]));
        printf("  Time    : %d¥n", char2ushort(&rcvBuffer[0x02B6]));
        printf("  Score   : %f¥n", char2double(&rcvBuffer[0x02B8]));
    }

    // 0x00010007 Work Item Done (Matching) Notify Response Send -> SC-20
    memset(sndBuffer2, 0, 0x004c);
    sndBuffer2[0x0000] = 0x07;
    sndBuffer2[0x0001] = 0x00;
    sndBuffer2[0x0002] = 0x01;
    sndBuffer2[0x0003] = 0x00;
    sndBuffer2[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer2[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer2[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer2[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer2[0x0008], termName);
    ret = send(dstSocket, sndBuffer2, 0x004c, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
        return -1;
    }

    ////////////////////////////////////////
    // Work ID Done Notify
    ////////////////////////////////////////

    // 0x10010008 Notify Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
    }
```

```
    int len = 0;
    printf("¥n----------------------------------------------------------¥n");
    printf("@ Work ID Done Notify¥n");
    printf("----------------------------------------------------------¥n¥n");
    printf("MsgID      : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID     : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName   : %s¥n", tmpBuffer);
    printf("Date       : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                          , rcvBuffer[0x4A]
                                                          , rcvBuffer[0x4B]
                                                          , rcvBuffer[0x4C]
                                                          , rcvBuffer[0x4D]
                                                          , rcvBuffer[0x4E]
    );
    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
    printf("WorkID     : %s¥n", tmpBuffer);

    // 0x00010008 Work ID Done Notify Response Send -> SC-20
    memset(sndBuffer2, 0, 0x004C);
    sndBuffer2[0x0000] = 0x08;
    sndBuffer2[0x0001] = 0x00;
    sndBuffer2[0x0002] = 0x01;
    sndBuffer2[0x0003] = 0x00;
    sndBuffer2[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer2[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer2[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer2[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer2[0x0008], termName);
    ret = send(dstSocket, sndBuffer2, 0x0048, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
        return -1;
    }

    // Close Scocket
    closesocket(dstSocket);
    closesocket(srcSocket);

    // Windows Only
    WSACleanup();

    return 0;
}
```

客户端/服务器模式

```
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <thread>

#pragma warning(disable:4996)

bool commandReady = false;

// char* -> short
short char2short(char* c)
{
    short ret = 0;
    ret |= ((short)c[0] & 0x00FF);
    ret |= (((short)c[1] << 8) & 0xFF00);
    return ret;
}


// char* -> unsigned short
unsigned short char2ushort(char* c)
{
    unsigned short ret = 0;
    ret |= ((unsigned short)c[0] & 0x00FF);
    ret |= (((unsigned short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> double
double char2double(char* c)
{
    double ret = 0.0;
    memcpy(&ret, c, 8);
    return ret;
}

// send function (Client)
int sendFunction(unsigned int reqID)
{
    int ret = 0;
    unsigned short port = 56109; // IP Port Number (Server/Client Mode SC20's IP Port = 56109(fixed))
    const char ipAddr[] = "192.168.1.8"; // IP Address String (depend SC20's Setting)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[] = "SC20"; // Terminal Name (Default="SC20")
    char sndBuffer[0x0188];
    int sndBufferSize;

    SOCKET mySocket;  // My PC Socket

    // My PC Socket Create
    mySocket = socket(AF_INET, SOCK_STREAM, 0);

    // My PC Client Setting
    struct sockaddr_in myAddr;
```

```
memset(&myAddr, 0, sizeof(myAddr));
myAddr.sin_port = htons(port);
myAddr.sin_family = AF_INET;
myAddr.sin_addr.s_addr = inet_addr(ipAddr);

// Connect To SC20
connect(mySocket, (struct sockaddr*) &myAddr, sizeof(myAddr));

// Send Buffer Setting (depend Reqest ID)
switch(reqID){

// Work ID Execute Command Buffer Sample
memset(sndBuffer, 0, 0x0188);
case 0x00000005:
    sndBufferSize = 0x0188;
    sndBuffer[0x0000] = 0x05;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x00;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
    sprintf(&sndBuffer[0x0048], "Default");
    sprintf(&sndBuffer[0x0088], "Work_1");
    sprintf(&sndBuffer[0x00C8], "Item_1");
    sprintf(&sndBuffer[0x0108], "User");
    sprintf(&sndBuffer[0x0148], "1234567890");
    break;

// Work Item Done (Matching) Notify Response Send -> SC-20 Sample
case 0x00010007:
    sndBufferSize = 0x004C;
    sndBuffer[0x0000] = 0x07;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x01;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
    break;

// 0x00010008 Work ID Done Notify Response Send -> SC-20 Sample
case 0x00010008:
    sndBufferSize = 0x004C;
    sndBuffer[0x0000] = 0x08;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x01;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
```

```
        break;
    }

    // Send Buffer -> SC20
    ret = send(mySocket, sndBuffer, sndBufferSize, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
    }

    closesocket(mySocket);
    return ret;
}

// receive thread (Server)
void receiveThread(void)
{
    unsigned short port = 56109; // IP Port Number (Default=56109)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[] = "SC20"; // Terminal Name (Default="SC20")
    SOCKET srcSocket;  // My PC Socket
    SOCKET dstSocket;  // SC20 Socket

    // sockaddr_in
    struct sockaddr_in srcAddr;
    struct sockaddr_in dstAddr;
    int dstAddrSize = sizeof(dstAddr);

    // buffer
    char rcvBuffer[4096];
    char tmpBuffer[4096];

    // sockaddr_in Setting
    memset(&srcAddr, 0, sizeof(srcAddr));
    srcAddr.sin_port = htons(port);
    srcAddr.sin_family = AF_INET;
    srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    // My PC Socket Create
    srcSocket = socket(AF_INET, SOCK_STREAM, 0);

    // My PC Socket Bind
    bind(srcSocket, (struct sockaddr *) &srcAddr, sizeof(srcAddr));

    // Receive Loop
    while(1){

        // Connect Listen
        listen(srcSocket, 1);

        // Connect Accept <- SC20
        printf("Waiting for connection ...¥n");
        dstSocket = accept(srcSocket, (struct sockaddr *) &dstAddr, &dstAddrSize);
        printf("Connected from %s¥n¥n", inet_ntoa(dstAddr.sin_addr));

        // Receive Buffer <- SC20
        int ret = recv(dstSocket, rcvBuffer, 4096, 0);
        if(0 >= ret){
```

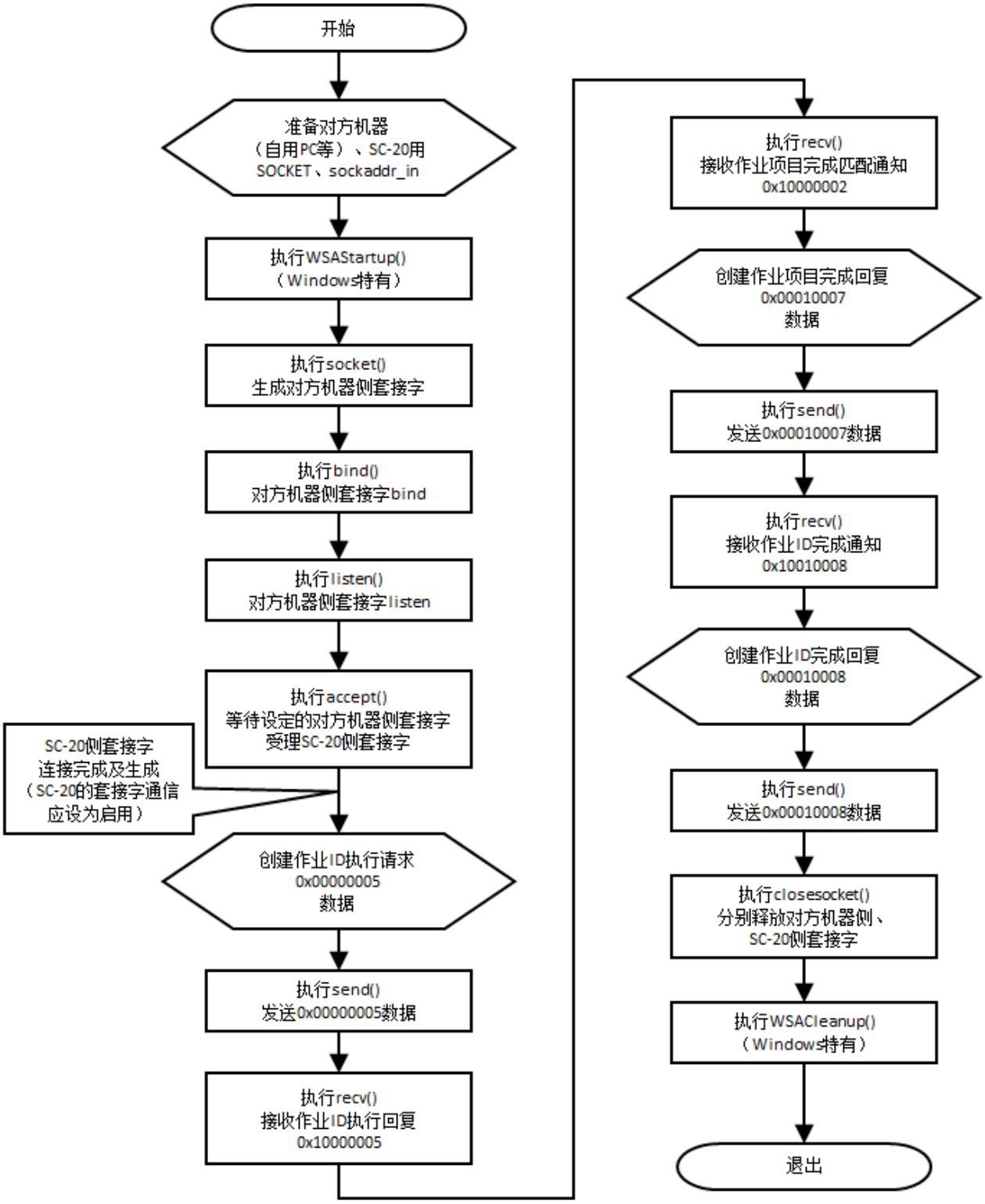
```
            printf("Socket Recieve Error¥n");
            break;
        }
        else{
            // resID Check
            unsigned int resID = ((unsigned int)rcvBuffer[3] << 24) | ((unsigned int)rcvBuffer[2] << 16) | ((unsigned
int)rcvBuffer[1] << 8) | ((unsigned int)rcvBuffer[0]);
            switch(resID){

            // Work ID Execute Command Response
            case 0x10000005:
                {
                    printf("¥n-----------------------------------------------------------¥n");
                    printf("@ Work ID Execute Command Response¥n");
                    printf("-----------------------------------------------------------¥n¥n");
                    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                    strcpy(tmpBuffer, &rcvBuffer[8]);
                    printf("TermName  : %s¥n", tmpBuffer);
                    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                        , rcvBuffer[0x4A]
                                                                        , rcvBuffer[0x4B]
                                                                        , rcvBuffer[0x4C]
                                                                        , rcvBuffer[0x4D]
                                                                        , rcvBuffer[0x4E]
                    );
                    short result = ((short)rcvBuffer[0x51] << 8) | (short)rcvBuffer[0x50];
                    printf("Result    : %d¥n", result);
                    printf("ErrorCode : %d¥n", (((unsigned short)rcvBuffer[0x53] & 0xFF) << 8) | ((unsigned
short)rcvBuffer[0x52] & 0xFF));
                    break;
                }

            // Work Item Done (Matching) Notify
            case 0x10010002:
                {
                    printf("¥n-----------------------------------------------------------¥n");
                    printf("@ Work Item Done (Matching) Notify¥n");
                    printf("-----------------------------------------------------------¥n¥n");
                    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                    strcpy(tmpBuffer, &rcvBuffer[8]);
                    printf("TermName  : %s¥n", tmpBuffer);
                    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                        , rcvBuffer[0x4A]
                                                                        , rcvBuffer[0x4B]
                                                                        , rcvBuffer[0x4C]
                                                                        , rcvBuffer[0x4D]
                                                                        , rcvBuffer[0x4E]
                    );
                    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
                    printf("WorkID    : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x0090]);
                    printf("WorkList  : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x00D0]);
```

```
                printf("WorkItem  : %s¥n", tmpBuffer);
                strcpy(tmpBuffer, &rcvBuffer[0x0110]);
                printf("WorkerID  : %s¥n", tmpBuffer);
                strcpy(tmpBuffer, &rcvBuffer[0x01D8]);
                printf("WorkNum   : %s¥n", tmpBuffer);
                printf("Result    : %d¥n", char2short(&rcvBuffer[0x02A0]));
                printf("Time      : %d¥n", char2ushort(&rcvBuffer[0x02A2]));
                printf("BaseScore : %f¥n", char2double(&rcvBuffer[0x02A4]));
                printf("BaseAngle : %d¥n", char2short(&rcvBuffer[0x02AC]));
                int checkNum = (int)char2short(&rcvBuffer[0x02AE]);
                printf("CheckNum  : %d¥n", checkNum);
                for(int i = 0; i < checkNum; i++){
                    int offset = i * 16;
                    printf("CheckIndex[%d]¥n", rcvBuffer[0x02B0 + offset]);
                    printf("  Mode    : %d¥n", rcvBuffer[0x02B1 + offset]);
                    printf("  Result  : %d¥n", char2short(&rcvBuffer[0x02B2]));
                    printf("  Angle   : %d¥n", char2short(&rcvBuffer[0x02B4]));
                    printf("  Time    : %d¥n", char2ushort(&rcvBuffer[0x02B6]));
                    printf("  Score   : %f¥n", char2double(&rcvBuffer[0x02B8]));
                }

                // Work Item Done (Matching) Notify Response Send -> SC20
                sendFunction(0x00010007);

                break;
            }

        // Work ID Done Notify
        case 0x10010008:
            {
                printf("¥n-------------------------------------------------------¥n");
                printf("@ Work ID Done Notify¥n");
                printf("-------------------------------------------------------¥n¥n");
                printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                strcpy(tmpBuffer, &rcvBuffer[8]);
                printf("TermName  : %s¥n", tmpBuffer);
                printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                    , rcvBuffer[0x4A]
                                                                    , rcvBuffer[0x4B]
                                                                    , rcvBuffer[0x4C]
                                                                    , rcvBuffer[0x4D]
                                                                    , rcvBuffer[0x4E]
                );
                strcpy(tmpBuffer, &rcvBuffer[0x0050]);
                printf("WorkID    : %s¥n", tmpBuffer);

                // Work ID Done Notify Response Send -> SC20
                sendFunction(0x00010008);

                Sleep(3000);
                commandReady = true;
                break;
            }
        }
    }
```

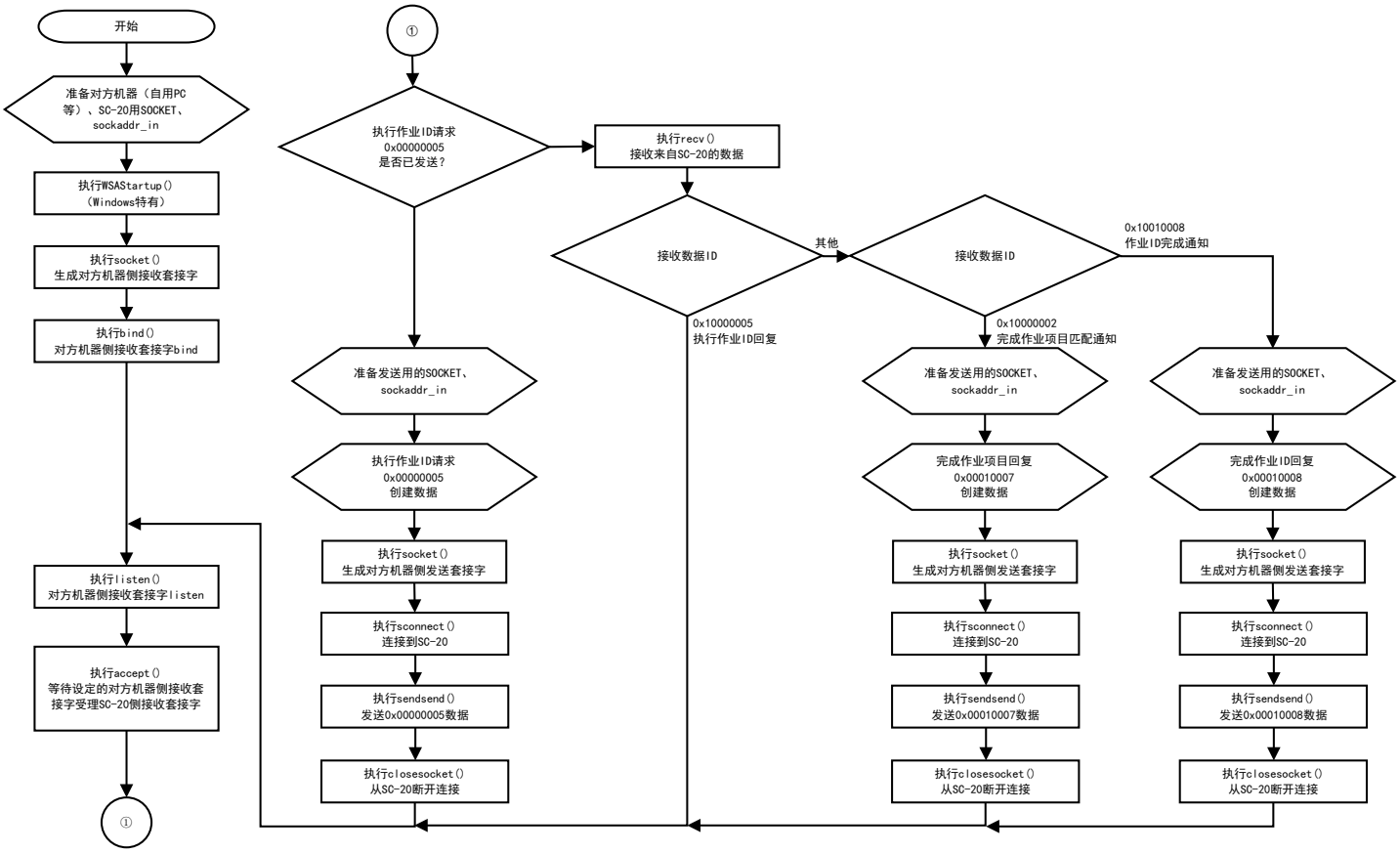
```
        // Close Scocket(dst)
        closesocket(dstSocket);
    }
    // Close Scocket(src)
    closesocket(srcSocket);
}

// main
int main()
{
    // Windows Only Process
    WSADATA data;
    WSAStartup(MAKEWORD(2,0), &data);

    // Receive Thread Start (My PC's Server)
    std::thread rcvThred(receiveThread);
    rcvThred.detach();

    // Thread Ready Wait
    Sleep(3000);
    commandReady = true;

    // Work ID Execute Command Trigger
    while(1){
        if(true == commandReady){
            printf("¥n¥n>>>> Put Enter¥n");
            getchar();
            sendFunction(0x00000005);
            commandReady = false;
        }
        else{
            Sleep(1000);
        }
    }

    // Windows Only
    WSACleanup();

    return 0;
}
```

6. 流程图

客户端模式


开始
准备对方机器
（自用PC等）、SC-20用
SOCKET、sockaddr_in
执行WSAStartup()
（Windows特有）
执行socket()
生成对方机器侧套接字
执行bind()
对方机器侧套接字bind
执行listen()
对方机器侧套接字listen
执行accept()
等待设定的对方机器侧套接字
受理SC-20侧套接字
SC-20侧套接字
连接完成及生成
（SC-20的套接字通信
应设为启用）
创建作业ID执行请求
0x00000005
数据
执行send()
发送0x00000005数据
执行recv()
接收作业ID执行回复
0x10000005
执行recv()
接收作业项目完成匹配通知
0x10000002
创建作业项目完成回复
0x00010007
数据
执行send()
发送0x00010007数据
执行recv()
接收作业ID完成通知
0x10010008
创建作业ID完成回复
0x00010008
数据
执行send()
发送0x00010008数据
执行closesocket()
分别释放对方机器侧、
SC-20侧套接字
执行WSACleanup()
（Windows特有）
退出

客户端/服务器模式


开始
准备对方机器（自用PC等）、SC-20用SOCKET、sockaddr_in
执行WSAStartup()（Windows特有）
执行socket()生成对方机器侧接收套接字
执行bind()对方机器侧接收套接字bind
执行listen()对方机器侧接收套接字listen
执行accept()等待设定的对方机器侧接收套接字受理SC-20侧接收套接字
①
①
执行作业ID请求0x00000005是否已发送？
准备发送用的SOCKET、sockaddr_in
执行作业ID请求0x00000005创建数据
执行socket()生成对方机器侧发送套接字
执行sconnect()连接到SC-20
执行sendsend()发送0x00000005数据
执行closesocket()从SC-20断开连接
执行recv()接收来自SC-20的数据
接收数据ID
0x10000005执行作业ID回复
其他
接收数据ID
0x10000002完成作业项目匹配通知
准备发送用的SOCKET、sockaddr_in
完成作业项目回复0x00010007创建数据
执行socket()生成对方机器侧发送套接字
执行sconnect()连接到SC-20
执行sendsend()发送0x00010007数据
执行closesocket()从SC-20断开连接
0x10010008作业ID完成通知
准备发送用的SOCKET、sockaddr_in
完成作业ID回复0x00010008创建数据
执行socket()生成对方机器侧发送套接字
执行sconnect()连接到SC-20
执行sendsend()发送0x00010008数据
执行closesocket()从SC-20断开连接

修订历史记录

版本	制作日期	修订条款	备考
1.0	2023/6/14	・初次发行	
2.0	2024/3/28	・追加客户端/服务器模式相关内容 ・删除了记述了通知类中不存在的 ID ・修改其他错误	