1510, TFDCN-127-003, 001

NACHI

FD 控制装置
操作说明书
套接字(socket)通信

第 3 版

• 在使用机器人之前，请详读本操作说明书，并请遵从所有关于安全事项与正文的指示。
• 关于本机器人的安装、操作、维修，请仅由接受过本公司机器人讲习的人员进行。
• 在使用本机器人的时候，必须遵守各个国家有关工业机器人的法律以及安全相关的法律条例。
• 务必将本操作说明书交付给实际操作的人员。
• 有 关本操作说明书的不明之处以及有关本机器人的售后服务，请向记载在封底中的敝社的服务中心查询。

株式会社 不二越

目　　录

第 1 章 概要

第 2 章 创建通信程序

第 3 章 套接字功能

第 4 章 程序创建实例

第 5 章 PC 端程序实例

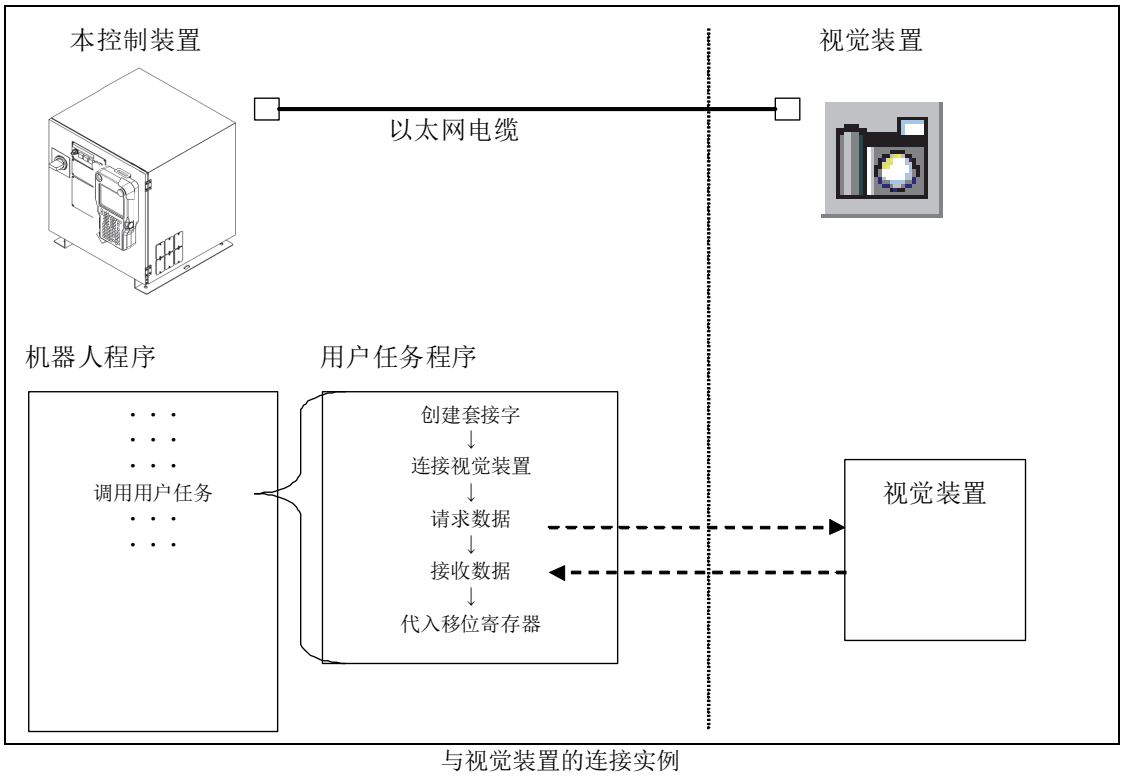
第1章 概要

1.1 概要

1.1.1 机器人语言 套接字(Socket)界面功能

本功能是通过用户任务程序使用各种应用指令进行以太网通信的功能。创建进行以太网通信的用户任务程序，便可以通过以太网从控制装置外部执行各种数据的参照和重写操作。

例）
1 通过个人计算机监控控制装置的整数变量
2 通过视觉装置等更改移位寄存器的值
3 通过个人计算机监控采用 SYSTEM 函数获取的机器人或控制装置的状态。



与视觉装置的连接实例

重要

只能通过用户任务程序使用套接字界面功能。无法通过作业程序使用该功能。
需要通过作业程序使用通信功能时，可以使用［FN671 CALLMCR］从作业程序中调用用户任务程序。

重点

机器人语言相关内容可以参见操作说明书“机器人语言”，用户任务相关内容可以参见操作说明书“用户任务”。

重点

一般来说，使用普通套接字的网络程序和术语等，根据需要，可以参考市售的专业书籍等资料。

1.1.2 所需设备

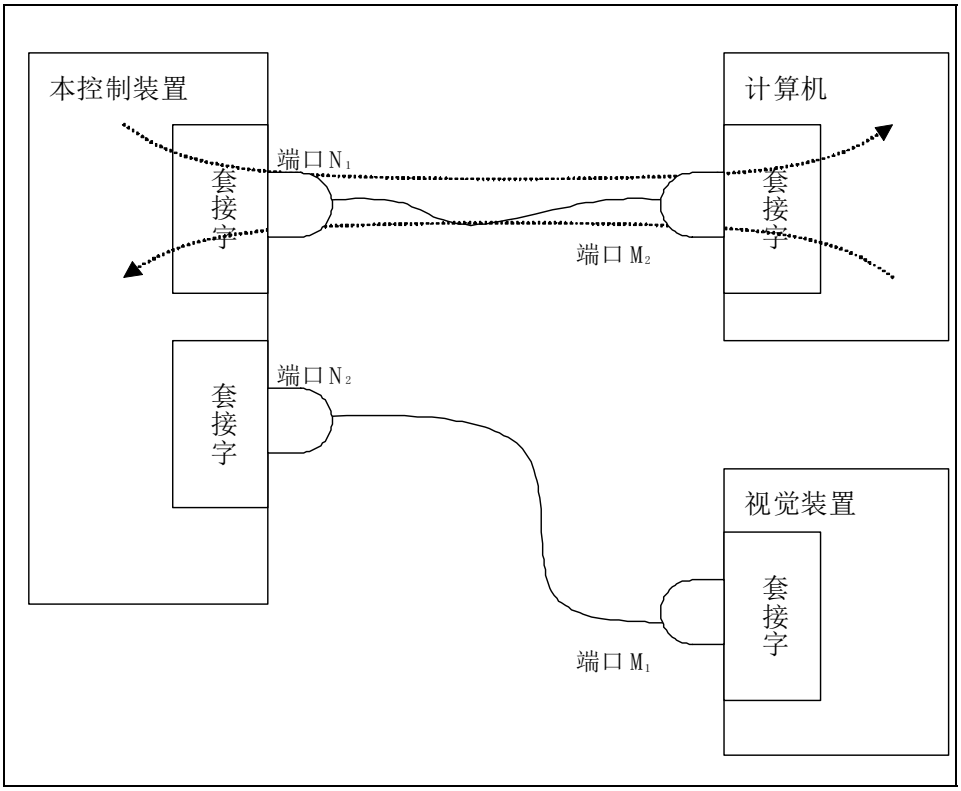
使用本功能需要准备以太网电缆。以太网电缆的连接可以参见操作说明书“设定篇”的以太网相关章节。

1.2 术语

本章将对本书中所使用的通信相关术语进行介绍。

1.2.1 套接字(Socket)

用于创建 TCP/IP 网络程序的函数组。定义网络连接、数据发送、数据接收等函数。一般情况下，利用以太网的网络时，通过套接字进行定义。
本控制装置中，通过将这些函数由机器人语言转换为可使用方式，从而进行以太网通信。



套接字概念图

1.2.2 端口编号

通信设备中用于识别通信服务的编号。由于用于识别，所以无法分配重复的端口编号。因此，本控制装置内已使用的端口编号无法使用。
另外，由于 1-1024 为止的编号被常用服务（FTP，HTTP 等）所占用，因此一般无法使用。

在本控制装置上，初始状态下使用的端口编号为
233，42353，51201

1.2.3 TCP/IP (Transport control protocol / Internet protocol)

通过网络进行通信时，无论以何种形式进行通信，如果通信设备之间互相不能识别，将无法进行通信。TCP/IP 是通信相关协议之一，其综合了常用的以太网通信协议。
通过本功能，可以进行执行了 TCP、UDP 协议的通信。

1.2.4 TCP (Transport control protocol)

TCP 是一种高可靠性的通信协议。
TCP 协议下，为了提高通信的可靠性，会确认通讯伙伴是否收到数据，如果没有收到，会再次发送相同数据，或判断网络的流量，降低数据传输速度。

1.2.5 UDP (User Datagram Protocol)

UDP 是用于高速通信的通信协议。
UDP 协议下，省去了 TCP 协议下对数据传输的确认，由此可以实现高速的通讯处理。
但是，即使通信伙伴没有收到数据，也无法进行检测。

1.2.6 字节序

表示处理数据时数据排列顺序的术语。
通信通常以 1 字节为单位发送数据。由于 1 字节只能表示 0～255，因此如果表示 10000 等大于 255 的数值时，必须按照［10000 = 39×256 + 16］拆分为 39/16，拆分为 1 字节单位后再进行发送。此时，从高字节（39）发送或从低字节（16）发送的排列顺序称之为字节序。
从高字节发送时，称之为大端序，从低字节发送时，称之为小端序。如果在收发数值数据时使用本功能，将以大端序进行发送和接收。

1.2.7 字节

一般情况下，8 位二进制为 1 字节。通信处理的数据大小以 1 字节为最小单位。

1.2.8 服务器/客户端

通信时接受对方的连接要求，提供服务或功能的程序称之为服务器。另外，要求连接服务器的程序称之为客户端。

NOTE

第2章 创建通信程序

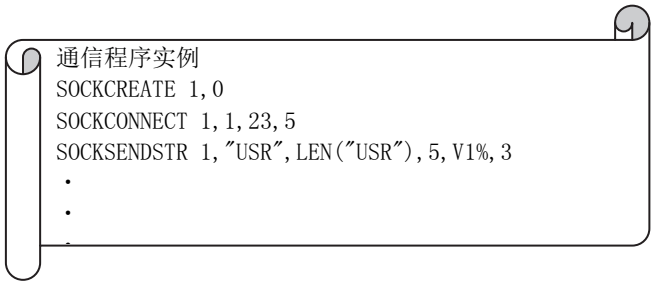
2.1 创建通信程序

2.1.1 TCP/IP 设定

进行以太网通信，需要设定本控制装置的 IP 地址、子网掩码等。请按照［常数设定］→［8 通信］→［2 以太网］→［TCP/IP］的顺序进行设定。
关于 TCP/IP 的设定，请参见操作说明书“设定篇”的以太网相关章节。

2.1.2 创建机器人语言程序

创建机器人语言执行的通信程序。通信功能的使用方法和详情可以参见第 3 章。
另外，机器人语言的语法可以参见操作说明书“机器人语言”。



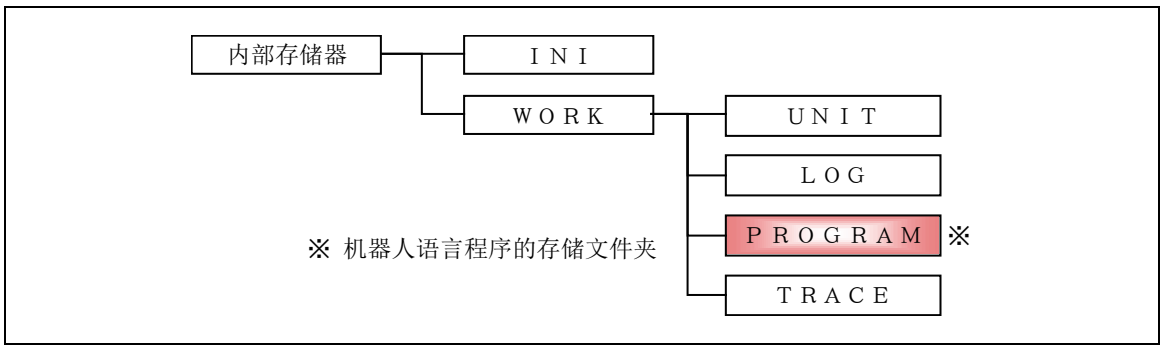
文件名规则	USERTASK-A.***（*** 为程序编号） ※用户任务的程序编号为 000～999。
编辑方法	请参见所使用个人计算机或文本编辑器的说明书。
文件大小	最大可设定为 64KB（65536 字节）。
句、行、字符的注意事项	1 行 254 个字符，每个程序最多可以记述 999 行。 不区分英文大小写。除了批注和字符串变量，其余全部使用半角字符。 可记述空白行。（编译时将忽略这些行）
其他	创建全角通过上档 JIS 执行，创建换行代码通过 CR+LF 执行。

2.1.3 传送至本控制装置 & 编译

为了将使用外部个人计算机创建的机器人语言程序读入本控制装置，请准备 USB 记忆体或连接以太网等。

从 USB 记忆体复制时，按照［维修］→［7 文件操作］→［1 拷贝］的顺序操作。
指定要写入机器人语言程序的文件夹为作业程序所在的 PROGRAM 文件夹。PROGRAM 文件夹以外的机器人语言程序不会成为编辑、编译的对象。
［文件操作］的方法可以参见操作说明书“基本操作篇”。

另外，从以太网传送机器人语言程序时，按照［维修］→［7 文件操作］→［8 文件转送（以太网 FTP）］的顺序操作。
此时，与从 USB 记忆体复制时相同，指定要写入机器人语言程序的文件夹为作业程序所在的 PROGRAM 文件夹。
采用 FTP 的文件传送方法可以参见操作说明书“设定篇”的以太网相关章节。



机器人语言程序的存储文件夹

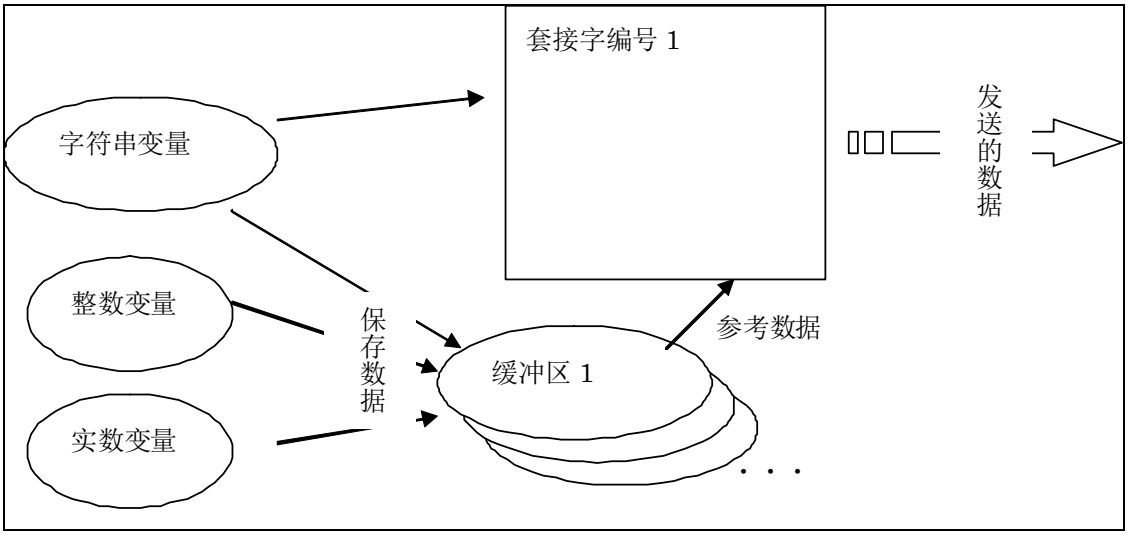
在所显示的画面上，按照［维修］→［9 程序转换］→［8 语言转换］的顺序编译程序。
编译的方法可以参见操作说明书“机器人语言”。

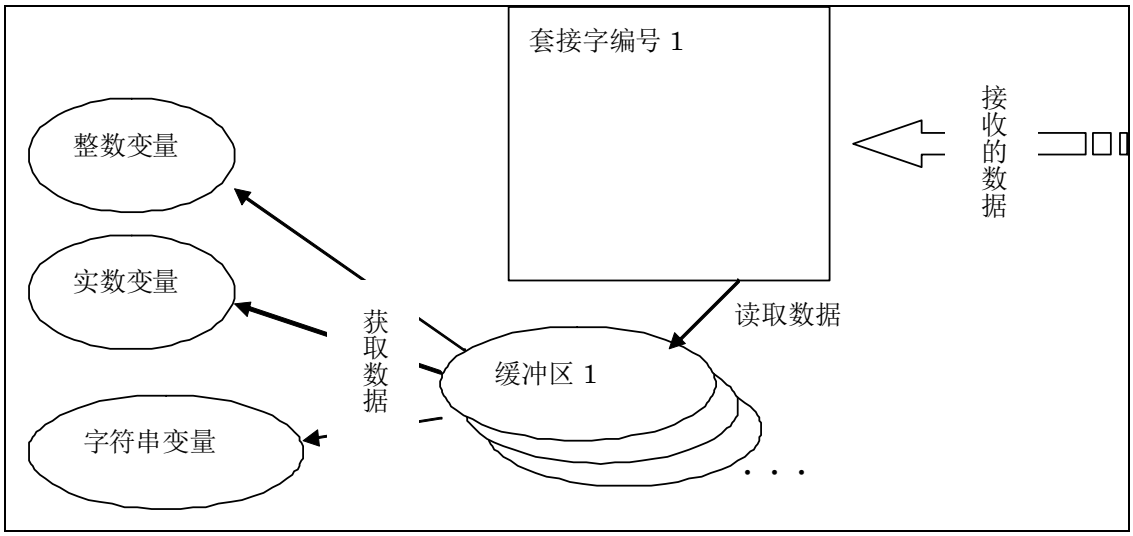
第3章 套接字功能

3.1 套接字功能

3.1.1 概要

套接字(Socket)功能一般通过缓冲区与网络交换数据。与缓冲区交换数据的操作采用专用应用指令。
整个本控制装置中，可以同时定义含有 1–16 号的 16 个套接字。另外，1024 Bytes 大小缓冲区可以使用 1–16 号的 16 个套接字。





发生通信相关错误时，套接字功能向错误变量中写入错误，以完成步骤。在通信相关错误中，无法停止程序再生。
套接字功能步骤完成后，可以观察错误变量，以确认通信状态。

重点

1 个套接字可以使用多个缓冲区。另外，多个套接字也可以使用 1 个缓冲区，但此时需要执行使用 I 等待等的互锁，以避免同时访问。

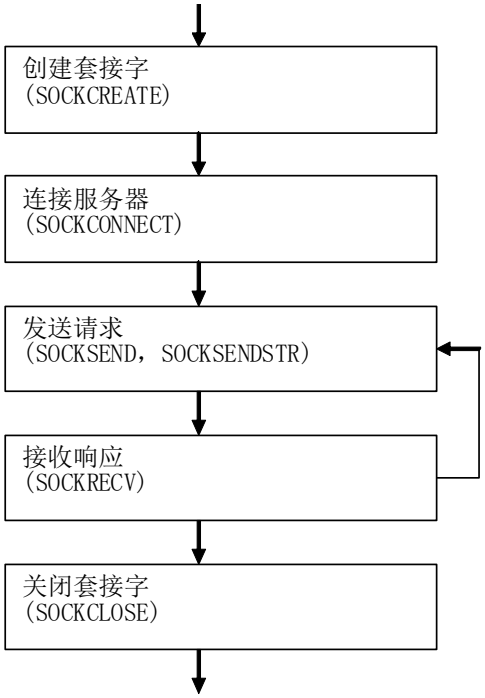
重点

用户任务通常可以是多个用户任务，能够同时启动相同程序，但在套接字通信中，由于无法创建相同编号的套接字，因此无法同时启动。

3.1.2 套接字程序的流程

下面描述套接字(Socket)程序的一般使用步骤。即使是复杂的程序，基本上也沿用这里介绍的流程。

客户端



客户端的套接字程序一般按照左图所示的步骤执行。

1.创建套接字
创建套接字。

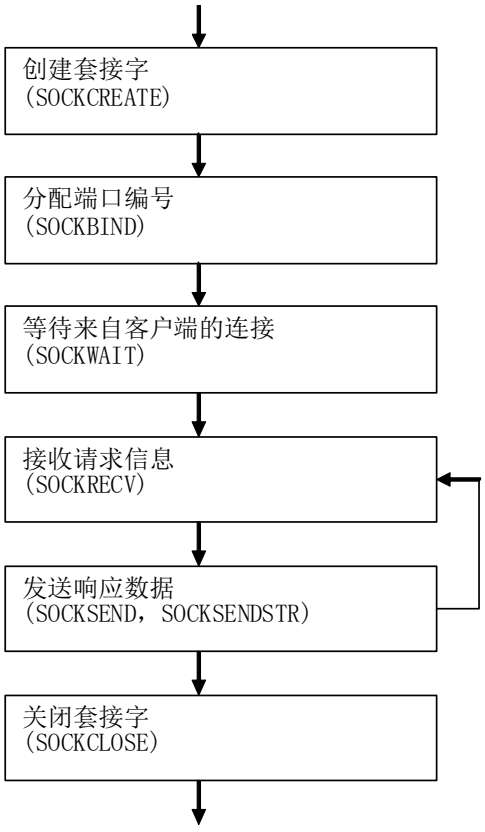
2.连接服务器
连接服务器指定的端口。

3.发送请求
向服务器发送服务（数据回复、登录等）的请求信息。

4.接收响应
接收请求的服务执行状态或请求的数据等。

5.关闭套接字
释放使用过的套接字编号。

服务器



服务器的套接字程序一般按照左图所示的步骤执行。

1.创建套接字
创建套接字。

2.分配端口编号
指定等待连接客户端的端口编号。

3.等待来自客户端的连接
停止程序，直到接收到来自客户端的连接请求。

4.接收请求信息
接收来自客户端的信息。

5.发送响应数据
根据来自客户端的请求，发送响应信息。

4.关闭套接字
释放使用过的套接字编号。

3.1.3 创建套接字

指令名称	SOCKCREATE
FN 代码	570
标题名称	插槽制作（创建套接字）
概要	创建套接字。

概要

创建套接字。使用套接字进行通信时，请在执行其他套接字功能之前，执行这个函数。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为已有套接字编号时，报错。 （1–16）
第 2 参数	TCP/UDP	指定采用 TCP 协议还是采用 UDP 协议来使用已指定的套接字。 如果指定 0，则采用 TCP 协议，如果指定 1，则采用 UDP 协议。 （0–1）

机器人语言书写实例

SOCKCREATE 1，0
创建套接字编号为 1 号的 TCP 协议套接字。

3.1.4 结束套接字

指令名称	SOCKCLOSE
FN 代码	571
标题名称	插槽关闭（关闭套接字）
概要	关闭套接字。

概要

清除通过 SOCKCREATE 创建的套接字，使其形成可以再次使用的状态。不执行 SOCKCLOSE 就结束使用套接字的用户任务程序时，自动执行关闭。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）

机器人语言书写实例

SOCKCLOSE 1
关闭已创建套接字编号为 1 号的套接字。

3.1.5 为套接字分配端口编号

指令名称	SOCKBIND
FN 代码	572
标题名称	套接绑定（套接字绑定）
概要	指定待机端口。

概要

将本控制装置当作服务器使用时，为套接字编号指定等待对方连接的待机端口。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	端口编号	指定服务器动作时的待机端口。指定为已使用的端口编号时，报错。 （1–65535）

机器人语言书写实例

SOCKBIND 1，48952

为已创建套接字编号为 1 号的套接字分配端口编号 48952。

3.1.6 等待连接客户端

指令名称	SOCKWAIT
FN 代码	573
标题名称	接收等待
概要	令已分配的端口编号等待来自外部的连接。

概要

令已分配的端口编号等待来自外部的连接。其间，程序停止。超时后，程序再次启动。
SOCKWAIT 只在采用 TCP 协议时使用。

参数

第 1 参数	等待连接套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	通信套接字编号	指定连接已连接过的客户端的套接字编号。客户端已连接后，以指定的套接字编号创建套接字。因此，通信结束后，需要使用 SOCKCLOSE 函数执行关闭。 （1–16）
第 3 参数	超时时间	以秒为单位设定等待连接的超时。如果指定为 0 时，则持续等待连接，没有超时限制。 （0–20）

画面显示实例

SOCKWAIT 1，2，5

通过套接字编号 1 号的套接字等待客户端的连接，与客户端连接后，执行与套接字编号 2 号的套接字的连接。5 秒以内没有来自客户端的连接时，超时。

3.1.7 连接服务器

指令名称	SOCKCONNECT
FN 代码	574
标题名称	服务器连接
概要	与服务器进行连接。

概要

采用 TCP 协议时，连接服务器。采用 UDP 协议时，不会连接服务器，但需要通过 SOCKCONNECT 设定 IP 地址。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	IP 地址	指定服务器 IP 地址的末尾字节。开头 3 字节使用与本控制装置 TCP/IP 的设定相同的字节。 （1–254）
第 3 参数	端口编号	指定服务器的待机端口。 （1–65535）
第 4 参数	超时时间	以秒为单位设定等待连接的超时。 （0–20）

机器人语言书写实例

SOCKCONNECT 1，101，80，10

使用套接字编号 1 号的套接字执行服务器连接要求。服务器的 IP 地址为将本控制装置设定的 IP 地址最后字节设定为 101 的值，端口编号为 80 号。10 秒内服务器没有响应时，超时。

3.1.8 发送缓冲区中的数据

指令名称	SOCKSEND
FN 代码	575
标题名称	数据发送
概要	发送存储于指定缓冲区的数据。

概要

发送存储于指定缓冲区的数据。未 Connect 时，如果对方已关闭，报错。对方未执行关闭处理就切断时，由于无法立即识别对方切断，因此在执行 SOCKSEND 的时点，有可能不会报错。
发送长度超过缓冲区大小的数据时，请分 2 次发送。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	缓冲区编号	指定缓冲区编号 （1–16）
第 3 参数	发送数据长度	设定以多少字节发送存储于指定缓冲区的数据。 （1–1024）
第 4 参数	超时时间	以秒为单位设定等待连接的超时。指定为 0 时，则持续等待发送完成，没有超时限制。 （0–20）
第 5 参数	整数变量	指定写入已发送数据大小（字节单位）的变量。 实际已发送的数据大小有可能小于以发送数据长度指定的数据大小。

机器人语言书写实例

SOCKSEND 1，1，10，5，V1%

以最大 10 字节将 1 号缓冲区保存的数据发送至通过 1 号套接字连接的通信设备。SOCKSEND 在数据发送后，将实际发送的数据大小写入整数变量 V1%，完成步骤。5 秒以内没有发送时，超时。

3.1.9 发送字符串数据

指令名称	SOCKSENDSTR
FN 代码	576
标题名称	字符串发送
概要	发送已指定的字符串。

概要

发送已指定的字符串。此时，可以附加标识字符串结束的终止符。
未 Connect 时，如果对方已关闭，报错。对方未执行关闭处理就切断时，由于无法立即识别对方切断，因此在执行 SOCKSENDSTR 的时点，有可能不会报错。

参数

第 1 参数	21 套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	字符串	指定要发送的字符串数据。字符串数据可以以字符串变量、字符串常数进行指定。
第 3 参数	发送数据长度	以字节单位指定要发送数据的长度。 （0–199）
第 4 参数	超时时间	以秒为单位设定等待连接的超时。如果指定为 0，则持续等待连接，没有超时限制。 （0–20）
第 5 参数	整数变量	指定写入已发送数据大小（字节单位）的变量。 实际已发送的数据大小有可能小于以发送数据长度指定的数据大小。
第 6 参数	终止字符	指定要发送字符串末尾附加的终止符。指定终止符时，发送数据的长度变长，变长的部分只限终止字符的字符数。省略终止符的指定操作时，执行与指定为 0 时相同的动作。 0：不追加终止字符。 1：字符串末尾追加“\r”后发送。 2：字符串末尾追加“\n”后发送。 3：字符串末尾追加“\r\n”后发送。 （0–3）

机器人语言书写实例

SOCKSENDSTR 1，“ABC”，LEN(“ABC”)，5，V1%

将“ABC”这个字符串发送至使用 1 号套接字连接的通信设备。实际发送的数据大小写入 V1%。5 秒以内没有发送时，超时。这里省略了第 6 参数，省略时等同于指定为 0。

3.1.10 数据接收

指令名称	SOCKRECV
FN 代码	577
标题名称	数据接收
概要	接收数据。

概要

接收数据。根据 TCP 和 UDP 协议的不同，该函数的动作所有变化。

TCP 时，在下面的情况下，SOCKRECV 函数步骤结束。
・实际接收数据的大小大于接受数据的长度。
・通信伙伴关闭了套接字
・超时
接收了小于接收数据长度的数据时，再次等待接收数据。重复数据接收，积累超过接收数据的长度后，函数步骤结束。

UDP 时，在下面的情况下，SOCKRECV 函数步骤结束。
・数据接收
・超时
UDP 协议下，接收数据后，将已接收的数据保存在缓冲区，结束函数步骤。因此，实际接收的数据长度有可能小于已指定的接收数据长度。另外，实际接收的数据长度大于已指定的接收数据长度时，将缩减超出的部分，并传回错误。
另外，数据接收后，通过执行 SOCKSEND 函数，可以回复已发送数据的通信伙伴。数据接收后，有别于已发送数据的通信伙伴，需要向其他通信设备发送数据时，可以使用 SOCKCONNECT 再次设定接入点。

参数

第 1 参数	套接字编号	指定要使用的套接字编号。指定为尚未创建的套接字编号时，报错。 （1–16）
第 2 参数	缓冲区编号	指定保存已接收数据的缓冲区编号。 （1–16）
第 3 参数	接收数据长度	以字节为单位指定要接收的数据大小。 （0–1024）
第 4 参数	超时时间	以秒为单位设定等待连接的超时。指定为 0，则持续等待接受数据，没有超时限制。 （0–20）
第 5 参数	整数变量	指定接收后写入已写入收发缓冲区数据大小（字节单位）的整数变量。实际已接收的数据大小有可能小于以接收数据长度指定的值。

机器人语言书写实例

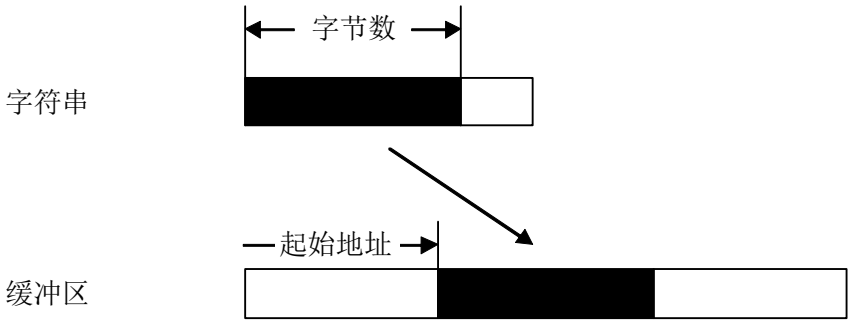
SOCKRECV 1，1，10，5，V1%
接收通过套接字编号 1 号的套接字连接的通信设备发送的数据，将最大 10 字节的数据保存到 1 号缓冲区。将实际接收的数据大小写入 V1%。5 秒以内无法接收时，超时。

3.1.11 将字符串存储到缓冲区

指令名称	SETSTR
FN 代码	580
标题名称	缓冲存储（字符串）
概要	将字符串数据写入缓冲区的任意位置。

概要

将字符串数据写入缓冲区的任意位置。要写入的字符串溢出缓冲区时（起始地址+字节数超过缓冲区大小时），发生［E2250 步骤数据异常］。



参数

第 1 参数	缓冲区编号	指定写入数据的缓冲区编号。 （1–16）
第 2 参数	字符串	指定写入缓冲区的字符串。可以用字符串变量和字符串常数指定字符串。
第 3 参数	开始位置	指定写入缓冲区的写入位置。 （0–1023）
第 4 参数	字节数	指定要写入的数据大小。 （1–199）

机器人语言书写实例

SETSTR 1, V1$, 0, 10

将存储于 V1$ 中字符串的开头 10 字节保有至 1 号缓冲区的开头位置。

3.1.12 将整数存储到缓冲区

指令名称	SETINT
FN 代码	581
标题名称	缓冲存储（整数）
概要	将整数保存到缓冲区的指定位置。

概要

将整数保存到缓冲区的指定位置。整数值以大端序写入 4 字节的带符号整数数据。

参数

第 1 参数	缓冲区编号	指定写入数据的缓冲区编号。 （1–16）
第 2 参数	整数	指定写入缓冲区的整数值。可以以整数常数和整数变量指定整数值。 （–2147483648–2147483647）
第 3 参数	起始地址	指定缓冲区的写入位置。从指定位置写入 4 字节数据。 （0–1020）

机器人语言书写实例

SETINT 1,25,0

将常数 25 保存至 1 号缓冲区开头 4 字节的区域。

3.1.13 将实数存储到缓冲区

指令名称	SETREAL
FN 代码	582
标题名称	缓冲存储（实数）
概要	将实数保存到缓冲区的指定位置。

概要

将实数保存到缓冲区的指定位置。实数值为以大端序写入 4 字节 IEEE754 定义的单精度浮点型表示的数据。

参数

第 1 参数	缓冲区编号	指定写入数据的缓冲区编号。 （1–16）
第 2 参数	实数	指定写入缓冲区的实数值。可以以实数常数和实数变量指定实数值。 （-10^{38}–10^{38}）
第 3 参数	起始地址	指定写入缓冲区的写入位置。从指定的位置写入 4 字节的数据。 （0–1020）

机器人语言书写实例

SETREAL 1，V2!，0

将存储于实数变量 V2!的值保存到 1 号缓冲区开头 4 字节内。

IEEE754 为定义浮点数（实数）表示方法的标准。

3.1.14 将 1 字节的数据存储到缓冲区

指令名称	SETBYTE
FN 代码	583
标题名称	缓冲存储
概要	将 1 字节的数据写入缓冲区的任意位置。

概要

将 1 字节的数据写入缓冲区的任意位置。可以用于将 SETSTR，SETINT，SETREAL 不支持的数据保存到缓冲区的情形。

参数

第 1 参数	缓冲区编号	指定写入数据的缓冲区编号。 （1–16）
第 2 参数	整数	指定写入缓冲区的值。可以用以常数或变量指定该值。 （0–255）
第 3 参数	起始地址	指定写入缓冲区的写入位置。从指定的位置写入 1 字节数据。 （0–1023）

画面显示实例

SETBYTE 1, 12, 3

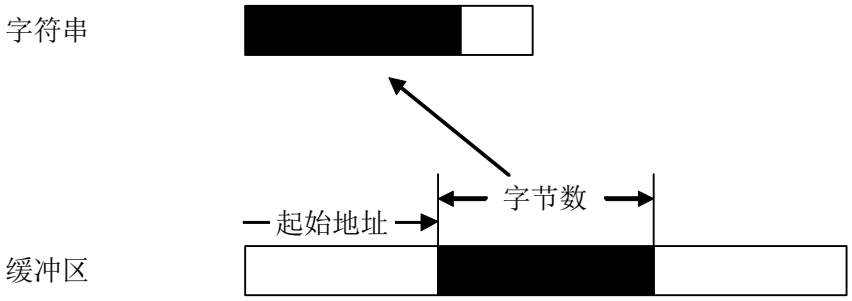
将整数常数 12 保存到 1 号缓冲区的第 3 个字节。

3.1.15 从缓冲区提取字符串

指令名称	GETSTR
FN 代码	584
标题名称	缓冲读取（字符串）
概要	从缓冲区的指定位置获取指定大小的数据。 将此数据保存到任意的字符串变量。

概要

从缓冲区的指定位置获取指定大小的数据，保存到任意的字符串变量。要读取的数据范围超出缓冲区时（起始地址+字节数超过缓冲区大小时），发生［E2250 步骤数据异常］。



参数

第 1 参数	缓冲区编号	指定读取数据的缓冲区编号。 （1–16）
第 2 参数	字符串变量	指定存储读出字符串数据的变量。
第 3 参数	起始地址	指定读取缓冲区的开头位置。 （0–1023）
第 4 参数	字节数	指定要提取字符串的大小。 （1–199）

机器人语言书写实例

GETSTR 1, V1$, 0, 3

将存储于 1 号缓冲区数据开头 3 字节复制到字符串变量 V1$。

3.1.16 从缓冲区提取整数值

指令名称	GETINT
FN 代码	585
标题名称	缓冲读取（整数）
概要	从缓冲区的任意位置读取4字节，保存到任意的整数变量。

概要

从缓冲区的任意位置读取 4 字节，保存到任意的整数变量。使用 GETINT 时，确认缓冲区保存的数据为大端序，且为长度为 4 字节的带符号整数值。其他整数值时，可以使用 GETBYTE 函数。

参数

第 1 参数	缓冲区编号	指定读取数据的缓冲区编号。 （1–16）
第 2 参数	整数变量	指定保存读出值的整数变量。
第 3 参数	起始地址	指定读取缓冲区的开头位置。从指定的位置读取 4 字节数据。 （0–1020）

机器人语言书写实例

GETINT 1, V1%, 10

从 1 号缓冲区保存数据的第 10 字节开始读取 4 字节数据，保存到整数变量 V1%中。

3.1.17 从缓冲区提取实数值

指令名称	GETREAL
FN 代码	586
标题名称	缓冲读取（实数）
概要	从缓冲区的任意位置读取 4 字节数据。 将此数据保存到任意的实数变量。

概要

从缓冲区的任意位置读取 4 字节数据，保存到任意的实数变量。使用 GETREAL 时，确认缓冲区保存的数据为大端序，且长度为 4 字节的 IEEE754 标准实数值。其他实数值可以使用 GETBYTE 函数。

参数

第 1 参数	缓冲区编号	指定读取数据的缓冲区编号。 （1–16）
第 2 参数	实数变量	指定保存读出值的实数变量。
第 3 参数	起始地址	指定读取缓冲区的开头位置。从指定的位置读取 4 字节数据。 （0–1020）

机器人语言书写实例

GETREAL 1, V1!, 3

从 1 号缓冲区保存数据的第 3 字节开始读取 4 字节数据，保存到实数变量 V1%中。

3.1.18 从缓冲区提取 1 字节的数据

指令名称	GETBYTE
FN 代码	587
标题名称	缓冲读取
概要	从缓冲区的任意位置读取 1 字节，保存到整数变量。

概要

从缓冲区的任意位置读取 1 字节，保存到任意的整数变量（0～255 的值）。

参数

第 1 参数	缓冲区编号	指定读取数据的缓冲区编号。 （1–16）
第 2 参数	整数变量	指定保存读出值的整数变量。
第 3 参数	起始地址	指定读取缓冲区的开头位置。从指定的位置读取 1 字节数据。 （0–1023）

机器人语言书写实例

GETBYTE 1, V1%, 5

将存储于 1 号缓冲区中数据的第 5 字节保存到整数变量 V1%中。

3.1.19 错误变量

如“3.1.1 概要”所述，套接字功能检测到通信异常时，将异常写入错误变量，结束功能处理。由于执行下述套接字功能时，错误变量被覆盖，因此应创建函数执行后可以立即确认的程序。
可以从机器人语言程序以 E1%,E2%的形式获取错误变量。

以 E1%输出的错误代码

编号	内容
0	至此处理已成功完成。
–1	通信服务的启动未完成。 启动本控制装置的电源时，执行通信相关功能的初始化。检测到这种错误时，可以等待片刻后再执行。
–2	参数处于非正常状态。 参数问题与通过 SOCKWAIT 等待连接的套接字编号相同时，被检测出来。参数在范围之外时，发生［E2250 步骤数据异常］，并停止执行程序。
–3	套接字已创建。 针对已创建的套接字编号执行了 SOCKCREATE 时，被检测出来。
–4	套接字未创建。 想要使用未执行 SOCKCREATE 的套接字进行通信时，被检测出来。
–5	使用了套接字的处理已执行。 使用了其他用户任务程序指定的套接字编号的处理正在执行中时，被检测出来。
–7	已超时。超时时，可以不使用套接字，而是将其关闭。
–8	IP 地址未设定。 采用 UDP 协议发送数据，IP 地址未指定时，被检测出来。可以使用 SOCKCONNECT 设定 IP 地址。
–999	上述以外的异常发生时，被检测出来。详情请确认 E2%。

以 E2%输出的错误代码

编号	内容
10022	通过 SOCKBIND 指定端口编号前，使用 SOCKWAIT 等指令，被检测出来。
10040	接收大于 UDP 通信指定缓冲区大小的数据，数据被减缩时，被检测出来。
10045	针对通过 SOCKCREATE 创建为 UDP 的套接字，执行 SOCKWAIT 等指令，被检测出来。SOCKWAIT 只限 TCP 通信时可以使用。
10048	指定 SOCKBIND 使用的端口编号时，被检测出来。
10054	在 TCP 通信中，由于连接后的故障等，与已切断的设备之间收发数据时，被检测出来。但是，在检测出切断之前，需要一定时间，因此在切断后无法立即检测出来。
10057	TCP 通信中，未通过 SOCKCONNECT 和 SOCAKWAIT 进行连接，就发送数据等时，被检测出来。

※ E2%时，将传回以 WinSock 规格定义的错误代码。如显示此表中没有的错误代码时，可以参见市售网络程序的专业书籍。

通过异常履历或变量监视器无法直接确认错误变量。另外，也不会显示用户任务监视器的错误编号。要确认错误变量时，依照下述内容创建程序。

■创建套接字时发生的错误变量确认方法实例

① 将错误变量带入整数变量，通过整数变量监视器确认。

程序实例①

```
'创建套接字
V101% = 0
V102% = 0
SOCKCREATE 1,0
IF E1%<0 THEN *ERROR
…
EXIT
*ERROR
V101% = E1%
V102% = E2%
EXIT
```

异常处理
将错误变量 E1、E2 代入整数变量 101、102，结束程序。

重点

② 在用户画面上显示错误内容，并进行确认。

程序实例②

```
'创建套接字
SOCKCREATE 1,0
IF E1%<0 THEN *ERROR
…
EXIT
*ERROR
WINDOW 0,0,200,100
PRINT #0,STR$(E1%)
PRINT #0,STR$(E2%)
SOCKCLOSE 1
PAUSE 3000
EXIT
```

异常处理
异常发生时，在 TP 上显示错误代码 3 秒后结束程序。

NOTE

第4章 程序创建实例

4.1 获取整数变量的值（服务器程序）

这是根据来自外部的要求，传回整数变量值的程序。
程序启动后，生成 TCP 协议的套接字，通过 48952 号端口等待连接。从外部连接后，接收 1 字节的数值数据（1～100）的请求信息，并据此以 4 字节的大端序传回整数变量 1～100(V1%～V100%)的值。
发送数据后，关闭套接字，结束程序。

要使用的变量

变量	编号	使用内容
整数变量	100	接收数据大小
整数变量	120	发送数据大小
整数变量	150	接收数据（请求的整数变量编号）

程序创建实例

```
'传回整数变量 SERVER
'创建套接字
指令①  SOCKCREATE 1,0
       IF E1%<0 THEN *ERROR
'分配端口编号
指令②  SOCKBIND 1,48952
       IF E1%<0 THEN *ERROR
'等待连接
指令③  SOCKWAIT 1,2,0
       IF E1%<0 THEN *ERROR
'数据接收
指令④  SOCKRECV 2,1,1,5,V100%
       IF E1%<0 THEN *ERROR
指令⑤  GETBYTE 1,V150%,0
       IF V150%<1 THEN *ERROR
       IF V150%>100 THEN *ERROR
'数据发送
指令⑥  SETINT 2,V%[V150%],0
指令⑦  SOCKSEND 2,2,4,5,V120%
       IF E1%<0 THEN *ERROR
'关闭套接字
       SOCKCLOSE 1
       SOCKCLOSE 2
       EXIT
*ERROR
       WINDOW 0,0,200,100
       PRINT #0,STR$(E1%)
       PRINT #0,STR$(E2%)
       SOCKCLOSE 1
       SOCKCLOSE 2
       PAUSE 3000
       EXIT
```

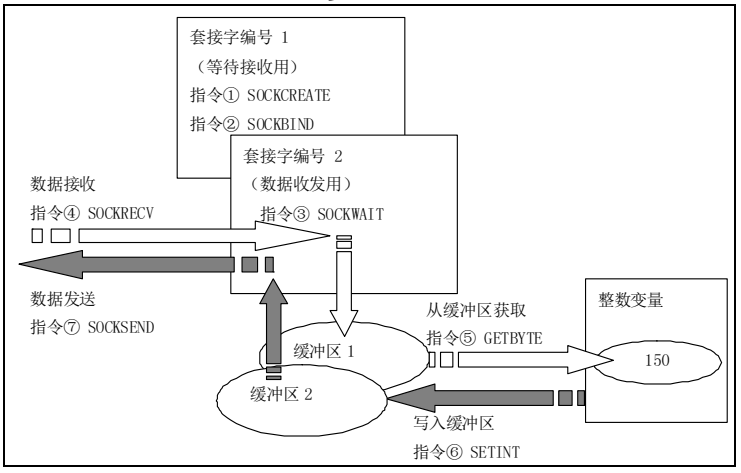
等待连接
等待来自客户端的连接。使用套接字编号 2 执行与已连接客户端的通信。

接收请求信息
接收来自客户端的请求信息。接收后检查数据，处于范围外时，请求报错。(有效范围 1～100)

数据发送
向客户端发送符合请求的数据。例 V%[1]→V1%

关闭套接字
关闭套接字。不仅限于用于等待接收的套接字，也需关闭用于收发数据的套接字。

异常处理
异常发生时，在 TP 上显示错误代码 3 秒后结束程序。
该例子中，为了便于理解，对异常进行批处理。



4.2 获取各单元的错误编号（服务器程序）

作为 UDP 协议的程序实例，这是传回本控制装置中发生的错误代码所使用的程序。
程序启动后，将创建 UDP 协议的套接字，并通过 48952 端口等待接收数据。作为来自客户端的请求信息，接收到单元编号后，从 SYSTEM 函数获取错误代码，发送整数值的数据。
发送数据后，关闭套接字，结束程序。
本程序使用变量定义文件[SAMPLE.INC]。

要使用的变量

变量	编号	使用内容
整数变量	100	接收数据大小
整数变量	120	发送数据大小
整数变量	150	接收数据（单元编号）
整数变量	200	错误代码

程序创建实例

```
INCLUDE “SAMPLE”
'创建套接字
SOCKCREATE SOCK_NO,UDP
IF E1%<0 THEN *ERROR

'分配端口编号
SOCKBIND SOCK_NO,PORT_NO
IF E1%<0 THEN *ERROR

'数据接收
SOCKRECV SOCK_NO,BUF_NO,1,0,V100%
IF E1%<0 THEN *ERROR
GETBYTE BUF_NO,UNIT_NO,0
IF UNIT_NO<1 THEN *ERROR
IF UNIT_NO>9 THEN *ERROR

'数据发送
E_CODE = SYSTEM%(140+UNIT_NO)
SETINT BUF_NO,E_CODE,0
SOCKSEND SOCK_NO,BUF_NO,4,5, V120%
IF E1%<0 THEN *ERROR

'关闭套接字
SOCKCLOSE 1
EXIT

*ERROR
WINDOW 0,0,200,100
PRINT #0,STR$(E1%)
PRINT #0,STR$(E2%)
SOCKCLOSE 1
PAUSE 3000
EXIT
```

接收请求信息
采用 UDP 协议时，由于不执行连接处理，无需等待连接（SOCKWAIT）立即就可以等待接收数据。接收后检查数据，处于范围外时，报错。(有效范围 1～9)

接收请求信息
使用 SYSTEM 函数取得指定单元的当前发生错误的代码，并将其传回。

异常处理
异常发生时，在 TP 上显示错误代码 3 秒后结束程序。
该例子中，为了便于理解，对异常进行批处理。

SAMPLE.INC

```
SOCK_NO,1
BUF_NO,1
PORT_NO,48952
TCP,0
UDP,1
UNIT_NO,V150%
E_CODE,V200%
```

SYSTEM 函数相关内容可以参见操作说明书“机器人语言”。
变量定义文件相关内容可以参见操作说明书的“用户任务”。

4.3 从视觉装置获取数据（客户端程序）

这是从视觉装置获取偏离量，代入移位寄存器所使用的程序。
与视觉装置的通信以字符串执行，并将表示换行的字符串（13、10 的 2 字节数据）当作分隔符。视觉装置采用接收到［GVA002［换行］］的请求后，首先执行错误响应，然后发送偏离量的方式。

要使用的变量

变量	编号	使用内容
整数变量	50	发送数据大小
整数变量	100	接收数据大小
整数变量	150	接收数据（错误响应）
整数变量	151	接收数据（偏离量）
字符串变量	10	将接收数据转换为字符串的数据
整数变量	160	将字符串转换为数值的数据

程序创建实例

```
'视觉连接

‘创建套接字
SOCKCREATE 1,0
IF E1%<0 THEN *ERROR

'连接视觉装置
SOCKCONNECT 1,1,23,5
IF E1%<0 THEN *ERROR

'发送数据请求信息
SOCKSENDSTR 1,"GVA002",LEN("GVA002"),0,V50%,3
IF E1%<0 THEN *ERROR

'接收响应信息
SOCKRECV 1,1,3,5,V100%
IF E1%<0 THEN *ERROR
GETBYTE 1,V200%,0
IF V200%<>49 THEN *ERROR

'数据接收
*GET_DATA
SOCKRECV 1,1,1,5,V100%
IF E1%<0 THEN *ERROR
GETBYTE 1,V151%,0
IF 13=V151% THEN *GET_DATA_END
V10$ = V10$ + CHR$(V151%)
GOTO *GET_DATA
*GET_DATA_END

SOCKRECV 1,1,1,5,V100%

'代入移位寄存器
V160% = VAL(V10$)
R1 = (0,V160%,0,0,0,0)
‘
SOCKCLOSE 1
EXIT

'错误处理
*ERROR
WINDOW 1,1,100,100
PRINT #0,STR$(E1%)
SOCKCLOSE 1
PAUSE 5000
EXIT
```

发送数据要求信息
向视觉装置发送数据请求信息。通过在 SOCKSENDSTR 的末尾参数附加 3，可以附加表示换行的 13、10 的 2 字节数据。

数据接收
获取逐个字符的数据，并发送数据，直到收到分隔符为止。

代入移位寄存器
使用普通函数将字符串转换为数值，并作为 Y 方向的偏移量代入移位寄存器 1。

异常处理
异常发生时，在 TP 上显示错误代码 3 秒后结束程序。
该例子中，为了便于理解，对异常进行批处理。

4.4 从视觉装置获取移位数据

将以整数变量 V1%保存后，若能调用本用户宏就可执行指令。如果以 V1%=10 条件执行指令，将在视觉装置上发送移位请求指令"T1"，并且以字符串接收用","分隔的移位数据。这些数据通过机器人控制装置读取各字节，将转换为实数便代入于移位寄存器。

IP 地址

机器人控制装置	192.168.1.1
视觉装置	192.168.1.10

要使用的 TCP 通信端口编号

10030

指令(V1%)

1	设定变更 0	"PW, 1, 0"
2	设定变更 1	"PW, 1, 1"
3	设定变更 2	"PW, 1, 2"
4	设定变更 3	"PW, 1, 3"
10	移位数据请求指令	"T1"
15	用户任务结束	

移位数据实例

数据 1		分隔符	数据 2		分隔符	数据 3		终止符
"1"	"0"	"," (44)	"2"	"0"	"," (44)	"3"	"0"	CR (13)

接收该数据后，如下所示将作为结果存储于字符串变数。

```
V20$ = "10"
V21$ = "20"
V22$ = "30"
```

需要该字符串需要转换成实数变量时，请使用如下所示函数 VAL()。

```
V1! = VAL(V20$)
V2! = VAL(V21$)
V3! = VAL(V22$)
```

如下所示已获取的实数变量可以在移位寄存器（移位变量）保存。

```
R1=(V1!,V2!,0,V3!,0,0)
```

这些程序仅为示例。通信数据的格式根据使用装置有所差异，因此应充分确定所使用装置的操作说明书便创建程序。

创建程序实例	说明
' ---------- ' INITIALIZE VARIABLES ' ---------- V11% = 10030 V1$ = "" V2$ = "PW, 1, 0" V3$ = "PW, 1, 1" V4$ = "PW, 1, 2" V5$ = "PW, 1, 3" V6$ = "T1" V7$ = "PW" V11$ = "" V119% = 0	**变量的初始化** 10030 : **端口**编号 V1$: 信息用 V2$: 设**定**转换**指令** 0 V3$: 设**定**转换**指令** 1 V4$: 设**定**转换**指令** 2 V5$: 设**定**转换**指令** 3 V6$: **移位数据**请**求指令** V7$: 设**定**转换**指令** 已执行**指令**
' ---------- ' SOCKET CREATE ' ---------- SOCKCREATE 1,0 IF E1% < 0 THEN *ERROR1 V119%=1	**套接字的生成** SOCKET=1/PROTOCOL=0(=TCP) 检查错误
' ---------- ' SOCKET CONNECT ' ---------- SOCKCONNECT 1, 10, V11%, 10	**套接字的连接** **套接字**编号 1 IP **地址** *.*.*.10 **端口**编号 V11%(=10030) **超**时 10 秒
' ---------- ' COMMAND CHECK ' ---------- ' V1%=1 : SETTING CHANGE 0 ' V1%=2 : SETTING CHANGE 1 ' V1%=3 : SETTING CHANGE 2 ' V1%=4 : SETTING CHANGE 3 ' V1%=10: TRIGGER ' V1%=15: USER TASK COMPLETE ' ---------- IF V1% = 1 THEN *CHGSETTING0 IF V1% = 2 THEN *CHGSETTING1 IF V1% = 3 THEN *CHGSETTING2 IF V1% = 4 THEN *CHGSETTING3 IF V1% = 10 THEN *TRIGGER IF V1% = 15 THEN *END V1$ = "Error3" GOTO *END	**解析指令** 以保存在 V1%的数值来执行标签跳跃。 设**定**变**更** 0 设**定**变**更** 1 设**定**变**更** 2 设**定**变**更** 3 **移位数据**请**求指令** **用**户**任**务结**束** 跳跃到 END 标签

（下文）	说明
' ----------	
' COMMAND V2$ = "PW, 1, 0"	**发送设定变更指令"PW"**
' ----------	
*CHGSETTING0	**套接字**编号 1
SOCKSENDSTR 1,V2$,LEN(V2$),10,V12%,3	发**送字符串** V2$(="PW, 1, 0")
IF E1% < 0 THEN *ERROR1	长**度** LEN(V2$)
V119%=3	**超**时 10 秒
GOSUB *CHK_RET_VAL3	发送**数据**长度 V12%
GOTO *END	终**止** 3(="\r\n")
' ----------	
' COMMAND V3$ = "PW, 1, 1"	
' ----------	
*CHGSETTING1	
SOCKSENDSTR 1,V3$,LEN(V3$),10,V13%,3	
IF E1% < 0 THEN *ERROR1	
V119%=4	
GOSUB *CHK_RET_VAL3	
GOTO *END	
' ----------	
' COMMAND V4$ = "PW, 1, 2"	
' ----------	
*CHGSETTING2	
SOCKSENDSTR 1,V4$,LEN(V4$),10,V14%,3	
IF E1% < 0 THEN *ERROR1	
V119%=5	
GOSUB *CHK_RET_VAL3	
GOTO *END	
' ----------	
' COMMAND V5$ = "PW, 1, 3"	
' ----------	
*CHGSETTING3	
SOCKSENDSTR 1,V5$,LEN(V5$),10,V15%,3	
IF E1% < 0 THEN *ERROR1	
V119%=6	
GOSUB *CHK_RET_VAL3	
GOTO *END	
' ----------	
' TRIGGER	**发送移位数据请求指令"T1"**
' ----------	
*TRIGGER	
SOCKSENDSTR 1,V6$,LEN(V6$),10,V15%,3	**套接字**编号 1
IF E1% < 0 THEN *ERROR1	发**送字符串** V6$(="T1")
V119%=7	长**度** LEN(V6$)
GOSUB *CHK_RET_VAL4	**超**时 10 **秒**
GOTO *END	**送信数据长** V15%
	终**止** 3(="\r\n")

（下文）	说明
' ----------	
' END	**用户任务结束**
' ----------	
*END	
SOCKCLOSE 1	关闭套接字
EXIT	退出用户任务
' ----------	
' ERROR PROCESS	**错误处理**
' ----------	
*ERROR1	
V1$ = "Error1"	保存错误信息
SOCKCLOSE 1	关闭套接字
EXIT	退出用户任务
*ERROR2	
V1$ = "Error2"	保存错误信息
SOCKCLOSE 1	关闭套接字
EXIT	退出用户任务
' ----------	
' RETURN VALUE CHECK (SETTING CHANGE)	**检查返回值（设定变更）**
' ----------	
*CHK_RET_VAL3	**套接字**编**号** 1
SOCKRECV 1,1,2,5,V3%	缓**冲区**编**号** 1
V119%=V119%+10	读取长度 2 **字**节
IF E1% < 0 THEN *ERROR1	**超**时 5 **秒**
	読取结束**数据**长度 V3%
GETSTR 1,V10$, 0, 2	缓**冲区**编**号** 1
	读入**字符串** V10$
	开始位置 0
	读取**数据**数量 2 **字**节
IF V10$<>V7$ THEN *ERROR2	所读取**字符串**不是"PW"便成为错误
RETURN	返回调用原

（下文）	说明
' ----------	
' RETURN VALUE CHECK (TRIGGER)	**检查返回值（触发信号）**
' ----------	
*CHK_RET_VAL4	
SOCKRECV 1,1,4,5,V4%	缓**冲区** 1 接收 4 **字**节
V119%=V119%+10	
IF E1% < 0 THEN *ERROR1	
GETSTR 1, V10$, 0, 2	V10$获取 2 字符
IF V10$<>V6$ THEN *ERROR2	V10$为"T1"时，进入以下
	*GETDATA
V52%=0	排列配字符的初始化
' ----------	
' READ THE SHIFT DATA	**移位数据读取**
' ----------	读取各字符
*GETDATA	
SOCKRECV 1,1,1,5,V5%	**套接字**编**号** 1 缓**冲区**编**号** 1 读取长度 1 **字**节 **超**时 5 **秒** 读取结束**数据**长度 V5%
IF E1% < 0 THEN *ERROR1	检查错误
GETBYTE 1,V51%, 0	V51%读取 1 **字**节
IF V51%=13 THEN *GETDATAEND	若为终止符，跳跃于*GETDATAEND
IF V51%=44 THEN *SETVALUE	若为**分隔符**，跳跃于 *SETVALUE
V11$=V11$+CHR$(V51%)	附加在**字符串** V11$的末尾
GOTO *GETDATA	重复
' ----------	
' COPY THE V11$ TO THE OTHER VARIABLE	**成为分隔符将复制字符串变量**
' ----------	
*SETVALUE	
V$[20+V52%]=V11$	复制**字符串**变量
V52%=V52%+1	递增排列配字符
V11$=""	缓**冲区用字符串**的初始化
GOTO *GETDATA	
' ----------	
' COPY THE V11$ TO THE OTHER VARIABLE	**成为终止符就结束读取**
' ----------	
*GETDATAEND	
V$[20+V52%]=V11$	保存最后数据
SOCKRECV 1,1,1,5,V5%	
V1!=VAL(V20$)	**字符串**转换为实数 X
V2!=VAL(V21$)	**字符串**转换为实数 Y
V3!=VAL(V22$)	**字符串**转换为实数 Rx
R1=(V1!,V2!,0,V3!,0,0)	代入移位变量
RETURN	返回调用原

4.5 数据接收方法

TCP 协议下接收数据时，了解数据结束的方法有下面几种。
1.预先决定数据大小
2.以其它数据发送数据大小
3.决定最后的字符

通过套接字进行通信时，需要运用某种方法了解数据的结束。“4.1 获取整数变量的值（服务器程序）”的程序中使用的是第 1 个方法，“4.3 从视觉装置获取数据（客户端程序）”的程序中使用的是第 3 个方法。
发送数据时也需要确认接收侧使用的是哪种方法。

4.5.1 预先决定数据大小

由于该方法可以预先知道大小，可以直接为 SOCKRECV 的第 3 个参数指定数据大小，以获取数据。例如，知道数据的大小为 4 字节时，如下所述指定 4。

```
SOCKRECV 1, 1, 4, 5, V1%
```

4.5.2 以其他数据发送数据大小

该方法以预先决定了大小的数据先行发送数据大小，其后发送可变长度的数据。接收这样的数据时，如下所述。（数据大小以 4 字节的大端序发送）

```
SOCKRECV 1, 1, 4, 5, V1%
GETINT 1,%2%,0
SOCKRECV 1, 1, V2%, 5, V1%
```

接收数据大小

接收可变长度数据

4.5.3 决定最后发送的字符

该方法可以发送数据最后的分隔符。此时，逐个字节地接收数据，在接收到分隔符时，将结束接收数据。这种情况如下所示。（作为分隔符，发送 0。）

```
*GET_DATA
SOCKRECV 1, 1, 1, 5, V1%
GETBYTE 1, V2%, 0
IF 0=V2% THEN *GET_DATA_END
V1$ = V1$ + CHR$(V2%)
GOTO *GET_DATA
*GET_DATA_END
```

由于决定最后发送字符的方法可以逐个字节执行接收处理，因此有步骤数增加，与其他方法相比有处理时间变长的。特别是用户任务的优先度较低时，处理时间将变长。

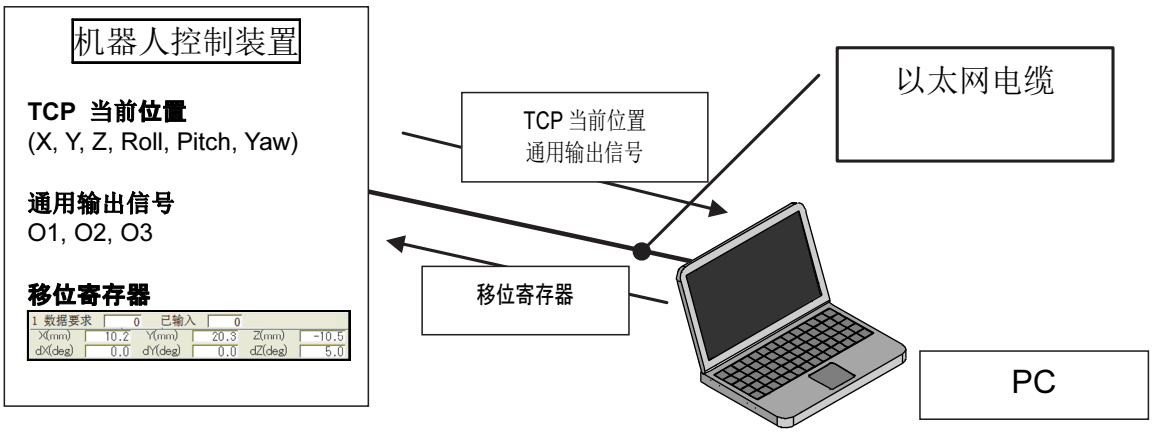
NOTE

第5章 PC 端程序实例

5.1 PC 与机器人控制装置的套接字通信

5.1.1 实例程序的概述

在此实例中，PC 将从机器人控制装置接收机器人的 TCP 当前位置和通用输出信号状态后，通过 PC 设定移位寄存器。



IP 地址

机器人控制装置	192.168.1.1
PC	192.168.1.10

※ 这些地址可通过修改实例程序而改变。

要使用的 TCP 通信端口编号

10030

※ 该端口可通过修改实例程序而改变。

PC 端实例程序

OS	Windows7
开发环境	Visual C++ 根据 Visual Studio 版本，可能需要设置链接“winsock”库。此时，请添加“ws2_32.lib”。
C++ 源代码	main_C.cpp (客户) main_S.cpp (服务器)
可执行文件	Socket_Client_test.exe (客户) Socket_Server_test.exe (服务器)

提示

这些程序仅为实例。这些程序本公司将不做担保。创建实际程序时，请事先检查程序。并且，关于开发环境(例如 Visual Studio 等)及 C++编程语言等，请参考市售书籍。

提示

通过所接收的移位数据来移动机器人时，请参照操作说明书“基于外部输入的移动功能”(TFDCN-047)。

提示

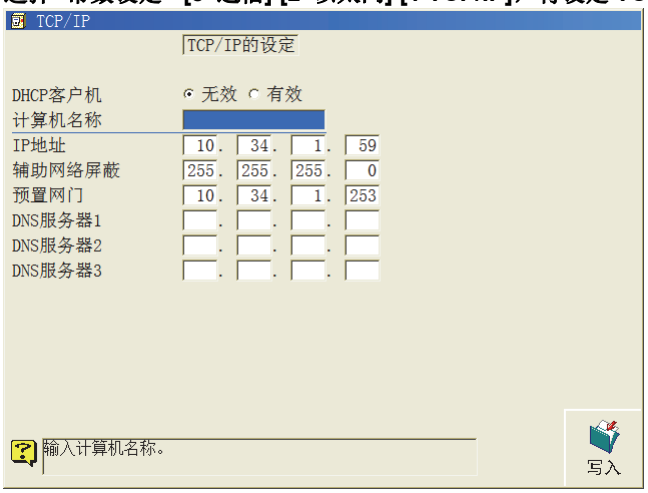
本程序中的源代码可通过 DVD 等提供.。
请联系本公司销售部门。

5.1.2 操作程序

操作程序的实例

1 **机器人控制装置端的设定**

选择<常数设定> [8 通信] [2 以太网] [1 TCP/IP]，将设定 TCP/IP。

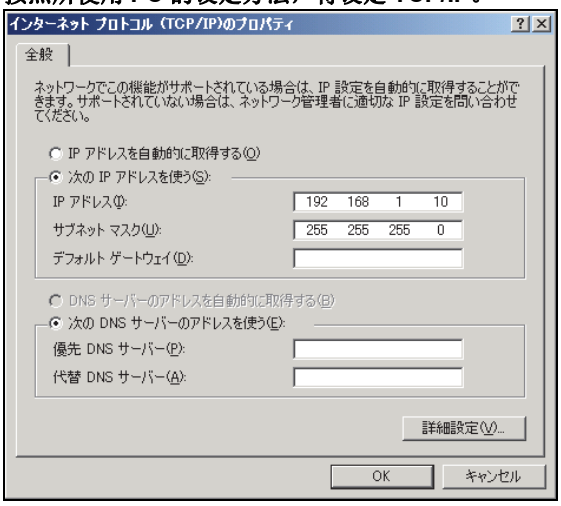


关于 TCP/IP 的详细设定，请参照操作说明书“设定篇”的以太网相关章节。

2 **请按<写入>并保持设定。**

3 **PC 端的设定**

按照所使用 PC 的设定方法，将设定 TCP/IP。



4 **用以太网电缆连接 PC 和机器人控制装置。**

在控制装置端将连接 CPU 基板上的 LAN 端口。

5 **请启动服务器端程序。**

>>服务器程序启动时将等待客服端程序。

(NOTE)

如 PC 为客户端，启动机器人控制装置上的服务器程序；

如机器人控制装置为客户端，启动 PC 上的服务器程序。

6 **接下来，启动客户端程序。**

7 **请在 PC 端程序上输入服务器的 IP 地址。**

(只有 PC 为客户端的情况)

关于启动机器人控制装置的用户任务，请参见操作说明书“用户任务”。

5.1.3 画面实例

PC 端

将显示机器人 TCP 当前位置和通用输出信号的状态。

```
----------------
Current Position
----------------
X     : 563
Y     : 1.53845e-007
Z     : 810
Roll  : 0
Pitch : -90
Yaw   : -180
Out3_____
Out2____ |
Out1___ ||
       |||
IO    : 010
```

请输入移位寄存器的设定值。
此处将输入 (dX， dY， dZ， dθZ， dθY， dθX)。

```
Set X [mm]
10.2
Set Y [mm]
20.3
Set Z [mm]
-10.5
Set Roll [deg]
5.0
Set Pitch [deg]
0
Set Yaw [deg]
0
```

dX = 10.2 [mm]

dY = 20.3 [mm]

dZ = -10.5[mm]

dθZ = 5.0 [deg]

dθY = 0.0 [deg]

dθX = 0.0 [deg]

机器人控制装置端

从 PC 接收的值将被设定在机器人控制装置的“移位寄存器”。通过<维修 > [3 监视器 1] [14 操作监视器] [2 移动寄存器]显示移位寄存器。

X	X 轴方向移位量[mm]	(dX)
Y	Y 轴方向移位量[mm]	(dY)
Z	Z 轴方向移位量[mm]	(dZ)
θX	X 轴周围旋转移位量[deg]	(dθX)
θY	Y 轴周围旋转移位量[deg]	(dθY)
θZ	Z 轴周围旋转移位量[deg]	(dθZ)

5.1.4 用户任务程序(服务器)创建实例

```
' TCP/IP_Socket(Server)

'------------------------------------------------------------
' Setup
'------------------------------------------------------------
DIM Port                AS INTEGER
DIM SendDataLen         AS INTEGER
DIM SendIODataLen       AS INTEGER
DIM ReceiveDataLen      AS INTEGER
DIM ReceiveBufferLen    AS INTEGER
Port                    = 10030                 'Port#
SendDataLen             = 12                    'Send Data Len
SendIODataLen           = 3                     'Send IO Data Len
ReceiveDataLen          = 8                     'Receive Data Len
ReceiveBufferLen        = ReceiveDataLen * 6    'Receive Buffer Len
'------------------------------------------------------------
' Initialize
'------------------------------------------------------------
V10! = 0            'Current X
V11! = 0            'Current Y
V12! = 0            'Current Z
V13! = 0            'Current Roll
V14! = 0            'Current Pitch
V15! = 0            'Current Yaw
V10$ = ""   'Send X
V11$ = ""   'Send Y
V12$ = ""   'Send Z
V13$ = ""   'Send Roll
V14$ = ""   'Send Pitch
V15$ = ""   'Send Yaw
V16$ = ""   'Send IO
V17$ = ""   'Send Variable
V1$   = ""          'Receive X
V2$   = ""          'Receive Y
V3$   = ""          'Receive Z
V4$   = ""          'Receive Roll
V5$   = ""          'Receive Pitch
V6$   = ""          'Receive Yaw
V1!   = 0           'Receive X
V2!   = 0           'Receive Y
V3!   = 0           'Receive Z
V4!   = 0           'Receive Roll
V5!   = 0           'Receive Pitch
V6!   = 0           'Receive Yaw
R1 = (0，0，0，0，0，0)      'Shift Register
```

定义执行套接字通信时的常数。

变量的初始化

```
'------------------------------------------------------
' Create
'------------------------------------------------------
SOCKCREATE 1，0
IF E1%<0 THEN *ERROR
'------------------------------------------------------
' Bind
'------------------------------------------------------
SOCKBIND 1，Port
IF E1%<0 THEN *ERROR
'------------------------------------------------------
' Wait
'------------------------------------------------------
SOCKWAIT 1，2，0
IF E1%<0 THEN *ERROR
'------------------------------------------------------
'GET Position
'------------------------------------------------------
V10! = SYSTEM!(810)             'X
V11! = SYSTEM!(811)             'Y
V12! = SYSTEM!(812)             'Z
V13! = SYSTEM!(813)             'Roll
V14! = SYSTEM!(814)             'Pitch
V15! = SYSTEM!(815)             'Yaw
'------------------------------------------------------
'Send
'------------------------------------------------------
V10$ =LEFT$(STR$(V10!)， SendDataLen)
V11$ =LEFT$(STR$(V11!)， SendDataLen)
V12$ =LEFT$(STR$(V12!)， SendDataLen)
V13$ =LEFT$(STR$(V13!)， SendDataLen)
V14$ =LEFT$(STR$(V14!)， SendDataLen)
V15$ =LEFT$(STR$(V15!)， SendDataLen)

V16$ =STR$(O[1])
V16$ =V16$ + STR$(O[2])
V16$ =V16$ + STR$(O[3])
V16$ =LEFT$(V16$， SendIODataLen)

V17$ =V10$+"，"+V11$+"，"+V12$+"，"+V13$+"，"+V14$+"，"+V15$+"，"+V16$+"，"

SOCKSENDSTR 2，V17$，LEN(V17$)，0，V105%，0
IF E1%<0 THEN *ERROR
```

与客户端 PC 连接。

TCP 当前值保存到变量。

TCP 当前值保存到字符串变量。

通用输出信号的状态保存到字符串变量。

数据发送到 PC。

```
'------------------------------------------------------------
' Receive
'------------------------------------------------------------
SOCKRECV 2，1，ReceiveBufferLen，0，V110%
IF E1%<0 THEN *ERROR

GETSTR 1， V1$， ReceiveDataLen * 0， ReceiveDataLen
GETSTR 1， V2$， ReceiveDataLen * 1， ReceiveDataLen
GETSTR 1， V3$， ReceiveDataLen * 2， ReceiveDataLen
GETSTR 1， V4$， ReceiveDataLen * 3， ReceiveDataLen
GETSTR 1， V5$， ReceiveDataLen * 4， ReceiveDataLen
GETSTR 1， V6$， ReceiveDataLen * 5， ReceiveDataLen
'------------------------------------------------------------
' Set Val
'------------------------------------------------------------
V1! = VAL(V1$)
V2! = VAL(V2$)
V3! = VAL(V3$)
V4! = VAL(V4$)
V5! = VAL(V5$)
V6! = VAL(V6$)
'------------------------------------------------------------
'Set Register
'------------------------------------------------------------
R1=(V1!，V2!，V3!，V4!，V5!，V6!)
'x，y，z，roll(theta_z)，pitch(theta_y)，yaw(theta_x)

'------------------------------------------------------------
' Close
'------------------------------------------------------------
SOCKCLOSE 1
SOCKCLOSE 2
EXIT

*ERROR
WINDOW 0，0，200，100
PRINT #0，STR$(E1%)
PRINT #0，STR$(E2%)
SOCKCLOSE 1
SOCKCLOSE 2
PAUSE 3000
EXIT
```

从 PC 接收数据。

所接收数据保存到变量。

数据保存到移位寄存器。

关闭套接字。

若失败套接字通信的话，
将执行错误处理。

5.1.5 PC 端程序(客户)创建实例

```
#include <stdio.h>
#include <winsock2.h>
#include <iostream>
#include <string.h>

#define PORT_NUM 10030
#define SEND_DATA_SIZE 8
#define SEND_BUFFER_LEN (SEND_DATA_SIZE * 6)
#define REC_DATA_SIZE 12
#define REC_DATA_NUM 7
#define REC_IO_DATA_SIZE 3
#define REC_BUFFER_LEN (REC_DATA_SIZE * 6 + REC_IO_DATA_SIZE + REC_DATA_NUM)

int main(void){

//------------------------------------------------------
//                                  Socket Setup
//------------------------------------------------------
SOCKET sock;

//Server information

struct sockaddr_in dest;

memset(&dest，  0，  sizeof(dest));

//Server address

//xxx.xxx.xxx.xxx
char destination[] = "000.000.000.000";

std::cout << "Set Server address" << std::endl;

std::cin >> destination;

//Socket communication

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2，0)，  &wsaData) != 0)
{
        std::cout << "WSAStartup failed" << std::endl;
        return 1;
}

//Set the information

dest.sin_port = htons(PORT_NUM);

dest.sin_family = AF_INET;

dest.sin_addr.s_addr = inet_addr(destination);
```

定义执行套接字通信时的常数。

与控制装置连接。

```
//Make Socket

sock = socket(AF_INET， SOCK_STREAM， 0);
if (sock == INVALID_SOCKET)
{
        std::cout << "socket : " << WSAGetLastError() << std::endl;
        return 1;
}

//Communicate to the Server

if(connect(sock， (struct sockaddr *) &dest， sizeof(dest)))
{
        std::cout << "Err Communicate to" << destination << std::endl;

        return -1;
}

std::cout << "Communication to " << destination << " completed" << std::endl;
```

与控制装置连接。

```
//------------------------------------------------------
//                          Recieve from Server
//------------------------------------------------------

char rec_buffer[REC_BUFFER_LEN] = {};
int check = 0;
char *temp = NULL;
char *ctx = NULL;
std::cout << "---------- " << std::endl;
std::cout << "Current Position " << std::endl;
std::cout << "---------- " << std::endl;

//Recieve
check = recv(sock， rec_buffer， sizeof(rec_buffer)， 0);
if(check < 1){
        std::cout << "recv : " << WSAGetLastError() << std::endl;
        return 1;
}
```

接收数据。

```
temp = strtok_s(rec_buffer，"，"，&ctx);
std::cout << "X       : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Y       : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Z       : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Roll    : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Pitch : " << temp << std::endl;
```

显示接收到的数据。

```
temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Yaw     : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);

std::cout << "Out3_____ " << std::endl;
std::cout << "Out2____ |" << std::endl;
std::cout << "Out1___ ||" << std::endl;
std::cout << "           |||" << std::endl;
std::cout << "IO     : " << temp << std::endl;

//-------------------------------------------------------
//                                    Send to Server
//-------------------------------------------------------

float input = 0.0;
char shift_x[] = "00000.00";
char shift_y[] = "00000.00";
char shift_z[] = "00000.00";
char shift_r[]   = "00000.00";
char shift_p[]   = "00000.00";
char shift_ya[] = "00000.00";

std::cout << "Set X [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_x，"%08.2f"，input);

std::cout << "Set Y [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_y，"%08.2f"，input);

std::cout << "Set Z [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_z，"%08.2f"，input);

std::cout << "Set Roll [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_r，"%08.2f"，input);

std::cout << "Set Pitch [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_p，"%08.2f"，input);

std::cout << "Set Yaw [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_ya，"%08.2f"，input);
```

显示接收到的数据。

输入要发送的数据。

```
char buffer[SEND_BUFFER_LEN+1];

sprintf_s(buffer，"%s%s%s%s%s%s"，shift_x，shift_y，shift_z，shift_r，shift_p，shift_ya);

std::cout << "Send Data "    << std::endl;
std::cout << buffer    << std::endl;

//Send
check = send(sock，  buffer，  sizeof(buffer)，  0);

//---------------------------------------------------------
//                                    End Socket
//---------------------------------------------------------

closesocket(sock);

WSACleanup();

return 0;

}
```

发送数据。

关闭套接字

5.1.6 用户任务程序（客户）创建实例

```
' TCP/IP_Socket(Client)

'-------------------------------------------------------
' Setup
'-------------------------------------------------------
DIM Port                    AS INTEGER
DIM IpAdd                   AS INTEGER
DIM SendDataLen   AS INTEGER
DIM SendIODataLen           AS INTEGER
DIM ReceiveDataLen          AS INTEGER
DIM ReceiveBufferLen        AS INTEGER
Port                = 10030             'Port#
IpAdd               = 10                'Server IP address
SendDataLen         = 12                'Send Data Len
SendIODataLen       = 3                 'Send IO Data Len
ReceiveDataLen = 8                      'Receive Data Len
ReceiveBufferLen = ReceiveDataLen * 6  'Receive Buffer Len
'-------------------------------------------------------
' Initialize
'-------------------------------------------------------
V10! = 0            'Current X
V11! = 0            'Current Y
V12! = 0            'Current Z
V13! = 0            'Current Roll
V14! = 0            'Current Pitch
V15! = 0            'Current Yaw
V10$ = ""           'Send X
V11$ = ""           'Send Y
V12$ = ""           'Send Z
V13$ = ""           'Send Roll
V14$ = ""           'Send Pitch
V15$ = ""           'Send Yaw
V16$ = ""           'Send IO
V17$ = ""           'Send Variable
V1$   = ""                  'Receive X
V2$   = ""                  'Receive Y
V3$   = ""                  'Receive Z
V4$   = ""                  'Receive Roll
V5$   = ""                  'Receive Pitch
V6$   = ""                  'Receive Yaw
V1!   = 0           'Receive X
V2!   = 0           'Receive Y
V3!   = 0           'Receive Z
V4!   = 0           'Receive Roll
V5!   = 0           'Receive Pitch
V6!   = 0           'Receive Yaw
R1 = (0，0，0，0，0，0)      'Shift Register
```

定义执行套接字通信时的常数。

变量的初始化

```
'------------------------------------------------------------
' Create
'------------------------------------------------------------
SOCKCREATE 2，0
IF E1%<0 THEN *ERROR
'------------------------------------------------------------
' Connect
'------------------------------------------------------------
SOCKCONNECT 2，IpAdd，Port，1
IF E1%<0 THEN *ERROR
'------------------------------------------------------------
'GET Position
'------------------------------------------------------------
V10! = SYSTEM!(810)                    'X
V11! = SYSTEM!(811)                    'Y
V12! = SYSTEM!(812)                    'Z
V13! = SYSTEM!(813)                    'Roll
V14! = SYSTEM!(814)                    'Pitch
V15! = SYSTEM!(815)                    'Yaw
'------------------------------------------------------------
'Send
'------------------------------------------------------------
V10$ =LEFT$(STR$(V10!)，  SendDataLen)
V11$ =LEFT$(STR$(V11!)，  SendDataLen)
V12$ =LEFT$(STR$(V12!)，  SendDataLen)
V13$ =LEFT$(STR$(V13!)，  SendDataLen)
V14$ =LEFT$(STR$(V14!)，  SendDataLen)
V15$ =LEFT$(STR$(V15!)，  SendDataLen)

V16$ =STR$(O[1])
V16$ =V16$ + STR$(O[2])
V16$ =V16$ + STR$(O[3])
V16$ =LEFT$(V16$，  SendIODataLen)

V17$ =V10$+"，"+V11$+"，"+V12$+"，"+V13$+"，"+V14$+"，"+V15$+"，"+V16$+"，"

SOCKSENDSTR 2，V17$，LEN(V17$)，0，V105%，0
IF E1%<0 THEN *ERROR
```

与服务器 PC 连接。

TCP 当前值保存到变量。

TCP 当前值保存到字符串变量。

通用输出信号的状态保存到字符串变量。

数据发送到 PC。

```
'------------------------------------------------------
' Receive
'------------------------------------------------------
SOCKRECV 2，1，ReceiveBufferLen，0，V110%
IF E1%<0 THEN *ERROR

GETSTR 1， V1$， ReceiveDataLen * 0， ReceiveDataLen
GETSTR 1， V2$， ReceiveDataLen * 1， ReceiveDataLen
GETSTR 1， V3$， ReceiveDataLen * 2， ReceiveDataLen
GETSTR 1， V4$， ReceiveDataLen * 3， ReceiveDataLen
GETSTR 1， V5$， ReceiveDataLen * 4， ReceiveDataLen
GETSTR 1， V6$， ReceiveDataLen * 5， ReceiveDataLen
'------------------------------------------------------
' Set Val
'------------------------------------------------------
V1! = VAL(V1$)
V2! = VAL(V2$)
V3! = VAL(V3$)
V4! = VAL(V4$)
V5! = VAL(V5$)
V6! = VAL(V6$)
'------------------------------------------------------
'Set Register
'------------------------------------------------------
R1=(V1!，V2!，V3!，V4!，V5!，V6!)
'x，y，z，roll(theta_z)，pitch(theta_y)，yaw(theta_x)

'------------------------------------------------------
' Close
'------------------------------------------------------
SOCKCLOSE 2
EXIT

*ERROR
WINDOW 0，0，200，100
PRINT #0，STR$(E1%)
PRINT #0，STR$(E2%)
SOCKCLOSE 2
PAUSE 3000
EXIT
```

从 PC 接收数据。

所接收数据保存到变量。

数据保存到移位寄存器。

关闭套接字。

若失败套接字通信的话，将执行错误处理。

5.1.7 PC 端程序（服务器）创建实例

```
#include <stdio.h>
#include <winsock2.h>
#include <iostream>
#include <string.h>

#define PORT_NUM 10030
#define SEND_DATA_SIZE 8
#define SEND_BUFFER_LEN (SEND_DATA_SIZE * 6)
#define REC_DATA_SIZE 12
#define REC_DATA_NUM 7
#define REC_IO_DATA_SIZE 3
#define REC_BUFFER_LEN (REC_DATA_SIZE * 6 + REC_IO_DATA_SIZE + REC_DATA_NUM)

int main(void){

//------------------------------------------------------
//                                  Socket Setup
//------------------------------------------------------
SOCKET sock0;
SOCKET sock;

//Socket information

struct sockaddr_in addr;
struct sockaddr_in client;

//Socket communication

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2，0)， &wsaData) != 0)
{
        std::cout << "WSAStartup failed" << std::endl;
        return 1;
}

//Make Socket

sock0 = socket(AF_INET， SOCK_STREAM， 0);
if (sock0 == INVALID_SOCKET)
{
        std::cout << "socket : " << WSAGetLastError() << std::endl;
        return 1;
}

//Set the information

addr.sin_port = htons(PORT_NUM);

addr.sin_family = AF_INET;

addr.sin_addr.S_un.S_addr = INADDR_ANY;
```

定义执行套接字通信时的常数。

与控制装置连接。

```
if (bind(sock0， (struct sockaddr *)&addr， sizeof(addr)) != 0)
{
        std::cout << "bind : " << WSAGetLastError() << std::endl;
        return 1;
}

//Communicate to the Client
if (listen(sock0， 5) != 0)
{
        std::cout << "listen : " << WSAGetLastError() << std::endl;
        return 1;
}

int len;

len = sizeof(client);
std::cout << "Waiting for connection ..." << std::endl;
sock = accept(sock0， (struct sockaddr *)&client， &len);
if(sock == INVALID_SOCKET)
{
        std::cout << "accept : " << WSAGetLastError() << std::endl;
        return 1;
}
```

与控制装置连接。

```
//-------------------------------------------------------
//                          Recieve from Client
//-------------------------------------------------------

char rec_buffer[REC_BUFFER_LEN] = {};
int check = 0;
char *temp = NULL;
char *ctx = NULL;
std::cout << "---------------- " << std::endl;
std::cout << "Current Position " << std::endl;
std::cout << "---------------- " << std::endl;

//Recieve
check = recv(sock， rec_buffer， sizeof(rec_buffer)， 0);
if(check < 1){
        std::cout << "recv : " << WSAGetLastError() << std::endl;
        return 1;
}
```

接收数据。

```
temp = strtok_s(rec_buffer，"，"，&ctx);
std::cout << "X        : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Y        : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Z        : " << temp << std::endl;
```

显示接收到的数据。

```
temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Roll   : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Pitch : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);
std::cout << "Yaw    : " << temp << std::endl;

temp = strtok_s(NULL，"，"，&ctx);

std::cout << "Out3_____ " << std::endl;
std::cout << "Out2____ |" << std::endl;
std::cout << "Out1___ ||" << std::endl;
std::cout << "           |||" << std::endl;
std::cout << "IO     : " << temp << std::endl;

//---------------------------------------------------------
//                                    Send to Client
//---------------------------------------------------------

float input = 0.0;
char shift_x[] = "00000.00";
char shift_y[] = "00000.00";
char shift_z[] = "00000.00";
char shift_r[]   = "00000.00";
char shift_p[]   = "00000.00";
char shift_ya[] = "00000.00";

std::cout << "Set X [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_x，"%08.2f"，input);

std::cout << "Set Y [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_y，"%08.2f"，input);

std::cout << "Set Z [mm]" << std::endl;
std::cin >> input;
sprintf_s(shift_z，"%08.2f"，input);

std::cout << "Set Roll [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_r，"%08.2f"，input);

std::cout << "Set Pitch [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_p，"%08.2f"，input);

std::cout << "Set Yaw [deg]" << std::endl;
std::cin >> input;
sprintf_s(shift_ya，"%08.2f"，input);
```

显示接收到的数据。

输入要发送的数据。

```
char buffer[SEND_BUFFER_LEN+1];

sprintf_s(buffer，"%s%s%s%s%s%s"，shift_x，shift_y，shift_z，shift_r，shift_p，shift_ya);

std::cout << "Send Data "    << std::endl;
std::cout << buffer    << std::endl;

//Send
check = send(sock，  buffer，  sizeof(buffer)，  0);

//---------------------------------------------------------
//                                    End Socket
//---------------------------------------------------------

closesocket(sock);

WSACleanup();

return 0;

}
```

发送数据。

关闭套接字。

NOTE

NACHI
株式会社 不二越

http://www.nachi-fujikoshi.co.jp/

总公司　东京都港区东新桥1-9-2　汐留住友大厦17层　邮编 105-0021
Tel +81-3-5568-5245 Fax +81-3-5568-5236

中国

那智不二越（上海）贸易有限公司
上海市普陀区丹巴路98弄7号　龙裕财富中心11层　邮编 200062
Tel 021-6915-2200　Fax 021-6915-5427

重庆分公司
重庆市江北区红鼎国际名苑C座17-18, 17-19　邮编 400020
Tel 023-8816-1967　Fax 023-8816-1968

沈阳分公司
辽宁省沈阳市沈河区悦宾街1号方圆大厦304室　邮编 110000
Tel 024-3120-2252　Fax 024-2250-5316

北京分公司
北京市朝阳区朝外大街乙12号 昆泰国际大厦 0-1110室　邮编 100020
Tel 010-5879-0181　Fax 010-5879-0182

长春事务所
长春市绿园区普阳街1688号长融大厦B座707室　邮编 130061
Tel 0431-8507-8700　Fax 0431-8507-8701

广州事务所
广州市番禺区东环路431号港信城B座505室　邮编 510120
Tel 020-2293-9503　Fax 020-2293-9503

那智不二越（江苏）精密机械有限公司
江苏省张家港市经济技术开发区(南区)南园路39号　邮编 215618
Tel 0512-3500-7616　Fax 0512-3500-7615

上海不二越精密轴承有限公司
上海市嘉定区马陆镇丰茂路258号易通工业园　邮编 201801
Tel 021-6915-6200　Fax 021-6915-6202

耐锯（上海）精密刀具有限公司
上海市嘉定区马陆镇丰茂路258号易通工业园　邮编 201801
Tel 021-6915-5899　Fax 021-6915-5898

东莞建越精密轴承有限公司
东莞市洪梅镇凼涌村
Tel 0769-8843-1300　Fax 0769-8843-1330

机器人事业部
富山市不二越本町1-1-1, JAPAN 邮编930-8511
Tel +81-76-423-5137
Fax +81-76-493-5252