1.2 Binary and character based communication

1.2.1. Overview

Purpose

The purpose of binary and character based communication is to:

- store information in a remote memory or on a remote disk
- let the robot communicate with other devices

What is included

To handle binary and character based communication, the RobotWare base functionality File and Serial Channel Handling gives you access to:

- instructions for manipulations of a file or serial channel
- instructions for writing to file or serial channel
- instruction for reading from file or serial channel
- functions for reading from file or serial channel.

Basic approach

This is the general approach for using binary and character based communication. For a more detailed example of how this is done, see *Code examples on page 14*.

1. Open a file or serial channel.
2. Read or write to the file or serial channel.
3. Close the file or serial channel.

Limitations

Access to files, serial channels and field busses cannot be performed from different RAPID tasks simultaneously. Such an access is performed by all instruction in binary and character based communication, as well as `WriteRawBytes` and `ReadRawBytes`. E.g. if a `ReadBin` instruction is executed in one task, it must be ready before a `WriteRawBytes` can execute in another task.

1.2.2. RAPID components

Data types

This is a brief description of each data type used for binary and character based communication. For more information, see the respective data type in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Data type	Description
iodev	`iodev` contains a reference to a file or serial channel. It can be linked to the physical unit with the instruction `Open` and then used for reading and writing.

Instructions

This is a brief description of each instruction used for binary and character based communication. For more information, see the respective instruction in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Instruction	Description
Open	`Open` is used to open a file or serial channel for reading or writing.
Close	`Close` is used to close a file or serial channel.
Rewind	`Rewind` sets the file position to the beginning of the file.
ClearIOBuff	`ClearIOBuff` is used to clear the input buffer of a serial channel. All buffered characters from the input serial channel are discarded.
Write	`Write` is used to write to a character based file or serial channel.
WriteBin	`WriteBin` is used to write a number of bytes to a binary serial channel or file.
WriteStrBin	`WriteStrBin` is used to write a string to a binary serial channel or file.
WriteAnyBin	`WriteAnyBin` is used to write any type of data to a binary serial channel or file.
ReadAnyBin	`ReadAnyBin` is used to read any type of data from a binary serial channel or file.

Functions

This is a brief description of each function used for binary and character based communication. For more information, see the respective instruction in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Function	Description
ReadNum	`ReadNum` is used to read a number from a character based file or serial channel.
ReadStr	`ReadStr` is used to read a string from a character based file or serial channel.
ReadBin	`ReadBin` is used to read a byte (8 bits) from a file or serial channel. This function works on both binary and character based files or serial channels.
ReadStrBin	`ReadStrBin` is used to read a string from a binary serial channel or file.

1.2.3. Code examples

Communication with character based file

This example show writing and reading to and from a character based file. The line "The number is :8" is written to FILE1.DOC. The contents of FILE1.DOC is then read and the output to the FlexPendant is "The number is :8" followed by "The number is 8".

```
PROC write_to_file()
  VAR iodev file;
  VAR num number:= 8;

  Open "HOME:" \File:= "FILE1.DOC", file;
  Write file, "The number is :"\Num:=number;
  Close file;
ENDPROC

PROC read_from_file()
  VAR iodev file;
  VAR num number;
  VAR string text;

  Open "HOME:" \File:= "FILE1.DOC", file \Read;
  TPWrite ReadStr(file);
  Rewind file;
  text := ReadStr(file\Delim:=":");
  number := ReadNum(file);
  Close file;
  TPWrite text \Num:=number;
ENDPROC
```

Communication with binary serial channel

In this example, the string "Hello", the current robot position and the string "Hi" is written to the binary serial channel com1.

```
PROC write_bin_chan()
  VAR iodev channel;
  VAR num out_buffer{20};
  VAR num input;
  VAR robtarget target;

  Open "com1:", channel\Bin;

  ! Write control character enq
  out_buffer{1} := 5;
  WriteBin channel, out_buffer, 1;

  ! Wait for  control character ack
  input := ReadBin (channel \Time:= 0.1);
  IF input = 6 THEN
```

Continues on next page

Continued

```
      ! Write "Hello" followed by  new line
      WriteStrBin channel, "Hello\0A";

      ! Write current robot position
      target := CRobT(\Tool:= tool1\WObj:= wobj1);
      WriteAnyBin channel, target;

      ! Set start text character (2=start text)
      out_buffer{1} := 2;

      ! Set character "H" (72="H")
      out_buffer{2} := 72;

      ! Set character "i"
      out_buffer{3} := StrToByte("i"\Char);

      ! Set new line character (10=new line)
      out_buffer{4} := 10;

      ! Set end text character (3=end text)
      out_buffer{5} := 3;

      ! Write the buffer with the line "Hi"
      ! to the channel
      WriteBin channel, out_buffer, 5;

    ENDIF
    Close channel;
  ENDPROC
```

In this example, the same sequence as above is read from the channel com2.

```
  PROC read_bin_chan()
    VAR iodev channel;
    VAR string text;
    VAR num bindata;
    VAR robtarget target;

    Open "com2:", channel \Bin;

    ! Clear input buffer for com2
    ClearIOBuff channel;

    ! Wait for input from com2 and then
    ! write the first 5 characters to FlexPendant
    TPWrite ReadStrBin (channel, 5);
```

Continues on next page

1.2.3. Code examples

Continued

```
    ! Read the new line character
    ReadStrBin channel, 1;

    ! Read the next input, interpreted as  robtarget
    ! and move robot to that target
    ReadAnyBin channel, target;
    MoveJ target, vmax, fine, tool1;

    ! Read text one character at the time
    ! until end of file
    bindata := ReadBin(channel);
    WHILE bindata <> EOF_BIN DO
       text := text + ByteToStr(bindata\Char);
       bindata := ReadBin(channel);
    ENDWHILE
    TPWrite text;
    Close channel;
ENDPROC
```