Device Module | Detailed functionality

Detailed Functionality

Communication protocols exists so that there can be communication between two entities. Here in our case, we are talking about FactoryStudio and other external devices used in the field; mainly PLCs, but they can be other types of devices or software.

Each device manufacturer and/or creator of a communication protocol defines the rules of communication messages. These messages are called TX and RX. TX messages are the messages that go from FactoryStudio to the device, and the RX messages are the messages that go from the device to FactoryStudio.

Each entity related to this “conversation” needs to know the communication protocol to understand what the TX and RX messages mean. Each byte within the message has a meaning. For example: Let's consider a fictitious communication protocol definition:

Protocol: [STX] [CMD] [OPR] [STA] [CNT] <DATA> [ETX]

Where,

[STX] Start Message = 02 hexa

[CMD] Command : 04 hexa – Read Action, 05 - hexa Write Action, 06 – Ack Answer

[OPR] Operand : 31 hexa – Integer (2bytes) , 32 hexa– Float (4bytes)

[STA] Start Address – Beginning address to read or write (two bytes)

[CNT] Counter – Quantity of bytes in the data packet to read or write (one byte)

<data> - data bytes related to the content read or write (each operand can have 2 or 4 bytes)

[ETX] End Message — 0D hexa

Based on this definition, we could say that the TX and RX below are valid:

TX: 02 05 31 00 00 04 0D (FS send to Device)

RX: 02 06 31 00 00 04 00 00 00 00 0D (Device answer to FS)

This means - TX:

02[STX]

05[CMD RD]

31 [Integer]

00 00 [Start Address : 0]

04[Quantity: 4 bytes]

0D [ETX]

This means - RX:

02 [STX]

06 [ACK ANSWER]

31 [Integer]

00 00 [Start Address : 0]

04 [Quantity: 4 bytes (two operands integer)]

00 00 [Operand 1 with data value 0]

00 00 [Operand 2 with data value 0]

0D [ETX]

After analyzing this fictitious protocol, we can conclude: Since the CNT is a single byte that indicates the number of bytes and the FF hex means 255 bytes, a block of communication can have a maximum of 255 bytes. Thus, each TX / RX can transmit a maximum of 127 (integer) or 64 (floats), which are the operands limits per communication block. The CNT, or 255 in this case, is what we call the MaximumBlockSize, which is defined as the maximum size of the communication block. As defined by the communication protocol, the user generally cannot customize the MaximumBlockSize.

Since a device's available operands can be much larger than the MaximumBlockSize, multiple TX/RX exchanges can be required for it to read or write on all the data.

In a TX/RX, a read dataset is called a ReadGroup. Each ReadGroup contains information regarding FS Tags and communication operands. Consequently, the amount of operands must be less than the MaximumBlockSize.

ReadGroups and WriteGroups are created during the device module startup. Also, the Points settings are sorted, and the Groups are sequentially created by taking into account the information of whether or not two different points can be in the same communication TX/RX. The information taken into account is: Node, Address (Operand, Address, Name), MaximumBlockSize, Read/Write Command, Polling Time.

Once the ReadGroups and WriteGroups are created and when the read or write event happens, the group is placed in a run queue. It is never queued more than once; it must run and exit the queue so that it can be queued again.

This execution queue is emptied by the execution of the communication driver. The communication driver takes the information the TX sends to the device, waits for the RX response, handles the data, assigns the values in the Tags if necessary, and gives the execution as complete. ******

It is important to realize that there is a wait time for the the RX response throughout this cycle. The wait time does not depend on FactoryStudio's execution; it depends on the device's execution, which can take more or less time. (In general, a PLC program has an execution cycle time that directly influences the time of treatment and response of the communication. This means a PLC with a longer cycle time will take longer to respond to an RX.)

If we imagine that this entire cycle takes around 20ms, we will have 50 TX/RX exchanges in 1s. By taking into account the maximum amount of operands for TX/RX, we know how many operands we can exchange in 1s.

This is the traditional form of communication for Master/Slave. However, many devices that accept TCPIP communication also accept multi master or multi connections, which means that more than one master can send a message at the same time. In FactoryStudio, this is done through the NodeConnections configuration. The NodeConnections is the amount of connections that each node will establish with the device for simultaneous TX/RX exchange.

In this case, instead of the PLC having to handle only one message per execution cycle, it will have to handle multiple messages, which can cause an increase in response time for each RX. Although we generally we have a performance gain almost proportional to the number of node connections, it is important to remember that devices have a maximum number of connections allowed, ranging from equipment to equipment.

The most optimized form of communication is the configuration of points with sequential and continuous addresses because all of the byes belonging to the databuffer are used. When we set up points with address jumps, there is a waste of communication. For example: Read address 0, 10, 50, and 100. We will read only 4 operands in a buffer exchanged with 100 operands; this is a great waste of communication.

Once we know the criteria for generating a ReadGoup, we can use a new Node or a different pooling time to force the operand to be separate from the previous ReadGroup. In some cases, mainly serial communication, we can decrease the amount of bytes in communication by creating more groups with smaller sizes. In general, this optimization does not make much sense for TCPIP communication because the speed of the communication makes the number of groups end up having greater influence than the size of the groups.

Another important optimization is for when we have a large volume of communication groups and want to give priority to some of them. In this case, as long as the communication is TCPIP and the device accepts multiple connections, you must create a channel so that a smaller group of points communicate more frequently. If this is not done, these points will be together with other groups in the execution queue. Remember that the same group cannot be in the queue more than once.

Sometimes, communication is not saved in the history and calculations are only displayed on the screens instead of being performed with the read field value. This type of data does not need to be read all the time, it can be read only when the screen that uses it is open. There is a setting in the AccessType ReadOnDisplay that enables the communication group if any data is on the screen. It is important to understand this operation in detail as it may not be very intuitive.

At device module startup, the ReadGroups and WriteGroups groups are created by taking into account all configured communication points, as already described earlier. At this point, we do not know if the AccessType configuration is ReadOnDisplay or Always; we simply create all the groups taking into account the factors already described in this document.

When a screen opens during execution, we check to see if there are points with the ReadOnDisplay setting and if the points are linked to the tags of the open screens. Points that have the ReadOnDisplay AccessType and that are not on the open screens are disabled, and the group related to it is only disabled if all of the configured points are also disabled (which may be enabled because they are the Always AccessType instead of ReadOnDisplay). This is important because the way groups are assembled is important when we think of the relationship of having only screen data or not. A group that always has a specific screen point along with an Always data will never be disabled, sparing no communication.