General Electric SRTP protocol

General Electric SRTP protocol (Service Request Transport Protocol)

Supported device types and versions
Communication line configuration
Line protocol parameters
Communication station configuration
I/O tag configuration
Notes on Fanuc Robot R-30iA/R-30iB
Literature
Changes and modifications
Document revisions

Supported device types and versions

The protocol is an implementation of the "Service Request Transport Protocol" developed by General Electric Automation and Controls (formerly GE Fanuc) for communication with PLCs. Almost all GE devices equipped with Ethernet port support GE SRTP. GE SRTP is a successor to the serial protocols SNP and SNPX for Ethernet media.

The protocol was tested against the Fanuc Robot R-30iB:

- Reading values from memory types %I, %IB, %Q, %QB, %M, %MB, %G, %GB, %AI, %AQ, %R, %RD, %RF worked.
- Writing values to the memory types %R, %RD, %RF worked. Other types that could be read, returned success on write. However, the next reading returned the original value.
- Both reading and writing of text values worked (e.g. string registers or comments to registers - see Notes on Fanuc Robot R-30iA/R-30iB.

Communication line configuration

- Communication line category: TCP/IP-TCP.
- The IP address is set according to a network configuration of a specific GE device.
  Note: Multiple IP addresses of the device can be configured (separated by commas or semicolons).
- The port number is 18245 by default.
- The line number is not used, set to 1.

Line protocol parameters

A dialog window of communication line configuration - **Protocol parameters** tab.
They influence some optional protocol parameters.

The following line protocol parameters are defined:

Parameter	Meaning	Unit / Size	Default value
Maximum Payload	Maximum size (in bytes) of data read by a single request. I/O tags whose addresses are consecutive will be grouped into reading requests so that the size of the response data does not exceed the configured value.	1 až 1024 B	100 B
Cycle Delay	Waiting after reading of all I/O tags on a station. This parameter allows you to set read cycles less than 1 second (the time parameters configured on the station have a resolution of 1 second).	ss.mss	100 ms
Debug Values	Activates debug info about the read/written values of I/O tags. Use this parameter only when communication must be debugged because it highly utilizes CPU and slows down the communication.	YES /NO	NO

Communication station configuration

- Communication protocol: **General Electric SRTP**.
- No address is specified, only a single device can communicate via one communication line.
  Note: There can be more stations on one line e.g. due to different time parameters (division into fast and slow I/O tags).

I/O tag configuration

Possible I/O tag types: **Ai, Ci, Di, TiR,TxtI, Ao, Co, Dout, ToR, TxtO**

I/O tag address has the format [+]*MemoryArea Position[:StringLength] [Count]*

- The "+" character is used for memory areas containing integer values and it means that the value will be interpreted as an Unsigned number (8, 16, or 32-bit unsigned integer).
  If the "+" character is not found, the value is interpreted as a Signed number (8, 16, or 32-bit signed integer).
  For memory areas containing bits or floating-point values, the "+" character is ignored.

- *MemoryArea* specifies the type of memory:

MemoryArea	Memory type description	Data size
%IGNORE	I/O tag will be ignored. Note: arbitrary character can follow after %IGNORE e.g. %IGNORE +%R1	-
%I	InputBit (Discrete inputs, bit access-mode)	1 bit
%IB	InputByte (Discrete inputs, byte access-mode)	1 byte
%Q	OutputBit (Discrete outputs, bit access-mode)	1 bit
%QB	OutputByte (Discrete outputs, byte access-mode)	1 byte
%T	TemporaryBit (Temporary references, bit access-mode)	1 bit
%TB	TemporaryByte (Temporary references, byte access-mode)	1 byte
%M	MemoryBit (Internal references, bit access-mode)	1 bit
%MB	MemoryByte (Internal references, byte access-mode)	1 byte
%S	SBit (System status references, bit access-mode)	1 bit
%SSB	SByte (System status references, byte access-mode)	1 byte
%SA	SABit (System status references A, bit access-mode)	1 bit
%SAB	SAByte (System status references A, byte access-mode)	1 byte
%SB	SBBit (System status references B, bit access-mode)	1 bit
%SBB	SBByte (System status references B, byte access-mode)	1 byte
%SC	SCBit (System status references C, bit access-mode)	1 bit
%SCB	SCByte (System status references C, byte access-mode)	1 byte
%G	GlobalBit (Discrete globals, bit access-mode)	1 bit
%GB	GlobalByte (Discrete globals, byte access-mode)	1 byte
%AI	AnalogInput (Analog input registers, byte access-mode)	2 bytes
%AQ	AnalogInput (Analog output registers, byte access-mode)	2 bytes
%R	Registers (System register reference, byte access-mode)	2 bytes
%RD	RegistersDouble (System register reference, 2 registers as a double word)	4 bytes
%RF	RegistersFloat (System register reference, 2 registers as a floating-point)	4 bytes
%L	LocalSubblock (Local registers, byte access-mode)	2 bytes
%P	ProgramBlockData (Program registers, byte access-mode)	2 bytes

  Note: the order of bytes and words is little-endian: B1 B2 for 2-byte types, B1 B2 B3 B4 for 4-byte types.

- *Position* defines the position of the object within the memory type. It is a positive 16-bit number (1-65535).
  Note: only positions 1-65534 are valid for the %RD and %RF memory types, as this is merely an integer/float interpretation of two consecutive registers in the %R memory area.

- *StringLength* specifies the length of the string in characters. This parameter is only allowed for memory type %R. Each register is interpreted as a 2-character string (little-endian format). For example, an I/O tag with address *%R1000:4* will read a 4-character string from registers 1000 and 1001.

- *Count* specifies the number of objects. This parameter only makes sense if the Destination Column is configured. It specifies the number of objects to be read and written to the structure.

- address examples:
  %R 1 20
  +%R 100
  +%AI25
  %AQ1 5
  %IB 12
  %R12111:80

Notes on Fanuc Robot R-30iA/R-30iB

According to FANUC Robot series R-J3/R-J3iB/R-30iA CONTROLLER CIMPLICITY HMI for Robots OPERATOR'S MANUAL, chapter 6 - *ADDRESS ASSIGNMENT TO POINTS*, subchapter 6.1 - *READING AND WRITING I/O SIGNALS (%I, %Q, %M, %AI, %AQ)* different types of I/O signals are mapped to SRTP protocol variables as follows:

Robot controller I/O signal	PLC address	Example
Digital Input DI[x]	%Qx	DI[1] <=> %Q1
Digital Output DO[x]	%Ix	DO[1] <=> %I1
Robot Input RI[x]	%Q(5000+x)	RI[1] <=> %Q5001
Robot Output RO[x]	%I(5000+x)	RO[1] <=> %I5001
UOP Input UI[x]	%Q(6000+x)	UI[1] <=> %Q6001
UOP Output UO[x]	%I(6000+x)	UO[1] <=> %I6001
SOP Input SI[x]	%Q(7000+x)	SI[0] <=> %Q7001
SOP Output SO[x]	%I(7000+x)	SO[0] <=> %I7001
Weld Input WI[x]	%Q(8000+x)	WI[0] <=> %Q8001
Weld Output WO[x]	%I(8000+x)	WO[0] <=> %Q8001
Wire Stick Input WSI[x]	%Q(8400+x)	WSI[0] <=> %Q8400
Wire Stick Output WSO[x]	%I(8400+x)	WSO[0] <=> %Q8400
Group Input GI[x]	%AQx	GI[1] <=> %AQ1
Group Output GO[x]	%AIx	GO[1] <=> %AI1
Analog Input AI[x]	%AQ(1000+x)	AI[1] <=> %AQ1001
Analog Output AO[x]	%AI(1000+x)	AO[1] <=> %AI1001
PMC keep relay DO[x] (x : 10001 – 10144) Ka.b	 %Ix %I((a*8)+b+10001)	 DO[10001] <=> %I10001 K2.5 <=> %I10022
PMC internal relay DO[x] (x : 11001 – 23000) Ra.b	 %M(x-11000) %M((a*8)+b+1)	 DO[11001] <=> %M1 R2.5 <=> %M22
PMCdata table GO[x] (x : 10001 – 12000) D(a*2), D((a*2)+1)	 %AI(x-6000) %AI(a+4001)	 GO[10001] <=> %AI4001 D4, D5 <=> %AI4003

According to the same document, subchapter 6.2 - *READING FROM AND WRITING TO REGISTERS (%R),* the standard mapping of registers from the robot controller to PLC addresses is as follows:

Robot controller data	PLC address	Example
Register	%Rx	R[1] <=> %R1

Values in registers are 16-bit signed numbers, the fractional part is rounded off.
The documentation further states that this standard mapping can be changed using the **$SNPX_ASG** system variable. This allows you to set up the multiplier for conversion and modify the mapping of registers. One register can be mapped to one %R variable (as 16-bit signed number), to two consecutive %R variables (32-bit signed number, use **%RD** in I/O tag address), or to two consecutive %R variables as a real number (32-bit float number, use **%RF** in I/O tag address).

The default mapping of R registers is related to the setting of $SNPX_ASG[1] to:

Item	Value
$ADDRESS	1
$SIZE	10000
$VAR_NAME	'R[1]@1.1'
$MULTIPLIFY	1.000

With this setting, the registers R[1] .. R[10000] are mapped to registers %R1 .. %R10000 as 16-bit signed numbers.

The standard mapping of PR position registers (intended for storing robot positions) ensures the setting of $SNPX_ASG[2]:

Item	Value
$ADDRESS	11000
$SIZE	100
$VAR_NAME	'PR[1]'
$MULTIPLIFY	0.000

With this setting, the individual position components (X, Y, Z, W, P, R, etc.) will be mapped as 32-bit float numbers at the addresses %RF11000, %RF11002, %RF11004, and so on (50 registers per one PR, a total of 100 registers per PR[1] and PR[2]). For more details see subchapter *6.3 READING FROM AND WRITING TO POSITION REGISTERS (% R)*.

The standard mapping of current position registers POS ensures the setting of $SNPX_ASG[3]:

Item	Value
$ADDRESS	12000
$SIZE	100
$VAR_NAME	'POS[0]'
$MULTIPLIFY	0.000

With this setting, the individual position components (X, Y, Z, W, P, R, etc.) will be mapped as 32-bit float numbers at the addresses %RF12000, %RF12002, %RF12004, and so on (50 registers per one POS, a total of 100 registers per POS[0] and POS[1]). For more details see subchapter *6.3 READING AND WRITING THE CURRENT POSITION (%R)*.

The mapping of ALM registers containing alarm history is possible by setting $SNPX_ASG[x]:

Item	Value
$ADDRESS	12100
$SIZE	300
$VAR_NAME	'ALM[1]'
$MULTIPLIFY	1.000

With this setting, the individual components of the alarm history (AlarmID, Alarm number, Alarm severity, Alarm message, etc) will be accessible at the configured addresses (100 registers per ALM, i.e. 300 registers will cover 3 alarms). For more details see subchapter *6.5 READING ALARM HISTORY (% R)*.

The mapping of PRG registers containing program execution status is possible by setting $SNPX_ASG[x]:

Item	Value
$ADDRESS	12400
$SIZE	18
$VAR_NAME	'PRG[1]'
$MULTIPLIFY	1.000

With this setting, the individual components of program execution status (Program name, Line number, Execution status, Calling program name) will be accessible at the configured addresses (18 registers per PRG). For more details see subchapter *6.6 READING THE PROGRAM EXECUTION STATUS (% R).*

The mapping of various system variables is also possible by setting $SNPX_ASG[x]. Example:

Item	Value
$ADDRESS	12420
$SIZE	2
$VAR_NAME	'$mcr.$genoverride'
$MULTIPLIFY	1.000

With this setting, the registers 12420 and 12421 will contain the OVERRIDE system variable (item $genoverride of system variable $mcr) as a 32-bit signed integer (address %RD12420, read/write access). For more details see subchapter *6.6 READING FROM AND WRITING INTO SYSTEM VARIABLES (%R).*

The mapping of comments to registers, position registers, and various I/O is also possible by setting $SNPX_ASG[x]. Example:

Item	Value
$ADDRESS	12440
$SIZE	160
$VAR_NAME	'R[C1]'
$MULTIPLIFY	1.000

With this setting, the registers %R12420 to %R12599 will contain comments to registers R1 to R4 (40 registers, i.e. 80 characters per comment - addresses %R12440:40, %R12480:40, %R12520:40 and %R12560:40). Similarly, it is possible to configure comments for other types of objects (e.g. 'PR[C1]', 'DI[C2]', 'GI[C1]', 'GO[C1]' etc). For more details see subchapter *6.8 READING AND WRITING THE COMMENT OF REGISTERS, POSITION REGISTERS, AND I/O (%R).*

The mapping of I/O values and simulation status is also possible by setting $SNPX_ASG[x]. Example:

Item	Value
$ADDRESS	12430
$SIZE	4
$VAR_NAME	'RO[S1]'
$MULTIPLIFY	1.000

With this setting, the registers %R12430 to %R12433 will contain the status of robot outputs RO[1] to RO[4] simulation. Similarly, it is possible to configure simulation status for other types of objects (e.g. 'DI[S1]', 'DO[S2]', 'DO[S1]', 'RI[S1]' etc) as well as values of these objects (e.g. 'DI[1]', 'DI[2]', 'UO[1]', 'SI[1]' at). For more details see subchapter *6.9 READING AND WRITING THE VALUE AND SIM STATUS OF I/O (%R).*

Literature

https://en.wikipedia.org/wiki/Service_Request_Transport_Protocol

FANUC Robot series R-J3/R-J3iB/R-30iA CONTROLLER CIMPLICITY HMI for Robots OPERATOR'S MANUAL (R-30iA CONTROLLER CIMPLICITY HMI for Robots_[B-82604EN_01].pdf)

Blog

You can read blogs about General Electric SRTP protocol:

- Communication - Fanuc robots
- Communication – Fanuc robots, part 2

Changes and modifications

-

Document revisions

- Ver. 1.0 - February 12th, 2020 - document creation.

Related pages:

Communication protocols