



V19.4.5  
Drivers

eScada.Drivers.AllenBradleyPCCC

**eScada.Drivers.AllenBradleyPCCC**

( PCC protocol – PLC5 addressing – Connected CIP Transport )

**OS availability**

Windows, Linux, RaspBian

**Atomic data type**

16 bit Word or 32 bit floating point oriented protocol.

**Hardware and documentation reference**

www.rockwellautomation.com

Delivery of CIP Over RA Serial DF1 Links (Rev. 1.1, 12-Oct-06)

**Parameters available in every section**

Channel: none

Device:	Node network	Ethernet/CNET
	DH+ Channel	DH+ (*)
	IP address	* Channel A, Channel B
	TCP Port	It can be IPV4
	Node path	Multiple addresses can be expressed separated using , (comma) e.g. 192.168.1.10,192.168.1.11
	Session Serial Number	A valid TCP port number.
	Request Packet Interval	Destination node path.
	Reconnect timeout [ms]	(only standard path 1,0 has been tested, even though other longer paths should work)
		It shall be a unique number for the connected device
		It must be greater than the maximum polling time
		Waiting time before a re-connection after COMM break-down

Group: none

Tag: none

**Remarks for devices**

The following attributes can be expressed for every device.

Bytes order actions      None, Swap bytes (little endians ↔ big endians adjustment)

String actions            None, Swap bytes in words

**Implemented data file**

Mainly this protocol is optimized using array of integers.

<b>O</b>	Output (16 bit)	<b>O:3:0</b>	<b>Array</b> – yes
<b>I</b>	Input (16 bit)	<b>I:1:0</b>	<b>Array</b> – yes
<b>S</b>	Status (16 bit)	<b>S2:7</b>	<b>Array</b> – yes
<b>ST</b>	Strings (82 bytes)	<b>ST20:0</b>	<b>Array</b> – yes
<b>A</b>	Ascii (2 bytes)	<b>A21:0</b>	<b>Array</b> – yes
<b>B</b>	Binary (16 bit)	<b>B4:0</b>	<b>Array</b> – yes
<b>N</b>	Integer (16 bit)	<b>N5:0</b>	<b>Array</b> – yes
<b>F</b>	Floating point (32 bit)	<b>F14:0</b>	<b>Array</b> – yes
<b>T</b>	Timers structures		
	16 bit	T4:0.PRE	Array – no, single item only
	16 bit	T4:0.ACC	Array – no, single item only
	bit	T4:0.EN	Array – no, single item only
	bit	T4:0.TT	Array – no, single item only
	bit	T4:0.DN	Array – no, single item only
<b>C</b>	Counters structures		
	16 bit	C4:0.PRE	Array – no, single item only
	16 bit	C4:0.ACC	Array – no, single item only
	bit	C4:0.CU	Array – no, single item only
	bit	C4:0.CD	Array – no, single item only
	bit	C4:0.DN	Array – no, single item only
	bit	C4:0.OV	Array – no, single item only
	bit	C4:0.UN	Array – no, single item only
	bit	C4:0.UA	Array – no, single item only
<b>R</b>	Control structures		
	16 bit	C4:0.LEN	Array – no, single item only
	16 bit	C4:0.POS	Array – no, single item only
	bit	C4:0.EN	Array – no, single item only
	bit	C4:0.EU	Array – no, single item only
	bit	C4:0.DN	Array – no, single item only
	bit	C4:0.EM	Array – no, single item only
	bit	C4:0.ER	Array – no, single item only
	bit	C4:0.UL	Array – no, single item only
	bit	C4:0.IN	Array – no, single item only
	bit	C4:0.FD	Array – no, single item only

**Remarks for single bit addressing**

This kind of addressing B9:7/7 is supported but only for single bit.

We suggest to use entire words with tag of bit data type instead.

This approach can be used for **O**, **I**, **B** or **N** areas.

**Addressing**

Variable type	Type	PLC type	Items
<b>Boolean</b> The number of items used declaring TAGs, must be multiples of 16 bit. Every group of boolean, starts from the first bit of its word. T, C and R file can be addressed using single bit only			
Single bit	Bit	O, I, S, B, N, T, C, R	(C)
<b>Byte</b> The number of items used declaring TAGs, must be multiple of 2 bytes. Every group of bytes, starts from the first byte of its word.			
Unsigned 8 bit	UInt8	O, I, S, B, N, A	(C)
Signed 8 bit	Int8		
<b>16 bit</b>			
Unsigned integer 16 bit	UInt16	O, I, S, B, N, A, T, C, R	(C)
Signed integer 16 bit	Int16		
<b>32 bit</b>			
Unsigned integer 32 bit	UInt32	O, I, N, F	(C)
Signed integer 32 bit	Int32		
Single precision 32 bit - ( IEEE 754 )	Float		
<b>64 bit</b>			
Unsigned integer 64 bit	UInt64	N, F	(C)
Signed integer 64 bit	Int64		
Double precision 64 bit - ( IEEE 754 )	Double		
<b>Strings</b> The string length used declaring TAGs, must be a multiples of 2 String bytes can be interpreted as ASCII, UTF-7, UTF-8, UTF-16 or UTF-32 encoding			
Array of bytes	String	N, A	(A, C)
Array of bytes. (Siemens S7) Array of bytes. (AllenBradley style)	S7String ABString	N, A, ST	(B, C)
<b>(A)</b> It depends on the string's length: e.g. if you want to read strings with a length of 10 chars each string, you can set a number of items of $74 / 10 = 7$ consecutive items.			
<b>(B)</b> It depends on the string's length: e.g. if you want to read strings with a length of 10 chars each string, you can set a number of items of $74 / (10+2) = 6$ consecutive items.			
<b>(C)</b> It depends on PLC model. The best way is to try with the maximum items you need.			

**S7 strings format**

They have got two bytes at the beginning.

The first byte is for max allowed string length, the second one is for the real string length.

These types of strings can be declared with a length of 255 bytes max.

**AB Strings format**

They have got one word (16 bit) at the beginning, it contains the string length.

**Consecutive items**

The number of consecutive read/write items, depends on the PLC model.

Please review 'Implemented data types' to better understand which types of basic object can be addressed using array of items.