

ASTRA

S5 Simatic Device Driver
www.renuelectronics.com

⌘ Tables of Contents ⌘

Preface	3
1. Introduction	3
2. Technical and Communication Details	4
3. Data Types and Addressing	5
4. Notes	10
5. Optimisations	11
6. Errors	11
7. Limitations	27

⌘ Preface ⌘

This document introduces you to [SIEMENS S5-SERIES PLC](#) device driver. This documents give you a broad idea of how to use S5 device driver with Astra. All the chapters, broadly, tell you about the capabilities and technical details of Matsushita device driver and how to use the driver.

⌘ Introduction ⌘

The intent of this document is to assist users of [SIEMENS S5-SERIES](#) driver in conjunction with the Astra MMI software package. A general knowledge of [SIEMENS S5-SERIES PLC](#) family is assumed. The addressing scheme is that of the SIEMENS programming software with some slight modifications which are explained. The description of different data types and the addressing scheme should be understood before attempting to use the driver in a Astra project.

The driver is intended to be used only on one PC serial port at a time and connected to only one PLC (as the SIMATIC S5 is a Point protocol).

The optimization features described in this document can improve performance, but they are not essential for use.

⌘ Technical & Communication Details ⌘

PLC Make :	_____
PLC Modles :	The driver supports the following PLC models – 90U, 95U, 100U, 102U, 103U, 115U.
PLC Memory :	BYTE Memory.
Communication Protocol :	SIMATIC S5 (Half Duplex).
Communication Parameters :	
Baud Rate -	9600
Parity -	ODD
Data Bits -	8
Stop Bits -	1
Cable Connections :	S5U PLC requires RS-485 to RS232 converter for serial communication. Any standard converter could be used.
Node ID :	Any Node ID can be used.

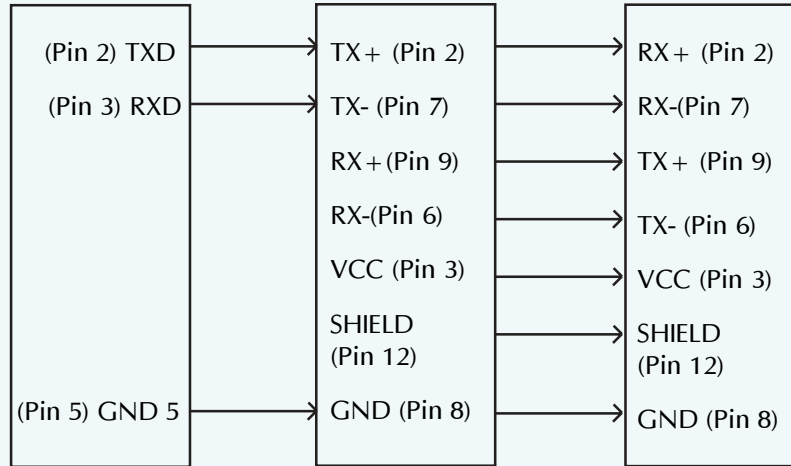
⌘ Technical & Communication Details ⌘

RS232:

PC side D type
9 pin male/female

SCAB smart cable
for S5U

S5U side



⌘ Data Types and Addressing ⌘

Data Types :

This section will describe the way S5 driver interprets the information from the PLC as different data types. The PLC programmer is responsible for ensuring that the referenced locations can logically be interpreted as correct type. This is particularly important for floating point numbers, as there are such bit configurations, that are incompatible with the IEEE floating point format.

All 16 bit words and 32 bit double words must start on 8 bit boundary for the Inputs, Outputs and Flags, and it should start on a 16 bit boundary for the Timers, Counters and Data Words. It is possible to overlap double words using this format. Say, that the Data Words – D002.000 and D002.001, both are defined as data type long, so they would share the 16 bit word at location D002.001, as either their high word or low word respectively. Since, this is probably not the desirable behavior, care should be taken to avoid such overlapping situations.

The address usage is as follows:

Tag Type	Example Address In ASTRA	Actual Addresses fetched
Discrete	I0.0 Q8.4 F15.7	I0.0 – 1 st bit from the LSB of the 0 th Input Register Byte. Q8.4 – 5 th bit from LSB of the 8 th Output Register Byte. F15.7 – 8 th bit from LSB of the 15 th Flag Register Byte.
Unsigned Integer or Integer	I8 Q15 F24 T5.V T5.M C6 D002.000 F100.B	I8 – High Byte I9 – Low Byte Q15 – High byte Q16 – Low Byte F24 – High Byte F25 – Low Byte T5 – 5 th Timer Word's base value. T5 – 5 th Timer Word's base multiplier in milliseconds. C6 – 6 th Counter Word Doo2.000 – 0 th Data Word in the 2 nd Data block Flag byte 100.

⌘ Data Types and Addressing ⌘

Tag Type	Example Address In ASTRA	Actual Addresses fetched
Large Integer or Real	I8	I8 – High Byte of High Word I9 – Low Byte of High Word I10 – High Byte of Low Word I11 – Low Byte of Low Word
	Q15	Q15 – High Byte of High Word Q16 – Low Byte of High Word Q17 – High Byte of Low Word Q18 – Low Byte of Low Word
	F24	F24 – High Byte of High Word F25 – Low Byte of High Word F26 – High Byte of Low Word F27 – Low Byte of Low Word
	D002.000	D002.000 – High Word D002.001 – Low Word

The Bit Interpretation is as follows:

Astra Tag Type	Interpretation in Astra	Description
Discrete	Single Discrete bit	0 is OFF and 1 in ON
Unsigned Integer	Unsigned 16 bit decimal word	Bit 0 is the low bit Bit 15 is the high bit
Integer	Signed 16 bit decimal word	Bit 15 is the sign bit Bit 0 is the low bit Bit 14 is the high bit
Large Integer	Signed 32 bit decimal word	Bits 0 – 15 are high word Bits 16 – 31 are the low word Bit 15 is the sign bit Bit 14 is the high bit Bit 16 is the low bit

⌘ Data Types and Addressing ⌘

Astra Tag Type	Interpretation in Astra	Description
Real	Signed 32 bit IEEE floating point number	<p>Bit 15 is the sign bit</p> <p>Bits 7 – 14 form the exponent, of which bit 14 is high bit and bit 7 is low bit.</p> <p>Bits 0 – 6 and 16 – 31 form the significant, of which bit 6 is high bit and bit 16 is low bit.</p>

Addressing :-

Inputs -

Type	Discrete	
Format	I <nth Input> . <nth Bit>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: I000.0 - I127.7 : I000.0 - I127.7 : I000.0 - I127.7 : I000.0 - I127.7 : I000.0 - I127.7 : I000.0 - I127.7
Read Write Access	Read Write	
Examples	1000.0 1127.7	: the first bit of the 0th Input for any model : the eighth bit of the 127th Input for any model

⌘ Data Types and Addressing ⌘

Type	Unsigned integer, Integer	
Format	I<nth Input>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: I000 - I126 : I000 - I126 : I000 - I126 : I000 - I126 : I000 - I126 : I000 - I126
Read Write Access	Read & Write	
Examples	1000 1003	: bits 0 – 15 referenced as a 16 bit word : bits 24 – 39 referenced as a 16 bit word

Type	Long, Real	
Format	I<nth Input>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: I000 - I124 : I000 - I124 : I000 - I124 : I000 - I124 : I000 - I124 : I000 - I124
Read Write Access	Read & Write	
Examples	1000 1005	: bits 0 – 31 referenced as a 32 bit double word : bits 40 – 71 referenced as a 32 bit double word

⌘ Data Types and Addressing ⌘

Outputs -

Type	Discrete	
Format	Q< nth Output > . < nth Bit >	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: Q000.0 – Q127.7 : Q000.0 – Q127.7 : Q000.0 – Q127.7 : Q000.0 – Q127.7 : Q000.0 – Q127.7 : Q000.0 – Q127.7
Read Write Access	Read Write	
Examples	Q000.0 Q127.7	: the first bit of the 0th Output for any model : the eighth bit of the 127th Output for any model

Type	Unsigned integer, Integer	
Format	Q< nth Output >	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: Q000 – Q126 : Q000 – Q126 : Q000 – Q126 : Q000 – Q126 : Q000 – Q126 : Q000 – Q126
Read Write Access	Read & Write	
Examples	Q000 Q003	: bits 0 - 15 referenced as a 16 bit word : bits 24 - 39 referenced as a 16 bit word

⌘ Data Types and Addressing ⌘

Type	Long, Real	
Format	Q<nth Output>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: Q000 – Q124 : Q000 – Q124 : Q000 – Q124 : Q000 – Q124 : Q000 – Q124 : Q000 – Q124
Read Write Access	Read & Write	
Examples	Q000 Q005	: bits 0 - 31 referenced as a 32 bit double word : bits 40 - 71 referenced as a 32 bit double word

Flags -

Type	Discrete	
Format	F<nth Flag>.<nth Bit>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: F000.0 – F127.7 : F000.0 – F255.7 : F000.0 – F127.7 : F000.0 – F127.7 : F000.0 – F255.7 : F000.0 – F255.7
Read Write Access	Read Write	
Examples	F000.0 F127.7	: the first bit of the 0 th Flag for any model : the eighth bit of the 127 th Flag for any model

⌘ Data Types and Addressing ⌘

Type	Unsigned Integer (Byte)	
Format	F < nth Flag > .B	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: F000.B – F127.B : F000.B – F255.B : F000.B – F127.B : F000.B – F127.B : F000.B – F255.B : F000.B – F255.B
Read Write Access	Read Write	
Examples	F000.B F127.B	: The 0 th Flag Byte for any model : The 127 th Flag Byte for any model

Type	Unsigned integer, Integer	
Format	F < nth Flag >	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: F000 – F126 : F000 – F254 : F000 – F126 : F000 – F126 : F000 – F254 : F000 – F254
Read Write Access	Read & Write	
Examples	F000 F003	: bits 0 - 15 referenced as a 16 bit word : bits 24 - 39 referenced as a 16 bit word



Data Types and Addressing



Type	Long, Real	
Format	F < nth Flag >	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: F000 – F124 : F000 – F252 : F000 – F124 : F000 – F124 : F000 – F252 : F000 – F252
Read Write Access	Read & Write	
Examples	F000 F005	: bits 0 - 31 referenced as a 32 bit double word : bits 40 - 71 referenced as a 32 bit double word

⌘ Data Types and Addressing ⌘

Timers -

Type	Unsigned integer	
Format	T<nth Timer>.<V / M>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: T000.V – T031.V / T000.M – T031.M : T000.V – T127.V / T000.M - T127.M : T000.V – T015.V / T000.M – T015.M : T000.V – T031.V / T000.M – T031.M : T000.V – T127.V / T000.M - T127.M : T000.V – T127.V / T000.M - T127.M
Read Write Access	Read only	
Examples	T003.V T003.M	: the 3rd Timer Base Value : the 3rd Timer Base Multiplier in milliseconds



Note : Above Timers need to be preset using the S5 Programming Software.
 Timer (in milliseconds) = Base Value * Base Multiplier.
 Timer(in seconds) = Base Value * Base Multiplier / 1000.

Counters -

Type	Unsigned integer, Integer	
Format	C<nth Counter>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: C000 – C031 : C000 – C127 : C000 – C015 : C000 – C031 : C000 – C127 : C000 – C127
Read Write Access	Read only	
Examples	C000 C003	: the 0th Counter : the 3rd Counter



Note : Above Counters need to be preset using the S5 programming software.

⌘ Data Types and Addressing ⌘

Data Words -

Type	Unsigned integer, Integer	
Format	D<nth Data Block> . <nth Data Word>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: D002.000 – D063.254 : D002.000 – D255.254 : D002.000 – D063.254 : D002.000 – D063.254 : D002.000 – D255.254 : D002.000 – D255.254
Read Write Access	Read & Write	
Examples	D002.000 D063.003	: bits 0 - 15 of the 2 nd Data Block referenced as a 16 bit word : bits 48 - 63 of the 63 rd Data Block referenced as a 16 bit word



Note: Above Data Blocks & Data Words need to be defined, using the S5 programming software, before use.

Type	Long, Real	
Format	D<nth Data Block> . <nth Data Word>	
Range	90U CPU 95U CPU 100U CPU 102U CPU 103U CPU 115U CPU	: D002.000 – D063.254 : D002.000 – D255.254 : D002.000 – D063.254 : D002.000 – D063.254 : D002.000 – D255.254 : D002.000 – D255.254
Read Write Access	Read & Write	
Examples	D002.000 D063.003	: bits 0 - 31 of the 2 nd Data Block referenced as a 32 bit word : bits 48 - 79 of the 63 rd Data Block referenced as a 32 bit word



Note: Above Data Blocks & Data Words need to be defined, using the S5 programming software, before use.



Notes



Notes :-



While reading or writing to the $I<n>$, $Q<n>$, $F<n>$ as unsigned integer or integer, the word formed is a concatenation of 2 bytes i.e. (n)th + (n+1)th byte, where (n)th byte is the higher byte and (n+1)th byte is the lower byte. For example: Q8 is the word formed by Q8 + Q9 bytes. So, if Q8 = 0x10 and Q9 = 0x20, then the word returned while requesting Q8 is (0x10 0x20) i.e. 428 in decimal.



Similarly, while reading or writing to the $I<n>$, $Q<n>$, $F<n>$ as larger integer or real, the double word formed is a concatenation of 4 bytes i.e. (n)th + (n+1)th + (n+2)th + (n+3)th bytes, where (n)th byte is the higher byte of the higher word, (n+1)th byte is the lower byte of the higher word, (n+2)th byte is the higher byte of the lower word and (n+3)th byte is the lower byte of the lower word. For example: F10 is the word formed by F10 + F11 + F12 + F13 bytes. So, if F10 = 0x10 and F11 = 0x20 and F12 = 0x30 and F13 = 0x40, then the double word returned while requesting F10 is (0x10 0x20 0x30 0x40) i.e. 270544960 in decimal.



While reading or writing to the $D<n>$. $<w>$ as large integer or real, the double word formed is a concatenation of 2 words i.e. (w)th + (w+1)th word, where (w)th word is the higher word and (w+1)th word is the lower word. For example: D008.002 is the word formed by D008.002 + D008.003 words. So, if D008.002 = (0x10 0x20) and D008.003 = (0x30 0x40) then the double word returned while requesting D008.002 is (0x10 0x20 0x30 0x40) i.e. 270544960 in decimal.

For Flags, Input and Output (Address starting with F,I,Q) now bit wise write is supported. This is achieved by first reading data from PLC and writing the required data with proper masking.

Example : If F100.1 is Discrete type of data defined. Then before writing this bit to the PLC data from Word F100 is read and only 2nd LSB is changed as per the entered value.



If user defines the WORD type tags with Flag byte or Output or Input then following thing should be remembered. When user defined Word F100 then the word consists of bytes F100 and F101 where F100 becomes MSB and F101 LSB. So whenever user performs write to this tag then both these bytes will be affected. Similarly in case of Word F101 byte at position F102 will be affected. The same case applies to Input and Output.

Astra Driver supports special consideration for Flag Bytes. You can define a tag as BYTE for Flag bytes. And this will affect only the byte not its adjacent byte as mentioned above. Following example will illustrate the use.

Example : Give the address of tag as **F100.b** with data type as unsigned integer. This will tell the driver that in all cases this tag should be treated as **byte** only. The **.b** given at the end of address field will indicate that the tag is of BYTE type. The write operation performed in this case will not affect subsequent byte like F101.



It is recommended that data should not be written to the Input Registers, as doing so may result into the PLC going into an error state.



Optimizations



Use the following guidelines so that you can get an optimum performance from the driver PLC combination.

- Whenever possible, use consecutive addresses, this reduces the overhead on the communication per requested data byte, word or double word.
- When a same address is to be used for two different tags in Astra, make sure that the scan time is the same for both the tags, this ensures that the address is fetched only once for both the tags.
- Use higher scan rates whenever the application allows to do so, this ensures that the critical tags with lower scan rates are fetched with minimum overhead.



Errors



The entire time a Astra project is running, the Event Logger displays the status and any errors that the program generates. The driver utilizes the Event Logger to display error messages regarding the driver. Below are the error messages, the probable cause and most likely solution to all the errors the driver can generate.

Errors displayed as strings

1. NULL Pointer for Login Data
2. NULL Pointer for Project Path
3. NULL Pointer for Tag Table
4. NULL Handle for Data Manager

Explanation : Internal Fatal Error.

Action : Contact Astra support.

5. Insufficient Memory for Request Manager
6. Insufficient Memory for Transaction Manager
7. Insufficient Memory for Device Manager

Explanation : Internal Fatal Error.

Action : Try making more memory available for the project.



Errors



8. Cannot Pagelock Tag Table

Explanation : Internal Fatal Error.

Action : Contact Astra support.

9. Cannot Open File PLCTAG.DAT

10. Cannot Read File PLCTAG.DAT

Explanation : Internal Fatal Error. The input file PLCTAG.DAT does not exist or is corrupt.

Action : Open the project in the configuration mode and close it, this process recompiles the PLCTAG.DAT file.

11. Insufficient Memory for Tag

12. Insufficient Memory for Tag2

13. Insufficient Memory for Tag Container

14. Insufficient Memory for Node

15. Insufficient Memory for Node Container

Explanation : Internal Fatal Error.

Action : Try making more memory available for the project.

16. No Tags in the Project

Explanation : Internal Fatal Error. The driver detected no valid tags in the project.

Action : Recheck the project in the configuration mode. See if any tags are assigned to this particular device. See if the Node details are correct.

17. No Valid Nodes in the Project

Explanation : Internal Fatal Error. The driver detected no valid nodes in the project.

Action : Recheck the project in the configuration mode. See if the Node details are correct.

18. Multidrop not Supported

Explanation : Internal Fatal Error. An attempt was made to attach two nodes on the same driver when Multidrop is not supported. (Not applicable for KOYO driver).

Action : Recheck the project in the configuration mode. See if the Node details are correct.

19. Multiple nodes with same ID

Explanation : Internal Fatal Error. An attempt was made to attach two nodes on the same driver with same Node IDs.

Action : Recheck the project in the configuration mode. See if the Node details are correct.



Errors



20. Insufficient Memory for Request

22. Insufficient Memory for Request Container

24. Insufficient Memory for Action

26. Cannot Create Communication Window

Explanation : Internal Fatal Error.

Action : Try making more memory available for the project.

21. Insufficient Memory for Request2

23. Insufficient Memory for Dummy Request

25. Insufficient Memory for Action Container

27. Cannot Open Communication Port

Explanation : Internal Fatal Error. Could not initialize the Communication port for the given settings.

Action : For the selected Communication port, check for -

- ⊙ If the port physically exists.
- ⊙ If the Communication hardware uses standard base addresses. COM1 uses hex 3F8 and COM2 uses hex 2F8.
- ⊙ If there is any IRQ contention at the hardware level. COM1 uses IRQ4 and COM2 uses IRQ3.
- ⊙ If any other program is already using the Communication port you have requested for
- ⊙ If any DOS level TSRs are running which are using the Communication port you have requested for.
- ⊙ If a mouse driver is installed on the same Communication port you have requested for in Windows environment.
- ⊙ If a mouse driver is installed on the same Communication port you have requested for on DOS environment.
- ⊙ If you have directly manipulated the PROJECT.INI file section [COM1] or [COM2], check if the settings for Baud Rate, Data Bits, Stop Bits and the Parity are standard. Try using the Communication port setting utility provided with Astra in case you are in doubts about the standard settings.

28. Cannot Build Communication DCB

Explanation : Internal Fatal Error. Could not initialize the Communication port for the given settings.

Action : If you have directly manipulated the PROJECT.INI file section [COM1] or [COM2], check if the settings for Baud Rate, Data Bits, Stop Bits and the Parity



Errors



are standard. Try using the Communication port setting utility provided with Astra in case you are in doubts about the standard settings.

29. Cannot Set Communication State

Explanation : Internal Fatal Error. Could not initialize the Communication port for the given settings.

Action : If you have directly manipulated the PROJECT.INI file section [COM1] or [COM2], check if the settings for Baud Rate, Data Bits, Stop Bits and the Parity are standard. Try using the Communication port setting utility provided with Astra in case you are in doubts about the standard settings.

30. NULL Pointer for Model Names

Explanation : Internal Fatal Error.

Action : Contact Astra support.

31. Read Queue Full

32. Device Time Out

Explanation : The Device did not respond and the Device driver timed out. The Driver will retry the request to Device for a specified number of times and if the Device still does not respond the driver will HALT its transactions with the Device.

Action : If this happens during **initialization**, check –

- ⊙ Whether the Device power is on.
- ⊙ Whether the cable connections to the device are proper.
- ⊙ Whether the Node ID settings are proper in case the Device supports it.
- ⊙ Whether the Device model is the same as configured in the Node Configuration.
- ⊙ Whether the Communication hardware is proper and works.
- ⊙ Whether strong EMI or RFI fields are existent which cause noise on the Communication line.
- ⊙ Whether some turnaround delay is required, try changing the entries in the DRIVERS.INI file. This may be typically required for faster PCs on which Astra runs.

If this happens during the **Run**, check –

- ⊙ Whether other applications block the Windows, in such a case the retry mechanism will normally re-establish the Communication.



Errors



- ⊙ Whether the cable connections have been disturbed.
- ⊙ Whether the Device has malfunctioned.
- ⊙ Whether the Communication hardware is proper and works.

33. Invalid IEEE Format

Explanation : The 32 bits read from the Device contained bit values such that it could not be interpreted as a valid IEEE format.

Action : Use OEM software and initialize floating type tags in the plc.

34. Write Queue Full

Explanation : The write request sent by the Astra is queued for faster execution, the current limit for the queue size is 150. If the queue is full this message will be prompted and the latest request will be ignored.

Action : Try configuring the Project such that at a time less than 150 write requests are raised. Also make sure that the Device gets enough time to serve these write requests.

35. No Valid Tags in the Project

Explanation : Internal Fatal Error. The driver detected no valid tags in the project.

Action : Recheck the project in the configuration mode. See if any tags are assigned to this particular device. See if the Node details are correct.

36. Insufficient Memory for Register

37. Insufficient Memory for Tag Container2

38. Insufficient Memory for Register Container

Explanation : Internal Fatal Error.

Action : Try making more memory available for the project.

39. Tag Address Invalid

40. Tag Address Invalid2.

Explanation : The address entered for a Tag is invalid.

Action : Reconfigure the project and check.



Errors



41. Driver Scan Halted

Explanation : The driver has stopped communicating with the device. This may happen in two situations –

- ⊙ When the initial scan is complete - in this case this is just a status information.
- ⊙ When time-out has occurred and retry for establishing communication has failed.

Action

: In the second case check –

- ⊙ If the cable connections have been disturbed.
- ⊙ If the Device has malfunctioned.
- ⊙ If the Communication hardware is proper and works.

42. Driver Scan Halted2

43. Cannot Find INI File Entry, Setting Default Port

Explanation : The [PROTOCOL] section in PROJECT.INI does not have the driver name against the COM1 or the COM2 entry. In such a case default COM1 is selected as the Communication port.

Action

: Run the Communication port setting utility provided with Astra and set all the parameters properly.

44. Cannot Initialise Driver Twice

45. Cannot Run Without Initialisation

46. Cannot Run Without Initialization2

47. Cannot Write Without Initialisation

48. Cannot Build Frames Without Initialisation

Explanation : Due to some abnormal termination in a previous run, the Device Driver has not unloaded itself and hence could not reinitialise itself.

Action

: Restart the project.

49. NULL Pointer for Queue

Explanation : Internal Fatal Error.

Action

: Contact ASTRA support.

50. Invalid IEEE Format2

Explanation : The 32 bits read from the Device contained bit values such that it could not be interpreted as a valid IEEE format.

Action

: Use OEM software and initialise floating type tags in the plc.



Errors



51. Cannot Pagelock Buffer

Explanation : Internal Fatal Error.

Action : Contact ASTRA support.

52. Device Response Delay

Explanation : The Device did not respond and the Device driver timed out. The Driver will retry the request to Device for a specified number of times and if the Device still does not respond the driver will HALT its transactions with the Device.

Action : If this happens during **Initialisation** check –

- ⊙ If the Device is powered on.
- ⊙ If the cable connections to the device are proper.
- ⊙ If the Device model is the same as configured in the Node Configuration.
- ⊙ If the Communication hardware is proper and working.
- ⊙ If strong EMI or RFI fields are existent which cause noise on the Communication line.

If this happens during the **Run** check –

- ⊙ If in case other applications block the Windows, in such a case the retry mechanism will normally re-establish the Communication.
- ⊙ If the cable connections have been disturbed.
- ⊙ If the Device has malfunctioned.
- ⊙ If the Communication hardware is proper and working.

53. Response Check Sum Error

Explanation : The Device did respond but the bytes received were corrupt. The Driver will retry the request to Device. It is also possible that the data entered by the user is out of range of the data type. This can happen for byte type of tags.

Action : If this happens during **Initialisation** check –

- ⊙ If the Communication hardware is proper and working.
- ⊙ If strong EMI or RFI fields are existent which cause noise on the Communication line.
- ⊙ If the Communication port settings are proper.



Errors



- ⊙ If this happens during the **Run** check –
- ⊙ If the cable connections have been disturbed.
- ⊙ If the Device has malfunctioned.
- ⊙ If the Communication hardware is proper and working.

54. Data Over Flow

Explanation : Unexpected data in large volume was received on the Communication port.

Action : Check –

- ⊙ If the cable connections have been disturbed.
- ⊙ If the Device has malfunctioned.
- ⊙ If the Communication hardware is proper and working.
- ⊙ Check the data entered by the user. In case of the byte type tags the data input range is between 0-255.

55. Model Name Invalid

Explanation : Internal Fatal Error. The model name associated with a particular Node was invalid.

Action : Open the project in the configuration mode. Check the model in the Node Configuration and close it.

56. Cannot Open File PLCTAG.DAT 2

57. Cannot Read File PLCTAG.DAT 2

58. Cannot Read File PLCTAG.DAT 3

Explanation : Internal Fatal Error. The input file PLCTAG.DAT does not exist or is corrupt.

Action : Open the project in the configuration mode and close it, this process recompiles the PLCTAG.DAT file.

59. Invalid BCD Format for a WORD

Explanation : The 16 bits read from the Device contained bit values such that it could not be interpreted as a valid BCD format.

Action : Use OEM software and initialise respective tags in the plc.

60. Invalid BCD Format for a DWORD

Explanation : The 32 bits read from the Device contained bit values such that it could not be interpreted as a valid BCD format.

Action : Use OEM software and initialise respective tags in the plc.



Errors



61. Invalid number for conversion to BCD for WORD

Explanation : The 32 bits read from the Device contained bit values such that it could not be interpreted as a valid BCD format.

Action : Use OEM software and initialise respective tags in the plc.

62. Invalid number for conversion to BCD for DWORD

Explanation : The 32 bits given for write from ASTRA to the Device contained bit values such that it could not be interpreted as a valid BCD format. Write will not be done in these cases.

Action : Avoid such values.

63. Error Composing Write Request

Explanation : Write request could not be composed. This may happen in two cases –

⊙ Invalid number for write.

⊙ Write Queue full.

Action : Avoid non interpretable values, Avoid writing too fast.

64. Error Composing Read Request After Write

Explanation : A read request immediately following a write request could not be composed.

This may happen in two cases –

⊙ Invalid number for write.

⊙ Write Queue full.

Action : Avoid non interpretable values, Avoid writing too fast.

65. Node Failed.

Explanation : Internal Fatal Error. The Node was not able to communicate. In case of Multidrop PLC system the node id given to the nodes may be same or cable from PC to PLC may be faulty.

Action : Open the project in the configuration mode check the Node Configuration and close it. For Multidrop communication check the node ID. Check the cable.

66. Cannot Open File NODES.DAT

67. Cannot Read File NODES.DAT 2

Explanation : Internal Fatal Error. The input file NODES.DAT does not exist or is corrupt.

Action : Open the project in the configuration mode and close it, this process recompiles the NODES.DAT file.



Errors



68. Node set on by user.

Explanation : Not an error . It indicates that node is selected by the user. For ASTRA generated default tags for a PLC, in that if command tag is 0 then this message is displayed.

Action : None , as it indicates that node is selected by the user.

69. Node set off by user.

Explanation : Not an error . It indicates that node is unselected by the user. For ASTRA generated default tags for a PLC in that if command tag is 1 then this message is displayed.

Action : None, as it indicates that node is unselected by the user.

73. Node manager proc address not defined.

Explanation : Internal Fatal Error.

Action : Contact ASTRA support.

300.

Explanation : Number 300 when displayed in the event logger indicates the error condition 'Garbage'. This indicates that NAK is received by the driver.

Action : Check with OEM software and the hardware configuration of PLC.



Limitations



In case of timers, the application engineer has to define 2 tags for a timer to be displayed – one for the timer base value and the other for the timer base multiplier. The multiplication of the timer base value and timer base multiplier gives the actual timer value in milliseconds.



Renu Electronics Pvt Ltd.
S.No. 2/6, Baner Road,
Pune 411045, India.
Tel: + 91 20 2729 2840,
Fax: + 91 20 2729 2839
Email: info@renuelectronics.com
Website: www.renuelectronics.com

