



# Customisations

ISaGRAF® Library

ARTECO MOTION TECH SPA. All rights reserved.

Information in this document is subject to change without notice and does not represent a commitment on the part of Artec. The software described in this document, including the information contained in all the databases, is provided under a license agreement or a non-disclosure agreement, and may be used or copied only in accordance with the terms of such agreements. It is against the law to copy the software except as specifically printed in the licence or non-disclosure agreement. No part of this manual may be reproduced in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Artec.

## Contents

Using retained variables .....	3
VersEpr .....	3
VersPer .....	4
IORF .....	5
Note .....	5
ReadAna .....	6
Note .....	6
SetAna .....	7
VarIndex .....	8
VarCpy .....	10
VarZip .....	12
MemSt .....	13
ConfSer .....	14
GstFlash .....	18
DecBin8 .....	20
DecBin16 .....	21
BinDec8 .....	23
BinDec16 .....	24
DecBCD8 .....	26
DecBCD16 .....	27
BCDDec8 .....	29
BCDDec16 .....	30
GstSer .....	32
ReadSer .....	34
WriteSer .....	36
RdKeyb .....	36
WrKeyb .....	41
WrNumber .....	51
ChgTAssi .....	53
Set_Ip .....	54
Annex A – Virtual Variables .....	55
Annex B – Utility for reading and writing variables via serial ports .....	56
Annex C – I/O Library .....	58
Digital I/Os .....	58
Analogue I/Os .....	61
Power supply status .....	61
Axis management data .....	62
Annex D – Flash card errors .....	63
Annex E – Arteco compiler for C167 .....	63

## Using retained variables

SU data cards are provided with a reserved memory for using retained variables, which varies according to the model.

The use of such variables is subject to the insertion of this memory's data into the **Application Run Time Options**. For instructions on how to fill in the **Retained variables** field, please refer to chapter **C.9.2 Application variables backup - Retained variables** of the **ISaGRAF User Guide**.

The SU data card reserved memory goes from hex address 0x114000 to hex address 0x11BFFF for su2xx, cnd51, and sumd data cards, while for su1xx, it goes from 0x10C000 to 0x10CFFF.

For internal reasons, only an even number of bytes can be allocated for each variable; therefore, to allocate 10 Boolean variables, for example, you will have to allocate 20 (10\*2) bytes and not 10; to allocate 10 timer variables you will need to allocate 60 bytes (10\*6) instead of 50.

The following recommended configurations can be used:

### **SU210, SU212, SUMD, CND51**

114000, A28, 114A30, 1454, 115EA0, 1E78, 117D20, 4000

Such configuration allows allocating:

- 1300 Boolean variables
- 1300 analogue variables
- 1300 timer variables
- 64 messages (256 characters)

### **SU110, SU112**

10C000, C8, 10C0D0, 258, 10C340, 258, 10C5A0, A00

Such configuration allows allocating:

- 100 Boolean variables
- 150 analogue variables
- 100 timer variables
- 10 messages (256 characters)

For further information on how to allocate additional variables in the memory, or on different configurations, please contact Artec spa.



## Topics:

En	BOO	Not used
versS	MSG	String indicating code and date of EPROM version.
versC	INT	EPROM code indicated as string in for the versS output

\* Code and date refer to kernel firmware, developed by Arteco, and do not depend on the application.

## Description:

This block allows reading the EPROM version. It supplies the EPROM code and its creation date. Two outputs are supplied, one for handling a message containing all information and a code to be used, and the other to monitor whether the EPROM has changed at each start-up. For the time being, the En input does not carry out any function. For further information on your version, please contact Arteco spa.

## VersPer



## Topics:

Typ	INT	Peripheral device, whose firmware version is required.
Opt	INT	Optional parameter. It identifies the peripheral device among several ones of the same type.
VersS	MSG	String indicating the code and date of peripheral device firmware version.
versC	INT	Peripheral device version code (where available).

\* Code and date refer to the firmware developed by Arteco, and do not depend on the application.

## Description:

This block allows reading the firmware version of any peripheral device connected to the SU system. It supplies the firmware code and its creation date. Two outputs are supplied, one for handling a message containing all information and a code to be used, and the other to monitor whether the version has changed at each start-up. The latter code may not be available for certain peripheral devices.

The Typ parameter value may refer to:

- 0) Bus axis expansion
- 1) Dsp expansion.
- 2) Can expansion.

The Opt parameter represents an optional parameter. In case of expansion, it can indicate the number of the node whose firmware version is required.

For further information on your version, please contact Arteco spa.

## IORF



### Topics:

En	BOO	If TRUE, every time the block is called, Input variables are refreshed with the current values of the physical Inputs; otherwise, the physical outputs are refreshed with the Output variable values. Such operation depends on the Modo (Mode) parameter.
Modo (Mode)	INT	Indicates the type of refresh required: 0 => Refreshes outputs. 1 => Refreshes inputs. 2 => Refreshes inputs and outputs.
Q	BOO	Not used. Always indicates TRUE.

### Description:

This block allows refreshing inputs and outputs at any point of the application program. If this block is not used, Input variables are refreshed at the beginning of every PLC program scan, while outputs are refreshed at the end of each PLC program scan.

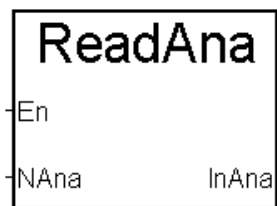
### Note

The result of each "branch" is immediately available to subsequent "branches", whether the result is an internal or an output result. Subsequent branches will, therefore, immediately consider the

theoretical value that outputs take on in the previous "branches". "Physical" activation of the output, on the other hand, takes place at the end of the PLC cycle.

Therefore, if a "branch" activates an output and another "branch" deactivates the same output, all intermediate "branches" will be "processed", considering the output as active; however, the physical output is never activated, unless the IORF block is used.

## ReadAna



### Topics:

En	BOO	If TRUE, it reads the analogue input and sends such reading to the InAna output. It works on the level and on the front; therefore, by keeping the En level high, the value will be updated every time the block is recalled.
NAna	INT	It indicates the analogue input, whose input voltage is required. Currently, 0, 1 or 2 are allowed.
InAna	INT	The reading of the analogue input voltage is expressed in mvolt. E.g. 2.5volt => 2500.

### Description:

This block allows reading analogue inputs of the SU210 data card, returning the voltage value expressed in mvolt.

There are three analogue inputs available; therefore, the Nana input value can be 0, 1 or 2.

### Note

The analogue digital converter has a resolution of 10 bits; therefore, the voltage value read must have a 19mvol resolution as a result of the division between the voltage range (+10v, -10v; therefore a range of 20v) and the maximum number returned by the converter ( $2^{10}$  therefore 1024).

The value read by this block is also available through the nS210Ana library (see Annex C I/O Library).

## SetAna



### Topics:

En	BOO	If TRUE, it sets the analogue output. It works on the level and on the front; therefore, by keeping the En level high, the value will be set every time the block is recalled.
Chan	INT	Indicates the analogue output, whose voltage must be set.
Vreq	INT	Indicates the value, in millivolts, to be set on the analogue output
Vout	INT	Analogue output value expressed in millivolts. E.g. 2.5volt => 2500.

### Description:

This block allows setting a voltage value, expressed in millivolts, on the analogue outputs.

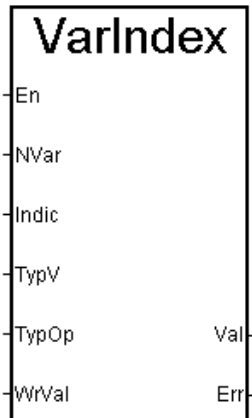
The SU210 data card has two analogue outputs that correspond to Chan = 5 and 6, two analogue outputs piloted in pwm that correspond to Chan = 7 and 8. The output referring to the axis can also be used by selecting Chan=0, as long as the axis has not been parameterised using the ParAxis block.

The CND51 data card has an analogue output, which can be selected by setting Chan = 5. The outputs referring to the axis can also be used by selecting Chan = 0, 1, 2, 3 and 4, as long as the corresponding axis have not been parameterised using the ParAxis block.

The SUMD data card has an analogue that can be selected with Chan=8.



## VarIndex



### Topics:

En	BOO	If TRUE, it returns the value of the variable, whose index is the sum of Nvar and Indic inputs.
NVar	INT	Represents the first addend that determines the Virtual Address of the variable, whose value is to be read. See description.
Indic	INT	Represents the second addend that determines the Virtual Address of the variable, whose value is to be read. See description.
TypV	INT	Indicates whether the requested variable is ISaGRAF or Virtual. 0 indicates a regularly declared ISaGRAF variable. 1 Indicates a virtual variable.
TypOp	INT	Indicates the type of operation to be carried out. 0 variable reading. 1 variable writing.
WrVal	INT	Value to be written in the variable. Only used when the TypOp parameter indicates writing (1).
Val	INT	Value of the variable whose virtual Address is indicated by the sum of the indexes represented by the Nvar and Indic inputs.
Err	BOO	Indicates whether the reading of the requested variable has been carried out correctly or not. FALSE indicates a correct reading, TRUE indicates a failed reading.

## Description:

This block allows reading or writing a variable value by using the Virtual Address.

The Virtual Address used to reach the variable to be read consists in the sum of two inputs (Nvar and Indic).

This block can be used for reading or writing a variable through indexing, where Nvar represents a starting variable and Indic represents an index that increases progressively.

This block manages only one block at a time. Refer to the VarCpy block to transfer entire blocks of variables.

Both ISaGRAF and virtual variables can be read using the TypV input to indicate the selection. As for Virtual variables, the index must be between 1 and 8000 decimal.

The Virtual Address is a hexadecimal, while this block input uses decimal values. Therefore, a variable with Virtual Address 1000 must be reached through an address 4096 (the decimal equivalent to hex address 1000). Should the Virtual Address refer to an analogue variable, the result Val will be an integer decimal; should it refer to a Boolean variable, Val can be equal to 0 (FALSE) or 1 (TRUE); should it refer to a timer variable, it will be an integer decimal that represents milliseconds.

In case of writing, the value assigned to WrVal shall be considered a whole number if the target variable is an integer. On the other hand, it shall be considered in msec if the target variable is a timer variable. Should the target variable be a Boolean variable, WrVal shall be assigned to FALSE if it is equal to zero, while in every other case (any number other than 0) it shall be assigned to TRUE.

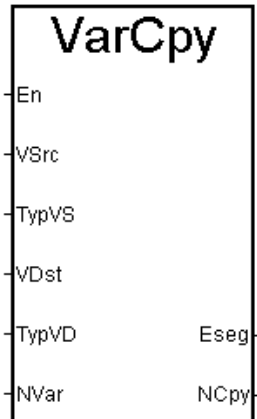
**Note: Although this block allows managing Boolean, integer and timer variables, its output, however is integer. To write such value in Boolean or timer variables, the Boo and Tmr blocks must be used respectively.**

## Attention:

Hexadecimal numbers can be introduced in the constant using the prefix 16#.

E.g. To read the 30 variables following the Virtual Address 2A00, you can supply a variable initialised at 16#2A00 to the NVal input and use Indic giving ten values from 1 to 30. The addresses read will range from 2A01 to 2A1E.

## VarCpy



### Topics:

En	BOO	On the positive edge it starts copying the variables.
VSrc	INT	Virtual Address of the first source variable.
TypVS	INT	Indicates whether the source variable is ISaGRAF or Virtual. 0 indicates a regularly declared ISaGRAF variable. 1 Indicates a virtual variable. In case of a copy of several variables, the type refers to the source variables.
VDst	INT	Virtual Address of the first target variable.
TypVD	INT	Indicates whether the target variable is ISaGRAF or Virtual. 0 indicates a regularly declared ISaGRAF variable. 1 Indicates a virtual variable. In case of a copy of several variables, the type refers to the target variables.
NVar	INT	Number of <u>consecutive</u> variables to be copied. See description.
Eseg	BOO	Indicates whether the variable copy operation is in progress. TRUE indicates an operation in progress. FALSE indicates a completed operation.
NCpy	INT	Indicates the actual number of copied variables. Normally, it is equal to NVar. Should it be lower than NVar, it means that some variables have not been copied. See description.

### Description:

This block allows copying a block of BOOLEAN, INTEGER ANALOGUE or TIMER variables in another group, even of a different type, as long as they belong to one of the categories mentioned above. The VSrc input indicates the Virtual Address of the first variable to be copied, while the VDst input indicates the Virtual Address of the first variable to be overwritten with new values. NVar indicates the number of consecutive variables that must be copied. By consecutive we refer to the Virtual Address value.

Variables can also be copied from and in the virtual variable area. TypVS and TypVD indicate the type of source and target variables (ISaGRAF or Virtual), respectively. As for Virtual variables, the index must be between 1 and 8000 decimal.

The Virtual Address is a hexadecimal, while this block input uses decimal values. Therefore, a variable with Virtual Address 1000 must be reached through an address 4096 (the decimal equivalent to hex address 1000).

Since copying large blocks of memory may require a certain amount of time, this operation is divided into several PLC cycles, so that every PLC cycle will not require more than 4 milliseconds. Should the variables to be copied be only a few, the operation will be completed during the subsequent PLC cycle; vice-versa, several cycles could be required. The Eseg and NCpy outputs allow knowing the progress of the copy operation.

On the positive edge of the En input, the Eseg output reaches a high level to indicate that the operation is in progress. When the Eseg output reaches a low level, the operation is completed and NCpy indicates the actual number of copied variables. This number must be equal to NVar; if not, there are either missing Virtual Addresses (undeclared) or non-allowed variables in the block interval.

Every PLC cycle allows knowing the variables that have actually been copied through the NCpy output, even if the operation is still in progress.

Different types of variables can be copied. Below are the conversions carried out in case of source and target variables of different type:

Target

- 1) Boolean. If the source variable is an integer equal to 0, the target is equal to FALSE; vice-versa it is equal to TRUE. If the source variable is a timer and equal to 0, the target is equal to FALSE; vice-versa it is equal to TRUE.
- 2) Integer. Should the source variable be Boolean equal to FALSE, the target will be equal to 0; vice-versa it will be equal to 1. Should the source variable be a timer equal to 0, its value will be zero; vice-versa the value of the timer variable will be expressed in milliseconds.
- 3) Timer. If the source variable is a Boolean equal to FALSE, the target is equal to 0; vice-versa it is equal to 1millisecond. If the source variable is an integer equal to 0, its value will be 0; vice-versa it will be equal to the value of the source expressed in milliseconds.

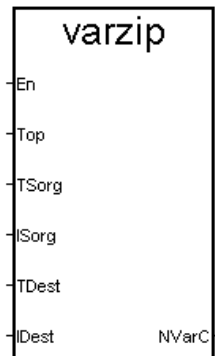
## Attention:

Hexadecimal numbers can be introduced in the constant using the prefix 16#.

E.g. To copy the 30 variables following the Virtual Address 2A00, you can supply a variable initialised at 16#2A00 (or the hexadecimal number 10752) to the VSrc input, the value 16#3000 (or the decimal number 12288) to the VDst input and indicate 30 NVar. The value of the variables with addresses ranging between 2A00 and 2A1E will be copied in the variables with addresses ranging between 3000 and 301E.

If the variables between 3003 and 3005 have not been declared, at the end of the copy operation NCpy will indicate 28.

## VarZip



### Topics:

En	BOO	If the En input is TRUE, the variables will be either compressed or extracted.
TOp	INT	Indicates the type of operation to be carried out: 0 → Compression of 32 variables (Boolean, Integer o Virtual) in one 32-bit variable (Integer or Virtual). 1 → Extraction of 32 variables (Boolean, Integer o Virtual) from one 32-bit variable (Integer or Virtual).
TSorg	INT	Indicates the type of source variable: 0 → IsaGRAF variable® (Integer, Boolean or Timer). 1 → Virtual Variable.
ISorg	INT	Address value (or index) of the source variable (if TOp=1) or of the first variable of the 32 to be compressed (TOp=0). The admitted values are 1-FFFF hexadecimal for ISaGRAF® variables and 1-8000 decimal for Virtual variables.
TDest	INT	Indicates the type of target variable: 0 → IsaGRAF variable® (Integer, Boolean or Timer). 1 → Virtual Variable.
IDest	INT	Address value (or index) of the target variable (if TOp=0) or of the first variable of the 32 to be extracted (TOp=1). The admitted values are 1-FFFF hexadecimal for ISaGRAF® variables and 1-8000 decimal for Virtual variables.
NVarC	INT	Indicates the result of the required operation. Its <b>positive value</b> indicates that the variables have been successfully compressed or extracted. Its value represents the actual number of compressed or extracted variables. Its <b>negative value</b> indicates an error. In particular, the following error codes can occur: -1: The source address is incorrect (it exceeds the maximum allowable values). -2: The source variable is incorrect (e.g. Boolean with TOp=1). -3: The target address is incorrect (it exceeds the maximum allowable values). -4: Undefined target variable. -5: Undefined source variable. -6: The target variable is incorrect (e.g. Boolean with TOp=0).

## Description:

This block allows compressing 32 variables with consecutive addresses in one single 32-bit variable, or extracting a 32-bit variable into 32 variables with consecutive addresses. Such variables can be either ISaGRAF® or virtual.

If TOP=0, the block allows compressing 32 variables with consecutive addresses in one single 32-bit variable. If the source variables are Boolean, the compression will be carried out as follows:

- i-sima variable=TRUE → i-simo bit of the target variable=1
- i-sima variable=FALSE → i-simo bit of the target variable=0

If the source variables are NOT Boolean (Virtual, Integer or Timer), compression is carried out as follows:

- i-sima variable  $\neq 0$  → i-simo bit of the target variable =1
- i-sima variable = 0 → i-simo bit of the target variable =0

Source variables can also be of different types (e.g. both Boolean and Integer), but they must all be defined and with consecutive addresses. Should there be one or more undefined variables (whose addresses, therefore, are not consecutive) among the 32, the block will give an error.

If the target variable is Boolean the block will give an error.

If TOP=1, the block allows extracting 32 variables with consecutive address from one single 32-bit variable (either Virtual or Integer ISaGRAF). If the target variables are Boolean, extraction is carried out as follows:

- i-simo bit of the source variable =1 → i-sima target variable =TRUE
- i-simo bit of the source variable =0 → i-sima target variable =FALSE

If the target variables are NOT Boolean (Virtual, Integer or Timer), extraction is carried out as follows:

- i-simo bit of the source variable =1 → i-sima target variable =1
- i-simo bit of the source variable =0 → i-sima target variable =0

Target variables can also be of different types (e.g. both Boolean and Integer), but they must all be defined and with consecutive addresses. Should there be one or more undefined variables (whose addresses, therefore, are not consecutive) among the 32 target variables, the block will give an error.

If the source variable is Boolean the block will give an error.

This operation is carried out every time the VarZip block is recalled and the En input is TRUE; if, on the other hand, the input is FALSE, no operation will be carried out and the NvarC output will remain equal to 0.

## MemSt



## Topics:

Type	INT	Indicates the type of memory whose use is required. 1 => Size of the application code. 2 => Size of the application data.
Used	INT	Used memory in KB.
Avail	INT	Available memory in KB.

## Description:

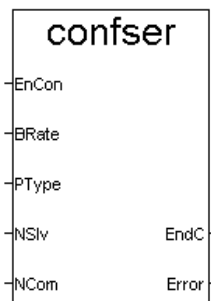
This block allows reading the memory used for the application as well as the memory available for both application data and application code.

The size referring to both the used memory and the available one is expressed in Kbyte.

The sum of the used memory and the available one gives the allowed overall size.

Should the application have a larger size, contact the Arteco SpA Technical Department.

## ConfSer



## Topics:

EnCon	BOO	The positive edge starts the ModBus protocol configuration and the initialisation of the three serial ports available for the WorkBench, ModBus and the Stepper Module protocols. This input must remain activated at least until the EndC output is activated.
BRate	INT	Indicates the required BaudRate value for the ModBus protocol. The codes to be entered are: 1 → BaudRate=57600 bit/sec 2 → BaudRate=38400 bit/sec 3 → BaudRate=19200 bit/sec 4 → BaudRate=9600 bit/sec 5 → BaudRate=4800 bit/sec
PType	INT	Indicates the required ModBus version (*).

Versions currently available are the following:

1 → "ModBus-ISaGRAF" version

2 → "ModBus 32Bit" version

3 → "ModBus Standard" version

(\*) see documentation on the available protocols.

NSlv	INT	Represents the slave (i.e., the SU210/CND51) identifier in the ModBus network. The values that can be used range between 01 and F7 Hex.
NCom	INT	Allows allocating one of the PLC SU210/CND51 serial ports to every available serial protocol (WorkBench, ModBus and Stepper Module). See the following table for the possible configurations and corresponding values to be assigned to such input.
EndC	BOO	The TRUE value indicates the end of the configuration operation (when the operation is completed correctly).
Error	INT	Error codes occurred during the configuration phase: 0 → No Error 1 → Error in the BaudRate setting 2 → The identifier allocated to the slave is not correct. 3 → The selected configuration is not available. 4 → A non-existing version has been selected.

## Description:

The *ConfSER* block allows configuring the three PLC SU210/CND51 serial ports.

In case of the **SU210** PLC, the COM1 and COM2 serial ports are not directly present on the PLC, but on the SU-COM-MOD expansion. Currently, such expansion **does not allow using the COM1 and COM2 serial ports simultaneously**, but only one or the other.

In case of PLC CND51 this limitation does not occur.

More in detail, the *ConfSER* block allows:

1. Allocate one of the three serial ports available to each of the implemented serial protocols (ISaGRAF WorkBench, ModBus and the Stepper Module connection).  
The possible configurations are listed in the table below.

Configuration	COM0 Serial port	COM1 Serial port	COM2 Serial port
<b>1</b>	ModBus	ISaGRAF WorkBench	Unused
<b>2</b>	ModBus	Unused	ISaGRAF WorkBench
<b>3</b>	ISaGRAF WorkBench	Unused	ModBus



4	ISaGRAF WorkBench	ModBus	Unused
5	ModBus	Stepper module serial connection	Unused
6	ModBus	Unused	Stepper module serial connection
7	ISaGRAF WorkBench	Stepper module serial connection	Unused
8	ISaGRAF WorkBench	Unused	Stepper module serial connection
9*	ModBus	ISaGRAF WorkBench	Stepper module serial connection
10*	ModBus	Stepper module serial connection	ISaGRAF WorkBench
11*	ISaGRAF WorkBench	Stepper module serial connection	ModBus
12*	ISaGRAF WorkBench	ModBus	Stepper module serial connection
13*	Unused	ModBus	ISaGRAF WorkBench
14*	Unused	ISaGRAF WorkBench	ModBus

\* only available on some hardware configurations.

2. Set the ModBus protocol configuration parameters.

Such parameters are the BaudRate, the "slave" identifier and the protocol version (Standard, "32Bit Data ModBus" or "ModBus-ISaGRAF").

By modifying the transmission speed, the TimeOut that defines the reception of the ModBus message will also be automatically modified.

should the ConfSER block not be used, the following **default configuration** will be maintained:

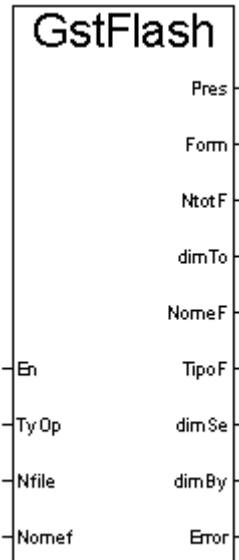
- Use of the "ModBus-ISaGRAF" version with a transmission speed of 9600 bit/sec and 01 Hex as slave identifier.
- Use of the COM0 serial port for the ModBus protocol and the COM2 serial port for WorkBench.  
**Important:** The stepper module protocol is not directly available upon start-up. It must be activated (and the serial port it uses must be configured) by means of the *ConfSER* block.

Should the *ConfSER* block be used to activate the stepper module serial connection protocol, the *EndC* output may require hundreds of milliseconds before becoming TRUE (to indicate the end of the configuration operation). This is because, in such case, also a “connection” phase between the PLC and the stepper module must be managed, along with the initialisation of the serial port. In such case, the *EndC* output is activated only once the synchronisation has occurred.

Obviously, during such time interval the execution of the PLC cycle continues and is NOT interrupted.

When allocating the serial ports to each available protocol, it is important to keep in mind that the ModBus wireless connection is available only on COM1, while the ISaGRAF WorkBench MODEM connection is available only on COM2.

## GstFlash



### Topics:

En	BOO	The positive edge starts the operation to be carried out on the flash card.
TyOp	INT	Allows selecting the operation to be carried out. The admitted values are: 0 → Display of the files contained in the flash card. 1 → Formatting of the flash card. 2 → Deletion of a file, whose number is indicated from the flash card. 3 → Deletion a file, whose name is indicated from the flash card.
Nfile	INT	Indicates the number of the file on which the required operation must be carried out: in case of <b>display</b> number 0, as well as number 1, indicates that the first file of the flash card must be displayed; number 2 indicates that the second file must be displayed; number n indicates that the nth file of the flash card must be displayed. In case of <b>deletion</b> of a file <b>given its number</b> represents the number of the file to be deleted from the flash card. In case of <b>formatting</b> and <b>deletion</b> of a file <b>given its name</b> this parameter is not significant.
Nomef	MSG	File name. This parameter is only significant for the deletion of a file given its name.

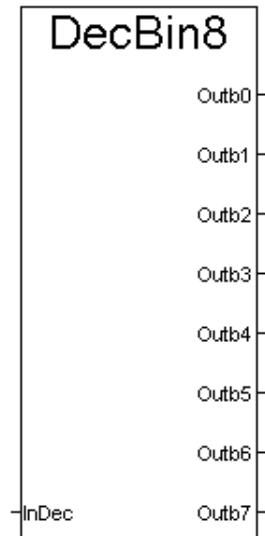
Pres	BOO	The value TRUE indicates that the flash card is present.
Form	BOO	The value TRUE indicates that the flash card is formatted.
NtotF	INT	Indicates the total number of files on the flash card.
DimTo	INT	Indicates the overall usable size of the flash card (different from the theoretical size due to the presence of restricted sectors).
NomeF	MSG	Name of the file onto which the operation was carried out. This parameter is significant only in case there are no files on the flash card.
Tipof	INT	Indicates the type of the displayed or deleted file. The admitted values are: 1 → Firmware. 2 → Application. 3 → Boolean variables. 4 → Integer variables. 5 → Timer variables. 6 → Message variables. 7 → Virtual Variable.
DimSe	INT	Size of the file under consideration (number of used flash card sectors).
DimBy	INT	Size of the file under consideration in bytes
Error	INT	Indicates whether the operation has been carried out correctly or an error has occurred: 0 → Operation completed successfully. 1 → Error in managing the flash card. 2 → The requested file is not present (incorrect parameter). 3 → Inconsistent flash card (incorrect or inconsistent data).

## Description:

The Gstflash block allows carrying out some operations on the external flash card. In particular, this block verifies whether the flash card is present or formatted, and, in the latter case, it displays the total number of files as well as the useable size. Moreover, the name and the type of files can also be displayed by entering the file number in the Nfile parameter. The flash card can also be formatted and files whose name or number is known can be deleted (in the order of appearance in the flash card). Should the operation be completed successfully, the Error parameter gives 0; if not, an error warning will be displayed.

For information on errors, refer to Annex D.

## DecBin8



### Topics:

InDec	INT	8-bit whole number to be converted in 8 bits.
Outb0	BOO	Value of the Bit 0 of the whole number at the InDec input. Less significant bit.
Outb1	BOO	Value of the Bit 1 of the whole number at the InDec input.
Outb2	BOO	Value of the Bit 2 of the whole number at the InDec input.
Outb3	BOO	Value of the Bit 3 of the whole number at the InDec input.
Outb4	BOO	Value of the Bit 4 of the whole number at the InDec input.
Outb5	BOO	Value of the Bit 5 of the whole number at the InDec input.
Outb6	BOO	Value of the Bit 6 of the whole number at the InDec input.
Outb7	BOO	Value of the Bit 7 of the whole number at the InDec input. Most significant bit.

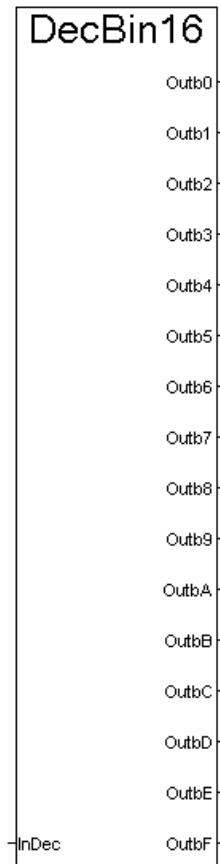
### Description:

This block allows converting an integer variable considering its less significant 8 bits in the InDec input, in 8 Boolean variables representing each its relative bit.

If the bit is set to "zero", the value of its output shall be FALSE; if it's "one" the output shall be TRUE.

The block does not provide for activation. Every time it's recalled, it converts the input data in the 8 output data.

## DecBin16



### Topics:

InDec	INT	16-bit whole number to be converted in 16 bits.
Outb0	BOO	Value of the Bit 0 of the whole number at the InDec input. Less significant bit.
Outb1	BOO	Value of the Bit 1 of the whole number at the InDec input.
Outb2	BOO	Value of the Bit 2 of the whole number at the InDec input.
Outb3	BOO	Value of the Bit 3 of the whole number at the InDec input.
Outb4	BOO	Value of the Bit 4 of the whole number at the InDec input.
Outb5	BOO	Value of the Bit 5 of the whole number at the InDec input.
Outb6	BOO	Value of the Bit 6 of the whole number at the InDec input.
Outb7	BOO	Value of the Bit 7 of the whole number at the InDec input.
Outb8	BOO	Value of the Bit 8 of the whole number at the InDec input.
Outb9	BOO	Value of the Bit 9 of the whole number at the InDec input.
OutbA	BOO	Value of the Bit 10 of the whole number at the InDec input.
OutbB	BOO	Value of the Bit 11 of the whole number at the InDec input.
OutbC	BOO	Value of the Bit 12 of the whole number at the InDec input.
OutbD	BOO	Value of the Bit 13 of the whole number at the InDec input.
OutbE	BOO	Value of the Bit 14 of the whole number at the InDec input.
OutbF	BOO	Value of the Bit 15 of the whole number at the InDec input. Most significant bit.

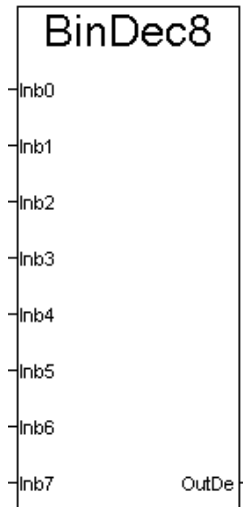
## **Description:**

This block allows converting an integer variable considering its less significant 16 bits in the InDec input, in 16 Boolean variables representing each its relative bit.

If the bit is set to "zero", the value of its output shall be FALSE; if it's "one" the output shall be TRUE.

The block does not provide for activation. Every time it's recalled, it converts the input data in the 16 output data.

## BinDec8



### Topics:

Inb0	BOO	Value of Bit 0 of the whole number to be present at the OutDe output. Less significant bit.
Inb1	BOO	Value of Bit 1 of the whole number to be present at the OutDe output.
Inb2	BOO	Value of Bit 2 of the whole number to be present at the OutDe output.
Inb3	BOO	Value of Bit 3 of the whole number to be present at the OutDe output.
Inb4	BOO	Value of Bit 4 of the whole number to be present at the OutDe output.
Inb5	BOO	Value of Bit 5 of the whole number to be present at the OutDe output.
Inb6	BOO	Value of Bit 6 of the whole number to be present at the OutDe output.
Inb7	BOO	Value of Bit 7 of the whole number to be present at the OutDe output. Most significant bit.
OutDe	INT	8-bit whole number that represents the combination of the single bits in their respective Boolean Inbnn inputs.

### Description:

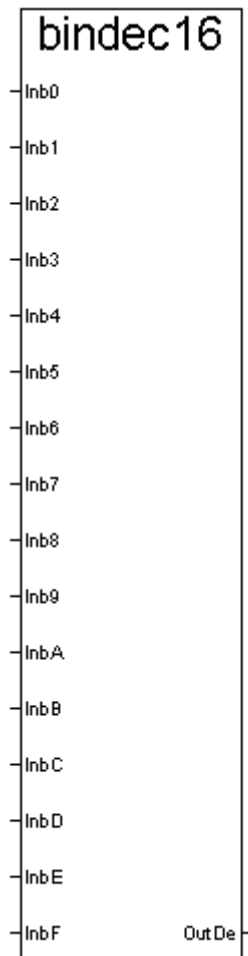
This block allows converting 8 Boolean variables at the inputs from Inb0 to Inb7 in 1 integer variable that represents the combination of the 8 Boolean inputs representing each a bit.

Should the Boolean variable be FALSE, the bit will be "zero"; should it be TRUE, the bit will be "one".

The block does not provide for activation. Every time it is recalled, it converts the 8 input data in one single output datum.



## BinDec16



### Topics:

Inb0	BOO	Value of Bit 0 of the whole number to be present at the OutDe output. Less significant bit.
Inb1	BOO	Value of Bit 1 of the whole number to be present at the OutDe output.
Inb2	BOO	Value of Bit 2 of the whole number to be present at the OutDe output.
Inb3	BOO	Value of Bit 3 of the whole number to be present at the OutDe output.
Inb4	BOO	Value of Bit 4 of the whole number to be present at the OutDe output.
Inb5	BOO	Value of Bit 5 of the whole number to be present at the OutDe output.
Inb6	BOO	Value of Bit 6 of the whole number to be present at the OutDe output.
Inb7	BOO	Value of Bit 7 of the whole number to be present at the OutDe output.
Inb8	BOO	Value of Bit 8 of the whole number to be present at the OutDe output.
Inb9	BOO	Value of Bit 9 of the whole number to be present at the OutDe output.
InbA	BOO	Value of Bit 10 of the whole number to be present at the OutDe output.
InbB	BOO	Value of Bit 11 of the whole number to be present at the OutDe output.
InbC	BOO	Value of Bit 12 of the whole number to be present at the OutDe output.
InbD	BOO	Value of Bit 13 of the whole number to be present at the OutDe output.

InbE	BOO	Value of Bit 14 of the whole number to be present at the OutDe output.
InbF	BOO	Value of Bit 15 of the whole number to be present at the OutDe output. Most significant bit.
OutDe	INT	16-bit whole number that represents the combination of the single bits in their respective Boolean Inbnn inputs.

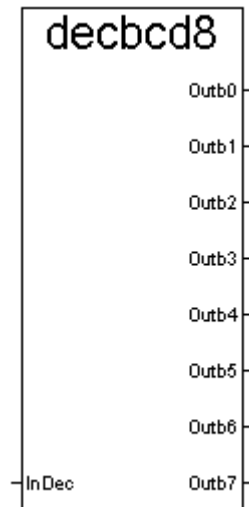
## Description:

This block allows converting 16 Boolean variables at the inputs from Inb0 to InbF in 1 integer variable that represents the combination of the 16 Boolean inputs representing each a bit.

Should the Boolean variable be FALSE, the bit will be "zero"; should it be TRUE, the bit will be "one".

The block does not provide for activation. Every time it is recalled, it converts the 16 input data in one single output datum.

## DecBCD8



### Topics:

InDec	INT	8-bit decimal integer number that is converted in bits (Outbn) corresponding to the BCD format.
Outb0	BOO	Value of Bit 0 mm BCD format of the whole number in the InDec input. Less significant bit.
Outb1	BOO	Value of Bit 1 in BCD format of the whole number in the InDec input.
Outb2	BOO	Value of Bit 2 in BCD format of the whole number in the InDec input.
Outb3	BOO	Value of Bit 3 in BCD format of the whole number in the InDec input.
Outb4	BOO	Value of Bit 4 in BCD format of the whole number in the InDec input.
Outb5	BOO	Value of Bit 5 in BCD format of the whole number in the InDec input.
Outb6	BOO	Value of Bit 6 in BCD format of the whole number in the InDec input.
Outb7	BOO	Value of Bit 7 in BCD format of the whole number in the InDec input. Most significant bit.

### Description:

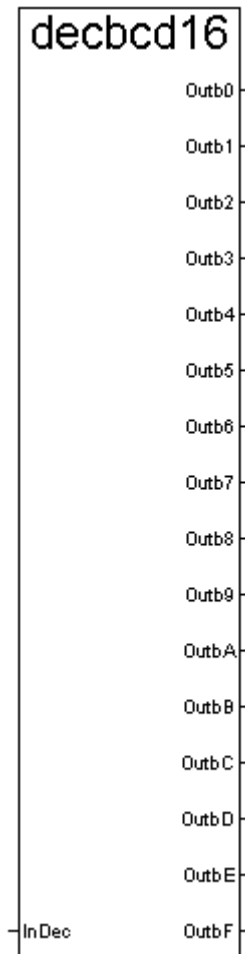
This block allows converting an integer variable at the InDec input into 8 Boolean variables (from Outb0 to Outb7), which represent the input decimal number in BCD format.

For the BCD format, every single digit of the decimal number to be converted must be represented by 4 bits (nibble). Therefore, a three digit decimal number is converted into a 12-bit binary number (3 nibbles). E.g. 99 decimal => 10011001 binary. Therefore, each nibble can vary from 0 (0000b) to 9 (1001b).

Numbers with decimal points cannot be represented. The highest representable number is 99.

The block does not provide for activation. Every time it's recalled, it converts the input datum in the 8 output data.

## DecBCD16



### Topics:

InDec	INT	16-bit decimal integer number that is converted into bits (Outbn) corresponding to the BCD format.
Outb0	BOO	Value of Bit 0 in BCD format of the whole number in the InDec input. Less significant bit.
Outb1	BOO	Value of Bit 1 in BCD format of the whole number in the InDec input.
Outb2	BOO	Value of Bit 2 in BCD format of the whole number in the InDec input.
Outb3	BOO	Value of Bit 3 in BCD format of the whole number in the InDec input.
Outb4	BOO	Value of Bit 4 in BCD format of the whole number in the InDec input.
Outb5	BOO	Value of Bit 5 in BCD format of the whole number in the InDec input.
Outb6	BOO	Value of Bit 6 in BCD format of the whole number in the InDec input.
Outb7	BOO	Value of Bit 7 in BCD format of the whole number in the InDec input.
Outb8	BOO	Value of Bit 8 in BCD format of the whole number in the InDec input.

Outb9	BOO	Value of Bit 9 in BCD format of the whole number in the InDec input.
OutbA	BOO	Value of Bit 10 in BCD format of the whole number in the InDec input.
OutbB	BOO	Value of Bit 11 in BCD format of the whole number in the InDec input.
OutbC	BOO	Value of Bit 12 in BCD format of the whole number in the InDec input.
OutbD	BOO	Value of Bit 13 in BCD format of the whole number in the InDec input.
OutbE	BOO	Value of Bit 14 in BCD format of the whole number in the InDec input.
OutbF	BOO	Value of Bit 15 in BCD format of the whole number in the InDec input. Most significant bit.

## Description:

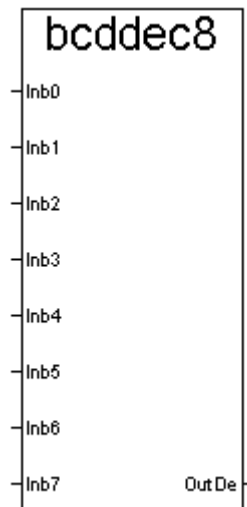
This block allows converting an integer variable at the InDec input into 16 Boolean variables (from Outb0 to OutbF), which represent the input decimal number in BCD format.

For the BCD format, every single digit of the decimal number to be converted must be represented by 4 bits (nibble). Therefore, a three digit decimal number is converted into a 12-bit binary number (3 nibbles). E.g. 99 decimal => 10011001 binary. Therefore, each nibble can vary from 0 (0000b) to 9 (1001b).

Numbers with decimal points cannot be represented. The highest representable number is 9999.

The block does not provide for activation. Every time it's recalled, it converts the input datum into the 16 output data.

## BCDDec8



### Topics:

Inb0	BOO	Bit 0 of the BCD whole number to be present on the OutDe output. Less significant bit.
Inb1	BOO	Value of Bit 1 of the whole number to be present at the OutDe output.
Inb2	BOO	Value of Bit 2 of the whole number to be present at the OutDe output.
Inb3	BOO	Value of Bit 3 of the whole number to be present at the OutDe output.
Inb4	BOO	Value of Bit 4 of the whole number to be present at the OutDe output.
Inb5	BOO	Value of Bit 5 of the whole number to be present at the OutDe output.
Inb6	BOO	Value of Bit 6 of the whole number to be present at the OutDe output.
Inb7	BOO	Value of Bit 7 of the whole number to be present at the OutDe output. Most significant bit.
OutDe	INT	8-bit whole number that represents the combination of the single bits in their respective Boolean Inbnn inputs in BCD format.

### Description:

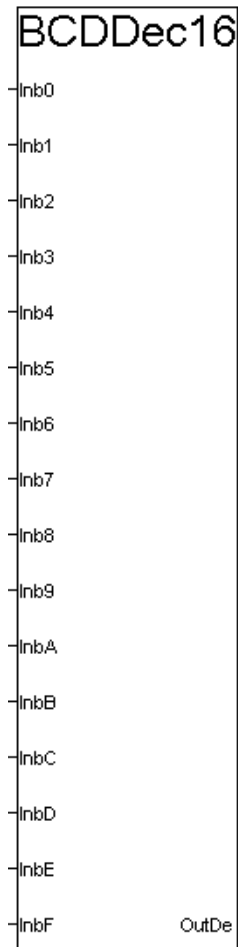
This block allows converting 8 Boolean variables, represented by a number in BCD format at inputs from Inb0 to Inb7, into an integer variable.

For the BCD format, every single digit must be represented by 4 bits (nibble). Therefore, a three digit decimal number is converted into a 12-bit binary number (3 nibbles). E.g. 99 decimal => 10011001 binary. Therefore, each nibble can vary from 0 (0000b) to 9 (1001b).

Numbers with decimal points cannot be represented. The highest representable number is 9999.

The block does not provide for activation. Every time it's recalled, it converts the input datum into the 16 output data.

## BCDDec16



### Topics:

Inb0	BOO	Bit 0 of the BCD number to be present on the OutDe output. Less significant bit.
Inb1	BOO	Value of Bit 1 of the BCD whole number to be present at the OutDe output.
Inb2	BOO	Value of Bit 2 of the BCD whole number to be present at the OutDe output.
Inb3	BOO	Value of Bit 3 of the BCD whole number to be present at the OutDe output.
Inb4	BOO	Value of Bit 4 of the BCD whole number to be present at the OutDe output.
Inb5	BOO	Value of Bit 5 of the BCD whole number to be present at the OutDe output.
Inb6	BOO	Value of Bit 6 of the BCD whole number to be present at the OutDe output.
Inb7	BOO	Value of Bit 7 of the BCD whole number to be present at the OutDe output.

Inb8	BOO	Value of Bit 8 of the BCD whole number to be present at the OutDe output.
Inb9	BOO	Value of Bit 9 of the BCD whole number to be present at the OutDe output.
InbA	BOO	Value of Bit 10 of the BCD whole number to be present at the OutDe output.
InbB	BOO	Value of Bit 11 of the BCD whole number to be present at the OutDe output.
InbC	BOO	Value of Bit 12 of the BCD whole number to be present at the OutDe output.
InbD	BOO	Value of Bit 13 of the BCD whole number to be present at the OutDe output.
InbE	BOO	Value of Bit 14 of the BCD whole number to be present at the OutDe output.
InbF	BOO	Value of Bit 15 of the BCD whole number to be present at the OutDe output. Most significant bit.
OutDe	INT	16-bit whole number in BCD format that represents the combination of the single bits in their respective Boolean Inbnn inputs in BCD format.

### Description:

This block allows converting 16 Boolean variables, represented by a number in BCD format at inputs from Inb0 to Inb7, into an integer variable.

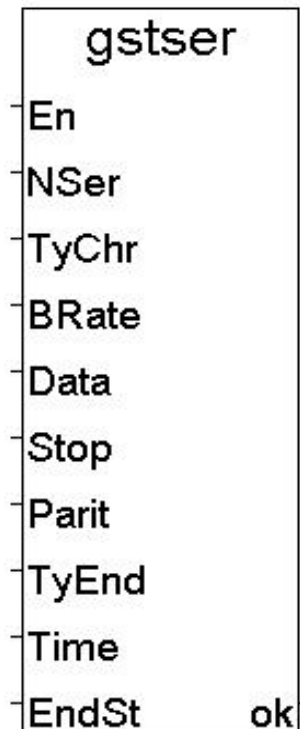
For the BCD format, every single digit must be represented by 4 bits (nibble). Therefore, a three digit decimal number is converted into a 12-bit binary number (3 nibbles). E.g. 99 decimal => 10011001 binary. Therefore, each nibble can vary from 0 (0000b) to 9 (1001b).

Numbers with decimal points cannot be represented. The highest representable number is 9999.

The block does not provide for activation. Every time it's recalled, it converts the input datum into the 16 output data.



## GstSer



### Topics:

En	BOO	The positive edge enables or disables the serial port management from the application program.
NSer	INT	Number of the serial port to be used.
TyChr	INT	0 = characters are managed in binary code 1 = characters are managed in ASCII code
Brate	INT	Indicates the required value for the serial port BaudRate. The codes to be entered are: 0 → Deactivates the serial port management from the application 1 → BaudRate=57600 bit/sec (not available) 2 → BaudRate=38400 bit/sec 3 → BaudRate=19200 bit/sec 4 → BaudRate=9600 bit/sec 5 → BaudRate=4800 bit/sec (not available)
Data	INT	Number of data bits (7 or 8)
Stop	INT	Number of stop bits (1 or 2)
Parit	INT	Type of parity 0. No parity 1. Even parity 2. Odd parity

TyEnd	INT	Type of termination for incoming messages 0 => None 1 => Time-set termination 2 => String termination
Time	INT	Used in case of time-set termination, it indicates the time in ms that must elapse from the last bit received in order to consider the message completed. This value can be rounded up or down to 1 ms by the control.
EndSt	MSG	Used in case of string termination, it indicates a string with one or more characters that indicates that the incoming message is completed.
Ok	INT	If the activation or deactivation of the serial port management is completed successfully, it shall be set to 1.

## Description:

This block, together with ReadSer and WriteSer described later on, allows the user to manage a serial line through an application that manages a customised protocol for sending and receiving strings with communication features selected by the user.

The CND51 (or SU21X) control does not manage the protocol but it sends strings prepared by the user through the application and returns the received messages, completed as required by the user, on a serial port.

The management of the serial port is activated on the positive edge of the En input.

To deactivate it, set Brate = 0 and generate a positive edge on the En input.

The number of the serial port to be used must be compatible with the other protocols managed by the data card. In particular, the management of the serial port cannot be activated by an application for a serial port configured for a different protocol (e.g.: for the ISaGRAF WorkBench). For further information on the possible configurations of the serial ports, see the customisation manual - ConfSer Block.

The TyChr input allows selecting whether the characters to be sent or received must be binary or ASCII-encoded.

Brate, Data, Stop and Parit inputs indicate the serial port configuration parameters.

**Attention:** should COM0 be selected, it will not be possible to set 7 bits data and any parity. This is the only combination that is not managed. The block signals an error.

The TyEnd input allows selecting the type of incoming message termination.

In case of string termination, the message is considered completed when all the characters of the termination string set in EndSt have been received.

In case of time-set termination, the message will be considered completed for an amount of time set by the Time parameter.

In case of no termination, it is the user who indicates the message termination using the ReadSer block (see ReadSer block description).

## ReadSer



### Topics:

En	BOO	On the positive edge, it enables the display of the messages received by the serial port. Such messages are recorded even when En is FALSE.
NSer	INT	Number of the serial port where to read the messages. It must be initialised with the GstSer block.
res	BOO	Resets the indication of completed message and, in case of protocol without termination, it allows resetting the message output buffer (see detailed description).
Ok	BOO	Indicates the message termination. In case of protocol without terminator, it is set to TRUE upon receiving the first character after reset.
Nchr	INT	Number of received characters (when ASCII-encoded) or bytes (when binary-encoded)
Mess	MSG	Received message

### Description:

This block displays the messages received on the serial port. It must be initialised with the GstSer block.

The En input enables the display of the messages received. If En is FALSE, the data card continues to record the received characters internally; however, the output buffer is not updated.

Below is a detailed description of the message reception phase.

- ◆ **String termination (TyEnd=2).** When the serial port receives the characters that define the end of the message, the OK output is set to TRUE; Nchr indicates the number of characters received (without counting those of the termination) and the message (without termination) is displayed on the Mess output. Upon receiving a new termination, the previous values are overwritten with new ones. To reset the OK output, RES must be set to TRUE.  
When binary-encoded, Nchr indicates the number of received bytes and the Mess output displays the byte hex encoding, i.e., for each byte 2 characters are obtained.
  
- ◆ **Time-set termination (TyEnd=2).** When an amount of time equal to that set in the GstSer block (Time parameter) elapses on the serial port from the last character received, the message is considered completed. The OK output is set to TRUE; Nchr indicates the number of received characters and the message is displayed on the Mess output. Once the subsequent message is completed, the previous values are overwritten with new ones. To reset the OK output, RES must be set to TRUE.  
When binary-encoded, Nchr indicates the number of received bytes and the Mess output displays the byte hex encoding, i.e., for each byte 2 characters are obtained.
  
- ◆ **Protocol without termination (TyEnd=0).** Upon receiving the first character, the OK output is set to TRUE, all the other characters received afterwards are queued and displayed on the Mess output and Nchr indicates their number. The message is considered completed when the res input is activated and the OK output is also reset. Upon receiving the first character following the res, the buffer is emptied.  
When binary-encoded, Nchr indicates the number of received bytes and the Mess output displays the byte hex encoding, i.e., for each byte 2 characters are obtained.

Should the characters be binary-encoded (TyChr=0), the message will consist in a string that represents the hex value of the received data.  
(e.g.: If a byte "0100 0001" is received, its hex encoding "41" is displayed).

## WriteSer



### Topics:

En	BOO	The positive edge transfers the message to the serial line.
NSer	INT	Number of the serial port where to read the messages. It must be initialised with the GstSer block.
Mess	MSG	Message to be sent.
Ok	INT	Indicates that the message has been sent.

### Description:

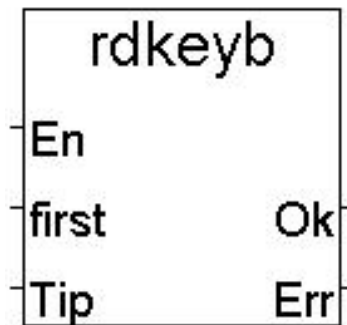
This block allows sending a string of characters or bytes to the serial port. It must be initialised with the GstSer block.

In case of binary encoding, the string must represent the hex encoding of the bytes to be sent; since for every byte there are 2 characters to be set, the string must consist of an even number of elements.

(e.g.: to send the byte "0100 0001", the string "41", which represents its hex encoding, must be set)

The control does not carry out any verification on the sent messages, every action or processing is left to the application program. Even in case of Protocols with string termination, for sending every message with the correct termination, it is the user who must queue the appropriate termination characters to the message.

## RdKeyb



## Topics:

En	BOO	On the positive edge it activates the reading of the keyboard keys and switches the LEDs on.
first	INT	Address of the first ISaGRAF variable containing the keyboard key and LED status
Tip	INT	Not used
Ok	BOO	Keyboard activated
Err	INT	Error warning: 0 -> No error 1 -> Incorrect parameters 2 -> Undeclared variables 3 -> Non-Boolean variables

## Description:

This block allows managing the KD keyboard for SU210, SU110 and SU112.

## SU210 data card

In the application there must be at least 46 ISaGRAF Boolean variables with consecutive addresses, which are used for such management and the first input of the block must contain the address of the first variable.

The keyboard is provided with 35 keys and 11 LEDs. The first 35 variables represent the status of the keys: by pressing a key, the corresponding variable assumes the value true.

The subsequent 11 variables represent the LED control: by setting a variable to true, the corresponding LED is activated.

Below is a map of corresponding variables/keys (or LEDs) for SU210.

var. 1) - SW1	User 1 key
var. 2) - SW2	F1
var. 3) - SW3	AUT
var. 4) - SW4	Key 1
var. 5) - SW5	Key 2
var. 6) - SW6	Key 3
var. 7) - SW7	Key 4
var. 8) - SW8	Key 5
var. 9) - SW9	User 2 key
var. 10) - SW10	F2
var. 11) - SW11	+ key
var. 12) - SW12	Key 6
var. 13) - SW13	Key 7
var. 14) - SW14	Key 8
var. 15) - SW15	Key 9
var. 16) - SW16	Key 0
var. 17) - SW17	User 3 key
var. 18) - SW18	F3
var. 19) - SW19	Down arrow
var. 20) - SW20	MAN
var. 21) - SW21	DEL
var. 22) - SW22	Up arrow
var. 23) - SW23	BSP
var. 24) - SW24	START
var. 25) - SW25	User 4 key
var. 26) - SW26	F4
var. 27) - SW27	- key
var. 28) - SW28	PROG
var. 29) - SW29	Left arrow
var. 30) - SW30	SHIFT
var. 31) - SW31	Right arrow
var. 32) - SW32	STOP
var. 33) - SW33	User 5 key
var. 34) - SW34	F5
var. 35) - SW35	ENTER
var. 36) - DL1	WD LED
var. 37) - DL2	RUN LED
var. 38) - DL3	SIM LED
var. 39) - DL4	EMG LED
var. 40) - DL5	LOW BATT LED
var. 41) - DL6	CAN LED
var. 42) - DL7	User 1 LED
var. 43) - DL8	User 2 LED
var. 44) - DL9	User 3 LED
var. 45) - DL10	User 4 LED
var. 46) - DL11	User 5 LED

## SU110 data card

In the application there must be at least 37 ISaGRAF Boolean variables with consecutive addresses, which are used for such management and the first input of the block must contain the address of the first variable.

The keyboard is provided with 37 keys but there are no LEDs. The 37 variables represent the status of the keys: by pressing a key, the corresponding variable assumes the value true.

## SU112 data card

In the application there must be at least 48 ISaGRAF Boolean variables with consecutive addresses, which are used for such management and the first input of the block must contain the address of the first variable.

The keyboard is provided with 40 keys and 8 LEDs. The first 40 variables represent the status of the keys: by pressing a key, the corresponding variable assumes the value true.

The subsequent 8 variables represent the LED control: by setting a variable to true, the corresponding LED is activated.

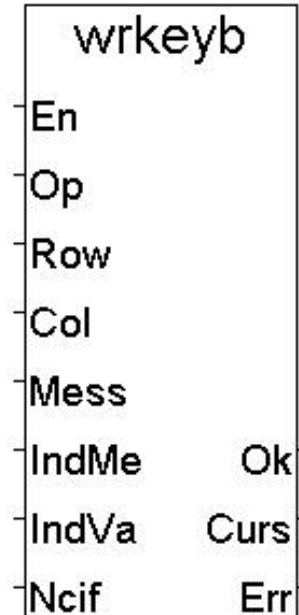
Below is a map of corresponding variables/keys (or LEDs) for SU110 and SU112.

var. 1) - SW1	Dot
var. 2) - SW2	Key 0
var. 3) - SW3	- key
var. 4) - SW4	STOP
var. 5) - SW5	AUT
var. 6) - SW6	ENTER
var. 7) - SW7	User g key (only for SU112)
var. 8) - SW8	User h key (only for SU112)
var. 9) - SW9	Key 7
var. 10) - SW10	Key 8
var. 11) - SW11	Key 9
var. 12) - SW12	START
var. 13) - SW13	Down arrow
var. 14) - SW14	ESC
var. 15) - SW15	User e key (only for SU112)
var. 16) - SW16	User f key (only for SU112)
var. 17) - SW17	Key 4
var. 18) - SW18	Key 5
var. 19) - SW19	Key 6
var. 20) - SW20	Left arrow
var. 21) - SW21	SHIFT
var. 22) - SW22	Right arrow
var. 23) - SW23	User c key (only for SU112)
var. 24) - SW24	User d key (only for SU112)



var. 25) - SW25 Key 1  
var. 26) - SW26 Key 2  
var. 27) - SW27 Key 3  
var. 28) - SW28 DEL  
var. 29) - SW29 Up arrow  
var. 30) - SW30 PROG  
var. 31) - SW31 User a key (only for SU112)  
var. 32) - SW32 User b key (only for SU112)  
var. 33) - SW33 F1  
var. 34) - SW34 F2  
var. 35) - SW35 F3  
var. 36) - SW36 F4  
var. 37) - SW37 F5  
var. 38) - SW38 HELP (only for SU112)  
var. 39) - SW39 F6 (only for SU112)  
var. 40) - SW40 F7 (only for SU112)  
var. 41) - DL1 User LED (only for SU112)  
var. 42) - DL2 User LED (only for SU112)  
var. 43) - DL3 User LED (only for SU112)  
var. 44) - DL4 User LED (only for SU112)  
var. 45) - DL5 User LED (only for SU112)  
var. 46) - DL6 User LED (only for SU112)  
var. 47) - DL7 User LED (only for SU112)  
var. 48) - DL8 User LED (only for SU112)

## WrKeyb



### Topics:

En	BOO	On the positive edge it activates the operation required for the keyboard display.
Row	INT	Number of the row (for SU210 its value is 0, 1 or 2, for SU110 0, 1, 4 and for SU112 0, 1, 8). If its value is 0 the current row is considered.
Col	INT	Number of the column (for SU210 0,1,2,...40, for SU110 0,1,2,...20, for SU112 0,1,.....,21). If its value is 0 the current column is considered.
Mess	MSG	String to be written on the display
IndMes	INT	Address of the variable (ISaGRAF or virtual) containing the string of characters to be printed, or the pattern of the character to be defined, or the pattern of the graphic column to be printed
IndVar	INT	Address of the ISaGRAF variable that stores the read value
Ncif	INT	Maximum number of characters to be entered

Op	INT	Required operation: 0 -> No operation 1 -> Writes a string on the display withdrawing it from the MESS input 2 -> Writes a string on the display withdrawing it from a variable whose address is indicated 3 -> Deletes the display 4 -> Deletes a row of the display 5 -> Displays the cursor and activates the blink 6 -> Displays the cursor and deactivates the blink 7 -> Hides the cursor and activates the blink 8 -> Hides the cursor and deactivates the blink 9 -> Places the cursor in the required position on the display 10 -> Prints the string taking it from Mess and waits for a value to be allocated to a variable entered via the keyboard 11 -> Prints the string taking it from IndMes address and waits for a value to be allocated to a variable entered via the keyboard 12 -> Shifts the cursor to the right as many times as indicated in Ncif 13 -> Shifts the cursor to the left as many times as indicated in Ncif 14 -> Enters as many blank characters as indicated in Ncif 15 -> Ends the entering of values via keyboard without allocating any value to the variable 16 -> Defines a user character withdrawing the pattern from the MESS input 17 -> Defines a user character withdrawing the pattern from a variable whose address is indicated 18 -> Displays the set of characters 19 -> Writes the single graphic columns on the display withdrawing the pattern directly from the input 20 -> Writes the single graphic columns on the display withdrawing the pattern from an isagraf variable message type whose address is indicated 21 -> Writes the single graphic columns on the display withdrawing the pattern from one or more analogue isagraf variable whose address of the first variable is indicated 22 -> Writes the single graphic columns on the display withdrawing the pattern from one or more virtual variables whose address of the first one is indicated
----	-----	---

Ok	BOO	TRUE = operation completed successfully
Curs	INT	Position of the cursor (for SU210 and SU110 from 1 to 80; for SU112 with graphic LCD from 1 to 168)
Err	INT	Error warning: 0 -> No error 1 -> Incorrect parameters 2 -> String too long 3 -> The indicated variable is of a different type other than the required one 4 -> Error in display management 5 -> The indicated variable does not exist 6 -> Ncif not compatible with the cursor position 7 -> Operation not possible because the display is in data entering mode 8 -> Operation not provided for

### Description:

This block allows managing the KD keyboard display from the ISaGRAF application.

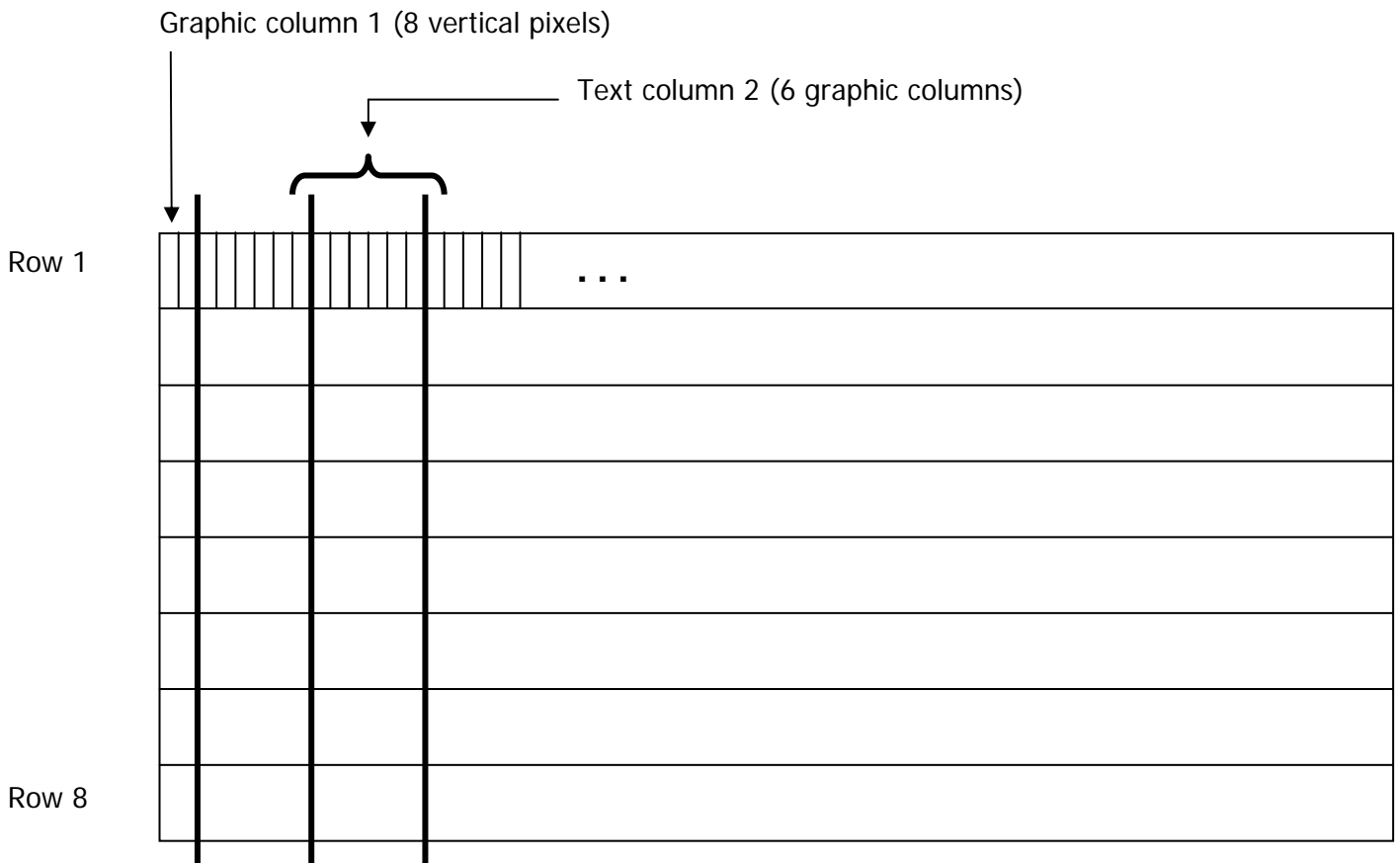
For the SU210 data card: the display is divided into 2 rows of 40 columns (2x40)

For the SU110 data card: the display is divided into 4 rows of 20 columns (4x40)

For the SU112 data card: the display can be alphanumeric (in such case, it's the same as for the SU110 data card) or graphic (64x128 pixel).

The latter consists of a set of 8 rows of 128 columns (that we'll call **graphic columns**, each consisting of 8 vertical pixels). The graphic display is also provided with text management: in such case, the character is given by 6 adjacent graphic columns (obtaining an 8x21 display, where 21 are what we'll call **text columns**); note that  $128/6=21$  with 2 over; therefore, for every row, the first two graphic columns must be skipped. (See figure below)

Finally, as for the graphic display, the user can define the characters with identification code between 221 and 255; we will call **ASCII code** the identification code (from 0 to 255) of the character in question, without taking into account the standardisation of the patterns of all characters.



The Op input allows selecting the operation to be carried out:

- 0 - No operation
- 1 - Writes on the display the string contained in the Mess input. The string is written starting from the position indicated by the Row and Col inputs (row and column respectively). If the value of such inputs is 0, the current row or column is considered.
- 2 - Writes on the display the string contained in the variable whose address is indicated in the Ind input. The string is written starting from the position indicated by the Row and Col inputs (row and column respectively). If the value of such inputs is 0, the current row or column is considered.
- 3 - Deletes the entire display and repositions the cursor at the start
- 4 - Deletes the row indicated by Row (the current one if row=0) and places the cursor at the beginning of such row
- 5 - Displays the cursor and activates the blink (only for text displays)
- 6 - Displays the cursor and deactivates the blink (only for text displays)

- 7 - Hides the cursor and activates the blink (only for text displays)
- 8 - Hides the cursor and deactivates the blink (only for text displays)
- 9 - Places the cursor on the required row and column; if their value is 0, the current row or column are considered)
- 10 - Prints a string withdrawing it from the Mess input and then waits for a numeric value to be entered via the keyboard to be assigned to the variable whose address is indicated in the IndVar input
- 11 - Prints a string withdrawing it from the IndMes input and then waits for a numeric value to be entered via the keyboard to be assigned to the variable whose address is indicated in the IndVar input
- 12 - Shifts the cursor to the right as many times as indicated in the Ncif input
- 13 - Shifts the cursor to the left as many times as indicated in the Ncif input
- 14 - Enters as many blank characters as indicated in the Ncif input
- 15 - Exits the value entering mode without assigning any value to the variable
- 16 - Defines the shape of the character by means of a hex string provided directly by the Mess input. In this mode, Row and Ncif inputs are not considered, while Col contains the ASCII code of the character to be defined; users can define only characters with ASCII code between 221 and 255 (only for graphic display)
- 17 - Defines the shape of the character by means of a hex string provided by an Isagraf® variable, whose address is indicated in the IndVar input. In this mode, Row and Ncif inputs are not considered, while Col contains the ASCII code of the character to be defined; users can define only characters with ASCII code between 221 and 255 (only for graphic display)
- 18 - Writes on the display the set of available characters (the graphic display also includes characters that can be defined by the user). Since the complete set of characters consists of 256 elements, it cannot be entirely displayed and, therefore, it must be divided into pages; the page in particular shall be selected by providing the corresponding value in the Col input (the first page is obtained providing 1, while the last one depends on the type of display). If the Col value is higher than the last page's value, error code 1 (incorrect parameter) will be provided.  
The valid page values:  
for the SU210 data card -> from 1 to 4  
for the SU110 and SU112 data cards with alphanumeric display -> from 1 to 4  
for the SU112 data card with graphic display -> from 1 to 3  
In this mode, the Row and Ncif inputs are not considered.

- 19 - Writes on the display a series of consecutive graphic columns, whose definition is provided directly by the Mess input by means of a hex string; The display is written starting from the position indicated by the Row and Col inputs (row and column respectively): see examples later on. If the value of such inputs is 0, the current row or column is considered. Ncif contains the number of graphic columns that are written, and must correspond to half the length of the string provided by Mess. (only for graphic display)
  
- 20 - Writes on the display a series of consecutive graphic columns, whose definition is provided by the hex string contained in an Isagraf® message variable, whose address is indicated in the IndVar input. The display is written starting from the position indicated by the Row and Col inputs (row and column respectively): see examples later on. If the value of such inputs is 0, the current row or column is considered. Ncif contains the number of graphic columns that are written, and must correspond to half the length of the string provided to Mess. (only for graphic display)
  
- 21 - Writes on the display a series of consecutive graphic columns, whose definition is withdrawn by one or more Isagraf® analogue variables, whose address of the first variable is indicated in the IndVar input. Each variable contains the definition of 4 columns, and Ncif determines its number, representing the number of columns to be screen printed. The display is written starting from the position indicated by the Row and Col inputs (row and column respectively): see examples later on. If the value of such inputs is 0, the current row or column is considered. (only for graphic display)
  
- 22 - Writes on the display a series of consecutive graphic columns, whose definition is withdrawn from one or more Virtual variables, whose address of the first variable is indicated in the IndVar input. Each variable contains the definition of 4 columns, and Ncif determines the number of used variables, representing the number of columns to be screen printed. The display is written starting from the position indicated by the Row and Col inputs (row and column respectively): see examples later on. If the value of such inputs is 0, the current row or column is considered. (only for graphic display)

Operations 1 and 2 allow writing a character through its base-10 3-digit ASCII code. Before using this code, a '\' (backslash) must be entered in the string. To write the backslash, '\\' must be entered in the string.

Example: The string 'abc\033ABC\\abc' generates 'abc!ABC\abc' on the display being 033 the ASCII code of the exclamation point.

NOTE:

Various displays do not follow the standard ASCII code exactly for all the characters; therefore, it is normal for the alphanumeric display to print a different character other than the backslash; however, the numbers and letters (both upper and lowercase) are always ensured.

Operations 10 and 11 place the display in value entering mode. In this mode, the cursor flashes and the user can enter a numeric value, with a maximum number of characters indicated in the Ncif input, to be entered via the keyboard.

If Ncif is such that its value exceeds the end of the row and continues in the following, an error will occur.

The accepted values are numbers between 0 and 9 and the minus (-) sign to indicate negative numbers.

Press BSP (Backspace) to delete the last digit entered in the SU210 data card, as for SU110 and SU112 data cards, press DEL.

To confirm the entered value, press ENTER. To exit the value entering mode without assigning any value to the SU210 data card, press DEL; as for the SU110 and SU112 data cards, press ESC, or enter control 15.

Operations 16, 17, 19 and 20 require the definition of the character or of the graphic column by means of a hex string, where the first value represents the lower part of the first column; the second value, the upper part of the first column; the third value the lower part of the second column, and so on.

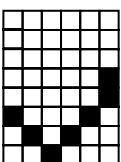
Each character is defined by 6 8-point columns and each column by two hex digits, for a total 12 hex digits per character. With operations 16 and 17, the entry of strings with lengths other than 12 is indicated as an error, while operations 19 and 20 allow strings of arbitrary length, as long as they are even, in order to print more rows.

Operations 21 and 22 define an arbitrary number of rows, using one variable every 4 columns. The first graphic column is described by the most significant byte (8 highest bits) of the 32-bit variable, and every byte represents the binary code of the column translated into decimal.



**Example 1:** Suppose you want to write the column on the side (black pixels indicate the illuminated pixels):

Having only one column implies that Ncif=1; therefore, one single variable will be sufficient and only whose highest byte will be written. Starting from the lowest pixel, the binary representation of the column is written, replacing the illuminated pixels with '1' and the other pixels with '0'; '00010011' is obtained. The other 3 bytes of the variable are not used; therefore, the binary representation of the variable ranges between the values '00010011,00000000,00000000,00000000' and '00010011,11111111,11111111,11111111', corresponding to decimal values 318767104 and 335544319 (the Windows scientific calculator can be used for the conversion)



**Example 2:** suppose you want to draw the symbol on the side. The columns are 6, therefore Ncif=6, and the used variables are 2. The binary representation of the first variable is given by the 4 columns to the far left: '00100000,01000000,10000000,01000000', equal to 541098048. The second variable will contain the definitions of columns 5 and 6: '00100000,00011000,xxxxxxx,xxxxxxx', where x indicates 1 or 0. Considering all x as 0, the decimal value 538443776 is obtained.



With display writing on the current position (Row and/or Col set to 0), both graphic columns and characters can be written consecutively; however, it is important to remember that, unlike graphic columns, characters are written on defined positions on the display.

The Curs output shows the cursor position at all times.

For the SU210 data card alphanumeric display:

- from 1 to 40 -> characters of the first row
- from 41 to 80 -> characters of the second row

For the SU110-SU112 data cards alphanumeric display:

- from 1 to 20 -> characters of the first row
- from 21 to 40 -> characters of the second row
- from 41 to 60 -> characters of the third row
- from 61 to 80 -> characters of the fourth row

For the SU112 data card alphanumeric display:

- from 1 to 21 -> characters of the first row
- from 22 to 42 -> characters of the second row
- from 43 to 63 -> characters of the third row
- from 64 to 84 -> characters of the fourth row
- from 85 to 105 -> characters of the fifth row
- from 106 to 126 -> characters of the sixth row
- from 127 to 147 -> characters of the seventh row
- from 148 to 168 -> characters of the eighth row

## Table of the graphic display characters (only for SU112)

ASCII CODE 0 ~ 31: CORRESPONDING TO CONTROL CHARACTERS (EMPTY)

ASCII CODE 221 ~ 255: AVAILABLE FOR USER DEFINITIONS

32														46
SPACE	!	"	#	\$	%	&	'	(	)	*	+	,	-	.
47														61
/	0	1	2	3	4	5	6	7	8	9	:	;	<	=
62														76
>	?	@	A	B	C	D	E	F	G	H	I	J	K	L
77														91
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[
92														106
\	]	^	_	`	a	b	c	d	e	f	g	h	i	j
107														121
k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
122														136
z	{		}	~		Ç	ü	é	â	ä	à	å	ç	ê
137														151
ë	è	ï	î	ì	Ä	Å	É	æ	Æ	ô	ö	ò	ù	û
152														166
ÿ	Ö	Ü	¢	£	¥	Pt	f	á	í	ó	ú	ñ	Ñ	ª
167														181
°	¿	®	¬	½	¼	¡	«	»	€	♫	♯	!		
182														196
197														211
SIGMA				ALFA	BETA	GAMMA	DELTA	EPSILON	FI	LAMBDA	MU	OMEGA	PI	RO
212														220
TAU	TETA	PSI	DELTA MAIUSC	FI MAIUSC	LAMBDA MAIUSC	OMEGA MAIUSC	PI MAIUSC	SIGMA MAIUSC						

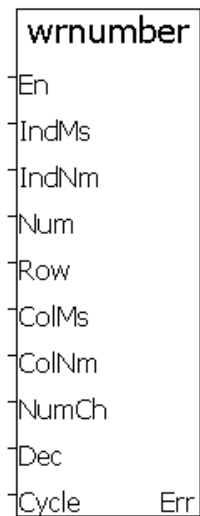
## Table of alphanumeric display characters

The following table is provided by the alphanumeric display manufacturer, and the ASCII code that identifies the character is provided as a hex code.

EXAMPLE OF READING: 'R' is identified by column 5 and row 2, thus obtaining the hex ASCII code 50, equal in decimals to:  $5 \times 16 + 2 = 82$ .

		Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)																		
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
Lower 4-bit (D0 to D3) of Character Code (Hexadecimal)	0 CG RAM (1)			0	Q	P	'	P						-	夕	≡	α	p		
	1 CG RAM (2)		!	1	A	Q	a	a						。	ア	チ	△	ä	q	
	2 CG RAM (3)		"	2	B	R	b	r							「	イ	ツ	×	ρ	θ
	3 CG RAM (4)		#	3	C	S	c	s							」	ウ	テ	ε	ε	ω
	4 CG RAM (5)		\$	4	D	T	d	t							、	エ	ト	μ	μ	Ω
	5 CG RAM (6)		%	5	E	U	e	u							・	オ	ナ	1	ü	ü
	6 CG RAM (7)		&	6	F	V	f	v							ヲ	カ	ニ	ヨ	ρ	Σ
	7 CG RAM (8)		'	7	G	W	g	w							ア	キ	ヌ	ラ	g	π
	8 CG RAM (1)		(	8	H	X	h	x							イ	ウ	ネ	リ	5	Σ
	9 CG RAM (2)		)	9	I	Y	i	y							ウ	ケ	ル	'	ü	ü
	A CG RAM (3)		*	:	J	Z	j	z							エ	コ	ハ	レ	j	キ
	B CG RAM (4)		+	:	K	[	k	[							オ	サ	ヒ	ロ	*	π
	C CG RAM (5)		,	<	L	¥	l	l							カ	シ	フ	ワ	φ	π
	D CG RAM (6)		-	=	M	]	m	]							ユ	ス	ハ	ン	も	÷
	E CG RAM (7)		.	>	N	^	n	^							ヨ	セ	ホ	'	ñ	ñ
	F CG RAM (8)		/	?	O	_	o	_							ッ	ッ	マ	'	ö	■

## WrNumber



### Topics:

En	BOO	When active, it continuously shows the number on the display.	
IndMs	INT	Possible message string address to be displayed with the number: in such case, the string is written only once on the positive edge of En.	
IndNm	INT	Variable address, whose number must be displayed.	If
Num	INT	Value to be shown if the IndNm input is equal to 0.	
Row	INT	Row where to write the number and/or the string.	
ColMs	INT	Column of the beginning of the message at the IndMs input.	
ColNm	INT	Starting column of the number at the IndNm or Num inputs.	
NumCh	INT	Number of characters, comma included, for displaying the number.	
Dec	INT	Number of decimals of the number to be displayed.	
Cycle	INT	Refresh time of the number expressed in msec. Such time is limited by the plc loop time and by the frequency of the block recall.	
Err	INT	Error occurred. If 0, the operation did not encounter any problems.	

## Description:

This block allows displaying numbers, even with decimals, always aligned to the right with a preset size, with display of number that is too large to be displayed by means of the printing of a series of question marks (?). The number to be displayed can be provided as a direct variable or by entering the virtual address of the variable itself. The number of decimals will determine how many digits to enter after the "." symbol; if zero, the number shall be displayed as an integer. The Cycle parameter allows deciding after how many msec to refresh the number, provided that it cannot occur more than once for every PLC cycle; therefore, the refresh time is limited by the cycle time. The Row input determines the row of the display where to write the number and/or the text string; ColIMs indicates the column where to start writing the text message (IndMs) and ColNm indicates the column where to start writing the number. The text is written only once on the positive edge of the En input and it can be overwritten by the number to be displayed. This allows setting a string with the name of the variable to be written, followed by a series of spaces that will be filled by the number followed by the measurement unit of the number.

(e.g.: message = "position           mm"  
          number =           "200.22"

This way, with one single block, data can be displayed with an automatic update and alignment.

## ChgTAssi



### Topics:

En	BOO	On the positive edge it activates the block
Time	INT	It assigns the time corresponding to the interval between two consecutive calls of the axis management routine expressed in ms
Res	BOO	Resets the Ok output
Err	INT	Emits the error code: 0= no error 1= Time value not valid 2= axis not stopped 3= hardware version that does not support the block
Ok	BOO	To be set when TIME is actually assigned
Tatt	INT	Represents the TIME value currently used

### Description:

This block allows assigning the time required to elapse between two consecutive calls of the axis management routine (Time input). During its use, the axis must be stopped.

When setting such time, it is important to remember that values that are too high degrade the axis performance, while values that are too low take time away from other PLC activities. Currently, the accepted values range between 2 ~ 13 (ms), and the Tatt output shows the value currently used. The Ok output is set when the set value is accepted; Res allows resetting them.

It can only be used on SU210.

On SU212, it only affects the axes on the basic data card.

## Set\_Ip



### Topics:

En	BOO	On the positive edge it activates the block
IP	MSG	Assigns the IP address of the Ethernet device within the LAN. The number must consist of 4 digits divided by the <i>point</i> character. (e.g.: 192.168.10.1
mask	MSG	Assigns the mask for evaluating the IP address within the LAN. The number must consist of 4 digits divided by the <i>point</i> character. (e.g.: 255.255.255.0
gatew	MSG	Indicates the address of the LAN Gateway to the Ethernet control. The number must consist of 4 digits divided by the <i>point</i> character. (e.g.: 192.168.10.100
Ok	BOO	The true status indicates the successful initialisation of the Ethernet board.
Err	INT	Emits the error code: 0= no error ≠0 = error. Contact Arteco spa

### Description:

This block allows assigning the IP address, the mask and gateway to the Ethernet board that contains the web server for displaying statistical Html pages or with particular codes to display the status of the internal variables.

For coding the instructions that allow displaying the values of internal variables, contact ARTECO SpA.

## Annex A – Virtual Variables

The term **virtual variable** refers to an area of the memory available for storing variables that can only be managed by Artec blocks without having to previously declare them.

There are 8000 virtual variables available with indexes ranging between 1 and 8000 included. Such variables are integer variables and can have values ranging between +2.147.483.647 and – 2.147.483.648.

Artec blocks intended for managing such variables are VarIndex, which allows reading and writing a single variable and VarCpy, which allows copying sets of variables from one area to another, with the option to copy variable values from the Virtual variable area to the ISaGRAF standard variables and vice-versa.

Virtual variables provide the user with a considerable number of variables without requiring their declaration.

An example could be a program whose data vary according to the processed piece. N variables can be declared containing the data of the piece to be processed and store the data of the various pieces in the virtual variables.

E.g.

If the program in progress uses 20 variables to process a piece and 10 pieces to be processed more commonly are to be stored, with the ISaGRAF variables  $20 \times 10 = 200$  variables must be declared.

Alternatively, 200 values can be stored in the virtual variables and the 20 variables of the piece to be processed from the virtual area can be recalled before starting processing, placing them in 20 standard ISaGRAF variables by means of the VarCpy control.

Values cannot be entered inside the virtual area in floating point format, but variables of different types can be transferred by means of the VarCpy control (see block description).

Virtual variables are all of Integer Analogue type.

Virtual variables are retained variables; therefore, after switching the data card off and then back on, they maintain their value.

From the v68 firmware version onwards, virtual variables maintain their value even after removing the application and downloading a new one. They are reset by resetting the data card.



## Annex B – Utility for reading and writing variables via serial ports

This utility allows reading the value of the variables of an ISaGRAF application (both standard and virtual variables) by means of a serial port. The values of all the variables are returned in a text file that can be easily modified. The generated files can be used to enter values in a different control or on the same one should they be lost.

This utility is called RWISaGRAFVar and must be launched from a DOS window or from resource management, in any case on a PC with Windows 95 or Windows 98 operating system.

Figure 1 shows the first request presented by the utility after execution.

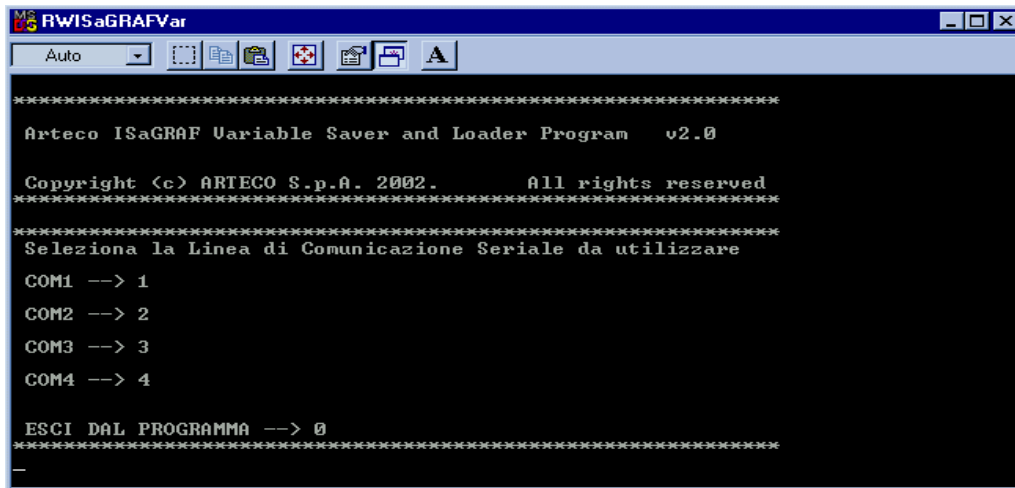


Fig. 1

After entering the number corresponding to the selected serial port, press <INVIO> (ENTER). Fig.2 shows the subsequent request.

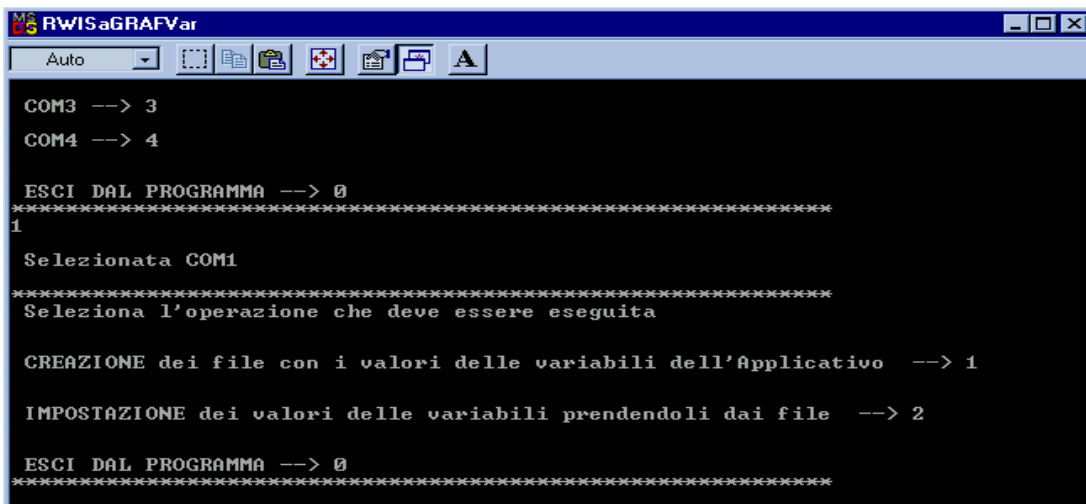
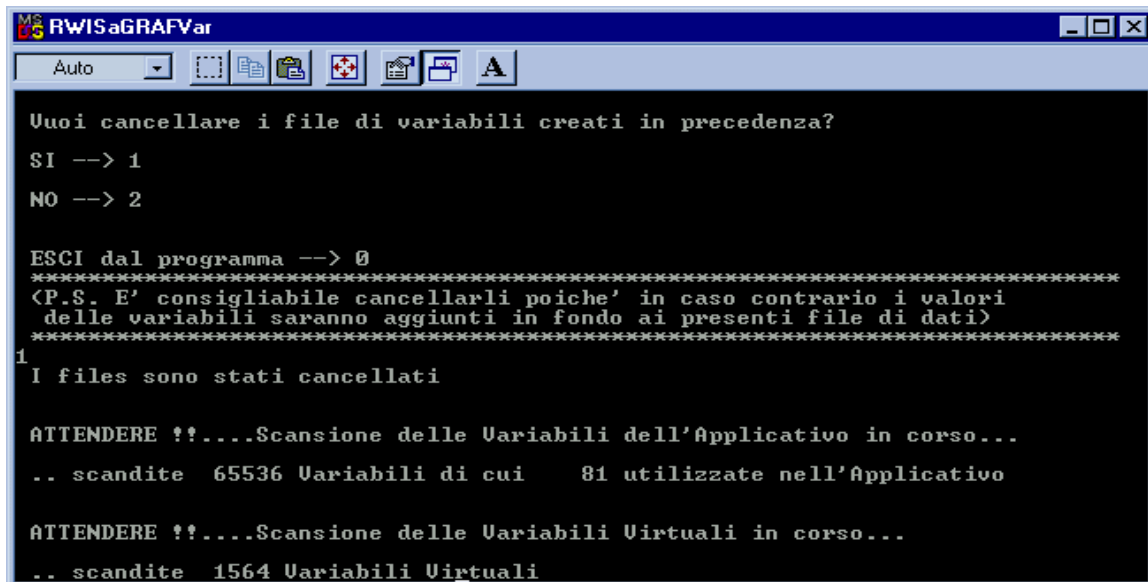


Fig. 2

Select 1 to create the files with the values of the variables. By pressing <INVIO> (ENTER), the utility asks whether to delete any files containing the values of the previously created variables. We recommend deleting them; otherwise, the new values will be queued. Enter 1 and press <INVIO> (ENTER). The program immediately starts downloading the variables from the control and creates four files called FileANA.dat, FileBOO.dat, FileTMR.dat and FileVIR.dat, for analogue, Boolean, timer and virtual values respectively.

While downloading the variables from the data card, the progress of the operation is displayed.



```

RWISaGRAFVar
Auto
Uuoi cancellare i file di variabili creati in precedenza?
SI --> 1
NO --> 2

ESCI dal programma --> 0
*****
<P.S. E' consigliabile cancellarli poiche' in caso contrario i valori
delle variabili saranno aggiunti in fondo ai presenti file di dati>
*****
1
I files sono stati cancellati

ATTENDERE !?...Scansione delle Variabili dell'Applicativo in corso...
.. scandite 65536 Variabili di cui 81 utilizzate nell'Applicativo

ATTENDERE !?...Scansione delle Variabili Virtuali in corso...
.. scandite 1564 Variabili Virtuali

```

Fig. 3

Every file appears as indicated below:

```

4094 0
4101 6
4102 1278
...
4210 0
4211 99
4212 16
4213 416
0 0

```

Where for every row, the number indicated on the left represents the Virtual Address of the variable, while the number on the right indicates its value.

The values of the variables can be modified before downloading them.

Files end with "0 0".

It is possible to download only the values of the variables, whose Virtual Address is declared, while all the virtual variables will be downloaded.

Once the files are generated, there will be a request whether to display them on the screen.

The use of the screen display option with a very high number of variables may cause problems. We recommend consulting them opening the files with a common text editor.

With reference to Figure 2, select 2 for transferring the value of the variables to the control.

The transfer of the value of the variables will start, indicating in real time the variables actually sent.



```

MS-RWISaGRAFVar
Auto
IMPOSTAZIONE dei valori delle variabili prendendoli dai file --> 2
ESCI DAL PROGRAMMA --> 0
*****
2
ATTENDERE !?...trasferimento delle Variabili dell'Applicativo in corso
... Trasferite 32 Variabili Booleane
Le variabili Booleane sono state impostate correttamente
... Trasferite 46 Variabili Analogiche
Le variabili Analogiche sono state impostate correttamente
... Trasferite 3 Variabili Timer
Le variabili Timer sono state impostate correttamente
ATTENDERE !?...trasferimento delle Variabili Virtuali in corso
...trasferite 1380 Variabili Virtuali

```

Fig. 4

### Indications for use.

This reading and writing utility must be used by connecting the SU210/250 data card on the serial port that supports the ISaGRAF ModBus protocol developed by Artec.

It is important to ensure that the electrical standard of the PC and data card serial ports is the same (not RS232 with RS485).

For any information, contact Artec spa.

## Annex C – I/O Library

The digital/analogue I/Os are managed through libraries provided by Artec SpA to be used to combine the I/O variables used in the application program at the physical inputs and outputs of the SU210 data card.

Below is a description of the libraries for managing Artec I/Os. However, for connecting variables with physical inputs/outputs, refer to the chapter **A.10 Using I/O connection editor** of the **ISaGRAF User Guide**.

### Digital I/Os

The SU210 system consists of a certain number of basic I/Os and a series of expansions. There are libraries available for managing both the basic I/Os and the expansions.

For every expansion card (or for basic I/Os) the library is indicated as ***I/O complex equipment***. The library for basic I/Os is called nS210\_IO and has 16 inputs and 16 outputs for a total 32 I/O. The expansion libraries are called S210\_X, followed by the number indicating its installation position compared to the basic card and by a letter indicating the type of expansion used. Expansion cards are divided into UP and DOWN, as indicated in fig.1.

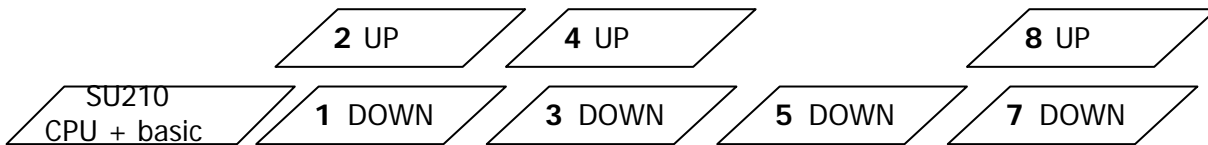


Fig.1

All expansions in line with the SU210 CPU card are called DOWN and have an odd number in the library, while expansions that are installed over other expansions are called UP and have an even number in the library (card number lower than + 1).

Letters indicate the selected type of card (regardless of the selected position in the expansion "chain"). Letter ***a*** indicates an expansion with 32 inputs, letter ***b*** indicates an expansion with 32 outputs, and letter ***c*** indicates an expansion with 16 inputs + 16 outputs.

There is no I/O complex equipment with even number and the letter ***c***, because the 16+16 card cannot be connected on top of another expansion but only in line with the SU210 data card.

This means that it is necessary to select the I/O complex equipment to be used (among all those provided by Artec SPA) according to the type of card to be used and its position in the connection "chain".

E.g., (see fig.2) the first expansion (e.g., 32 inputs) connected must use the S210\_X1a library. The second expansion (e.g., 32 outputs) will use the S210\_X2b library if it is connected on top of the previous one, while it will use the S210\_X3b library if it is "queued" to the present one.

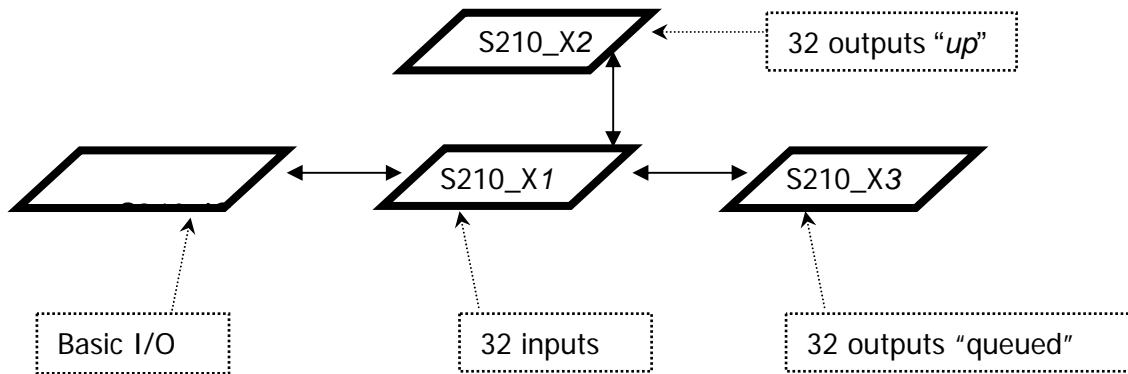


Fig.2

I/O complex equipment must be changed only if a card is replaced with one of a different type.

Every I/O (I/O complex equipment) library is divided into boards, containing each a certain number of inputs and/or outputs, see fig.3.

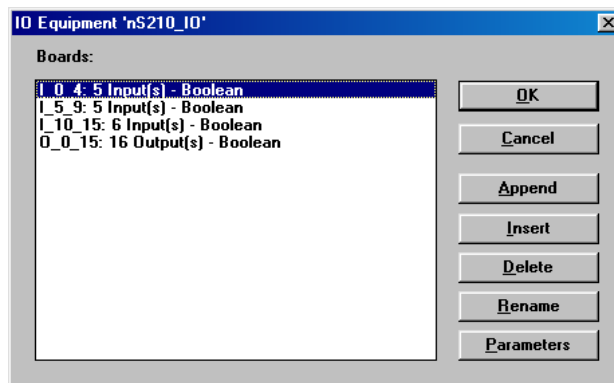


Fig.3

E.g., nS210\_IO contains 3 input boards for a total 16 inputs (5+5+6) and 1 output board with 16 outputs.

Expansions (S210\_X) are divided in three ways:

- S210\_X...a: 4 boards with 8 inputs for a total 32 inputs.
- S210\_X...b: 4 boards with 8 outputs for a total 32 outputs.
- S210\_X...c: 2 boards with 8 inputs for a total 16 inputs and 2 boards with 8 outputs for a total 16 outputs.

Such division does not depend on the expansion's position, but only on its hardware implementation.

Should the number of used expansions be higher than the number of drivers provided in the first place. Contact Artec for new drivers or for the documentation for creating them.

## Analogue I/Os

The su210 system is equipped with five inputs for reading the analogue channels that can be managed via the application program.

The library to be used for reading such channels is provided by Artec SpA and is called nS210Ana. This library is in the I/O complex equipment category and consists of a single board called *anal*.

There are five analogue channels that represent the voltage value in the dedicated input (see connection diagram) expressed in mvolt.

In particular:

- 1) User input on J2 pin 12.
- 2) User input on P4 pin 4.
- 3) Power voltage reading input.
- 4) Not available.
- 5) Not available.

## Power supply status

The su210 system allows reading information concerning the card power supply status from the application program by means of the nS210Ali library.

Information available includes:

- Prot\_Out : Output protection control. Allows detecting anomalies at the digital outputs, such as short circuit, open load (current below 50mA) and overtemperature of the output pilot device.
- Fuse\_Out : Indicates the activation of the user output protection fuses.
- Fuse\_Enc : Indicates the activation of the encoder power supply protection fuses.
- Batt\_NO : Allows knowing whether the battery is charged or not from the application. TRUE indicates that the battery charge is not sufficient to maintain the data in the memory. The battery needs to be replaced as soon as possible without turning the control off. See technical and maintenance manual.

- Liv\_Alim : Allows reading the power supply voltage from the application program. See technical and maintenance manual.

### Axis management data

The nS210Ass library provides information concerning the axis management.

Information available includes:

- Emerg : Emergency input. TRUE indicates that everything is ok, FALSE indicates an emergency status.
- FC\_Min : Minimum sensor input. FALSE indicates the reached minimum limit switch.
- FC\_Max : Maximum sensor input. FALSE indicates the reached maximum limit switch.
- FC\_Zero : Zero sensor input. TRUE indicates an active zero sensor.
- Mrk\_Zero : Zero notch input. TRUE indicates the activated and present zero notch.
- Hold : Hold input. Not used. Available for user program.
- IN\_SP : Special inputs used for capturing a position at interrupt.
- PosAtt : Reading of the current position expressed in mm. Replaces the ReadPos block.
- VelAtt : Reading of the current speed expressed in mm/sec. It replaces the ReadPos block.
- Vout : Output voltage expressed in millivolt.

All information is available in read-only mode. The lock function of any of such inputs does not affect the axis manager in any way.

The library for reading the sensors of axis expansion card EXP\_AX is called *expaxe* and provides the same information as the nS210Ass library, except for the special inputs IN\_SP and Hold.

The library for reading the sensors of axis expansion card EXP\_AX is called *expaxe* and provides the same information as the nS210Ass library, except for the special inputs IN\_SP and Hold.

## Annex D – Flash card errors

0	No Error	No error at start-up
1	Reading/writing error	Error occurred during low level operations. Error detected while accessing the flash card.
2	Error in the first sector	The first sector of the flash card contains incorrect data: the number of contained files does not correspond with the files that are actually present; the required file is not present.
3	Found firmware	There is a firmware file on the flash card and an operation that is not compatible with it has been requested.
4	Application found	There is an application on the flash card and an operation that is not compatible with it has been requested.
5	File with the same name	There is a file on the flash card with the indicated name and an operation that is not compatible with it has been requested.
6	Not enough space	There is not enough space on the flash card to save the file.
7	Data error	The data on the flash card are inconsistent: the space actually used by a file does not correspond to the value stored in the flash card.

## Annex E – Arteco compiler for C167

The application developed in ISaGRAF® environment can be specially compiled for the processor in the SU210, SU212, SU250, CND51 PLC. This allows reducing the PLC cycle time considerably, according to the chosen language and to the percentage of function blocks in the overall application.

To install the compiler, refer to Arteco SpA. Below is a description of how to use it, i.e., how to compile and upload the generated application.

### Stage 1 – Compilation option

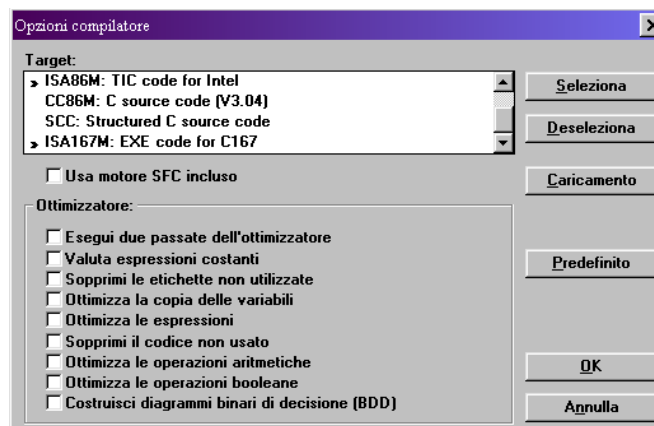


Fig. 1



In the selected project, choose "Compila" (Compile) and "Opzioni Compilatore" (Compiler Options). The window below will appear. To use the Arteco compiler for C167, select the highlighted item:

**ISA167M: EXE code for C167.**

Upon compilation, the "Compila Applicazione" (Compile Application) control will display the information below, to indicate the successful compilation by means of the compiler for C167.

**Verifica**

**Code generation : code for SABC167 : Release <0> Version <1> Revision <1>**

**Nessun errore rilevato**

Once the compilation is completed, the application can be downloaded in the target.

## Stage 2 – Download

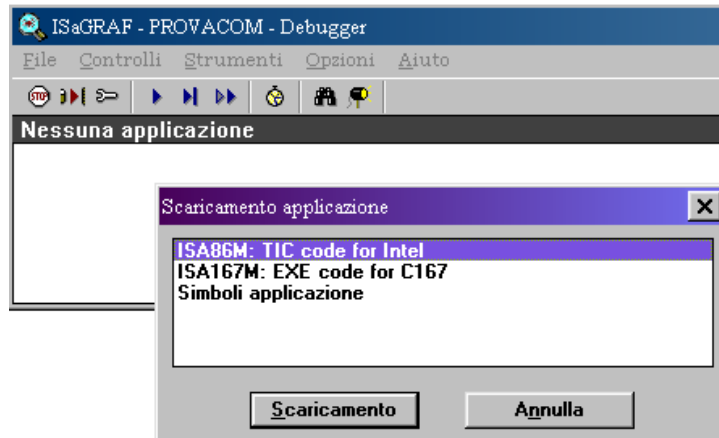


Fig. 2

After connecting to the target through “Debug”, proceed with the download. By pressing the appropriate button, a window will open with the following items “**ISA86M: TIC code for Intel**” and “**ISA167M: EXE code for C167**”. The first one sends the compiled application to the target in traditional format (interpreted code), while the second one sends the compiled application for SU data card to the target.

Through Workbench, all of the debug functions continue to be used even with the compiled application, except for the breakpoint in SFC environment, which is disabled.

The size of the compiled application program is significantly larger than the interpreted program (TIC Code); therefore the application download time through workbench turns out to be considerably longer.

The compiled application “EXE Code” can be saved on a SIM and restored as the “TIC Code” application. Restoration from SIM has an unvaried duration, regardless of the application and/or the type of executable file (TIC or EXE).

The “EXE Code for C167” format is only admitted on data cards that support it. The versions that allow this type of application are the 49th dated 12/03/2001 and the versions following the 59th included.

Upon downloading an EXE application through workbench, the target version is checked and if it does not support the format in question, the option will not be shown in the selection mask (see Fig.2).

Uploading an EXE application through SIM on a target that does not allow doing so may cause malfunctions of the SU data card.



**ARTECO MOTION TECH SPA.**

Via Gentili, 22 – 48018 Faenza (RA) Italy

Phone: +39 0546 645777 – Fax +39 0546 645750

[info@arteco-cnc.com](mailto:info@arteco-cnc.com) – [www.arteco-cnc.com](http://www.arteco-cnc.com)