Specification of an OPC Server for the CAEN SY127 HV supply

Philippe GRAS

Oct., 28 1998, rev. May, 12 1999

In this document we define the specification of the OPC Server for the CAEN SY127 HV supply we have proposed to design. OPC, OLE [1] for Process Control, is a standard interface for process control allowing standard communication (that is without need of custom drivers) between devices and software. The device and the software have to be OPC designed. Generally the device is delivered with an application that runs on a PC and provides the OPC interface. The device or the application delivered with is called OPC server and the software that we want to connect to the device, OPC client. We have proposed to write an application to provide an OPC interface for a not OPC-designed device. We will call this application the OPC server. After having given you an overview of our system, we will define the parameters of the power supply that will be accessible from outside, then we will give the OPC interfaces that will be implemented.

Revision of May, 12th 1999

I have added some notes to specify what is actually not supported by the OPC Server, we have now implemented (version 1.0). It is specially about Caen channel groups. Those groups are not supported in consequence of difficulties to deal with arrays with OPC. The difficulties are:

- The OPC clients we used for testing during development do not support arrays.
- The engineering units are common for all the elements of an array.

1 Architecture of our system

1.1 SY127 system physical architecture

The system where the OPC server will be used has to include at minimum:

- a PC with Windows NT to run the OPC server,
- an A303A PC CAENET Controller, which is a card to plug in an ISA slot of the PC to interface it to CAENET Bus, used by SY127 crates,
- the SY127 crates: theoretically up to 99,
- A128HS communication controllers to control the crates from the caenet bus: 1 by crate,
- Channel boards which provide each 4 channels: up to 10 boards on each crate.

The drawing of figure 1 shows the interconnections between these elements.
Now the physical architecture of our system is defined we will see how the software part is organised.

1.2 Software architecture

OPC is built on the COM and DCOM layer. COM is an acronym for Compound Object Model. COM is the standard of Microsoft allowing programs (dynamic libraries or executables), written in any language but according to this standard, to communicate. DCOM (Distributed COM) is roughly the extension of COM to remote systems, allowing to two programs on two different computers to communicate.

[1] Object Linking Embedding. This technology was developed by Microsoft.

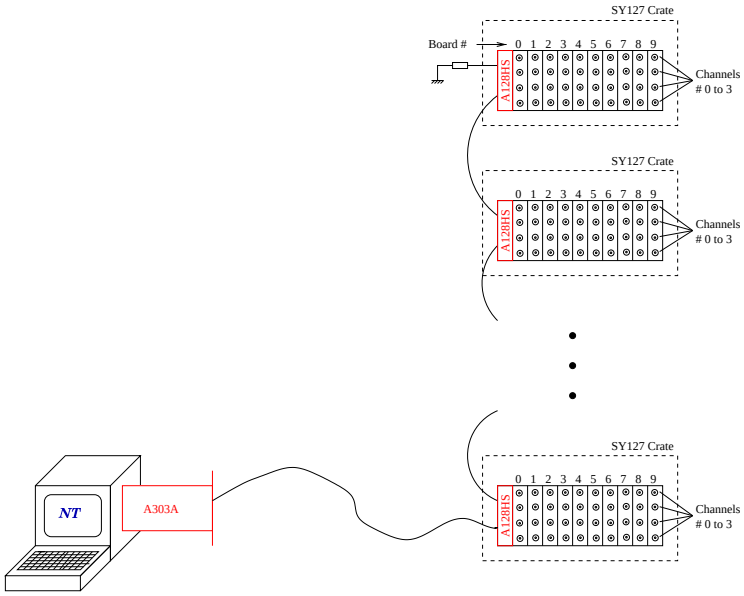


Figure 1: Architecture of a sample system using the OPC Server for SY127.

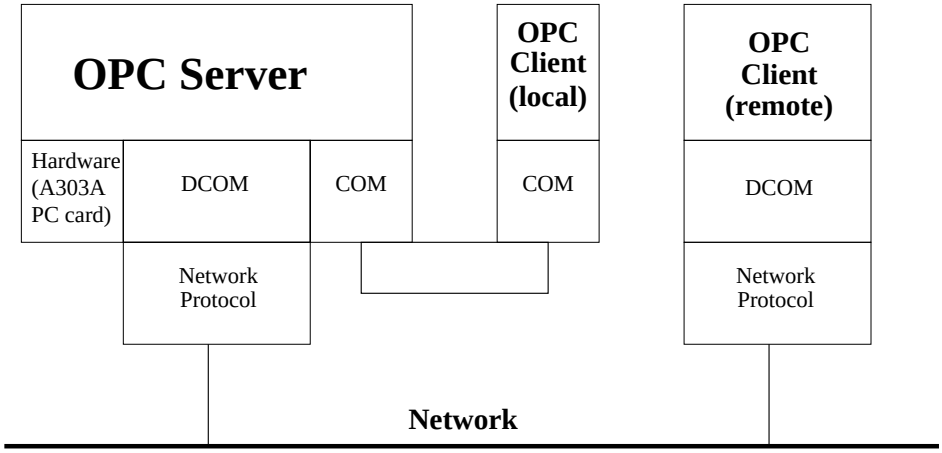


Figure 2: Architecture of the software.

The programs that are connected to an OPC server are called clients. OPC is multi-client and multi-server, that is several clients can be connected to a common server and one compound can be the client of several servers.

The figure 2 shows the layer architecture of the software.

2 OPC Items

The OPC server will offer to its clients to access to a certain number of parameters defining the state of the power supply. The parameters accessible by the clients are called OPC Items.

First, let's see the parameters of the SY127 that the manufacturer let us to access. Then we will define the OPC server parameters accessible by the OPC clients, i.e. the OPC items.

2.1 CAEN SY127 parameters

The parameters of the CAEN SY127 are listed in the table 1. The SY127 crate allows us to define groups of channels to access in only one command to the value of a parameter of each channel of a channel set.A

write access will give the same value for all the channels belonging to the group, a read access will return the value of each channel of the channel group. 7 channel groups can be defined. There is one more channel group which is a set of all the channels. This channel group is called "group all". Write access to a channel group can be done in relative value, that is value will be add the original value of the concerned parameter for each channel of the group.

According to the list of parameters of table 1 we will define the OPC items our server will have.

2.2 Notion of OPC Item and of OPC group

The OPC server will propose to the client a certain number of parameters that will define the state of the device. As the parameter may be the client can read it, write it or both. These parameters are called OPC items.

Among all the OPC items the OPC server can provide the client can choose the ones it wants to access. In order to do it, the client will define a set of item, called OPC group, by giving a name for the set and the list of the items composing it. The client can define as many OPC groups it wants. The client can add or remove any item of a group it has previously defined at any time. A single item can belong to several groups defined by one client.

N.B.: we shall not confuse an OPC group with the channel groups described in part 2.1.

We will define the OPC parameters of the OPC server according to the CAEN SY127 parameters previously browsed. We will distinguish three types of items:

- channel dependent items: for one type of parameter, e.g. V0, we will need as many items as we've got channels,
- channel group dependent items,
- crate dependent items.

For the group dependent items we have define for one readable and writable physical parameter three OPC items:

- one for read access,
- one for write access in absolute value mode,
- one for write access in relative value mode.

This is because, when the OPC client will ask to the OPC server to read the value the OPC server will return an array containing **the value of each channel** composing the channel group and when the client ask to write a value it will send to the server a **value common for all the channels composing the channel group**[3].

These items are listed in table 2.2. The data of an OPC Item are of VARIANT type. VARIANT type is part of OLE Automation technology. A variant is a structure that can contain different type of data. The type of the data is identified in a field of the VARIANT structure. The "Variant type" given in the table 2.2 is the type of the data contained in the VARIANT structure. The table 2 gives the description of these types.

In the VT_ARRAY's the values will be ordered in increasing channel number. The item cr*xx*.gr*zz*.Get.Ch allows to get the channel corresponding to each array index value.

3 OPC Interfaces supported

An OPC server is composed of 3 COM object:

- OPCItem object.
- OPCGroup object. This object allows the clients to organise the items it needs.
- OPCServer object. This object allows the client to manage OPCGroup objects.

[2]The switching between value 0 and value 1 is performed by two external NIM input levels (VSEL,ISEL)

[3]Since the clients used to test the OPC Server do not support arrays as item values, the current version (1.0) of the OPC Server does not support group dependent items.

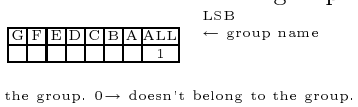
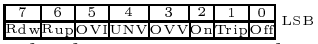
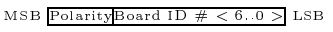
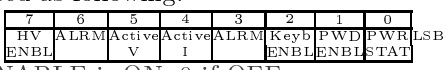
Parameter Name	Size (bytes)	**R**eadable/**W**riteable	Number 1 by **Ch**annel or **Cr**ate	Description
V0	2	RW	1/Ch	Channel voltage value number 0. [2]
V1	2	RW	1/Ch	Channel voltage value number 1. [2]
I0	2	RW	1/Ch	Channel intensity value number 0. [2]
I1	2	RW	1/Ch	Channel intensisty value number 1. [2]
Rup	2	RW	1/Ch	Ramp up speed value.
Rdwn	2	RW	1/Ch	Ramp down speed value.
TripVal	2	RW	1/Ch	The trip delay value.
ChName	10	RW	1/Ch	The name of the channel (up to 9 char, terminated by the null character.)
Ch2GrpAss	1	RW	1/Ch	The assignement of the channel to the CAEN groups. 
Status	1		1/Ch	One byte composed as following: 
Rdw	1/8	R	1/Ch	set to 1 when the channel voltage is ramping down.
Rup	1/8	R	1/Ch	set to 1 when the channel voltage is ramping up.
OVI	1/8	R	1/Ch	set to 1 when the channel is in over-current state.
UNV	1/8	R	1/Ch	set to 1 when the channel is in under-voltage state.
OVV	1/8	R	1/Ch	set to 1 when the channel is in over-voltage state.
On	1/8	RW	1/Ch	set to 1 when the channel is ON. Writable through an ON/OFF command which affects too the OFF bit.
Trip	1/8	RW	1/Ch	set to 1 when the channel has tripped.
Off	1/8	RW	1/Ch	set to 1 when the channel is OFF. Writable through an ON/OFF command which affects too the ON bit.
Vmon	2	R	1/Ch	The monitored value of the channel voltage.
Imon	2	R	1/Ch	The monitored value of the channel current.
BoardID	1	R	1/Ch	One byte composed as following:  The BoardID identify the type of 4-channel board.
Protect	1		1/Cr	One byte composed as following: 
HV ENBL	1/8	R	1/Cr	Set to 1 if HV-ENABLE is ON, 0 if OFF.
ALRM	1/8	R	1/Cr	Set to 1 when an alarm for OVV, UNV, TRIP occured in any channel. Set to 0 by a clear alarm command. Bit 3 and 7 of the Protect byte have the same meaning
Active V	1/8	R	1/Cr	Set to 0 if active voltage value is V0, set to 1 if it is V1.
Active I	1/8	R	1/Cr	Set to 0 if active current value is I0, set to 1 if it is I1.
Keyb ENBL	1/8	RW	1/Cr	Set to 1 if Keyboard is enable, to 0 if disable.
PWD	1/8	RW	1/Cr	Set to 1 if password is enable, to 0 if disable.
PWR STAT	1/8	RW	1/Cr	If 1 and if the channel were ON before a power-off it will be automatically switched ON at the next power-on.
MainfID	44	R	1/Cr	Mainframe identifier packet. This contains the string "SY127 Vx.x (Main Vy.y)" for software version 6.6 or higher.

Table 1: Parameters of the SY127 HV supply system.

Variant type	Description
VT_R4	float coded in 4 bytes
VT_UI1	unsigned char (1byte)
VT_BOOL	boolean
VT_EMPTY	no data
VT_ARRAY of Y	secured array of Y type elements
VT_BSTR	String (Unicode).

Table 2: Description of the variant types.

The functions that a COM object provides to its clients are regrouped by functionalities in what is called an COM interfaces, in this case OPC interfaces. The OPC specification define the interfaces than the OPC objects have to provide. Some are optional. In this part we give the list of the OPC interfaces which will be implemented in the OPC server for SY127. For the functions that composed the OPC interfaces, see OPC Data Access Specification 1.0A pp 16ff.

3.1 Custom interfaces

The custom interfaces are made for the client programs implemented with a compiled language.

OPCServer object

- IOPCServer. This is the main interface of the OPCServer object.
- IOPCBrowseServerAdressSpace. This interface allows a client to ask for the list of available items. We want to implement this interface which is optional according to OPC specifications v1.0A.
- [IOPCServerPublicGroups]. This interface allows to several clients to use a common group which is called a public group. We may be interested by this optional interface in a next version.

OPCGroup object

- IOPCGroupStateMgt. This interface is used to clone groups and to set the name of a group.
- IOPCSyncIO. This interface allows a client to perform synchronous input/output.
- IOPCAsyncIO. This interface allows a client to perform asynchronous input/output. Although this interface is not optional according to OPC specification 1.0A, it will be implemented only in a next version.
- IOPCItemMgt. This interface allows a client to add, remove and control the behaviour of the items of an OPC group.
- IDataObject. This interface allows the creation of an Advice connection between a client and an OPC group. See the OLE programming manual for more details. Although this interface is not optional according to OPC specification 1.0A, it will be implemented only in versions containing the IOPCSyncIO interface.
- [IOPCPublicGroupStateMgt]. This interface allows to convert a private group to a public group. It will be implemented in the possible version containing IOPCServerPublicGroupsDisp interface.

EnumOPCItemAttributes object

- IEnumOPCItemAttributes. This interface allows a client to know the contents of a group and the attribute of each item.

	Item ID *xx* : crate number (2 decimal digits) *yy* : channel number (2 decimal digits) *zz* : ch. group number (2 decimal digits)	Unit	Variant type	**R**eadable/**W**riteable	Description
Channel dependent items	cr*xx*.ch*yy*.V0	V	VT_R4	RW	Value number 0 of the channel voltage setting.
	cr*xx*.ch*yy*.V1	V	VT_R4	RW	Value number 1 of the channel voltage setting.
	cr*xx*.ch*yy*.I0	mA	VT_R4	RW	Value number 0 of the channel maximum current setting.
	cr*xx*.ch*yy*.I1	mA	VT_R4	RW	Value number 1 of the channel maximum current setting.
	cr*xx*.ch*yy*.Vmon	V	VT_R4	RW	Monitored value of the channel voltage.
	cr*xx*.ch*yy*.Imon	μA	VT_R4	RW	Monitored value of the channel current.
	cr*xx*.ch*yy*.RupS	$V \cdot s^{-1}$	VT_R4	RW	Value of the ramp-up speed setting.
	cr*xx*.ch*yy*.RdwnS	$V \cdot s^{-1}$	VT_R4	RW	Value of the ramp-down speed setting.
	cr*xx*.ch*yy*.TripD	s	VT_R4	RW	Value of trip delay.
	cr*xx*.ch*yy*.Status		VT_UI1	R	Status byte of the channel. (See Status in the table 1).
	cr*xx*.ch*yy*.Rdwn		VT_BOOL	R	bit 7 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.Rup		VT_BOOL	R	bit 6 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.OVI		VT_BOOL	R	bit 5 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.UNV		VT_BOOL	R	bit 4 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.OVV		VT_BOOL	R	bit 3 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.On		VT_BOOL	**RW**	bit 2 and inverse of bit 0 of the Status byte. (see table 1)
	cr*xx*.ch*yy*.Trip		VT_BOOL	R	bit 1 of the Status byte. (see table 1).
	cr*xx*.ch*yy*.BoardID		VT_UI1	R	Identification byte of the board. See details in the table 1).
	cr*xx*.ch*yy*.Groups		VT_UI1	RW	Channel to group assignment byte. See Ch2GrpAss in the table 1).
	cr*xx*.ch*yy*.Name		VT_BSTR	RW	Name of the channel.
Group dependent items (not yet supported) (be con't)	cr*xx*.gr*zz*.Get.V0	V	VT_ARRAY of VT_R4	R	Values number 0 of the channel voltage settings.
	cr*xx*.gr*zz*.Get.V1	V	VT_ARRAY of VT_R4	R	Values number 1 of the channel voltage settings.
	cr*xx*.gr*zz*.Get.I0	mA	VT_ARRAY of VT_R4	R	Values number 0 of the channel maximum current settings.
	cr*xx*.gr*zz*.Get.I1	mA	VT_ARRAY of VT_R4	R	Values number 1 of the channel maximum current settings.
	cr*xx*.gr*zz*.Get.Vmon	V	VT_ARRAY of VT_R4	R	Monitored values of the channel voltages.
	cr*xx*.gr*zz*.Get.Imon	μA	VT_ARRAY of VT_R4	R	Monitored values of the channel currents.
	cr*xx*.gr*zz*.Get.RupS	$V \cdot s^{-1}$	VT_ARRAY of VT_R4	R	Values of the ramp-up speed settings.
	cr*xx*.gr*zz*.Get.RdwnS	$V \cdot s^{-1}$	VT_ARRAY of VT_R4	R	Values of the ramp-down speed settings.
	cr*xx*.gr*zz*.Get.TripD	s	VT_R4	R	Value of trip delay.
	cr*xx*.gr*zz*.Get.Status		VT_ARRAY of VT_UI1	R	Status bytes of the channel. See details in the table 1.
	cr*xx*.gr*zz*.Get.Rdwn		VT_ARRAY of VT_BOOL	R	bit 7 of the Status byte. (see table 1)

Table 3: OPC items of the server. (1/3)

	Item ID *xx* : crate number (2 decimal digits) *yy* : channel number (2 decimal digits) *zz* : ch. group number (2 decimal digits)	Unit	Variant type	**R**eadable/**W**riteable	Description
Group dependent items (not yet supported) (con't)	cr*xx*.gr*zz*.Get.Rup		VT_ARRAY of VT_BOOL	R	bit 6 of the Status byte. (see table 1)
	cr*xx*.gr*zz*.Get.OVI		VT_ARRAY of VT_BOOL	R	bit 5 of the Status byte. (see table 1
	cr*xx*.gr*zz*.Get.OVV		VT_ARRAY of VT_BOOL	R	bit 4 of the Status byte. (see table 1
	cr*xx*.gr*zz*.Get.On		VT_ARRAY of VT_BOOL	R	bit 3 and inverse of bit 0 of the Status byte. (see table 1
	cr*xx*.gr*zz*.Get.Trip		VT_ARRAY of VT_BOOL	R	bit 1 and inverse of bit 0 of the Status byte. (see table 1
	cr*xx*.gr*zz*.Get.BoardID		VT_ARRAY of VT_UI1	R	Identification byte of the board. See details in the table 1.
	cr*xx*.gr*zz*.Get.Groups		VT_ARRAY of VT_UI1	R	Channel to group assignment byte. See details in the table 1.
	cr*xx*.gr*zz*.Get.Ch		VT_ARRAY of VT_UI1	R	Numbers of the channels composing the group.
	cr*xx*.gr*zz*.aSet.V0	V	VT_R4	W	Set the channel voltages V0 to a common value.
	cr*xx*.gr*zz*.aSet.V1	V	VT_R4	W	Set the channel voltages V1 to a common value.
	cr*xx*.gr*zz*.aSet.I0	mA	VT_R4	W	Set the channel maximum currents I0 to a common value.
	cr*xx*.gr*zz*.aSet.I1	mA	VT_R4	W	Set the channel maximum currents I1 to a common value.
	cr*xx*.gr*zz*.aSet.RupS	$V \cdot s^{-1}$	VT_R4	W	Set the channel ramp-up speeds to a common value.
	cr*xx*.gr*zz*.aSet.RdwnS	$V \cdot s^{-1}$	VT_R4	W	Set the channel ramp-down speeds to a common value.
	cr*xx*.gr*zz*.aSet.TripD	s	VT_R4	W	Set the channel trip delays to a common value.
	cr*xx*.gr*zz*.aSet.On		VT_BOOL	W	Set to 1 bit 3 and to 0 bit 0 of the status byte of the channels. (see table 1)
	cr*xx*.gr*zz*.rSet.V0	V	VT_R4	W	Add a value to the channel voltage V0 values.
	cr*xx*.gr*zz*.rSet.V1	V	VT_R4	W	Add a value to the channel voltage V1 values.
	cr*xx*.gr*zz*.rSet.I0	mA	VT_R4	W	Add a value to the channel maximum current I0 values.
	cr*xx*.gr*zz*.rSet.I1	mA	VT_R4	W	Add a value to the channel maximum current I1 values.
	cr*xx*.gr*zz*.rSet.RupS	$V \cdot s^{-1}$	VT_R4	W	Add a value to the channel ramp-up speed values.
	cr*xx*.gr*zz*.rSet.RdwnS	$V \cdot s^{-1}$	VT_R4	W	Add a value to the channel ramp-down speed values.
	cr*xx*.gr*zz*.rSet.TripD	s	VT_R4	W	Add a value to the channel trip delay values.

Table 4: OPC items of the server.(2/3)

	Item ID *xx* : crate number (2 decimal digits) *yy* : channel number (2 decimal digits) *zz* : ch. group number (2 decimal digits)	Unit	Variant type	Readable/Writeable	Description
Crate dependent items	cr*xx*.Protect		VT_UI1	RW	Protection Byte of the crate. See details in the table 1. Only the RW bits (see below) are affected by a write access.
	cr*xx*.HVENBL		VT_BOOL	R	bit 7 of the crate protection byte.
	cr*xx*.ALRM		VT_BOOL	R	bit 6 and bit 3 of the crate protection byte.
	cr*xx*.ActiveV		VT_BOOL	R	bit 5 of the crate protection byte.
	cr*xx*.ActiveI		VT_BOOL	R	bit 4 of the crate protection byte.
	cr*xx*.KbdENBL		VT_BOOL	RW	bit 2 of the crate protection byte.
	cr*xx*.PWD		VT_BOOL	RW	bit 1 of the crate protection byte.
	cr*xx*.PWRStat		VT_BOOL	RW	bit 0 of the crate protection byte.
	cr*xx*.MFrameID		VT_BSTR	R	Mainframe identifier of the crate. See details in table 1.
	cr*xx*.BoardID		VT_ARRAY of 10 VT_UI1	R	Identifiers of the 4-channel boards of the crate. See details in table 1.
	cr*xx*.ClrAlrm		VT_EMPTY	W	Write access to this item clear the alarms.
	cr*xx*.FormatEEPROM		VT_EMPTY	W	Write access to this item then to the cr*xx*.ConfFormatEEPROM item clear the EEPROM of the crate which stores all the crate parameters during power off.
	cr*xx*.ConfFormatEEPROM		VT_EMPTY	W	see cr*xx*.FormatEEPROM above.

Table 5: OPC items of the server.(3/3)

3.2 Automation interfaces

Those interfaces are made to be called by a script language command. Therefore it permits, for example, to connect to the OPC server from an Excel sheet. Although those interfaces are not optional according to the v1.0A OPC specification, the automation interface is not implemented in the National Instrument IAS toolkit.

We should be able to get from the OPC foundation a dll that wraps the custom interface to provide an automation interface.

OPCServer object

- IOPCServerDisp. Equivalent to IOPCServer interface but for automation.
- IOPCBrowseServerAdressSpaceDisp. Equivalent to IOPCServerAdressSpace but for automation.
- [IOPCServerPublicGroupsDisp]. Equivalent to IOPCServerPublicGroups but for automation. Will be implemented only if the IOPCServerPublicGroups will be implemented.

OPCGroup object

- IOPCItemMgtDisp. Automation equivalent to IOPCItemMgt interface.
- IOPCGroupStateMgtDisp. Automation equivalent to IOPCGroupStateMgt interface.
- IOPCSyncIODisp. Automation equivalent to IOPCSyncIO interface.
- IOPCAsyncIODisp. Automation equivalent to IOPCAsyncIODisp interface. This won't be implemented in versions not including the IOPCAsyncIODisp interface.
- [IOPCPublicGroupStateMgtDisp]. Automation equivalent to IOPCPublicGroupStateMgt.This won't be implemented in versions not including the IOPCServerPublicGroups interface.

OPCItem object

- IOPCItemDisp. This interface provides properties and methods on the Item object.

This OPC server for SY127 will be implemented in C++ on Windows NT. But the DCOM implementation of Software AG American for Unix platform may allow to implemented this OPC Server for Unix.