Parameter & Persistence Library

The ControlSphere Parameter and Persistence Service Library (CPP) provides an optimized means to store and retrieve variable values to and from well-formatted, human-readable data files. This library provides numerous advantages over the Retain/Persist, Recipes, and Parameters features built-into the CODESYS IDE, particularly in object-based design. Once a Function Block implements the CPP Library, the parameter, persistent, and recipe data for each instance are automatically managed – there are no lists to be created or maintained. The files(s) created by the CPP Library may be edited in a spreadsheet editor and/or moved to other systems to duplicate the persistent, parameter, or recipe values on those systems. This library includes a demo project.

Produktbeschreibung

Features, advantages, and differentiators of the CPP Library

- The CPP Library stores data that will survive a power-cycle, reset, new download, or equipment change; or can be used in a production run. The CPP Library is often used for Parameters, Persistent variables, and Recipes.
- CPP-managed variables are automatically registered when a Function Block is instantiated. There are no persistence or recipe lists to create or maintain. Just add an instance of a Function Block and its CPP variables are automatically managed in one or more CSV files in the runtime file system. If more instances of Function Blocks are added to the design, their variables are automatically managed by the CPP Library. No further user effort is required.
- The CPP Library will create the initial CSV file containing the default values for all the CPP variables, in all the instances of Function Blocks that implement the CPP Library. This saves the user from the laborious and error-prone task of hand-typing this file.
- Since the CPP stores its variable values in well-organized human-readable ASCII files rather than binary files, the CPP files can be easily modified if persistence, parameter, or recipe variables values are added or removed from Function Blocks, or instances of Function Blocks are added or removed. Being able to modify the files allows the CPP persistence values to be moved to systems which are not identical to the originating system. Of course, at any time the user can have the CPP write a new CSV file which will include all the variables or instances, as described in the previous bullet. The CPP Library provides the maximum possible flexibility so your data is easily reused and is never lost.
- Parameters are set for all configurable objects from a consolidated source file which is easily managed in any spreadsheet software. There is no need to manually traverse the device tree searching for and opening each instance individually to set its parameters. Plus

the parameters can be set either offline or online. In addition, the CPP Library does not have hierarchy restrictions like the built-in parameter tool – it works with nested Function Blocks to any size and shape of hierarchy.

- Product variations can be accommodated by placing different sets of parameter values in different parameter files. The proper file can be selected on startup based on the identification of the product. This same feature can be used to save and restore different recipes in a file format that is much more convenient than the built-in recipe tool.
- Parameter values are stored in a human readable and editable file, not buried and inaccessible in the project file. These files may be stored in the project device tree if desired so its variable values are automatically downloaded to the target on login. Or, the files can also be copied directly to the target so the CODESYS IDE is not required in a mass production environment.
- The CPP Library does not require copying values to and from a global variable list on each process cycle. So it is much more CPU efficient than the built-in persistence feature.
- The CPP can initiate a file read or write from the same Task where the file action is executed (the variables will thenall be atomic), or the initiation can be in a different task from the execution (the file I/O will not pause the Task).
- Variables can all be managed in the same CSV file, or sets of variables can be partitioned to be managed in different CSV files (useful to differentiate between one-time system configuration variables, operating mode variables set by the operator via a visualization, and persistent process variables).
- The CPP can save all the variables in a Group for all the instances of Function Blocks, or it can save the variable values for one instance at a time. This is useful for saving the tuning parameters on an instance by instance basis.

Object-based programming has become the standard in worldwide IT software programming and is rapidly being adopted into PLC programming. To see the benefits of object-based programming and examples of how this looks in the world of industrial controls programming, please view these videos and articles:

https://www.youtube.com/watch?v=PkJYQeIUmIM

https://www.youtube.com/watch?v=vRGaW4L762k

https://www.controleng.com/articles/leverage-object-oriented-industrial-programming/

The object of object-based programming is for objects (Function Blocks) to be fully self-contained and self-reliant. When an instance of an object is declared, that instance should then automatically handle all the services it needs. The traditional programing technique of creating and maintaining separate persistent variable lists, alarm lists, log lists, and parameter lists does not scale well to today's large and complex PLC programs.

The native CODESYS IDE lacks support for Object Oriented Programming in several areas:

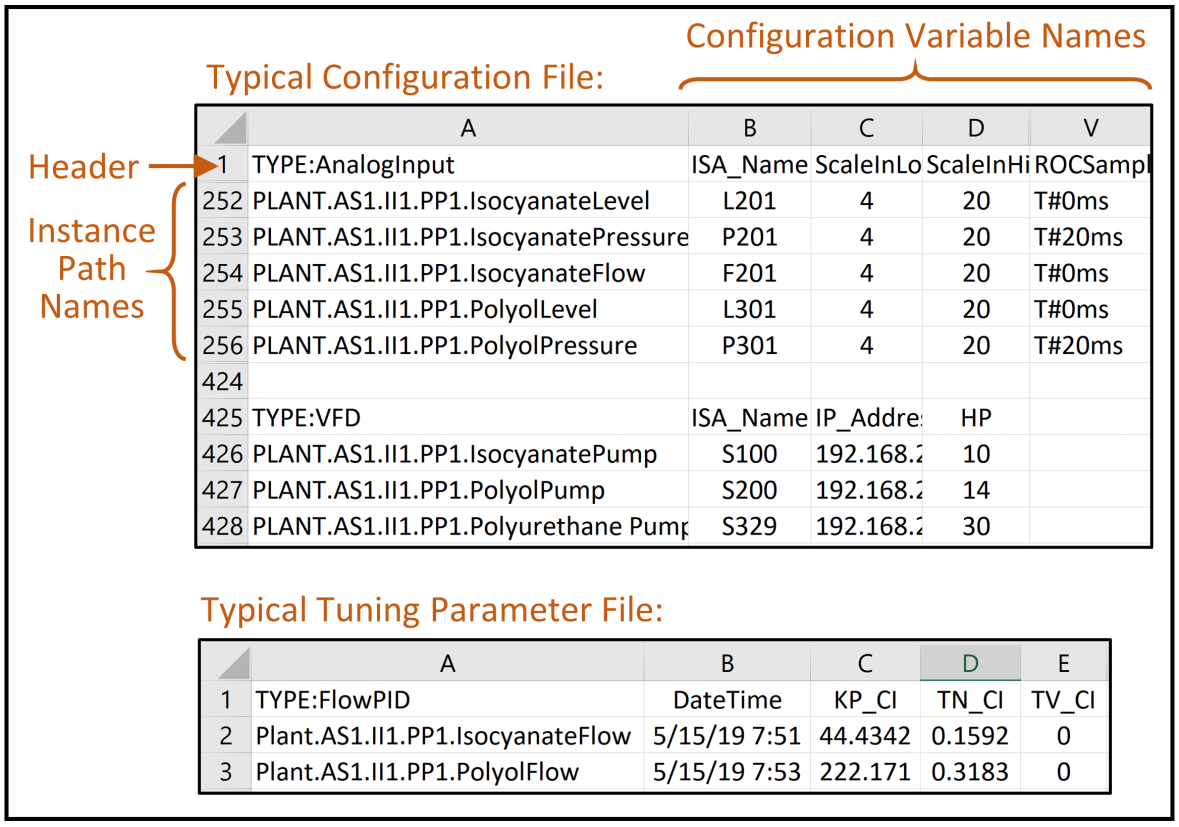
1. Alarms

2. Logging/Trending
3. Object Oriented I/O
4. Persistent Variables
5. Parameters and Configurations
6. Recipes

The ControlSphere OOA library addresses issue number 1 and this ControlSphere CPP Library addresses issues 4, 5 and 6. A prototype is available for issue number 3 and a solution for issue number 2 is in the plans. Contact ControlSphere Engineering for more details.

The CPP Library is available in a low-cost, unlimited, node-locked version (and with significant quantity discounts). A single-site unlimited source-code license is also available which is not node locked. Contact sales@controlsphere.pro for more details.

The following graphics describe the format of the CPP files, shows a typical CPP file, and shows the implementation of the base class, interfaces, and methods necessary for a Function Block to gain support of the CPP Library.



	A	B	C	D	V
1	TYPE:AnalogInput	ISA_Name	ScaleInLo	ScaleInHi	ROCSampl
252	PLANT.AS1.II1.PP1.IsocyanateLevel	L201	4	20	T#0ms
253	PLANT.AS1.II1.PP1.IsocyanatePressure	P201	4	20	T#20ms
254	PLANT.AS1.II1.PP1.IsocyanateFlow	F201	4	20	T#0ms
255	PLANT.AS1.II1.PP1.PolyolLevel	L301	4	20	T#0ms
256	PLANT.AS1.II1.PP1.PolyolPressure	P301	4	20	T#20ms
424					
425	TYPE:VFD	ISA_Name	IP_Addre:	HP	
426	PLANT.AS1.II1.PP1.IsocyanatePump	S100	192.168.2	10	
427	PLANT.AS1.II1.PP1.PolyolPump	S200	192.168.2	14	
428	PLANT.AS1.II1.PP1.Polyurethane Pump	S329	192.168.2	30	

	A	B	C	D	E
1	TYPE:FlowPID	DateTime	KP_CI	TN_CI	TV_CI
2	Plant.AS1.II1.PP1.IsocyanateFlow	5/15/19 7:51	44.4342	0.1592	0
3	Plant.AS1.II1.PP1.PolyolFlow	5/15/19 7:53	222.171	0.3183	0

Figure 1: CPP CSV File Format

The next graphics shows a CPP file for an actual design. The CPP Library initially creates this file with all the default variable values for every instance of every Function Block that implements the CPP interface. The user can then edit the values that need to be different than the initial values, and read the file back into the system to update the runtime values. Or, these values can be set during runtime (such as from a Visualization or a persistence program variable) and written again into storage that will survive a power-cycle, new download, or equipment change.

	A	B	C	D	E	F	G	H	I	J
1		DT#2020-03-08-11:21:28								
2	TYPE:CALCLOADSFB	ISA_Name_CI	BoreInch	RodInch	GasRatio	InitiGas	MxOpPres	CalPos		
3	MAIN.CYLINDERS.GENOASTAYSAIL.CALCLOADS	Genoa Stay Sail	1.75	0.5	10	5	500	250		
4	MAIN.CYLINDERS.HEADSTAY.CALCLOADS	Head Stay	3.125	0.875	10	1.00E-06	500	125		
5	MAIN.CYLINDERS.JIBCUNNINGHAM.CALCLOADS	Jib Cunningham	2.188	0.625	10	5	500	200		
6	MAIN.CYLINDERS.JIBINOUT.CALCLOADS	Jib In Out	1.75	0.5	10	5	500	200		
7	MAIN.CYLINDERS.JIBUPDOWN.CALCLOADS	Jib Up Down	1.75	0.5	10	5	500	250		
8	MAIN.CYLINDERS.LOWERDEFLECTOR.CALCLOADS	Lower Deflector	1.75	0.5	10	5	500	275		
9	MAIN.CYLINDERS.MAINCUNNINGHAM.CALCLOADS	Main Cunningham	1.125	0.375	10	5	500	200		
10	MAIN.CYLINDERS.OUTHAUL.CALCLOADS	Out Haul	1.5	0.438	10	5	500	125		
11	MAIN.CYLINDERS.SPARE.CALCLOADS	Spare	1.75	0.5	10	5	500	250		
12	MAIN.CYLINDERS.UPPERDEFLECTOR.CALCLOADS	Upper Deflector	2.75	0.813	10	5	500	275		
13	MAIN.CYLINDERS.VANG.CALCLOADS	Vang	2.375	1.25	10	5	500	232.5		
14										
15	TYPE:SADECylinderSystemFB	ISA_Name_CI	Disable	DblClick	DCTime	Class	ExtendPos	RetractPos	Stroke	
16	MAIN.CYLINDERS.GENOASTAYSAIL	Genoa Stay Sail	FALSE	1	T#400ms	2	0	1000	500	
17	MAIN.CYLINDERS.HEADSTAY	Head Stay	FALSE	1	T#400ms	2	396.24	621.0302	250	
18	MAIN.CYLINDERS.JIBCUNNINGHAM	Jib Cunningham	FALSE	1	T#400ms	1	220.98	595.122	400	
19	MAIN.CYLINDERS.JIBINOUT	Jib In Out	FALSE	1	T#400ms	2	0	1000	400	
20	MAIN.CYLINDERS.JIBUPDOWN	Jib Up Down	FALSE	1	T#400ms	4	0	1000	500	
21	MAIN.CYLINDERS.LOWERDEFLECTOR	Lower Deflector	FALSE	1	T#400ms	2	196.596	710.184	550	
22	MAIN.CYLINDERS.MAINCUNNINGHAM	Main Cunningham	FALSE	1	T#400ms	3	0	1000	400	
23	MAIN.CYLINDERS.OUTHAUL	Out Haul	FALSE	1	T#400ms	1	0	1000	250	
24	MAIN.CYLINDERS.SPARE	Spare	FALSE	1	T#400ms	1	0	1000	500	
25	MAIN.CYLINDERS.UPPERDEFLECTOR	Upper Deflector	FALSE	1	T#400ms	2	164.592	645.4152	550	
26	MAIN.CYLINDERS.VANG	Vang	FALSE	1	T#400ms	2	0	1000	465	
27										
28	TYPE:TravelerSystemFB	ISA_Name_CI	Disable	MinFlowI	PortRaceM	PortCruis	StbdRaceM	StbdCruiseM	MinPressu	RaceMaxI
29	MAIN.CYLINDERS.TRAVELER	Traveler	FALSE	350	700	450	700	450	100	60
30										
31	TYPE:WinchSystemFB	ISA_Name_CI	Disable	Speed1 Mi	Speed1 Ma	Speed1 C	Speed1 Mi	Speed1 Race	Speed1 Cru	Speed2 M
32	MAIN.WINCHES.MAINSHEETWINCH	Main Sheet Winch	FALSE	350	700	550	100	600	600	35
33	MAIN.WINCHES.PORTPITWINCH	Port Pit Winch	FALSE	350	700	550	100	400	400	35
34	MAIN.WINCHES.PORTPRIMARYWINCH	Port Primary Winch	FALSE	350	700	550	100	650	600	35

Figure 2: Typical CPP file, courtesy Marine Hydraulics

For a Function Block to gain access to the CPP Library services, the Function Block must EXTENDS the CCP EquipmentBaseClass and IMPLEMENTS the CPP PersistInt Interface. The PersistInt Methods must then be implemented to manage the values and names of the parameter, persistence, and recipe variables.

The next graphic demonstrates the coding for a single Persistence group. However, the CPP Library also supports multiple groups so that different sets of variables can be managed in different CSV files. This is typically useful to separate one-time configuration variables (which are set by the system programmer) and persistent variables (which are often set by the operator through a visualization screen), and process variables (which must survive a download or equipment change).

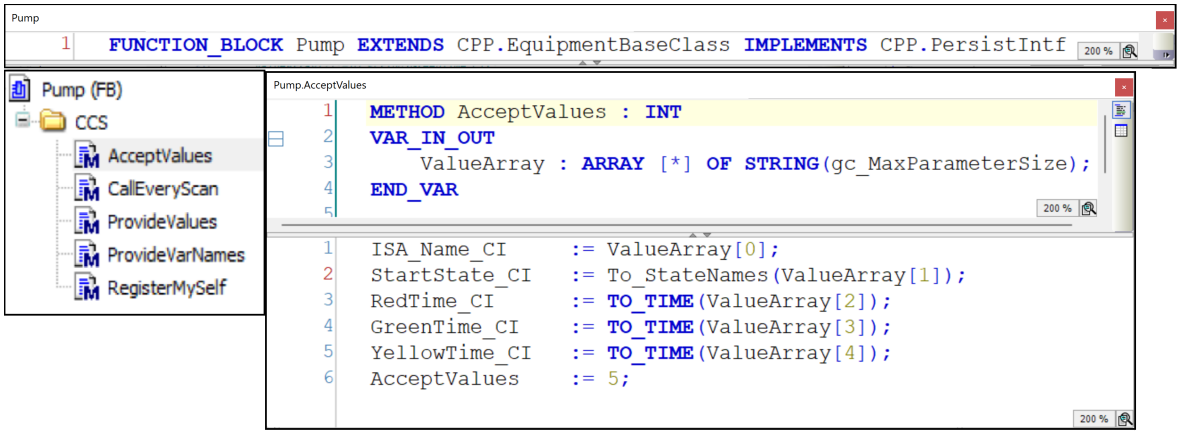


Figure 3: Typical Function Block Declaration and Methods

Features, advantages, and differentiators of the CPP Library

- Single Device License: The license can be used on the target device/PLC on which the CODESYS Runtime System SL is installed. Licenses are activated on a software-based license container (soft container), which is permanently connected to the controller. Alternatively, the license can be stored on a CODESYS Key (USB-Dongle). By replugging the CODESYS Key, the license can be used on any other controller. Without a license, the library may be used in the Demo mode with up to 6 instances of FBs that use the library.
- Unlimited single-site source-code license. Contact sales@controlsphere.pro

Manufacturer

Supplier	ControlSphere Engineering USA
Support	Technical support is not included with this product. To purchase support, email support@controlsphere.pro
Product	ControlSphere Parameter and Persistence Library
Oder Numer	2101000018
Sales	CODESYS Store https://store.codesys.com https.//us.store.codesys.com
Scope of Delivery	.zip file

Allgemeine Informationen

Lieferant:

Control Sphere Engineering
1211 San Juan Drive
32159, The Villages, Florida
USA

Support:

Technical support is not included with this product. To purchase support, email support@controlsphere.pro

Artikelname:

Parameter & Persistence Library (CPP)

Artikelnummer:

2101000018

Vertrieb / Bezugsquelle:

CODESYS Store
https://store.codesys.com

Lieferumfang:

.zip file (download from CODESYS Store), license key

Systemvoraussetzungen und Einschränkungen

Programming System	CODESYS Development System V3.5.18.0
Target System	CODESYS Control V3.5.18.0
Supported Platforms / Devices	CODESYS Runtime SL Products Notice: Use the project ‘Device Reader’ to find out the supported features of your device. ‘Device Reader’ is available for free in the CODESYS Store.
Additional Requirements	None
Licensing	Single-License (SL), one activation per CODESYS PLC. Six FB Instance demo mode
Restrictions	CPP Library source code must be compiled before being released from the developer
Required Accessory	CODESYS Runtime Key or SoftContainer

Bitte beachten Sie: Technische Änderungen, Druckfehler und Irrtümer vorbehalten. Es gilt der Inhalt der aktuellen Online-Version dieses Dokuments.

Erstellungsdatum: 30.05.2023