# **Overview**

This document is meant to assist in guiding you through setting up and developing with a CL-713/714 display in CODESYS.

# **INSTALLING PACKAGE**

To add the CL-713/714 device to CODESYS you will first need to obtain the .package file, which can be downloaded from the following location:

### \*DOWNLOAD .PACKAGE FILE HERE\*

Once downloaded you will need to install it into your CODESYS environment.

1. Launch your CODESYS Installer



2. Select "Change" and then "Install File(s)"

CODESYS Installer							-	
<i></i>							Add Installa	ation
Version CODESYS 64 3.5.20.0	D							
Location C:\Program Files\CO	DESYS 3.5.20.	0\CODESYS					Browse	► <b>9</b>
Channel for Setups Releases			Channel for Ad	d-ons		Update Mode All		
Add-ons								
					Install File(s)	Export Configuration	Import Cor	nfigurat
Installed	Browse	Updates						
😫 CAN Bus API				1.0.0.3				

3. Navigate to and select the .package file that was downloaded



Note: CODESYS may warn that the package file is not signed and prompts you if you would like to continue with the installation, continue with the installation. Once the process is complete you can launch CODESYS and have the ability to add this device to your project.

# **CREATING A PROJECT**

- 1. Launch CODESYS and select File->New Project
- 2. Select Standard Project

🖄 New Pro	ject				:
Categories		Templates			
Lib	raries ojects	Empty project	HMI project	Standard project	Standard project w
A project or	ontaining one device, one an	plication and an e	empty implement	tation for PLC	PRG
n project d	sindarining one device, one ap	pileadori, and an c	imply implement		
Name	NewProject				

3. When it launches it will ask what Device and Language you would like to use, for the Device choose the HED CL-713/714 Controller

Standard P	roject		$\times$					
	You are about to create a new standard project. This wizard will create the following objects within this project: - One programmable device as specified below - A program PLC_PRG in the language specified below - A cyclic task which calls PLC_PRG - A reference to the newest version of the Standard library currently installed.							
	Device	HED CL713/CL714 Controller (HED Hydro Electronic Devices)	$\sim$					
	PLC_PRG in	Structured Text (ST)	$\sim$					
		OK Cancel						

# ADDING I/O

- 1. To add I/O to the CL-713/714 right click the top-level device and then select Add Device
- 2. Under Miscellaneous follow the tree down until you see Connector A. Select that and click Add Device.

Add Device						
me Connector_A						
Action						
Append device O Insert device (	OPlug device OU	Ipdate device				
tring for a full text search	Vendor	<all vendors=""></all>				~
Name	Vendor		Version	Description		
Miscellaneous						
HED Hydro Electronic Device	S					
Connector A	HED Hydro Elec	tronic Devices	1.0.0.3	Conncetor A with IO 's		
GPIO	CODESYS		3.5.17.0	GPIOs using Sysfs		
] Group by category 🗌 Display all v	ersions (for experts o	nly) 🗌 Displa	ay outdated v	ersions		
Name: Connector A					^	
Vendor: HED Hydro Electronic De Categories:	evices					
Version: 1.0.0.3 Order Number:						
Description: Conncetor A with I	10 ´s				×	<u></u>
ppend selected device as last child	1 of					
evice						
You can select another target no	de in the navigator w	hile this windov	v is open.)			
					Add Device	
					Add Device	(1064

3. After being added your Device tree should look similar to this:



 To add IO you can right click on any of the <Empty> sockets, select Plug Device and choose an IO from the list

String for a full text search	Vendor	<all vendors=""></all>			
Name	Vendor		Version	Description	
	venuor		VEISION	Description	
E HED Hydra Electropic Davison					
Buttons					
💷 🖂 Output					
🗷 🧰 Power-Supply					
AmbientLight-Sensor	HED Hydro Elec	tronic Devices	1.0.0.1	Reads Ambient-Light-Sensor and activate Auto-Brightness-Functionality	
💮 🗊 Camera-Controller	HED Hydro Elec	tronic Devices	1.0.0.3	Camera-Controller (On/Off, Switch Source)	
Pin6 - Batteryvoltage	HED Hydro Elec	tronic Devices	1.0.0.3	Pin6 Voltage of Battery in mV	

Each IO has a Pin associated with it letting you know what pins can be configured for what IO:

Plug Device				
Name Pin1 Digital Ouptut				
Asian				
Action	~			
O Append device O Insert device O Plug d	evice O Up	pdate device		
String for a full text search	Vendor	<all vendors=""></all>		
Name		Vendor	Version	Description
HED Hydro Electronic Devices				
🖮 🚞 CL713/714				
🗷 🔤 Buttons				
🗐 - 📴 Input				
😟 💼 Counter				
🗷 🚞 Encoder				
🗄 🛅 Frequency				
🖻 · 🚞 PWM				
🖷 🛅 RTD				
🖻 🖓 🖾 STB				
Pin 10 - Input - Switch	n to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin10 - Configured as Switch to Battery-Input
Pin 9 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin9 - Configured as Switch to Battery-Input
Pin1 - Input - Switch t	o Battery	HED Hydro Electronic Devices	1.0.0.2	Connector A Pin1 - Configured as Switch to Battery-Input
Pin 12 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin12 - Configured as Switch to Battery-Input
Pin 15 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin15 - Configured as Switch to Battery-Input
Pin 16 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin16 - Configured as Switch to Battery-Input
Pin 17 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin17 - Configured as Switch to Battery-Input
Pin 18 - Input - Switch	to Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin18 - Configured as Switch to Battery-Input
Pin2 - Input - Switch t	o Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin2 - Configured as Switch to Battery-Input
Pin3 - Input - Switch t	o Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin3 - Configured as Switch to Battery-Input
Pin4 - Input - Switch t	o Battery	HED Hydro Electronic Devices	1.0.0.1	Connector A Pin4 - Configured as Switch to Battery-Input
🗏 🗎 🛅 STG				

As a note, you can configure a pin with an input and an output and the system will not flag an error so be careful and keep track of what pins are configured

5. It is highly recommended that you adjust the PLC Settings to allow the use of symbolic access of IO to make programming easier. You can do this by double clicking the CL-713/714 Device and going to the PLC Settings.

Input_Tristate 🗙 🔐 Device 🗙	Pin 10_Input_VTD	Pin10_Input_RTD Pin10_Input_PWM		Pin 1
Communication Settings	Application for I/O handling	Application	~	
Applications	PLC Settings			
Backup and Restore	Behavior for outputs in stop	Keep current values $\sim$		
Files	Always update variables	Disabled (update only if used in a task)	$\sim$	
Log	Bus Cycle Options Bus cycle task	<unspecified></unspecified>	~	
PLC Settings	,			
PLC Shell	Additional Settings Generate force variables for	or IO mapping 🛛 Enable diagnosis for devices		
Users and Groups	Show I/O warnings as error	Enable symbolic access for IOs		
Access Rights				
Symbol Rights				

Keep in mind that this setting is a per project setting and will need to be done for every new project that you do (the property is tied to project not workspace).

# **I/O TYPES AND CONFIGURATION**

### STB/STG/TRISTATE INPUTS

STB, STG, and TRISTATE inputs refer to digital inputs where the input is ON when connected to their designated type and OFF when not i.e. STB is ON when it sees battery voltage and OFF when it see ground and vice versa for STG, it is ON when it sees ground and OFF when it sees battery voltage. TRISTATE specifically can detect when connected to a source, ground, or open.

All three of these inputs have 2 properties associated with them for configuration, DebounceTimeOn and DebounceTimeOff

HED:CA713714 IEC Objects	Parameter	Туре	Value	Default Value	Unit	Description
21-4 D	PebounceTimeOn	DWORD	5	5	10ms	Configuration of DebounceTime On
Pin1 Parameters	DebounceTimeOff	DWORD	5	5	10ms	Configuration of DebounceTime Off
Pin1 I/O Mapping	😑 🖗 libHEDIIO					
r mi i jo r hopping	HEDIIO Config Status Feedback	STRING				Status from last Configuration
Status	HEDIIO Input Feedback	STRING				Input-Status (Realtime)

Both of these values default to 5 with a unit of 10mS, so by default the debounce time for both is 50mS. Debounce is a property that the input must see the value for at least this long to be considered ON/OFF.

For example, a STB input sees battery voltage, it must see that voltage for at least 50mS before the input will consider it valid and report the input as ON. With the symbolic IO enabled, accessing the input value would be done from looking at the input property.



### **RTD INPUT**

RTD inputs refer to inputs that read in a resistance value and report back the resistance in ohms. This input has 1 property, Analog HS Filter Frequency. That property should be set to 0 so that the input is read only once per loop. With symbolic IO enabled, accessing the input would be done from the Input\_RTD property.



## **VTD INPUT**

VTD inputs refer to inputs that read in a voltage and report back what that voltage is in mV. Generally, this is a 0-5V range with the exception of the Battery Voltage Pin. This input has 1 property, Analog HS Filter Frequency. That property should be set to 0 so that the input is read only once per loop. With symbolic IO enabled, accessing the input would be done from the Input\_VTD property.



#### **PWM INPUT**

PWM inputs read and report back the on-percentage duty cycle of a square wave signal in a percentage of 0-100.0% (real value reported is 0-1000). This input type currently has no adjustable parameters. Using the symbolic IO you can access the input value by looking at the Input\_PWM property.



### FREQENCY INPUT

Frequency inputs read and report back the frequency of a square wave signal in Hz. This input type currently has no adjustable parameters. Having the symbolic IO enabled you can access the input value by looking at the \_Input\_Frequency property.



#### **COUNTER INPUT**

The counter input is a pulse counter that will report back the number of square wave pulses the input experiences over the loop time of the application. The counts will be accumulated based on the falling edge detected. There aren't any adjustable parameters for this input type. If you have the symbolic IO enabled you can read the number of pulses by accessing the \_Input\_Counter property.

PLC	_PRG	
⊟	1 2 3 4	PROGRAM PLC_PRG VAR Input_Val: DWORD := 0; END_VAR
	1	<pre>Input_Val := Pinl0_Input_Counter.Pinl0_Input_Counter;</pre>

### BUTTONS

The buttons are grouped together into a single socket called Buttons. Add them to a socket just as you would for any other I/O (Right click Plug Device). Double clicking on the object you can view the I/O Mapping to see all 4 buttons and the 4 Navigation buttons pre-defined.

🕅 Pin10_Input_VTD 🕅 Pin1	10_Input_RTD 🕅 Pin 10_Ir	nput_PWM	Pin 10_	Input_Freque	ncy	😫 Pir	10_Input_Counter
HED:CA713714 IEC Objects	Find		Filter Show all				- 🕂 Add FB
MyInterface2 I/O Mapping	Variable	Mapping	Channel Button 1	Address	Туре ВООІ	Unit	Description
Status			Button 2	%IX36.1	BOOL		
Information	-   · · · · · · · · · · · · · · · · · ·		Button 3 Button 4	%IX36.2 %IX36.3	BOOL		
	-		Button Up Button Down	%IX36.4 %IX36.5	BOOL		
	*		Button Left	%IX36.6	BOOL		
	↓		Button Right	%IX36.7	BOOL		

If you set up your device to utilize the symbolic IO you can reference the buttons themselves based on the Channel Name

2	Pin 1	l0_Input_VTD			Pin10_Input_RTD	Pin10_1
	1	PROGRAM	PLC	PRG		
	2	VAR				
	з	END_VAR				
	-					
	1	Buttons.	Butt	on 1	ž.	

Otherwise, you will have to reference them by their Address which can change depending on what socket you plugged the Buttons into on the Connector A Device.

	_RT
2 VAR 3 END_VAR	
3 END_VAR	
Buttons Button 1.	
Buttons Button 1.	
Buttons Button 1:	
Buttons Button 1.	
1 Buttons Button 1.	
1 Buttons Button 1:	
- Davoons.Davoon 1,	
2 %IX36.0;	

### **BUTTON BACKLIGHT**

This is an output that allows you to control the backlight brightness of the buttons. If you have the symbolic IO mapping enabled you can access the 2 properties easily to assign a value to. The ButtonBacklight\_Set\_Brightness is what you would assign a value of 0-1000 (100.0%) to control the backlight. The second property \_Actual\_Brightness is what the driver is actually setting the backlight to and would only differ from the \_Set\_Brightness if the controller was throttling the backlight due to thermal conditions. Use the \_Actual\_Brightness for displaying what the backlight is on a visualization and use \_Set\_Brightness to actually control the backlight.



#### **KEYSWITCH AND BATTERY INPUTS**

The Keyswitch and Battery Inputs are already pre-socketed into the project when you add the CL-713/CL-714 device. These monitor battery voltage (in mV) and the switched power pin (TRUE/FALSE for detection) for you. With the symbolic IO enabled you can access the values read by them by looking at their respective properties.

PLC_	PRG	
Э	1 2 3 4 5	<pre>PROGRAM PLC_PRG VAR Batt_Val: DWORD := 0; Key_Val: BOOL := FALSE; END VAR</pre>
	1 2	Batt_Val := Pin6_Batteryvoltage.Pin6_Batteryvoltage_mV_; Key_Val := KeySwitch_Input.KeySwitch_Input;
	3	

#### AMBIENT LIGHT SENSOR

The ambient light sensor is another input that is pre-socketed into the project upon adding the CL-713 device. This is a sensor that reports back a reading, in lux, based on the amount of light it is exposed to where the higher the value the more light it is exposed to. With the symbolic IO enabled you can access the value read by the sensor but looking at the sensor property.



#### **ENCODER INPUT**

The encoder inputs are a pair of pins that act as 1 input type. These inputs do not have any configurable parameters and reading it will return a signed value giving you the amount of change and in which direction for that given application loop cycle. You will need to create and maintain your own count within your code. With the symbolic IO enabled you can obtain the value by looking at the input parameter.



## **DIGITAL OUTPUT**

Digital outputs are controlled to be ON and OFF without any variance between. These outputs will have a few configuration parameters:

- Softwarefuse trip point: The value in mA in which a software fuse will be set at so that when the output is detected to go over that point it will trigger an overcurrent error.
- Softwarefuse Debounce time: The time in 10mS in which the output current needs to be above the trip point for the overcurrent error to trip.
- OpenCircuit-Detection: A value in mA in which the open circuit detection will turn on when below it.
- Open when Off-Detection-Mode: Can enable or disable the detection for open circuit when off.
- Load Type: Can enable certain protections on the pin based on the type of load, Load NA and Non-Inductive are standard loads and lastly if you are connecting to an inductive load ensure you use the Inductive Load type.

With the symbolic IO enabled, all that needs to happen to turn the output on or off would be to set the Boolean parameter true or false.



## **PWM OUTPUT**

The PWM output is an output that has a fixed frequency and a variable duty cycle that you command from 0-1000 (0-100.0%). This output type has some configurable parameters:

- Softwarefuse trip point: The value in mA in which a software fuse will be set at so that when the output is detected to go over that point it will trigger an overcurrent error.
- Softwarefuse Debounce time: The time in 10mS in which the output current needs to be above the trip point for the overcurrent error to trip.
- OpenCircuit-Detection: A value in mA in which the open circuit detection will turn on when below it.
- Open when Off-Detection-Mode: Can enable or disable the detection for open circuit when off.
- Load Type: Can enable certain protections on the pin based on the type of load, Load NA and Non-Inductive are standard loads and lastly if you are connecting to an inductive load ensure you use the Inductive Load type.
- Frequency: The frequency in which you desire the output to operate at ranging from 50-1000Hz.

With the symbolic IO enabled, all that needs to happen to turn the output on or off would be to set the output parameter to a value ranging from 0-1000.

PLC_PRG	
1	PROGRAM PLC_PRG
2	VAR
3	END_VAR
1	<pre>Pinl_PWM_Output.Pinl_PWM_Output := 755; //Turn pin 1 PWM ON at 75.5%</pre>

### FREQUENCY OUTPUT

The Frequency output is an output that has a fixed Duty Cycle and a variable Frequency that you command up to 1000Hz. This output type has some configurable parameters:

- Softwarefuse trip point: The value in mA in which a software fuse will be set at so that when the output is detected to go over that point it will trigger an overcurrent error.
- Softwarefuse Debounce time: The time in 10mS in which the output current needs to be above the trip point for the overcurrent error to trip.
- OpenCircuit-Detection: A value in mA in which the open circuit detection will trip.
- Open when Off-Detection-Mode: Can enable or disable the detection for open circuit when off.
- Load Type: Can enable certain protections on the pin based on the type of load, Load NA and Non-Inductive are standard loads and lastly if you are connecting to an inductive load ensure you use the Inductive Load type.
- Duty Cycle: Set the Duty Cycle of the signal from 100 to 900 (10.0-90.0%)

With the symbolic IO enabled, all that needs to happen to turn the output on or off would be to set the output parameter to a value ranging from 0-1000 and it will operate at the configured duty cycle with the commanded frequency.

PLC_PRG	
1 2 3	PROGRAM PLC_PRG VAR END_VAR
1 2	<pre>Pinl_Frequency_Output.Pinl_Frequency_Output := 100; //Turn on at 100Hz</pre>

## ECC OUTPUT

The ECC output is an estimated constant current output, meaning it will try to output a specific amount of current that the user commands, in mA. Since there is no feedback to measure the actual current, the estimated value is determined internally through circuitry and algorithms. This output has a few things to configure with a bit of a nuance.

- Softwarefuse trip point: The value in mA in which a software fuse will be set at so that when the output is detected to go over that point it will trigger an overcurrent error.
- Softwarefuse Debounce time: The time in 10mS in which the output current needs to be above the trip point for the overcurrent error to trip.
- OpenCircuit-Detection: A value in mA in which the open circuit detection will turn on when below it.
- Open when Off-Detection-Mode: Can enable or disable the detection for open circuit when off.
- Load Type: This output will require the load type to be set so you must use the Inductive or Non Inductive load type to allow the calculations to run correctly.
- Frequency: The frequency in which you desire the output to operate at ranging from 50-1000Hz.
- Duty Cycle Offset: This property is an offset to start the PID output command at a higher setpoint to allow for slightly faster response/ramp up to target.
- P gain: The P Gain term for the internal PID control loop that will run to maintain a constant current output.
- I gain: The I Gain term for the internal PID control loop that will run to maintain a constant current output.
- D gain: The D Gain term for the internal PID control loop that will run to maintain a constant current output.

Having the symbolic IO enabled you can command a current by setting the property to the value you desire.

PLC_PRG	i	
1	PROGRAM PLC_PRG	
2	VAR	
3	END_VAR	
1	<pre>Pinl_ECC_Output.Pinl_ECC_Output := 1750;</pre>	//Command output to produce 1750mA

#### **BACKLIGHT OUTPUT**

This is an output that allows you to control the backlight brightness of the display. If you have the symbolic IO mapping enabled you can access the 2 properties easily to assign a value to. The LCDBacklight\_Set\_Brightness is what you would assign a value of 0-1000 (100.0%) to control the backlight. The second property \_Actual\_Brightness is what the driver is actually setting the backlight to and would only differ from the \_Set\_Brightness if the controller was throttling the backlight due to thermal conditions. Use the \_Actual\_Brightness for displaying what the backlight is on a visualization and use Set Brightness to actually control the backlight.

PL	C_PRG	
⊟	1 2 3 4	PROGRAM PLC_PRG VAR Screen_Brightness: DWORD := 0; END_VAR 100 % 🕅
	1 2	LCDBacklight.LCDBacklight_Set_Brightness := 1000; //Full 100.0% brightness Screen_Brightness := LCDBacklight.LCDBacklight_Actual_Brightness; //Display what the screen brightness is

#### **5V POWER SUPPLY**

The 5V supply is actually 2 pins, a supply (Pin 18) and a return (Pin 17). You will need to plug both into sockets in order to use this feature. Both of the pins will have a min and max voltage setting, the min must remain below the max. You will want to set your min and max for both pins and then use the \_Activate property to turn them on (both pins need to be turned on for it to work properly).



## **CAN – SENDING AND RECEIVING CAN MESSAGES/SIGNALS**

There are a few ways of sending and receiving CAN signals but the way we suggest will be through the use of a J1939 CAN library stack that is already included with the CL-713/714 Device.

Before sending or receiving CAN messages, you need to define the CAN bus devices. To begin, right click on the HED CL713/CL714 Controller and select Add Device

J1939_Test			
E · I Device (HED CL713/CL714 Contr		General	1
	*	Cut	E
🖻 💮 Application		Сору	
📲 📶 Library Manager	e	Paste	
PLC_PRG (PRG)	$\times$	Delete	
=-∰ Task Configuration =-ॐ MainTask		Refactoring	
PLC_PRG	Ē.	Properties	
ا VISU_TASK الطبي العند (USU_TASK) الحي	*::	Add Object	
Sisualization Manage		Add Folder	
TargetVisu		Add Device	
Engine Data Scree		Update Device	

It will open a window where you can expand Fieldbuses. Under Fieldbuses you can select CANbus then select CANbus (not NetX CANbus).

) Plua device	OU	pdate device		
Ver	ndor	<all vendors=""></all>		
Vendor	Vers	ion Descriptio		
CODESYS	4.2.0	0.0 Needed for		
CODESYS	4.2.0	0.0 CANbus or		
🗄 🖞 Home&Building Automation				
🗄 - 📖 Modbus				
PROFIBUS				
	Plug device Vendor CODESYS CODESYS	Plug device O U Vendor Vers CODESYS 4.2.0 CODESYS 4.2.0		

Double click on your CANbus device to bring up its properties editor. Under the General tab is where you select the CAN line and set the baud rate (CAN 1 is Network 0; CAN 2 is Network 1).

CANbus X			
General	General		
Log	Network	0	CAN
CANbus IEC Objects	Baud rate (kbit/s)	250 ~	

Once configured, you will need to add the J1939 Device by right clicking on the CANbus Device and selecting Add Device. You now have the option to add a J1939 Manager or a CANopen Manager. If you have an EDS file, select CANopen. For engine/chassis interfaces select SAE J1939 Manager.

Name	J1939_Manager						
Action	Action						
• Арр	oend device 🔘 Insert device 🤇	) Plug device	Up	date device			
String 1	for a full text search	Ver	ndor 占	<all vendors=""></all>			
Nam	e	Vendor	Versior	Description			
B- f	Fieldbuses						
E E	CANopen						
B	👫 J1939						
	🖹 - 👫 J1939 Manager						
	💮 👚 🗍 J1939_Manager	CODESYS	4.1.0.0	J1939 Manager			

It is at the J1939 Manager that you can import DBC files to be used later by your J1939 devices. Under the General tab you can Install DBC files as databases to be used later.

CANbus J1939_Engine	J1939_Interface X J1939_Engine_TX
General	Databases
Log	Install Uninstall Set as default
J1939 I/O Mapping	
J1939 IEC Objects	
Status	
Information	

You are now ready to add a J1939 ECU device – you need to create a separate device for each source address your application will be sending or receiving messages.

Action							
Append device      Insert device      Plug device      Update device							
String for a full text search		Vendor	<all vendors=""></all>				
Name	Vendor	Version	Description				
🖃 🔟 Fieldbuses							
🖃 👫 J1939							
🖃 - 👫 J1939 ECU							
🚹 J1939_ECU	CODESYS	4.1.0.0	J1939 ECU				
Untitled1							
🖃 👚 Device (HED CL713/CL714 C	ontroller)						
PLC Logic							
Application							
Library Manager	r						
PLC_PRG (PRG)							
i≡ with a second secon							
E 11939 Manager (J1939 Manager)							
J1939_ECU (J1939_ECU)							

To begin adding messages and/or signals you can double click on the J1939\_ECU device to bring up a GUI that will allow you to begin configuring that CAN device.

es <b>→</b> म X	J1939_ECU 🗙	
Dutitled 1	General	General
BLC Logic	TX Signals	Preferred address 0
Library Manager	Log	Local device
Task Configuration	J1939 IEC Objects	✓ ECU NAME
Main Lask	Status	NAME (64 bit): 16≢ 0 Arbitrary Address Capable □
In Information (Canadas)	Information	Industry Group 0: Global, applies to all industries V
11939_ECU (11939_ECU)		Vehicle System Instance 0
		Vehicle System 0
		Reserved
		Manufacturer Code 0
		Identity Number 0
		Communication Watchdog
		Enable Communication Watchdog
		Watchdog Time (in ms):

Preferred Address is the Source Address of this device, i.e. most engines would be 0 and some transmissions would be 3 or 11 etc.

Please note, as this is an external device (Local Device is not checked) the messages defined under Tx Signals are the messages that will be Received by the CL-713. If you want to use the CAN Device to send messages, you need to check Local Devices.

To add a message that would be transmitted by that remoted device to the display you can go into TX Signals and click Add PG

General	Enable	PGN/SPN
TX Signals		
Log		
J1939 IEC Objects		
Status		
Information		
	Add P	G Add

By default, there is no database loaded but if you had a database of messages you could choose directly from that, otherwise Custom needs to be chosen. Under Custom you can define the parameters of the message with a restriction that the PGN (PDU Specific and PDU Format) must remain within the standard J1939 defined space – you cannot define a proprietary message for the device configured as is. To define a proprietary message, you would need to set the device to a Local Device.

After adding a PGN you can then begin adding signals to that PGN, defining how long the signal will be, scaling, offset, name, byte order, etc. Once the signal is added you can then adjust the bit position in which it will start from.

- Length is the length of this data value in bits
- Byte Position is the starting bit within the 8 data bytes of the CAN message
- **Bit Position** is the starting bit
- Scaling is a multiplier to the value extracted from the CAN message.
- Offest is added to the value extracted from the CAN message

Enab	e	PGN/SPN	Name	Bit Position	Length	Туре
8		0(16#0)	TSC1		8 Butes	DOD
		0(10#0)			obytes	r Zr
		898	RPM_Command	8	16 Bits	

To utilize the data you are extracting from CAN messages, simply use the Signal name in your application. This line sets a local variable RPM\_Command to the Engine\_Speed read from a CAN message and adds 50 to that value.

#### RPM\_Command := Engine\_Speed + 50;

Choosing to click the box for Local Device this would designate the ECU device to be a simulated CAN device coming from the controller/display itself. You will see that a new tab appears, P2P Rx Signals

^	2	1133a_ECO X		
-		General	General	
		TX Signals	Preferred address 0	-
		P2P RX Signals	✓ Local device	
		Log	I ECU NAME	
		J1939 IEC Objects	NAME (64 bit): 16# 0	
			Arbitrary Address Capable	
		Status	Industry Group	0: Global, applies
		Information	Vehicle System Instance	0 ≑
			Vehicle System	0
			Reserved	
			Function	0
			Function Instance	0
			ECU Instance	

When set as a local device the Tx Signals are the defined messages that the display will transmit out and the P2P RX Signals are the messages it would expect to receive. The Preferred Address in regard to Tx Signals is the Source Address that the message will be transmitted with, but in regard to P2P RX Signals it would be the Destination Address of the message ID. Adjusting the P2P RX Signal source address will be done within that tab directly under the message settings of the PG you have defined.

To define a message to be **transmitted from** the display go under Tx Signals, like before, and click the Add PG and then define your message under the Custom tab. With the device now selected as local, you will be able to define a PGN within the proprietary realm of Identifiers. For these transmitted messages, selected the PGN allows you to configure how the message is sent: On Change, Cyclic, etc.

Enable	PGN/SPN	Name	Bit Position	Length	Туре
+ 🖂	0 (16#0)	TECI		8 Buter	DOD
~ <b></b>	0(10#0)	1501		obytes	FZF

To define a message to be **received by** the display, you would click the P2P RX Signals and add a PG there. Or you can add a new device to receive those messages.

# **CREATING SCREENS**

Screens are added through the use of visualizations within CODESYS.

- 1. Right click on Application and then Add Object->Visualization
- 2. It is recommended after the visualization is created to right click on it and adjust the properties such that it has a fixed size compatible with the display screen size of 800x480 pixels.

🚆 Properties - Visualization [I	Device: PLC Logic: Application]	×
Common Build Access Cont	rol Visualization	
Use Visualization as	/isualization	$\sim$
Dialog is opaque		
O Use automatic detected	visualization size	
Include backgrou	nd image	
Use specified visualization	ion size	
Visualization Size Width 800	Height 480	
Internal		
Use visualization and cor	tainer size properties	
	OK Cancel	Apply

3. From here create your HMI as you normally would within CODESYS

# SWITCHING BETWEEN SCREENS IN YOUR APPLICATION

If your application has more than one screen, you will need a way to change which screen is being shown on the display. There are many methods to do this, and most can be found by doing a quick search on the internet or using one of the many CODESYS examples, but here is one of those methods.

1. Open the Visualization Manager to find the "Use CurrentVisu variable" and ensure that is checked



2. Create your separate visualizations visualizations



3. To change what visualization will be visible you will need to make use of that CurrentVisu variable by setting it equal to the visualization you wish to show.



This small example is making use of the buttons to change to a specific screen which sets the CurrentVisu variable equal to the name of the visualization you wish to have shown.

# **CHANGING OF SPLASH SCREEN**

The splash file is a special file that represents the image used for the boot time splash screen. This file cannot be made within CODESYS, but it can be transferred/downloaded from within the CODESYS environment.

To do this you will need to create an image outside of the CODESYS environment following a few simple rules:

- 1. The image type is one of: PNG, BMP, JPG, or GIF
- 2. The file name is named splash with no file extension.
- 3. The resolution of the file must be exactly 800x480.

Once the file is made you will be able to add it to your project by right clicking on Application -> Add Object -> External File which will bring up a window for you to navigate to the file and select an option:

Add External File X
Create a reference to an external file
File path
C:\Users\rymans\Pictures\splash
Name
splash
File Handling
Change Tracking Reload the file automatically Prompt whether to reload the file Do nothing
Display File Properties
Online handling
✓ Transfer with Download/Online Change
Target path (relative to "\$PlcLogic\$" on the device)
Add Cancel

Please ensure to keep the Target Path empty, there are internal scripts within the display that will properly place the file on the target. You may select "Embed into project" option if you wish to keep the image saved/contained within the project itself, though it is not required.

Once added to the project you should be able to see the splash within your Application tree structure:



After the image is added to the application you may login to your device and download as you normally would and application (following section below). That will get the file onto the device, but the splash will not be immediately available. The device will need to run some internal scripts on a reboot which will relocate the needed file and force another reboot of the device. Upon that second reboot, your splash image should appear as the splash going forward.

# **PROGRAMMING DISPLAY THROUGH CODESYS**

By default your display should come with the CODESYS runtime installed on it from the factory. To download your CODESYS application to it you will need the M12 USB cable connected to the USB port on your display. Power the display up and then plug the USB into your PC. From there you can doubleclick on the top level device in your project

-EP CANVILSTING
Device (HED CL713/CL714 Controller)
□ III PLC Logic
🖹 🧔 Application
🖓 🎁 Library Manager
PLC_PRG (PRG)
🖻 🎆 Task Configuration
i 📣 martin

Next, under Communication Settings click Scan Network

Communication Settings	Scan Network Gateway - Device -
Applications	
Backup and Restore	
Files	
Log	Gateway

Once the scanning window comes up you can click the scan network and look for the Linux-Display.

Select Device		×
Select the Network Path to the Controller		
Gateway-1	Device Name: Gateway-1 Driver: TCP/IP	Scan Network
	IP-Address: localhost Port: 1217	

Double-click on the display, you may be asked to create a user management profile for that display if you have not done so already, if you have then proceed logging into the user account. After logging in your target should show as the Linux Device

Scan Network Gateway - D	Device 🔻	
	Gateway	
	Gateway-1 $\checkmark$	Linux-Display-22228070516 (active) $\sim$
	IP-Address: localhost	Device Name: Linux-Display-22228070516
	Port: 1217	Device Address: 0301.400B
		Target ID: 1837 0001
		Target Type: 4096
		Target Vendor: HED Hydro Electronic Devices
		Target Version: 3.5.20.0

Once the target is set you can then load your application by selecting the Online->Login, Alt+F8, or clicking the log in icon at the top. This will prompt the download of your application.

Window Help				
🦄 🦄   🛱   🏪 - 😚   🎬   .	Application [Device: PLC Logic] 🔹 🥰 💖 🕟 🔳 🔏   💭 🖙 🖄 🗦			
J1939_ECU PRG Device X				
mmunication Settings	Scan Network Gateway - Device -			

# **PRE-PROGRAMMING OPTION**

We offer a pre-programming option where you provide HED the necessary files and we place your application on the display(s) in production so they ship ready to go from the factory. To take advantage of this, please work with your HED Sales Representative.

The file(s) we would need would be the .projecarchive file. To create a project archive: in your project, select File -> Project Archive -> Save Archive. Once the file is created send that to us along with any permissions information we may need, pending your permissions and user management within the project, to open the project to download to a display.