

AN002-Control a Clean Power VFD with PLC using Modbus TCP



Table of Contents

Motor Speed Control with Clean Power VFD & PLC using Modbus TCP	3
Application details: VFD & PLC using Modbus TCP	4
Implementation guide VFD & PLC using Modbus TCP	6
Prepare the Hardware - VFD & PLC using Modbus TCP	7
Connect the Hardware - VFD & PLC using Modbus TCP	13
Configure the PLC - VFD & PLC using Modbus TCP	15
Configure the VFD - VFD & PLC using Modbus TCP	16
With Schneider Electric M221	18
A generic Structured Text program.....	24
Heartbeat register	29
Benefits and results - VFD & PLC using Modbus TCP	30
Conclusion - VFD & PLC using Modbus TCP	31

Motor Speed Control with Clean Power VFD & PLC using Modbus TCP

Introduction to Integrating Clean Power VFD with PLC using Modbus TCP

In this application note, we will explore the integration of the Clean Power Variable Frequency Drive (VFD) with a Programmable Logic Controller (PLC) using Modbus TCP

The Clean Power VFD is a highly efficient product designed to control motor speed and torque by producing a pure sine wave, which helps to minimize harmonic distortion and improve the lifespan of equipment.

Modbus TCP is a robust protocol widely used in industrial automation for communication between devices such as PLCs and VFDs. It allows for efficient and reliable data exchange over Ethernet networks.

By the end of this note, you will have a comprehensive understanding of how the integration of Clean Power VFD and PLC can revolutionize motor control, leading to significant energy savings and extended equipment lifespan.

The goal of this guide is to demonstrate how to control motor speed, manage the direction of rotation, and monitor alarm statuses using Modbus TCP communication between a PLC and a Clean Power VFD.



Application details: VFD & PLC using Modbus TCP

Application Overview

The Clean Power VFD adjusts the frequency and voltage supplied to the motor, thereby controlling its speed.

The operation of the VFD is remotely managed using a Programmable Logic Controller (PLC).

The PLC provides a set point to the VFD, which determines the desired motor speed. This set point can be established either automatically by the PLC or manually by an operator using a potentiometer or any other human-machine interface.

This application note focuses on the control command done through Modbus TCP, for interface purposes we use a simple potentiometer and a selector switch to use predefined speeds as well as a control of the spinning direction

Upon receiving the set point, the PLC adjusts the speed of the VFD. The VFD, in turn, controls the motor speed according to the set point. This precise control mechanism allows optimal motor performance, significant energy savings, and extended equipment lifespan.

This application overview delves into the integration of the Clean Power VFD with a PLC for advanced motor speed control. It highlights the process of set point determination, the role of the PLC in controlling the VFD through Modbus TCP, and how the VFD ultimately controls the motor speed.

The overview underscores the benefits of this integration, including improved energy efficiency, reduced harmonics, and extended equipment lifespan

Why use or not a PLC to control a VFD application

SmartD VFDs do have built-in functionalities for basic motor control that allow them to get rid of an added PLC ... but not always.

Here's a breakdown of the pros and cons of using a PLC (Programmable Logic Controller) alongside a VFD, even for basic operations:

PLC control of VFD – Pros:

- **Flexibility:** PLCs offer a wider range of programming capabilities. You can create complex control algorithms that factor in sensor data, external conditions, and safety protocols for a more precise and responsive system.
- **Scalability:** A single PLC can control multiple VFDs, motors, and other devices within a system. This reduces hardware complexity and simplifies wiring.
- **Data Logging and Monitoring:** PLCs can record operational data from the VFD and other connected devices. This allows for performance analysis, troubleshooting, and preventative maintenance.

- **Communication:** PLCs can communicate with other control systems using various protocols, enabling integration into larger industrial automation networks.

PLC control of VFD – Cons:

- **Cost:** Adding a PLC increases the initial cost of the system.
- **Programming Complexity:** Programming a PLC requires some expertise, adding an extra layer of complexity to setup and maintenance.

VFD control without PLC – Pros:

- **Simplicity:** Setting up a basic VFD for on/off and speed control is often straightforward.
- **Cost-effective:** For simple applications, using just a VFD is a more budget-friendly option.

VFD control without PLC – Cons:

- **Limited Functionality:** Complex control logic or integration with other devices becomes difficult or impossible.
- **Scalability Issues:** Controlling multiple VFDs or motors requires additional wiring and can become complex.
- **Data Limited:** Monitoring and recording operational data is typically limited to basic parameters offered by the VFD.

In conclusion, for basic operations, a VFD on its own might be sufficient. But if you need:

- Precise control with complex logic
- Scalability for multiple devices
- Data recording and communication
- Integration into a larger automation system

Then using a PLC to control your VFD offers significant advantages despite the added cost and programming complexity.

Implementation guide VFD & PLC using Modbus TCP

Step-by-step instructions on integrating and configuring the Clean Power VFD for the specific application.

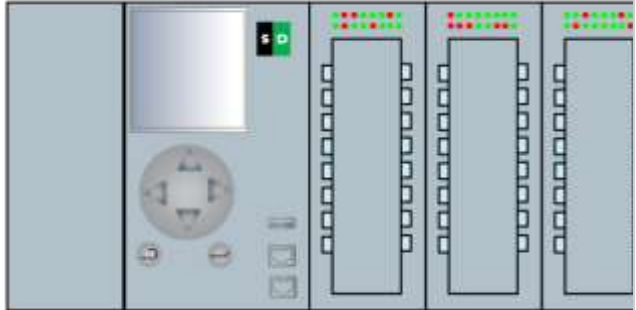
1. **Prepare the Hardware:** Gather all the necessary hardware components including the Clean Power VFD, a PLC, a human-machine interface for manual set point determination, and the motor you wish to control, not forgetting the protections according to local and international regulations.
2. **Connect the Hardware:** Connect the Clean Power VFD to the motor following the user manual's instructions.
Ensure the PLC is connected to the VFD and the chosen human-machine interface.
3. **Program the PLC:** Write a PLC program that will provide a set point to the VFD. This set point can be determined either automatically by the PLC or manually by an operator using the chosen human-machine interface.
Set the control command between the PLC and the VFD through Modbus TCP.
4. **Configure the VFD:** Set up the Clean Power VFD according to the user guide.
This includes setting the input voltage, output current, and other parameters based on your specific motor and application requirements.
5. **Test the System:** Once the hardware is connected and the PLC program is written, test the system.
Adjust the set point and observe the motor speed.
Ensure the VFD is correctly controlling the motor speed and rotation direction according to the set point and the RUN commands.
6. **Monitor and Adjust:** After the initial setup and testing, continue to monitor the system and make any necessary adjustments.
This could include fine-tuning the set point or making changes to the PLC program.

By following these steps, you can successfully integrate and configure the Clean Power VFD with a PLC for advanced motor speed control using Modbus TCP.

. This will allow optimal motor performance, significant energy savings, and extended equipment lifespan.

Please note that these instructions are general guidelines. For specific instructions related to your particular hardware and application, refer to the manufacturer's documentation.

Prepare the Hardware - VFD & PLC using Modbus TCP

Component	The usual rule of selection	Benefits of VFD using SmartD VFD
Motor	The selection of the motor depends on the specific requirements of your application. Consider factors such as the type of load (constant torque, variable torque, etc.), the required speed range, the power rating, and the operating environment. Make sure the motor is compatible with a VFD, as not all motors are designed to work with VFDs	Compatibility with Non-VFD-Rated Motors: While not all motors are designed to work with VFDs, the pure sine wave output of the Clean Power VFD can be more compatible with non-VFD-rated motors. This is because the sine wave is closer to the natural power supply of the motor, reducing potential damage to the motor's insulation caused by the high voltage spikes of a square wave. However, it's always recommended to check with the motor manufacturer or a qualified engineer to ensure compatibility between the motor and the VFD.
VFD	The first criteria of selection are the line and motor voltages. Then select the VFD Based on the Motor's Current Requirements: The VFD should be able to supply the motor's full-load current under normal running conditions. Moreover, the VFD should be able to handle the inrush current during the motor's startup phase. For instance, if your motor has a full-load current of 28A, you could select the Clean Power VFD model SDB-I-2185-A, which has an output current (ND) of 34A.	Speed up the VFD selection: Match Clean Power VFD Ratings with Motor Power. Then check the rated current vs the motor operation.
PLC	List the inputs and outputs of the application, as well as the communication ports and protocol to select the PLC architecture that fits the application	
PLC I/O's list	Communication: • Modbus TCP Parameters exchanged on communication media • VFD => PLC ◦ VFD ready ◦ VFD alarm • PLC => VFD ◦ Motor run forward ◦ Motor run reverse ◦ Stop ◦ Speed reference Digital Inputs: • Mode Auto • Mode Manual	

Component	The usual rule of selection	Benefits of VFD using SmartD VFD
	- Speed selection Digital Outputs: - Tower light for - Equipment running - Equipment fault - Equipment powered Analog Inputs: - Manual Speed setpoint (0..10V) - VFD speed feedback Analog outputs: - Speed reference (4..20mA)	
Electrical distribution and protection	From the connection point of the system to the power network, use - a power main switch. - a 24VDC power supply to power up the various devices, select it regarding the needed current. - a circuit breaker to protect the input of the DC power supply - Circuit breaker and Fuse as recommended in the user guide to protect the VFD.	  

Component	The usual rule of selection	Benefits of VFD using SmartD VFD
		 
Safety	Select a safety relay and emergency actuator according to the risk assessment of your application.	For low risk processes, requiring a simple stop category 0, the Clean Power VFD has 2 STO terminals 

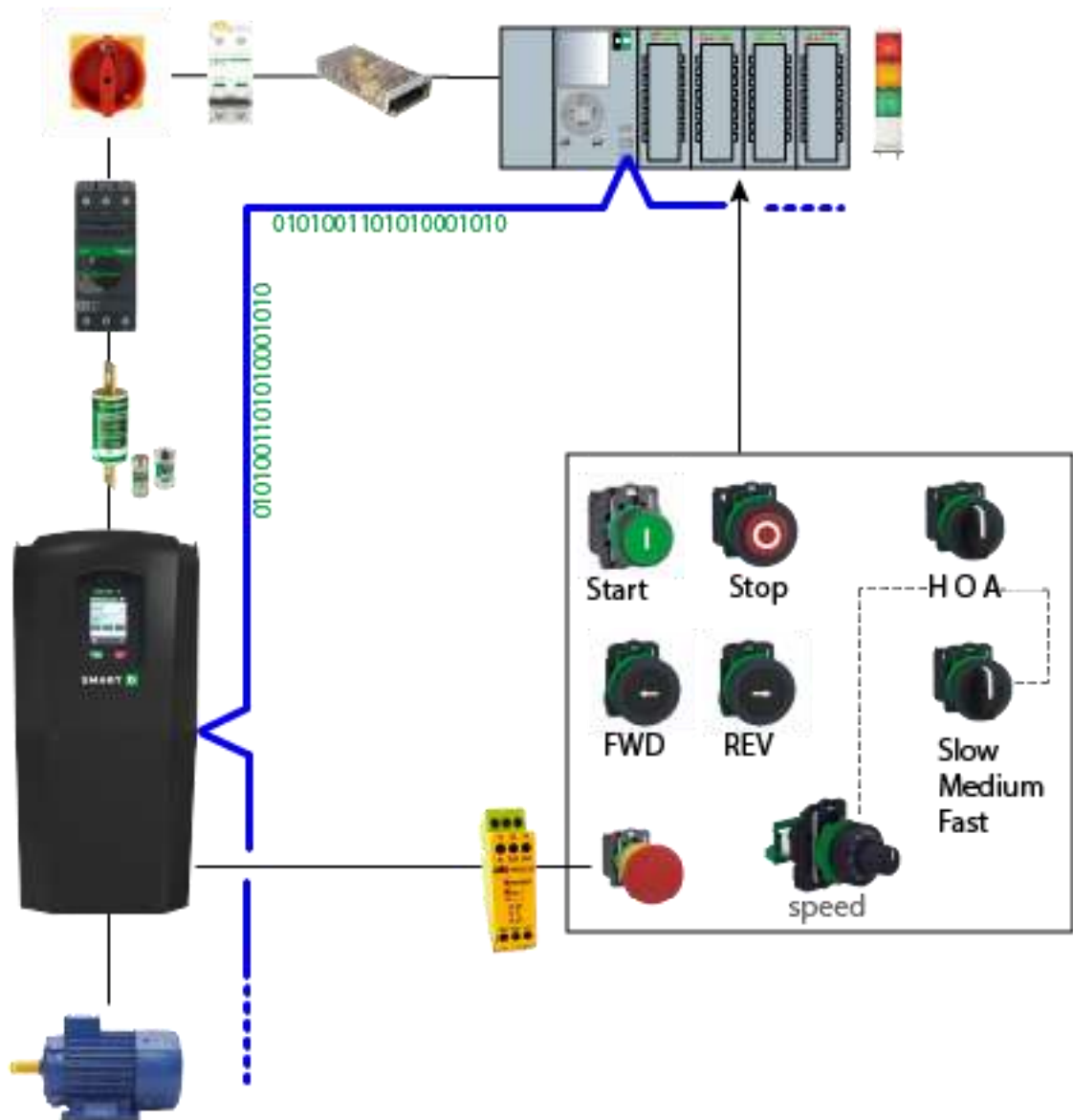
Component	The usual rule of selection	Benefits of VFD using SmartD VFD
Control and interface	• Switch 3 pos: man/auto/off • Switch 3 pos: speed selection • Potentiometer manual setpoint. • Pilot lights or stack lights • Push-Button Start motor forward direction • Push-Button Start motor reverser direction • Push-Button stop • Push-Button Reset	
Control cables	Control cables must be selected according to the control voltage and the local and international regulations. They should be shielded to increase noise resistance of equipment and connected at the ground at both ends.	For optimal protection of low-voltage digital signals, double-shielded cable is the preferred choice of SmartD. However, single-shielded twisted pair cable can also be a suitable option.
Ethernet Cable	For most PLC to VFD Modbus TCP applications, a Cat5e or Cat6 shielded Ethernet cable with an industrial-grade jacket and high-quality connectors is recommended. Ensure the cable is suitable for the environmental conditions and complies with relevant standards for reliable performance. • Cat5e (Category 5e): Suitable for most industrial applications, supporting up to 1 Gbps speeds and a frequency of up to 100 MHz. • Cat6 (Category 6): Provides higher performance with support for up to 10 Gbps speeds and a frequency of up to 250 MHz, which can be beneficial in environments with high electromagnetic interference (EMI).	

Component	The usual rule of selection	Benefits of VFD using SmartD VFD
	Shielded Twisted Pair (STP): Provides additional protection against electromagnetic interference (EMI) and radio frequency interference (RFI). This is especially important in industrial environments with high levels of electrical noise. Industrial-Grade Jacket: Look for cables with industrial-grade jackets resistant to oil, chemicals, and extreme temperatures. Cable Length Limits: Standard Ethernet cables can support distances up to 100 meters (328 feet). For longer distances, consider using Ethernet switches or fiber optic cables. Use high-quality, industrial-grade RJ45 connectors. Ensure they are properly crimped and provide a secure connection. Ensure the cable is rated for the operating temperature range of the environment where it will be used. Ensure the cable meets relevant industry standards such as TIA/EIA, ISO/IEC, and IEEE for Ethernet performance and safety.	
Power cables	copper wire. Select the section according to the local and international regulations. The cable's insulation should be designed to withstand the transients produced by the Pulse Width Modulation (PWM). The cable should have three symmetrical insulated grounding conductors. Because the introduction of additional harmonic currents in the equipment grounding circuits can lead to premature bearing failures.	Clean Power VFDs offer a pure sine wave output, reducing harmonic distortion and enhancing equipment longevity. Key features include: • Filter-Free Operation: Eliminates the need for costly filters, reducing expenses and maintenance requirements. • Standard Cable: Compatibility: Allows the use of regular motor cables, lowering initial costs and simplifying installation. • Harmonic Reduction: With ultra-low harmonics on both input and output, they reduce electrical noise and improve energy efficiency. The active front end ensures near-unity power factor, further boosting efficiency and cost-effectiveness. • Shield: it is highly recommended to use shielded cables to protect nearby low-voltage and communications cables from electromagnetic fields generated by the power signal. It must be properly ground, like all parts: motor, VFD ... including control and panels.

Component	The usual rule of selection	Benefits of VFD using SmartD VFD
Power cables length	High voltage spikes increase with cable length, making longer cable runs more prone to premature insulation failure. If the cable run is greater than 300 feet (about 100 m), special considerations may be needed	Clean Power VFDs offer a pure sine wave output, reducing harmonic distortion and enhancing equipment longevity. Key features include Extended Cable Length: These VFDs prevent energy accumulation in cables, limiting you only by voltage drop.

Connect the Hardware - VFD & PLC using Modbus TCP

Architecture layout



The control panel's main function is to define the operating mode of the system:

Automatic, manual, or off.

In automatic mode, the directions buttons, and the stop and start buttons command the motor, while the selection between Slow, Medium, and Fast defines the speed.

In manual mode, the directions buttons, and the stop and start buttons command the motor, while a potentiometer defines the speed setpoint.

Depending on the required safety level of the system, an emergency stop push-button can be added to perform a stop category 0.

The VFD is protected by fuses (check the user guide recommendation) and a mechanical circuit breaker.

A power main switch powers the whole system on and off.

The PLC is powered by a 24 VDC auxiliary supply that is also used to deliver 24 VDC to the VFD controls and the peripheral automation components.

Wiring instruction

Grounding

The Clean Power VFD must be connected to the earth ground of the AC supply.

Ensure that grounding wires meet the technical standards for electrical equipment.

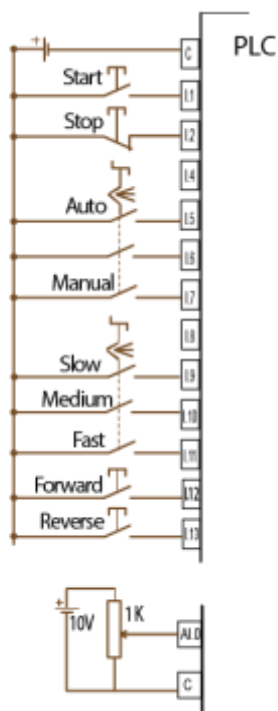
Impedance must conform to national and local regulations, safety regulations, and electrical codes

Furthermore, ensure that the length of grounding wires is minimized.

Power connections:

Follow the [user guide recommendations](#)

Control connections:



Configure the PLC - VFD & PLC using Modbus TCP

[Refer to the Clean Power VFD user guide for full information](#)

You must set several key parameters to configure a PLC for Modbus TCP communication. Here's a general guideline that can be applied to most PLCs:

1- Network Settings:

- **IP Address:** Assign a unique IP address to the PLC.
- **Subnet Mask:** Set the subnet mask to define the network segment.
- **Gateway:** Configure the gateway address if the PLC needs to communicate across different network segments.

2- Modbus TCP Configuration:

- **Enable Modbus TCP:** Ensure that Modbus TCP is enabled in the PLC settings.
- **Port Number:** The default port for Modbus TCP is 502, but it can be changed if necessary.
- **Unit Identifier (Slave ID):** Set the Modbus unit ID to 1.

3- Communication Parameters:

- **Supported Function Codes:**

Code	Function
03	Read holding registers.
06	Write a single register.
16	Write multiple registers
23	Read/Write multiple registers.
43	Read Device Identification

- **Register Addresses:** Parameters, Enumerations, and registers are detailed on the downloadable file: [SmartD Modbus communication parameters](#)

4- Connection Management:

- **Connection Type:** Define the connection type and client or server mode.
- **Timeouts and Retries:** Set communication timeouts and retry limits for robust operation.

5- Error Handling:

- **Status Monitoring:** Implement monitoring of communication status and error codes to handle issues such as lost connections or invalid responses.

Configure the VFD - VFD & PLC using Modbus TCP

Use the VFD's user interface (Mobile App / HMI)

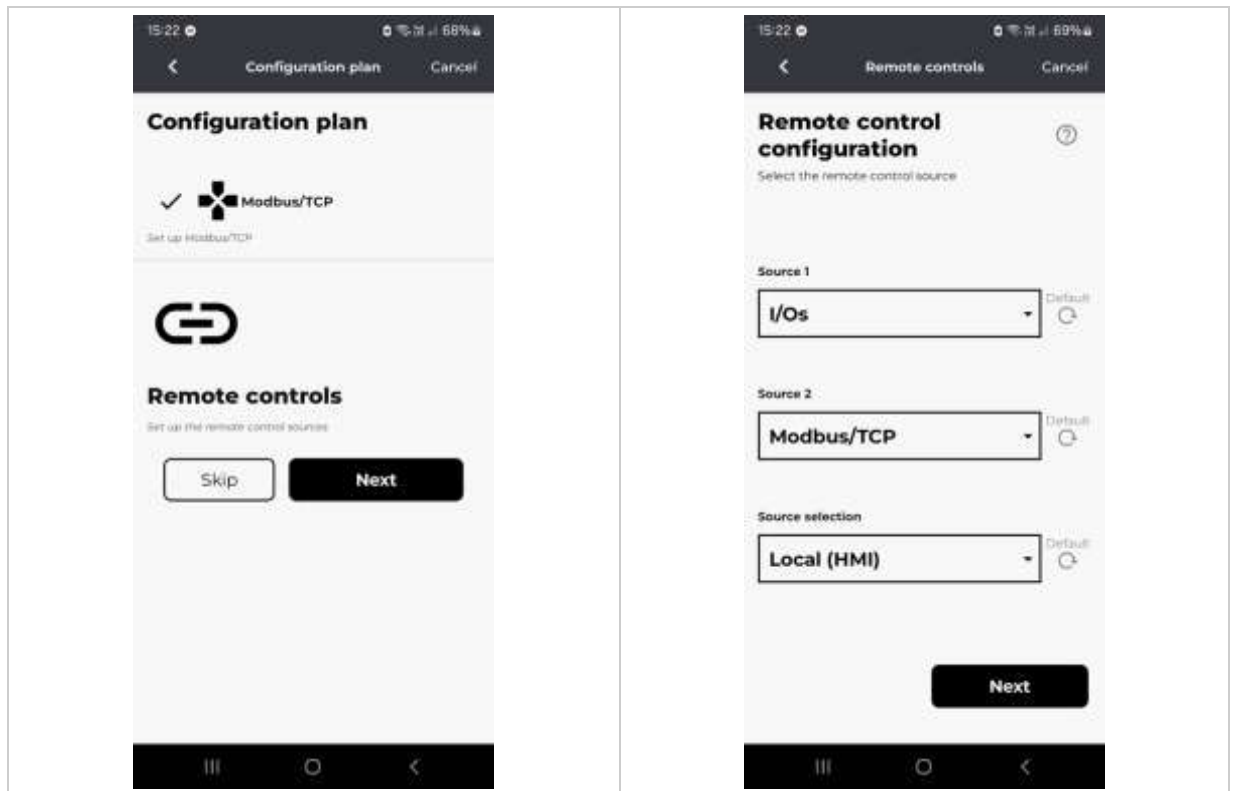
[Refer to the Clean Power VFD User Guide for full information](#)

1- Ensure Modbus communication is enabled using the mobile app. Note that disabling Modbus prevents remote control from Modbus.

Access to Modbus TCP settings	Select LAN	Enable Modbus TCP

2- Configure the VFD to switch between remote and local control.

Access to remote controls settings	Select Modbus TCP as Source2 and keep local HMI to switch between sources
------------------------------------	---



The communication speed is set automatically.

Note that for your PLC setting:

- MAC Address: Find it on the product label.
- Mode: Use DHCP only for network settings.
- Device ID: The VFD accepts requests with slave ID 0x01.

With Schneider Electric M221

This topic explains how to configure a Modicon M221 PLC as a Modbus TCP client to control a SmartD Clean Power VFD.

The example uses the M221 Modbus TCP IOScanner to exchange WORD registers with the VFD.

The application shows how to:

- send a motor start and stop command;
- select forward or reverse direction;
- write a frequency reference;
- read motor state, direction, and actual frequency;
- read active alarm flags;
- poll the SmartD heartbeat register.

The following registers are value-based WORD registers. They must be written or decoded as complete register values. Do not set or read individual bits for these registers.

Function	SmartD register	Address	Decimal offset	Type	M221 object example
Motor state feedback	OperationalData.motor_state	16#0400	1024	WORD (Enumeration) - MCS	%IWN300.0.0
Direction feedback	OperationalData.direction	16#0401	1025	WORD (Enumeration) - MCD	%IWN300.0.1
Actual frequency feedback	OperationalData.frequency	16#0402	1026	Unsigned16, factor 100	%IWN300.0.2
Motor command	SetPoints.command	16#0410	1040	WORD (Enumeration) - MCC	%QWN300.0.0
Direction command	SetPoints.direction	16#0411	1041	WORD (Enumeration) - MCD	%QWN300.0.1
Frequency reference	SetPoints.frequency	16#0412	1042	Unsigned16, factor 100	%QWN300.0.2

IMPORTANT: In the M221 IOScanner, %QWN objects are output register values sent by the PLC to the VFD. %IWN objects are input register values received by the PLC from the VFD. These are WORD objects, not coils.

Command and status values used in this example of implementation

Enumeration	Value	Meaning	Use
MCC	0	Unspecified	No command specified. Do not use as a stop command.
MCC	1	On	Start motor / request motor on.
MCC	2	Off	Stop motor / request motor off.
MCD	0	Unspecified	No direction specified.

MCD	1	Forward	Forward rotation request or feedback.
MCD	2	Reverse	Reverse rotation request or feedback.
MCS	2	Halted	Motor halted feedback.
MCS	3	Ramping Up	Motor is accelerating.
MCS	4	Running	Motor running in steady state.
MCS	5	Ramping Down	Motor is decelerating.

NOTE: MCS includes additional values such as Disabled, Autotuning, Paused, Magnetizing, and Starting. Add those values to your PLC logic if the application must display or react to them.

`%IWN300.channel.word` // WORD values received from the VFD

`%QWN300.channel.word` // WORD values sent to the VFD

NOTE: If your project uses another device ID or another channel number, adapt the %IWN and %QWN addresses. Do not change the SmartD register values or the value-based decoding rules.

Configure the M221 Ethernet port

1. Open the project in EcoStruxure Machine Expert – Basic.
2. Open the Configuration tab.
3. Select the M221 Ethernet port, ETH1.
4. Assign an IP address, subnet mask, and gateway address suitable for the VFD network.
5. Keep the PLC and the VFD in the same subnet unless the network design intentionally uses routing.
6. Apply the configuration.

Enable Modbus TCP IOScanner and add the VFD

1. In the Configuration tab, expand ETH1.
2. Open Modbus TCP.
3. Select Client mode.
4. Enable Modbus TCP IOScanner.
5. Add a Generic remote device.
6. Enter the VFD IP address.
7. Keep the Unit ID used by your existing SmartD project or by the VFD configuration. If no special Unit ID is specified, keep the M221 default used in the project.
8. Set an appropriate response timeout for the network. A typical first value is 10 x 100 ms = 1 s.

Configure IOScanner channels

Use separate channels for the cyclic motor command/status exchange, alarm reading, alarm acknowledgement, and heartbeat. This makes the logic easier to troubleshoot.

Channel	Purpose	Message type	Trigger	Read offset / length	Write offset / length	M221 objects
0	Motor command and feedback	Read/Write multiple words - Modbus 0x17	Cyclic, for example 20 x 10 ms = 200 ms	1024 / 3	1040 / 3	%IWN300.0.0..2 %QWN300.0.0..2
1	Active alarm flags	Read multiple words - Modbus 0x03	Cyclic, for example 50 x 10 ms = 500 ms	3072 / 5	Not used	%IWN300.1.0..4

2	Alarm acknowledgement flags	Write multiple words - Modbus 0x10	Rising edge, for example %M10	Not used	3104 / 5	%QWN300.2.0..4
3	Heartbeat	Read multiple words - Modbus 0x03	Cyclic, for example 200 x 10 ms = 2 s	65283 / 1	Not used	%IWN300.3.0

IMPORTANT: The offsets above are decimal offsets. They correspond to the hexadecimal addresses from the SmartD spreadsheet: 16#0400, 16#0410, 16#0C00, 16#0C20, and 16#FF03.

Channel 0 - motor command and feedback

M221 object	Direction	SmartD register	Address	Offset	Data type	Meaning
%IWN300.0.0	VFD to PLC	OperationalData.motor_state	16#0400	1024	WORD enum MCS	Motor state feedback
%IWN300.0.1	VFD to PLC	OperationalData.direction	16#0401	1025	WORD enum MCD	Actual direction feedback
%IWN300.0.2	VFD to PLC	OperationalData.frequency	16#0402	1026	Unsigned16, x100	Actual frequency feedback
%QWN300.0.0	PLC to VFD	SetPoints.command	16#0410	1040	WORD enum MCC	Motor command
%QWN300.0.1	PLC to VFD	SetPoints.direction	16#0411	1041	WORD enum MCD	Direction command
%QWN300.0.2	PLC to VFD	SetPoints.frequency	16#0412	1042	Unsigned16, x100	Frequency reference

Channel - active alarm flags

M221 object	SmartD register	Address	Offset	Type	Meaning
%IWN300.1.0	OperationalData.activeFlags #1	16#0C00	3072	FLAGS AL1	Active alarm flags, group 1
%IWN300.1.1	OperationalData.activeFlags #2	16#0C01	3073	FLAGS AL2	Active alarm flags, group 2
%IWN300.1.2	OperationalData.activeFlags #3	16#0C02	3074	FLAGS AL3	Active alarm flags, group 3
%IWN300.1.3	OperationalData.activeFlags #4	16#0C03	3075	FLAGS AL4	Active alarm flags, group 4
%IWN300.1.4	OperationalData.activeFlags #5	16#0C04	3076	FLAGS AL5	Active alarm flags, group 5

Channel - heartbeat

M221 object	SmartD register	Address	Offset	Type	Meaning
%IWN300.3.0	Heartbeat	16#FF03	65283	Unsigned16	Read the heartbeat counter and reset the communication-loss watchdog.

Recommended symbols

Create symbols for the I/O Scanner objects before writing the ladder logic. This avoids hard-to-read code and reduces the risk of mixing up channel numbers.

Symbol	Address	Comment
VFD_MotorState	%IWN300.0.0	Motor state feedback - MCS value
VFD_DirectionFb	%IWN300.0.1	Actual direction feedback - MCD value
VFD_FreqFb_x100	%IWN300.0.2	Actual frequency multiplied by 100

VFD_Command	%QWN300.0.0	Motor command - MCC value
VFD_DirectionCmd	%QWN300.0.1	Direction command - MCD value
VFD_FreqRef_x100	%QWN300.0.2	Frequency reference multiplied by 100
VFD_AL1_Active	%IWN300.1.0	Active alarm flags AL1
VFD_AL2_Active	%IWN300.1.1	Active alarm flags AL2
VFD_AL3_Active	%IWN300.1.2	Active alarm flags AL3
VFD_AL4_Active	%IWN300.1.3	Active alarm flags AL4
VFD_AL5_Active	%IWN300.1.4	Active alarm flags AL5
VFD_Heartbeat	%IWN300.3.0	Heartbeat value read from the VFD

Write motor commands

IMPORTANT: Write the direction and frequency reference before writing the On command. This avoids sending an On request with an old direction or old speed reference.

Start forward

To start the motor in forward direction at 30.00 Hz, write the following complete WORD values:

```
VFD_DirectionCmd := 1 // MCD = Forward
VFD_FreqRef_x100 := 3000 // 30.00 Hz x 100
VFD_Command := 1 // MCC = On
```

Start reverse

To start the motor in reverse direction at 30.00 Hz, write the following complete WORD values:

```
VFD_DirectionCmd := 2 // MCD = Reverse
VFD_FreqRef_x100 := 3000 // 30.00 Hz x 100
VFD_Command := 1 // MCC = On
```

Stop the motor

To stop the motor, write value 2 to the command register:

```
VFD_Command := 2 // MCC = Off
```

Decode motor feedback

Decode feedback by comparing the complete WORD value. Do not test individual bits for the motor state or direction feedback registers.

Condition	Correct comparison	Meaning
Motor halted	VFD_MotorState = 2	MCS = Halted
Motor ramping up	VFD_MotorState = 3	MCS = Ramping Up
Motor running	VFD_MotorState = 4	MCS = Running
Motor ramping down	VFD_MotorState = 5	MCS = Ramping Down
Direction forward	VFD_DirectionFb = 1	MCD = Forward
Direction reverse	VFD_DirectionFb = 2	MCD = Reverse

```
Motor_Running := (VFD_MotorState = 4)
Motor_RampingUp := (VFD_MotorState = 3)
Motor_RampingDown := (VFD_MotorState = 5)
Motor_Halted := (VFD_MotorState = 2)
```

Scale frequency values

The SmartD frequency registers in this example use a multiplier factor of 100. This means the register value is the frequency in Hz multiplied by 100.

Frequency	Register value
30.00 Hz	3000
45.00 Hz	4500
60.00 Hz	6000

FrequencyReference_x100 := FrequencyReference_Hz x 100

ActualFrequency_Hz := VFD_FreqFb_x100 / 100

Decode alarm flags

Alarm registers are different from the command and status registers. The SmartD alarm registers are defined as FLAGS (Unsigned16). These registers are true bit fields. For these registers only, decode the individual bits.

Alarm group	M221 object	Example bit	Example meaning
AL1	VFD_AL1_Active	Bit 1	STO
AL1	VFD_AL1_Active	Bit 2	STO Loss
AL1	VFD_AL1_Active	Bit 5	DC Overvoltage
AL1	VFD_AL1_Active	Bit 13	Motor overcurrent
AL1	VFD_AL1_Active	Bit 14	Motor overtemp
AL1	VFD_AL1_Active	Bit 15	Motor Overload
AL2	VFD_AL2_Active	Bit 8	Cooling fan 1 fault
AL2	VFD_AL2_Active	Bit 9	Cooling fan 2 fault
AL3	VFD_AL3_Active	Bit 1	Simultaneous forward and reverse command
AL3	VFD_AL3_Active	Bit 10	AFE Fault

For a FLAGS register, bit n corresponds to mask 2^n . For example:

Bit	Decimal mask	Hex mask
1	2	16#0002
5	32	16#0020
10	1024	16#0400
13	8192	16#2000
14	16384	16#4000
15	32768	16#8000

Poll the heartbeat register

The SmartD heartbeat register is located at 16#FF03 / decimal offset 65283. In this example, the M221 reads it cyclically every 2 seconds through channel 3.

Channel 3:

Message type: Read multiple words – Modbus 0x03

Trigger: Cyclic

Cycle time: 200 x 10 ms = 2 s

Read offset: 65283

Read length: 1

M221 object: %IWN300.3.0

IMPORTANT: Keep the heartbeat poll active during normal remote-control operation. The heartbeat read resets the communication-loss watchdog according to the SmartD Modbus spreadsheet.

A generic Structured Text program

```
PROGRAM Main
```

```
VAR CONSTANT
```

```
// SmartD Modbus register offsets - decimal holding register offsets
```

```
// Feedback registers: OperationalData.*
```

```
REG_READ_START : UINT := 1024; // 16#0400
```

```
REG_MOTOR_STATE : UINT := 1024; // OperationalData.motor_state
```

```
REG_ACTUAL_DIRECTION : UINT := 1025; // OperationalData.direction
```

```
REG_ACTUAL_FREQUENCY : UINT := 1026; // OperationalData.frequency, Hz x 100
```

```
// Command registers: SetPoints.*
```

```
REG_WRITE_START : UINT := 1040; // 16#0410
```

```
REG_COMMAND : UINT := 1040; // SetPoints.command
```

```
REG_DIRECTION : UINT := 1041; // SetPoints.direction
```

```
REG_FREQUENCY : UINT := 1042; // SetPoints.frequency, Hz x 100
```

```
// MCC - Motor command enumeration
```

```
MCC_UNSPECIFIED : WORD := 0;
```

```
MCC_ON : WORD := 1; // Start motor
```

```
MCC_OFF : WORD := 2; // Stop motor
```

```
// MCD - Motor direction enumeration
```

```
MCD_UNSPECIFIED : WORD := 0;
```

```
MCD_FORWARD : WORD := 1;
```

```
MCD_REVERSE : WORD := 2;
```

```
// MCS - Motor state enumeration
```

```
MCS_UNSPECIFIED : WORD := 0;
```

```
MCS_DISABLED : WORD := 1;
```

```
MCS_HALTED : WORD := 2;
```

```
MCS_RAMPING_UP : WORD := 3;
```

```
MCS_RUNNING : WORD := 4;
```

```
MCS_RAMPING_DOWN : WORD := 5;
```

```
MCS_AUTOTUNING : WORD := 6;
```

```
MCS_PAUSED : WORD := 7;
```

```
MCS_MAGNETIZING : WORD := 8;
```

```
MCS_STARTING : WORD := 9;
```

```
END_VAR
```

```
VAR
```

```
// Timer variables
```

```
InitialDelayTimer : TON; // Initial delay before communication starts
```

```
PollTimer : TON; // Cyclic polling timer
```

```
CommunicationEnabled : BOOL;
```

```
PollPulse : BOOL;
```

```
// Modbus communication variables
```

```
ReadVar : READ_VAR;
```

```
WriteVar : WRITE_VAR;
```

```
ReadData : ARRAY [0..2] OF WORD; // 1024, 1025, 1026
```

```
WriteData : ARRAY [0..2] OF WORD; // 1040, 1041, 1042
```

```
ExecuteRead : BOOL;
```

```
ExecuteWrite : BOOL;
```

```
Busy : BOOL;
```

```
ReadActive : BOOL;
```

```
WriteActive : BOOL;
```

```

Done : BOOL;
Error : BOOL;

//VFD feedback values read from OperationalData.*
VFDMotorState : WORD;
VFDActualDirection : WORD;
VFDActualFrequencyX100 : WORD;

// Decoded VFD feedback
VFDDisabled : BOOL;
VFDEHalted : BOOL;
VFDRunningUp : BOOL;
VFDRunning : BOOL;
VFDRunningDown : BOOL;
VFDForward : BOOL;
VFDReverse : BOOL;

// PLC commands
RunButton : BOOL; // Input for run command
StopButton : BOOL; // Input for stop command
ForwardInput : BOOL; // Input for forward direction
ReverseInput : BOOL; // Input for reverse direction

// Command values to write to SetPoints.*
RequestedCommand : WORD;
RequestedDirection : WORD;
SpeedSetpointX100 : WORD; // Frequency reference in Hz x 100

// Change detection for write requests
LastRequestedCommand : WORD := 16#FFFF;
LastRequestedDirection : WORD := 16#FFFF;
LastSpeedSetpointX100 : WORD := 16#FFFF;
WriteRequest : BOOL;
FirstWriteRequired : BOOL := TRUE;

// Optional: mapped memory word for speed setpoint
// Example: 3000 = 30.00 Hz
UserSpeedSetpointX100 AT %MW22 : WORD;
END_VAR

// -----
// Initialization
// -----

IF %S0 THEN
  // First scan after cold start.
  // Keep the drive stopped until the application explicitly receives a run command.
  RequestedCommand := MCC_OFF;
  RequestedDirection := MCD_FORWARD;
  FirstWriteRequired := TRUE;
END_IF;

// -----
// Timer logic
// -----

// Initial delay before starting Modbus communication
InitialDelayTimer(IN := NOT CommunicationEnabled, PT := T#5S);

IF InitialDelayTimer.Q THEN

```

```

CommunicationEnabled := TRUE;
END_IF;

// Generate one polling pulse every 1 second after the initial delay.
// The timer resets itself for one scan when Q becomes TRUE.
PollTimer(IN := CommunicationEnabled AND NOT PollTimer.Q, PT := T#1S);
PollPulse := PollTimer.Q;

// -----
// Build command values from PLC inputs
// -----

// Stop has priority if both Run and Stop are requested.
IF StopButton THEN
  RequestedCommand := MCC_OFF;
ELSIF RunButton THEN
  RequestedCommand := MCC_ON;
END_IF;

// Direction command.
// If both direction inputs are active, do not change the last valid direction.
// If neither direction input is active, keep the last valid direction.
IF ForwardInput AND NOT ReverseInput THEN
  RequestedDirection := MCD_FORWARD;
ELSIF ReverseInput AND NOT ForwardInput THEN
  RequestedDirection := MCD_REVERSE;
END_IF;

// Frequency setpoint.
// The SmartD frequency register uses Hz x 100.
// Example: 30.00 Hz = 3000.
SpeedSetpointX100 := UserSpeedSetpointX100;

// -----
// Detect when a write is required
// -----

WriteRequest :=
  FirstWriteRequired OR
  (RequestedCommand <> LastRequestedCommand) OR
  (RequestedDirection <> LastRequestedDirection) OR
  (SpeedSetpointX100 <> LastSpeedSetpointX100);

// -----
// Start Modbus write transaction
// -----

IF CommunicationEnabled AND WriteRequest AND NOT Busy THEN

  // Register order starting at 1040:
  // WriteData[0] -> 1040 SetPoints.command
  // WriteData[1] -> 1041 SetPoints.direction
  // WriteData[2] -> 1042 SetPoints.frequency

  WriteData[0] := RequestedCommand;
  WriteData[1] := RequestedDirection;
  WriteData[2] := SpeedSetpointX100;

  ExecuteWrite := TRUE;

```

```

Busy := TRUE;
WriteActive := TRUE;

ELSE
ExecuteWrite := FALSE;
END_IF;

// Generic WRITE_VAR call.
// Keep the same block structure as your original example.
// Adapt LINK, ID, OBJTYPE, and parameter names to the PLC library actually used.
WriteVar(
EXECUTE := ExecuteWrite,
LINK := 3, // ETH1 - adapt to PLC configuration
ID := 1, // Modbus server unit ID - adapt if required
TIMEOUT := 100,
OBJTYPE := 16, // Write multiple holding registers - adapt to library
FIRSTOBJ := REG_WRITE_START,
QUANTITY := 3,
DATA := WriteData
);

IF WriteActive AND WriteVar.DONE THEN

LastRequestedCommand := RequestedCommand;
LastRequestedDirection := RequestedDirection;
LastSpeedSetpointX100 := SpeedSetpointX100;

FirstWriteRequired := FALSE;
WriteActive := FALSE;
Busy := FALSE;
Done := TRUE;

END_IF;

IF WriteActive AND WriteVar.ERROR THEN
WriteActive := FALSE;
Busy := FALSE;
Error := TRUE;
END_IF;

// -----
// Start Modbus read transaction
// -----

IF CommunicationEnabled AND PollPulse AND NOT Busy THEN

ExecuteRead := TRUE;
Busy := TRUE;
ReadActive := TRUE;

ELSE
ExecuteRead := FALSE;
END_IF;

// Generic READ_VAR call.
// Read the feedback registers starting at 1024, not the command registers at 1040.
ReadVar(
EXECUTE := ExecuteRead,
LINK := 3, // ETH1 - adapt to PLC configuration

```

```

ID := 1, // Modbus server unit ID - adapt if required
TIMEOUT := 100,
OBJTYPE := 3, // Read holding registers - adapt to library
FIRSTOBJ := REG_READ_START,
QUANTITY := 3,
DATA := ReadData
);

IF ReadActive AND ReadVar.DONE THEN

  // Register order starting at 1024:
  // ReadData[0] -> 1024 OperationalData.motor_state
  // ReadData[1] -> 1025 OperationalData.direction
  // ReadData[2] -> 1026 OperationalData.frequency

  VFDMotorState := ReadData[0];
  VFDActualDirection := ReadData[1];
  VFDActualFrequencyX100 := ReadData[2];

  ReadActive := FALSE;
  Busy := FALSE;
  Done := TRUE;

END_IF;

IF ReadActive AND ReadVar.ERROR THEN
  ReadActive := FALSE;
  Busy := FALSE;
  Error := TRUE;
END_IF;

// -----
// Decode VFD feedback using complete WORD values
// -----

// Do not use VFDMotorState.1, VFDMotorState.2, etc.
// OperationalData.motor_state is a WORD enumeration, not a bit field.

VFDDisabled := (VFDMotorState = MCS_DISABLED);
VFDHalted := (VFDMotorState = MCS_HALTED);
VFD RampingUp := (VFDMotorState = MCS_RAMPING_UP);
VFDRunning := (VFDMotorState = MCS_RUNNING);
VFD RampingDown := (VFDMotorState = MCS_RAMPING_DOWN);

// Direction feedback is also a WORD enumeration.
VFDForward := (VFDActualDirection = MCD_FORWARD);
VFDReverse := (VFDActualDirection = MCD_REVERSE);

// -----
// End of program
// -----

END_PROGRAM

```

Heartbeat register

The Heartbeat register is a communication integrity indicator between a PLC and a VFD within a MODBUS/TCP network.

Its primary purpose is to confirm an active and stable connection by requiring periodic polling.

The Clean Power VFD's Modbus socket will close with an inactivity of 30 seconds.

To keep the TCP socket open, we recommend reading the Heartbeat register regularly, typically every 10 seconds.

If the Heartbeat register is not polled within the recommended time, the VFD will interpret the lack of response as a communication failure, triggering a MODBUS Communication Loss alarm.

Benefits and results – VFD & PLC using Modbus TCP

Creating communication between the VFD and PLC using Modbus TCP offers several significant advantages:

Advantages of Modbus TCP Communication

1. Enhanced Data Exchange:

Rich Data Set: Modbus TCP allows the exchange of more detailed and extensive data between the VFD and PLC. Parameters such as current, voltage, power, speed, and fault diagnostics can be easily monitored and controlled.

Real-time Feedback: Continuous and real-time monitoring of VFD status and performance metrics leads to more accurate and timely adjustments, improving system efficiency and reliability.

2. Simplified Wiring and Reduced Installation Costs:

Reduced Wiring Complexity: Modbus TCP uses standard Ethernet cabling, reducing the amount of wiring compared to traditional I/O connections.

Lower Installation Costs: Less wiring and simplified network infrastructure result in lower installation and commissioning costs.

3. Scalability and Flexibility:

Easier Expansion: Adding new devices to the network is straightforward, making the system scalable without significant changes to the physical wiring.

Flexible Configuration: Modbus TCP allows for flexible network configurations, including star, linear, and ring topologies, making it easier to adapt to various installation environments.

4. Improved Diagnostics and Maintenance:

Advanced Diagnostics: Detailed diagnostic information can be retrieved over the network, enabling proactive maintenance and quicker troubleshooting.

Remote Monitoring and Control: Modbus TCP enables remote access to the VFD for monitoring and control, allowing for quicker response times and reduced need for on-site visits.

5. Integration with Other Systems:

Seamless Integration: Modbus TCP is widely used and supported, facilitating easy integration with other control systems, SCADA systems, and enterprise-level software.

Interoperability: The open standard nature of Modbus TCP ensures compatibility with a wide range of devices from different manufacturers, enhancing interoperability.

6. Data Logging and Analysis:

Enhanced Data Logging: Continuous data logging over the network allows for more comprehensive data analysis and performance tracking.

Historical Data: Historical data can be easily stored and retrieved, aiding in performance optimization and predictive maintenance strategies.

7. Increased Safety and Reliability:

Reduced Physical Points of Failure: With fewer physical connections, the risk of failure points is reduced, increasing system reliability.

Enhanced Safety: Real-time monitoring and control over the network can enhance safety by enabling quick responses to fault conditions and operational anomalies.

In summary, using Modbus TCP communication between the VFD and PLC provides a modern, efficient, and scalable solution that enhances data exchange, reduces installation complexity, and improves overall system performance and reliability.

Conclusion – VFD & PLC using Modbus TCP

This application note has outlined the substantial benefits and enhanced performance capabilities provided by SmartD Technologies' Clean Power VFD when integrated with a Programmable Logic Controller (PLC).

By leveraging advanced control algorithms and maintaining optimal motor function, the Clean Power VFD ensures superior efficiency, reliability, and sustainability in motor control applications.

We encourage industries looking to upgrade or install new motor control systems to consider the Clean Power VFD for its exceptional benefits.

For further information, and detailed specifications, or to initiate an implementation in your operations, please visit our website or contact our support team.

Let SmartD help you achieve operational excellence with cleaner, more efficient power solutions.