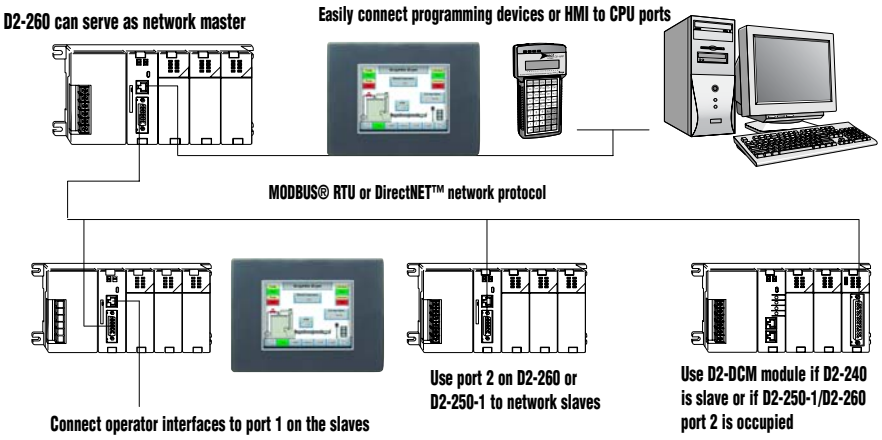
D2-260 Key Features



D2-260: Our most powerful DL205 CPU

Our D2-260 CPU provides all the capabilities of the other DL205 CPUs (as well as our D4-450 CPU), plus several additional features rarely found in a PLC of this size. With such an incredible array of features, you may be able to replace PLCs costing hundreds (or thousands) more.

Release 4.0 or higher of *Direct*SOFT is required to program the D2-260. If you're using a handheld programmer, version 2.10 of the handheld programmer firmware is required. Here are a few key features about the D2-260 CPU:

Local expansion I/O

The D2-260 supports local expansion up to five total bases (one CPU base and four expansion bases). Expansion bases are commonly used when there are not enough slots available in the CPU base, when the base power budget will be exceeded, or when placing an I/O base at a location away from the CPU base (but within the expansion cable limits). All local and expansion I/O points are updated on every CPU scan. Each local expansion base requires the D2-CM module in the CPU slot. The local CPU base requires the D2-EM Expansion Module, as well as each expansion base. For more information on local expansion, refer to the Expansion Modules pages later in this section.

Powerful built-in CPU communications

The D2-260 offers two communications ports that provide a vast array of communication possibilities. The top RJ-12 RS-232 port can be used for connection to a *C-more* or DV-1000 operator interface panel, or as a single K-sequence or *Direct*NET slave. The 15-pin bottom port (port 2) supports RS-232 or RS-422/RS485. This port offers several different protocol options such as:

- K-sequence
- ***Direct***NET Master/Slave
- Modbus RTU Master/Slave
- ASCII In/Out Communications

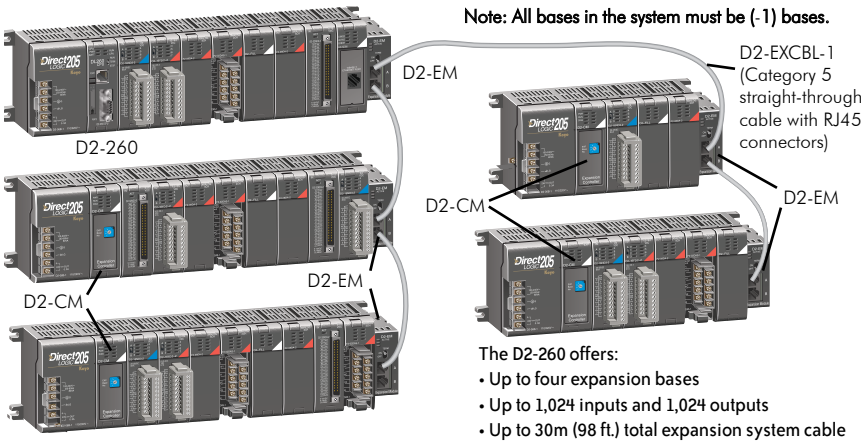
Port 2 can also serve as a remote I/O master. The D2-260 supports the Ethernet Communication module and Data Communication Module for additional communications ports.

16 PID loops with auto-tuning

The D2-260 CPU can process up to 16 PID loops directly in the CPU. You can select from various control modes including automatic, manual, and cascade. There are also a wide variety of alarms including Process Variable, Rate of Change, and Deviation. The loop operation parameters (Process Variable, Setpoint, Setpoint Limits, etc.) are stored in V-memory, which allows easy access from operator interfaces or HMIs. Setup is accomplished with easy-to-use setup menus and monitoring views in *Direct*SOFT programming.

The auto-tuning feature is easy to use and can reduce setup and maintenance time. Basically, the CPU uses the auto- tuning feature to automatically determine near optimum loop settings. See the D2-250-1 CPU section for a PID loop control block diagram.

D2-260 local expansion system



The D2-260 offers:

- Up to four expansion bases
- Up to 1,024 inputs and 1,024 outputs
- Up to 30m (98 ft.) total expansion system cable

D2-260 Key Features

Full array of instructions

The right instruction can greatly simplify your programming task and can save hours of programming time.

The D2-260 supports over 280 powerful instructions, such as:

- Four types of drum sequencers
- Leading / trailing edge triggered one-shots
- Bit-of-word manipulation
- Floating point conversions
- Trigonometric functions
- Table instructions
- ASCII IN/OUT instructions

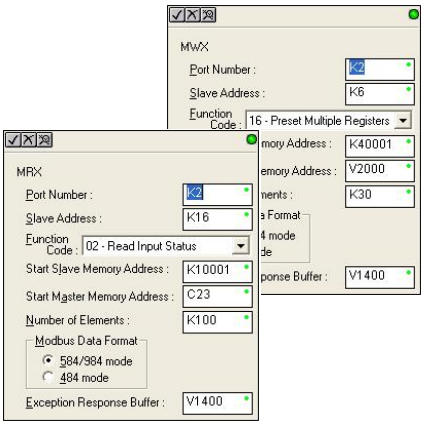
For a complete list of instructions supported by all DL205 CPUs, see the end of this section.

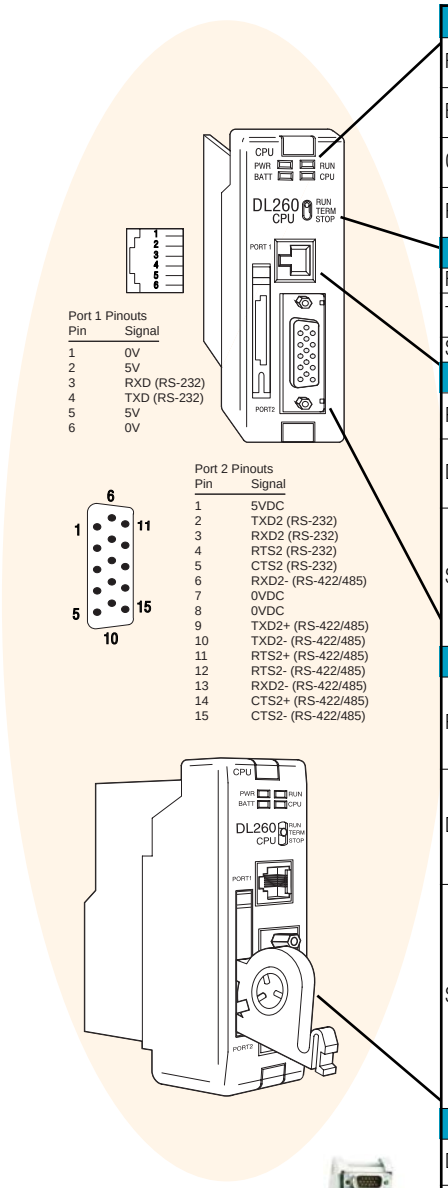
Modbus RTU instructions

The D2-260 CPU supports easy-to-use Modbus Read/Write instructions that expand our existing Modbus network instruction capabilities. The MRX or MWX instructions allow you to enter native Modbus addressing in your ladder program with no need to perform octal-to-decimal conversion. We added Function codes 05, 06 and the ability to read Slave Exception Codes. These flexible instructions allow the user to select the following parameters within one instruction window:

- 584/984 or 484 Modbus data type
- Slave node (0-247)
- Function code
- Modbus starting master / slave memory address
- Number of bits
- Exception code starting address

Examples of MRX and MWX instructions in *Direct*SOFT





Port 1 Pinouts

Pin	Signal
1	0V
2	5V
3	RXD (RS-232)
4	TXD (RS-232)
5	5V
6	0V

Port 2 Pinouts

Pin	Signal
1	5VDC
2	TXD2 (RS-232)
3	RXD2 (RS-232)
4	RTS2 (RS-232)
5	CTS2 (RS-232)
6	RXD2- (RS-422/485)
7	0VDC
8	0VDC
9	TXD2+ (RS-422/485)
10	TXD2- (RS-422/485)
11	RTS2+ (RS-422/485)
12	RTS2- (RS-422/485)
13	RXD2- (RS-422/485)
14	CTS2+ (RS-422/485)
15	CTS2- (RS-422/485)

CPU Status Indicators		
RUN	ON	CPU is in RUN mode
	OFF	CPU is in PROGRAM mode
BATT	ON	Battery backup voltage is low
	OFF	Battery backup voltage is OK or disabled
CPU	ON	CPU internal diagnostics detects error
	OFF	CPU is OK
PWR	ON	CPU power good
	OFF	CPU power failure

Mode Switch	
RUN	Puts CPU into RUN mode
TERM	Allows peripherals (HPP, *Direct*SOFT) to select the mode of operation
STOP	Forces CPU out of RUN mode

Port 1	
Protocols	K-sequence slave, *Direct*NET™ slave, Modbus RTU slave
Devices	Can connect w/HPP, *Direct*SOFT, *C-more*, DV-1000, O/I panels, or any *Direct*NET master
Specs.	6P6C phone jack connector RS-232 9,600 baud Fixed address Odd parity only 8 data bits one start, one stop asynchronous, half-duplex, DTE

Port 2	
Protocols	K-sequence slave, *Direct*NET Master/Slave, Modbus RTU Master/Slave, ASCII IN/OUT, Remote I/O Master
Devices	Can connect w/many devices, such as PCs running *Direct*SOFT, DSData, HMI packages, *C-more*, DV-1000, other O/I panels, any *Direct*NET or Modbus RTU master or slave, or ASCII devices
Specs.	HD15 connector RS-232, RS-422/485* 300/600/1200/2400/4800 9600/19.2K/38.4K baud Odd, even, or no parity Selectable address (1-90, HEX 1 – 5A) 8 data bits, one start, one stop Asynchronous, Half-duplex, DTE

Battery (Optional)	
D2-BAT-1	Coin type, 3.0V Lithium battery, 560mA, battery number CR2354

Note: Batteries are not needed for program backup. However, you should order a battery if you have parameters in V-memory that must be maintained in case of a power outage.
****RS485 for Modbus protocol only***

*ZIP*Link communications adapter modules

DN-15TB

ZIPLink cables and communications adapter modules offer fast and convenient screw terminal connections for the D2-260 bottom port. They are RS-232/422 DIP switch selectable. See the Terminal Blocks and Wiring section in this catalog for ***ZIP***Link part numbers and descriptions.

On-board memory

The D2-260 has 15.5K words of flash memory on board for your program plus 14.2K words of data registers. With flash memory, you don't have to worry about losing the program due to a bad battery.

Built-in remote I/O connection

The bottom port on the D2-260 can be used as a master for serial remote I/O networks (see the D2-RSSS later in this section for details).

D2-260 Key Features

ASCII communications instructions

The D2-260 CPU supports several easy-to-use instructions that allow ASCII strings to be read into and written from the PLC communications ports.

Raw ASCII: Port 2 can be used for either reading or writing raw ASCII strings, but not for both.

Embedded ASCII characters: The D2-260 can decipher ASCII embedded within a supported protocol (K-Sequence, *Direct*Net, Modbus, Ethernet) via the CPU ports, H2-ECOM or D2-DCM.

Here's how the D2-260 can receive ASCII input strings:

1. ASCII IN (AIN) - This instruction configures port 2 for raw ASCII input strings with parameters such as fixed and variable length ASCII strings, termination characters, byte swapping options, and instruction control bits. Use barcode scanners, weight scales, etc. to write raw ASCII input strings into port 2 based on the (AIN) instruction's parameters.
2. Write embedded ASCII strings directly to V-memory from an external HMI or similar master device via a supported communications protocol using the CPU ports, H2-ECOM or D2-DCM. The AIN instruction is not used in this case.
3. If a D2-260 PLC is a master on a network, the Network Read instruction (RX) can be used to read embedded ASCII data from a slave device via a supported communications protocol using port 2, H2-ECOM or D2-DCM. The RX instruction places the data directly into V-memory.

Here's how the D2-260 can write ASCII output strings:

1. Print from V-memory (PRINTV) - Use this instruction to write raw ASCII strings out of port 2 to a display panel or a serial printer, etc. The instruction features the starting V-memory address, string length, byte swapping options, etc. When the instruction's permissive bit is enabled, the string is written to port 2.
2. Print to V-memory (VPRINT) - Use this instruction to create pre-coded ASCII strings in the PLC (i.e. alarm messages). When the instruction's permissive bit is enabled, the message is loaded into a pre-defined V-memory address location. Then the (PRINTV) instruction may be used to write the pre-coded ASCII string out of port 2. American, European and Asian Time/Date stamps are supported.
3. Print Message (PRINT) - This existing instruction can be used to create pre-coded ASCII strings in the PLC. When the instruction's permissive bit is enabled, the string is written to port 2. The VPRINT/PRINTV instruction combination is more powerful and flexible than the PRINT instruction.
4. If a D2-260 PLC is a master on a network, the Network Write instruction (WX) can be used to write embedded ASCII data to an HMI or slave device directly from V-memory via a supported communications protocol using port 2, H2-ECOM or D2-DCM.

Additional instructions that help manage the ASCII strings

The following instructions can be very helpful in managing the ASCII strings within the CPU's V-memory:

ASCII Find (AFIND) - Finds where a specific portion of the ASCII string is located in continuous V-memory addresses. Forward and reverse searches are supported.

ASCII Extract (AEX) - Extracts a specific portion (usually some data value) from the ASCII find location or other known ASCII data location.

Compare V-memory (CMPV) - This instruction is used to compare two blocks of V-memory addresses and is usually used to detect a change in an ASCII string. Compared data types must be of the same format (i.e. BCD, ASCII, etc.).

Swap Bytes (SWAPB) - Usually used to swap V-memory bytes on ASCII data that was written directly to V-memory from an external HMI or similar master device via a communications protocol. The AIN and AEX instructions have a built-in byte swap feature.

Examples of AIN and VPRINT instructions in *Direct*SOFT