Introduction to Serial Communications



In this Appendix

Introduction to Serial Communications K–2

Introduction to Serial Communications

*Direct*LOGIC® PLCs have two built-in serial communication ports which can be used to communicate to other PLCs or to other serial devices. In order to fully understand the capabilities and limitations of the serial ports, a brief introduction to serial communications is in order.

There are three major components to any serial communications setup:

- Wiring standard
- Communications protocol
- Communications parameters

Each of these will be discussed in more detail as they apply to *Direct*LOGIC PLCs.

Wiring Standards

There are three different wiring standards that can be used with most of the *Direct*LOGIC PLCs: RS-232C, RS-422 and RS-485. DL05 PLCs only support RS-232C, although RS-422 can be accomplished by using converters, such as the FA-ISOCON.

RS-232C is a point-to-point wiring standard with a practical wiring distance of 15 meters, or 50 feet, maximum. This means that only two devices can communicate on an RS-232C network, a single master device and a single slave device, and the total cable length cannot exceed 50 feet. AutomationDirect L19772 cable (Belden® 8102), or equivalent, is recommended for RS-232C networks.

Ports 1 and 2 on the DL05 use RJ12 phone type connectors (see pages 4-4 and 4-5 for the cable connections).

Communications Protocols

A communications protocol is the 'language' the devices on a network use to communicate with each other. All the devices on the network must use the same communications protocol in order to be able to communicate with each other. The protocols avaliable in the *Direct*LOGIC DL05 PLCs are listed in the following table.

DL05 Communications Protocols							
Protocol	**Master**	**Slave**	**Port 1***	**Port 2**	**RS-232C**	**RS-422**	**RS-485**
K-Sequence	No	Yes	Yes	Yes	Yes	No	No
***Direct*NET**	Yes	Yes	Yes	Yes	Yes	Yes**	No
Modbus RTU	Yes	Yes	Yes	Yes	Yes	Yes**	No
ASCII	Out	In	No	Yes	Yes	Yes**	No

* Port 1 supports slave only and is only RS-232C with fixed communications parameters of 9600 baud, 8 data bits, 1 start bit, 1 stop bit, odd parity and station address 1. It is an asynchronous, half-duplex DTE port and auto-selects between K-Sequence, *Direct*NET and Modbus RTU protocols.

** RS-422 is available on Port 2 using an RS-422 converter such as the FA-ISOCON.

K-Sequence protocol is not available for use by a master DL05 PLC. Therefore, it cannot be used for networking between PLCs. Its primary use in the DL05 PLC is as a slave to *Direct*SOFT programming software and to an operator interface.

***Direct*NET** protocol is available for use by a master or by a slave DL05 PLC. This, and the fact that it is 'native' protocol, makes it ideal for PLC-to-PLC communication over a point-to-point or multipoint network using the RX and WX instructions.

Modbus RTU protocol is a very common industry standard protocol, and can be used by a master or slave DL05 to communicate with a wide variety of industrial devices which support this protocol.

ASCII is another very common industry standard protocol, and is commonly used where alpha-numeric character data is to be transferred. Many input devices, such as barcode readers and electronic scales, use ASCII protocol. Many output devices accept ASCII commands as well.

No matter which wiring standard or protocol is used, there are several communications parameters to select for each device before it will be able to communicate. These parameters include:

Baud Rate	Flow Control
Data Bits	Echo Suppression
Parity	Timeouts
Stop Bits	Delay Times
Station Address	Format

All of these parameters may not be necessary, or available, for your application. The parameters used will depend on the protocol being used and whether the device is a master or slave.



***NOTE:** An important point to remember is that when the same parameter is available in the master and in the slave (i.e. Baud Rate, Parity, Stop Bits, etc), the settings must match.*

DL05 Port Specifications

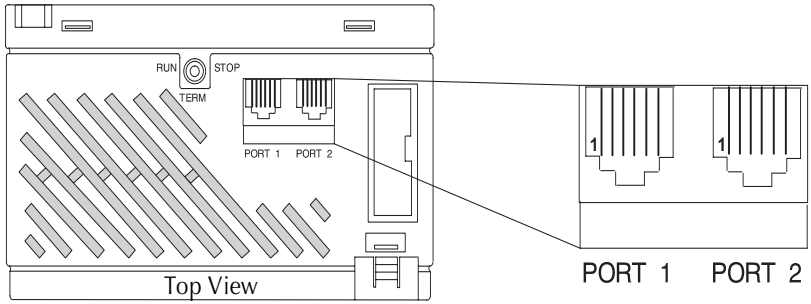
Communications Port 1	
Port 1	Connects to HPP, *Direct*SOFT 5, operator interfaces, etc.
	6-pin, RS232C
	Communication speed (baud): 9600 (fixed)
	Parity: odd (fixed)
	Station Address: 1 (fixed)
	8 data bits
	1 start, 1 stop bit
	Asynchronous, half-duplex, DTE
	Protocol (auto-select): K-sequence (slave only), *Direct*NET (slave only), Modbus RTU (slave only)

Communications Port 2	
Port 2	Connects to HPP, *Direct*SOFT 5, operator interfaces, etc.
	6-pin, multifunction port, RS232C, RS422, RS485 (RS485 with 2-wire is only available for Modbus and Non-sequence).
	Communication speed (baud): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400
	Parity: odd (default), even, none
	Station Address: 1 (default)
	8 data bits
	1 start, 1 stop bit
	Asynchronous, half-duplex, DTE
	Protocol (auto-select): K-sequence (slave only), *Direct*NET (master/slave), Modbus RTU (master/slave), non-sequence/print

DL05 Port Pinouts

Port 1 Pin Descriptions		
1	0V	Power (-) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive data (RS-232C)
4	TXD	Transmit data (RS-232C)
5	5V	Power (+) connection
6	0V	Power (-) connection (GND)

Port 2 Pin Descriptions		
1	0V	Power (-) connection (GND)
2	5V	Power (+) connection
3	TXD	Transmit data (RS-232C)
4	RXD	Receive data (RS-232C)
5	RTS	Ready to send (RS-232C)
6	CTS	Clear to send (RS232C)



Note that the default configuration for port 2 is:

Auto-detect among K-Sequence, *Direct*NET, and Modbus RTU protocols

Timeout = Base Timeout x 1 (800 ms)

RTS on delay time = 0 ms

RTS off delay time = 0ms

Station Number = 1

Baud rate = 19200

Stop bits = 1

Parity = odd

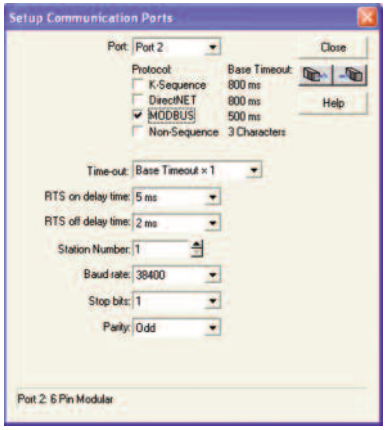
Format = Hex

Port Setup Using *Direct*SOFT or Ladder Logic Instructions

Port 2 on the DL05 can be configured for communications using the various protocols which have been previously mentioned. Also, the communications parameters can be configured to match the parameters in the other device(s) with which the PLC will be communicating. The port may be configured using the *Direct*SOFT PLC programming software, or by using ladder logic within the PLC program. It is important to note that the settings for port 2 are never saved to disk with *Direct*SOFT, so if you are using port 2 in other than its default configuration it is a good idea to include the port setup in the ladder program, typically on a first scan bit, or in an initialization subroutine.



To set up port 2 using *Direct*SOFT, the PLC must be turned on and connected to *Direct*SOFT. With the PLC Setup toolbar displayed, select the **Port 2** button or select **PLC > Setup > Setup Sec. Comm Port...** from the menu bar located at the top of the programming window. A dialog box like the one below will appear. Make the appropriate settings and write them to the PLC.



In order to set up port 2 in relay ladder logic the appropriate values must be written to V7655 (Word 1), V7656 (Word 2) and V7650 (Word 3, for ASCII only) to specify the settings for the port. Then write the 'setup complete' flag (K0500) to V7657 (Word 3) to request the CPU to accept the port settings. Once the CPU sees the 'setup complete' flag in V7657 it will test the port settings which have selected for validity, and then change the value in V7657 according to the results of this test. If the port settings are valid, the CPU will change the value in V7657 to 0A00 ('A' for Accepted). If there was an error in the port settings, the CPU will change the value in V7657 to 0E00 ('E' for Error).



NOTE: *This is a Helpful Hint. Rather than build the setup words manually from the tables, use DirectSOFT to set up the port as desired then use a Dataview to view the setup words as BCD/HEX. Then simply use these numbers in the setup code.*

The data that is written to the port setup words has two formats. The format that is used depends on whether K-Sequence, *Direct*NET, Modbus RTU (method 1) or ASCII (method 2) is selected.

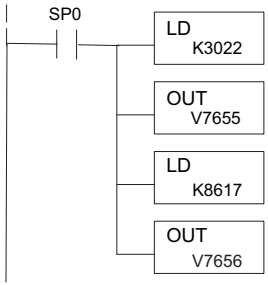
Port 2 Setup for RLL Using K-Sequence, *Direct*NET or Modbus RTU

V7655 (Word 1)	RTS On-delay	Timeout (% of timeout)	Protocol	RTS Off-delay
0yyy 0ttt mmmm mxxx	yyy	ttt	mmmmm	xxx
	000 = 0ms	000 = 100%	10000 = K-Sequence	000 = 0ms
	001 = 2ms	001 = 120%	01000 = *Direct*NET	001 = 2ms
	010 = 5ms	010 = 150%	00100 = Modbus RTU	010 = 5ms
	011 = 10ms	011 = 200%		011 = 10ms
	100 = 20ms	100 = 500%		100 = 20ms
	101 = 50ms	101 = 1000%		101 = 50ms
	110 = 100ms	110 = 2000%		110 = 100ms
	111 = 500ms	111 = 5000%		111 = 500ms

V7656 (Word 2)	Parity	Stop Bits	Baud Rate
K-Sequence, *Direct*NET & Modbus RTU			
pps0 0bbb xaaa aaaa	pp	s	bbb
	00 = None	0 = 1 bit	000 = 300
	10 = Odd	1 = 2 bits	001 = 600
	11 = Even		010 = 1200
			011 = 2400
			100 = 4800
			101 = 9600
			110 = 19200
			111 = 38400

V7656 (Word 2) cont'd	Protocol	Port 2 Address
K-Sequence, *Direct*NET & Modbus RTU	(*Direct*NET)	*Direct*NET and Modbus RTU
pps0 0bbb xaaa aaaa	x	aaaaaaa
	0 = Hex	*Direct*NET: 1-90
	1 = ASCII	Modbus RTU: 1-247

Use the ladder logic shown below to set up port 2 for Modbus protocol for the following: RTS On-delay of 10ms, Base timeout x1, RTS Off-delay of 5ms, Odd parity, 1 Stop bit, 19,200 baud or Station Number 23.



Port 2 Setup for RLL Using ASCII

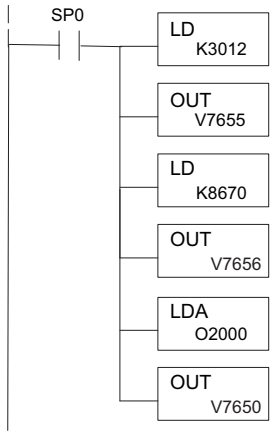
V7655 (Word 1)	RTS On-delay	Timeout (in% of std. timeout)	Protocol	RTS Off-delay
0yyy 0ttt mmmm mxxx	yyy	ttt	mmmmm	xxx
	000 = 0ms	000 = 100%	00010 = Non-Sequence	000 = 0ms
	001 = 2ms	001 = 120%		001 = 2ms
	010 = 5ms	010 = 150%		010 = 5ms
	011 = 10ms	011 = 200%		011 = 10ms
	100 = 20ms	100 = 500%		100 = 20ms
	101 = 50ms	101 = 1000%		101 = 50ms
	110 = 100ms	110 = 2000%		110 = 100ms
	111 = 500ms	111 = 5000%		111 = 500ms

V7656 (Word 2)	Parity	Stop Bits	Baud Rate
K-Sequence, *Direct*NET & Modbus RTU			
pps0 0bbb aaaa aaaa	pp	s	bbb
	00 = None	0 = 1 bit	000 = 300
	10 = Odd	1 = 2 bits	001 = 600
	11 = Even		010 = 1200
			011 = 2400
			100 = 4800
			101 = 9600
			110 = 19200
			111 = 38400

V7656 (Word 2) cont'd	Protocol Mode
K-Sequence, *Direct*NET & Modbus RTU	aaaaaaaa
	01110000 = No flow control
	01110001 = Xon/Xoff flow control
	01110010 = RTS flow control
	01110011 = Xon/Xoff and RTS flow control

V7650 (Word 3)	V-memory address for data
	Hex value of the V-memory location to temporarily store the ASCII data coming into the PLC. Set this parameter to an unused V-memory location which has enough consecutive memory locations free to store the longest string that will come into the PLC.

Use the ladder logic shown below to set up port 2 for Non-sequence (ASCII) communications with the following: RTS On-delay of 10ms, Base timeout x1, RTS Off-delay of 5ms, Odd parity, 1 Stop bit, 19,200 baud, 8 data bits, V-memory buffer starting at V2000 and no flow control.



K-Sequence Communications

The K-Sequence protocol can be used for communication with *Direct*SOFT, an operator interface or any other device that can be a K-Sequence master. The DL05 PLC can be a K-Sequence slave on either port 1 or port 2. The DL05 PLC cannot be a K-Sequence master.

In order to use port 2 for K-Sequence communications you first need to set up the port using either *Direct*SOFT or ladder logic as previously described.

*Direct*NET Communications

The *Direct*NET protocol can be used to communicate to another PLC or to other devices that can use the *Direct*NET protocol. The DL05 can be used as either a master using port 2 or a slave using either port 1 or port 2.

In order to use port 2 for *Direct*NET communications you must first setup the port using either *Direct*SOFT or ladder logic as previously described.

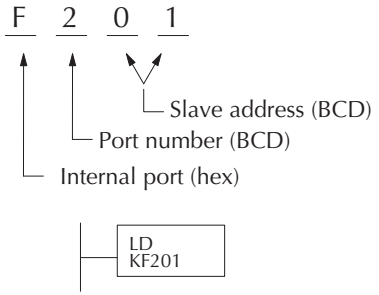
For network slave operation, nothing more needs to be done. Port 2 will function as a slave unless network communications instructions are executed by the ladder logic program.

For a network master operation you will simply need to add some ladder rungs using the network communication instructions RX and/or WX. Only one network communication instruction should be executed at any given time. If you have just a few network communications instructions in your program, you can use discrete bits to interlock them. If you are using many network communications instructions, a counter or a shift register will be a more convenient way to interlock the instructions.

The following step-by-step procedure will provide the information necessary to set up your ladder program to receive data from a network slave.

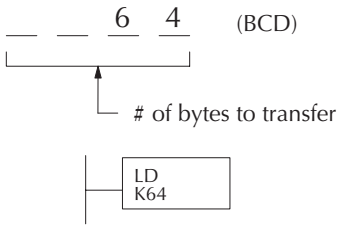
Step 1: Identify Master Port # and Slave

The first Load (LD) instruction identifies the communications port number on the network master (DL05) and the address of the slave station. This instruction can address up to 99 Modbus slaves, or 90 *Direct*NET slaves. The format of the word is shown to the right. The "F2" in the upper byte indicates the use of the the port on the right on the DL05 PLC, port number 2. The lower byte contains the slave address number in BCD (01 to 99).



Step 2: Load Number of Bytes to Transfer

The second Load (LD) instruction determines the number of bytes which will be transferred between the master and slave in the subsequent WX or RX instruction. The value to be loaded is in BCD format (decimal), from 1 to 128 bytes.



The number of bytes specified also depends on the type of data you want to obtain. For example, the DL05 Input points can be accessed by V-memory locations or as X input locations. However, if you only want X0 – X27, you'll have to use the X input data type because the V-memory locations can only be accessed in 2-byte increments. The following table shows the byte ranges for the various types of *Direct*LOGIC products.

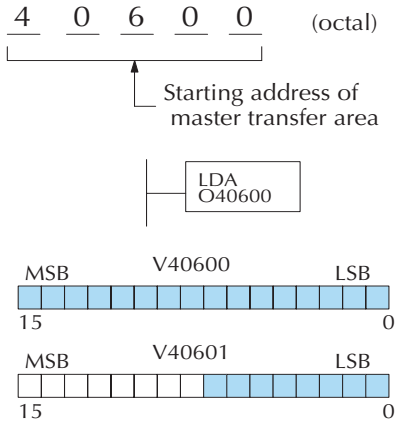
DL05 Memory	Bits per unit	Bytes
V-memory T / C current value	16 16	2 2
Inputs (X, SP)	8	1
Outputs (Y, C, Stage, T/C bits)	8	1
Scratch Pad Memory	8	1
Diagnostic Status	8	1

Step 3: Specify Master Memory Area

The third instruction in the RX or WX sequence is a Load Address (LDA) instruction. Its purpose is to load the starting address of the memory area to be transferred. Entered as an octal number, the LDA instruction converts it to hex and places the result in the accumulator.

For a WX instruction, the DL05 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the DL05 CPU reads the number of bytes previously specified from the slave, placing the received data into its memory area beginning at the LDA address specified.

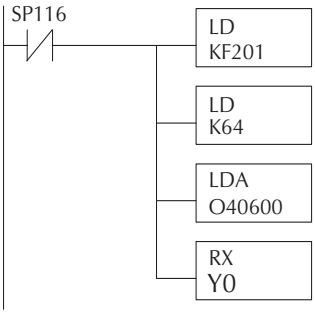


***NOTE**: Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified. The remaining 8 bits are not affected.*

Step 4: Specify Slave Memory Area

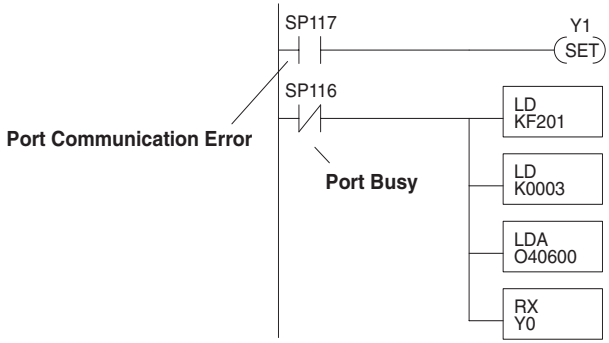
The last instruction in our sequence is the WX or RX instruction itself. Use WX to write to the slave, and RX to read from the slave. All four of our instructions are shown to the right. In the last instruction, you must specify the starting address and a valid data type for the slave.

- ***Direct***NET slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- Modbus DL05 slaves – specify the same address in the WX and RX instruction as the slave's native I/O address



Communications from a Ladder Program

Typically network communications will last longer than 1 scan. The program must wait for the communications to finish before starting the next transaction.



Port 2, which can be a master, has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy"(SP116), and the other indicates "Port Communication Error"(SP117). The example above shows the use of these contacts for a network master that only reads a device (RX). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request.

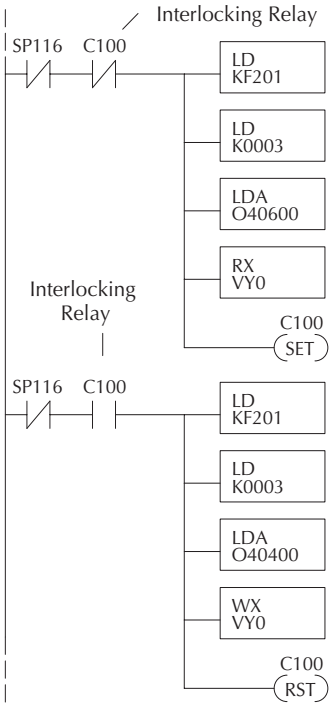
The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

Multiple Read and Write Interlocks

If you are using multiple reads and writes in the RLL program, you have to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time.

In the example to the right, after the RX instruction is executed, C100 is set. When the port has finished the communication task, the second routine is executed and C100 is reset.

If you're using RLL^{PLUS} Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



Modbus RTU Communications

The Modbus RTU protocol can be used for communication with any device that uses the Modbus RTU protocol. The protocol is very common and is probably the closest thing to an "industry standard" protocol in existence. The DL05 can be a Modbus RTU slave on either port 1 or port 2, and it can be a Modbus RTU master on port 2.

In order to use port 2 for Modbus RTU communications you must first set up the port using either *Direct*SOFT or ladder logic as previously described.

For network slave operation, nothing more needs to be done. Port 2 will function as a slave unless network communications instructions are executed by the ladder logic program.

For network master operation the Modbus RTU network communication instructions MRX and/or MWX must be added to the ladder program. If more than one network communication instruction is used, the rungs need to be interlocked to ensure that only one communication instruction is executed at any given time. If only a few network communications instructions are used in your program, discrete bits can be used to interlock them. If many network communications instructions are used, either a counter or a shift register will be a more convenient way to interlock the instructions.