

# PLC Support

## *System additions*

Wed, Nov 6, 2002

Support for PLC hardware has been added to the system code as used in the IRM/PowerPC front end software. This note describes the approach taken and the additions that were necessary for this support.

### ***PLC hardware***

The PLC hardware is interfaced via RS-232 serial or via ethernet. The ethernet support for PLCs allows for communicating with as many as 15 PLCs, which is likely more than may be practical. The serial port interface only supports one such connection. The serial port option is only available for the IRM systems, as the single serial port available on the PowerPC MVME-2401 boards is usurped by vxWorks for debugging uses. The `PLCDirect` protocol is described in some detail by Dave Peterson in his note, `PLC Direct Ethernet protocol for ENET modules`, 4/7/98.

### ***Local application***

The communications needed is managed via a local application. The ethernet support is handled by the `EPLC` local application, and the serial support is handled by the `PLCQ` local application. In either case, the paradigm on which the support is based is via memory within the PLC hardware. One range of memory in the PLC is adopted for analog data words; a second range of memory is adopted for digital data words. The ladder logic in the PLC stores analog data in the first area, and it provides digital status and control in the second area. For example, if there is to be access to a D/A or numeric parameter, it should be found in the analog area, and a setting is made to one of these words. The ladder logic notices this new value and performs whatever changes are implied, including targeting D/A hardware. Similarly, the ladder logic monitors parts of the digital data area for digital data changes, such as a bit being set, and it can operate external hardware, if appropriate. Thus, the scheme is analogous to communication between two CPU boards via through shared memory. Parameters to the local application specify the two areas of PLC memory that are to be used, and they also specify the base analog channel number and base digital Bit number for mapping the two areas into the local data pool. For the ethernet case, the IP address is also specified via the `EPLC` parameters.

To perform data acquisition, periodic communications by the local application with the PLC hardware read out the analog and digital data areas and map this data into the analog and digital data pools of the front end. Also, a message queue is used between the system code and the local application in order to convey setting operations that should be performed. The message queue is monitored by the local application to check whether settings should be done, which merely result in performing the necessary communications to write a word, either analog or digital, to PLC memory.

Take an example. A numeric data word is to be set by writing a value into PLC memory. This amounts to an analog setting operation. The format of the analog control field in the related analog channel descriptor is used to specify that it is a PLC operation, along with which PLC interface is to be used (if more than one is supported), a data type number, and a target 16-bit memory address within the PLC. The analog setting action in this case is to write a message to a message queue. For the ethernet case, the names of the queues are `PLCx`, where `x` ranges from 0–E, allowing for 15 possible PLC “unit numbers.” For the serial case, the name `PLCQ` is used. For each message queue, which is created the first time it is necessary to write to the message queue, a local application instance is required to support its communications. Successfully writing to the message queue is enough to qualify for a successful setting.

On the local application side, the message queue is routinely monitored for records having been placed there. If one is found, the record is formatted into a setting message (memory write) that is sent to the PLC. This should provoke a response from the PLC. But there is no way to get reply status back to the code that requested the setting. One may assure success by the changing the data being set. For the analog case, the values set may be part of the PLC analog data that is periodically delivered to the local analog data pool. For the digital case, the same may be true.

Analog data is generally treated by the system code in units of 2-byte data words. Digital data may be supported by bits, bytes and words. But for the PLC hardware, the unit of setting support is always a 2-byte word. This means that a bit setting, for example, must be turned into a 2-byte word setting. In order to do this, the value of the other 15 bits must be discovered, say by monitoring the last word value in the local data pool that came from the PLC. The user may think of operating a bit, but the communications to the PLC must send a word.

A more complex scenario of support could certainly be designed, such as using a message protocol that the ladder logic would have to interpret. But the shared memory paradigm is an easy one that can bridge the gap between the front end and the PLC that can be helpful during debugging. The PLC software only has to worry about the two areas of its memory into which all of the data it provides is placed, as well as all of the data that it monitors for supporting settings. Copies of this memory can be seen from both sides. PLC memory is analogous to a set of hardware registers.

### ***System additions***

The communications and data acquisition are handled by the local application, such as EPLC, which is separately compiled, downloaded, and enabled to augment the front end system. Setting support is built into the system code. But the communications supported via the EPLC must be used to deliver the setting address and data to the PLC. A message queue conveys the appropriate setting information to the local application. The first time the system code performs a setting to a PLC, it checks for the appropriate message queue. If it is not available, it creates one. Setting parameters are written to this message queue for the local application to find and act upon. Meanwhile, the local application checks whether the appropriate message queue exists. If it does exist, it reads from it, and if a record is found, it formats the information into a setting message and transmits it to the PLC interface. The system code creates the message queue the first time it needs to write something there. The local application reads from the message queue and acts on any setting records it finds there.

System additions required were made to the setting logic that is used for both digital and analog settings. For the analog case, it is handled via a specific analog type code found in the analog descriptor record for the targeted analog channel, where the related addressing information is found. There are three variations for the digital case. One supports digital bit I/O, another supports byte I/O, and the last one supports word (2-byte) I/O. All of these cases result in the PLCQ routine being called, which supports building a record to be written to the message queue. All digital cases rely on the addressing information found in a pair of entries (one for each of the two bytes) in the BADDR (Binary Address) table. For the PLC cases, the pair of entries are equal, each referring to the single digital data word in the PLC memory space.

### ***Addressing information***

Here is an example of the 4-byte analog control field for a PLC-based D/A or other 2-byte value:

```
19 01 0601
```

The 19 specifies the analog control type that refers to PLC-based analog data words. The 01 specifies data type 1. For the ethernet-based PLCs, the upper nibble of this byte specifies the interface in the range 0–E. The 0601 value is the target memory address in the PLC analog data area of memory.

An example of the pair of BADDR entries required to map to a PLC digital data word are:

```
8201 0801    8201 0801
```

The 82 byte signals this is special PLC addressing, where the 01 has the same meaning described for the analog case. The 0801 value is the target memory address in the PLC digital data area. There are two other special values in the high byte that specify special (not local memory) addressing. The value of 80 specifies 1553 addressing, and the value of 81 specifies SRM addressing. The implication is that local memory is not targeted if any of these special values is used in the high byte of a CPU address.

### ***Configuring a front end for access to PLC data***

Choose two areas in the PLC memory that can be used to support communications. Install an instance of the EPLC local application, assuming the ethernet connection, filling in these parameters that include the base addresses in PLC memory and the number of words of analog and digital data that must be accessed. Choose a period between data acquisitions in 15 Hz cycles. Specify the data type number and the PLC “unit number.” Also specify the IP address of the PLC interface. When the local application is enabled, updated information should be found in the front end data pool. To support settings to these PLC data words, install suitable analog control fields in the analog channel descriptors. Install suitable BADDR entries for supporting digital settings.