
THE NEED FOR SPEED

A PAPER BY MIKE GRANBY.

red lion

INTRODUCTION

This paper is designed to provide a few hints and tips on how to get your Edict-97 communications working as quickly as possible. While it should help you get the best out of your system, remember that there are certain hard limits imposed on the performance of any communications system by the protocol definition and other items beyond our control, so don't expect miracles. Instead, follow the suggestions in the following sections, and you'll at least know that your system is operating in an optimized form.

THE FIRST THING

If your system is grossly slow with update times that appear to be in the seconds, the first thing to check is that communications is actually taking place without error. Unfortunately, current Red Lion operator panels do not have a dedicated comms error light, so you will have to place a status text animation item on a display page to show the value of the `CommsError` system variable. If this variable is non-zero, the terminal is experiencing comms errors, and these must be fixed before you even think about speeding things up.

To try and fix comms errors, start with the basis. Do you have the Baud rate set correctly? Is the parity setting correct? Do you have the correct protocol selected? If you are using an Allen-Bradley protocol, are you sure that you have created all the necessary data files in your controller and that you are not reading beyond the end of those files? Remember that if the file is 10 registers in size, the last register that it contains will be N7:9 – something that may be obvious, but nevertheless trips many people up.

If you are using RS-485, look at the red light on the back of the terminal, and make sure that it is not “on” solidly. If it is flashing most of the time, or sitting there in an “off” state and flashing occasionally, you have nothing to worry about, but if the light appears to be “on” for most of the time, you may have to make sure that the line is biased into an inactive state when not driven by any device. Most Red Lion cables include a 1K8 resistor from RxB to GND to achieve just this. If you don't have one, fit one and see what happens. Don't worry about termination resistors, by the way. Unless you are driving huge lengths of cable, the lack of such resistors isn't going to cause any problems, and they'll just complicate matters.

If you can't fix the comms error, telephone Red Lion technical support and explain your problem. They will help you out, and may even send you a simple database that should establish comms between the terminal and your device. Remember, though, that there is no point moving on to the next stage until you have got rid of any comms errors.

GET WITH THE TIMES

Another thing to check is that you have the latest version of the Edict firmware. We have recently added features to optimize the way in which write requests are handled, so that they are given priority over reads. If you find that your PLC writes are too slow in applications such as push-button replacement, contact Red Lion and make sure you're up-to-date.

HOW EDICT WORKS

Before moving on, let us consider how Edict actually moves data around. Unlike many other HMIs, Edict does not read data “on demand” when a display page is selected. Rather, Edict performs communications continually, reading data into what are called “comms blocks” so that it is immediately available when it is requested. Edict has to operate in this way because its support of User Programs and indirect addressing means that it can’t tell ahead of time what data items are going to be required at what point.

Even if you use direct PLC references by placing a PLC register name in square brackets and using it in an expression, Edict is still working via comms blocks. This time, though, a series of automatic blocks are created in the background, with the start address and size of each block being optimized to best access the required data. For reasons detailed below, direct reference can actually provide a simple way to set-up optimized communication, although in theory they achieve nothing that cannot be achieved by using the Comms Block table.

ORGANIZING DATA

Once you realize that Edict transfers data in blocks, it should be obvious that the best way to organize your data in the PLC is to have a single area of registers which Edict can read from the controller in a single request. This block should be as small as possible, which means that the registers in the block should only be those needed by the HMI. A similar block should exist for those values to be written to the PLC, although in reality as writes are much less common than reads, this is somewhat less important.

Of course, for an existing PLC program, you can guarantee that the data that you want to read will be scattered all over the controller’s tables. Although it seems like a pain, it really is worth writing a few rungs of ladder to “marshal” this data into a single block. Your communications will be much faster, and you’ll also find it easier to test your software allowing you view all your comms registers in a single place.

If you are not prepared to do this, or if the PLC program cannot be changed for whatever reason, the trick is now to create the smallest set of blocks that can efficiently read the data that we want to display. As mentioned above, the easiest way to do this is to use Edict’s direct PLC reference feature, whereby a PLC register is directly included in an expression by simply placing the register name in square brackets. Edict will then generate the minimum set of blocks required, and tune the read rates to get best performances.

FOR THE BLOCK HEADS

If you decide to use the Comms Blocks table, the first thing to avoid is to define a single block to cover the whole of the PLC’s address space! Remember that Edict will read all the registers in this block over and over again, killing your comms update times and producing a very slow system. In reality, you’ll only need a small subset of the registers in the PLC, so design your blocks to access those registers with as few reads as possible.

Of course, the process of setting up blocks in a way that optimally reads all the data you want is not easy, which is why direct references often produces a better result. If you do set up

entries in the Comms Blocks table, you will have the advantage of being able to control such things as update rates and so on, but these rarely make up for the trouble required to optimize the communications unless your PLC data is organized so well that a single pair of comms blocks can do the job for you.

When using the Comms Block table, consider setting the Update column for each block to Dynamic. This allows Edict to vary the update rate according to whether or not the data is currently being accessed. The exact way in which this is done can be controlled from the Item Properties dialog box, but please don't fiddle with these values unless you really understand what is going on! Note that the automatic blocks used by the direct reference system always use Dynamic mode, allowing Edict to selectively read data as it is required.

Note that in some version of the Edict firmware, the default settings for Dynamic mode may result in very slow display updates. Remember that since automatic blocks use this mode by default, systems using direct references are also likely to encounter this problem. The issue occurs because Edict misjudges whether or not a block is needed, such that blocks that are required for the current page are mistakenly marked as off-demand. To fix this problem, select the Properties option from the Item menu while in the Comms Blocks window. Enter a value of 5 in the Demand Decay setting, and press OK to save the change. If your comms update time is greater than three seconds, use a value equal to twice this time, instead.

ALL ABOUT BITS

Although not directly connected to comms optimization, it is worth closing by saying something about how Edict writes bits. Some newer drivers directly access bit types, and are capable of turning on or off a single bit within a register. Many drivers, however, only work in 16-bit words, and writing a single bit will in fact write the whole word. This means that if the PLC program has changed other bits in the target word since it was last read into the HMI, those changes may be overwritten. The moral? Only perform bit writes to words which do not contain other bits being changed by the PLC. If this isn't possible, you are going to have to write a couple of rungs of ladder to allow you to target a "clean" data word.

CONCLUSION

If you get your system set up correctly, Edict comms can be fast enough for virtually any application. There are a few holes to fall into however, most notably unrecognized comms errors and trying to read far too much data. If in doubt, display **CommsError** on your terminal, and use direct PLC references to allow Edict to do the optimization for you.