

# SNS PLC Programming Guidelines



A U.S. Department of Energy Multilaboratory Project

SNS

SPALLATION NEUTRON SOURCE

Argonne National Laboratory • Brookhaven National Laboratory • Thomas Jefferson National Accelerator Facility • Lawrence Berkeley National Laboratory • Los Alamos National Laboratory • Oak Ridge National Laboratory

# SNS PLC PROGRAMMING GUIDELINES

January 2001

---

Dave Gurd  
Senior Team Leader, Controls Group

---

Date

## CONTENTS

|  |    |
|--|----|
| 1. Scope .....   | 3  |
| 2. The ControlLogix PLC and Ethernet .....                       | 3  |
| 3. PLC Programming (Not Related to Ethernet Communications)..... | 4  |
| 3.1 PLC I/O Processing.....                                      | 4  |
| 3.2 Contacts, Normally Open or Closed .....                      | 4  |
| 3.3 PLC Configuration Requirements .....                         | 5  |
| 3.4 Alias Tags .....   | 5  |
| 4. Guidelines for Integrating the PLC into EPICS .....           | 5  |
| 4.1 EPICS Inputs and Outputs to the PLC .....                    | 5  |
| 4.2 Scan Rates .....   | 7  |
| 4.3 Engineering Units Conversion .....                           | 8  |
| 4.4 Alarm and Interlock Handling.....                            | 9  |
| 4.5 PLC Diagnostics .....  | 10 |
| 4.6 Local/Remote Operation .....                                 | 10 |
| 4.7 Command Signals from EPICS to the PLC.....                   | 11 |
| 4.8 Signal Naming.....   | 11 |
| 4.9 PLC Component Naming .....                                   | 12 |

## SNS PLC PROGRAMMING GUIDELINES

### Acronyms used:

|             |  |
|-------------|--|
| A-B         | Allen Bradley                                      |
| EPICS       | Experimental Physics and Industrial Control System |
| EtherNet/IP | Allen-Bradley “Ethernet Industrial Protocol”       |
| IOC         | Input/Output Controller                            |
| PLC         | Programmable Logic Controller                      |
| T/C         | Thermocouple                                       |

### 1. SCOPE

These guidelines apply to Allen-Bradley ControlLogix PLCs that communicate with EPICS IOCs via the 1756-ENET Ethernet interface.<sup>1</sup> These guidelines focus primarily on how to implement the PLC-EPICS interface, but cover some other PLC programming issues as well.

### 2. THE CONTROLLOGIX PLC AND ETHERNET

The ControlLogix PLC uses “tags” to store pieces of data internally. This allows use of meaningful names in the PLC ladder logic and makes arrays and user-defined tags available.

An IOC can access any tag in the “Controller Tags” section of the PLC ladder logic. These tags are global variables, and they include the tags of I/O modules that are automatically created by the RS Logix programming software. (Note that the IOC cannot access “Program Tags”). The EPICS system can read and write scalar BOOL, SINT, INT, DINT, REAL, and tags, as well as array elements of those types.

Set-up of the Ethernet interface is different from a ControlNet interface in that no additional setup is required for either the PLC (e.g. you don’t have to “publish” selected tags) or the ControlNet network (e.g. you don’t have to go through the configuration process of distributing bandwidth between the nodes on the network).

Each single tag transfer to EPICS takes ~8ms, so data elements should be combined and transferred as array tags. For example, transferring an array of REAL[40] takes only a little longer than 8ms whereas individual transfers of 40 REALS would take  $40 \times 8\text{ms} = 320\text{ms}$ . The IOC software automatically combines requests for array elements into whole-array transfers, so you

---

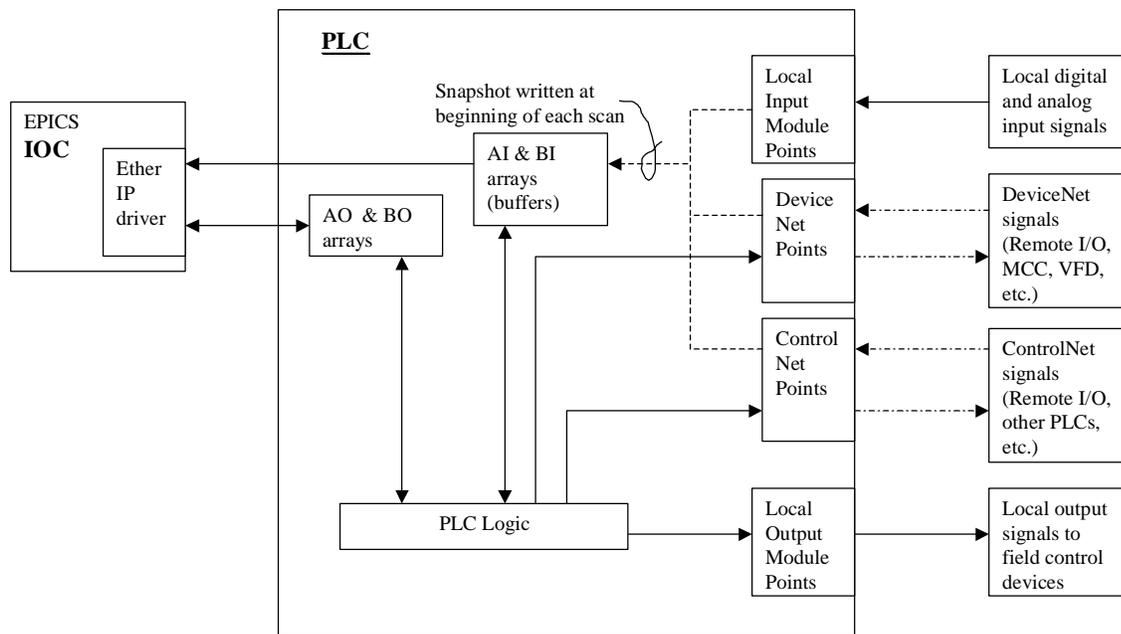
<sup>1</sup> Early in the SNS project two methods of interfacing EPICS to ControlLogix PLCs were evaluated: EtherNet/IP and ControlNet. After evaluation the EtherNet/IP approach was chosen as the standard interface to be used by SNS. At the time this was written there were no plans to support the ControlNet interface to IOCs. However, should ControlNet (or any other ControlLogix PLC interface) be supported at a later date, many elements of these guidelines could be applied to that interface as well. Note that ControlNet is still the preferred method for PLC-to-PLC communications.

can lower the transfer overhead by using array tags. The PLC Ethernet driver has a transfer buffer limit of 500 bytes, so the sizes of the arrays that may be transferred have finite limits.

### 3. PLC PROGRAMMING (NOT RELATED TO ETHERNET COMMUNICATIONS)

#### 3.1 PLC I/O Processing

Allen-Bradley PLCs asynchronously read inputs, so inputs can change state part of the way through a scan and affect the way the PLC logic is designed to function. Therefore all PLC inputs should be scanned into input buffer arrays (analog and discrete) at the beginning of each program scan. This will ensure that any sequence-dependent logic will operate on a time-consistent set of inputs.



Note (1) – Solid lines indicate “copy values to/from continuously/in-real-time/asynchronously”, i.e. normal operation  
 Note (2) – Dashed lines indicate “copy values to/from buffers at the beginning of the PLC scan”  
 Note (3) - Dash/dot lines indicate asynchronous, constant-time-interval-based transfers. ControlNet parameters will be handled the same way as DeviceNet parameters. These will be set up to scan at predetermined time intervals (e.g. 200msec.) on a parameter-by-parameter basis.

**Figure 3.1 – PLC Buffering**

PLC output modules should be configured to set the outputs to a known fail-safe state in the event communication to the PLC processor fails, the PLC is not in the “Run” Mode, or the output module fails.

#### 3.2 Contacts, Normally Open or Closed

All switches and contacts should be wired in a fail-safe manner such that (1) a closed contact indicates a "good", "normal", or "known" condition and (2) an open contact indicates an "alarm", "off normal", or "unknown" condition.

Position switches should be furnished on valves to provide positive indication of the valve position at each end of travel. Each limit switch should be wired so that a **closed** contact indicates the valve is in the position being monitored, e.g., an upper limit contact will be closed when the valve is at its upper limit. With this configuration, operators' screens will show the valve in transition if both contacts are open. This is also the resulting (and desired) indication if the valve I/O cable is disconnected.

Status switches (e.g. a high-pressure switch, a low-level switch, etc.) should be wired fail-safe such that a closed contact indicates a "normal" or "good" condition. Thus, if a wire is cut the "off normal" or "alarm" condition is indicated.

Travel limit switches on motor actuated devices should be wired so that an open contact indicates the device is in the position monitored by the switch. This provides fail-safe operation. If a wire is broken, the indication will be that the device is at its limit position and the motor drive will not attempt to move the device past its limit position.

### **3.3 PLC Configuration Requirements**

Because we plan to use diagnostics to detect module problems, do not configure I/O modules for rack optimization. (While rack optimization conserves ControlNet connections, it does not allow the diagnostics to function).

Variables should be passed between PLCs using arrays in order to conserve the number of ControlNet connections. The ControlNet arrays should be named as specified in section 4.8 below. ControlNet node numbering and other ControlNet conventions should follow guidelines provided by Allen Bradley. The PLC controller name in RSLogix should be the same as the SNS device name of the PLC.

### **3.4 Alias Tags**

Use "alias tags" in the PLC to keep the ladder logic readable. For performance reasons, the data for e.g. "water inlet temperature" might reside in the REAL array element P10R5[17] (see array name guidance in a later section). The RS Logix software allows for the creation of an alias (e.g. "WtrTIn = P10R5[17]") so that the programmer can use the more meaningful alias tag in the ladder logic while the EPICS IOC can transfer the value as part of the P10R5 array.

Use "alias tags" in the PLC to keep the ladder logic readable. e.g.: For performance reasons the data for "water inlet temperature" might reside in the REAL array element P10R5[17] (see array name guidance in a later section). The RS Logix software allows for the creation of an alias (e.g. "WtrTIn = P10R5[17]") so that the programmer can use the more meaningful alias tag in the ladder logic while the EPICS IOC can transfer the value as part of the P10R5 array.

## **4. GUIDELINES FOR INTEGRATING THE PLC INTO EPICS**

### **4.1 EPICS Inputs and Outputs to the PLC**

EPICS inputs should be read from the same input buffer that the PLC logic reads from. (Ref. figure 3.1 above).

Keep the arrays unidirectional: Arrays of data that the PLC produces for the IOC to read should be different from arrays that the IOC writes into for the PLC to read.

Array transfer to and from EPICS is more efficient than transferring individual tags. EtherNet/IP data transfer packets cannot be longer than (approximately) 500 bytes, so array sizes must be limited. Transferred data includes tag names whose length can vary. Rather than try to manually figure out the absolute maximum number of elements that can be transferred for a given array, it is recommended that you use “medium-sized” arrays and let the driver handle the transfer size calculations. The EtherNet/IP driver will check the exact request and response lengths and build a data packet sized as large as permissible (i.e. below the 500 byte limit). Therefore if you use sizes of 40 elements for REAL arrays and 350 elements for BOOL arrays, the EtherNet/IP driver will typically be able to combine 2-3 requests into one transfer.

Listed below are suggested ways of handling the data arrays being sent to EPICS.

1. Binary data - put bits (digital, binary data) into Boolean arrays (BOOL type for the ControlLogix PLC). This allows efficient transfers between the IOC and the PLC. Array size should be no larger than 350 bits.

To make the PLC code more readable, aliases can be used e.g.

```
BOOL bin_array[350]
BOOL InpValveOpen = bin_array[5]
```

Put related bits next to each other in the array. Multi-bit binary inputs in EPICS expect the bits to be contiguous. An example of this type of device is a valve with a bit for open and a bit for closed which is read into a 2 bit, 4 state mbbi record:

```
0, 0 ⇒ open/traveling
1, 0 ⇒ at left limit switch
0, 1 ⇒ at right limit switch
1, 1 ⇒ error
```

The same applies to a value where limits are checked within the PLC program. e.g. Binary tags used to indicate a limit excursion could be put in adjacent array elements like this:

```
P30B[42] = low limit hit, P30B[43] = high limit hit.
```

Then the individual limit conditions can be read as well as the combination of “OK” (00), “low” (10), “high” (01) or “error” (11).

Try to avoid DINT arrays where only few bits per DINT are used. The same applies to INT arrays. Instead, put those bits into BOOL arrays.

2. Analog data – PLC analog values (REAL, DINT, INT) should also be combined in array tags for efficiency. Array size should be no larger than 40 values. As with binary data, alias tags can be used to make the ladder logic more readable, e.g.:

```
REAL readbacks[40]
REAL in_flow_sensor = readbacks[2]
```

Arrays should also be grouped by update rate if the IOC can transfer some of them at a slower rate, reducing the network load (e.g. have arrays for readbacks read at 10Hz, arrays read at 1 Hz, etc.). Arrays may also be grouped by function: e.g. setpoints, readbacks, limits, etc. This can help to limit the individual array size as required by the PLC buffer limit.

Note that a given array can consist of only one type of variable. Structures are not supported and must not be used.

As indicated above, one design approach is to put related signals in the same relative location in separate arrays. However this might not be the best approach for all systems. Study the analog signals for your system and organize them into groups. Establish and document a pattern for each grouping of signals. Design the EPICS buffers for easy comprehension and efficient transfer of data between the PLC and EPICS.

## 4.2 Scan Rates

The rate at which the EtherNet/IP driver reads from the PLC is configured in the EPICS record via the user-specified SCAN rate parameter. When several records refer to elements of an array tag, the driver uses the fastest requested scan rate. Thus one should set the scan rate for all elements of a given array to the same rate.

For records configured with “SCAN = xxx” seconds, the maximum scan rate is 10 Hz by default. Higher rates can be used, but the rate is limited by the ~8ms transfer time per request. The absolute upper limit is ~100 Hz where only one tag is transferred. For records configured with “SCAN=I/O Intr”, the INP/OUT field specifies the scan period. In that case, non-standard periods can be used, e.g. 20Hz.

Keep in mind that network transfers could be delayed. If a PLC output has to be on for 0.1 second, do not expect the IOC to toggle it on and then off in this time. Instead, implement a handshaking mechanism via “operate” and “done” tags. When the IOC writes to the operate tag, the PLC will switch the output on for exactly 0.1 seconds. When done, the PLC will increment a “done” tag. This way the 0.1 second duration is not hampered by network delays.

The ControlLogix Ethernet adapter module receives and transmits Ethernet packets over the network. It does some work on the packets and exchanges data with the PLC processor module. The PLC module processes I/O during its overhead time slice, which by default is set to 10% of the processor time. If there is a lot of PLC-IOC Ethernet traffic then there may not be sufficient time for all data to be exchanged between the Ethernet adapter and the PLC processor, resulting in lost data. When a continuous task (i.e. ladder logic) is running in the PLC, data starts getting lost when more than approximately 60-70 Ethernet packets per second are processed. The number of packets processed may be increased by increasing the overhead time slice.

Three (3) Ethernet packets are generated for each read and write operation between the PLC and the IOC EtherNet/IP driver. The approximate number of packets per second to be processed for EtherNet/IP communication may be calculated by:

$$(NAR + NAW) * 3$$

where:

NAR = number of arrays read from the PLC per second

NAW = number of arrays written to the PLC per second

There may also be other Ethernet communications to a PLC. When a programming terminal is connected to the PLC via Ethernet, 10 to 30 packets per second are exchanged between the PLC and the programming terminal. For some systems, there may be Ethernet messages between/among PLCs. This traffic must be included in the total number of packets processed.

### 4.3 Engineering Units Conversion

It is “designers choice” as to where to convert to engineering units: in the PLC or the IOC. There are advantages and disadvantages for each method. Once the decision is made it should be consistently applied for a given PLC interface.

Use real values, not integers, for all analog signals transferred.

Converting in the IOC: Converting to engineering units in the IOC has the following advantages:

- The raw value, the conversion method and parameters, as well as the engineering value and units, are accessible over Channel Access.
- The configuration can be kept in the overall configuration database. Changes can be archived and restored.

Converting in the PLC Modules: Thermocouple and RTD signals can be converted to engineering units by the T/C and RTD PLC input modules. Linear analog input signals can be converted to engineering units by the analog PLC input modules. An example of a linear module conversion follows.

Suppose we have a pressure transmitter with a 4 – 20 mA output that is calibrated such that 4 mA = 0 atm and 20 mA = 30 atm. The analog input module should be configured per the following table:

**Table 4.3-1. Example Input Module Engineering Units Conversion**

| <b>Input signal (mA):</b> | <b>Value in PLC (atm):</b> |
|---------------------------|----------------------------|
| 0                         | -7.5                       |
| 4                         | 0.0                        |
| 8                         | 7.5                        |
| 12                        | 15.0                       |
| 16                        | 22.5                       |
| 20                        | 30.0                       |

Scale analog output modules as follows. Configure the current modules to scale the 4 to 20 ma signals so that 0.0 mA = -25.0% and 20.0 mA = 100.0 %. This allows the module to continue sending valid data if there is a slight offset at 4 mA. Configure the voltage modules to scale the 0 to 10 VDC signal to 0.0 to 100.0 %.

**Table 4.3-2. Example Output Module Engineering Units Conversion**

| <b>Value in PLC (%):</b> | <b>Output signal (mA):</b> |
|--------------------------|----------------------------|
| -25                      | 0                          |
| 0                        | 4                          |
| 25                       | 8                          |
| 50                       | 12                         |
| 75                       | 16                         |
| 100                      | 20                         |

While it is possible to perform non-linear engineering units conversions using PLC logic, it is generally easier to do so in the IOC. Therefore, for signals requiring non-linear engineering unit conversions the PLC input modules should be configured such that the engineering units are in “mA” (i.e. 4 mA input  $\Rightarrow$  4 mA engineering units value and 20 mA input  $\Rightarrow$  20 mA engineering units value). EPICS can then convert the non-linear analog input signal to actual engineering units.

In the IOC, use the linearization (LINR) value “No Conversion” to preserve PLC-converted values. When connecting ai/ao records to a REAL tag, the VAL field is used automatically (no conversion). For other tag data types, the RVAL field is used and IOC-side conversions can take place.

#### **4.4 Alarm and Interlock Handling**

For the discussion that follows, a distinction is made between “interlock limits” and “alarm limits”. Exceeding alarm limits results in operator notification (i.e. display actions only). Exceeding interlock limits results in protective control actions.

Alarm limits should be kept in the EPICS database. EPICS provides a fully-functional alarm handler, and it is simply easier to maintain alarm values in one place.

Interlock limits should be defined in the PLC. Don’t forget to provide some deadband on return-to-normal to prevent interlock chatter. To allow meaningful operator screens, the PLC should provide the IOC with the current process value, the limits, and interlock status for each interlock. This arrangement does not allow operators to change the interlock limits from EPICS, but at least the limits can be displayed for operators. If write access is desirable, the interface can be expanded to include analog output records. These will allow operators to modify the interlock limits via EPICS. Access can be restricted/denied through standard EPICS access control if necessary.

To generate alarms in EPICS, the alarm limits are set in the readback channel, e.g.:

```
ai - readback,  
alarm limits {LOLO=0, LOW=1, HIGH=9, HIHI=10}
```

If we maintain the alarm limits in the IOC, we can read/write the alarm limits using standard EPICS tools. Note that with the configuration described the alarm setpoints remain separate from any interlock limits (e.g. the HIHI alarm setpoint will not change with high\_interlock\_limit.VAL).

It is acknowledged that there may be cases where the designer might choose to put an alarm limit in the PLC, e.g. when it is desired that the alarm and interlock limits share the same value. For those hopefully-rare cases, alarm limit checking should be handled as follows:

- The alarm limit check in the PLC should result in an alarm status bit being transferred to the IOC. This alarm status can then be annunciated by the EPICS alarm handler.
- The alarm limit check should include deadband to prevent alarm chatter.
- When the alarm condition no longer exists, the PLC alarm bit should be automatically reset. (The EPICS alarm will remain latched until the operator acknowledges and clears the alarm. Also note that if the alarm condition still exists, an EPICS reset command will have no effect on the PLC alarm latch bit).

#### **4.5 PLC Diagnostics**

The PLC should provide the following diagnostic information as tags:

1. Heartbeat
2. Logic execution time (at least the last time, preferably also min/max). This can then be used to inform operators (and in some cases IOCs) of latency problems.
3. Module failure status. Use the module status bits to alarm I/O failures by monitoring if the word is zero (normal) or non-zero (failure).
4. Status of external connections (e.g. to Flex I/O).

The EPICS database should keep pseudo-channels for state of the communication to the PLC (e.g. number of timeouts/errors in communication with PLC, tag's maximum scan time, etc.).

The EPICS Alarm Handler should be used to display any alarms generated by these status bits. Signal names should conform to the standard SNS signal naming convention (e.g. CF\_TA:PLC8701\_01EN01:Sts). See section 4.9 for PLC component naming guidelines.

#### **4.6 Local/Remote Operation**

If a PLC value can be set by both an IOC and a local operator interface (e.g. an Allen-Bradley PanelView terminal), a local/remote switch is required. The preferred approach is to use a physical switch located near the local operator panel, with its status being read by the PLC. Alternatively a software tag in the PLC can be used. In either case there must be a BOOL tag in the PLC to indicate local/remote mode.

For each output "X" that can be affected by both a local and a remote operator, the PLC should have three tags:

local X - written to by the local panel,  
remote X - written to by EPICS,  
X - the actual output.

Depending on the local/remote switch setting, the ladder logic transfers the "local" or "remote" value to the actual output tag. In remote mode, the local input is therefore ignored. In local mode, the IOC might continue to run its loop and write to the "remote" value, while a "local

mode" indicator on the EPICS screen explains why the hardware output is not reacting to the IOC.

Alternatively, the interface could be set up so that a PLC value is changeable from either EPICS or a local operator panel (i.e. both write to the same PLC register). This approach takes advantage of the fact that EPICS output values are generally transferred only when they change in EPICS. (Be aware that they may also be transferred when a value sharing the same array changes). Thus it is possible to design the PLC and IOC logic to write to the same PLC register without EPICS continuously overwriting the register. EPICS can be configured to update an IOC output register with the latest actual value in the PLC. The IOC record's "OMSL" field must be set for "supervisory mode" for this to occur. The IOC output record will then periodically scan the actual PLC output tag for changes. In principle this configuration (i.e. a single IOC output connected to a single output tag in the PLC) would work in most cases. But an implementation based on a local/remote switch with local/remote/output tags is cleaner and more fool-proof.

#### 4.7 Command Signals from EPICS to the PLC

All Boolean EPICS command signals should be momentary contacts that are latched in the PLC<sup>2</sup>. Configure EPICS (the '.HIGH' field for 'bo' records) to hold command signals high long enough for the PLC to latch the command. (e.g. 1 or 2 sec.). The scan time used by the EtherNet/IP driver ('S t' where t is the scan time in seconds) for output records should be set shorter than the 'HIGH' time to assure that each command is transmitted to the PLC. The EtherNet/IP scan time should be no more than 1 second for any output record. (NOTE: The EtherNet/IP scan time MUST be the same value for all records that are elements of the same array).

With this approach it is possible to assert both "on" and "off" commands at the same time. The PLC logic should be set up so that the safer state always prevails.

As indicated in the previous section, the PLC can change the value of a command signal in EPICS. This feature can be used to provide bumpless local/remote control switching of EPICS-based control loops. e.g. EPICS can have one of its PID loop outputs updated to match the corresponding PLC output value when the loop is in local mode.

#### 4.8 Signal Naming

The SNS naming standard should be applied to EPICS process variable names. The following convention should be used for naming PLC arrays:

PLC Array Name convention format: PIIITn[x]:

- P – Producer of the data
  - E = EPICS
  - F = Flex I/O

---

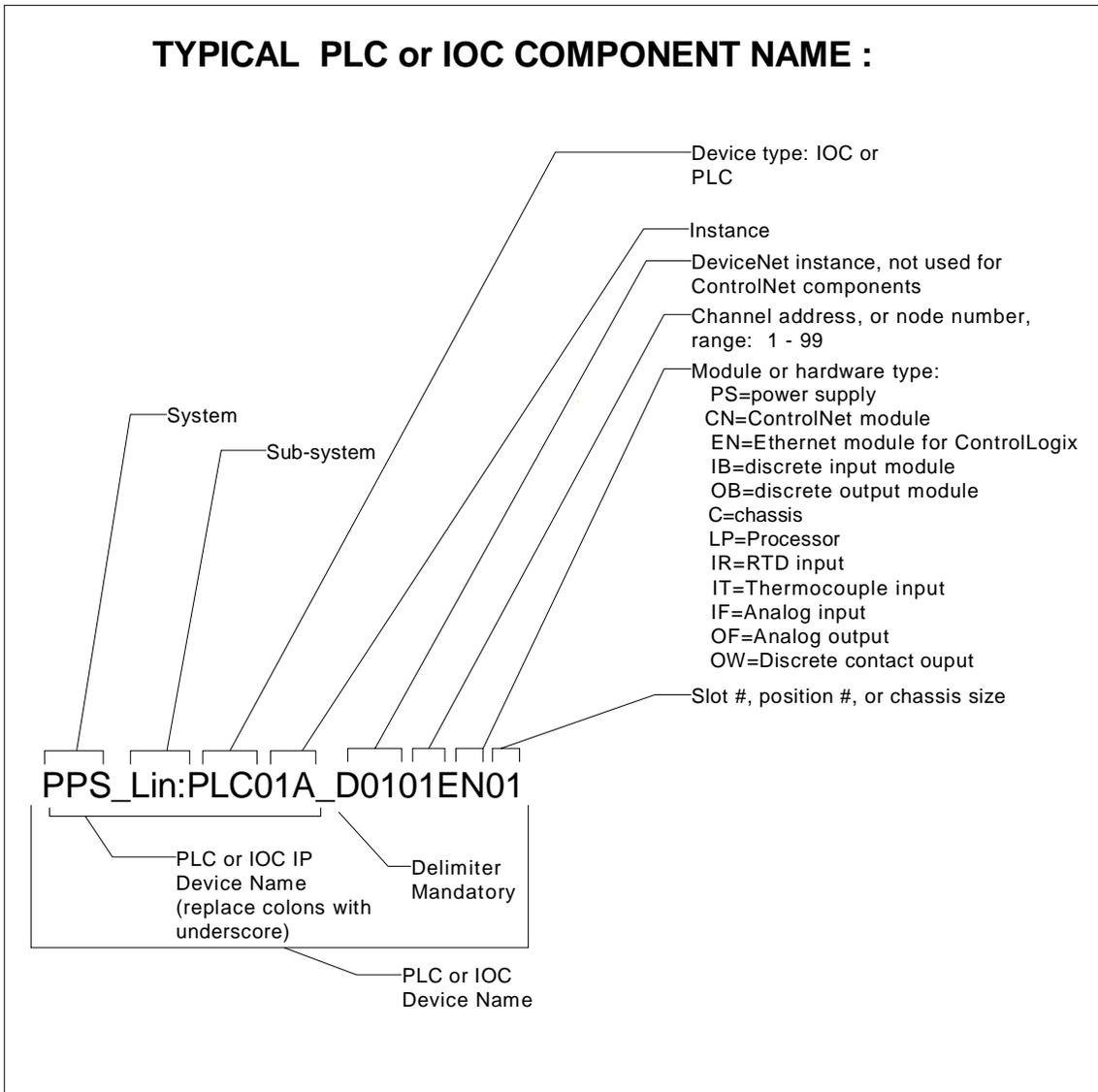
<sup>2</sup>There is some small possibility that due to the asynchronous nature of the EtherNet/IP communications a momentary-contact-style command message could get lost. A handshake approach to implementing EPICS-to-PLC commands might reduce the chances of losing a command message, but this approach was dropped due to the significant complexity it adds to the logic.

- P = PLC
- ...
- IIII – PLC/EPICS/Flex I/O Identifier (e.g.: 4323, C20, etc.)
- T – Type of array
  - B = Boolean
  - D = Double Integer
  - N = Integer
  - R = Real
- n – Array Instance for the type of array
  - Use odd numbers for data going to EPICS (1,3,5,...)
  - Use even numbers for data coming from EPICS (2,4,6,...)
- x – Array Element

For EtherNet/IP, PLC instance number is technically not necessary because the tag namespace is limited to a given PLC. But since ControlNet requires this, use of PLC instance number is recommended in any case.

#### **4.9 PLC Component Naming**

PLC components should be named according to the convention illustrated in figure 4.9 below (e.g. for PLC component failure alarms).



**Figure 4.9** – PLC Component Naming Diagram

**References:**

“Interfacing the ControlLogix PLC over EtherNet/IP”, K. Kasemir et al, ICALEPCS 2001 paper THAP020.

See also help files maintained with the EtherNet/IP driver software in the SNS CVS repository.