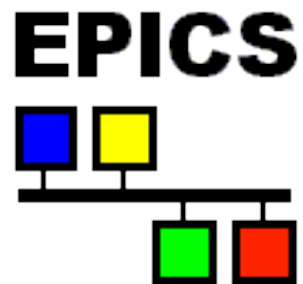


EPICS EtherIP Driver



EtherNet/IP

Kay Kasemir

kasemirk@ornl.gov

July 2017

EtherNet/IP ?

- **Industrial Protocol,**
a.k.a.
 - **ControlNet-over-Ethernet**
 - **TCP encapsulation of CIP (Control & Information Protocol)**
- **Open DeviceNet Vendor Association (ODVA)**

Routing, messages, objects, services, data types

EPICS “ether-ip” Support

EtherNet/IP spec:

- ✓ Connect & read serial number of ENET module



Allen Bradley 1756-RM005A-EN-E.pdf:

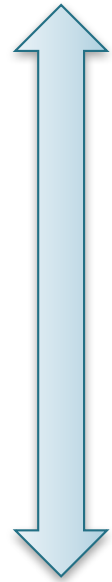
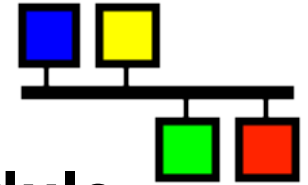
- ✓ Read & write PLC tags



EPICS support:

- ✓ IOC records read/write

EPICS



History, Status

- **Developed ~2000 for SNS**
 - Key to its operation: Vacuum, cryo, water, ...
 - Now used by many EPICS sites
- **Convenient**
 - Compared to serial, ControlNet, ...:
No special interface hardware nor wiring
 - Can read/write any controller tag on the PLC,
no “publish” required
 - “Fast enough”
- **Supports AllenBradley ControlLogix,
CompactLogix**

PLC Requirements

- ENET Module

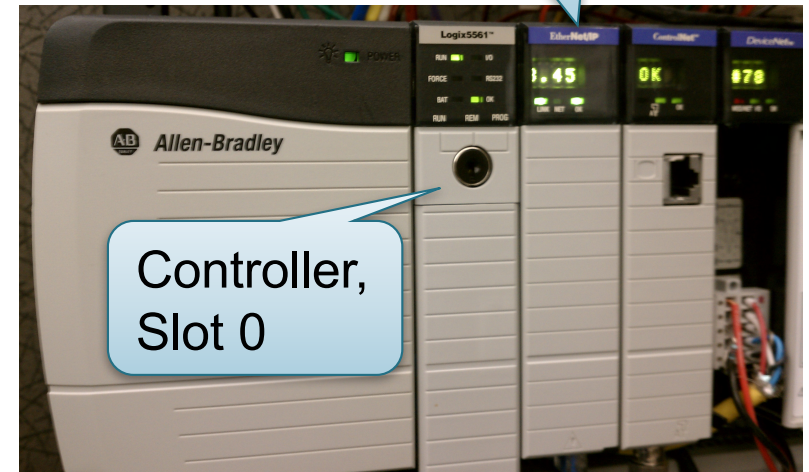
- Know the IP address
- Know the CPU slot number

One PLC could have multiple controllers!

- Tags

- Know tag names
- “Controller” level,
not! private to program
- No need to “produce”

ENET Module,
IP 10.1.2.47



Program Tags - MainProgram

Scope:	MainProgram	Show:	Show All	Sort:	Tag Name
Tag Name	Alias For	Base Tag	Type		
north_tank_mix			BOOL		
north_tank_pressure			REAL		
north_tank_temp			REAL		
+one_shots			DINT		
+recipe			TANK[3]		
+recipe_number			DINT		
replace_bit			BOOL		
+running_hours			COUNTER		
+running_seconds			TIMER		
start			BOOL		
stop			BOOL		

Monitor Tags Edit Tags

EPICS Records

Input Records (ai, bi, mbbi, mbbiDirect, stringin)

```
field(DTYP, "EtherIP")  
field(INP,  "@my_plc name_of_tag")  
field(SCAN, "1 second")
```

Output Records (ao, bo, mbbo, mbboDirect)

```
field(DTYP, "EtherIP")  
field(OUT,  "@my_plc name_of_tag")
```

PLC Data Type ⇔ Record Type

REAL, INT, DINT ⇔ **ai, ao (REAL?VAL:RVAL)**

INT, DINT, BOOL[] ⇔ **bi, bo, mbbi, mbbo, mbbiDirect, mbboDirect**

BOOL ⇔ **bi, bo**

STRING ⇔ **stringin, stringout**

For arrays, access one element:

```
field(INP, "@my_plc my_array[42]")
```

For structures, access one element:

```
field(INP, "@my_plc gadget.ps.voltage")
```

Starting the IOC: Development

Type this on one line:

eipIoc

-p my_plc=10.1.2.47

-m "P=MyPrefix,N=2"

-d example.db

'softloc' with EtherIP built in

-p "name=ip,slot"

Define PLC name for "@my_plc"
in INP/OUT links,
IP address, optional CPU slot

-m "macro=value"

Define database macros

-d path/to/some.db

Load database

Starting the IOC: Production Setup

Add driver to 'makeBaseApp' type IOC:

configure/RELEASE:

ETHER_IP=/path/to/share/ether_ip

yourApp/src/Makefile:

your_DBD += ether_ip.dbd

your_LIBS += ether_ip


iocBoot/iocYours/st.cmd:

drvEtherIP_init()

drvEtherIP_define_PLC("my_plc", "10.1.2.47", 0)

dbLoadRecords(...

Arrays, Aliases

- **Arrays are MUCH more effective**
 - Read scalar REAL tag: 8ms
 - Read REAL[40]: 8ms
 - Read BOOL[352]: 8ms
 - Read 352 separate BOOL tags: $\sim 352 * 8 \text{ ms}$
 - **Records with INP or OUT that access**
 - INP="@plc my_array[2]"
 - INP="@plc my_array[39]"
 - Driver reads my_array[0..39]
-
- 
1. Arrange data in array tags
 2. Configure INP/OUT to use array elements
 3. May use alias tags within PLC logic

Driver Combines Tags used by Records

Records

1. INP “@plc1 **arrayA[2]**”, SCAN “2 second”
2. INP “@plc1 **arrayA[3]**”, SCAN “2 second”
3. INP “@plc1 **arrayB[13]**”, SCAN “1 second”
4. INP “@plc2 **arrayA[2]**”, SCAN “2 second”

Driver

1. “plc1”

- Handle “**arrayA[0-3]**” every 2 seconds
- Handle “**arrayB[0-13]**” every 1 second

2. “plc2”

- Handle “**arrayA[0-2]**” every 2 seconds

Reading, Writing, Scanning

Records

1. INP “@plc array[2]”, SCAN “2 second”
2. INP “@plc array[3]”, SCAN “1 second”
3. INP “@plc array[13]”, SCAN “I/O Intr”
4. OUT “@plc array[5]”, SCAN “Passive”

Driver

“array[0-13]” read every 1 second

1. Reads most recent value[2] every 2 seconds
2. .. [3] every 1 second, data may be 0.999 sec old
3. Processes whenever value [13] changes
4. Reads (though *output!*) when value [5] changes.
When processed, updates value [5], marks whole tag for writing. Driver will then once *write* instead of read.

Scan Flag “ S <seconds>”

Records

1. INP “@plc array[2]”, SCAN “2 second”
2. INP “@plc array[3]”, SCAN “**1** second”
3. INP “@plc array[13]”, SCAN “I/O Intr”
4. OUT “@plc array[5]”, SCAN “Passive”

Driver

Uses the “fastest” SCAN of any record related to a tag.

Records 2 & 3 should add “@plc array[..] **S 1**” to provide suggested scan period, otherwise warning

“cannot decode SCAN field, no scan flag given”

Elementary “ E” Flag

Records

1. INP “@plc array[190]”
2. OUT “@plc array[191]”

Driver

1. Reads array[0..191], but we may only use [190], [191]
→
“@plc array[190] E” causes transfer of just that array element
2. Writes all of [0..191] even if just 191 was changed, *writing* elements which we otherwise wanted to *read*
 - a) Use separate arrays for “writing” and “reading”
 - b) Add “ E” to write just that element, not whole array

Bitfiddly Stuff

Internally, DINT and BOOL[32] look similar.

Driver assumes **bit** for binary records.

`"some_tag[40]"`

a) ai, ao:

Element 40 of a REAL, INT, DINT array

b) bi, bo, mbb*:

Bit 40 of a BOOL array,

or

bit 40 of a DINT array, i.e. bit 8 of DINT[1]

To force [i] to be the i^{th} array element:

`"some_tag[1] B 8"`

Diagnosis on IOC

- **Set records' TPRO field**
 - Show when tag is read/written by that record
- **drvEtherIP_dump**
 - Dumps all tags and their value
- **drvEtherIP_report *level***
 - Dumps info about each plc, period list, tag
 - Level 0..10
- **EIP_verbosity(7), EIP_verbosity(10)**
 - Show warnings or details of protocol

The Elusive Buffer Limit

Driver combines several reads and writes for tags into one network transfer until either the request or the response reaches `EIP_buffer_limit`

How large? About 500 bytes, details hairy

Set too large

→ PLC error “Buffer too small, partial data only”

Set too small

→ Inefficient, can't combine many tags

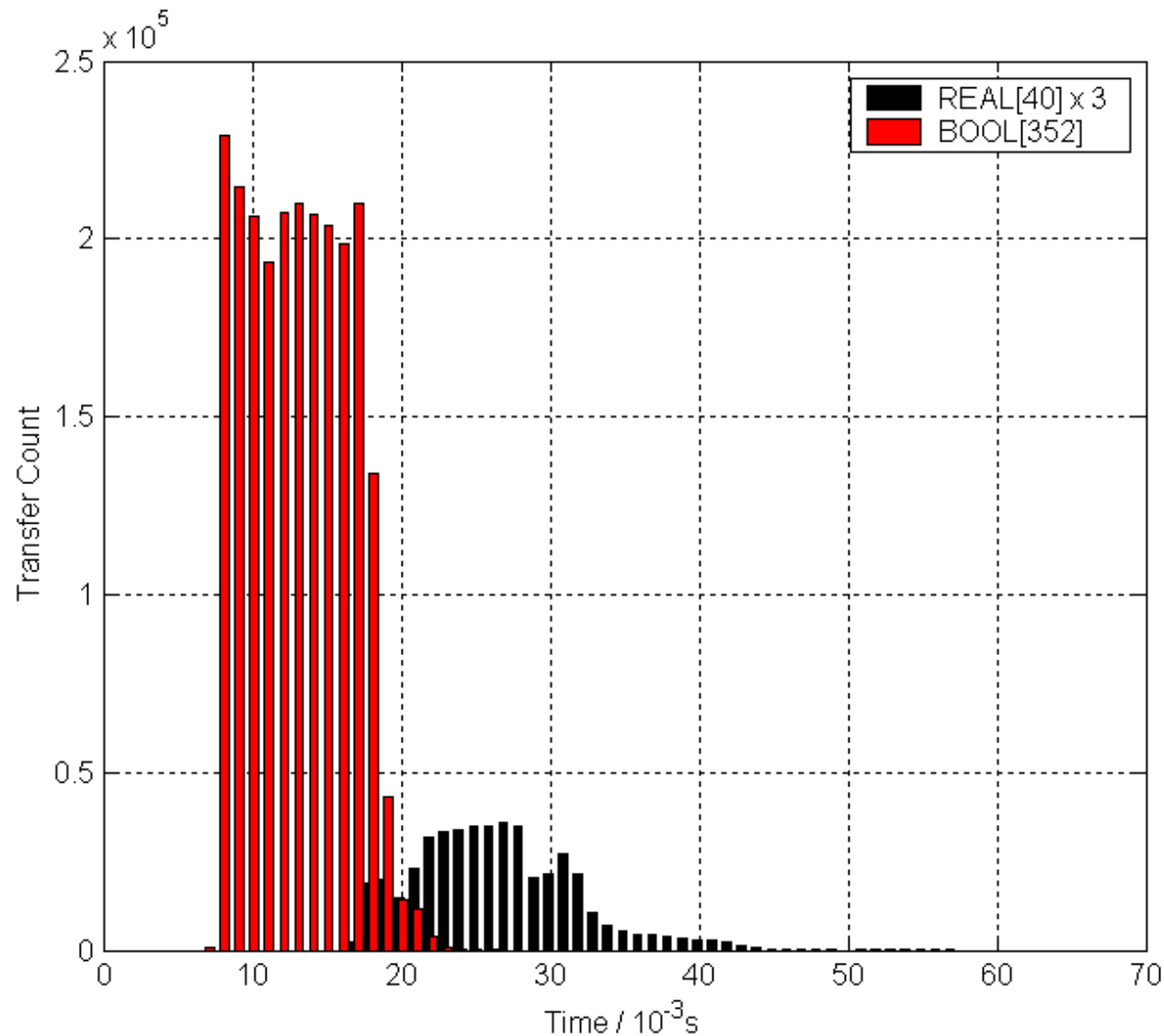
Set way too small

→ Can't read a single `REAL[40]` array, driver error
“Tag ‘xyz’ exceeds buffer limit of .. Bytes”

Don't mess with the default `EIP_buffer_limit`.

**Keep arrays to `REAL/DINT/INT[40]` and `BOOL[400]`.
Driver will then combine several of those in one transfer.**

Transfer Time Histogram



352 BI @ 10Hz, 3 x REAL[40] @ 2 Hz

Summary

- **Arrange PLC data in arrays**
 - Separate ones for read, write
 - Maybe 40 elements per array (400 for BOOL)
- **Point records' INP/OUT to those array tags**
- **Done**