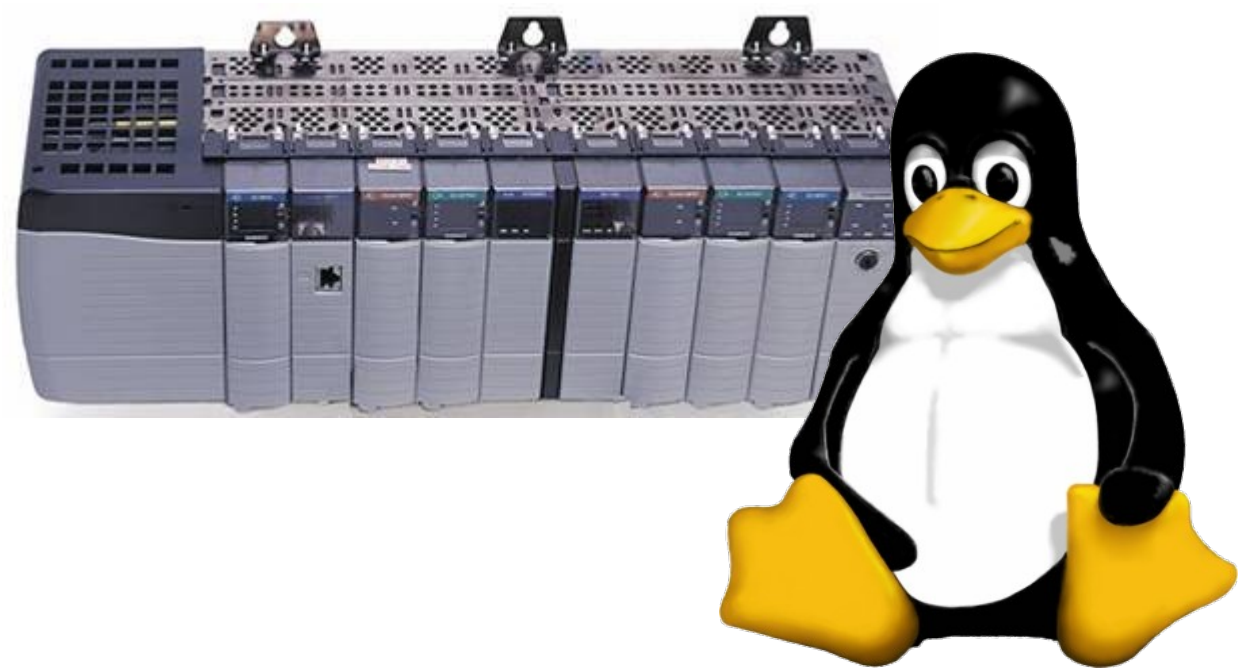
TuxEip



Copyright

WARNING

DON'T connect to a PLC unless you are certain it is safe to do so!!! It is assumed that you are experienced in PLC programming/troubleshooting and that you know EXACTLY what you are doing. PLC's are used to control industrial processes, motors, steam valves, hydraulic presses, etc.

You are ABSOLUTELY RESPONSIBLE for ensuring that NO-ONE is in danger of being injured or killed because you affected the operation of a running PLC.
Also expect that buggy drivers could write data even when you expect that they will read only !!!

Purpose	TuxEip is a communications library that facilitates communications to Allen Bradley Controllers, using it, a programmer can write programs to Read/Write controller/program scoped tag data in the following PLC : ControlLogix, PLC5, SLC500, Micrologix, FlexLogix (and probably much more, but that's all i have tested). These PLC are reachable using Ethernet, ControlNet, DH+. As this library implement misc functions to interact with objects defines by the CIP protocol (Connection Manager, Message router...), you can easily add your own function to act on such devices.
Road map	It remain a lot of things to do, but the most important is to support optimized packet for the Logix platform.(There is a sample of optimized read packet for PLC5 / SLC500 in TuxSql which is an another project of mine).
Trademark	ControlNet is a trademark of ControlNet International, Ltd. CIP, DeviceNet, are trademarks of Open DeviceNet Vendor Association, Inc. EtherNet/IP is a trademark of ControlNet International under license by Open DeviceNet Vendor Association, Inc. Allen-Bradley, ControlLogix, DH+, FlexLogix, PLC-5, Micrologix, and SLC are trademarks of Rockwell Automation. Ethernet is a trademark of Digital Equipment Corporation, Intel, and Xerox Corporation. All other trademarks referenced herein are property of their respective owners.

Content

1 Overview

1.1 Ethernet/IP

EtherNet/IP is an encapsulation protocol which use the Control Information Protocol.

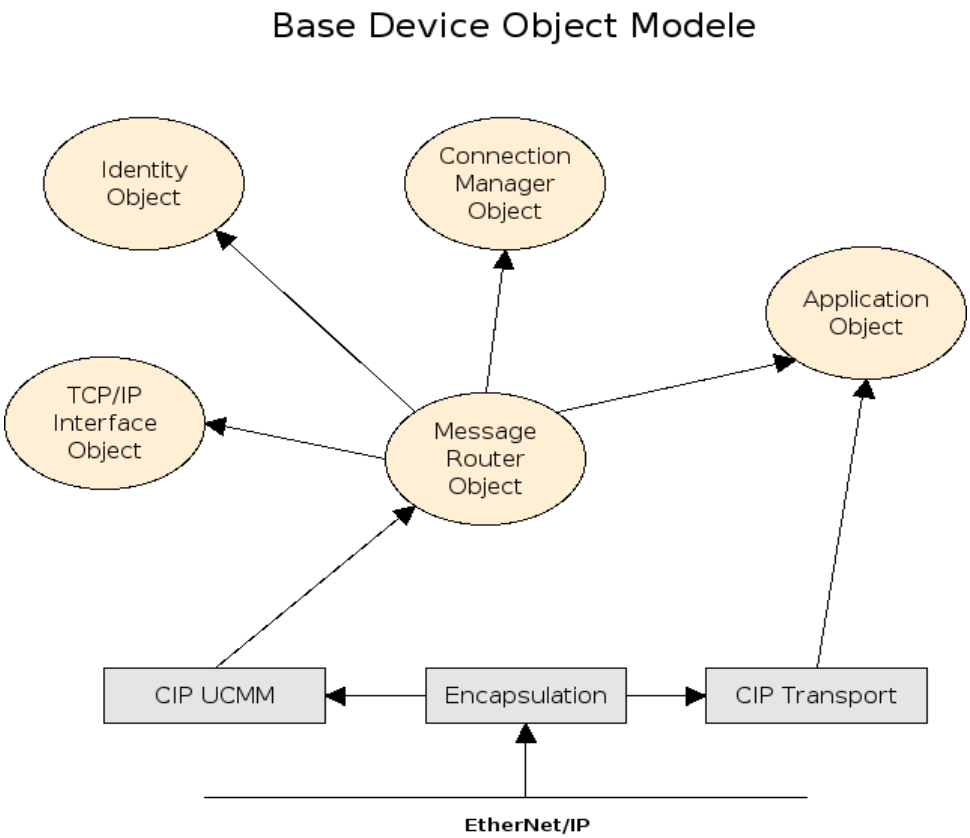


Illustration 1: Base device Object Model

1.2 Organisation

This library is divide in many files :

- « EIP_Const » define many constant used by EtherNet/IP protocol.
- « CIP_ » prefixed files contain generic CIP constants and misc subroutines.
- « Ethernet_IP » contain implementation of the EtherNet/IP protocol.
- « SendData » is used to send / receive data over Ethernet, the whole library depends on it.
  (In fact, TuxEip use function pointer to the send/receive subroutines defined in this file so you can write your own.)
- « MR » contain the Message Router implementation
- « CM » contain the Connection Manager implementation
- « AB » contain generic Allen Bradley data types and subroutines
- « PLC » contain subroutines to access PLC5 and SLC500 controllers ...(controllers which use the PCCC protocol)
- « LGX » contain subroutines to access ControlLogix ...(controllers which use the CIP protocol)

2 Global Variables

2.1 Identity

CM.h

Each device have to provide a vendor Id and a serial number, in TuxEip they are defined like this (you can also overwrite it before accessing the network) :

- (CIP_UINT) _OriginatorVendorID=TuxPlcVendorId;
- (CIP_UDINT) _OriginatorSerialNumber=0x12345678;

2.1 Send / receive

SendData.h

As we discuss before, TuxEip use function pointer to send / receive data.

The library define 3 functions :

- int (***CipSendData**)(int sock,Encap_Header *header)=&_CipSendData;
- Encap_Header *(***CipRecvData**)(int sock,int timeout)=&_CipRecvData;
- Encap_Header *(***CipSendData_WaitReply**)(int sock,Encap_Header *header,int sendtimeout,int rcvtimeout)=&_CipSendData_WaitReply;

By this way, you can define your own functions.

2.2 Connection parameters

CM.h

When connecting on a device, the Connection Manager use this defaults values

- BYTE _Priority=0x0A;
- CIP_USINT _TimeOut_Ticks=0x05;
- WORD _Parameters=0x43f8; Attention this parameters depend on the network type (ControlNet / DH+)
- BYTE _Transport=0xa3;
- BYTE _TimeOutMultiplier=0x01;

2.3 Error handling

ErrCodes.h

TuxEip define many variables for handling and reporting errors (this variables are thread safe).

- cip_debuglevel = LogError
  Possible values are :
  - LogNone (0) : nothing is report
  - LogError (1) : errors are display
  - LogTrace (2) : function entering and exiting are display
  - LogDebug (3) : data frames are display

- cip_err_type : which kind of error we have :
  - Internal_Error : An error in TuxEip itself (time out, unsupported data type ...)
  - Sys_Error : A system error (broken pipe, memory error ...)
  - EIP_Error : Ethernet/IP encapsulation protocol error
  - MR_Error : Message router error
  - CM_Error : Connection Manager error
  - AB_Error : error while accessing a Logix base controller (ControlLogix,FlexLogix ...)
  - PCCC_Error : error while accessing a PCCC base controller (PLC5, SLC500,MicroLogix 1100 ...)

- cip_errno : error number
- cip_ext_errno : extended error number
- cip_err_msg : plain text error message

3 Sweeping statement about functions

3.1 Functions declaration

There is two sort of function in TuxEip :

- « _ » prefixed functions are predominantly “privates” functions (internally use by the library) while they are much more complex to use.
- Other functions that we can call “public” functions are in fact macro statement refers to “privates” function, passing adequate arguments and constants. Sometimes these functions are very same.

3.2 Return values

Functions can return two types of values

- Pointer :
  Attention, if case of error many functions will return you a null pointer, alternatively you will receive a pointer on a data structure (you will have to free it, sometimes there is a special function to do so)
- Numerical value :
  In case of error, you will receive a negative value, otherwise a null or positive value.
  The signification of the value you receive vary according the function you call.

4 Standard scheme of communications using TuxEip

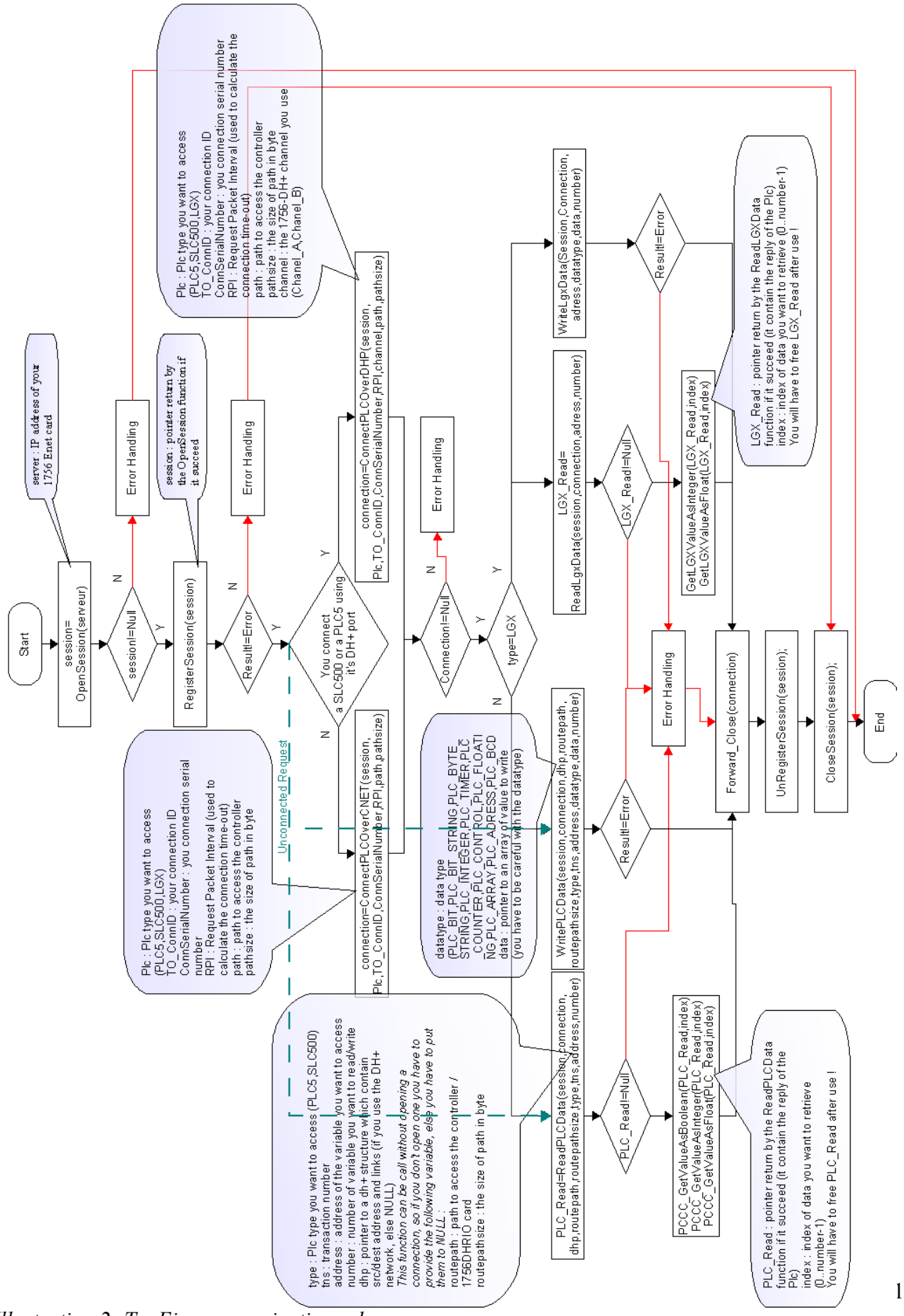


Illustration 2: TuxEip communication scheme

5 In depth travel inside TuxEip data structures

5.1 Session

typedef struct Eip_Session;	
int sock;	socket open on your 1756-ENET card
CIP_UDINT Session_Handle;	Session handle return by the RegisterSession function if it succeed
CIP_DINT Sender_ContextL,Sender_ContextH;	You can use this to your own usage
int timeout;	Time-out used to send / receive frames
int References;	Not used
void *Data;	You can use this pointer to link a session to personal data

5.2 Connection

typedef struct Eip_Connection	
Eip_Session *Session;	Pointer to the session use by this connection
int References;	Not used
void *Data;	You can use this pointer to link a connection to personal data
CIP_UINT ConnectionSerialNumber;	All is in the name...
CIP_UINT OriginatorVendorID;	
CIP_UDINT OriginatorSerialNumber;	
CIP_UDINT OT_ConnID;	Originator to Target connection Identifier
CIP_INT packet;	Packet identifier
BYTE Path_size;	Size of the following path (in bytes)
	Target's path

- ConnectionSerialNumber : be careful, you will have to provide a **matchless** serial number
- Connection Manager use OriginatorVendorID, ConnectionSerialNumber, and OriginatorSerialNumber to identify an existing connection.

5.3 LGX_Read

typedef struct LGX_Read	
LGX_Data_Type type;	Data type (LGX_BOOL,LGX_BITARRAY,LGX_SINT,LGX_INT=,LGX_DINT,LGX_REAL ...)
int Varcount;	Number of variable you have read
int totalsize;	Size in byte of the following data
int elementsize;	Size in byte of one variable
	Data

5.4 PLC_Read

typedef struct PLC_Read	
PLC_Data_Type type;	Data type (PLC_BIT,PLC_BIT_STRING,PLC_BYTE_STRING,PLC_INTEGER,PLC_TIMER,PLC_COUNTER,PLC_CONTROL,PLC_FLOATING,PLC_ARRAY,PLC_ADRESS,PLC_BCD...)
int Varcount;	Number of variable you have read
int totalsize;	Size in byte of the following data
int elementsize;	Size in byte of one variable
unsigned int mask;	Used for boolean
	Data

5.5 DHP_Header

typedef struct DHP_Header	
CIP_UINT Dest_link[1]	0
CIP_UINT Dest_adress	Address of the destination node
CIP_UINT Src_link[1]	0
CIP_UINT Src_adress	Address of the source node

1) This parameters are used when you need to access physically different s DH+ network (refers to Rockwell documentation) and are not need by TuxEip.

6 How to Determine Connection Path ,connection and read / write functions

6.1 Connection path

The path is a series of port / link address pairs (identical to the Communication Path syntax in RSLogix 5000 Message Configuration dialog) where :

- A port describes a way out of a device via a network or backplane.
- A link address is a destination node ,if the corresponding port is a backplane then the link address is the slot number,else it's a network address.

Interface Module	Port 1	Port 2	Port 3
1756- ENET/ENBT	Backplane	Ethernet network	N/A
1756-DHRIO	Backplane	DH+ Network on Ch. A	DH+ Network on Ch. B
1756-CNET	Backplane	ControlNet network	N/A

Note : When using TuxEip the first port is your Ethernet interface, so your path begin with the IP address of your 1756-ENET/ENBT card.

6.2 Connection function

Choosing the right function

Processor	Last network segment is DH+	Last network segment is not DH+
PLC5	ConnectPLCOverDHP[1]	ConnectPLCOverCNET
SLC500		
Micrologix	ConnectPLCOverCNET	
ControlLogix		
FlexLogix		

1. The Path must target the last 1756-DHRIO you use

Fill in fields with the right values

	ConnectPLCOverDHP	ConnectPLCOverCNET
session	The session pointer return by the OpenSession function	
Plc		
TO_ConnID	Originator to Target connection Identifier (for your own use)	
ConnSerialNumber	ConnectionSerialNumber : be careful, you will have to provide a **matchless** serial number	
RPI	Request Packet Interval (used to calculate the connection time-out)	
channel	Channel of the 1756-DHRIO you use (Channel_A or Channel_B)	NA
path	Path to the last 1756-DHRIO you use	Path to the logic controller
pathsize	Size of the path (in bytes)	

6.3 Read / write functions

Choosing the right function

Processor	Function to use	Rq
PLC5	ReadPLCData / WritePLCData[1]	Plc type=PLC
SLC500		Plc type=SLC
Micrologix	ReadPLCData / WritePLCData	Plc type=SLC
ControlLogix	ReadLgxData / WriteLgxData	
FlexLogix		

1. In this case, you are not obliged to open a connection

	ReadPLCData		WritePLCData	
	Connected	Unconnected	Connected	Unconnected
session	The session pointer return by the OpenSession function			
connection	Connection pointer return by ConnectPLCOver...	NULL	Connection pointer return by ConnectPLCOver...	NULL
dhp	A pointer to a Dh+ structure (if you use the ConnectPLCOverDHP function)else NULL			
routepath	NULL	Path to you logic controller	NULL	Path to you logic controller
routepathsize	0	Size of the routepath (in bytes)	0	Size of the routepath (in bytes)
type	Plc type (refers to the previous notice board)			
tns	Transaction number			
address	Address to identify the variables in your plc			
datatype	NA		Data type of what you want to write	
data	NA		Pointer to your data	
number	Number of variables you want to read / write			

	ReadLGXData	WriteLGXData
session	The session pointer return by the OpenSession function	
connection	Connection pointer return by ConnectPLCOver	
address	Address to identify the variables in your plc	
datatype	NA	Data type of what you want to write
data	NA	Pointer to your data
number	Number of variables you want to read / write	

6.4 Example

Here is a sample factory network, we will use it to explain the connection path.

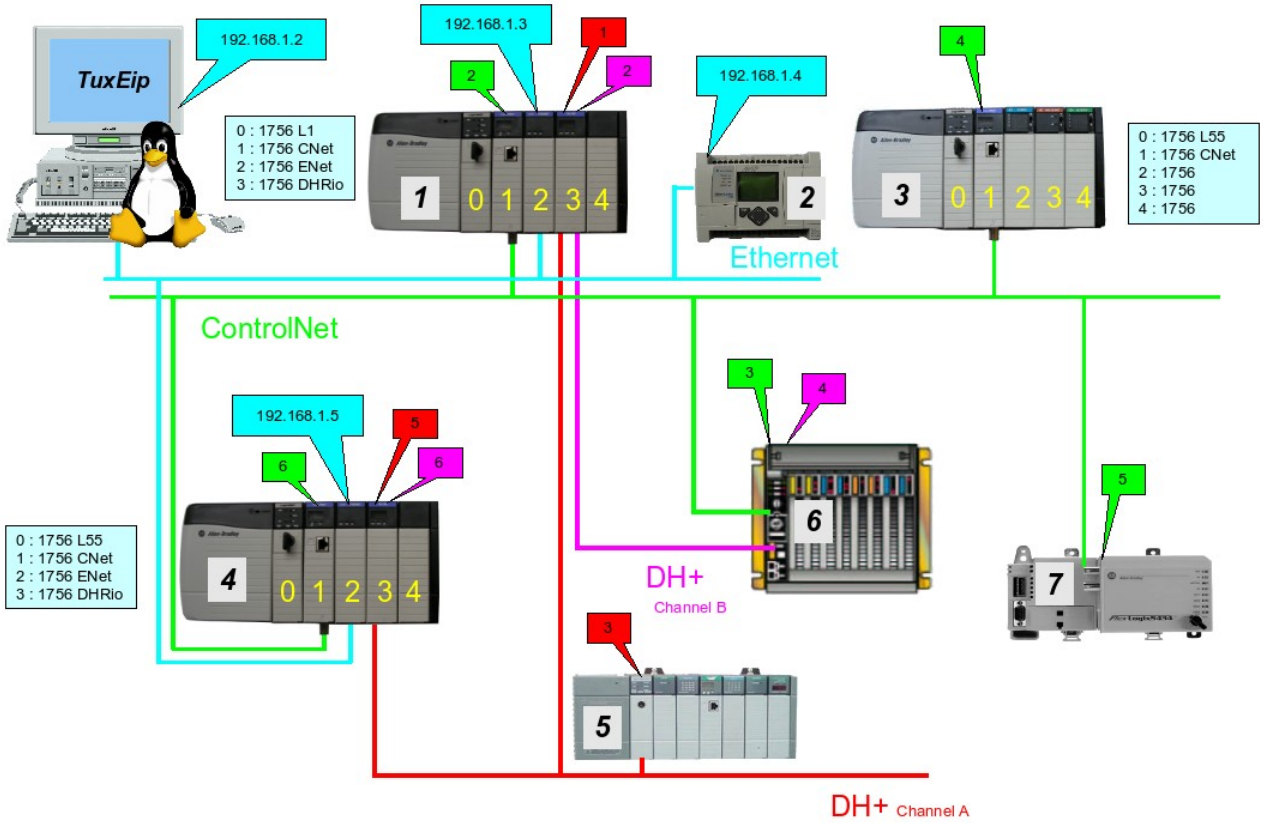


Illustration 3: Factory network

6.4.1 Example 1 : Accessing ControlLogix (Rack id 1,slot 0) via Ethernet :

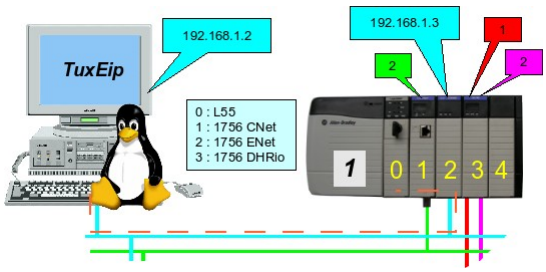


Illustration 4: example 1

IP of your 1756 ENET card	192.168.1.3
Backplane	1
L55 in slot 0	0
Path	192.168.1.3,1,0
Connection method	ConnectPLCOverCNET
Read / Write function	ReadLgxData / WriteLgxData

6.4.2 Example 2 : Accessing Micrologix (Rack id 2) via Ethernet :

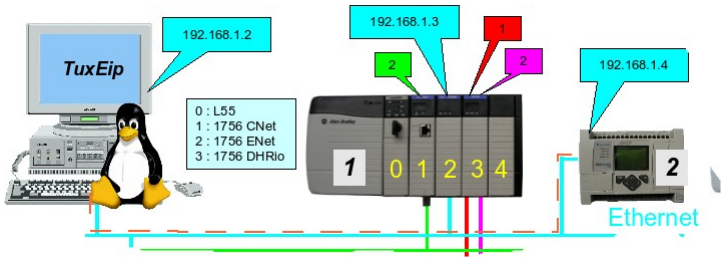


Illustration 5: example 2

IP of your Micrologix controller	192.168.1.4
Backplane	1
slot 0 (logic controller)	0
Path [1]	192.168.1.4,1,0
Connection method	ConnectPLCOverCNET
Read / Write function	ReadPLCData / WritePLCData
Plc type	SLC

1 The whole « Logix » controllers are build using the same scheme, communications card are plug into a backplane (sometime its a real one like ControlLogix's backplane, alternatively it is a virtual one like for Micrologix, FlexLogix)

6.4.3 Example 3 : Accessing PLC5 (Rack id 6) via ControlNet :

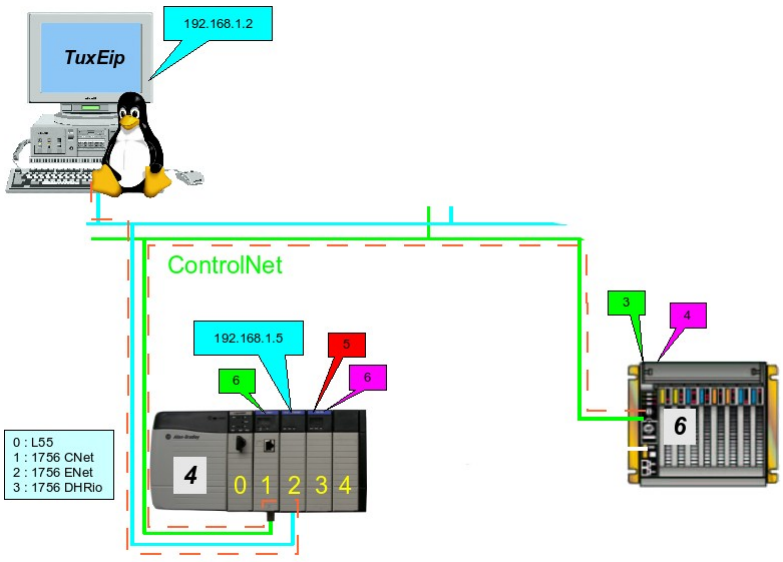


Illustration 6: example 3

As we can see in chapter 4 "TuxEip communication scheme" there is two way to access PLC5 and SLC500 logic controllers :

- Connected way

IP of your 1756 ENET card	192.168.1.5
Backplane	1
slot 1 (1756-CNET)	1
Port 2 (ControlNet network)	2
Node 3 (PLC5 node address)	3
Path	192.168.1.5,1,1,2,3
Connection method	ConnectPLCOverCNET
Read / Write function	ReadPLCData / WritePLCData
Plc type	PLC

- Unconnected way

  To Read / Write data, you will have to fill in the "routepath" and "routepathsize" (see section 6.3)

6.4.4 Example 4 : Accessing PLC5 (Rack id 6) via ControlNet and DH+ :

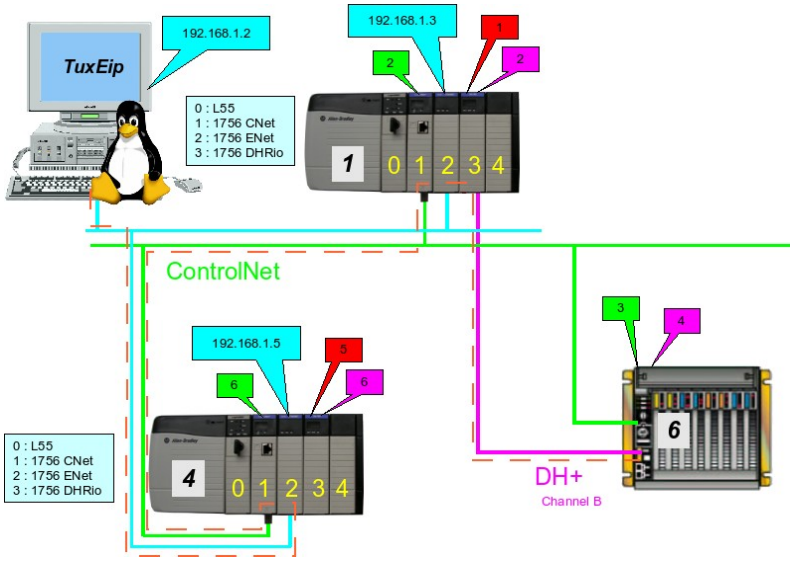
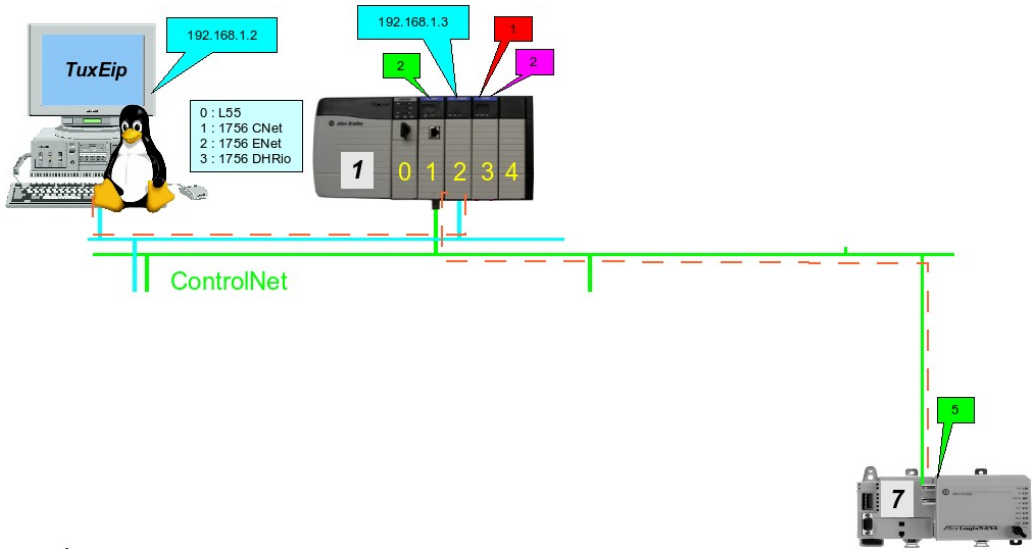


Illustration 7: example 4

IP of your 1756 ENET card		192.168.1.5
Backplane		1
slot 1 (1756-CNET)		1
Port 2 (ControlNet network)		2
Node 2 (1756-CNET Rack 1)		2
Backplane		1
slot 3 (1756-DHRIO)		3
Path		192.168.1.5,1,1,2,2,1,3
Connection method		ConnectPLCOverDHP (channel= Channel_B)
Read / Write function		ReadPLCData / WritePLCData
Plc type		PLC
dhp	Dest_link	0
	Dest_adress	4
	Src_link	0
	Src_adress	2

6.4.5 Example 5 : Accessing FlexLogix (Rack id 7) via ControlNet



IP of your 1756 ENET card	192.168.1.3
Backplane	1
slot 1 (1756-CNET)	1
Port 2 (ControlNet network)	2
Node 5 (1788-CNC)	5
Virtual Backplane of the FlexLogix	1
Virtual Slot of the FlexLogix UC	0
Path	192.168.1.3,1,1,2,5,1,0
Connection method	ConnectPLCOverCNET
Read / Write function	ReadLGXData / WriteLGXData