The University of Alabama in Huntsville
~ 08/01/2019 ~

ENIP.cpp
Reference Manual

A comprehensive documentation of ENIP.cpp during the period of its development phase.

The ENIP.cpp file executes all of the EtherNet/IP functions supported by the OpenPLC. Within this report, each functionality, code structure, and version updates will be detailed.

Table of Contents

Requirements

The ENIP.cpp file is run through OpenPLC. OpenPLC is an open-source Programmable Logic Controller developed by Thiago Alves at the University of Alabama in Huntsville. The project is dedicated to provide a low cost industrial solution for automation and research. This required software is free to download. More information regarding OpenPLC may be viewed here.

Struct Declarations

The received data from the network is parsed into structures by its attribute. The structures contain pointers to the specific byte where the attribute begins. Due to numerous types in which data may be presented, various structs were utilized to hold data within the respected EtherNet/IP format to uphold the network's integrity.
The struct declarations are contained within the header file **enipStruct.h**.

Note: *The number of bytes taken by each attribute within each struct is denoted by //[?]*

Enip_Header

This struct holds data that is similar for each EtherNet/IP type. From this struct the specific EtherNet/IP type and command code is determined.

```
struct enip_header
{
    unsigned char *command;//[2];
    unsigned char *length;//[2];
    unsigned char *session_handle;//[4];
    unsigned char *status;//[4];
    unsigned char *sender_context;//[8];
    unsigned char *options;//[4];
    unsigned char *data;
};
```

Enip_Data_Unknown

This struct holds parsed data corresponding to the Enip Type: Unknown.

```
struct enip_data_Unknown
{
    unsigned char *interface_handle;//[4]
    unsigned char *timeout;//[2]
    unsigned char *item_count;//[2]

    unsigned char *item1_id;//[2]
    unsigned char *item1_length;//[2]
    unsigned char *item1_data;//[1]

    unsigned char *item2_id;//[2]
    unsigned char *item2_length;//[2]
    unsigned char *item2_data;
};
```

Enip_Data_Unconnected

This struct holds parsed data corresponding to the Enip Type: Unconnected.

```
struct enip_data_Unconnected
{
    unsigned char *interface_handle;//[4]
    unsigned char *timeout;//[2]
    unsigned char *item_count;//[2]

    unsigned char *item1_id;//[2]
    unsigned char *item1_length;//[2]

    unsigned char *item2_id;//[2]
    unsigned char *item2_length;//[2]

    unsigned char *service;//[1]   0x4b (Request)
    unsigned char *request_pathSize;//[1]
    unsigned char *request_path;//[4]
    unsigned char *requestor_idLength;//[1]
    unsigned char *vendor_id;//[2]
    unsigned char *serial_number;//[4]
    unsigned char *data;
};
```

Enip_Data_Connected

This struct holds parsed data corresponding to the Enip Type: Connected and a command code of 0x6f.

```
struct enip_data_Connected
{
      unsigned char *interface_handle;//[4]
      unsigned char *timeout;//[2]
      unsigned char *item_count;//[2]

      unsigned char *item1_id;//[2]
      unsigned char *item1_length;//[2]

      unsigned char *item2_id;//[2]
      unsigned char *item2_length;//[2]

      unsigned char *service;//[1]
      unsigned char *request_pathSize;//[1]
      unsigned char *request_path;//[4]
      unsigned char *actual_timeout;//[2]
      unsigned char *o2t_netConnectID;//[4]
      unsigned char *t2o_netConnectID;//[4]
      unsigned char *connect_serialNo;//[2]
      unsigned char *orig_vendorNo;//[2]
      unsigned char *orig_serialNo;//[4]
      unsigned char *timeout_multiplier;//[1]
      unsigned char *reserved;//[3]
      unsigned char *o2t_rpi;//[4]
      unsigned char *o2t_netConnectParam;//[2]
      unsigned char *t2o_rpi;//[4]
      unsigned char *t2o_netConnectParam;//[2]
      unsigned char *transport_trigger;//[1]
      unsigned char *connection_pathSize;//[1]
      unsigned char *connection_path;//[?]
};
```

Enip_Data_Connected_0x70

This struct holds parsed data corresponding to the Enip Type: Connected and a command code of 0x70.

```
struct enip_data_Connected_0x70
{
      unsigned char *interface_handle;//[4]
      unsigned char *timeout;//[2]
      unsigned char *item_count;//[2]

      unsigned char *item1_id;//[2]
      unsigned char *item1_length;//[2]
      unsigned char *connection_id;//[4]

      unsigned char *item2_id;//[2]
      unsigned char *item2_length;//[2]
      unsigned char *sequence_count;//[2]

      unsigned char *service;//[1]
      unsigned char *request_pathSize;//[1]
      unsigned char *request_path;//[4]
      unsigned char *requestor_id;//[7]
      unsigned char *pcccData;//[?]
};
```

Functions

The ENIP.cpp file contains numerous functions to accomplish various tasks. Therefore, this section will be broken into different types as may be seen below.

- ***Hex to Uint16_t Conversion -*** *These functions convert lengths encoded in hex bytes into a variable type uint16_t. This is needed when calculations requiring lengths of data is needed.*
- ***Write to Output File -*** *These functions will write the current contents of a structure to an output file. This is a useful function for debugging.*
- ***Message Parsing -*** *These functions contain the code utilized to read data from the byte stream buffer into its corresponding struct variable.*
- ***Command Code Execution -*** *These functions contain the appropriate response message process for each supported command code.*
- ***Enip Type Selection -*** *These functions comprise the selection of the ENIP type*
- ***(Main Function) Message Processing -*** *This is the main function through which command functions are chosen to run. Upon entering the ENIP.cpp file, this is the first function that is executed.*

Hex to Uint16_t Conversion

get_HeaderLength()

```
uint16_t get_HeaderLength(struct enip_header *header)
```

Description

This function converts the structure *enip_header*'s *attribute length from a hex byte type to a uint16_t.*

Parameters

- **header -** Instance of struct object enip_header containing needed data size

Return

The size of enip_data_Unknown->item2_length as a uint16_t type

get_Item2_DataLength()

```
uint16_t get_Item2_DataLength(struct enip_data_Unknown *enipDataUnknown)
```

Description

This function converts the structure *enip_data_Unknown*'s *attribute item2_length from a hex byte type to a uint16_t.*

ENIP Type: *Unknown*

Parameters

- **enipDataUnknown -** Instance of struct object enip_data_Unknown containing needed data size

Return

The size of enip_data_Unknown->item2_length as a uint16_t type

get_Item2_DataLength_Unconnected()

```
uint16_t get_Item2_DataLength_Unconnected(struct enip_data_Unconnected *enipDataUnconnected)
```

Description

This function converts the structure *enip_data_Unconnected*'s *attribute item2_length from a hex byte type to a uint16_t.*

ENIP Type: *Unconnected*

Parameters

- **enipDataUnconnected -** Instance of struct object enip_data_Unconnected containing needed data size

Return

The size of enip_data_Unconnected->item2_length as a uint16_t type

get_Item2_DataLength_Connected_0x70()

```
uint16_t get_Item2_DataLength_Connected_0x70(struct enip_data_Connected_0x70 *enipDataConnected_0x70)
```

Description

This function converts the structure *enip_data_Connected_0x70*'s *attribute item2_length from a hex byte type to a uint16_t.*

ENIP Type: *Connected_0x70*

Parameters

- **enipDataConnected_0x70 -** Instance of struct object enip_data_Connected_0x70 containing needed data size

Return

The size of enip_data_Connected_0x70->item2_length as a uint16_t type

Write to Output File

The output files are utilized solely as a debugging tool for the user. These functions output each structure's contents to a designated output file. These files may be found within the outputFileFunctions.cpp file.

Note*: To change the output filename or where the file will be created, the string path variable on line 42 may be modified.*

```
string path = "C:\\Users\\hhanb\\outputFileOpenPLC.txt";
```

writeHeaderContents()

```
void writeHeaderContents(struct enip_header *header)
```

Description

This function will write the contents of the enip_header struct to a designated output file.

Parameters

- **header -** Instance of struct object enip_header desired to be outputted into a text document

Return

none

writeDataContents()

```
void writeDataContents(struct enip_data_Unknown *enipDataUnknown)
```

Description

This function will write the contents of the enip_data_Unknown struct to a designated output file.

Parameters

- **enipDataUnknown -** Instance of struct object enip_data_Unknown desired to be outputted into a text document

Return

none

Message Parsing

parseEnipHeader()

```
int parseEnipHeader(unsigned char *buffer, int buffer_size, struct enip_header *header, struct enip_data_Unknown *enipDataUnknown)
```

Description

This function parses the input socket data into the contents of the enip_header struct.

Parameters

- **buffer -** Unsigned character array containing input data from socket
- **buffer_size -** The size of the buffer as an integer value
- **header -** Instance of struct object enip_header to contain the buffer's header after parsing.
- **enipDataUnknown -** Instance of struct object enip_data_Unknown to contain the buffer's ENIP data after parsing.

Return

The size of the ENIP data as a uint16_t value

parseEnipUnknown()

```
void parseEnipUnknown(unsigned char *buffer, struct enip_data_Unknown *enipDataUnknown)
```

Description

This function parses the input socket data into the contents of the enip_data_Unknown struct.

Note: *This function is written to accommodate the Unknown ENIP type.*

Parameters

- **buffer -** Unsigned character array containing input data from socket
- **enipDataUnknown -** Instance of struct object enip_data_Unknown to contain the buffer's ENIP data after parsing.

Return

None

parseEnipUnconnected()

```
void parseEnipUnconnected(unsigned char *buffer, struct enip_data_Unconnected *enipDataUnconnected)
```

Description

This function parses the input socket data into the contents of the enip_data_Unconnected struct.

Note: *This function is written to accommodate the Unconnected ENIP type.*

Parameters

- **buffer -** Unsigned character array containing input data from socket
- **enipDataUnconnected -** Instance of struct object enip_data_Unconnected to contain the buffer's ENIP data after parsing.

Return

None

parseEnipConnected()

```
void parseEnipConnected(unsigned char *buffer, struct enip_data_Connected *enipDataConnected)
```

Description

This function parses the input socket data into the contents of the enip_data_Connected struct.

Note: *This function is written to accommodate the Connected ENIP type with a command code 0x6f.*

Parameters

- **buffer -** Unsigned character array containing input data from socket
- **enipDataConnected -** Instance of struct object enip_data_Connected to contain the buffer's ENIP data after parsing.

Return

none

ENIP Type Selection

getEnipType()

```
int getEnipType(struct enip_data enipData)
```

Description

This function returns an integer value denoting the ENIP type containing the PCCC data

ENIP Type: ***Unknown***	1
ENIP Type: ***Unconnected***	2
ENIP Type: ***Connected (0x6f and 0x70)***	3
ENIP Type: ***Unsupported***	-1 *(throws an error)*

Parameters

- **enipData -** The struct containing ENIP data that needs to be identified by type

Return

An integer representation of the ENIP type. The value will be either 1, 2, 3 or -1.

Command Code Execution

These functions contain the appropriate procedure to craft a response based upon the received command code.

registerEnipSession()

```
int registerEnipSession(struct enip_header *header)
```

Description

This function responds with a message that registers an ENIP session

Command Code: *0x65*

Parameters

- **header -** Instance of struct object enip_header containing currently received input data to execute the command

Return

The size of the response message in bytes as an integer value

sendRRData()

```
int sendRRData(int enipType, struct enip_header *header, struct enip_data_Unknown *enipDataUnknown, struct enip_data_Unconnected *enipDataUnconnected, struct enip_data_Connected *enipDataConnected)
```

Description

This function responds with a message that receives the input data request and replies with the queried information

Command Code: *0x6f*

Parameters

- **enipType -** Integer value denoting the ENIP Type
- **header -** Instance of struct object enip_header containing currently received input data to execute the command
- **enipDataUnknown -** Instance of struct object enip_data_Unknown containing currently received input data to execute the command
- **enipDataUnconnected -** Instance of struct object enip_data_Unconnected containing currently received input data to execute the command
- **enipDataConnected -** Instance of struct object enip_data_Connected containing currently received input data to execute the command

Return

The size of the response message in bytes as an integer value

sendUnitData()

```
int sendUnitData(struct enip_header *header, struct enip_data_Connected_0x70 *enipDataConnected_0x70)
```

Description

This function responds with a message that receives the input data request and replies with the queried information. This function is specifically used with the Connected ENIP Type.

Command Code: *0x70*

Parameters

- **header -** Instance of struct object enip_header containing currently received input data to execute the command
- **enipDataConnected_0x70 -** Instance of struct object enip_data_Connected_0x70 containing currently received input data to execute the command

Return

The size of the response message in bytes as an integer value

(Main Function) Message Processing

processEnipMessage()

Note: *As more functionality is added, the command code functions should be included within this function.*

```
int processEnipMessage(unsigned char *buffer, int buffer_size)
```

Description

This function contains the mechanism that controls which command code function will be executed.

Parameters

- **buffer -** Unsigned character array containing input data from socket
- **buffer_size -** The size of the buffer as an integer value

Return

The size of the response message in bytes as an integer value

Version Updates

Latest Update: Version 0.6

Version 0.1 (??/??/????)	The first version of ENIP.cpp documented featuring: ● Support of command code 0x65
Version 0.2 (05/16/2019)	Additional functionality added: ● Support of command code 0x6f with Enip Type: Unknown ● Output file functionality testing
Version 0.3 (05/30/2019)	Additional functionality added: ● Support of command code 0x6f with Enip Type: Unconnected ● Output file test for Type: Unconnected ● Added mechanism to determine which Enip Type is exhibited ● Added mechanism to select corresponding response based on Enip Type ● enip_Data struct implemented ● Some function parameters have been modified
Version 0.4 (06/28/2019)	Additional functionality added: ● Support of command code 0x6f with Enip Type: Connected ● Support of command code 0x70 with Enip Type: Connected ● Output file test for Type: Connected
Version 0.5 (07/09/2019)	Additional functionality added: ● Refactored command code functions ● Added support for PCCC.cpp to allow for variable and dynamic PCCC response messages ● Removed hard coded PCCC response
Version 0.6 (07/09/2019)	Additional functionality added: ● Refactored code into three files ○ enipStruct.h ○ outputFileFunctions.cpp ○ enip.cpp ●
Version 0.7 (08/01/2019)	Additional functionality added: ● *Work in Progress* ○ pccc.cpp ● pccc.cpp – adding functionality ○ Read/Write from PLC Address ○ Digital/Analog Read/Write ○ Craft complete response packet