Cloud PLC 4.0 API Manual

Contents

1. CPLC_Config_Digital_Outputs

Function Prototype	void CPLC_Config_Digital_Outputs(void)
Description	Configures digital pins D49, D48, D47 and D46 as OUTPUT
Parameters	None
Return Type	None
Function call Example	CPLC_Config_Digital_Outputs();

2. CPLC_Config_Digital_Inputs

FunctionPrototype	void CPLC_Config_Digital_Inputs(void)
Description	Configures digital pins D45, D44, D43, D42, D38, D41, D40 and D37 as INPUT
Parameters	None
Return Type	None
Function call Example	CPLC_Config_Digital_Inputs();

3. CPLC_Analog_Read

Function Prototype	int CPLC_Analog_Read(uint8_t pin)
Description	Read analog pin
Parameters	pin - Analog pin number
Return Type	ADC value
Function call Example	Int adcValue = CPLC_Analog_Read();

4. CPLC_Analog_Write

Function Prototype	void CPLC_Analog_Write(uint8_t pin, uint8_t value)
Description	Write analog pins
Parameters	pin - Analog pin number
Return Type	Value - Value to be written(0-255)
Function call Example	CPLC_Analog_Write (pin, 255);

5. CPLC_Config_Digital_IO

FunctionPrototype	void CPLC_Config_Digital_IO(uint8_t pin, uint8_t direction)
Description	Configures the port pin as digital output pin(D45, D44, D43, D42, D38, D41, D40 and D37)
Parameters	pin[IN] - One of the pin number mentioned above direction[IN] - OUTPUT/INPUT
Return Type	None
Function call Example	CPLC_Config_Digital_IO (D45, INPUT); CPLC_Config_Digital_IO (D45, OUTPUT);

6. CPLC_Digital_IO_Write

FunctionPrototype	void CPLC_Digital_IO_Write(uint8_t pin, uint8_t value)
Description	Writes the IO pins 49, 48, 47 and 46 as OUTPUT
Parameters	pin[IN] - One of the pin number mentioned above value[IN] - High/low
Return Type	None
Function call Example	CPLC_Digital_IO_Write (49, HIGH); //4 pin is high CPLC_Digital_IO_Write (49, LOW);

7. CPLC_Digital_IO_Write_HIGH

FunctionPrototype	void CPLC_Digital_IO_Write_HIGH(uint8_t pin)
Description	Writes HIGH value IO pins 49, 48, 47 and 46 as OUTPUT
Parameters	pin[IN] - One of the pin number mentioned above
Return Type	None
Function call Example	CPLC_Digital_IO_Write _HIGH (49);

8. CPLC_Digital_IO_Write_LOW

FunctionPrototype	void CPLC_Digital_IO_Write_LOW(uint8_t pin)
Description	Writes LOW value IO pins 49, 48, 47 and 46 as OUTPUT
Parameters	pin[IN] - One of the pin number mentioned above
Return Type	None
Function call Example	CPLC_Digital_IO_Write _ LOW (49);

9. CPLC_Digital_IO_Write_Toggle

FunctionPrototype	void CPLC_Digital_IO_Write_Toggle(uint8_t pin,uint8_t value)
Description	Toggles the value IO pins 49, 48, 47 and 46 as OUTPUT
Parameters	pin[IN] - One of the pin number mentioned above value[IN] - High/low
Return Type	None

Function call Example	CPLC_Digital_IO_Write _ Toggle (49,HIGH); //pin 4 is HIGH
	CPLC_Digital_IO_Write _ Toggle (49,LOW);

10. CPLC_Digital_IO_Read

Function Prototype	uint8_t CPLC_Digital_IO_Read(uint8_t pin)
Description	Reads the IO pin (Applicable to D45, D44, D43, D42, D38, D41, D40 and D37)
Parameters	pin[IN] - One of the pin number mentioned above
Return Type	Value - High/low
Function call Example	uint8_t Value = CPLC_Digital_IO_Read (D45);

11. sendATcommand

Function Prototype	static uint8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout)
Description	Function to send AT Command to GPRS
Parameters	ATcommand(IN) - AT command to be send expected_answer[IN] - Expected response (normally "OK") timeout[IN] - How long to wait for the response
Return Type	Status
Function call Example	uint8_t status = sendATcommand(“AT/r/n” , ”OK/r/n”, 2000);

12. CPLC_GSM_Send_SMS

Function Prototype	void CPLC_GSM_Send_SMS(char *numb, char *msg)
Description	Send sms to GSM device
Parameters	numb[IN] - Phone number msg{IN} - Message to be sent
Return Type	None
Function call Example	char num[] = “99xxxxxxxx”; char msg[] = “Hello World”; CPLC_GSM_Send_SMS (num, msg);

13. CPLC_GSM_Read_SMS

Function Prototype	void CPLC_GSM_Read_SMS(uint8_t index)
Description	Receive sms to GSM device
Parameters	index[IN] - Memory location number to read the message
Return Type	None
Function call Example	CPLC_GSM_Read_SMS ( );

14. CPLC_GSM_MakeCall

Function Prototype	void CPLC_GSM_MakeCall(char *numb)
Description	Make a Call
Parameters	numb[IN] - Phone number
Return Type	None
Function call Example	char num[] = “99xxxxxxxx” CPLC_GSM_MakeCall (num);

15. CPLC_Config_GPRS

Function Prototype	void CPLC_Config_GPRS(char* APN)
Description	Function to configure GPRS
Parameters	APN[IN] - APN of GSM network operation
Return Type	None
Function call Example	CPLC_Config_GPRS (“tp://airtelgprs.com”); NOTE: APN is shown for airtel sim,change the APN if other network operator is used.

16. CPLC_GPRS_Upload_FTP

Function Prototype	void CPLC_GPRS_Upload_FTP(char* apn, char* host, char* userName, char* password, char* fileName, char* folder)
Description	Function to Upload files on FTP server
Parameters	APN[IN] - APN of GSM network operation host[IN] - Host url userName[IN] - Username password[IN] - Password filename[IN] - FileName to upload the file folder[IN] - Folder name in local pc to fetch the file from
Return Type	None
Function call Example	char host[]="ftp://RDL.varmatrix.com"; char user[]="RDLwalabc123"; char pass[]="xxxxxxxx"; char file[]="12_7_2018_adc.csv"; char folder[]="test"; CPLC_GPRS_Upload_FTP (host, user, pass, file, folder);

17. CPLC_GPRS_Upload_JSON

Function Prototype	void CPLC_GPRS_Upload_JSON(String values, char* apn, char* url, char* ext, char* type)
Description	Function to Upload log data using JSON method
Parameters	data[IN] - Data to be pushed to server url[IN] - Url of the server ext[IN] - Url extension type[IN] - Application/json or application/x-www-form-urlencoded
Return Type	None
Function call Example	char data[]="\"id\":\"Hello world\" "; char url[]="iotpi.in"; char ext[]="/rfidwebserver/rfidinsert.php"); char type[]="application/json"; CPLC_GPRS_Upload_JSON (data, url, ext, type);

18. CPLC_GPRS_Upload_MQTT

Function Prototype	void CPLC_GPRS_Upload_MQTT(char* values, char* apn, char* host, char* port, char* username, char* password, char* topic)
Description	Function to Upload log data using MQTT protocol
Parameters	data[IN] - Data to be pushed to cloud host[IN] - MQTThost port[IN] - MQTTport number username[IN] - MQTTusername password[IN] - MQTTpassword topic[IN] - MQTTtopic
Return Type	None
Function call Example	char data[] =”Hello world”; char host[] =”m12.cloudmqtt.com”; char port[] =”11068”; char user[]=”iihbfshs”; char pass[]=”xxxxxxxxxxx”; char topic[]=”RDL”; CPLC_GPRS_Upload_MQTT (data, host, port, user, pass, topic);

19. CPLC_Modbus_Master_Read_Raw

Function Prototype	void CPLC_Modbus_Master_Read_Raw(uint8_t funCode, uint8_t slave_id, uint16_t start_addr, uint16_t len, uint16_t *data)
Description	This function sends data using modbus protocol
Parameters	funCode [IN] - Function code (readCoils/ readDiscreteInputs/ readInputRegisters/ readHoldingRegisters) slave_id [IN] - Slave id of the device start_addr [IN] - Starting address to read data frame len [IN] - Length of the data data [OUT] - Read data
Return Type	None
Function call Example	uint8_t data[50]={0}; CPLC_Modbus_Master_Read_Raw (2, 8, 40001, 5,data);

20. CPLC_SD_Init

Function Prototype	void CPLC_SD_Init(void)
Description	This function Initializes the SD card.
Parameters	None
Return Type	None
Function call Example	CPLC_SD_Init();

21. CPLC_SD_Write

Function Prototype	void CPLC_SD_Write(char *fileName, char *data)
Description	This function write data to SD card file
Parameters	filename[IN] - Filename to write to data{IN} -Data to be written in file
Return Type	None
Function call Example	CPLC_SD_Write ("test_RDL_DL.txt", "Hello World\n"); NOTE: max 255 bytes can be written once

22. CPLC_SD_Read

Function Prototype	void CPLC_SD_Read(char *fileName, char *data)
Description	This function Reads data from SD card file
Parameters	filename[IN] - Filename to write to data{Out} -Data to be written in file
Return Type	None
Function call Example	uint8_t data[50]={0}; CPLC_SD_Read ("test_RDL_DL.txt", data); NOTE: max 255 bytes can be read once

23. CPLC_RTC_Set_Time

Function Prototype	void CPLC_RTC_Set_Time(uint16_t year, uint8_t month, uint8_t date, uint8_t hour, uint8_t min, uint8_t sec)
Description	This function Sets RTC date and Time
Parameters	Year[IN] - Year Month [IN] - Month of the year Date[IN] - Date of the month Hour[IN] - Hour Min[IN] - Minute Sec[IN] - Second
Return Type	None
Function call Example	CPLC_RTC_Set_Time (2018, 11, 16, 11, 57, 2);

24. CPLC_RTC_Init

Function Prototype	void CPLC_RTC_Init(void)
Description	This function Initializes RTC
Parameters	None
Return Type	None
Function call Example	CPLC_RTC_Init() ;

25. CPLC_RTC_Get_Time

Function Prototype	DateTime CPLC_RTC_Get_Time(void)
Description	This function Sets RTC date and Time
Parameters	None
Return Type	DateTime
Function call Example	CPLC_RTC_Get_Time();

26. CPLC_FRAM_Read

Function Prototype	void CPLC_FRAM_Read (uint16_t framAddr, uint8_t* data, uint8_t len)
Description	Function to read data from FRAM
Parameters	framAddr[IN] - 16 bit address to read the data from data[OUT] - Pointer to the data read len[IN] - Number of bytes to read
Return Type	None
Function call Example	uint8_t data[50] ={0}; CPLC_FRAM_Read (0x0010, data, 10);

27. CPLC_FRAM_Write

Function Prototype	void CPLC_FRAM_Write(uint16_t framAddr, uint8_t* data, uint8_t len)
Description	Function to store data in FRAM
Parameters	data[IN] -Pointer to the data to be written framAddr[IN] - 16 bit address to write the data to
Return Type	None
Function call Example	uint8_t data =”Hello world”; CPLC_FRAM_Write (data, 0x0010);

28. CPLC_PWM_Write

Function Prototype	void CPLC_PWM_Write(uint8_t pin, uint8_t ms)
Description	Function for PWM – fading
Parameters	pin[IN] -configures D4; ms[IN] - Delay in ms
Return Type	None
Function call Example	CPLC_PWM_Write (D4,30);

29. CPLC_WBflash_init

Function Prototype	bool CPLC_WBflash_init()
Description	Function to Init Flash
Parameters	None
Return Type	Status
Function call Example	bool status =0; status = CPLC_WBflash_init ();

30. CPLC_WBflash_Read_JDEC_ID

Function Prototype	void CPLC_WBflash_Read_JDEC_ID(uint8_t *buf)
Description	Function to read JDEC id
Parameters	Buf[OUT] - Pointer to hold JDEC ID
Return Type	None
Function call Example	uint8_t JDEC_Id[10]={0}; CPLC_WBflash_Read_JDEC_ID (JDEC_Id);

31. CPLC_WBflash_sram_write

Function Prototype	void CPLC_WBflash_sram_write(uint32_t addr,uint8_t *buf, uint32_t data_length)
Description	Function to write into Flash
Parameters	addr[IN] - address where data to be written Buf[IN] - Pointer to data to be written data_length[IN] - Length of data to be written
Return Type	None
Function call Example	uint8_t buff[]=”Hello world”; CPLC_WBflash_sram_write (0x00000010,buff,strlen(buff));

32. CPLC_WBflash_wait

Function Prototype	static int CPLC_WBflash_wait(void)
Description	Function to wait for data write/erase to complete
Parameters	None
Return Type	Status
Function call Example	Int status =0; Status = CPLC_WBflash_wait ();

33. CPLC_WBflash_sram_read_data

Function Prototype	void CPLC_WBflash_sram_read_data(uint32_t addr, uint32_t len, bool ischar)
Description	Function to read data
Parameters	Addr[IN] - Address to read data from len[IN] - Length of data to be read ischar - true for ascii values, false for hex bytes
Return Type	None
Function call Example	uint8_t buff[50] ={0}; CPLC_WBflash_sram_read_data (0x00000010, buff, 10);

34. CPLC_WBflash_blockErase4K

Function Prototype	void CPLC_WBflash_blockErase4K(uint32_t addr)
Description	Function to erase data in 4k blocks(sector)
Parameters	Addr[IN] - Address to erase the sector
Return Type	None
Function call Example	CPLC_WBflash_blockErase4K (0x00000000);

35. CPLC_WBflash_Chip_Erase

Function Prototype	void CPLC_WBflash_Chip_Erase(void)
Description	Function to erase the entire chip
Parameters	None
Return Type	None
Function call Example	CPLC_WBflash_Chip_Erase ();

36. CPLC_Serial_Read_Byte

Function Prototype	char CPLC_Serial_Read_Byte(void)
Description	Function to read the data
Parameters	None
Return Type	Byte of data
Function call Example	char inByte=0; inByte = CPLC_Serial_Read_Byte ();

37. CPLC_Serial_Write_Byte

Function Prototype	void CPLC_Serial_Write_Byte(uint8_t byte)
Description	Function to read and write serial data
Parameters	Byte[IN] - 1 Byte to write
Return Type	None
Function call Example	CPLC_Serial_Write_Byte ('H');

38. CPLC_SPI_Read

Function Prototype	uint8_t CPLC_SPI_Read(uint32_t clock, uint8_t bitOrder, uint8_t dataMode, uint8_t slavePin)
Description	Function to Read SPI data
Parameters	Clock[IN] - Max clock frequency(speed) of spi BitOrder[IN] - MSBFIRST/LSBFIRST dataMode[IN] - SPI_MODE0/SPI_MODE1/SPI_MODE2/SPI_MODE3 SlavePin[IN] - Chip select/ Slave select pin
Return Type	Read data
Function call Example	CPLC_SPI_Read (14000000,MSBFIRST,SPI_MODE0,D10);

39. CPLC_SPI_Write

Function Prototype	void CPLC_SPI_Write(uint32_t clock, uint8_t bitOrder, uint8_t dataMode, uint8_t data, uint8_t slavePin)
Description	Function to Write data on SPI bus
Parameters	clock[IN] - Max clock frequency(speed) of spi bitOrder[IN] - MSBFIRST/LSBFIRST dataMode[IN] - SPI_MODE0/SPI_MODE1/SPI_MODE2/SPI_MODE3 data[IN] - Data to be written slavePin[IN] - Chip select/Slave select pin
Return Type	None
Function call Example	CPLC_SPI_Write (14000000,MSBFIRST,SPI_MODE0,'H', D10);

40. CPLC_I2C_Read

Function Prototype	uint8_t CPLC_I2C_Read(uint8_t slave_id, uint8_t len, uint8_t* data)
Description	Function to Read I2C data
Parameters	slave_id[IN] - Slave device id len[IN] - Number of bytes to read data[OUT] - Pointer to data read
Return Type	Read data
Function call Example	uint8_t data[10] ={0}; CPLC_I2C_Read (12, 5, data);

41. CPLC_I2C_Write

Function Prototype	void CPLC_I2C_Write(uint8_t slave_id, uint8_t len, uint8_t* data)
Description	Function to Write data on I2C bus
Parameters	slave_id [IN] - Slave device id len[IN] - Number of bytes to be written data[IN] - Pointer to Data to be written
Return Type	None
Function call Example	Uint8_t data[]="Hello world"; CPLC_I2C_Write (12, strlen(data), data);

42. CPLC_Reset

Function Prototype	void CPLC _Reset(uint8_t resetPin)
Description	Function for Soft Reset on Arduino
Parameters	ResetPin[IN] - Pin number which is 36
Return Type	None
Function call Example	CPLC_Reset (36);

43. CPLC_PWDT_Init

Function Prototype	void CPLC_PWDT_Init(void)
Description	Function for Physical Watch Dog Timer Initialization
Parameters	None
Return Type	None
Function call Example	CPLC_PWDT_Init (); NOTE: minimum 0,maximum 2mins

44. CPLC_PWDT_Reset

Function Prototype	void CPLC_PWDT_Reset(uint32_t timeout)
Description	Function to reset Physical Watch Dog Timer
Parameters	Timeout - after which a pulse must be given
Return Type	None
Function call Example	CPLC_PWDT_Reset ();

45. CPLC_Pulse_Count

Function Prototype	uint8_t CPLC_Pulse_Count(uint8_t pin_irq)
Description	Function for Physical Watch Dog Timer
Parameters	Pin - Interrupt Pin number
Return Type	Count
Function call Example	CPLC_Pulse_Count ();

46. CPLC_EEPROM_Write

Function Prototype	void CPLC_EEPROM_Write(uint8_t addr, uint8_t val)
Description	Function for writing data into EEPROM
Parameters	Addr[IN] - Address to write data Val[IN] - Value or data to write
Return Type	None
Function call Example	CPLC_EEPROM_Write (100, 20);

47. CPLC_EEPROM_Read

Function Prototype	uint8_t CPLC_EEPROM_Read(uint8_t addr)
Description	Function for reading data from EEPROM
Parameters	Addr[IN] - Address to write data val[IN] - Value or data to write
Return Type	Byte of data
Function call Example	CPLC_EEPROM_Read (100);

48. CPLC_Modbus_Master_Init

Function Prototype	void CPLC_Modbus_Master_Init(void)
Description	This function Inits the modbus RTU
Parameters	None
Return Type	None
Function call Example	CPLC_Modbus_Master_Init ();

49. CPLC_GSM_Init

Function Prototype	void CPLC_GSM_Init(void)
Description	Function to power on GSM
Parameters	None
Return Type	None
Function call Example	CPLC_GSM_Init ();

50. CPLC_Induino_Init_I2C_Master

Function Prototype	void CPLC_Induino_Init_I2C_Master(void)
Description	This function De Initializes the SPI interface in Master board
Parameters	None
Return Type	None
Function call Example	CPLC_Induino_Init_I2C_Master ();

51. CPLC_Induino_Init_I2C_Slave

Function Prototype	void CPLC_Induino_Init_I2C_Slave(void)
Description	This function Initializes the Induino board as I2C Slave
Parameters	None
Return Type	None
Function call Example	CPLC_Induino_Init_I2C_Slave ();

52. CPLC_I2C_Master_Read

Function Prototype	double CPLC_I2C_Master_Read(int slaveAddr, uint8_t len)
Description	This function Reads data from I2C Slave
Parameters	len[IN] - number of bytes to read
Return Type	Thermo Couple Value
Function call Example	CPLC_I2C_Master_Read (12, strlen(data));

53. CPLC_Ethernet_MQTT_Publish_Message

Function Prototype	void CPLC_Ethernet_MQTT_Publish_Message(char* mqttServer, int mqttPort, char* mqttUser, char* mqttPassword, char* mqttTopic, char* mqttData)
Description	This function publishes data on mqtt server
Parameters	mqttServer[IN] - server/host url mqttPort[IN] - port number mqttUser[IN] - username mqttPassword[IN] - password mqttTopic[IN] - topic where data to be to published mqttData[IN] - data/message to be published
Return Type	None
Function call Example	Char Server[]=137.188.116.12 Char Port[]= ”Hello world” Char User[]= ”RDLabc123” Char Password[]=”xxxxxxxx” Char Topic[]=”RDL” Char Data[]=”Hello world”; CPLC_Ethernet_MQTT_Publish_Message(Server,Port,User,Password,Topic,Data);

54. CPLC_Ethernet_JSON_Send_Message

Function Prototype	void CPLC_Ethernet_JSON_Send_Message(char* jsonServer, char* jsonExtn, char* jsonType, char* data)
Description	This function publishes data on mqtt server
Parameters	jsonServer[IN] - server/host url jsonExtn[IN] - url extension jsonType[IN] - content type data[IN] - data/message to be uploaded
Return Type	None
Function call Example	Char Server[]=137.188.116.12 Char Extention[]=”/vvv/post/postgastempxx.php?” Char Type[]=String Char Data[]=”Hello world” CPLC_Ethernet_JSON_Send_Message(Server, Extention, Type,Data);

55. CPLC_Ethernet_SNMP_Init

Function Prototype	void CPLC_Ethernet_SNMP_Init (void)
Description	This function Inits ethernet over SNMP
Parameters	None
Return Type	None
Function call Example	CPLC_Ethernet_SNMP_Init ();

56. CPLC_Ethernet_SNMP_Test

Function Prototype	void CPLC_Ethernet_SNMP_Test (void)
Description	This function tests ethernet over SNMP
Parameters	None
Return Type	None
Function call Example	CPLC_Ethernet_SNMP_Test ();

57. _MCP23017_Init_IO

Function Prototype	void CPLC_IO_Expander_MCP23017_Init_IO(uint8_t pin, uint8_t direction)
Description	This function Initializes the IO expander board(MCP23017) for communication though RDL bus via I2C
Parameters	pin[IN] - pin number to be configured direction[IN] - INPUT/OUTPUT
Return Type	None
Function call Example	CPLC_IO_Expander_MCP23017_Init_IO (49,OUTPUT); CPLC_IO_Expander_MCP23017_Init_IO (D45,INPUT);

58. CPLC_IO_Expander_MCP23017_Init_Interrupt

Function Prototype	void CPLC_IO_Expander_MCP23017_Init_Interrupt(uint8_t pin)
Description	This function Initializes the IO expander board(MCP23017) for communication though RDL bus via I2C
Parameters	pin[IN] - pin number to be configured(D2,D6,D7,D8) direction[IN] - INPUT/OUTPUT
Return Type	None
Function call Example	CPLC_IO_Expander_MCP23017_Init_Interrupt (2);

59. CPLC_IO_Expander_MCP23017_write

Function Prototype	void CPLC_IO_Expander_MCP23017_write(uint8_t pin, uint8_t value)
Description	This function Reads IO pins(MCP23017) though RDL bus via I2C
Parameters	pin[IN] - pin number to be configured value[IN] - HIGH/LOW
Return Type	None

Function call Example	CPLC_IO_Expander_MCP23017_write (D49,HIGH);//D4 output pin HIGH
	CPLC_IO_Expander_MCP23017_write (D45,HIGH);//D2 input pin HIGH

60. CPLC_IO_Expander_MCP23017_Read

Function Prototype	uint8_t CPLC_IO_Expander_MCP23017_Read(uint8_t pin)
Description	This function Reads IO pins(MCP23017) though RDL bus via I2C
Parameters	pin[IN] - pin number to be configured(D2,D6,D7,D8)
Return Type	pin value
Function call Example	CPLC_IO_Expander_MCP23017_Read (D2); CPLC_IO_Expander_MCP23017_Read (D6);

61. mcp23017_ISR

Function Prototype	void mcp23017_ISR(void)
Description	MCP23017 interrupt routine handles the button press since this is the only active interrupt
Parameters	None
Return Type	None
Function call Example	void mcp23017_ISR ();

62. CPLC_ESP32_HTTP_SetUp

Function Prototype	void CPLC _ESP32_HTTP_SetUp(const char* ssid, const char* password)
Description	This function Sets up HTTP connection
Parameters	ssid - your Network Name password - your Network Password
Return Type	Char ssid[] = RDL_ADMIN Char Password[] =RDL123
Function call Example	CPLC _ESP32_HTTP_SetUp (ssid, password);

63. CPLC _ESP32_HTTP_Get

Function Prototype	void CPLC _ESP32_HTTP_Get(const char* Url)
Description	This function does HTTP Get request from the url provided
Parameters	url - url from where HTTP get request needs to be made
Return Type	char url[]=”iotpi.in”;
Function call Example	CPLC _ESP32_HTTP_ Get (url);

64. CPLC_Stack_Push

Function Prototype	void CPLC_Stack_Push(byte *data, uint8_t len)
Description	This function pushes data on stack
Parameters	Data - data to be pushed len - number of bytes to push
Return Type	None
Function call Example	CPLC_Stack_Push (byte*0A,uint8_t 20);

65. CPLC_Stack_Pop

Function Prototype	byte* CPLC_Stack_Pop(uint8_t len)
Description	This function pops data from stack
Parameters	len{IN} - number of bytes to pop
Return Type	None
Function call Example	CPLC_Stack_Pop (uint8_t 20);

66. CPLC_ThermoCouple_Init

Function Prototype	void CPLC_ThermoCouple_Init(void)
Description	Initialization function for ThermoCouple Sensor(MAX31855K)
Parameters	None
Return Type	None
Function call Example	CPLC_ThermoCouple_Init ();

67. CPLC_ThermoCouple_Read

Function Prototype	void CPLC_ThermoCouple_Init(void)
Description	Reads ThermoCoupler(MAX31855) via SPI
Parameters	None
Return Type	temperature value in double(8 bytes)
Function call Example	CPLC_ThermoCouple_Read ();

68. CPLC_Arithmetic_Compare_LessThan

Function Prototype	bool CPLC_Arithmetic_Compare_LessThan(uint8_t num1, uint8_t num2)
Description	This function compares if num1 is less than num2
Parameters	num1 - number1 num2– number2
Return Type	true if num1 < num2
Function call Example	num1=5 num2=6 CPLC_Arithmetic_Compare_LessThan(num1 , num2);

69. CPLC_Arithmetic_Compare_LessThanOrEqualTo

Function Prototype	bool CPLC_Arithmetic_Compare_LessThanOrEqualTo(uint8_t num1, uint8_t num2)
Description	This function compares if num1 is less than or equal to num2
Parameters	num1 - number1 num2– number2
Return Type	true if num1 <=num2
Function call Example	num1=5 num2=6 CPLC_Arithmetic_Compare_LessThanOrEqualTo (num1 ,num2);

70. CPLC_Arithmetic_Compare_GreaterThan

Function Prototype	bool CPLC_Arithmetic_Compare_GreaterThan(uint8_t num1, uint8_t num2)
Description	This function compares if num1 is Greater than num2
Parameters	num1 - number1 num2– number2
Return Type	true if num1 >num2
Function call Example	num1=4 num2=6 CPLC_Arithmetic_Compare_GreaterThan(num1 ,num2);

71. CPLC_Arithmetic_Compare_GreaterThanOrEqualTo

Function Prototype	bool CPLC_Arithmetic_Compare_GreaterThanOrEqualTo(uint8_t num1, uint8_t num2)
Description	This function compares if num1 is Greater than or equal to num2
Parameters	num1 - number1 num2– number2
Return Type	true if num1 >=num2
Function call Example	num1=4 num2=7 CPLC_Arithmetic_Compare_GreaterThanOrEqualTo(num1, num2);

72. CPLC_Arithmetic_Compare_EqualsTo

Function Prototype	bool CPLC_Arithmetic_Compare_EqualsTo(uint8_t num1, uint8_t num2)
Description	This function compares if num1 is equal to num2
Parameters	num1 - number1 num2– number2
Return Type	true if num1 ==num2
Function call Example	num1=4 num2=4 CPLC_Arithmetic_Compare_EqualsTo(num1, num2);

73. CPLC_Logical_ShiftRight

Function Prototype	uint8_t CPLC_Logical_ShiftRight(uint8_t data, uint8_t shift)
Description	This function shifts the data to the right.
Parameters	data - data to be shifted shift - number to be shifted right
Return Type	None
Function call Example	CPLC_Logical_ShiftRightt();

74. CPLC_Logical_ShiftLeft

Function Prototype	uint8_t CPLC_Logical_Shiftleft(uint8_t data, uint8_t shift)
Description	This function shifts the data to the left.
Parameters	data - data to be shifted shift - number to be shifted left
Return Type	None
Function call Example	CPLC_Logical_Shift left ();

75. CPLC_Delay

Function Prototype	void CPLC_Delay(unsigned long delay_ms)
Description	This function sets the delay
Parameters	Delay_ms – delay in millisecond
Return Type	None
Function call Example	CPLC_Delay (1000);

76. CPLC_Arithmetic_Add

Function Prototype	Uint16_t CPLC_Arithmetic_Add(uint8_t num1, uint8_t num2)
Description	This function adds num1 with num2
Parameters	num1 - number1 num2– number2
Return Type	Num1+num2
Function call Example	CPLC_Arithmetic_Add ();

77. CPLC_Arithmetic_Subtract

Function Prototype	Uint8_t CPLC_Arithmetic_Subtract(uint8_t num1, uint8_t num2)
Description	This function subtracts num1 with num2
Parameters	num1 - number1 num2– number2
Return Type	Num1-num2
Function call Example	CPLC_Arithmetic_ Subtract ();

78. CPLC_Arithmetic_Multiply

Function Prototype	Uint8_t CPLC_Arithmetic_Multiply(uint8_t num1, uint8_t num2)
Description	This function Multiplies num1 with num2
Parameters	num1 - number1 num2– number2
Return Type	Num1*num2
Function call Example	CPLC_Arithmetic_ Multiply ();

79. CPLC_Arithmetic_Divide

Function Prototype	Uint8_t CPLC_Arithmetic_Divide(uint8_t num1, uint8_t num2)
Description	This function divides num1 with num2
Parameters	num1 - number1 num2– number2
Return Type	Num1/num2
Function call Example	CPLC_Arithmetic_ Divide ();

80. CPLC_Condition_Wait_Serial_String

Function Prototype	void CPLC_Condition_Wait_Serial_String(String str)
Description	This function waits as long as the str matches with serial buffer
Parameters	str[IN] - string that needs to be matched with
Return Type	None
Function call Example	CPLC_Condition_Wait_Serial_String ();

81. CPLC_Condition_Wait_IO

Function Prototype	void CPLC_Condition_Wait_IO(uint8_t pin, uint8_t value)
Description	This function waits as long as the digital pin is high
Parameters	pin[IN] - pin number to be configured value[IN] - high/low
Return Type	None
Function call Example	void CPLC_Condition_Wait_IO ();

82. CPLC_Conditional_IF

Function Prototype	void CPLC_Conditional_IF(bool condition, func_Category_t fCat1, func_Code_t fCode1, void *params1, func_Category_t fCat2, func_Code_t fCode2, void *params2)
Description	This function executes conditional statements
Parameters	condition - true/false fCat1 - function category of function 1 on if condition = true fCode1 - function code of function 1 on if condition = true params1 - pointer to structure holding funnction parameters of function 1 on if condition = true fCat2 - function category of function 2 on if condition = false fCode2 - function code of function 2 on if condition = false params2 - pointer to structure holding funnction parameters of function 2 on if condition = false
Return Type	None
Function call Example	void CPLC_Conditional_IF ();

83. CPLC_IO_Expander_MCP23017_Write_Toggle

Function Prototype	Void CPLC_IO_Expander_MCP23017_Write_Toggle(uint8_t pin, uint8_t value)
Description	This function inverts the value on IO pin(MCP23017) through RDL bus via I2C
Parameters	pin[IN] - pin number to be configured value[IN] - HIGH/LOW
Return Type	None
Function call Example	Void CPLC_IO_Expander_MCP23017_Write_Toggle ();

84. Sample Code :

MQTT using QUECTEL M95 GSM Modem:

```
#includ <CPLC.h>
char data[] = "Hello RDL111!!";
char* host = "m12.cloudmqtt.com";
//char* host = "52.3.184.147";
char* port = "11068";
char* username = "iihbfshs";
char* password = "b_LmA_W7_kZl";
char* topic = "ethernet";
//char *APN = "tp://airtelgprs.com"; //for airtel sim
//char *APN = "internet"; //for Idea sim
char *APN = "TATA.DOCOMO.INTERNET"; //for DoCoMo sim
void setup(){
Serial.begin(9600);
Serial.println("Start..");
CPLC_GPRS_Upload_MQTT(data, APN, host, port, username, password, topic);
}
void loop()
{
}
```

Modbus Code:

```
#include <CPLC.h>
/* Reading the device MFR 2810 */
uint8_t funCode = 0x04; //read holding register
uint8_t slave_id = 1;
uint16_t start_addr = 3850; //address to read Line to Neutral Voltage for
device MFR 2810
uint16_t len = 1; //datatype is 32-bit Float for MFR 2810
uint32_t data[110] = {0};
uint8_t result = 0;

void setup()
{
  DL_Modbus_Master_Init();
}

void loop()
{
  Serial.print("Line to Neutral Voltage(raw value) = ");
  CPLC_Modbus_Master_Read_Raw(funCode, slave_id, start_addr, len, data);
  delay(2000);
}
```