Easy Master user manual

Document revision 1.0   27/04/2022

EasyCAT

A B & T
Tecnologie Informatiche

http://www.easycatshield.com/
http://www.bausano.net/

1 – The Easy Master

The Easy Master is an Arduino library that implements a very basic EtherCAT® master, mainly intended for testing and experimentation with the EasyCAT line of boards.
It has been designed to require very limited resources, particulary in term of RAM, to enable it to run on an Arduino Uno that only has 2K RAM, partly already occupied.

To do this it cannot fully comply with the EtherCAT® specifications, and therefore the communication is limited to the cyclic PDO data exchange and does not support acyclic SDO commands through the mailboxex.
If instead you need a master that is fully compliant with the EtherCAT® specs see the Appendix.

2 – System requirement

An Arduino UNO, or MEGA, ZERO, DUE.

An Ethernet 2 shield or equivalent, using the WIZ5500 ethernet controller, to provide access to the network. No ethernet library is required as the communication functions are embedded into the Easy Master.

The Easy Navigator V 2.0 or upper, to generate the *NetworkInfo.h* file.

3 – Using the Easy Master

The first step is obviously to install the Easy Master library, using the standard procedure, and to create an Arduino project that will contain the master application.
Then we have to create the *NetworkInfo.h* file.

3.1 – The NetworkInfo.h file

This is the file on which the Easy Master rely to know how the EtherCAT® network is composed and therefore how to configure the slaves and how to exchange data with them.
It is created scanning the EtherCAT® network with the Easy Navigator: for information how to do this please see the Easy Navigator user manual.

The *NetworkInfo.h* file is divided in 6 parts:

1) *Network size* – This parts includes the number of slaves that must be present on the network and the total value of the working counter expected from them. These values are checked at startup and at run time.

2) *Slaves names* – These are the names of the slaves that must be connected to the network. At startup the master will extract the name of each slave from its EEPROM and compare it with the expected one.

3) *Entries of the input PDOs* – This part contains the structures with the names and data types of the input variables, i.e. the data that the master can read from the slaves.
We must use these names in our applications to read data from the slaves.

4) *Entries of the output PDOs* – This is the same of the previous part but related to the output variables, i.e. the data that the master can write to the slaves.
We must use these names in our applications to write data to the slaves.

5) *SM configuration* – These are the data that the master uses to configure the slaves sinchronization managers, during startup.

6) *FMMU configuration* – These are the data that the master uses to configure the slaves field memory manager units, during startup.

Once we have created this file we have to copy it in the Arduino project folder of the master application, next to the *.ino* file. The file doesn't require any user intervention so please don't edit it.

3.2 – Library interface

The user application interfaces with the library through three functions.

1) *Init (uint8_t SampTime)*
Initializes the the library and the ethernet interface, performs a network scan to check that all expected slave are present, initializes them and starts the cyclic sampling of the network. **(Fig. 2)**
The parameter *SampTime* is the network sampling time in mS: accepted value are from 1 to 100.

2) *StartApp ()*
Synchronizes the user application: it is mandatory to call this function at the beginning of the user application. **(Fig.3)**

3) *EndApp ()*
Synchronizes the user application: it is mandatory to call this function at the end of the user application. **(Fig.3)**

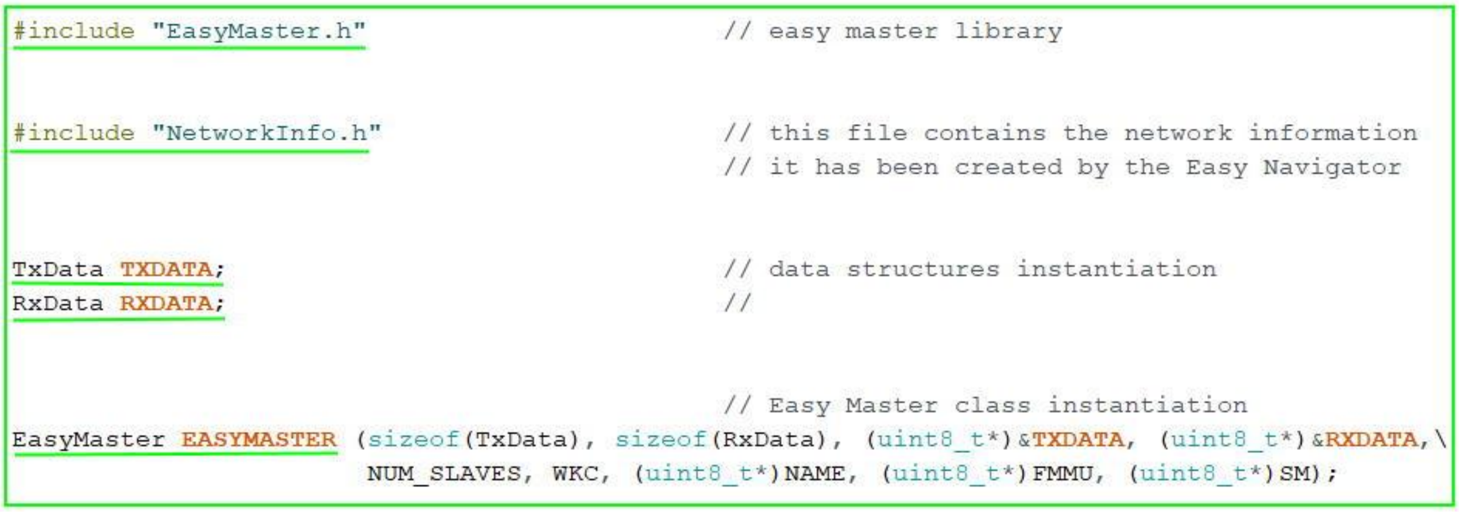
Please note that the data exchange between the master and the slaves occures in the interrupt function included in the library and consequently it is not necessary to call any function to do this.

3.3 – The user application

In the source code of the user application we have to do the following things in order to exchange data with the EtherCAT® slaves.

At the beginning.

- Include the *EasyMaster.h* file to access the library.
- Include the NeworkInfo.h file to have all the information needed about the network.
- Instantiate the TXDATA and RXDATA structures that give us access to the data to exchange with the slaves.
- Instantiate the EASYMASTER class.

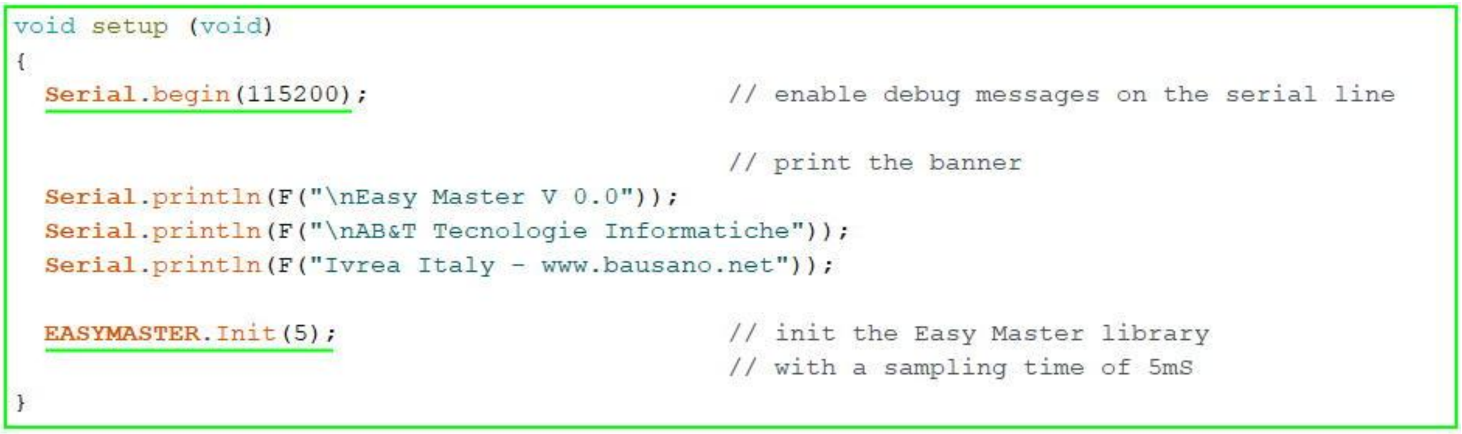


```
#include "EasyMaster.h"                          // easy master library


#include "NetworkInfo.h"                         // this file contains the network information
                                                 // it has been created by the Easy Navigator


TxData TXDATA;                                   // data structures instantiation
RxData RXDATA;                                   //


                                                 // Easy Master class instantiation
EasyMaster EASYMASTER (sizeof(TxData), sizeof(RxData), (uint8_t*)&TXDATA, (uint8_t*)&RXDATA,\
                       NUM_SLAVES, WKC, (uint8_t*)NAME, (uint8_t*)FMMU, (uint8_t*)SM);
```

Fig. 1 – Directives and definitions

In the Setup() function.

- Init the serial line. This enables same useful debug messages to be printed out on the serial line.
- Init the Easy Master library with the network sampling time in mS as a parameter. Accepted values are from 1 to 50 mS.

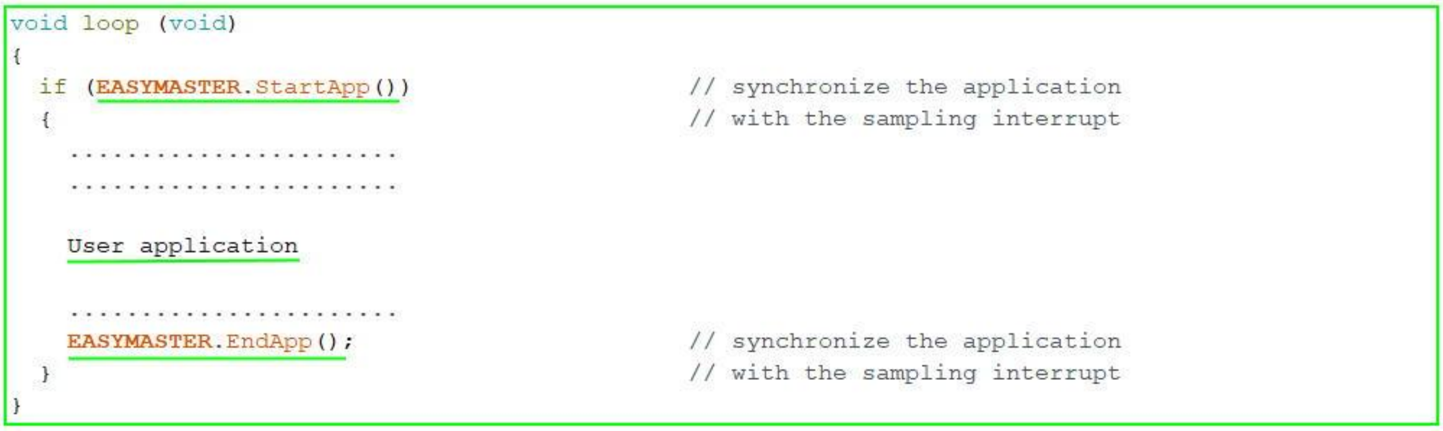


```
void setup (void)
{
  Serial.begin(115200);                          // enable debug messages on the serial line

                                                 // print the banner
  Serial.println(F("\nEasy Master V 0.0"));
  Serial.println(F("\nAB&T Tecnologie Informatiche"));
  Serial.println(F("Ivrea Italy - www.bausano.net"));

  EASYMASTER.Init(5);                            // init the Easy Master library
                                                 // with a sampling time of 5mS
}
```

Fig. 2 – Application setup

In the Loop() function.

- Here is where the actual application resides.
  It must be inside an if statement of the *EASYMASTER.StartApp()* function and must be terminated by the *EASYMASTER.EndApp()* function.
  This allows the application to be synchronized with the network sampling interrupt.

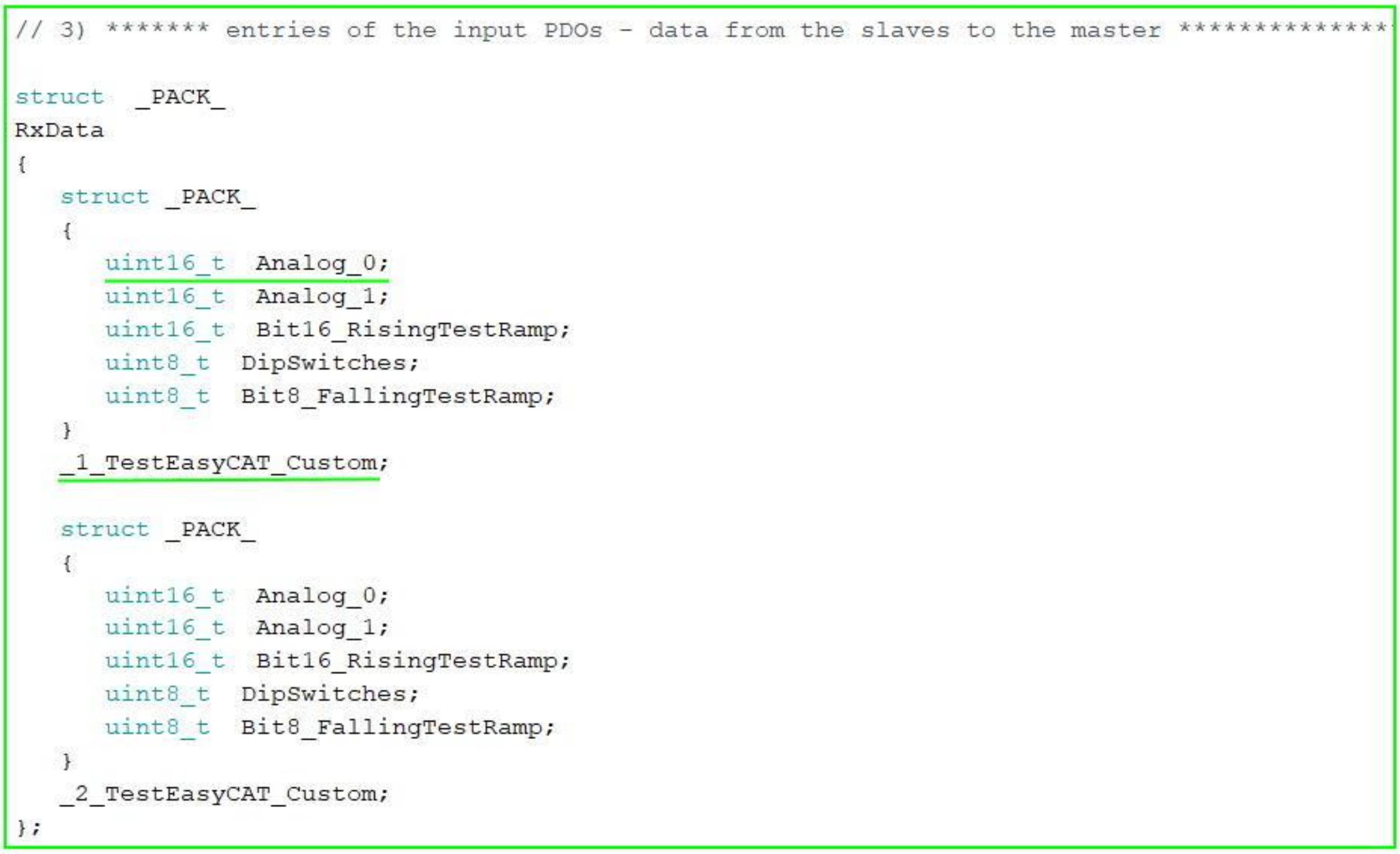


```
void loop (void)
{
  if (EASYMASTER.StartApp())                    // synchronize the application
  {                                             // with the sampling interrupt
    ......................
    ......................

    User application

    ......................
    EASYMASTER.EndApp();                        // synchronize the application
  }                                             // with the sampling interrupt
}
```

Fig. 3 – Application Loop

3.4 – Accessing the network data

To read or write data from or to the slaves we have to look for their names inside the NetworkInfo.h file.
For instance if we want to read the first analog input of the first slave we have to look for its name in section 3.
t



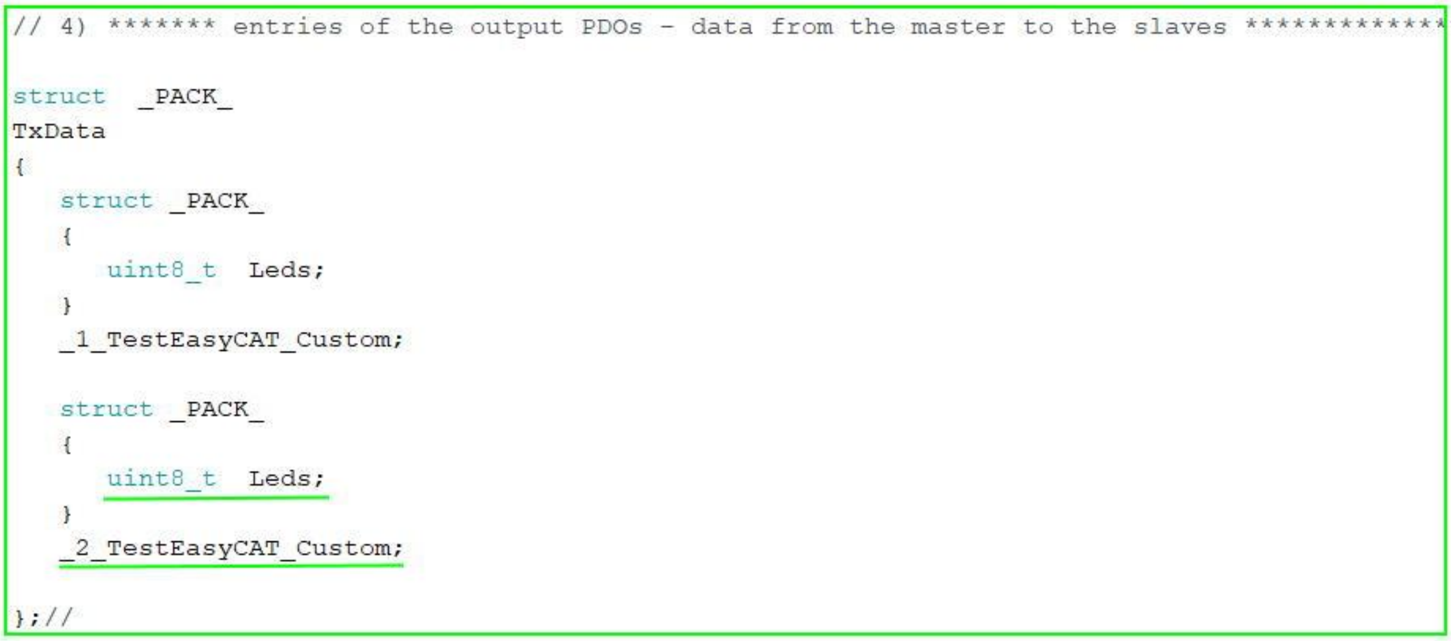
```
// 3) ******* entries of the input PDOs - data from the slaves to the master **************

struct  _PACK_
RxData
{
   struct _PACK_
   {
      uint16_t  Analog_0;
      uint16_t  Analog_1;
      uint16_t  Bit16_RisingTestRamp;
      uint8_t  DipSwitches;
      uint8_t  Bit8_FallingTestRamp;
   }
   _1_TestEasyCAT_Custom;

   struct _PACK_
   {
      uint16_t  Analog_0;
      uint16_t  Analog_1;
      uint16_t  Bit16_RisingTestRamp;
      uint8_t  DipSwitches;
      uint8_t  Bit8_FallingTestRamp;
   }
   _2_TestEasyCAT_Custom;
};
```

Fig. 4 – Input entries

In section 3 we find that we have to write:

```
uint16_t MyInputVar;

MyInputVar = _1_TestEasyCAT_Custom.Analog_0;
```

Similarly if we want to write data to the leds of the second slave we have to find the variable name in section 4.



```
// 4) ******* entries of the output PDOs - data from the master to the slaves *************

struct  _PACK_
TxData
{
   struct _PACK_
   {
      uint8_t  Leds;
   }
   _1_TestEasyCAT_Custom;

   struct _PACK_
   {
      uint8_t  Leds;
   }
   _2_TestEasyCAT_Custom;

};//
```

Fig. 5 – Output entries

In section 4 we find that we have to write:

```
uint8_t MyOutputVar = SOME_VALUE;

_2_TestEasyCAT_Custom.Leds = MyOutputVar;
```

4 – Error checking

The master checks for several error conditions, both during the initialization and during runtime. If one error is detected the network sampling is stopped, info about the error are printed out on the serial line and then the firmware remains in loop forever blinking the Arduino led.
The following error conditions are checked:

During initialization

- Number of the slaves.
- Name of the slaves.
- Operational state reached by all the slaves.

During runtime

- All slaves are in operational state.
- The working counter value is as expected.

------ **Appendix** ---

Useful links

AB&T website
https://www.bausano.net/en/

EasyCAT shield page on the AB&T website
https://www.bausano.net/en/hardware/ethercat-e-arduino/easycat.html

EasyCAT review on You Tube
https://www.youtube.com/watch?v=i_3PEEJ6QY8&t=7s

Easy Master library download
https://www.bausano.net/images/arduino-easycat/EasyMaster.zip

Easy Configurator download
https://www.bausano.net/images/arduino-easycat/EasyConfigurator.zip

Easy Configurator user manual
https://www.bausano.net/images/arduino-easycat/EasyConfigurator_UserManual.pdf

Easy Navigator download
https://www.bausano.net/images/arduino-easycat/EasyNavigator.zip

Easy Navigator user manual
https://www.bausano.net/images/arduino-easycat/Easy_Navigator_UserManual.pdf

EtherCAT Technology Group ETG
https://www.ethercat.org/default.htm

EasyCAT shield page on the ETG website
https://www.ethercat.org/en/products/791FFAA126AD43859920EA64384AD4FD.htm

SOEM EtherCAT master library by RT-Labs
https://github.com/OpenEtherCATsociety/SOEM

IgH EtherCAT master for Linux by EtherLab
https://etherlab.org/en/ethercat/

SOEM EtherCAT master library for Arduino by lipoyang
https://github.com/lipoyang/SOEM4Arduino

Arduino DUE EtherCAT master video by 西村備山
https://www.youtube.com/watch?v=ZB0Fy-w4dPM

------ Document history --

Rev 1.0 First release   27/04/2022