CanFestival2. Version 2.03
The CANOpen stack manual

Francis Dupin

November 25, 2005

LIVIC, Unité mixte de recherche INRETS-LCPC sur les Interractions
Véhicule-Infrastructure-Conducteur.
14, route de la minière. 78000 Versailles. Tel : 01 40 43 29 01.
http://www.inrets.fr/ur/livic

Contents

LIVIC, Unité mixte de recherche INRETS-LCPC sur les Interractions
Véhicule-Infrastructure-Conducteur.
14, route de la minière. 78000 Versailles. Tel : 01 40 43 29 01.
http://www.inrets.fr/ur/livic

Introduction

This document describe the CANOpen layer.

1 What is CANopen

CANopen is a CAN based highter layer protocol. It defines some protocols to

- Configure a CAN network.
- Transmit data to a specific node or in broadcast.
- Administrate the network. For example detecting a not responding node.

The documentation can be found in the *Can in automation* website : http://www.can-cia.de/canopen. The most important document about CANopen is the normative *CiA Draft Standard 301, version 4.02*. You can now download with no cost the specification in *Can in automation* website.

To continue reading this document, let us assume that you have read some papers introducing CANopen.

2 Our CANopen layer specification

- Should be conform to the latest DS301. V.4.02 13 february 2002.
- Source code in C-ANSI.
- 主从功能实现。
- Sending SYNC implemented.
- 每个节点1个SDO服务器。
- 无限的 SDO 客户机
- SDO传输模式：正常、快速下载上传。
- 无限的 PDO receive.
- 无限的 PDO transmit.
- 对象数据类型实现： unsigned 8, unsigned 16, unsigned 32 bits.

- Slave state 全部实现.
- NMT 来改变从机的状态 已实现。
- PDO传输方式 : on request, every reception of 0 to n SYNC, on event.
- NMT Heartbeat implemented : 一个节点可以监控从其他节点接收到的心跳。
- NMT NodeGuard 节点保护已实现.
- TIME (时间戳) : 没有实现.
- EMCY (emergency objects) : Not implemented.
- The CANopen layer can manage only one CAN port.

3 How to start

3.1 Tools

The CANopen library is comming with some tools :

- Arbracan : The Linux Driver for the Adlink 7841 CAN-PCI board. By Edouard Tisserand. It is working only on Linux kernel 2.4.
- the code to use a PCI or USB to CAN Peak system board. But we do not provide the driver (free download). It is working on Linux kernel 2.4 and 2.6. It should work also on Windows while the peaksystem driver has the same api for both systems.
- The code to make a master or slave node running on PC-Linux. compiled with the C GNU compiler.
- The code to make a master or slave node on a microcontroller Motorola MC9S12DP256 (HC12 family), compiled with the C GNU compiler.
- XML tools and php script to help the development. See below.

3.2 To do a Linux node

To do a CANopen node running on PC-Linux, you need :

- At least one board Adlink 7841 CAN-PCI or a Peak system board.
- A linux distribution working on a Personal Computer, with a kernel 2.4 if you wants to use the Adlink board, or 2.4 to 2.6 for a Peak system board.
- A Java runtime (optional but highly recommended).
- The PHP scripting language (optional but highly recommended).

3.3 To do a Motorola HC12 node

To do a CANopen node running on a microncontroller Motorola MC9S12DP256 you need :

- A board with this chip. We are using the T-board from Electronikladden. At least about 40 kBytes of program memory.
- The compiler GNU gcc.
- A Java runtime (optional but highly recommended).
- The PHP scripting language (optional but highly recommended).
- A tool to flash the memory. (We are using the hight cost Lauterbach debugger).

3.4 Running the examples

In *CanOpen/* are 4 examples : *AppliMaster_Linux, AppliSlave_Linux, AppliMaster_HC12, AppliSlave_HC12.*
Usage : Connect only one master to one slave. Always start the slave first.

What do the examples The slave node is waiting a nodeguard from the master. The master is sending the nodeguard, and waiting for the response. Then, it sends some SDO to write on the slave dictionary to configure it :

- Whose mapped data to put in each PDO receive and transmit.

- Defines the transmission mode for the PDO transmit.
- Gives a COBID to each PDO. It is this operation which defines which PDO a node is sending or receiving from/to others nodes. It is a logical linking.
- Defines what are the heartbeats to receive from others nodes and the timeout values.
- Defines at what frequency the node must send its heartbeat.

After that, the master puts the slave in operational mode. The node is then sending every second the time in a PDO sent when receiving a SYNC, and a PDO with the minute value when it changes, by a PDO ent *on event*. Every minutes the slave simulates an error, and sends to the master the errorCode and value in a PDO. Then, the master reset the error by a PDO. If a node is disconnected, by a reset for example, the other node complains because it does not receive any heartbeat.

3.4.1 Running a Linux node

By default the project is compiling for a board PeakSystem. To change this, edit CanOpen/includeMakefileLinux and put INTERFACE=ADLINK_7841_ARBRACAN

Installing the PCI-CAN or USB-CAN PeakSystem driver :

- Download the driver at http://www.peak-system.com/linux
- install it :

```
# uncompress
tar xvzf peak-linux-driver.3.21.tar.gz
cd  peak-linux-driver-3.21
#Read the Peak intallation document.
su root
make clean
make
make install

modprobe pcan
```

```
# Test :
cd test
./receivetest -f=/dev/pcan0
# connect the board to generator of can message.
# If it does not receive the message, put a 120 ohm resistor in the
network.
```

Installing the PCI-CAN driver for Adlink 7841. (Linux 2.4 only) :

- Edit CanOpen/MakefileLinux and uncomment the lines concerning Arbracan.
- Edit CanOpen/includeMakefileLinux and put INTERFACE=ADLINK_7841_ARBRACAN
- Compile CanFestival2. (See below)
- load Arbracan module :

  ```
  cd CanOpen/ArbraCan/loader
  su root
  ./arbracan_250kbps start
  ```

compiling CanFestival2 :

```
cd CanOpen
make -f MakefileLinux depend clean all
```

Connecting the nodes : Connect the two ports of the board. One Can port is for the master node, the other is for a slave node.

Starting the slave node : Open a new console and type :

```
cd CanOpen/AppliSlave_Linux
appliSlave
```

Starting the master node : Open a new console and type :

```
cd CanOpen/AppliMaster_Linux
appliMaster
```

3.4.2 Running a HC12 node

compiling :

```
cd CanOpen
make -f MakefileHC12 depend clean all
```

It will compile all the CANopen code for HC12.

flashing the memory : Use your prefered loader ! If you are using a debugger Lauterbach, you can load the bash file : *trace32_flash_programmer.cmm*. It loads directly the elf file.

Connecting to a serial RS232 console : Connect the portS(TxD0) of the HC12 to a console configured at 19200 bauds 8N1, via a Max232 chip to adapt the electricals levels. On Linux, you can use *minicom*. Connecting to a console is usefull to read the messages, but not required.

Connecting to the CAN network : Connect the port CAN0 (pin PM0, PM1) to the network via a CAN controller. On our board, the CAN controler is a PCA82C250 chip.

starting the node : Press the reset of your HC12 board.

3.4.3 Examples configuration tested

We have tested the library with a 3 nodes network :

- A master example.
- A slave example.
- A receive only node : CanAnalizer for IXXAT.

The configurations tested :

- AppliMaster_Linux connected to AppliSlave_Linux
- AppliMaster_HC12 connected to AppliSlave_Linux
- AppliMaster_Linux connected to AppliSlave_HC12

AppliMaster_HC12 connected to AppliSlave_HC12 have not been tested, but should work.

4 Developping a new node

最简单的方法是复制并重命名示例Applixxx之一，然后修改它。
所有应用程序都必须在这个目录中完成。
如果希望将目录移到CanOpen/之外，则必须修改 makefile。

4.1 How is organised the CANopen library

CanOpen/CanOpenMain Contains the .c files. They are target processor independant files. The only changes to do on theses files is commenting or uncommenting the #define at the top of the files to add or remove the debugging code.

CanOpen/include Contains the .h files.

- *CanOpen/include/*.h* : target processor independant.
- *CanOpen/include/linux* : Important definitions to fit the library to Linux. Edit the files *applicfg.h* and *timerhw.h*
- *CanOpen/include/hc12* : Important definitions to fit the library to HCS12. Edit the files *applicfg.h* and *timerhw.h*

CanOpen/CanOpenDriverLinux Contains the Linux CANopen driver for Linux. Need a PCI-CAN Adlink board. Should not be modified.

CanOpen/CanOpenDriverLinuxPeak Contains the Linux CANopen driver for Linux. Need a PCI-CAN or USB-CAN PeakSystem board. Should not be modified.

CanOpen/CanOpenDriverLinux Contains the Linux CANopen driver for Linux. Should not be modified.

CanOpen/CanOpenDriverHC12 Contains the CANopen driver part for HC12. Should not be modified. It uses the port CAN0 (pin PM0 and PM1).

CanOpen/Your_application The directory where you put your files.
objdict.c : It is a very important file, which contains the descritpion of the CANopen object dictionary of your node. It is difficult to adapt it by hand to your application, so we have developped some tools to generate it.

Computing the object Dictionary : objdict.c

1. Go to *CanOpen/XML_TOOLS/jaxe* and launch *run_objdict.sh*
   It loads Jaxe, with a XML-schema grammar for the description of an object dictionary.
   Jaxe is a XML editor in GPL licence that we are re-distributing. It needs a Java virtual machine. Edit with Jaxe your description save your file as *objdict.xml*. Jaxe computes also the file *objdict.html*, which is a human readable version of *objdict.xml*.

2. Loads then in your web browser the file *makeobjetdict.php*. Of course, you have installed before a webserver with PHP or you have a command line PHP version with Saxe support. *makeobjetdict.php* reads the file *objdict.xml* and compute *objdict.c*

3. Move *objdict.xml*, *objdict.html*, *objdict.c* in
   CanOpen/Your_application Returns *CanOpen/Your_application* and compile your code by :
   make -f Makefilexxx

Important notes

- 验证 objdict.xml 语法的文件是
  CanOpen/XML-TOOLS/jaxe/objdict.xsd

- The file (xslt) to compute *objdict.html* is
  CanOpen/XML-TOOLS/jaxe/objdict.xsl

- 在编辑完objdict.xml 之后，必须验证它，因为条目的大多数验证都是在这里完成的。
  要进行验证，您不必使用Jaxe。
  您还可以下载 Apache for Java XML工具。

  Then, the command line is something like :

```
java dom.ASBuilder  -f -a ./objdict.xsd  -i ./objdict.xml
```

LIVIC, Unité mixte de recherche INRETS-LCPC sur les Interractions Véhicule-Infrastructure-Conducteur.
14, route de la minière. 78000 Versailles. Tel : 01 40 43 29 01.
http://www.inrets.fr/ur/livic

5 FAQ

5.0.1 Does the code compiles on Windows ?

Yes, In fact, we use the CANopen layer in a huge project on Windows XP. For the Adlink 7841, we are using the driver provided by Adlink. We have had some problems with it, especially while waiting the messages received on event. Well, we have got round (more or less..) the bugs, and we have done a master node, compiled with Visual Studio C++. We plan to replace the Adlink 7841, while Adlink is not able to distibute a good driver, by a Peaksystem board.

Because some files in the project are in C++, and the CANopen layer in C, I think we have put a compilation option /TC or /TP. See the MSDN documentation about that.

You have to interface the library to the CAN board you are using. To do that, add a new directory *CanOpenDriverxxx* in *CanOpen/* and put your code inside, as we have done for HC12 and Linux.

In *CanOpen/include*, create also a new directory and put the two files *applicfg.h, timerhw.h*.

5.0.2 How to fit the library to an other microcontrôler ?

First, be sure that you have at least 40K bytes of program memory, and about 2k of RAM. It is not a such hard work. Be kind not to put specific code for your microcontroler in the code which is target independant. Put a prefix for the object files you create. Example on HC12 :
nmtMaster.c $\Longrightarrow$ hc12_nmtMaster.o

5.0.3 [LINUX] : How to use an other board than Adlink 7841 or Peaksystem board ?

If you have a driver, it should be easy. copy and rename the directory *CanOpen/CanOpenDriverLinux* and change the content of *receiveMsgHandler(), f_can_receive(), f_can_send()*. Add others specific functions, for example to open and close your board. If you have no driver, you have to develop it before.

5.0.4 [HC12] : What board are you using ?

A T-board from elektronikladen with a MC9S12DP256 or MC9S12DG256.

5.0.5 [HC12] : Does the code compile with an other compiler than GNU gcc ?

It is known to work with Metrowerks CodeWarrior. Here are some tips from Philippe Foureys. :

Interrupt functions

Code for GCC

```
// prototype

void __attribute__((interrupt))timer3Hdl(void):

// function
void __attribute__((interrupt))timer3Hdl(void)
{...}
```

Code for CodeWarrior

```
// protoype
void interrupt timer3Hdl(void);

// function
pragma CODE_SEG__NEAR_SEG_NON_BANKED
void interrupt timer3Hdl(void)
{...}
pragma CODE_SEG_DEFAULT
```

Interrupt lock, unlock

Code for GCC

```
void unlock (void)
{
  __asm__ __volatile__("cli");
}

void lock (void)
```

```
{
  unsigned short mask;
  __asm__ __volatile__("tpa\n\tsei":"=d"(mask));
}
```

Code for CodeWarrior

```
void unlock (void)
{
  __asm("cli");
}

void lock (void)
{
  unsigned short mask;
  __asm
 {
  tpa:tsei:"=d"(mask);
 }
}
```

Initialize function

Code for GCC

```
void initCanHCS12 (void)
{
  //Init the HCS12 microcontroler for CanOpen
  initHCS12();
   // Init the HCS12  CAN driver
  const canBusInit bi0 = {
    0,    /* no low power                 */
    0,    /* no time stamp                */
    1,    /* enable MSCAN                 */
    0,    /* clock source : oscillator (In fact, it is not used)   */
    0,    /* no loop back                 */
    0,    /* no listen only               */
    0,    /* no low pass filter for wk up */
CAN_Baudrates[CAN_BAUDRATE_250K],
    {
      0x00,    /* Filter on 16 bits. See Motorola Block Guide V02.14 fig 4-3 */
      0x00, 0xFF, /* filter 0 hight accept all msg      */
      0x00, 0xFF, /* filter 0 low accept all msg        */
      0x00, 0xFF, /* filter 1 hight filter all of  msg  */
      0x00, 0xFF, /* filter 1 low filter all of  msg    */
      0x00, 0xFF, /* filter 2 hight filter most of  msg */
      0x00, 0xFF, /* filter 2 low filter most of  msg   */
      0x00, 0xFF, /* filter 3 hight filter most of  msg */
      0x00, 0xFF, /* filter 3 low filter most of  msg   */
    }
  };
```

Code for CodeWarrior

```
void initCanHCS12 (void)
{
  //Init the HCS12 microcontroler for CanOpen
  initHCS12();
   // Init the HCS12  CAN driver
  const canBusInit bi0 = {
    0,    /* no low power                 */
    0,    /* no time stamp                */
    1,    /* enable MSCAN                 */
    0,    /* clock source : oscillator (In fact, it is not used)   */
    0,    /* no loop back                 */
    0,    /* no listen only               */
    0,    /* no low pass filter for wk up */
    {
     1, /* clksrc */
     3, /* brp    */
     0, /* sjw    */
     0, /* samp   */
     1, /* tseg2  */
     12,/* tseg1  */
    },

    {
      0x00,    /* Filter on 16 bits. See Motorola Block Guide V02.14 fig 4-3 */
      0x00, 0xFF, /* filter 0 hight accept all msg      */
      0x00, 0xFF, /* filter 0 low accept all msg        */
      0x00, 0xFF, /* filter 1 hight filter all of  msg  */
      0x00, 0xFF, /* filter 1 low filter all of  msg    */
      0x00, 0xFF, /* filter 2 hight filter most of  msg */
      0x00, 0xFF, /* filter 2 low filter most of  msg   */
      0x00, 0xFF, /* filter 3 hight filter most of  msg */
      0x00, 0xFF, /* filter 3 low filter most of  msg   */
    }
  };
```

5.0.6 [HC12] : Does the code works in banked memory ?

No. Today it seems that the port of gcc is bogged for using the banked memory. So, unfortunately, we are limited to 48 Kbytes of memory code.

5.0.7 [HC12] : What GCC version are you using ?

We are using the stable RPM release 2.2 :

- GNU Gcc 3.0.4. Build 20030501
- Newlib 1.10.0 Build 20030421
- GNU Binutils 2.12.1 Build 20030427

5.0.8 Is canfestival2 conform to DS301 v.4.02 ?

Thanks to Philippe Foureys (IUT of Valence), a slave node have been tested with the *National Instrument CanOpen Conformance Test*. It passed the test with success.
Some very small unconformity have been found in very unusual situations, for example in the SDO code response to wrong messages.

A Warnings and errors messages management

A.1 Warnings messages

DEBUG _WAR _CONSOLE _ON	**DEBUG _CAN**	**print-Msg-WarTo-Console**	**Printing long message on console**	**Printing short message on console.** (number and value only)	**Sending number and value in a PDO.**, only if the node is a slave, in operational state.
DEF	DEF	1		yes	
DEF	DEF	0			
DEF	UNDEF	1	yes		
DEF	UNDEF	0			
UNDEF	X	X			

A.2 Errors messages

DEBUG _ERR _CONSOLE _ON	**DEBUG _CAN**	**PDO _ERROR**	**printMsg-ErrTo-Console**	**Printing long message on console**	**Printing short message on console.** (number and value only)	**Sending number and value in a PDO.**, only if the node is a slave, in operational state.
DEF	DEF	X	1		yes	yes
DEF	DEF	X	0	yes		yes
DEF	UNDEF	X	1			yes
DEF	UNDEF	X	0			yes
UNDEF	X	DEF	X			yes
UNDEF	X	UNDEF	X			

B Web resources

CIA : Can in Automation

Many documentation on CANopen.
http://www.can-cia.de

Resources and training in CANopen

http://www.esacademy.com

Elektronikladen HCS12 T-board

http://www.elektronikladen.de/en_hcs12tb.html

Gnu gcc compiler for HC12

http://m68hc11.serveftp.org/m68hc11_port.php

Motorola documentation on HC12

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC9S12DP256

Lauterbach debugger for HC12

http://www.lauterbach.com

PHP. Script language

http://www.php.net

Jaxe, XML editor written in Java

http://jaxe.sourceforge.net