

CanFestival3. Version 3.0

The CANOpen stack manual

Table of contents

1 - Introduction.....	2
2 - What is CANopen.....	2
3 - CanFestival Features.....	2
3.1) Tools	2
3.2) Standard conformance.....	2
4 - How to start.....	3
4.1) Host requirements.....	3
4.2) Linux Targets.....	3
4.2.1)Normal Linux node.....	3
4.2.2)Real-Time Linux node.....	4
4.2.3)CAN devices.....	4
4.2.4)Linux Example:.....	4
4.3) Motorola HC12.....	5
4.3.1)Running a HC12 node.....	5
4.3.2)HCS12 Examples.....	5
4.4) Compilation.....	5
5 - Developing with CanFestival.....	6
5.1) Understanding Canfestival.....	6
5.1.1)CanFestival Project tree layout.....	6
5.1.2)Implement CanFestival in your application.....	7
5.1.3)CanFestival CAN interfaces.....	7
5.1.4)CanFestival events scheduling.....	8
5.2) Developing a new node.....	9
5.2.1)Using Dictionary Editor GUI.....	9
5.2.2)Generating the object Dictionary.....	15
6 - FAQ.....	16
6.1) General.....	16
6.1.1)Does the code compiles on Windows ?.....	16
6.1.2)How to fit the library to an other microcontrôler ?.....	16
6.1.3)Is CanFestival3 conform to DS301 v.4.02 ?.....	16
6.2) LINUX.....	16
6.2.1)How to use a Peaksystem CAN board ?.....	16
6.2.2)How to use an unsupported CAN board ?.....	16
6.3) HCS12.....	16
6.3.1)Which board are you using ?.....	16
6.3.2)Does the code compile with an other compiler than GNU gcc ?.....	17
6.3.3)Who to use warnings and errors messages ?.....	19
6.3.4)Does the code works in banked memory ?.....	20
6.3.5)What GCC version are you using ?.....	20
7 - Documentation resources.....	20
8 - About the project.....	22
8.1) Contributors	22
8.2) Getting support.....	22
8.3) Contributing.....	23
8.4) License.....	23

1 - Introduction

This document describe the CANOpen layer.CanFestival is an OpenSource (LGPL) CANOpen framework.

This project, initiated by Edouard TISSERANT in 2001, as grown thanks to Francis DUPIN and other contributors.

Today, CanFestival focuses on providing an ANSI-C platform independent CANOpen stack that can be implemented as master or slave nodes on PCs, Real-time IPCs, and Microcontrollers.

CanFestival is a project supported by Lolitech.

2 - What is CANopen

CANopen is a CAN based high level 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.

3 - CanFestival Features

3.1) Tools

The CANopen library is coming with some tools :

- Object Dictionary editor GUI. WxPython Model-View-Controller based GUI, that help a lot in generating object dictionary source code for each node.
- A home made configure script, that let you chose compile time options such as target CPU/HOST and/or CAN,TIMER, LED or NVRAM drivers.

3.2) Standard conformance

a)Multi-Platform

- Library source code is C-ANSI.
- Driver and examples coding conventions merely depend on target specific contributor/compiler.
- Unix compatible interfaces and examples should compile and run on any Unix system (tested on GNU/Linux and GNU/FreeBSD).

b)CanOpen conformance

- Should be conform to the latest DS301. V.4.02 13 february 2002.

- Master and Slave functionality implemented.
- Sending SYNC implemented.
- 1 SDO server per node.
- Unlimited SDO client.
- SDO transmission mode : normal, expedited download and upload.
- Unlimited PDO receive.
- Unlimited PDO transmit.
- Object Data type implemented : 8, 16, 32 bits values, an fixed length strings.
- Slave state full implemented.
- NMT to change slave's state implemented.
- PDO transmission mode : on request, every reception of 0 to n SYNC, on event.
- NMT Heartbeat implemented : A node can be either heartbeat producer or receiver.
- NMT NodeGuard implemented : Not fully implemented.
- TIME (time Stamp) : Not implemented.
- EMCY (emergency objects) : Not implemented.

4 - How to start

4.1) Host requirements

What you need on your development workstation.

- Linux, FreeBSD, or any Unix with GNU toolchain.
- The GNU C compiler (gcc) or any other ANSI-C compiler for your target platform.
- GNU Make
- Bash
- Python, with wxPython modules installed (at least version 2.6.3). You will also need Xpdf, and the official 301_v04000201.pdf file in order to get GUI context sensitive help.

4.2) Linux Targets

Linux target is default configure target. Corresponding configure switch is:

```
--target=generic
```

4.2.1) Normal Linux node

Configure switch:

```
--timers=unix
```

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

- A working linux distribution
- One or more Peak system PC CAN interface and the last Peak Linux driver installed.

4.2.2)Real-Time Linux node

Configure switch:

```
--timers=xeno
```

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

- A working linux distribution patched with XENOMAI 2.1 or greater.
- One or more Peak system PC CAN interface and the last Peak Real Time Linux driver installed.

4.2.3)CAN devices

Curently supported CAN devices and corresponding configure switch:

a)Peak systems

Configure switch:

```
--can=peak
```

PeakSystems CAN interface is automatically chosen as default CAN interface if libpcan is present in the system.

Please download driver at <http://www.peak-system.com/linux> and follow instructions in order to install driver on your system.

b)Virtual CAN interfaces (for test/debug)

Configure switch:

```
--can=virtual
```

Virtual CAN interface use Unix pipes to emulate a virtual CAN network. Each message issued from a node is repeat to all other nodes. Currently only works with nodes running in the same process, and does not support work with Xenomai.

4.2.4)Linux Example:

TestMasterSlave :

```
*****
*   TestMasterSlave                                     *
*                                                                 *
*   A simple example for PC. It does implement 2 CanOpen      *
*   nodes in the same process. A master and a slave. Both    *
*   communicate together, exchanging periodically NMT, SYNC, *
*   SDO and PDO.                                             *
*                                                                 *
*   If you have chosen virtual CAN driver, just type         *
*   ./TestMasterSlave                                       *
*                                                                 *
*   Else you need to sepcify bus:                            *
*                                                                 *
*       -s : slave CAN bus [default 0, peak first PCI]      *
*       -m : master CAN bus [default 1, peak second PCI]    *
*                                                                 *
*****
```

4.3) Motorola HC12

Configure switch:

```
--target=hcs12
```

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

- The compiler GNU gcc for HC12 : m6811-elf.
- A board with this chip. We are using the T-board from Elektronkladden. At least about 40 kBytes of program memory.
- A tool to flash the memory. (We are using the high cost Lauterbach debugger).

4.3.1)Running a HC12 node

a)Compiling :

```
./configure --target=hc12  
make all
```

b)Flashing the memory :

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

c)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.

d)Connecting to the CAN network :

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

e)starting the node :

Press the reset of your HC12 board.

4.3.2)HCS12 Examples

GeneSync_HCS12 :

An example of HCS12 based CanOpen node sending SYNC messages periodically.

4.4) Compilation

Call ./configure -help to see all available compile time options.

After invoking ./configure with your platform specific switches, just type make.

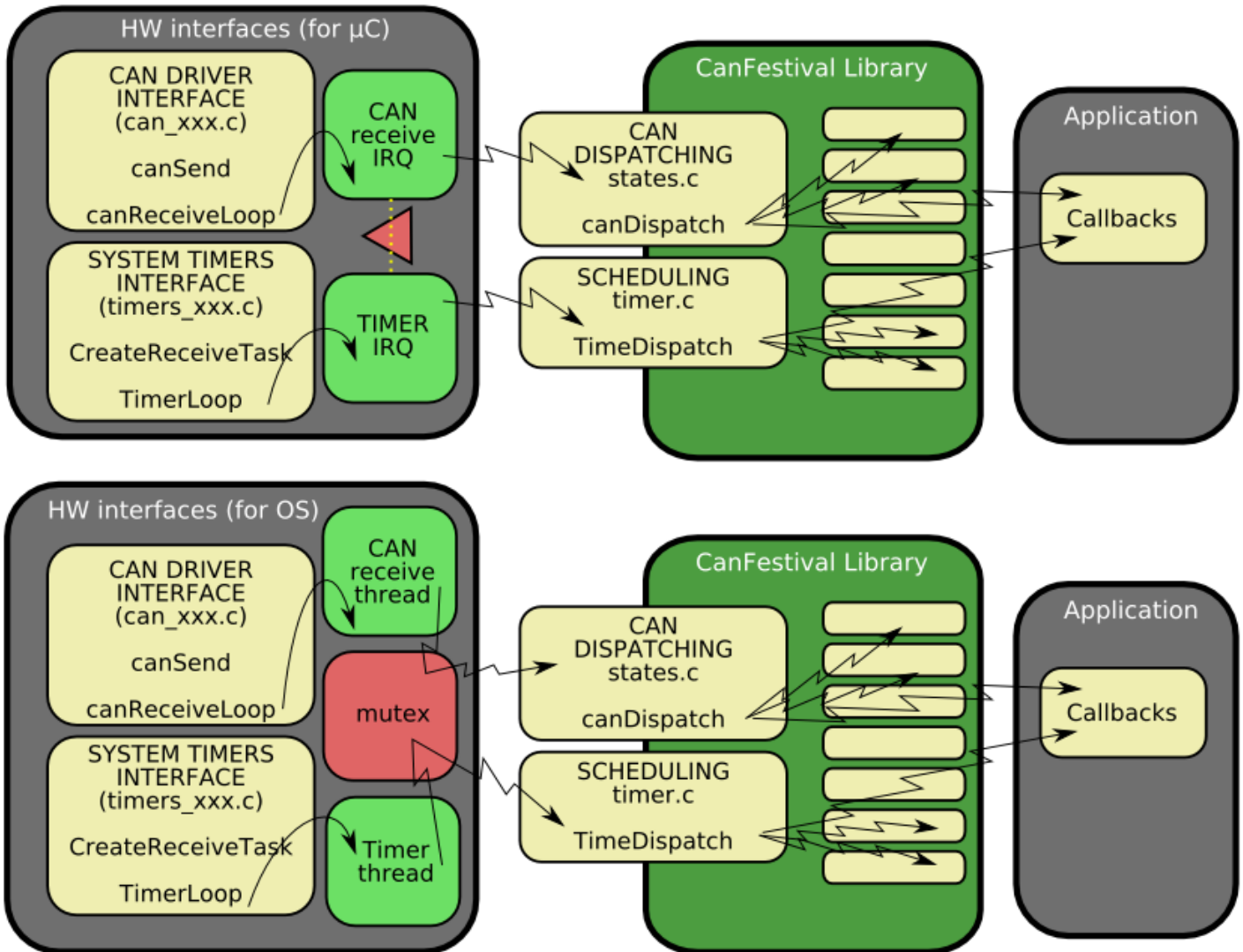
```
./configure [options]  
make  
make install
```


5.1.3) CanFestival CAN interfaces

Because most CAN controllers and drivers implement FIFOs, CanFestival consider sending message as a non blocking operation.

In order to prevent reentrant calls to the stack, messages reception is implemented differently on μ C and OS.:

- μ C must provide interruption masking for timer and can receive IT
- OS must provide a receive thread, a timer thread and a mutex. CAN reception is a blocking operation.



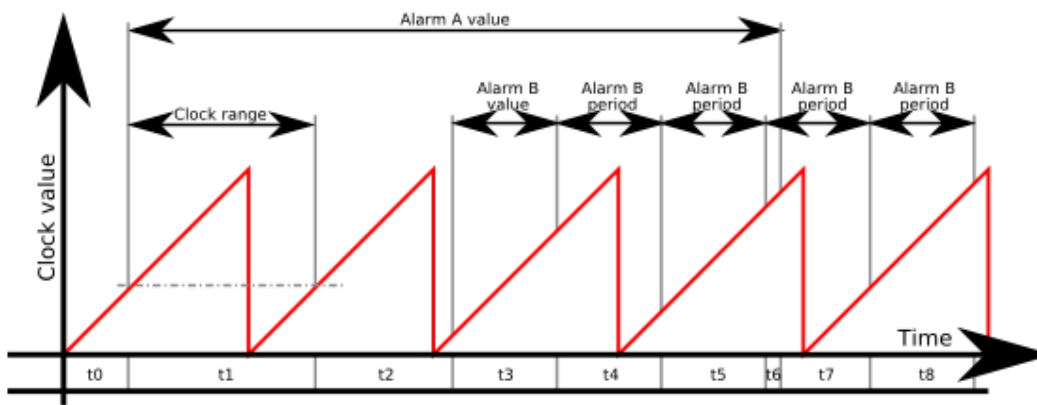
5.1.4) CanFestival events scheduling

A CanOpen node must be able to take delayed actions.

As examples, periodic sync emission, heartbeat production or SDO timeout need to set some alarms that will be called later and do the job.

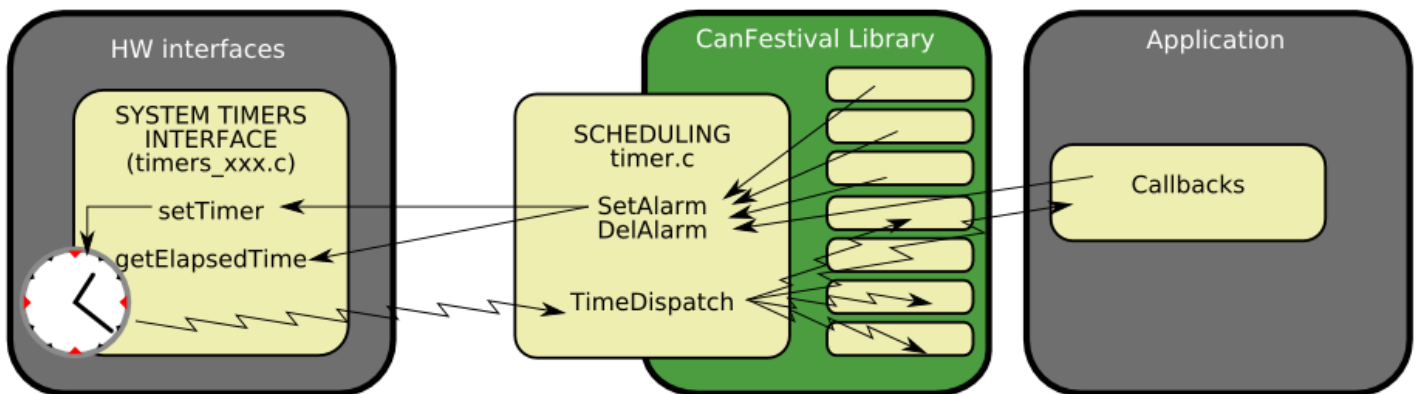
μ C generally do not have enough free timers to handle all the CanOpen needs directly. Moreover, CanFestival internal data may be corrupt by reentrant calls.

CanFestival implement a micro-scheduler (timer.c). It uses only one timer to mimic many timers. It manage an alarm table, and call alarms at desired time.



Scheduler can handle short clock value ranges limitation found on some μC . As an example, value range for a 16bit clock counter with $4\mu\text{s}$ tick is crossed within 0.26 seconds... Long alarms must be segmented.

Chronogram illustrate a long alarm (A) and a short periodic alarm (B), with a $A \text{ value} > \text{clock range} > B \text{ value}$. Values $t_0 \dots t_8$ are successive `setTimer` call parameter values. t_1 illustrates an intermediate call to `TimeDispatch`, caused by a delay longer than clock range. Because of long alarm segmentation, at the end of t_1 , `TimeDispatch` call will not trig any alarm callback.



5.2) Developing a new node

Using provided examples as a base for your new node is generally a good idea. You can also use the provided *.od files as a base for your node object dictionary.

5.2.1) Using Dictionary Editor GUI

a) Usage

```
cd objdictgen
python objdictedit.py [od files...]
```

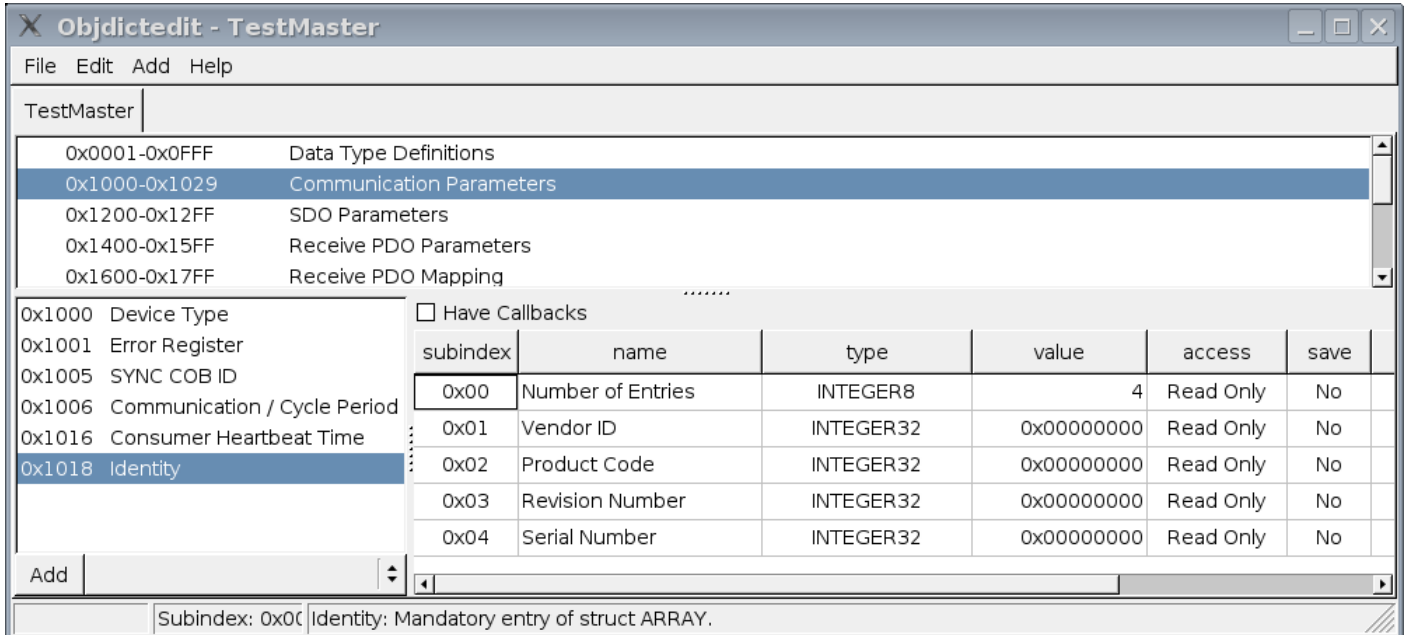
b) About

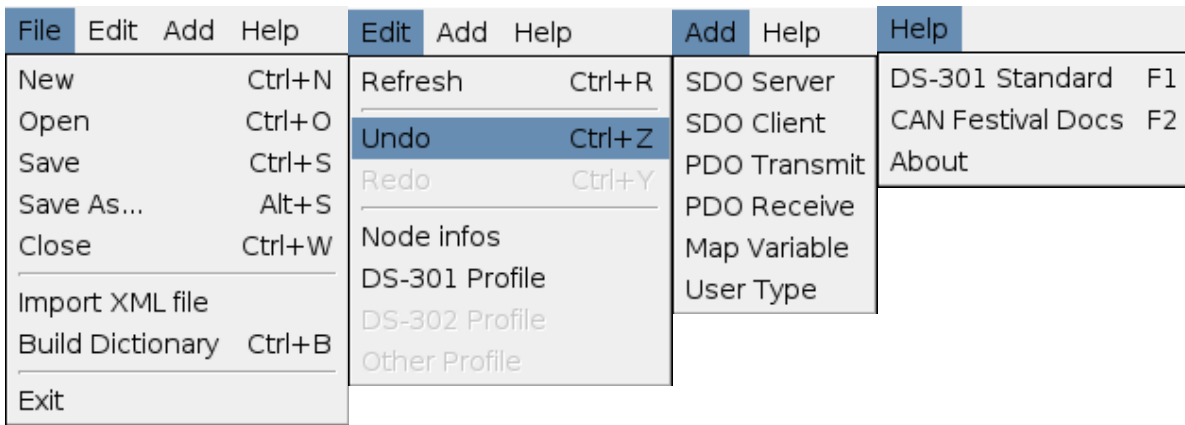
The Object Dictionary editor GUI is a python application that use the Model-View-Controller design pattern. It depends on WxPython to display view on any supported platform.



c)Main view

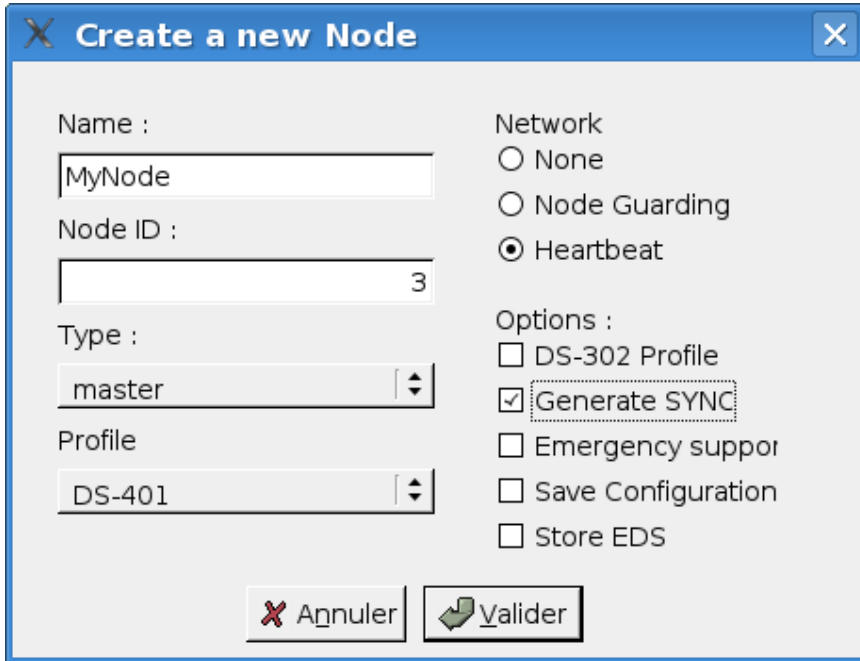
Top list let you choose dictionary section, bottom left list is the selected index in that dictionary, and bottom right list are edited sub-indexes.





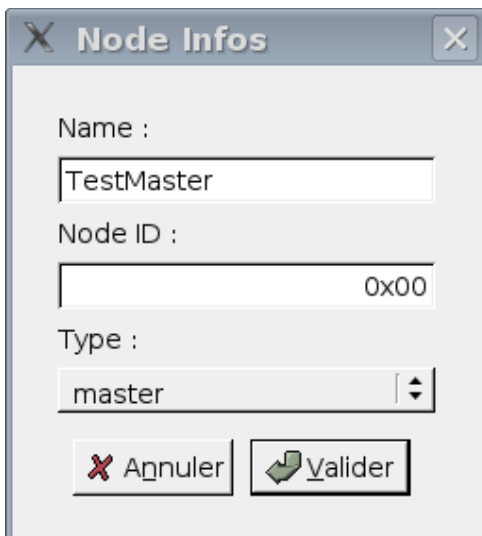
d)New node

Edit your node name, ID and type. Choose your inherited specific profile.



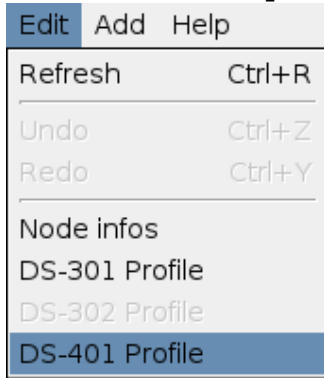
e)Node info

Edit your node name, ID and type.

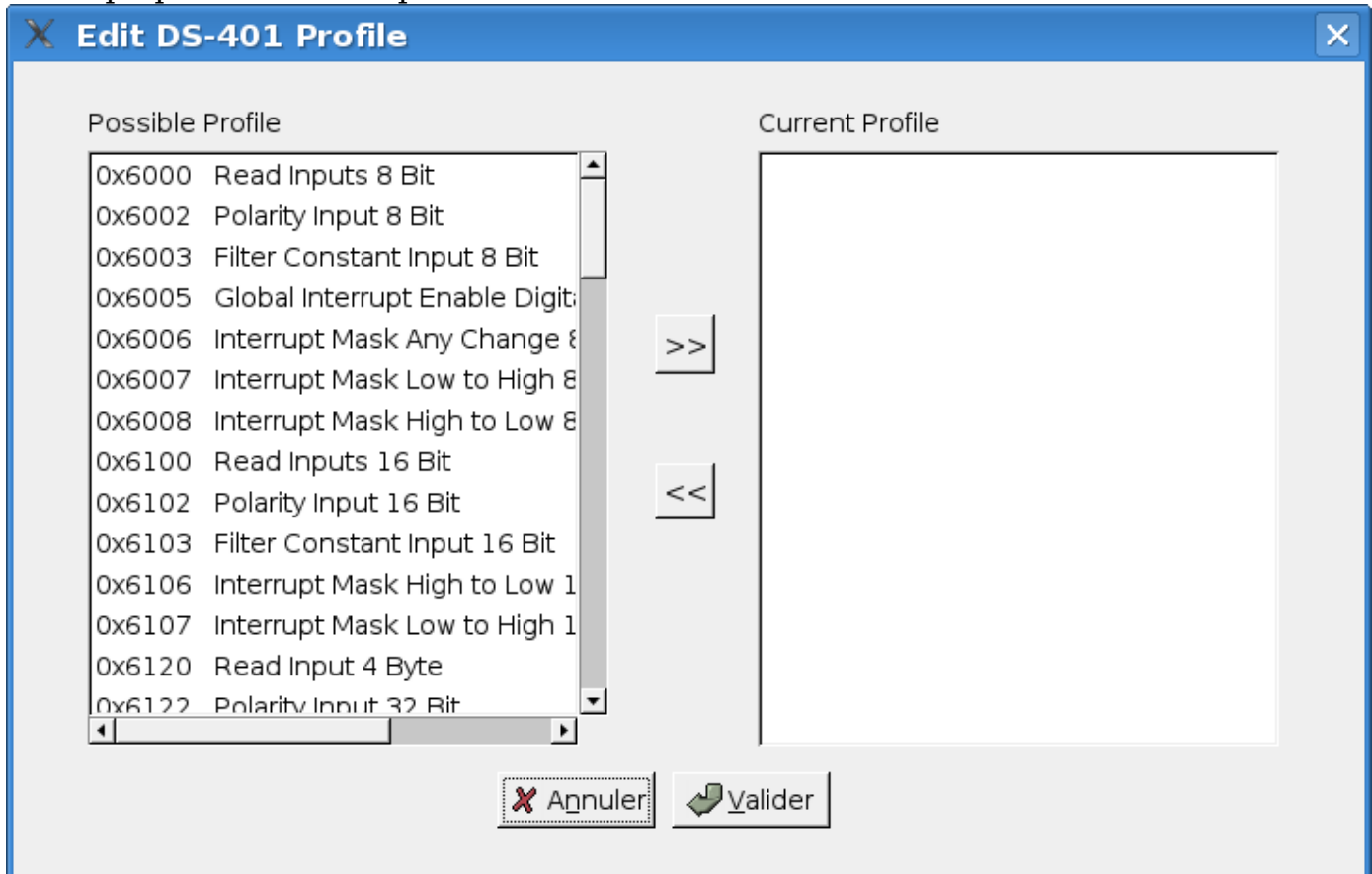


f) Profile editor

Chose the used profile to edit.

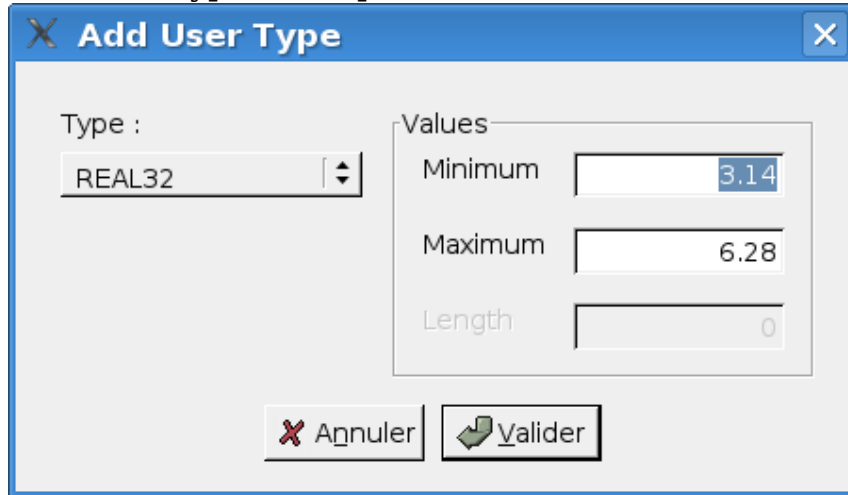


Pick up optional chosen profile entries.



g)User types

Use User Types to implement value boundaries, and string length



Add User Type

Type : REAL32

Values

Minimum 3.14

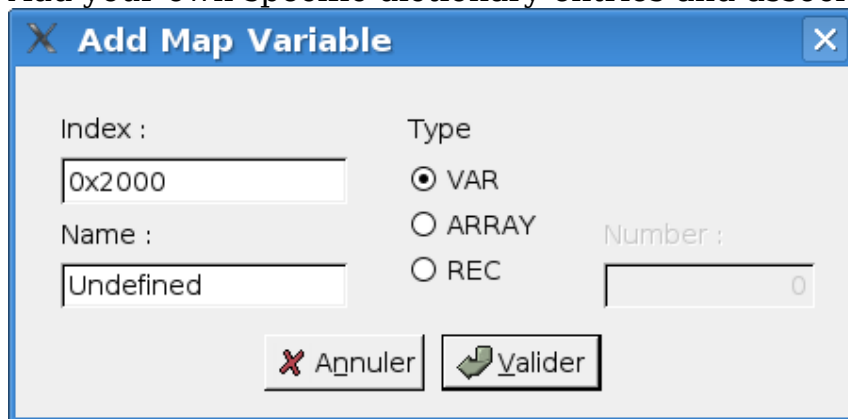
Maximum 6.28

Length 0

Annuler Valider

h)Mapped variable

Add your own specific dictionary entries and associated mapped variables.



Add Map Variable

Index : 0x2000

Name : Undefined

Type

VAR

ARRAY

REC

Number : 0

Annuler Valider

i) Integrated help

Using F1 key, you can get context sensitive help.

Object 1018h: Identity Object

The object at index 1018h contains general information about the device.
The Vendor ID (sub-index 1h) contains a unique value allocated to each manufacturer.
The manufacturer-specific Product code (sub-index 2h) identifies a specific device version.
The manufacturer-specific Revision number (sub-index 3h) consists of a major revision number and a minor revision number. The major revision number identifies a specific CANopen behaviour. If the CANopen functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions with the same CANopen behaviour.

31 16 15 0
MSB major revision number minor revision number LSB

Figure 63: Structure of Revision number

The manufacturer-specific Serial number (sub-index 4h) identifies a specific device.

OBJECT DESCRIPTION

INDEX	1018h
Name	Identity Object
Object Code	RECORD
Data Type	Identity
Category	Mandatory

Page 101 of 135 125% Quit

In order to do that, official 301_v04000201.pdf file must be placed into doc/ directory, and xpdf must be present on your system.

CAN Festival Reference

CanFestival Scheduling

A CanOpen must be able to take delayed actions.

As examples, periodic sync emission, heartbeat production or SDO timeout need to set some alarms that will be called later and do the job.

μ C generally do not have many enough free timers to handle all the CanOpen needs directly. Moreover, CanFestival internal data may be corrupt by reentrant calls.

CanFestival implement a mini-scheduler (timer.c). It uses only one timer to mimic many timers. It manage an alarm table, and call alarms at time.

Scheduler can handle short clock value ranges limitation found on some μ C. As an example, value range counter with 4μ s tick is crossed within 0.26 seconds... Long alarms must be segmented.

Chronogram illustrate a long alarm (A) and a short periodic alarm (B), with a A value > clock range > B

5.2.2) Generating the object Dictionary

Once object dictionary has been edited and saved, you have to generate object dictionary C code for your CanFestival node.

a) With GUI

Menu entry "File/Build Dictionary".

New	Ctrl+N
Open	Ctrl+O
Save	Ctrl+S
Save As...	Alt+S
Close	Ctrl+W

Import XML file	
Build Dictionary	Ctrl+B

Exit	

Choose C file to create or overwrite. Header file will be also created with the same prefix as C file.

b)With command line

Usage of objdictgen.py :
python objdictgen.py XMLFilePath CfilePath

6 - FAQ

6.1) General

6.1.1)Does the code compiles on Windows ?

Yes, some user got success wompiling with Visual Studio C++.

Because CANopen layer is coded with C, put a compilation option /TC or /TP if you plant to mix C++ files. See the MSDN documentation about that.

6.1.2)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.

You have to create target specific interface to HW resources. Take model on bundled interfaces provided in drivers/ and create your own interface. You also have to update Makefile.in files for target specific cflags and options. Chose `-targer=` configure switch to compile your specific interface.

You are welcome to contribute-back your own interfaces ! Other Canfestival users will use it and provide feedback, tests and enhancements.

6.1.3)Is CanFestival3 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.

6.2) LINUX

6.2.1)How to use a Peakssystem CAN board ?

Just install peak driver and then compile and install Canfestival. Peak driver is detected at compile time.

6.2.2)How to use an unsupported CAN board ?

You have to install the specific driver on your system, with necessary libs and headers. Use `can_peak.c/h` or `can_virtual.c/h` as an example, and adapt it tou your driver API. Execute configure script and choose `--can=mydriver`

6.3) HCS12

6.3.1)Which board are you using ?

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

6.3.2) 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. :

a) Interrupt functions

i) Code for GCC:

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

ii) Code for CodeWarrior

```
// protoype
void interrupt timer3Hdl(void);
// function
#pragma CODE_SEG_NEAR_SEG_NON_BANKED
void interrupt timer3Hdl(void)
{...}
#pragma CODE_SEG_DEFAULT
```

b) Interrupt lock, unlock

i) Code for GCC

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

ii) Code for CodeWarrior

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


c)Initialize function

i)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 */
        }
    };
};
```

ii)Code for CodeWarrior

```

void initCanHCS12 (void)
{
    //Init the HCS12 microcontroller 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 */
        }
    };
};

```

6.3.3)Who to use warnings and errors messages ?

a)Warnings messages

DEBUG_W AR_CONS OLE_ON	DEBUG _CAN	PrintMsg- 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	yes		
DEF	UNDEF	1			
DEF	UNDEF	0			
UNDEF	X	X			

b)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			

6.3.4)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.

6.3.5)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

7 - Documentation resources

a)CIA : Can in Automation

Many documentation on CANopen.

<http://www.can-cia.de>

b)Resources and training in CANopen

<http://www.esacademy.com>

c)Elektronikladen HCS12 T-board

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

d)Gnu gcc compiler for HC12

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

e)Motorola documentation on HC12

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

f)Lauterbach debugger for HC12

<http://www.lauterbach.com>

g)Python language

<http://www.python.org>

8 - About the project

8.1) Contributors

The logo for LIVIC, featuring the word "LIVIC" in a bold, white, sans-serif font, centered within a grey rectangular background.

Unité mixte de recherche INRETS-LCPC
sur les Interactions Véhicule-Infrastructure-Conducteur
14, route de la minière
78000 Versailles
FRANCE
Tel : +33 1 40 43 29 01

<http://www.inrets.fr/ur/livic>

Contributors : Francis DUPIN
Camille BOSSARD
Laurent ROMIEUX

The logo for LOLI Tech, with "LOLI" in a stylized, multi-colored font and "Tech" in a grey, outlined font. Below it, the text "Logiciel Libre et Technologie" is written in a smaller, grey font.

LOLITECH
204, rue du Haut du Pin
88470 Saint-Michel sur Meurthe
FRANCE
Tel : +33 3 29 52 95 67

<http://www.lolitech.fr>

Contributors : Edouard TISSERANT (Original author)
Laurent BESSARD

Many thanks to the other contributors for their great work:

Raphael ZULLIGER
David DUMINY (sté A6R)
Zakaria BELAMRI

8.2) Getting support

Send your feedback and bug reports to canfestival-devel@lists.sourceforge.net.

For commercial support, training and specific integration and developments, please ask LOLITECH (see contributors).

8.3) Contributing

You are free to contribute your specific interfaces back to the project. This way, you can hope to get support from CanFestival users community.

Please send your patch to canfestival-devel@lists.sourceforge.net.

Feel free to create some new predefined DS-4xx profiles (*.prf) in objdictgen/config, as much as possible respectful to the official specifications.

8.4) License

All the project is licensed with LGPL. This mean you can link CanFestival with any code without being obliged to publish it.

```
#This file is part of CanFestival, a library implementing CanOpen Stack.
#
#Copyright (C): Edouard TISSERANT, Francis DUPIN and Laurent BESSARD
#
#See COPYING file for copyrights details.
#
#This library is free software; you can redistribute it and/or
#modify it under the terms of the GNU Lesser General Public
#License as published by the Free Software Foundation; either
#version 2.1 of the License, or (at your option) any later version.
#
#This library is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
#Lesser General Public License for more details.
#
#You should have received a copy of the GNU Lesser General Public
#License along with this library; if not, write to the Free Software
#Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```