NOVIA
UNIVERSITY OF APPLIED SCIENCES

Automatic Testing of a CANopen Node

Hui Liang

Bachelor's thesis

Electrical Engineering

Vaasa 2013

BACHELOR'S THESIS

Author:	Hui Liang
Degree programme:	Electrical Engineering
Specialization:	Automation Engineering
Supervisors:	Dag Björklund, Novia
	Simon Byholm, TK Engineering Oy

Title: *Automatic Testing of a CANopen Node*

---

10 May 2013 34 pages 1 appendix

Abstract

This Bachelor's thesis was commissioned by TK Engineering Oy in Vaasa. The goals of the thesis were to test a prototype CANopen node, called UWASA Node for conformance to the CiA 301 standard, and to develop the automatic performance test software and the automatic CiA 401 test software. A test report that describes to the designer what needs to be corrected and improved is made in this thesis. For the CiA 301 test there is a CANopen conformance test tool that can be used. The automatic performance test software and the automatic CiA 401 test software are developed in this thesis.

Language: English Key words: CANopen, CiA 301, testing, CiA 401

The thesis is available in the electronic library theseus.fi

EXAMENSARBETE

Författare:	Hui Liang
Utbildningsprogram:	Elektroteknik
Fördjupning:	Automationsteknik
Handledare:	Dag Björklund , Novia Simon Byholm, TK Engineering Oy

Titel: *Automatisk Testning av en CANopen Nod*

---

10 Maj 2013 34 sidor 1 bilaga

Abstrakt

Uppdragsgivare till detta examensarbete var TK Engineering Oy i Vasa. Målet med arbetet var att testa en prototyp CANopen nod, kallad UWASA, för överensstämmelse med CANopen CiA 301 standarden, samt att utveckla mjukvara för automatiserade prestandatest samt CiA 401 test. Vidare skulle testrapporter förberedas så att utvecklaren av UWASA mjukvaran vet vad som bör korrigeras och förbättras. För CiA 301 testet fans det ett CANopen testverktyg tillgängligt som kunde utnyttjas. Ett automatiskt testverktyg för prestandatest samt test av överensstämmelse med CiA DS401 utvecklades i detta arbete.

Språk: Engelska Nyckelord: CANopen, , CiA 301, testing, CiA 401

Förvaras på Theseus.fi

TABLE OF CONTENTS

ABBREVIATIONS

TKE	TK Engineering Oy
CAN	Control Area Network
CiA	CAN in Automation
OSI	Open System Interconnection
ISO	International Organization for Standardization
OD	Object Dictionary
NMT	Network Management
COB	Communication Object
EDS	Electronic Data Sheet
SDO	Service Data Object
PDO	Process Data Object
TPDO	Transmit PDO
RPDO	Receive PDO
EMCY	Emergency
SYNC	Synchronization
RTR	Remote Transmission Request
CCT	CANopen Conformance Test
ms	millisecond
LSS	Layer Setting Service
PC	Personal Computer

FOREWARD

I am heartily thankful to my supervisors, Mr. Dag Björklund and Mr. Simon Byholm, for their supervision and support during this project. A special thanks to Mr. Timo Kesti, the CEO of TK Engineering Oy, who gave me the chance to do the practical work. I am also thanking Mr. Bertil Bäck and other staff from TKE who offered help during the project.

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This chapter gives a brief introduction of the goal of the thesis, the background of the device as well as a brief introduction of the company at which the main work has been done.

1.1 Task

The goal of this thesis is to test the prototype UWASA node for conformance to the CANopen CiA DS301 (CAN in Automation Draft Specification 301), to develop the automatic test software for the performance test according to an internal CANopen Device Specification, and to develop the automatic test software for testing that against the CiA DS401 standard. A test report describing to the designers what needs to be corrected and improved and some other documents should be done.

1.2 Background

TK Engineering Oy

The thesis work is done at TK Engineering Oy in Vaasa, Finland. TK Engineering Oy is a high-tech company which focuses on education and training, engineering, electronics and software development, imports and sales.

In automation industries, TK Engineering Oy offers wide services including education, design and implementation to maintenance tasks, such as CAN traffic analyzing, CANopen device testing and debugging, protocol analyzing etc. TK Engineering Oy is also a member of CAN in Automation and has its own CANopen vendor-ID for their products. TKE has three main technologies, CAN, CANopen and PROFIBUS. TKE also supports other bus technologies such as J1939, ISOBUS, NMEA2000 and LIN via partner networks. /1/

UWASA Node

The UWASA Node (Figure 1) is a modular and stackable wireless sensor platform. It is a generic hardware platform which "*allows fast adaptation for development of different wireless automation applications by providing means of stacking relatively simple slave boards*". /2/

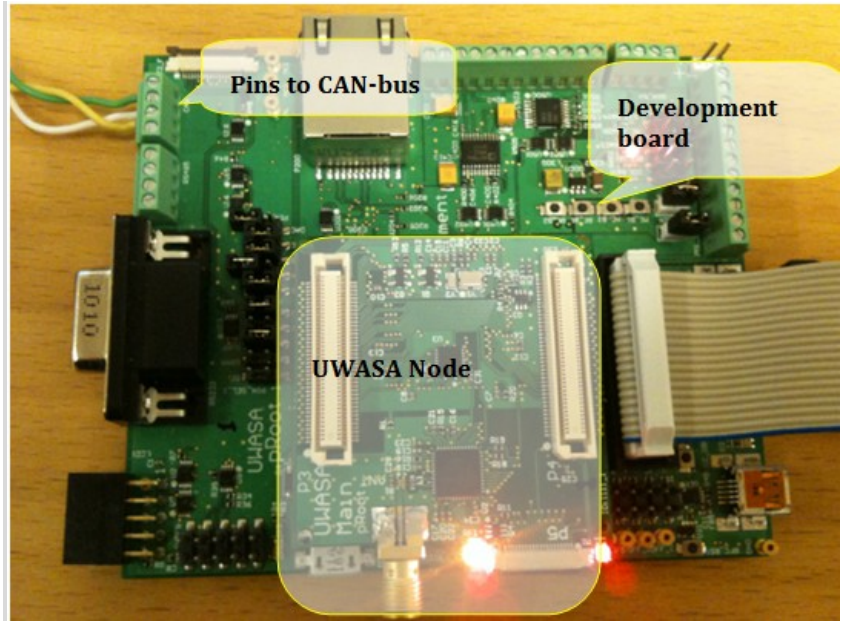

Figure 1 UWASA Node and the development board

The main controller of the UWASA Node is LPC2378 of NXP, which is an ARM7TDMI-S 32bit Pipelined processor. The LPC2378 supports a lot of serial interfaces, such as 2 CAN channels, USB 2.0, 4 UART and so on. The UWASA Node has CAN bus and it supports CANopen protocol which is implemented by using one open source CANopen stack called "CAN Festival" in UWASA Node. /3/

There is also a development board which provides essential development interfaces for the main controller, such as the debug emulator. The debug software used is "IAR Embedded Workbench for ARM 6.50" and the debug emulator is "Segger J-Link", which is a USB powered JTAG emulator. The source code of the UWASA Node is only for testing and it does not contain any other functionality of the UWASA Node.

As shown above in Figure 1, the pins to the CAN-bus are on the left upper corner of the development board. During the practical work the UWASA Node is connected to the CAN network via these pins. In this thesis the practical work is interactive, which means that after running the CAN conformance test, the source code will be updated every time by the designer, and then the updated source code is downloaded to the UWASA Node and sent for testing again.

2 CAN and CANopen

This chapter briefly introduces the history of CAN bus, compares the CAN bus and the Open System Interconnection (OSI) 7-layer reference model and describes the CANopen protocol structure.

2.1 CAN Protocol

The Controller Area Network, also known as CAN or CAN-bus, was developed by the company called Robert Bosch GmbH in the mid-1980s for the automobile industry. With the rapid development of the automobile industry, CAN-bus is widely used not only in automobiles, but also in many other applications, such as aerospace, industrial control and so on. The first CAN specification was published by Robert Bosch GmbH. It is called CAN 2.0 Specification- Part A. Later on, an upgraded version of the CAN specification, which is called CAN 2.0 Specification- Part B, was released and it is compatible with the Part A version. /4/

At the beginning of 1992, the CAN in Automation (CiA), a nonprofit international users and manufacturers association, was established in Germany. The aim of CiA was to promote the developments of the CAN protocol. It published CiA specifications which cover physical layer definitions, application layer and device profiles. Nowadays, about 560 companies all over the world are members of the CiA group. /5/

In 1993, CAN became an International Organization for Standardization (ISO) standard. The ISO-11898 standard, which is based on CAN 2.0 Specification, is the most widely used physical layer standard for CAN networks. Compared with OSI 7-layer reference model, the higher-layer CAN protocol only partly implemented the functionality from the higher layers in the OSI 7-layer reference model, as can be seen in Figure 2. /6/

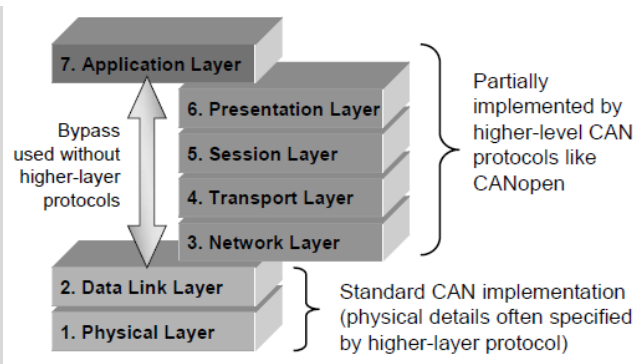

Figure 2 CAN Protocol and OSI 7-layer Reference Model /6/

The OSI 7-layer Reference Model is the standard reference communication model in the network across different equipment and applications. Nowadays, most network protocols are based on this reference model. Below is a brief description of the seven protocol layers in the OSI reference model, along with a description of the corresponding CAN protocol layers.

The physical layer

In the OSI 7-layer model, the physical layer is the first layer. It defines the physical and electric specifications for the devices, such as which kind of cable should be used, the layout of pins, voltages, signal timing and so on. It defines "bit-level" communication. /7/

The physical layer of the CAN protocol specifies the Bit-Encoding, Bit-Timing and synchronization and defines the physical media used in the CAN-bus. The physical layer is specified in detail in the ISO-11898 standard. Figure 3 shows the CAN bus topology structure. The CAN protocol recommended that each end of the network should be terminated by a 120 Ω resistor (Generally, when testing or analyzing the short CAN-bus at laboratory room, one 120 Ω resistor is enough. For testing the UWASA Node, one 120 Ω resistor is connected to the short CAN-bus.). The length of the CAN bus is determined by the baud rate. All the devices that are connected to the CAN-bus should have the same baud rate so that they can communicate with each other. /8/

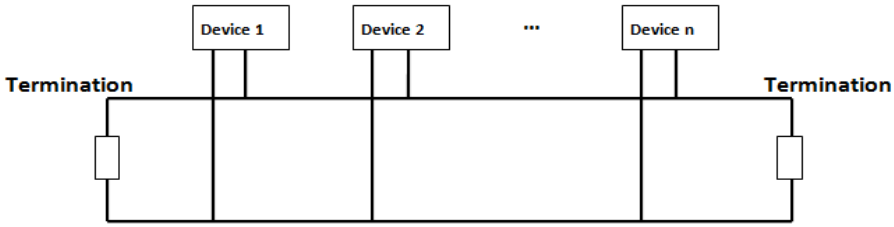

Figure 3 CAN Network Topology Structure/8/

The Data Link Layer

In the OSI 7-layer model, the data link layer is the link between the physical layer and the network layer. The data link layer is divided into two sublayers, the Logical Link Control (LLC) sublayer and the Media Access Control (MAC) sublayer. The Logical Link Control (LLC) is used to detect the errors that may occur in the physical layer and possibly it corrects the errors. The Multiple Access Control (MAC) technique handles the access to the bus, such as preventing or handling the collisions when many nodes are connected to the bus and want to write to the bus simultaneously. /7/

For the CAN protocol, the ISO 11898 standard specifies the LLC sublayer and the MAC sublayer of the CAN data link layer in detail. As a CAN bus is a multidrop bus topology, it uses the Carrier Sense Multiple Access (CSMA) as a Media Access Control (MAC) method to prevent or handle collisions. Only the CANopen node that sends the CAN message with highest priority can transmit the message preferentially, while the other nodes will detect the collision and terminate the transmission.

The CAN data link layer also formats the frame of the data. Take a data frame that contains node data for transmission with 11 identifier bits (specified in CAN 2.0 Specification- Part A) as an example to briefly introduce the standard data frame format.

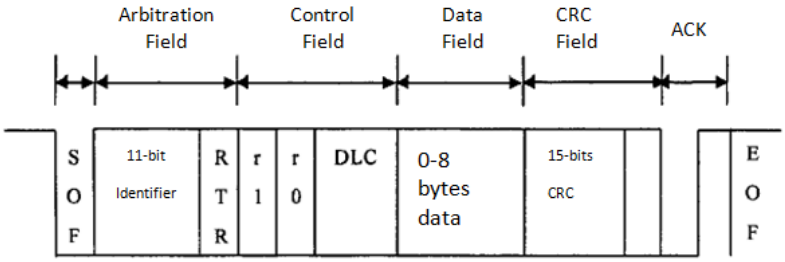

Figure 4 Standard Data Frame Format/9/(modified)

In Figure 4, the SOF and the EOF are short for start of frame and end of frame respectively. The 11 identifier is the communication object identifier (COB-ID). The Remote Transmission Request (RTR) bit has to be dominant and it has to be recessive within a Remote Frame. In the control field, the Data Length Code (DLC) specifies the number of bytes in the data field. There are 0-8 bytes data in the data field. The Cyclic Redundancy Check (CRC) filed is used for detecting errors. The Acknowledge (ACK) field will be filled by the node that receives this frame. /9/

2.2 CANopen Protocol

The most important part of the CANopen protocol is that it defines the protocol for the application layer. CANopen protocol also implements part of the functionality from the network layer of the OSI model to the presentation layer (Figure 2).

The Application Layer

In the OSI 7-layer model, the application layer is the highest layer that actually interacts with the operating system or the software applications. It provides direct process when the end user wants to transfer files, send emails or implement other network related activities. /10/

The CANopen protocol is mainly an application layer protocol. The following section 2.2.1 gives a description of the concept of the CANopen protocol.

2.2.1 CANopen Protocol Overview

The overall structure of the CAN and CANopen protocol is shown in Figure 5 (Corresponding to layer 1, layer 2 and layer 7 of the OSI model in Figure 2).

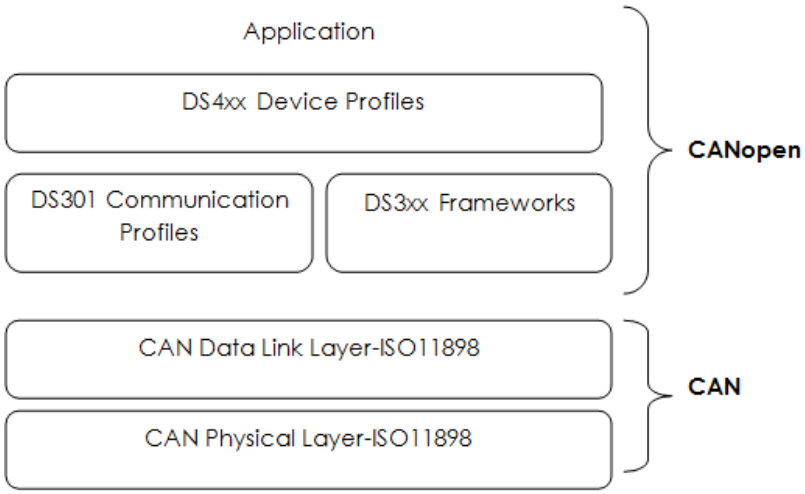

Figure 5 The structure of the CAN and CANopen protocol

It can be seen in Figure 5 that the DS301, DS3xx and DS4xx are all included in the CANopen protocol. They are built on top of the CAN standards. The CiA DS301, "CANopen Application Layer and Communication Profile" is the most important profile in the whole CANopen protocol. All the CANopen devices should implement CiA DS301. It defines the basic communication, the data structuring, and the network management methods used in the CAN network.

The CiA DS3xx series are extended communication profiles that are based on the DS301. They are called frameworks, which are used to meet the communication requirement in particular applications. Some examples are as follows:

- CANopen Framework for CANopen Mangers and Programmable CANopen Devices (CiA Draft Standard Proposal 302 (DSP302))
- CANopen Framework for Safety-relevant Communication (CiA DSP304)
- CANopen Layer Setting Services and Protocols (LSS, CiA DSP305)
- CANopen Framework for Maritime Electronics (CiA DSP307) /11/

The CANopen Device Profiles define the means of communication in a specific type of equipment that is connected to the CAN-bus. In an industrial environment, each type of device that is connected to the CAN-bus should have a device profile. However, so far only a few standard device profiles are defined, such as the device profile for generic I/O modules (CiA DS401), the device profile for drives and motion control (CiA DS402), the device profile for encoders (CiA DS406) and so on./11/

For testing the UWASA Node, the CiA DS301 and the CiA DS401 were studied and sections 2.2.2 and 2.2.3 below give a brief introduction of these two standards.

2.2.2 CiA DS301 Standard

As mentioned above, CiA DS301 is the most important CANopen protocol. This section will briefly introduce the most important part of the CiA DS301 standard.

In CiA DS301, several important concepts are defined, such as Object Dictionary, Service Data Object (SDO), Process Data Object (PDO), Network Management (NMT) and other features (e.g. SYNC, TIME and EMCY). The Object Dictionary is the interface between the communication interface and the application process as Figure 6 shows. The communication of the device's configuration parameters is implemented via SDO, while the communication of the process parameters is implemented via PDO. The Network Management (NMT) is used to control the communication state of network nodes and to monitor the nodes. SYNC (synchronization) is used to implement the synchronization of data transmission. TIME provides the generic reference time frame for the nodes. EMCY (emergency) is the communication object that is used in emergency management when an internal error of the device has occurred. /12/

1. Object Dictionary (OD)

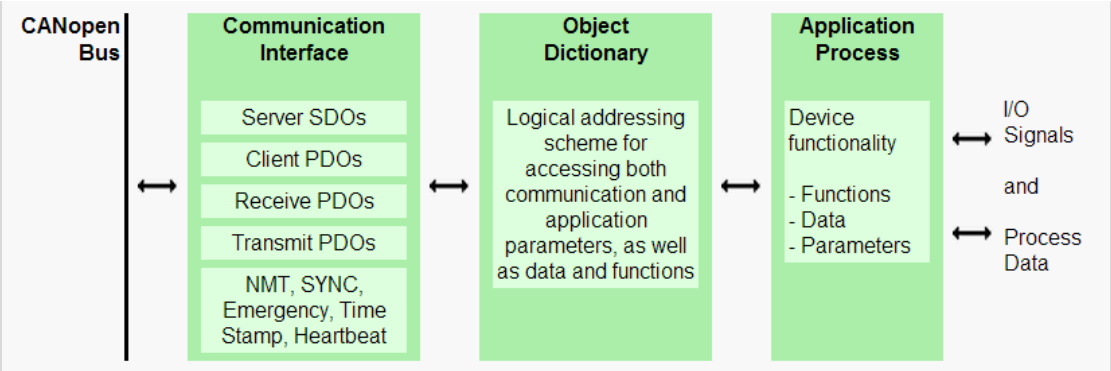

Figure 6 CANopen Communication Structure/12/

The Object Dictionary is equivalent to a list of parameters. It is a table that has the same structure for all types of devices. All the configuration parameters and process parameters of the device are stored in the Object Dictionary. Each CANopen device has only one Object Dictionary. It is possible to access the parameters in the Object Dictionary of the CANopen device via the CAN bus by using a logical addressing system (16-bit index and 8-bit sub index). The 65536 possible indexes are divided into sections for different types of data (Table 1). /12/

Table 1 Object Dictionary Organization /13/

Index Range	Description
0x0000	Reserved
0x0001 – 0x0FFF	Data Types
0x1000 – 0x1FFF	Communication Entries
0x2000 – 0x5FFF	Manufacturer Specific
0x6000 – 0x9FFF	Device Profile Parameters
0xA000 – 0xFFFF	Reserved

2. Electronic Data Sheet (EDS)

The Electronic Data Sheet (EDS) file is used in CANopen to specify the supported Object Dictionary entries of a CANopen node. Each CANopen product should have its own EDS file. The CANopen Conformance Test also tests if the device implements all the Object Dictionary entries specified in its EDS file (refer to section 3.2).

An example of an Object Dictionary entry in EDS file is:

```
[1018sub1]

ParameterName=Vender ID

ObjectType=0x7

DataType=0x0007

AccessType=ro

DefaultValue=0x0203004B

PDOMapping=0
```

[1018sub1] means that this entry is sub index 1 of the index 0x1018. ParameterName is the name of this Object Dictionary entry. ObjectType=Ox7 means that the type of the object is a variable. DataType=0x0007 means that the data type is an unsigned 32-bit integer defined in object 0x0007.AccessType=ro means that the access type is read only. DefaultValue=0x0203004B specifies the default value of this entry. PDOMapping=0 means that this object does not support PDO mapping.

3. Service Data Object (SDO)

The Service Data Object (SDO) is used to establish the Client/Server relationship between two CANopen devices. The device that supports Service Data Object is called SDO Server. The Client device can access the Object Dictionary of the Server device by reading or writing its index and sub index. Generally, in a CANopen network, the only one master node plays the role of the SDO Client for all the other nodes. The master node can configure parameters for other nodes via SDO. Two COB-IDs are needed in one SDO request/response round: one for the SDO Client sending requests to the SDO Server and one for the SDO Server sending responses back to the SDO Client. If the SDO request is not successful, the abort message containing the abort code defined in the CiA DS301 standard should be sent. /14/

Although the SDO can access all the parameters in the Object Dictionary, it is not an efficient way to access process parameters. Therefore, an efficient communication method is defined in CANopen protocol—the Process Data Objects (PDOs).

4. Process Data Object (PDO)

The Process Data Object (PDO) service is based on the producer/consumer relationship model of network communication. The PDO is used to transmit real-time data. The PDO that is received by the consumer device is called its Receive PDO (RPDO) while the PDO that is sent by the producer device is called its Transmit PDO (TPDO). Each PDO must be assigned a CAN message identifier (COB-ID). The CiA DS301 defines the default CAN message identifiers for 4 TPDOs and 4 RPDOs. /12//14/

PDO Transmission Mode

There are transmission types of these kinds:

1. Synchronous transmission

   The CANopen device will transmit the PDO or handle the PDO when it receives a synchronization object (SYNC) from the network management node.

2. Asynchronous transmission

   The PDO is event driven. When there is a change of the process data, the PDO will be transmitted according to the setting of the CANopen protocol implemented in the CANopen device. /15/

PDO Message-Triggering Mode

There are four PDO message-triggering modes.

1. Event driven

   The PDO may be transmitted at any time when an internal event of the CANopen device has occurred.

2. Timer driven

   The timer can trigger the PDO even if no event occurs.

3. Remote Request

   One device can ask another device to send the PDO by transmitting a remote transmission request (RTR).

4. Synchronized

   One CANopen device will transmit the PDO or handle the PDO when it receives a synchronization object (SYNC) from a synchronization application. /15/

PDO Mapping

The PDO Producer and the PDO Consumer both know the meaning of the data of the PDO message based on the PDO mapping mechanism. Each byte of the PDO message is mapping to the parameter of one specific entry in the Object Dictionary. The mapping relationship is stored in both the PDO Producer's and the PDO Consumer's Object Dictionaries. Not every parameter is mappable in the Object Dictionary. Generally, process data allows PDO mapping, other configuration parameters can be accessed via SDO reading or writing. /16/

Network Management (NMT)

The CANopen Network Management is based on the master/slave structure. There is only one NMT master node in one CANopen network. The other nodes are NMT slave nodes. As mentioned earlier, the NMT service has two tasks:

- Control of the communication state of network nodes
- Node monitoring by using node guarding and heartbeat /17/

The control of the communication state of network nodes means that the NMT master node in the CANopen network can change the state of the NMT slave node by sending a command. There are four NMT states in the communication process of the slave node: Initialization, Pre-operational, Operational and Stopped. Figure 7 illustrates the relationship of the different NMT states of the NMT slave node. After the slave node resets or powers on, the slave node will enter initialization state. After initialization, the slave node will send back the boot-up message to inform the NMT master node and enter NMT state pre-operational automatically. Under NMT state pre-operational,

the NMT master node can configure parameters for the NMT slave node. The NMT slave node will enter NMT state operational and start to communicate in the CANopen network after it has received the start remote node command sent by the NMT master node. If the NMT slave node has a severe communication failure, the NMT master will set the NMT slave node enter NMT state stopped. /17/

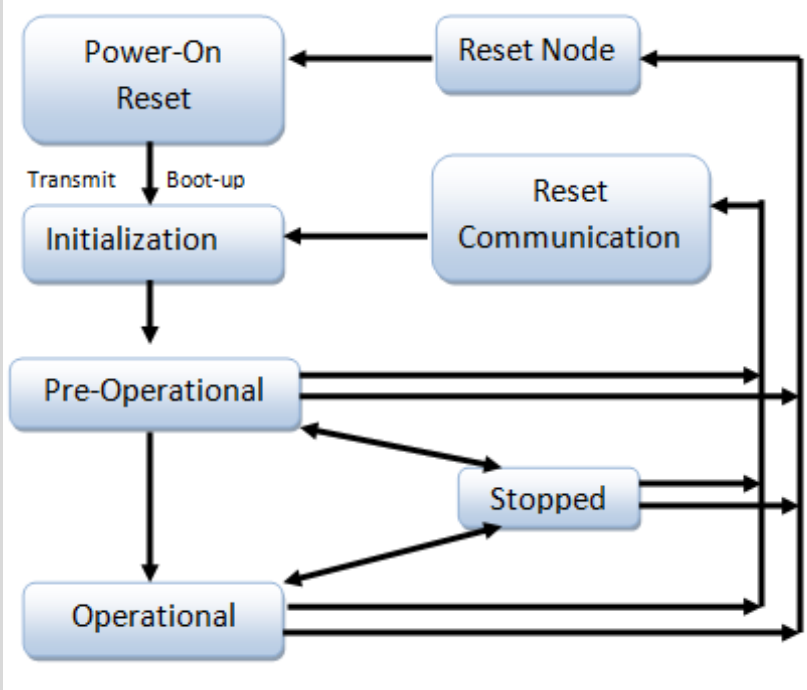

Figure 7 The Network Management States of a Slave/18/

Node guarding and heartbeat are two mechanisms of CANopen device monitoring. They are both used to check if the node can communicate normally.

1. Node guarding
   The NMT master node broadcasts the remote request message cyclically and the NMT slave node will response with a message that contains the information which could indicate the node identifier and the current NMT state. If the NMT master node cannot receive the response message within a certain time (node life time), it will be assessed as a communication failure of the node. /14/

2. Heartbeat
   The CANopen slave node will send the heartbeat message cyclically. For example, for a certain node A, all the nodes that have communication relationship with node A should check node A's heartbeat message. If those nodes do not receive the heartbeat of the node A, they will consider that there is a communication failure of the node A. In the CANopen protocol, the heartbeat mechanism is recommended because it does not need the involvement of the NMT master node. /14/

At different NMT states, the different communication objects are allowed if the CANopen device is involved in the communication. The relationship between NMT states and the communication objects are listed in Table 2.

Table 2 Relationship between NMT states and Communication Objects/18/

	Pre-operational	Operational	Stopped
PDO		X	
SDO	X	X	
TIME & SYNC	X	X	
EMCY	X	X	
Heartbeat/ Node guarding	X	X	X

2.2.3 CiA DS401 Standard

CiA DS401 is a CANopen Device Profile for Generic I/O Modules. Many CANopen devices implement this profile.

It supports 64 bits digital inputs, 64 bits digital outputs, 12 analogue inputs (16 bits) and 12 analogue outputs (16 bits) by default. CIA 401 also defines the default PDO mapping parameter. For each I/O module, it defines corresponding OD entries and relative feature parameters of digital inputs/outputs and analogue inputs/outputs. /19/

The UWASA Node supports digital inputs (8-bit access) and analogue inputs (16-bit). Thus the digital inputs and analogue inputs functionality should be tested in this thesis. The CiA DS401 test can be found in chapter 5. The digital inputs and analogue inputs functionality are briefly introduced below.

Default PDO Mapping

In CiA 401, four default TPDOs and four default RPDOs for digital/analogue inputs/outputs are defined. (Table 3)

Table 3 Default TPDO and RPDO mapping of CiA 401

TPDO1	Mapping to digital inputs
TPDO2	Mapping to maximum 4 analogue inputs
TPDO3	Mapping to maximum 4 analogue inputs
TPDO4	Mapping to maximum 4 analogue inputs
RPDO1	Mapping to digital outputs
RPDO2	Mapping to maximum 4 analogue outputs
RPDO3	Mapping to maximum 4 analogue outputs
RPDO4	Mapping to maximum 4 analogue outputs

Digital Inputs

The default maximum digital input is 64-bits. There are also many different access methods defined, such as 1-bit, 8-bit, 16-bit or 32-bit access to digital inputs or outputs. By default, 8-bit access shall be supported. Figure 8 shows the relationship between the digital input objects for an 8-bit access. /19/

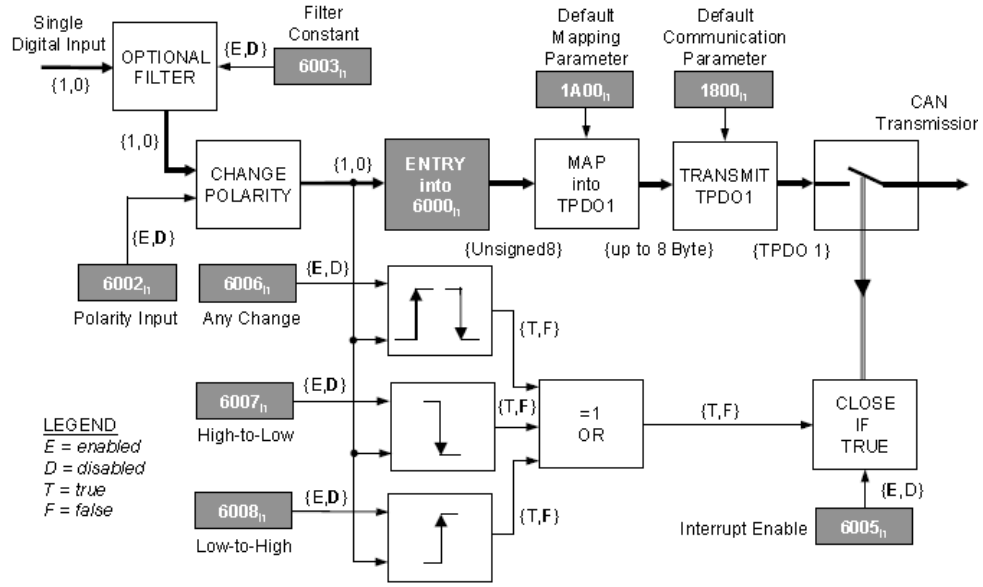

Figure 8 Block diagram for 8-bit access digital inputs /19/

For UWASA Node, the polarity function and the interrupts (corresponding objects 6005h, 6006h, 6007h and 6008h) are all supported and should be tested.

Analogue Inputs

There are many different access methods defined to access analogue inputs, such as 8-bit, 16-bit, 32-bit access or float access. By default, the CANopen node can have 16-bit access to maximum 12 analogue inputs, each TPDO can map to maximum 4 analogue inputs. Figure 9 shows the relationship between the analogue input objects for 16-bit access. /19/

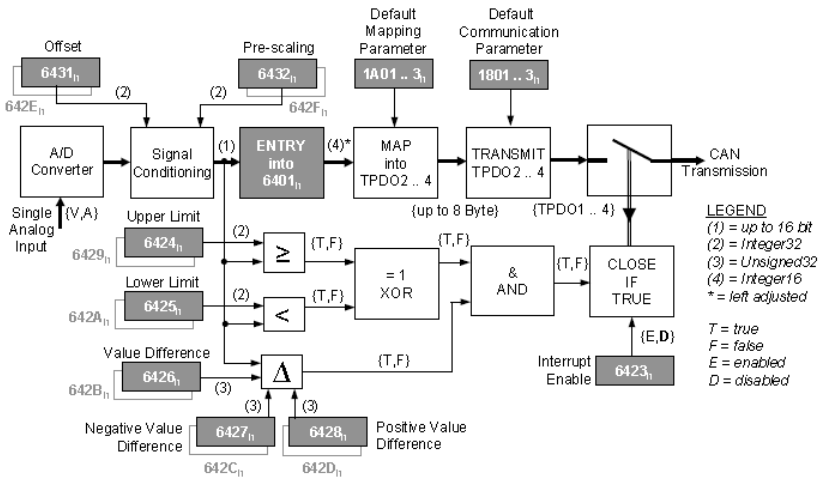

Figure 9 Block diagram for 16-bit access analogue inputs /19/

For UWASA Node, four 16-bit access analogue inputs are supported and mapped to TPDO2. All the analogue inputs only support the cyclic event-driven transmit mechanism which can be tested in the CiA401 test.

2.3 Summary of this chapter

This chapter has mainly introduced the structure of the CAN and CANopen protocol. The most important protocol CiA DS301 has been emphasized. The basic concepts of the CANopen in CiA DS301 have been introduced, such as Object Dictionary, SDO, PDO and NMT. Then the frequently-used CiA DS401 Device Profile for Generic I/O Modules is presented. These protocols play significant roles in the implementation of CANopen.

3 CANopen Conformance Test

The CANopen Conformance test (CCT) can help the manufacturers to check whether the CANopen implementation of their CANopen devices is compliant to the CiA DS301 CANopen specification. There is an official CANopen conformance test software that can be used to test the UWASA Node against the latest CiA DS301 standard. The test tool is made by CiA. Although the test tool is mainly focused on the application layer, it can also detect some errors in the lower layers. /20/

3.1 Test Set-up

Figure 10 shows the test set-up structure and Figure 11 shows the hardware connection.

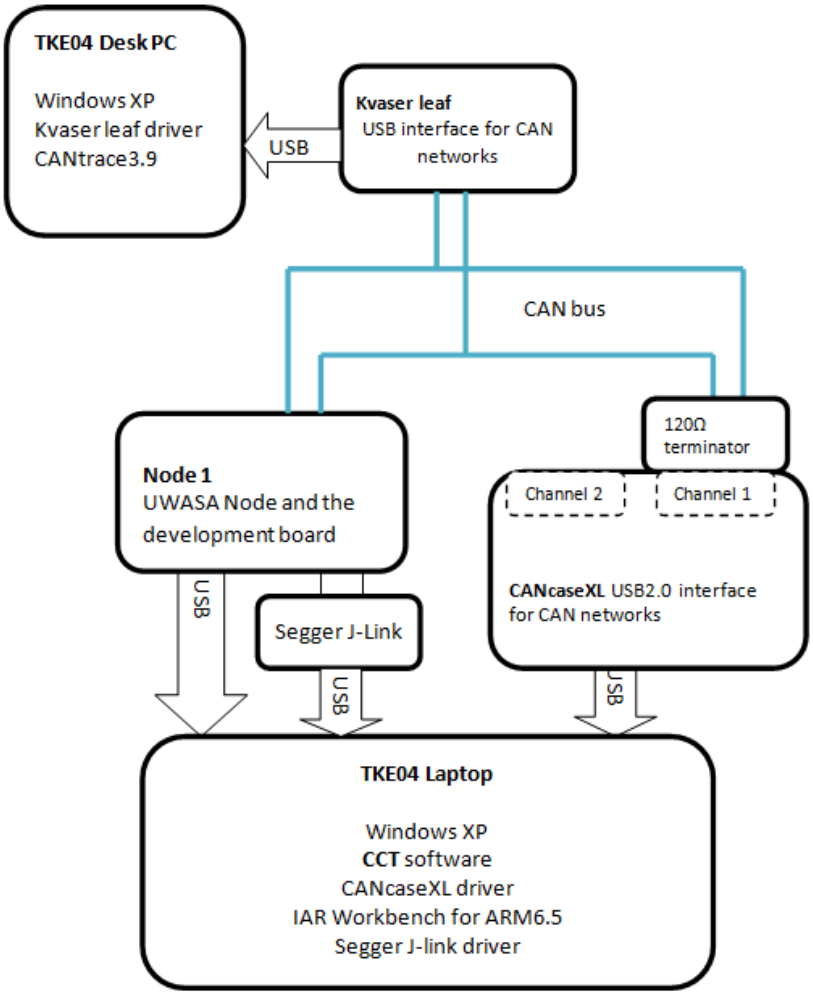

Figure 10 CANopen Conformance Test Set-up: Structure

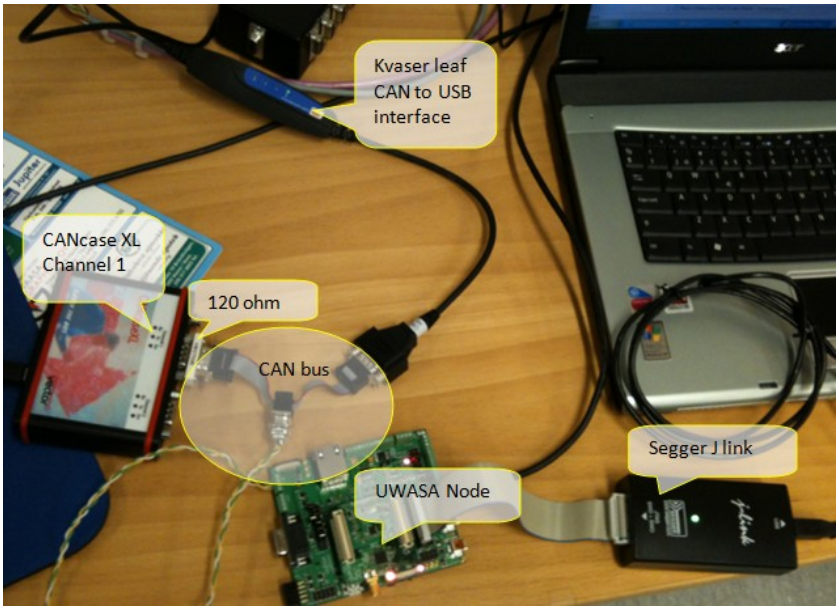

Figure 11 CANopen Conformance Test Set-up: Hardware Connection

Below is a list of the hardware and the software used in the CANopen conformance test.

CANcaseXL

It is a USB 2.0 interface for CAN networks made by the company called Vector. It has two separate CAN channels. Channel 1 is used in the CCT. In the CCT, one 120 ohm resistor (terminator) is connected to the channel 1 before connecting CAN bus as shown in Figure 11 above.

Kvaser Leaf Professional HS

It is a one channel USB interface for CAN made by the company called Kvaser. It offers the possibility to easily connect several interfaces to a PC. /21/

In the CCT, Kvaser leaf is connected to the desk PC and the baud rate is configured in the CANtrace software.

CANtrace

CANtrace(version 3.9) is a piece of software that is used for analysing the CAN-bus, such as the monitoring, sending and logging of CAN-messages. /1/

TKE04 laptop

The operating system is windows XP. The CCT tool, the CANcaseXL driver, the Segger J-link driver as well as the IAR Embedded Workbench for ARM 6.50 are installed on this laptop.

TKE04 desk PC

The operating system is windows XP. The CANtrace3.9 and the Kvaser leaf driver are installed on this computer.

3.2 Test Procedure

The CANopen conformance test can be started after building the testing environment. The CCT software consists of the EDS test and the device test. Before the test starts, some parameters need to be set. The UWASA Node used the following parameters:

- Node ID: 1
- Baud rate: 500 kbps
- Vendor ID: 1234 (a temporary vendor ID, only used at current stage)

After configuring the baud rate of the CANcaseXL and Kvaser leaf Professional HS to 500 kbps, the EDS file should be imported to the CCT tool and parameters set as shown in Figure 12.

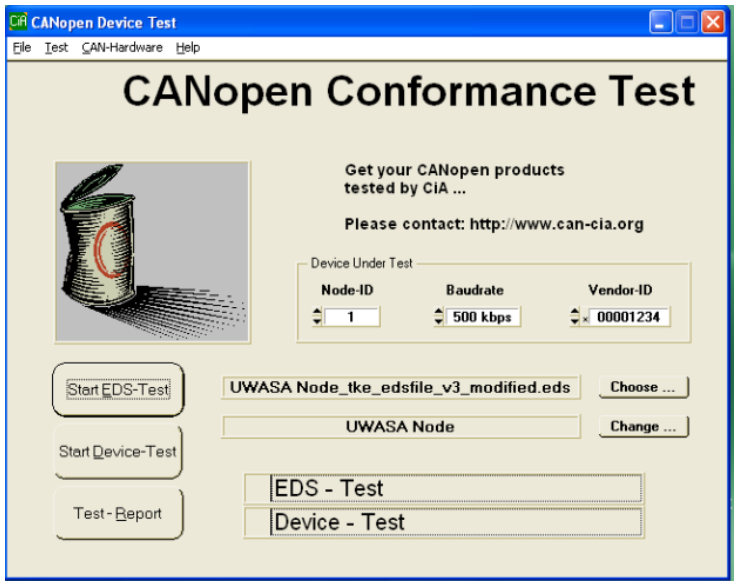

Figure 12 CANopen Conformance Test Tool Interface

EDS Test

The EDS file is very important for the CANopen device. Thus it is necessary to test the EDS file. Figure 13 shows the EDS test window of the CCT tool.

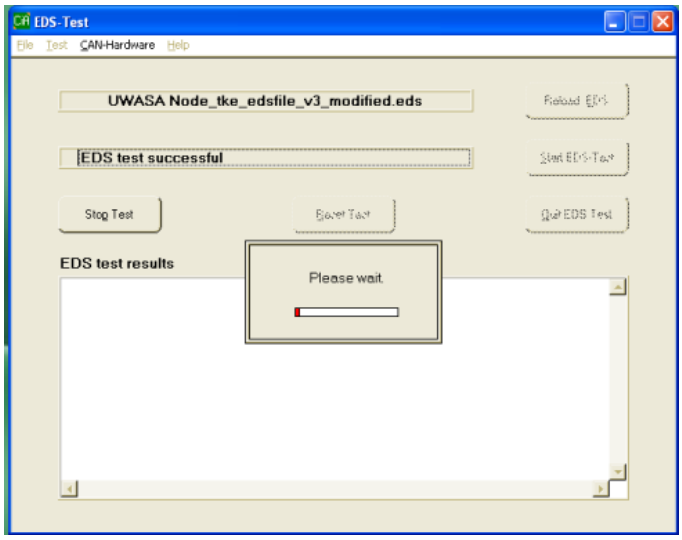

Figure 13 CCT Tool - EDS Test Window

The CCT tool reads the EDS file and then compares the entries according to the test procedure and generates a test report. The test shows that the following requirements are fulfilled by the contents of the EDS:

- Value ranges are not exceeded,
- Mandatory entries are supported,
- References are pointing to existing entries, and
- The EDS is consistent. /22/

Device Test

The device test consists of two parts, the protocol verification and state and transition tests. Figure 14 illustrates the window of the device test in the CCT tool.

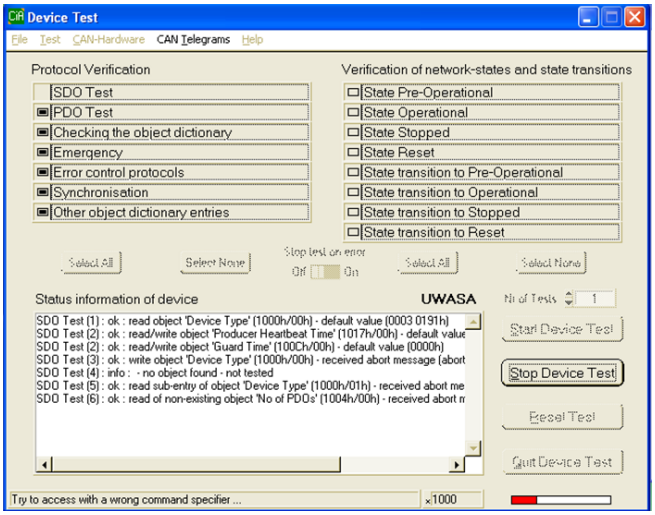

Figure 14 CCT Tool - Device Test Window

During testing, the test telegram function can be turned on and the test log can be saved after running the CANopen Conformance Test. Figure 15 shows the test log during the device test. The test log is very important when analyzing the test result later on.

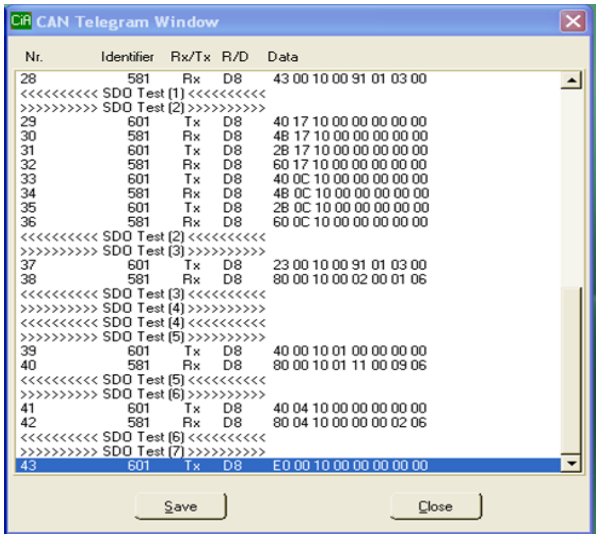

Figure 15 CCT Tool - CAN Telegram Window

3.3 Test Result

After running the CCT tool, a test report with test result (either fail or pass) will be generated and the test log can be used to analyze each test. The analyses of some failed test cases can be found in the appendix.

4 Performance Test

In the CANopen standard there are no specific performance criteria. However, it specifies how to measure and evaluate the performance criteria (see more in CiA 308 and CiA 313). In most applications, CANopen products should meet their manufacturers' specific performance criteria. In this thesis the performance of the UWASA Node was tested against the *Wärtsilä CANopen Device Specification* (WCDS, an internal document). The performance test software with generic test cases is made during this thesis work. This chapter introduces the performance test software and how to use it to test the UWASA Node as an example.

4.1 Performance Test Set-up

The performance test hardware environment should be set-up. The CAN interface used in this test is called Kvaser BlackBird SemiPro, which is a CAN-WLAN converter that can have one, two or three CAN channels depending on the application. In the performance test two channels of Kvaser BlackBird SemiPro and the UWASA Node are connected to the CAN-bus. The CANtrace3.9 is used to monitor the CAN-bus traffic during the test. Figure 16 shows the hardware connection of the performance test set-up.

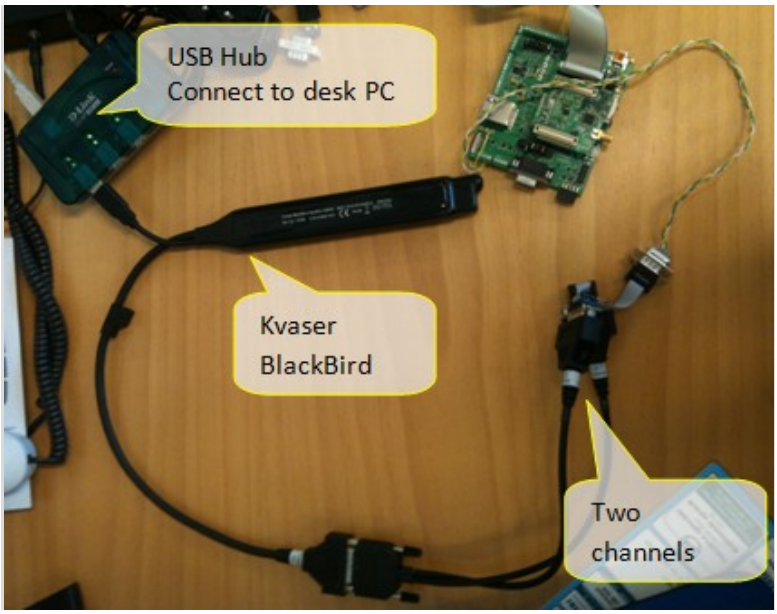

Figure 16 Hardware Connection of Performance Test

4.2 Developing of the Performance Test Software

The performance test software is written in C++. It is a windows 32 console application. The software was developed using Microsoft Visual Studio 2005. Since the Kvaser BlackBird SemiPro is used in the performance test, some functions of the Kvaser CANlib Application Programming Interface (API) are studied before developing the test cases. The Kvaser driver class is already inherited from the Kvaser CANlib API and there is a ready-to-use test framework, which asserts if two values are equal or non-equal and asserts if one value is true or false. Therefore my main job has been to develop the test cases.

4.2.1 Items that need to be tested

According to the *Wärtsilä CANopen Device Specification* and the *CiA 313 CANopen Performance Testing*, the following items should be tested:

SDO Read Response Time

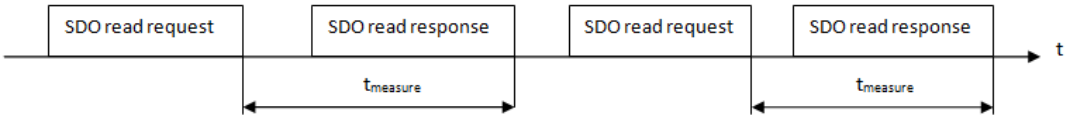

Figure 17 SDO Read Response Time/23/

For the objects with access type: read and write, or read only, try SDO read access and measure the response time ($t_{measure}$ in Figure 17).

SDO Write Response Time

For the objects with access type: read and write, or write only, try SDO write access and measure the response time.

PDO Cycle Time

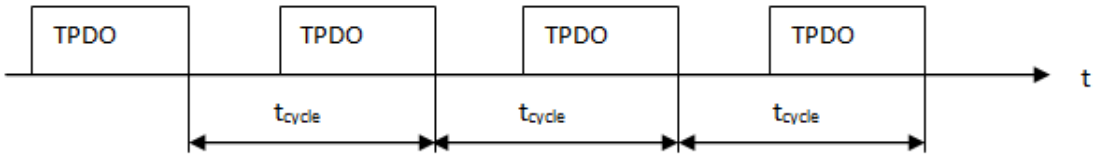

Figure 18 PDO Cycle Time/23/

The device shall have at least one Transmit PDO (TPDO). The PDO cycle time (t_{cycle} in Figure 18) is the maximum interval for PDO transmission if the

transmission type is set to FE_h / FF_h(event driven). The PDO cycle time should be configured in the event-timer of one TPDO. Then measure the actual PDO cycle time and see if it is within the maximum derivation of the PDO cycle time set in the event-timer. /24/

PDO RTR Response Time

If one Transmit PDO (TPDO) can be triggered by the remote transmission request (RTR), then trigger the TPDO by RTR and measure the response time. /24/

Boot-up Time

The boot-up time is the time from resetting the device to the boot-up message. /24/

LSS Response Time

If the device supports Layer Setting Service (LSS) protocol (CiA 305), then try to configure the device by LSS and measure the response time.

Heartbeat Producer Deviation

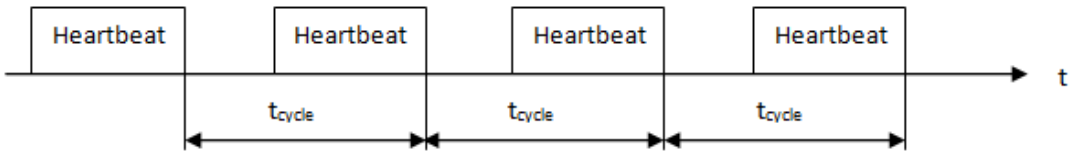

Figure 19 Producer Heartbeat time

The time (t_{cycle} in Figure 19) between two consecutive heartbeat messages is measured and it should be within the maximum deviation of the configured producer heartbeat time in the object 0x1017.

4.2.2 Design of Test Cases

This section illustrates some of the generic test cases that I have made. The following tests take the UWASA Node as an example to describe the approach of designing the test cases.

SDO Read/Write Response Time Test

These two test cases are to measure the SDO read/write response time of the device. The *CiA313 Performance Testing* divided the objects into different types based on their relationships to the purpose of the CANopen device. For example, objects such as output values, input values and command registers, which are mapped to PDOs, are type I objects. Type I objects are essential, performance-related objects that define the major purpose of the CANopen device. For different types of the objects, the SDO read/write response time tests are different. In this paper only the mandatory SDO read/write response times of type I objects without the bus traffic are measured. /24/

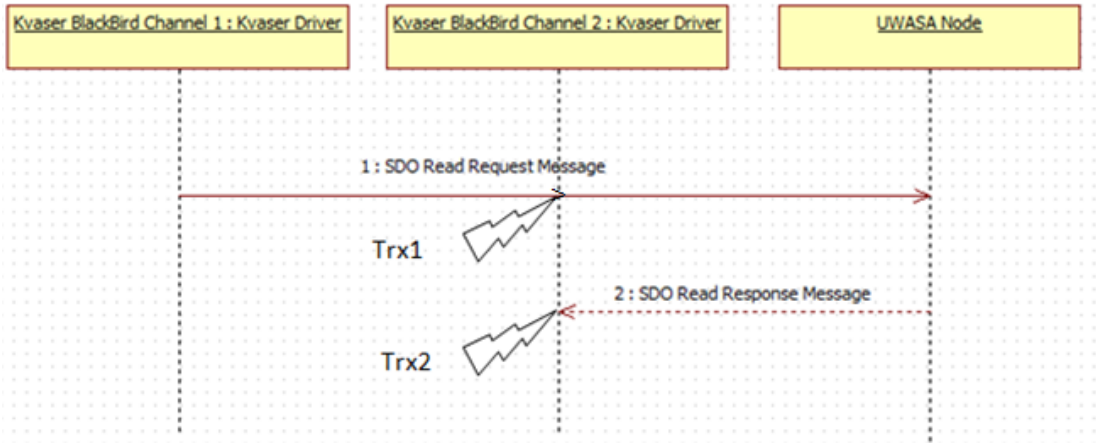

Figure 20 SDO Read Response Time Test

Take SDO read response time test as an example. Figure 20 shows how to measure the SDO read response time. Channel 1 of Kvaser BlackBird is used to send the SDO read request message to the CAN-bus. Then both channel 2 of Kvaser BlackBird and the UWASA Node receive the message. The time at which the SDO read request message arrives to channel 2 is marked as Trx1. Then the UWASA Node will send the SDO read response message to the CAN-bus. The time at which the SDO read response message arrives to channel 2 is marked as Trx2. Thus, the SDO read response time is equal to Trx2 - Trx1. Using a for-loop to repeat the above measurement for 1000 times, then the maximum, minimum and average SDO read response times are calculated. The flow chart of this test case is shown in Figure 21.

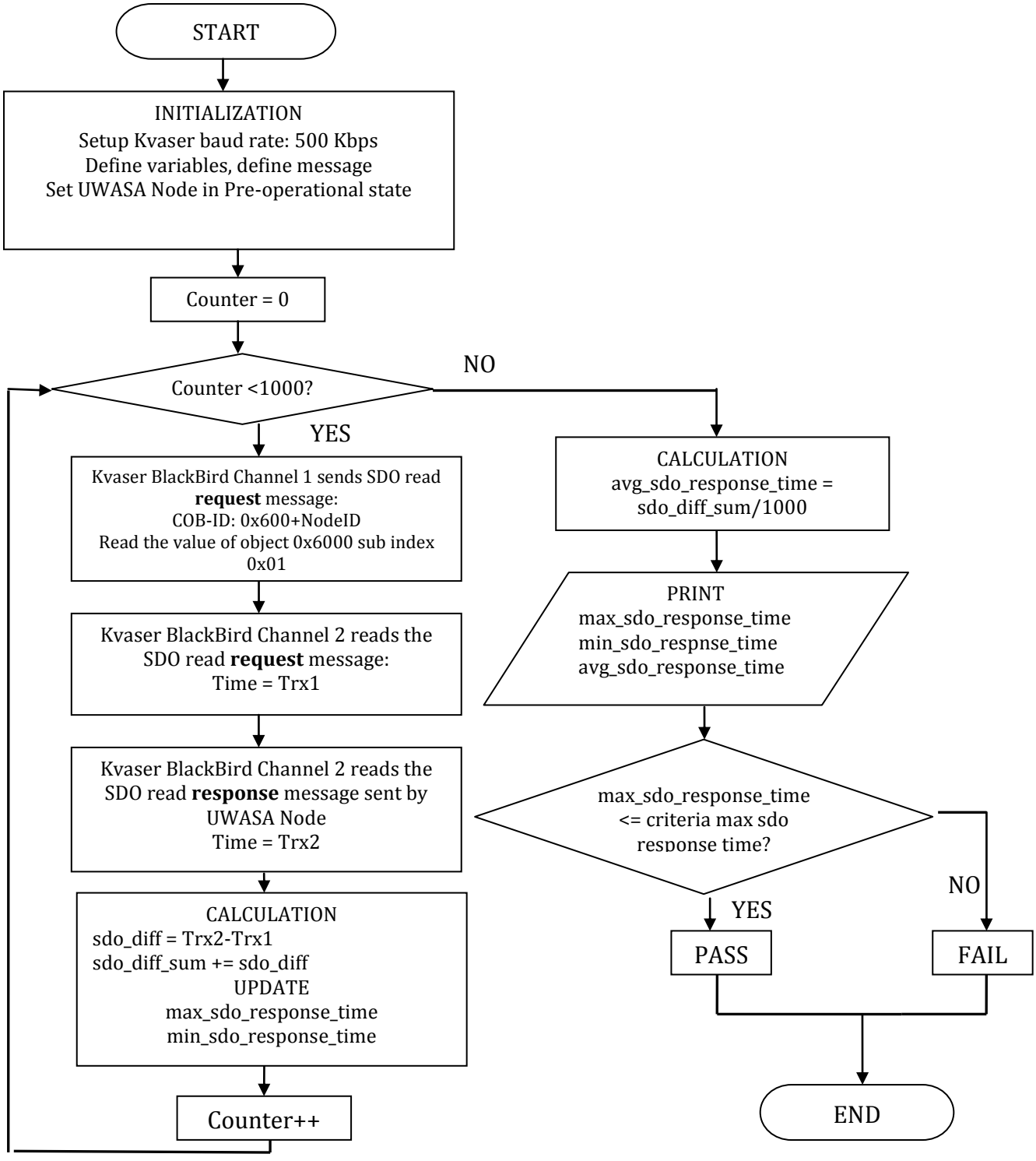

Figure 21 Flow Chart of SDO Read Response Time Test Case

Heartbeat Producer Deviation Test

This test is to test the time (t_{cycle} in Figure 19) between two consecutive heartbeat messages. The test steps that are listed below have been designed in order to test the UWASA Node:

1. Initialization

    - Set-up Kvaser's baud rate to 500 Kbps
    - Define variables, define messages
    - Kvaser BlackBird Channel 1 resets the UWASA Node and configures it as a heartbeat producer with a producer heartbeat time of 100 ms (write 0x64 to object 0x1017, write 0x00 to both object 0x100c and 0x100d to disable the guard time and the life time factor)
    - Kvaser BlackBird Channel 2 reads the first heartbeat message (COB-ID is 0x700+Node-ID) and stores the time stamp as `Trx_old`

2. Measurement (Inside the for-loop, repeat for 100 times)

    - Kvaser BlackBird Channel 2 reads the heartbeat message (COB-ID is 0x700+Node-ID) and stores the time stamp as `Trx_new`
    - Measure the t_{cycle} (Figure 19): `heartbeat_time = Trx_new - Trx_old;`
    - Update the total heartbeat time: `heartbeat_sum += heartbeat_time;`
    - Update the maximum- and minimum heartbeat
    - `Trx_old = Trx_new;`

3. Calculation

    Calculate the average heartbeat time: `avg_heartbeat_time = heartbeat_sum /100;`

4. Print

    Print the average, the maximum and the minimum heartbeat times on the screen

5. Assertion

    Check if the maximum and the minimum heartbeat times are within the maximum deviation (e.g. less than 5%)

    ```
    ASSERT_TRUE(max_heartbeat_time <= 105);

    ASSERT_TRUE( min_heartbeat_time >= 95);
    ```

    If the `max_heartbeat_time` and the `min_heatbeat_time` are within the maximum deviation, then the test is passed, otherwise the test is failed.

6. Switch off the heartbeat (write 0x00 to object 0x1017) and close the test.

4.2.3 Performance Test Software

After finishing all the test cases, the performance test software is ready-to-use. The menu of the performance test software is shown in Figure 22. As can be seen in the figure, the user can either type in the number 0 to run all the test cases in the software or type in the number on the left side of each test case to run the specified test case.

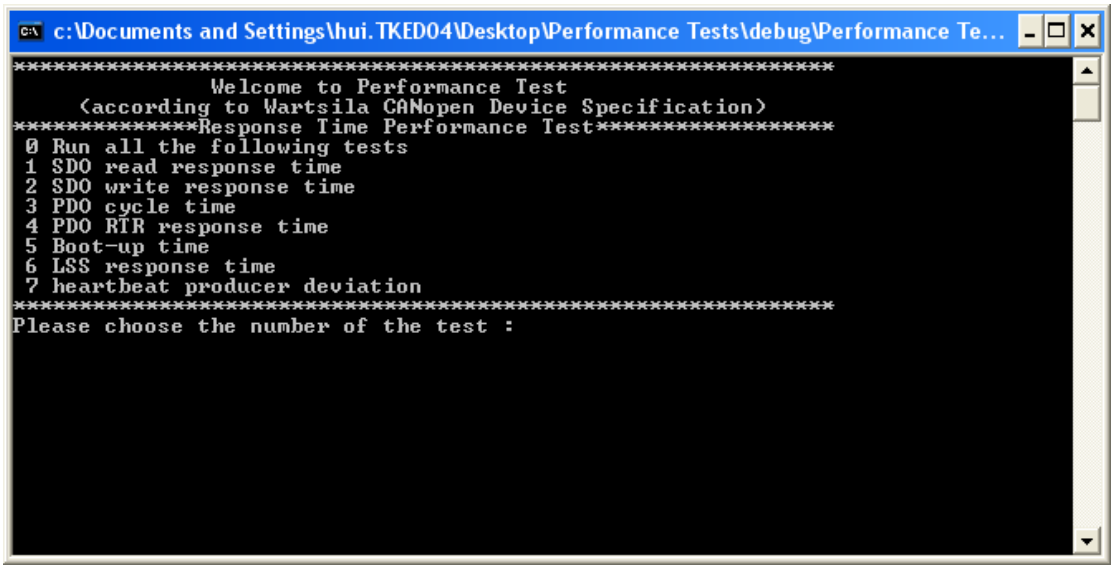

Figure 22 Menu of the Performance Test Software

4.3 Performance Test Result

After running this performance test software, the items in section 4.2.1 are tested and the test result with some relevant values (e.g. the corresponding average, maximum and minimum times) are printed on the screen. Then the next step is to integrate the test results to the test report for the UWASA Node. Below is an example of the performance test result in the test report.

Heartbeat Producer Deviation - 100ms

No traffic average: 55 ms

No traffic maximum: 100 ms

No traffic minimum: 1 ms

Criteria: Derivation: no more than 5%, in this case, the time should be between 95 ms and 105 ms

Result: Failed

As the text above shows, the UWASA Node has failed this test, since it has a minimum heartbeat time of 1 ms, which is much lower than 95 ms. Later on the designer will correct the source code of the UWASA Node based on this test report.

The UWASA Node has passed the SDO Read/Write Response Time Test, the PDO Cycle Time Test and the Boot-up Time Test. But it fails the PDO RTR Response Time Test and the LSS Response Time Test due to the fact that it does not support the PDO RTR and the LSS functionalities. The PDO RTR and the LSS functionalities are recommended by TK Engineering Oy, thus the software designer needs to implement these two features in the UWASA Node.

5 CiA DS401 Test

As mentioned earlier, the UWASA Node supports the digital inputs and the analogue inputs. Therefore the main task of the CiA DS401 test is to develop test cases to automatically test the functionality of digital inputs and analogue inputs of the UWASA Node.

5.1 CiA DS401 Test Set-up

The performance test set-up is also used in the CiA DS401 test (Refer to section 4.1).

5.2 CiA DS401 Test Software

Some of the test cases that have been developed are listed below:

Object 1000h Device Type Test

The default PDO mapping, digital inputs and analogue inputs were supported in the UWASA Node. Thus the value from bit-16 to bit-23 in object *0x1000 Device Type* should be set to the corresponding value (0x5) according to the CiA DS401 standard.

Accordingly, in this test case the Kvaser BlackBird Channel 1 sends a read request message to the CAN-bus to read the value of the object 0x1000, and then the UWASA Node will send the SDO response message to the CAN-bus. The Kvaser BlackBird Channel 2 will read the data of this SDO response message to check if the value from bit-16 to bit-23 is 0x5.

```
ASSERT_EQUAL(value, 5);
```

The “value” contains the data from bit-16 to bit-23. If the value is equal to 0x05, then the test is passed, otherwise, the test is failed.

Digital Inputs Polarity Test

The UWASA Node supports the polarity of the digital inputs. The corresponding bits of the digital inputs should be inverted when the polarity for the corresponding bits are enabled (Figure 8). For instance, if the digital inputs of the UWASA Node are 0xF0 and the polarity register is 0x00 (The polarity for all the 8-bits digital inputs are disabled), then the digital inputs should be 0xF0, not-inverted. If the polarity for all the 8-bits digital inputs is enabled (i.e. write 0xFF to the polarity register: object 0x6002 sub index 0x01), then the digital inputs should be 0x0F.

Transmission Trigger: Low to High Test

The UWASA Node only transmits the TPDO when the digital input changes from 0 to 1, if the corresponding bits in object *0x6007 Interrupt Mask Low To High 8 Bit* are enabled and the global interrupt register (0x6005) is enabled.

In order to simulate the change of the digital inputs, a specific way of making the test was created. This needs the interaction of the designer of the source code of the UWASA Node. The idea is to create the factory test registers to work as an intermediate digital inputs/analogue inputs register.

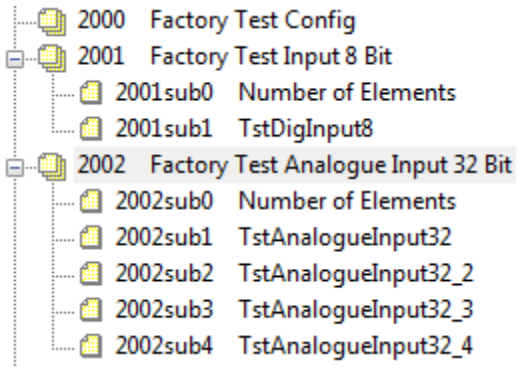

Figure 23 Factory Test Registers

As Figure 23 shows, the software designer of the UWASA Node needs to implement the following functionality to the source code of the UWASA Node:

When the *Factory Test Config* register (object 0x2000, type: boolean) is enabled, the value in the object 0x2001 (intermediate register of the digital inputs) will be copied to the digital inputs (currently the digital inputs are variables in the source code. The digital inputs are in the object *0x6000 Read Input 8 Bit*) and the values in the object 0x2002 (intermediate register of the analogue inputs) will be copied to the analogue inputs (The analogue inputs of the UWASA Node are all variables at current stage, they are in object *0x6401 Read Analogue Input 16 Bit*).

In this way, the digital/analogue inputs functionalities can be tested using the following test steps.

Test steps:

1. Reset node (default: all the interrupts are disabled)

2. Disable polarity, disable global interrupts, and disable all three kinds of interrupt masks

3. Configure TPDO1, set transmission type: event-driven device profile

4. Enable global interrupt

5. Enable *Factory Test Config* (object 0x2000)

6. Enable low-to-high interrupt mask for bit-1 (check low 4-bits digital inputs)

7. Set UWASA Node in operational state

8. Write 0x00 to object 0x2001(intermediate register of the digital inputs, will be called IRDI in following text)

9. Toggle bit-1 of the IRDI up, check the PDO. (The PDO should be sent and received)

10. Toggle bit-6 of the IRDI up and bit-1 down, check the PDO. (The PDO should not be sent)

11. Enable low-to-high interrupt mask for bit-6 (check high 4-bits digital inputs)

12. Write 0x00 to IRDI

13. Toggle bit-6 of the IRDI up, check the PDO. (The PDO should be sent and received)

14. Toggle bit-1 of the IRDI up and bit-6 down, check the PDO. (The PDO should not be sent)

15. Close

Cyclic Transmission Analogue Inputs Test

The UWASA Node supports the cyclic transmission mode for analogue inputs. Thus this is tested in the same way when testing the digital inputs. The object 0x2002 (intermediate register of the analogue inputs) is used in this test to simulate the change of the analogue inputs. The test case configures the TPDO2 (contain the analogue inputs) to be sent every 100 millisecond. Then change the value in object 0x2002 (intermediate register of the analogue inputs) and check if the new TPDO2 contains the new value. If the TPDO2 contains the new value for the analogue inputs, then the test is passed, otherwise the test is failed.

After finishing all the test cases, the CiA DS401 test software is ready-to-use. The menu of the software is shown in Figure 24. As can be seen in the figure, the user can either type in the number 0 to run all the test cases in the software or type in the number on the left side of each test case to run the specified test case.

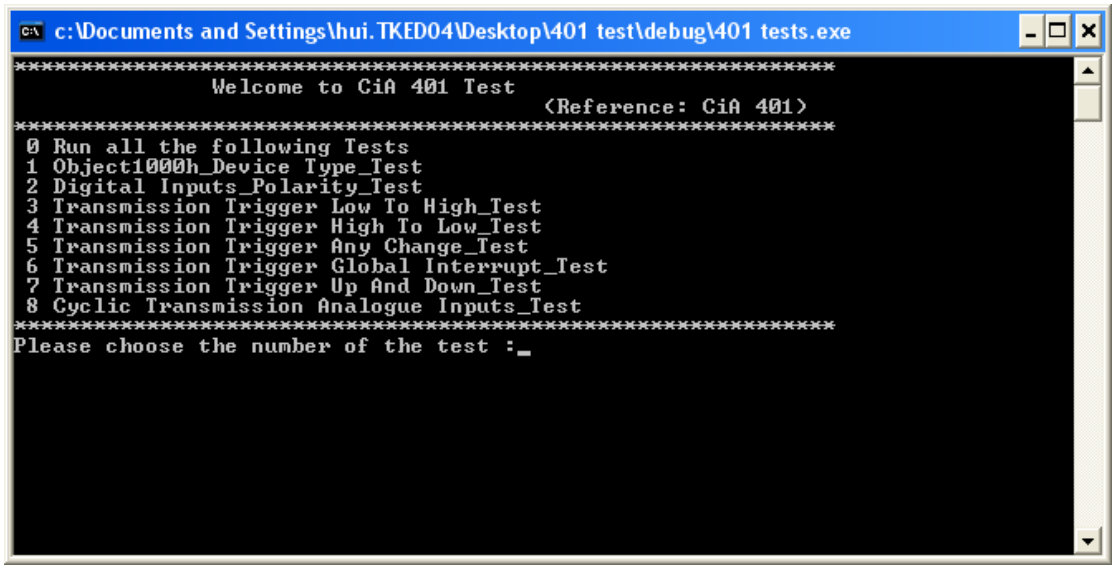

Figure 24 Menu of the CiA DS401 Test Software

5.3 CiA DS401 Test Result

After running the CiA DS401 test software to test the UWASA Node, the test result is printed on the screen (Figure 25). The UWASA Node has passed all the CiA DS401 test cases.

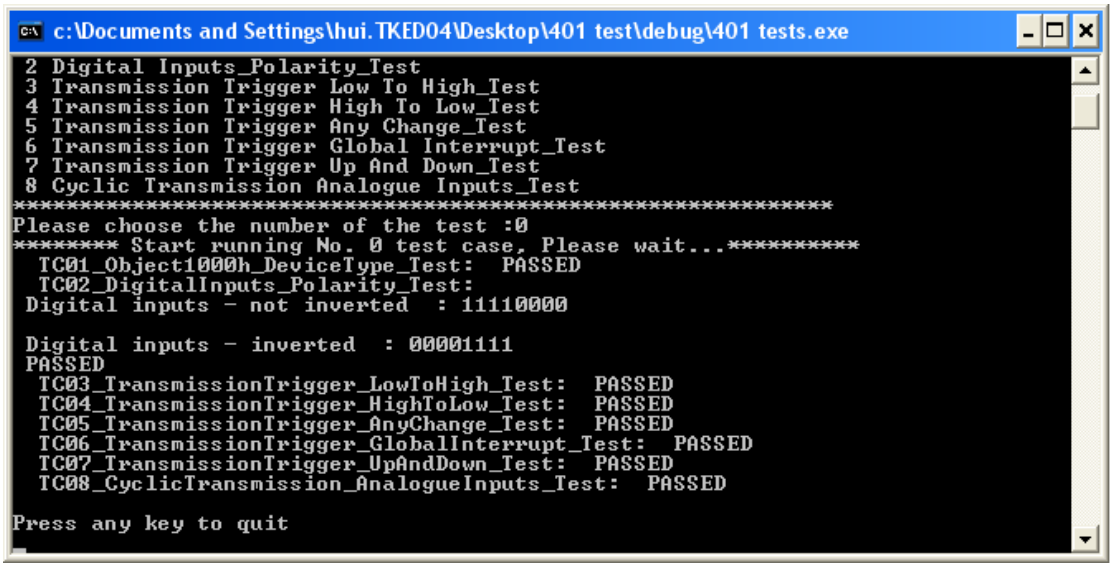

Figure 25 The CiA DS401 Test Result of the UWASA Node

6 Results and Conclusions

The software designer of the UWASA Node is working on the corrections based on the test report that I have made. The automatic performance test software and the automatic CiA DS401 test software can be used in the future to test other CANopen devices with a little modification of the test cases. The code of the automatic test software can be optimized in the future, if needed.

7 Discussion

This thesis is interesting and instructive. In this project I have spent quite much time to understand the CAN and the CANopen protocols and learnt how to develop test cases according to the CiA standards. Meanwhile, I have learnt that it is very important for the CANopen products to pass the relevant tests before they are released to the market. The automatic testing not only reports the bugs of the software and helps the software developer, but also saves time and money and improves the accuracy. Therefore, the automatic testing is needed at the developing stage of the software development cycle.

8 References

/1/ TK Engineering Oy. (w.y.)
www.tke.fi(retrieved 10.12.2012)

/2/ Redzheb, Ali, Y., Yigitler, H. (w.y.). *The UWASA Node Reference Manual.* Vaasa: University of Vaasa, Department of Computer Science Communication and Systems Engineering Group

/3/ CanFestival. (w.y.). *Free software CANopen framework*
http://www.canfestival.org/(retrieved 3.3.2013)

/4/ Olaf, P., Andrew, A. & Christian, K. (2003). *Embedded Networking with CAN and CANopen*. ISBN 0-929392-78-7
Page xv.

/5/ CAN in Automation (CiA). (w.y.) *About CAN in Automation (CiA)*
http://www.can-CiA.org/index.php?id=aboutCiA (retrieved 2.2.2013)

/6/ Olaf, P., Andrew, A. & Christian, K. (2003). *Embedded Networking with CAN and CANopen*. ISBN 0-929392-78-7
Page 18.

/7/ Tech-FAQ. (w.y.). *The OSI Model – What It Is; Why It Matters; Why It Doesn't Matter.*
http://www.tech-faq.com/osi-model.html (retrieved 3.2.2013)

/8/ Frank, P. (2008). *CANopen Basics Introduction.*
http://www.canopensolutions.com/english/about_canopen/about_canopen.shtml(retrieved 1.3.2013)

/9/ CAN in Automation (CiA). *CAN Specification 2.0, Part A*
http://www.can-CiA.org/index.php?id=520(retrieved 18.2.2013)

/10/ Jeff, T. (w.y.). *How OSI works*
http://computer.howstuffworks.com/osi1.htm(retrieved 18.2.2013)

/11/ Olaf, P., Andrew, A. & Christian, K. (2003). *Embedded Networking with CAN and CANopen*. ISBN 0-929392-78-7
Pages 114-115.

/12/ Frank, P. (2008). *CANopen Basics - Communication*
http://www.canopensolutions.com/english/about_canopen/communication.shtml(retrieved 10.3.2013)

/13/ Olaf, P., Andrew, A. & Christian, K. (2003). *Embedded Networking with CAN and CANopen*. ISBN 0-929392-78-7
Page 44.

/14/ CAN in Automation (CiA). (2012). *CiA 301 Work Draft CANopen application layer and communication profile. Version: 4.2.0.74*

/15/ Martin, R. (w.y.) *Configuration Guideline for CANopen Networks*

http://www.can-CiA.org/fileadmin/CiA/files/icc/9/rostan.pdf (retrieved 11.3.2013)

/16/ Beckhoff Information System. (w.y.). *BECKHOFF Fieldbus Components: CANopen Communication: Process Data Objects (PDO)* http://infosys.beckhoff.com/italiano.php?content=../content/1040/fc510x/html/co_comprocessdata.htm&id= (retrieved 14.3.2013)

/17/ Frank, P. (2008). *CANopen Basics - Network Management* http://www.canopensolutions.com/english/about_canopen/canopen-management.shtml(retrieved 14.3.2013)

/18/ Olaf, P., Andrew, A. & Christian, K. (2003). *Embedded Networking with CAN and CANopen*. ISBN 0-929392-78-7
Pages 84-85.

/19/ CAN in Automation (CiA). (2012). *CiA 401 Final Work Draft Device profile for generic I/O modules. Version: 3.0.83*

/20/ CAN in Automation (CiA). (w.y.). *CANopen conformance test tool* http://www.can-cia.org/index.php?id=conformance(retrieved 20.3.2013)

/21/ Kvaser AB. (2013). *Kvaser Leaf Professional HS* http://www.kvaser.com/index.php?option=com_php&Itemid=258&eaninput=7330130002432(retrieved 20.3.2013)

/22/ CAN in Automation (CiA). (2001). *CANopen Conformance Test Version 2.0 Help*

/23/ CAN in Automation (CiA). (2006). *CiA 308 Performance measurement basics. Version: 1.0.1*

/24/ CAN in Automation (CiA). (2006). *CiA 313 Work Draft Proposal Performance testing. Version: 0.0*

Part of the CANopen Conformance Test Report

Below is part of the test report generated by the CANopen Conformance Test Tool (version 2.0.02) and the corresponding analysis with the help of the test log generated by the CCT tool and the CiA DS301 standard.

```
**************************************

        CANopen Test Suite

        ^^^^^^^^^^^^^^^^^^^

    (c) by CAN in Automation (CiA)

        National Instruments

        2000, 2001

**************************************
```

Device : UWASA Node V0.1

Company : Uwasa

EDS-File: UWASA Node_tke_edsfile_v3_modified.eds V1.2

Tester : Hui Liang

Result : FAILED at Tue Feb 26 12:00:04 2013

****

Device Test result:

SDO Test (1) : ok : read object 'Device Type' (1000h/00h) - default value (0003 0191h)

SDO Test (2) : ok : read/write object 'Producer Heartbeat Time' (1017h/00h) - default value (0000h)

SDO Test (2) : ok : read/write object 'Guard Time' (100Ch/00h) - default value (0000h)

...

SDO Test (10) : error : write object (1F51h/01h) with value higher than highlimit – accepted

The test log of the SDO Test (10) is:

```
>>>>>>>>>> SDO Test (10) >>>>>>>>>>

Line No.   COB-ID Direction Data length    Data (byte 0 to byte 7)

53          601        Tx          D8          40 51 1F 01 00 00 00 00
```

54 581 Rx D8 4F 51 1F 01 00 00 00 00

55 601 Tx D8 2F 51 1F 01 FF 00 00 00

56 581 Rx D8 60 51 1F 01 00 00 00 00

***** SDO Test (10) : error : write object (1F51h/01h) with value higher than highlimit - accepted *****

57 601 Tx D8 2F 51 1F 01 00 00 00 00

58 581 Rx D8 60 51 1F 01 00 00 00 00

<<<<<<<<<< SDO Test (10) <<<<<<<<<<

Analysis:

According to the CiA DS301 standard, the structure of the SDO write request message and the structure of the SDO write response message are shown in the following tables.

Table 1 The Structure of the SDO Write Request Message

Byte 0	Byte 1-2	Byte 3	Byte 4-7
Command byte	OD index	OD sub index	Data (max. 4 bytes)

Table 2 The Structure of the SDO Write Response Message

Byte 0	Byte 1-2	Byte 3	Byte 4-7
Command byte	OD index	OD sub index	Empty bytes*

*: If the SDO request is unsuccessful, a SDO abort message should be sent by the device. Byte 4 to byte 7 will contain the abort code and the byte 0 "command specifier" will be 0x80.

In the test log above, line 55 "2F 51 1F 01 FF 00 00 00" means that the CCT tool wrote 0xFF to object 0x1F51 sub index 0x01 and the UWASA Node responded with the message "60 51 1F 01 00 00 00 00" where the "command specifier" 0x60 means that the UWASA Node accepted this SDO request. However, in the EDS file, the high limit of 1F51h/01h is 0x2. Therefore, in this case, the UWASA Node should respond with an abort message instead of accepting 0xFF.

According to the help document of the CCT tool, the correct abort code is: 0609 0031h – value of parameter written too high, or 0609 0030h – value range of parameter exceeded (only for write access), or 0800 0000h – general error.

…

PDO Test (16) : error : write object 'TPDO2 / mapping' - not map able - accepted

PDO Test (16) : error : write object 'TPDO1 / mapping' - not map able – accepted

Analysis:

In this test, the CCT tries to map an existing object that can't be mapped to any supported TPDO. The device should return the abort message with the abort code:

0604 0041h – objects cannot be mapped to the PDO or object does not exist in the object dictionary: 0602 0000h

The UWASA Node should report abort message instead of accepting.

Test Log:

>>>>>>>>>> PDO Test (16) >>>>>>>>>>

5556	601	Tx	D8	23 01 18 01 81 00 00 80
5557	581	Rx	D8	60 01 18 01 00 00 00 00
5558	601	Tx	D8	2F 01 1A 00 00 00 00 00
5559	581	Rx	D8	60 01 1A 00 00 00 00 00
5560	601	Tx	D8	23 01 1A 01 08 00 01 64
5561	581	Rx	D8	60 01 1A 01 00 00 00 00
5562	601	Tx	D8	2F 01 1A 00 01 00 00 00
5563	581	Rx	D8	60 01 1A 00 00 00 00 00
5564	601	Tx	D8	23 01 18 01 81 00 00 80
5565	581	Rx	D8	60 01 18 01 00 00 00 00

***** PDO Test (16) : error : write object 'TPDO2 / mapping' - not map able - accepted *****

5566	601	Tx	D8	23 00 18 01 81 01 00 80

5567	581	Rx	D8	60 00 18 01 00 00 00 00
5568	601	Tx	D8	2F 00 1A 00 00 00 00 00
5569	581	Rx	D8	60 00 1A 00 00 00 00 00
5570	601	Tx	D8	23 00 1A 01 08 00 01 64
5571	581	Rx	D8	60 00 1A 01 00 00 00 00
5572	601	Tx	D8	2F 00 1A 00 01 00 00 00
5573	581	Rx	D8	60 00 1A 00 00 00 00 00
5574	601	Tx	D8	23 00 18 01 81 01 00 80
5575	581	Rx	D8	60 00 18 01 00 00 00 00

***** PDO Test (16) : error : write object 'TPDO1 / mapping' - not map able - accepted *****

<<<<<<<<<< PDO Test (16) <<<<<<<<<<

...