

ESX-TC3G

User Manual



Version of this document: V1.03r0

List of Contents

1	Index of Abbreviations.....	6
2	Introduction	13
3	General Information	14
3.1	Contact	14
3.2	Copyright.....	15
3.3	Declaration of Conformity.....	16
3.4	Warranty	17
3.5	Documents	17
3.6	Target Group	17
3.7	Used Symbols and Formats.....	18
3.8	Disposal	19
3.9	TC3G_History	20
4	Getting started with TC3G	22
4.1	Additional Available Supplies.....	22
4.2	Insert SIM Card.....	22
4.3	Connect TC3G.....	23
4.4	Connect to Development Box	23
4.5	Setting up the Serial Interface.....	23
4.6	Power up Device	24
4.7	How to Secure the System	25
5	Hardware.....	27
5.1	Block Diagram	27
5.2	Technical Data.....	28
5.2.1	Power Supply	28
5.2.2	Processor and System Memory	28
5.2.3	Communication Interfaces.....	29
5.2.4	Inputs and Outputs	30
5.2.5	System Data	31
5.3	Pin Assignment.....	32
5.4	Mounting Guidelines.....	33
5.5	Connecting Guidelines	36
5.6	Housing	36
5.7	Connector.....	38
5.8	Antenna.....	38
6	Software	40
6.1	System Information.....	40
6.2	Boot up / Shut down sequence.....	42
6.3	Wakeup	44
6.3.1	Ignition	45
6.3.2	Real Time Clock	46
6.3.3	Motion Sensor	48

6.4	Network handling.....	50
6.4.1	GSM	51
6.4.2	Ethernet	56
6.4.3	WLAN	58
6.4.4	IP handling	64
6.4.5	CAN	66
6.4.6	Bluetooth	67
6.5	Memory handling.....	71
6.5.1	NOR Flash.....	74
6.5.2	NAND Flash	82
6.5.3	EEPROM	83
6.6	Watchdog.....	85
6.7	Temperature Sensor	86
6.8	GPS	86
6.9	Serial.....	87
6.10	Digital Input / Output.....	87
6.11	Universal Serial Bus.....	89
6.12	Beeper	90
6.13	LED.....	91
7	Teleservice Application Framework.....	93
7.1	Overview	93
7.2	TAF Components.....	94
7.2.1	System daemon	95
7.2.2	Data daemon	97
7.2.3	Logger daemon	99
7.2.4	GPS daemon.....	101
7.2.5	GSM daemon	103
7.2.6	Network daemon.....	107
7.2.7	Server daemon.....	112
7.2.8	Signal daemon	116
7.2.9	Mail daemon.....	122
7.2.10	machines.cloud daemon.....	129
7.3	TAF Library	139
7.3.1	Introduction	139
7.3.2	Notation	141
7.3.3	D-Bus Utils.....	148
7.3.4	System.....	166
7.3.5	Datapool	173
7.3.6	Datalogger.....	203
7.3.7	GPS.....	243
7.3.8	SMS	246
7.3.9	Network	251
7.3.10	Server	256
7.3.11	GSM	263
7.3.12	Signal.....	265
7.3.13	Utils.....	269

8	Development Tools.....	286
8.1	Create Own Root File System.....	286
8.1.1	Download, extract and test setup.....	288
8.1.2	Adapt the root file skeleton.....	289
8.1.3	Enable or Disable Buildroot Packages.....	289
8.1.4	Extend BR by adding packages.....	293
8.1.5	STW Build Scripts	293
8.2	Create Own Application	295
8.2.1	Create New Project.....	295
8.2.2	Code Blocks.....	297
8.2.3	Command Line	300
8.3	Toolchain.....	302
8.3.1	Linux.....	302
8.3.2	Windows	312
8.3.3	Examples.....	319
8.3.4	Libraries	320
9	Update the Device	321
9.1	BSP Components.....	321
9.2	Windows Updater	322
9.3	Linux Updater.....	324
9.4	STW Update Mechanism.....	328
9.4.1	STW Update Procedure	328
9.4.2	STW Update Procedure Flow Charts.....	333
10	Application Notes	335
10.1	Communication Interfaces.....	335
10.1.1	Setting up the Serial Interface	335
10.1.2	TFTP	336
10.1.3	NFS.....	336
10.1.4	Telnet	337
10.1.5	I/O Pin's.....	339
10.1.6	GSM	340
10.1.7	CAN	341
10.1.8	Bluetooth M2M	343
10.1.9	E-Mail	346

11 Utilities Tools.....	348
11.1 stw_dptool	348
11.2 stw_GetGPS.....	349
11.3 stw_ReadACC	350
11.4 stw_acc2can.....	352
11.5 stw_show_gps.....	353
11.6 stw_flash_client	354
11.7 stw_SendSMS.....	355
11.8 stw_RecvSMS	355
11.9 stw_CANsnapshot	356
11.10 stw_GetGSMinfo	357
11.11 stw_RetentionRule.....	359
11.12 stw_TrafficMonitor	360
11.13 can_bridge - CAN interface bridging tool.....	361
11.14 kefex_client - STW KEFEX console client.....	362
11.15 Lighttpd Webserver.....	366
12 Open Source Licenses	367
12.1 Linux Kernel License	367
12.2 U-Boot License	373
12.3 busybox License	377
12.4 uClibc License	383
13 Qualification Tests	391
13.1 Compliance Information	391
13.2 Electrical Safety	391
13.3 Electromagnetic Compatibility e1/E1	394
13.4 Electromagnetic Compatibility CE.....	395
13.5 EMC Radio and Telecommunications Terminal Equipment.....	395
13.6 Environmental Influences	396
14 Index.....	398

1 Index of Abbreviations

Abbreviation	Meaning
acm	admission control mandatory
AES	Advanced Encryption Standard
AP	Access Point
API	application programming interface
APN	Access Point Name
bcc	Blind Carbin Copy (e-mail)
bdaddr	Bluetooth Address
BIOS	Basic Input Output System
BLOB	Binary Large Objects
BSP	Board Support Package
BT	Bluetooth
CAN	Controller Area Network
CANopen	Controller Area Network open (protocol)
CBC-MAC	cipher block chaining message authentication code
cc	Carbin Copy (e-mail)
CCMP	Counter-Mode/CBC-MAC Protocol
CEST	Central European Summer Time
cmd	Command
COF	coefficient of friction
COL2	CANopen Layer 2
COM1	Communication Interface No 1
CPU	Central Processor Unit

Abbreviation	Meaning
D-Bus	Desktop-Bus
DF	Data Flash
DFS	Dynamic Frequency Selection
DHCP	Dynamic Host Configuration Protocol
DLC	Data Logging Configuration
DLF	Data Log File
DNS	Domain Name System
DNSMASQ	DNS forwarder and DHCP server for small computer networks
DPL	Data Pool List
DSL	Digital Subscriber Line
DSUB	D sub-miniature (connector design)
EDR	enhanced data rate
EEPROM	electrically erasable programmable read-only memory
EGNOS	European Geostationary Navigation Overlay Service (SBAS)
EIRP	equivalent isotropically radiated power
EOF	End of File
ETH	Ethernet
FBE	Free Buffer Enquiry
FDT	Flattened Device Tree
FIT	Flattened Image Tree
FPU	Floating Point Unit
FRAM	Ferroelectric Random Access Memory
ftp	File Transfer Protocol

Abbreviation	Meaning
ftpd	FTP Daemon
GAGAN	GPS Aided GEO Augmented Navigation (indian SBAS)
GCC	GNU Compiler Collection
GCC	Galileo Control Center
GDB	GNU Project debugger
GID	Group Identifier (Linux)
GLONASS	Globalnaja nawigazionnaja sputnikowaja sistema (russian satellite navigation system)
GND	GNU Debugger
GNSS	Global Navigation Satellite System
GNU	GNU's Not Unix (recursive acronym) Free Software
GPL	GNU General Public License
GPO	Group Policy object
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Groupe Spécial Mobile (Global System for Mobile Communications)
GUI	graphical user interface
HCI	Host Controller Interface
HSPA	High Speed Packet Access (extension for UMTS)
HTTP	Hypertext Transfer Protocol
hwclock	Hardware Clock (RTC)
IDE	integrated development environment
IEEE	Institute of Electrical and Electronics Engineers

Abbreviation	Meaning
IGN	Ignition
IPC	Internet Protocol Communications
ISCAN	Inquiry Scan Flag (Bluetooth device is detectable)
ISO	International Organization for Standardization
JFFS2	Journalling Flash File System version 2
KL15	Klemme 15 (name of pin for ignition signal)
LFFORMAT	Logger File Format
LFJNAME	Log File Job Name
LGPL	Lesser General Public License (GNU)
LTS	Long Term Support (UBUNTU Linux)
M2M	Machine to Machine
MAC	Media Access Control
MCC	Mobile Country Code
MDT	Memory Technology Device (Subsystem for Linux)
mmc	Microsoft Management Console (Certificates)
MNC	Mobile Network Code
MOT	Motion Sensor (wake-up signal)
MSAS	Multi-functional Satellite Augmentation System (japanese SBAS)
mtd	Memory Technology Device
MTU	maximum transmission unit
NAND	Not AND (logical gate)
NAT	Network Address Translation
NFS	Network File System

Abbreviation	Meaning
NMEA	National Marine Electronics Association (standards for GPS communication)
NOR	Not Or (logical gate)
NRTI	Network Response Time Indicator
OBEX	Object Exchange (protocol)
OS	Operating System
PAP	Password Authentication Protocol
PFX	Personal information exchange (certificates)
pid	Process Identifier
PIN	Personal Identification Number
ppc	Power PC
PPP	Point-to-Point Protocol
pppd	PPP Dial
PSCAN	Page Scan Flag (Bluetooth device accepts connection requests)
PSK	Pre-Shared Key
QZSS	Quasi-Zenith Satellite System (japanese SBAS)
RAM	random access memory
RFCOMM	Radio Frequency Communication
ROM	read only memory
rootfs	Root File System
RTC	Real Time Clock
RTS	Runtime System
RWD	Read Write DirectMessage (user permission)
SBAS	Satellite Based Augmentation System

Abbreviation	Meaning
SIM	subscriber identity module
SMA	Sub Miniature version A (connector)
smime	Secure / Multipurpose Internet Mail Extensions
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNTP	Simple Network Time Protocol
src	source (Linux folder)
srv	Service (Linux folder)
ssh	secure shell
SSID	Service Set Identifier
ssl	Secure Sockets Layer
stderr	Standard Error Output (Linux)
stdout	Standard Output (Linux)
SW	Software
TAF	Teleservice Application Framework
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TFTPD	Trivial File Transfer Protocol Daemon (server)
TKIP	Temporal Key Integrity Protocol
tty	Input- and output device interface
UDP	User Datagram-Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus

Abbreviation	Meaning
UTC	Universal Time Coordinated
UTF	Universal Character Set Transformation Format
VDI	Virtual Box Disk Image
VI	visual (text editor within Linux)
WAAS	Wide Area Augmentation System (SBAS)
WEP	Wired Equivalent Privacy (WLAN encryption)
WLAN	Wireless Local Area Network
WMM	Wi-Fi Multimedia
WPA	Wi-Fi Protected Access (WLAN encryption)
XML	Extensible Markup Language
XOR	exclusive OR (logic gate)

2 Introduction

This is the user manual of the on-board module TC3G. It describes all features of the hardware and software of the TC3G.

The TC3G is a linux based communication and teleservice module with multiple interfaces for practically every conceivable application.

Technology provided by the TC3G

- On-board connectivity: Integration into on-board vehicle and machine control CANbus and Ethernet networks; reading of your relevant machine data (machine status, operating hours, maintenance warnings).
- Wireless connectivity: The TC3G is available with WiFi and Bluetooth, cellular communication, and GNSS functionality.
- Cloud connectivity: Prepared for connecting to STW's "machines.cloud" platform; storing of information regarding vehicle parameters, settings and behavior. Simple and straightforward connection to other server and cloud platforms.

Application examples

- Connecting fleet vehicles to optimize operating hours
- Forward planning of maintenance and repairs to minimize or avoid unplanned downtime
- Possibility for remote maintenance e.g. transferring software updates



3 General Information

3.1 Contact

Germany	<p>Sensor Technik Wiedemann GmbH Am Bärenwald 6 D-87600 Kaufbeuren Phone: +49-8341-9505-0 Fax: +49-8341-9505-55 Web: www.sensor-technik.de (http://www.sensor-technik.de/) Email: info@sensor-technik.de (mailto:info@sensor-technik.de) Email support: support@sensor-technik.de (mailto:support@sensor-technik.de)</p>
USA	<p>STW Technic, LP Electronic Controls 3000 Northwoods Pkwy. Suite 240 Norcross, GA 30071 Phone: (770) 242-1002 Fax: (770) 242-1006 Web: www.stw-technic.com (http://www.stw-technic.com/) Email: sales@stw-technic.com (mailto:sales@stw-technic.com)</p>
UK	<p>Sensor-Technik UK Ltd. Unit 21M, Bedford Heights Business Centre, Manton Lane, Bedford Bedfordshire, MK41 7PH Phone: +44 (0)1234-270770 Fax: +44 (0)1234-348803 Web: www.sensor-technik.co.uk (http://www.sensor-technik.co.uk/) Email: support@stwtechnic.freshdesk.com (mailto:support@stwtechnic.freshdesk.com)</p>

If you have found an error in this document, or have a suggestion as to how this document could be improved, please write to documentation@sensor-technik.de (<mailto:documentation@sensor-technik.de>)

3.2 Copyright

Copyright © Sensor-Technik Wiedemann GmbH 24.07.2017
Am Bärenwald 6, 87600 Kaufbeuren, Germany
All rights reserved.

The information provided in this document contains function descriptions that in case of actual use do not always apply as described due to the configuration of the product. An obligation to provide the respective functions shall only exist if expressly agreed in the terms of contract.

Subject to change without prior notice.

ESX and powerMELA are registered trademarks of the Sensor-Technik Wiedemann GmbH.

Other product names, companies, logos, and other brands that are referenced in this documentation are the property of their respective owners.

3.3 Declaration of Conformity

Pioneering new technologies
Pioneering new technologies



Sensor-Technik Wiedemann GmbH
Am Bärenwald 6 · 87600 Kaufbeuren · DEUTSCHLAND

EU-Konformitätserklärung EU-Declaration of Conformity

Hiermit erklären wir, dass das nachfolgende Produkt
We hereby declare that the following product

Teleservicemodul ESX-TC3G / Teleservice module ESX-TC3G

den grundlegenden Sicherheits- und Gesundheitsanforderungen der
nachfolgend aufgeführten EU-Richtlinien entspricht.
complies with the essential health and safety requirements of the following EU Directives.:



RoHS-Richtlinie / Restriction of the use of certain hazardous substances Directive (RoHS)	2011/65/EU
Funk-Richtlinie / Radio (and Telecommunication Terminal) Equipment Directive (RED/RTTE)	2014/53/EU

Angewandte harmonisierte Normen: Applied harmonized standards:

Angewandte harmonisierte Normen für RoHS-Richtlinie:
Applied harmonized standards for RoHS Directive:
EN 50581:2012

Angewandte sonstige Normen und technische Spezifikationen: Applied other standards and technical specifications:

Sonstige angewandte Normen für Funkanlagen-Richtlinie:
Other applied standards for Radio (and Telecommunication Terminal) Equipment Directive:
EN 301 489-1 V1.9.2 :2011
EN 301 489-7 V1.3.1 :2005
EN 301 489-17 V2.2.1 :2012

Angaben zur Person des bevollmächtigten Unterzeichners:
Personal data for the authorized signer:
Wolfgang Wiedemann, Inhaber und Geschäftsführer Sensor-Technik Wiedemann
Wolfgang Wiedemann, owner and managing director of Sensor-Technik Wiedemann

Kaufbeuren, 10.02.2017


Unterschrift / signature

3.4 Warranty

Warranty will be void if:

- The warranty seals of STW have been removed or damaged
- The product was opened by unauthorized persons
- Damage has been caused as a result of use, storage or installation that does not comply with the user manual

3.5 Documents

Overview of documents that are available for the TC3G:

Document title	Publisher	Description
TSP - TC3G - Release Notes	STW	Contains Release Notes of the communication driver PSM
Datasheet - TECHNICAL DATA - TC3G	STW	Contains an overview of the technical data of the ECU.
TC3G User Manual	STW	This document, user manuals with information about the handling and programming of the TC3G.
Getting Started for machines.cloud	STW	A getting started guide to connect your TC3G to machines.cloud and use the TC3G with the Vehicle Data System (VDS) for data logging.

Accessibility and archiving:

Keep this document and allow access for everyone working with this product.

This document contains instructions that must be observed and followed.

3.6 Target Group

This manual describes the handling and characteristics of the TC3G. It is designed to be a comprehensive source of information about the TC3G for system design, engineering and maintenance personnel.

System development, installation and commissioning of the TC3G must only be carried out by trained and experienced personnel who are sufficiently familiar with the used components and with the complete system.

To be able to understand and use the TC3G this personnel must be familiar with:

- The programming language C and shell scripting
- Installation and usage of communication interfaces like CAN bus, RS232, ETH, WIFI, GSM, GPS and BT
- STW offers training courses for commissioning and programming the TC3G. Please see the training catalog of our Academy on the homepage of STW: www.sensor-technik.de (<http://www.sensor-technik.de/>)

3.7 Used Symbols and Formats

The following symbols and formats are used in our manuals to mark important information:



WARNING:

Warning on faults and errors during the application development.



NOTE:

A note provides additional and important information of the system behavior.



RECOMMENDATION:

A recommendation provides recommended actions that can provide an easier live.



REQUIREMENT:

A requirement are actions that must be adhered to. Otherwise safe system operations cannot be maintained.

3.8 Disposal

The disposal of the unit shall be conform to the recycling regulations of the country and regions in which they are disposed.

Dispose the TC3G according to the valid local regulations.

3.9 TC3G_History

Current Document Version: V1.03r0				
Version	Date	Editor	Changes	link to changed data
v1.03r0	22.05.2017	TSC	Added chapter	stw_TrafficMonitor
	18.04.2017	FBE	Added chapter	machines.cloud (see "machines.cloud daemon" on page 129)
	10.02.2017	APP	Added chapter	stw_GetGSMInfo
	17.01.2017	APP	Added chapter	stw_RetentionRule
v1.03r0	01.09.2016	APP	Updated chapter	ygsmd
			Added chapter	GSM
			Adapted GPS D-BUS structure, changed interface	SMS
				stw_CANsnapshot
v1.02r0	21.07.2016	APP DRA FBE	Added chapter	T_DBUS_GPS_Data (see "ygpsd_get_gps_data" on page 243)
			Updated chapter	
			Add OS recommendation	
v1.02r0	21.07.2016	APP DRA FBE	Added chapter	STW Utilities (see "Utilities Tools" on page 348)
			Updated chapter	prepare your OS (see "Download, extract and test setup" on page 288)
			Add OS recommendation	OS recommendaton
v1.01r2	09.06.2016	TF	Added chapter for qualification tests	Qualification Tests (see "Qualification Tests" on page 391)
v1.01r1	29.10.2015	APP	Added chapter	Debugging (Win) (see "Setup the Compiler for Debugging" on page 315)
	01.03.2016	APP	Updated chapter	Debugging (Linux) (see "Setup the Compiler for Debugging" on page 308)
				Overlay Filesystem (see "Overlay Filesystem" on page 79)
v1.01r0	12.02.2015	APP	Added D-BUS functions for ysignal to libtaf	Signal (see "Signal" on page 265)
		APP	Added UserMessage callback to libtaf	dbus_..._callbacks (see "dbus_initialize_request_callbacks" on page 159)
		APP	Updated chapter	Signal Daemon (see "Signal daemon" on page 116)
	09.06.2015	APP	Added chapter	Mail Daemon (see "Mail daemon" on page 122)
		APP	Added Communication Interface	E-Mail (see "E-Mail" on page 346)
	01.10.2015	FBE	Added chapter	
	13.10.2015	APP	Added chapter	

Current Document Version: V1.03r0				
				<p>OpenSourceLicense (see "Open Source Licenses" on page 367)</p> <p>Customer_RootFS (see "Create Own Root File System" on page 286)</p>
v1.00r1	28.10.2014	APP	Updated chapter	Linux Updater
	29.10.2014	APP	Added chapter	Signal Daemon (see " Signal daemon " on page 116)
	24.11.2014	FBE	Added 5 GHz Access Point description	WLAN (see " WLAN " on page 58)
		APP	Added chapter	Overlay Filesystem (see " Overlay Filesystem " on page 79)
v1.00r0	26.08.2014	DRA	Document created	---

4 Getting started with TC3G

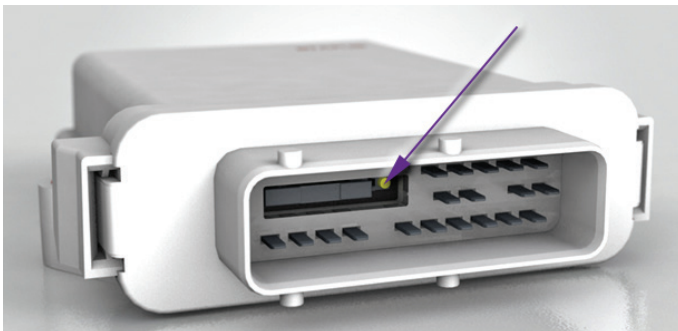
This chapter describes what to do to get started with your TC3G.

4.1 Additional Available Supplies

Additional available supplies for the TC3G that can be ordered from STW:

Position	Description	STW part number
1	Development box (see " Connect to Development Box " on page 23) for TC3G	33518
2	Power supply 5 V ... 15 V (the TC3G needs at least 9 V DC)	53711
3	Accessories kit for mating connector (see " Connector " on page 38)	48083

4.2 Insert SIM Card



To be able to use the mobile feature of the TC3G a SIM card must be installed. The TC3G can handle any kind of common network provider. The SIM card slot is located inside the connector.

How to install a SIM card

1. Press the yellow button beside the card tray.
2. Remove the SIM card holder
3. Place the SIM card into the tray.
4. Slide in the fitted holder carefully.

4.3 Connect TC3G

There are two ways to connect the TC3G:

- Manufacture your own connecting cable: The mating connector for wire harness see Pin Assignment (see "[Pin Assignment](#)" on page 32), or
- Cabling for development purposes: Use the development box, see Development Box (see "[Connect to Development Box](#)" on page 23)

4.4 Connect to Development Box

STW offers a development box for the TC3G, for purchase see Additional Available Supplies (see "[Additional Available Supplies](#)" on page 22).

How to connect the TC3G with your development system:

1. Directly plug in the TC3G in the 29 pin connector on top of the development box. See Pin Assignment (see "[Pin Assignment](#)" on page 32) for further information.
2. Connect the development box to your computer. Use a straight-through cable with two 9 pins DSUB female connectors. Connect one end of the RS232 cable with the RS232-1 connector of your development box. Connect the other end of the cable to the COM1 port of your PC. You can also use a USB to RS232 converter (recommended converter from GIGAWARE).
3. Connect the development box to your network. Use a CAT 5 Ethernet cable. Connect the cable with the RJ45-connector to the back of the development box. Connect the other end of the cable to a network switch or router.
Use a cross over cable when connecting the development box to a hub or with a PC.
4. Connect the power supply to the back of the development box. See Additional Available Supplies (see "[Additional Available Supplies](#)" on page 22) for a suitable power supply.



4.5 Setting up the Serial Interface

Terminal program for the serial interface

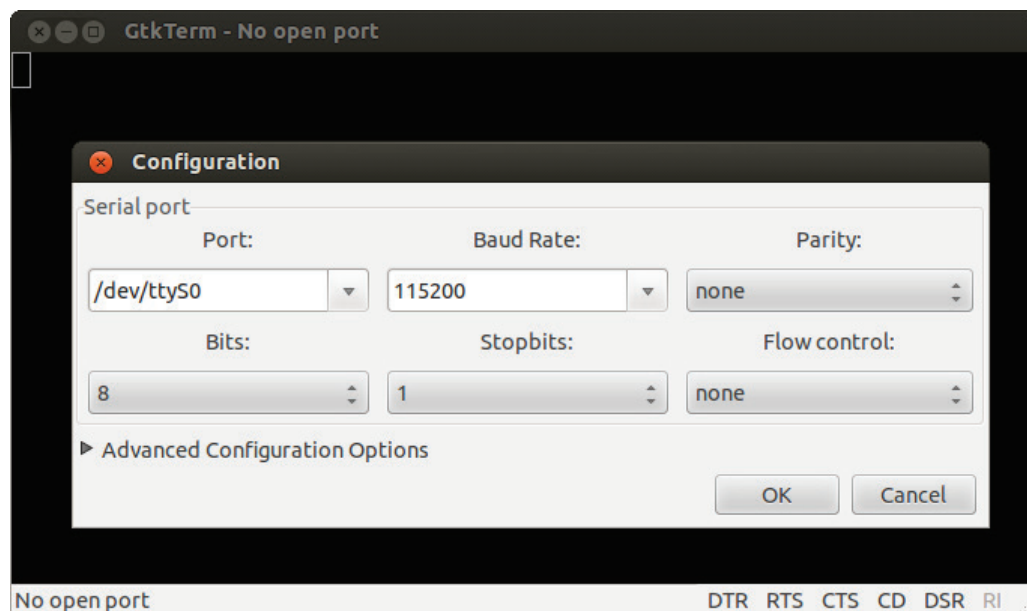
Use a serial straight-through cable to connect the TC3G with your PC.

Depending on the used operating system on the used PC for development, a RS232 terminal must be installed.

Recommended programs for a RS232 terminal:

- GtTerm for computers using a Linux operating system
- Tera Term for Windows desktop PCs

GtTerm used for RS232 terminal program:



Settings for the serial port

Property	Value/Description
Port	state here your port
Baud rate	115200
Data	8bit
Parity	none
Stop	1 bit
Flow control	none

4.6 Power up Device

How to connect and power up the TC3G

- Turn OFF the TC3G (in case it was previously turned on)
- Start your RS232 terminal (in this case: GtTerm)
- Turn power ON to TC3G (Switch on +UB and D+ on the front site of the development box)

- On the RS232 terminal you can see the following U-Boot messages:

```
CPU: MPC5200B v2.2, Core v1.4 at 396 MHz
Bus 132 MHz, IPB 132 MHz, PCI 66 MHz
Board: STW TC3
I2C: 343 kHz, ready
DRAM: 128 MiB
Flash: 64 MiB
NAND: 1024 MiB
In: serial
Out: serial
Err: serial
Net: FEC

Hit 3 * ESC key to stop autoboot...
## Booting kernel from Legacy Image at ff700000 ...
Image Name: Linux 3.4
Created: 2013-04-04 6:49:05 UTC
Image Type: PowerPC Linux Kernel Image (uncompressed)
Data Size: 4899968 Bytes = 4.7 MiB
Load Address: 00000000
Entry Point: 00000000
## Flattened Device Tree blob at fffc0000
Booting using the fdt blob at 0xffffc0000
Loading Kernel Image ...
```

System Login

The default admin user is "root" without any password:

```
start: [ telnetd ]
start: [ ssh ]
start: [ wakeup ]
start: [ shalt ]
start: [ messagebus ]
start: [ gps ]
start: [ ysysd_daemon ]
start: [ bt ]
start: [ wlan_init ]
start: [ hosts ]
start wlan_deamon (adhoc)
start dnsmasq on wlan0
start: [ taf ]
start: [ rc.local ]
start: [ taf ]
start: [ rc.local ]

+++++++ Welcome to ++++++
TC3-??????????? (Var tc3_A_A)
Have a lot of fun...
+++++++
TC3-??????????? login: root
#
```

4.7 How to Secure the System

How to set a password for "root"

- Log in with root:

```
+++++++ Welcome to ++++++
TC3G-141846121005 (Var 66660)
Have a lot of fun...
+++++++
TC3G-141846121005 login: root
#
```

2. Use command "passwd" and type in a password:

```
# passwd
Changing password for root
New password:
Retype password:
Password for root changed by root
#
```

How to set up a firewall with iptables

The TC3G can be secured with iptables. iptables can be used to build internet firewalls.

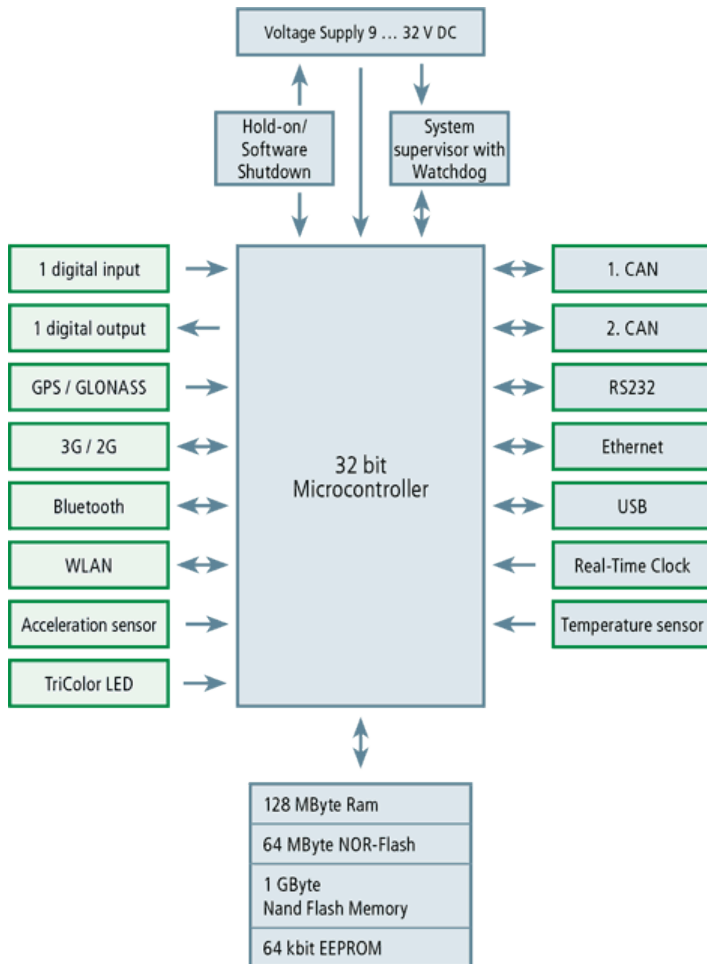
iptables is the userspace command line program used to configure the packet filtering ruleset.

For further information go to: <https://www.netfilter.org/> <https://www.netfilter.org/>

5 Hardware

Here is an overview provided about the technical components of the TC3G.

5.1 Block Diagram



5.2 Technical Data

The technical data provides an overview of all technical characteristics of the components of the TC3G.

5.2.1 Power Supply

Component	Description	Range
DC voltage supply	Voltage at +UB power supply	9 V ... 32 V DC Build in reverse protection; load dumps up to 40V
Current consumption		
- Stand-by	Sum of input currents at +UB ($U_{KL15} = 0$ V, ignition off) - standby supply for GPS-RAM, motion wakeup and RTC	≤ 1 mA
- ECU active	+UB supply current ($U_{KL15} > U_{KL15HIGH}$, no external load, no USB device connected, GPRS transmission, GPS reception and WLAN Adhoc mode)	typ. 350 mA at +UB = 12 V DC typ. 200 mA at +UB = 24 V DC
Ignition pin	Internal pull down resistor	3 kOhm
	switch on threshold	> 2.6 V
	switch off threshold	< 1.5 V
	Protected against short to ground and battery plus	



NOTE:

The current consumption can be higher, depending on the USB devices that are connected to the TC3G and depending on the load that is connected to the digital output.

5.2.2 Processor and System Memory

Type	Features
Processor	32 bit controller, Freescale MPC5200B, 400 MHz
RAM	128 MByte DDR-SDRAM

Type	Features
EEPROM	8 kByte, for user (typical endurance according to manufacturer: 1,000,000 erase/program cycles @ 25°C)
NAND Flash memory	1 GByte for data
NOR Flash memory	64 MByte for program
Watchdog	1 s trigger time, supervising internal supply voltages and for power on reset
Temperature sensor	measuring range -55°C to +150°C
RTC	real time clock with internal gold cap for buffering time and data for a few days (approx. 4 days) system wakeup function (if Ub is connected)
Acceleration sensor	measuring ranges +/-2g or +/-4g or +/-8g or +/-16g (configurable) in 3 axis (X, Y, Z) measuring rate up to 400Hz standby power supply powered for wakeup functionality system wakeup (if Ub is connected) triggered by vibration

5.2.3 Communication Interfaces


Type	Maximal available counts	Description
GPS / GLONASS	1	GLONASS & GPS simultaneously, 1 ... 10 Hz update rate, 33 tracking channels, SBAS (WAAS, EGNOS, MSAS, GAGAN, QZSS)
GSM / GPRS	1	Five-Band 3G - HSPA+, Quad-Band 2G - GPRS/EDGE
CAN	2	CAN 2.0 B, high-speed and low-speed, baud rate from 40 kbit/s to 1 Mbit/s
RS232	1	Serial interface with programmable baud rate up to 115 kbit/s
Ethernet	1	IEEE 802.3, 10/100 Mbit/s, hardware variant with additional connector
Bluetooth	1	Bluetooth V.2.1 + EDR (enhanced data rate) Bluetooth V.4 (BLE bluetooth low energy) Class 1 (+17 dBm) internal or external antenna option (via SMA connector)
WLAN	1	IEEE 802.11 a/b/g/n - 2.4 GHz / 5.5 GHz Infrastructure mode with WEP64, WEP128, WPA, WPA2

Type	Maximal available counts	Description
		(TKIP/PSK) security AdHoc mode (WEP) internal or external antenna option (via SMA connector)
USB Host	1	USB1.1 host port (low / full speed) - service interface
Indicator	1	Buzzer (beeper)

5.2.4 Inputs and Outputs

Type	Maximal available counts	Feature	Range
Digital input	1	Maximum input voltage	36 V DC
		Internal pull down resistor	10 kOhm
		ON-threshold	1.7 V ... 3.3 V
		OFF-threshold	0.7 V ... 2 V
Digital Output	1	Maximum output voltage	8 V ... 32 V DC (min. +UB-1 V ... max. +UB)
		Maximum output current	0.5 A (thermal over current limitation)
		ON-threshold	1.7 V ... 3.3 V
		OFF-threshold	0.7 V ... 2 V
		ON state resistor	max. 0.3 Ohm
		Internal pull down resistor	10 kOhm
		High side switch to +UB	

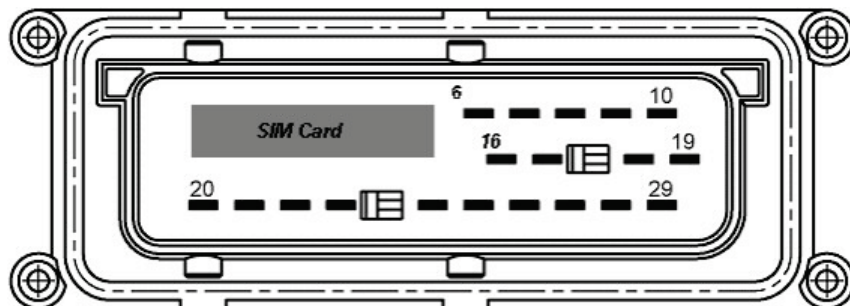
5.2.5 System Data

Component	Description	Value
Operating system	Linux, Board Support package (incl. source code) open source, development environment	--
Connectors	Automotive Typ (Tyco / AMP): for 1x WiFi or 1x 2G/3G and 1x GNSS: For matching connector refer to Connector (see " Connector " on page 38)	19 pin cable suited plugs SMA plugs.
Protection class	when connected	IP6k7
Weight	--	0.3 kg (0.67 lbs)
Dimensions	--	ca. 183 mm x 117 mm x 36 mm (7.21" x 4.61" x 1.42")
Operating temperature	housing temperature	-30°C .. +60°C (-22°F .. +140°F)
Internal Temperature sensor	measurement range	-55°C ... +150°C
Real time clock (RTC)	Gold cap buffered with wakeup function Real time clock with internal gold cap for buffering time and data for approximately 4 days system wakeup function (when +UB is connected)	--
Certificates and approvals	Tests according to standards of the automotive, agricultural and construction machinery industry	
	 conformity	

5.3 Pin Assignment

Matching Connector

For mating connector see Connector (see "[Connector](#)" on page 38).



Pin numbers of the TC3G:

Pin Number	Description
6	USB Ground
7	Digital Output (int. 10kOhm pull down, max. 400[mA] @ 12V)
8	+UB Power supply (9-32VDC)
9	GND (shield USB & WLAN)
10	KL15 / D+ (switched power / ignition switch)
16	USB 5V (power supply USB devices)
17	Digital Input (int. 10kOhm pull down)
18	RS232 RxD (connect to PC - SUB-D Pin3)
19	RS232 TxD (connect to PC - SUB-D Pin2)
20	CAN1 low (termination required)
21	CAN1 high (termination required)
22	CAN2 low (termination required)
23	CAN2 high (termination required)
24	USB D-
25	USB D+
26	Ethernet Rx- (equals a PCs RJ45 - Pin 6)

Pin Number	Description
27	Ethernet Rx+ (equals a PCs RJ45 - Pin 3)
28	Ethernet Tx- (equals a PCs RJ45 - Pin 2)
29	Ethernet Tx+ (equals a PCs RJ45 - Pin 1)
remaining pins	N/A


NOTE:

Cross the Rx/Tx pairs (6 <-> 2 and 3 <-> 1) for a 1:1 ethernet connection.


NOTE:

RS232 needs a common ground for communication (PC-GND <-> TC3G-GND)

5.4 Mounting Guidelines

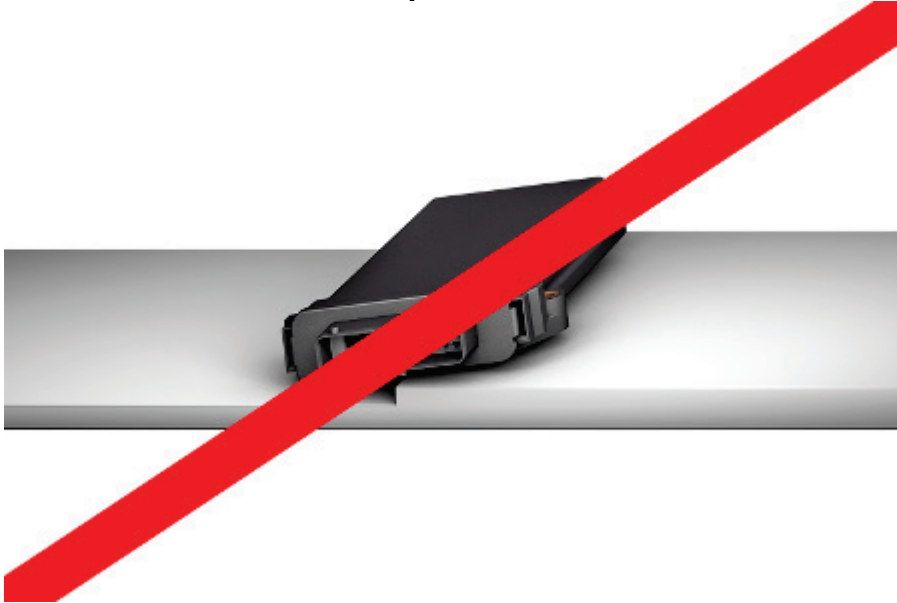
- Make sure that you use the total number of two screws to attach the TC3G to your vehicle.
- Tighten each screw with a preload force of $F = 6.7 \text{ kN} - 9.0 \text{ kN}$. Use the stated preload force to determine the torque:
The torque of a screw depends of its type, its structure robustness, and coefficient of friction (COF). To determine the torque of a screw, use the stated preload force and the data of your used screw.

Sample torque:

Type of screw	Material	Structure robustness	COF	Preload force	Torque [τ]
DIN EN ISO 4762 – M6	Steel	8.8	0.12	6.7 kN - 9.0 kN	7 Nm - 9.5Nm

- Make sure that you operate the TC3G always within the electrical and mechanical specifications defined by STW. If you are not sure about your application, do not hesitate to contact STW's design team.

- Make sure that the surface where you install the TC3G is flat.



- Do not attach any other parts or devices onto the TC3G. Attached parts can lead to an unstable position and prevent ventilation.
- Do not paint the surface of the TC3G. The paint can damage the housing and gasket of the TC3G.
- Provide proper ventilation. The TC3G generates surplus heat during operation.
- Keep the TC3G away from hot air and hot surfaces. Hot air or hot surfaces can heat the TC3G to a not allowed temperature range.
- Make sure to keep a minimum distance of 20 cm between the used antenna and people.
- Do not cover the antenna (WLAN, BLUETOOTH, GPS) with an electromagnetic shield. Covered with a electromagnetic shield, the antenna can not receive any signal.
Do not cover the TC3G with an electromagnetic shield, if you are using a variant of the TC3G with an internal antenna.

- For internal antenna variant of the TC3G, install the TC3G with the labels pointing up. The labels pointing up, is the installation position for the best GPS reception.



- Do not install the TC3G to your device with the connector or the cable harness pointing upwards. If the connector or the cable harness points upwards, condensation water can not drain off.



- For external antenna variant of the TC3G, use the following allowed and recommended installation positions.



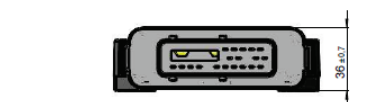
5.5 Connecting Guidelines

- Use a dummy plug for not used pins of your used connector. The rated IP Code for the TC3G is only fulfilled, if the TC3G is connected to a fully assembled connector. Consider the data and specifications required for IP rating from the connector's manufacturer you are using.
- Use STW's recommended type of connectors (see "[Connector](#)" on page 38).
- Attach the cable harness of the TC3G <100mm (3.9") from the connector with a strain relief.
- Attach the strain relief of the cable harness at the same surface, where the TC3G is attached. If the cable harness and the TC3G are attached to the same surface, the vibration level of both is the same. Different vibration levels can cause damage to the connector and to the cable.
- Requirement in accordance to the EC type-approval of the Kraftfahrt-Bundesamt (KBA) - Federal Motor Transport Authority:
All vehicle types with a 12 V respectively 24 V - electrical wiring and battery(-) at the body.

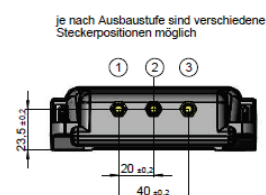
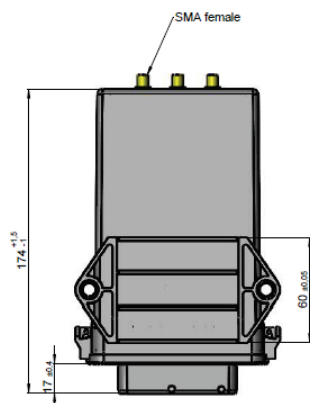
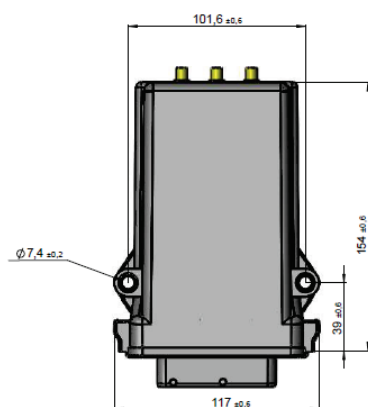
5.6 Housing

External Antenna Version

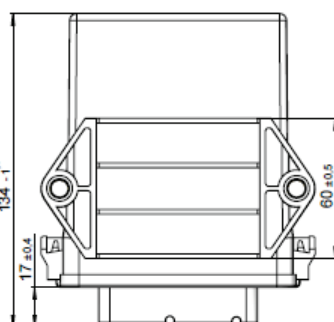
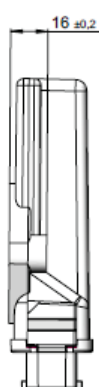
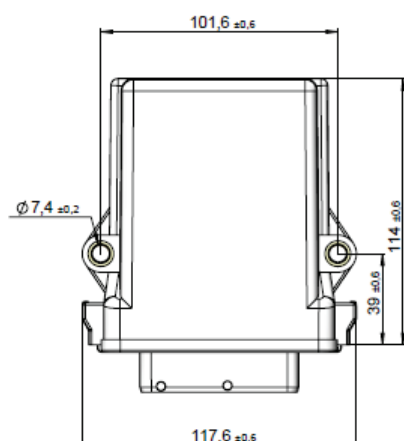
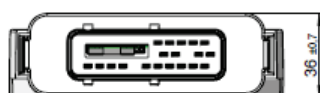




29 Pin Connector Kit as example	
Description	AMP- Part No.
1x 29 pin connector housing (main part of housing)	963443-1
1x housing cover/ cap	965643-1
19x crimp contact for JPT 929 940-3	929940-1
19x seals for large contact pins	828904-1
10x blanking plug	828922-1



Internal Antenna Version

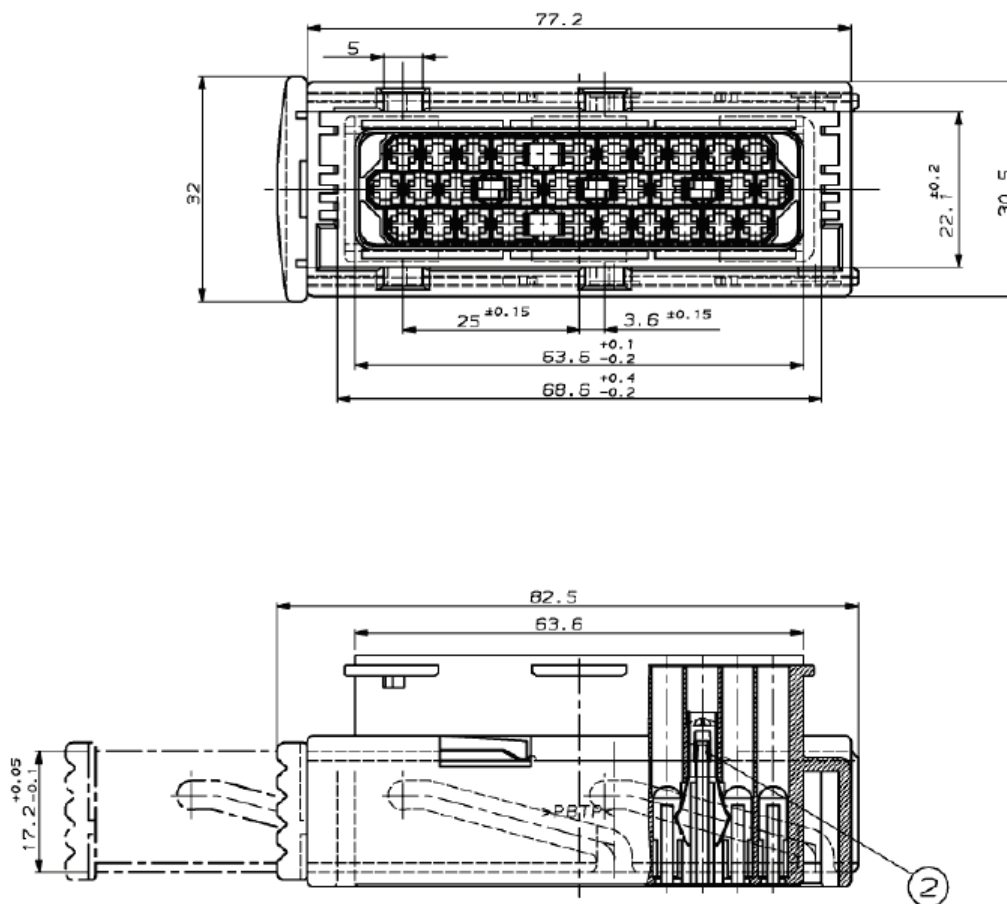


5.7 Connector

Mating connector

The mating connector can be ordered from STW article no: 48083

- Manufacturer: TYCO
- Type: 3 rows female, 29 pin (19 pins used, 10 blank)
- Manufacturer No.: 1-963449-2
- Contact material: JPT 0.5-1.0 mm² CuFe



5.8 Antenna

Recommended antenna

Manufacturer	Name/Type	Communication interfaces	Connector type
Hirschmann	Roof Mount Antenna CGW 70 26 59 SF S	WLAN, BLUETOOTH CELLULAR (GSM) GPS	SMA male

Manufacturer	Name/Type	Communication interfaces	Connector type
Hirschmann	Low profile antenna GPS 18 90 LP / P / SMA / SMA / 3.0	CELLULAR (GSM) GPS	SMA male


REQUIREMENT:

Make sure to keep a minimum distance of 20 cm between the used antenna and people.

Connector Type

Use SMA male connectors on the antenna side.



SMA Male

6 Software

Motivation

The "Software" chapter includes a detailed descriptions of the software interfaces that are provided with the TC3G system. The operating system of the TC3G is a embedded GNU/Linux, which is is configured by STW.

To check for the latest version of the operating system for the TC3G go to the STW FTP server: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>)

The operating system is separated in two spaces:

- kernel space
- user space.

The user space is created by the root file system. The user space provides interface files that are generated by the kernel. These files are not all human readable and are more system helpers (applications and scripts).

Thus, not all user space interfaces are inevitably described by the "Software" chapter. It was decided to categorize the different user space interfaces into levels.

The following levels are defined:

- Teleservice Application Framework: This level includes the TAF Library, which is written in C and can be used by the user to create teleservice applications, and the TAF Components. This includes the teleservice daemons and their configuration files.
- High Level Hardware Access: This level includes scripted files (scripts) and tools (system helpers) which provide more complex functionalities. It uses the Low Level Interfaces to process functionalities, e.g. mount a device.
- Low Level Hardware Access: This level includes files (interfaces) which are created by the specific drivers from the kernel. These files provide information about the driver status, read / write values to the driver, etc.

6.1 System Information

System Information describes how to get the version information about the components of the board support package (BSP) of the TC3G.

The BSP consists of the components:

- Root file system
- Linux kernel
- Device tree block
- U-Boot

How to get the system information:

System component	Path with input information	Output information	Example	Description
Root file system	cat /etc/br-version	<STW-V<X.XXrX> / <buildroot version> / <br_base-svn revision...> / <br_target-svn	STW-V1.01r0 / br-2011.02 / ... / ... /	This user space interface includes: <ul style="list-style-type: none"> • The version of the

System component	Path with input information	Output information	Example	Description
		revision...>		<p>STW root file system</p> <ul style="list-style-type: none"> The buildroot version of the system. The buildroot base revision The buildroot target revision
Linux kernel	/bin/uname -r	<OS version> / <STW kernel version>	3.10.94-rt102/STW-v1_03r0+	Provides information about the used operating system and the used Linux kernel of the TC3G.
Device tree block	cat /proc/device-tree/dts_version	<STW-V<X.XXrX>	STW-V0.01r0	Version of the device tree block for the TC3G.
U-Boot	fw_printenv ver	ver=u-boot XXXX.XX-svnYYYY (MMM DD yyyy - hh:mm:ss) ecu_b STW-VX.XXrX	ver=u-boot 2012.10-svn9064 (Jun 01 2016 - 10:58:25) tc3_b STW-V2.05r1	Version of the U-Boot used in the TC3G.

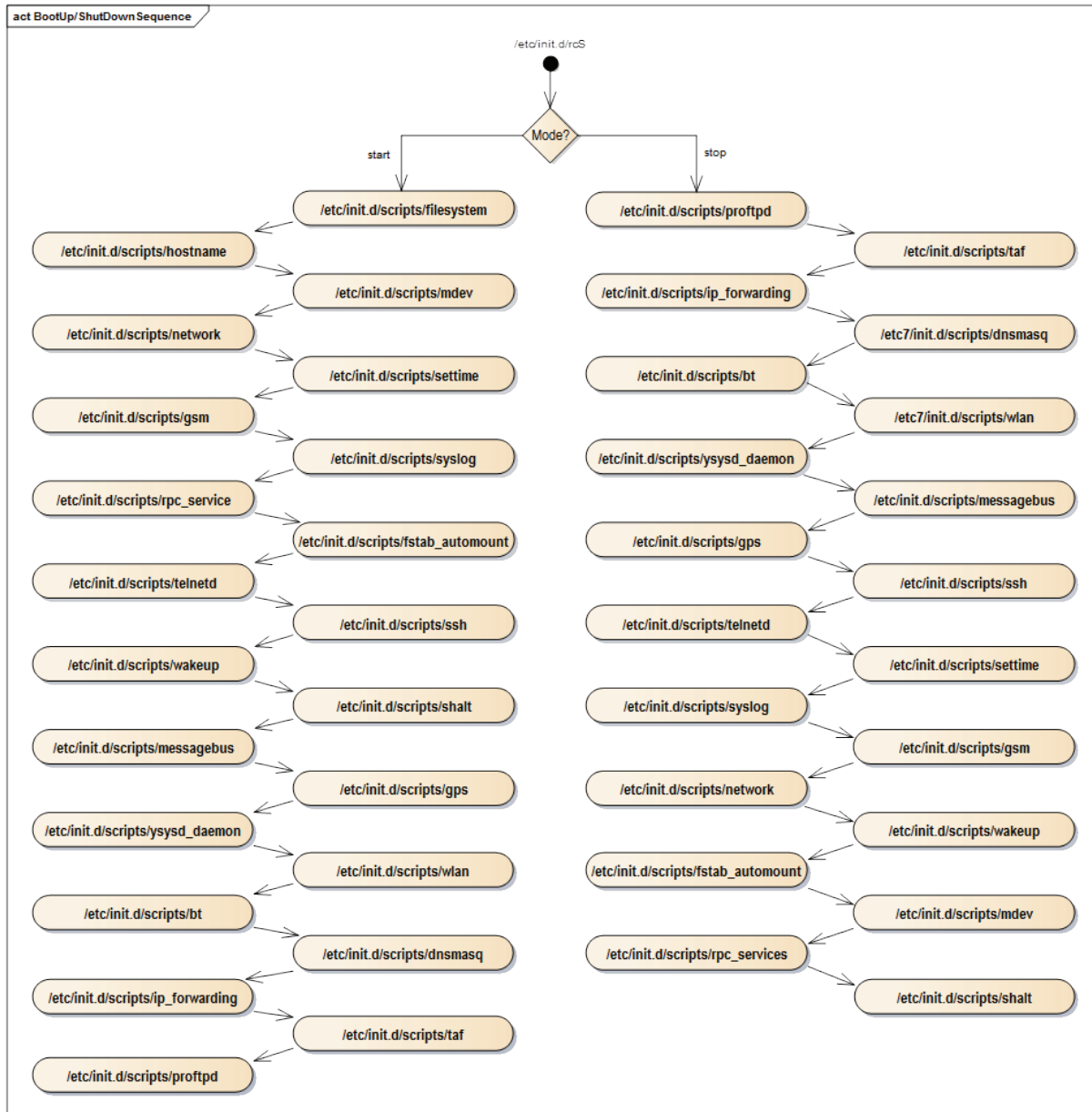
Example of how to get system information:

```
# cat /etc/br-version
STW-V2.00r0 / br-2011.02 / br_base-svn9234-http://stwfue/svn/y/linux/mpc5200/rootfs/buildroot\_2011.02/buildroot/trunk/pjt / br_target-svn9342-http://stwfue/svn/y/linux/mpc5200/rootfs/buildroot\_2011.02/targets/tc3g/trunk
#
# /bin/uname -r
3.10.94-rt102/STW-v1_03r0+
#
# cat /proc/device-tree/dts_version
STW-V0.01r0#
#
# fw_printenv ver
ver=u-boot 2012.10-svn9064 (Jun 01 2016 - 10:58:25) tc3_b STW-V2.05r1
#
```

6.2 Boot up / Shut down sequence

The following files are used on the TC3G device for boot up, shut down and basic initialization of the system.

Files executed during boot up and shut down sequence:



/etc/inittab

The inittab file controls which processes are started/executed during boot up and during normal operation.

/etc/init.d/rcS

A collection of initializations are performed when executing the script /etc/init.d/rcS as specified in the initialization table /etc/inittab. The specified initializations are performed through scripts that are located in the folder /etc/rc.d/. First /etc/init.d/rcS sets the job control functions of the shell and loads the latest configuration information from /etc/rc.d/rc.conf. If rcS was called with the parameter "start", the variable "services" will be set with "cfg_services" from rc.conf. Now the script has all scripts which must be called for startup. Otherwise the list of script names for "shutdown" will be set.

/etc/init.d/rc.conf

rc.conf is a setup script. It returns all script names to rcS script that must be called on startup or on shutdown. In general the rc.conf is the configuration file for initscripts. It configures what daemons to start at boot up, the basic network daemon, and certain aspects of hardware discovery. For this target the rc.conf exports some global settings which configures the behavior of the device interfaces. It only supports the configuration of single interfaces. All this settings can be changed by user.

Example: rc.conf contains the start up and shut down sequence of the system

```
# For startup following scripts will be called:
cfg_services="filesystem hostname mdev network settime gsm syslog rpc_services
fstab_automount telnetd ssh wakeup shalt messagebus gps ysysd_daemon wlan bt dnsmasq
ip_forwarding taf proftpd"

# For shutdown following scrips will be called:
cfg_services_r="proftpd taf ip_forwarding dnsmasq bt wlan ysysd_daemon messagebus gps ssh
telnetd settime syslog gsm network wakeup fstab_automount mdev rpc_services shalt"
```

/etc/init.d/rc.local

The script is responsible for performing all user defined actions. Use this script to execute programs and scripts automatically at the end of the system boot up process and at the beginning of the shut down process. If the parameter "start" is set, then the state is checked how the device is woken up.

In case of wakeup by ignition (IGN): All scripts and commands for the case IGN are called.

In case of wakeup by real time clock (RTC): All scripts and commands for the wakeup by RTC are called.

In case of wakeup by Motion Sensor (MOT): All scripts and commands for the wakeup by MOT are called.

Or, if nothing else fits, all scripts and commands for the wakeup by the "unknown wakeup" reason are called.

In case of "stop", the same procedure as on "start" will be used. First the wakeup reason will be checked. If one of these reasons fits, all proper scripts and commands will be called and executed.

6.3 Wakeup

For the TC3G are the following sources for wakeup possible:

- Wakeup by ignition
- Wakeup by RTC alarm
- Wakeup by MOT (motion was detected)

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/shalt	p1	-	<p>This is a shell script which switches the self-preservation (/proc/stw_shutdown/shalt) On/Off.</p> <p>p1: defines what action should be done</p> <p>< start > or < on ></p> <ul style="list-style-type: none"> • sets the self-preservation on <p>< stop > or < off ></p> <ul style="list-style-type: none"> • sets the self-preservation off
/etc/init.d/scripts/ysysd_daemon	p1	-	<p>This is a shell script which executes the system daemon (/usr/local/bin/ysysd).</p> <p>p1: defines what action should be done</p> <p>< start ></p> <ul style="list-style-type: none"> • executes /usr/local/bin/ysysd <p>< stop ></p> <ul style="list-style-type: none"> • kills ysysd
/etc/init.d/scripts/wakeup	p1	-	<p>This is a shell script which handles the wakeup reasons at start up and on shut down.</p> <p>Wakeup reasons:</p> <ul style="list-style-type: none"> • IGN .. System was started by ignition • RTC .. System was started by RTC alarm • MOT .. System was started by motion (from motion sensor) <p>The wakeup reason can be read out from /tmp/wakeup.</p> <p>p1: defines what action should be done</p> <p>< start ></p> <ul style="list-style-type: none"> • reads the wakeup reason and writes it to /tmp/wakeup • activates /proc/stw_shutdown/shalt • disables RTC-Alarm

Path	Input information	Output information	Description
			<ul style="list-style-type: none"> disables motion sensor wakeup <p>< stop ></p> <ul style="list-style-type: none"> Sets of the wakeup sources before the system is shut down

How to set parameters for wakeup:

The parameters are set over the interface rc.conf:

Example for set wakeup parameters

```
# Set wakeup parameter to etc/init.d/rc.conf

# Activate motion wakeup with "yes", else deactivate
export MOTION_WAKEUP="yes"
# Enable date + time wakeup with "tomorrow_at" or
# Enable wakeup after seconds with "after_sec", else disable
export RTC_WAKEUP="after_sec"
# Time in seconds if RTC_WAKEUP="after_sec"
# In combination with "after_sec" RTC_WAKEUP_TIME holds the value in seconds.
# In combination with "tomorrow_at" RTC_WAKEUP_TIME hold the value in hhhmmss.
export RTC_WAKEUP_TIME=10

# Run time in seconds after shutdown (ignition off).
# Set the default value here or wakeup-reason depending on:
# Ignition wakeup time in second, default is 0.
export SD_RUN_TIME_IGN=0
# Motion wakeup time in seconds, default is 300s.
export SD_RUN_TIME_MOT=300
# RTC wakeup time in seconds, default is 20s.
export SD_RUN_TIME_RTC=20
# Shutdown run time path includes the time in seconds.
export SD_RUN_TIME_FILE="/tmp/sdruntime"
```

6.3.1 Ignition

The state of the ignition can be retrieved over the function of the system daemon.

Precondition:

The system daemon (see "[System daemon](#)" on page 95) must be already started. Per default the system daemon is already started.

Library function for ignition

Function	Brief description
ysysd_get_ignition_status (see " ysysd_get_ignition_status " on page 169)	The TAF library function returns the ignition status.

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/stw_shutdown/shalt	p1	-	<p>This user space interface allows the user to switch On or switch Off the self-preservation.</p> <p>p1:</p> <p>< 0 ></p> <ul style="list-style-type: none"> the self-preservation is switched Off <p>< 1 ></p> <ul style="list-style-type: none"> the self-preservation is switched On
/proc/stw_shutdown/dplus	-	p1	<p>This user space interface includes the state of the ignition pin.</p> <p>p1:</p> <p>< 0 ></p> <ul style="list-style-type: none"> the state of the ignition is Off <p>< 1 ></p> <ul style="list-style-type: none"> the state of the ignition is On

6.3.2 Real Time Clock

The hardware Real Time Clock (RTC) is used to synchronize the kernel clock during the boot up and the shut down sequence. During boot up the kernel clock is synchronized to the RTC. During shut down the RTC is synchronized to the kernel clock.



NOTE:

The RTC is not used by the kernel during runtime.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/settime	p1	-	<p>This shell script starts the clock synchronization over SNTP.</p> <p>settime is active as soon as it is written to rc.conf.</p> <p>p1:</p> <p>< start ></p> <ul style="list-style-type: none"> starts clock synchronization over SNTP

Path	Input information	Output information	Description
			<p>service</p> <p>< stop ></p> <ul style="list-style-type: none"> • syncs the RTC to kernel clock


NOTE:

If the SNTP client is not activated and the settime script is not fully implemented, then the user has to configure the SNTP client and may have to changed the settime script if necessary.

Interface at rc.conf:
Activate / Deactivate SNTP

```
# Activate "YES" / deactivate "NO" export TRY_Sntp in /etc/init.d/rc.conf
export TRY_Sntp="YES"
```

Example: How to set system time:

Set the system time by using the path /bin/date that is provided by the BusyBox multi-call binary. In this example the system time is set to Mon Nov 5 10:53:00 UTC 2012.

```
# date -s 201211051053.00
Mon Nov 5 10:53:00 UTC 2012
```

Low Level Hardware Access

Path	Input information	Output information	Description
/sys/class/rtc/rtc0/date	-	p1	<p>This user space interface gets the current date from the kernel clock.</p> <p>p1:</p> <p><YYYY-MM-DD></p> <ul style="list-style-type: none"> • returns the date formatted like the given syntax
/sys/class/rtc/rtc0/time	-	p1	<p>This user space interface gets the current time from the kernel clock.</p> <p>p1:</p> <p><hh:mm:ss></p> <ul style="list-style-type: none"> • returns the time formatted like the given syntax

Path	Input information	Output information	Description
/sys/class/rtc/rtc0/wakealarm	p1	-	<p>This user space interface can be used for setting the hardware RTC wakeup alarm. After a shut down of the RTC, the wakeup alarm will be activated.</p> <p>p1:</p> <p>< 0 ></p> <ul style="list-style-type: none"> sets off hardware RTC wakeup alarm <p>< 1..n ></p> <ul style="list-style-type: none"> sets the countdown in seconds for startup

Example: Handle wakeup alarm

Sets the wakeup alarm off:

```
# echo 0 > /sys/class/rtc/rtc0/wakealarm
```

Sets the wakeup alarm to 10 seconds:

```
# echo 10 > /sys/class/rtc/rtc0/wakealarm
```

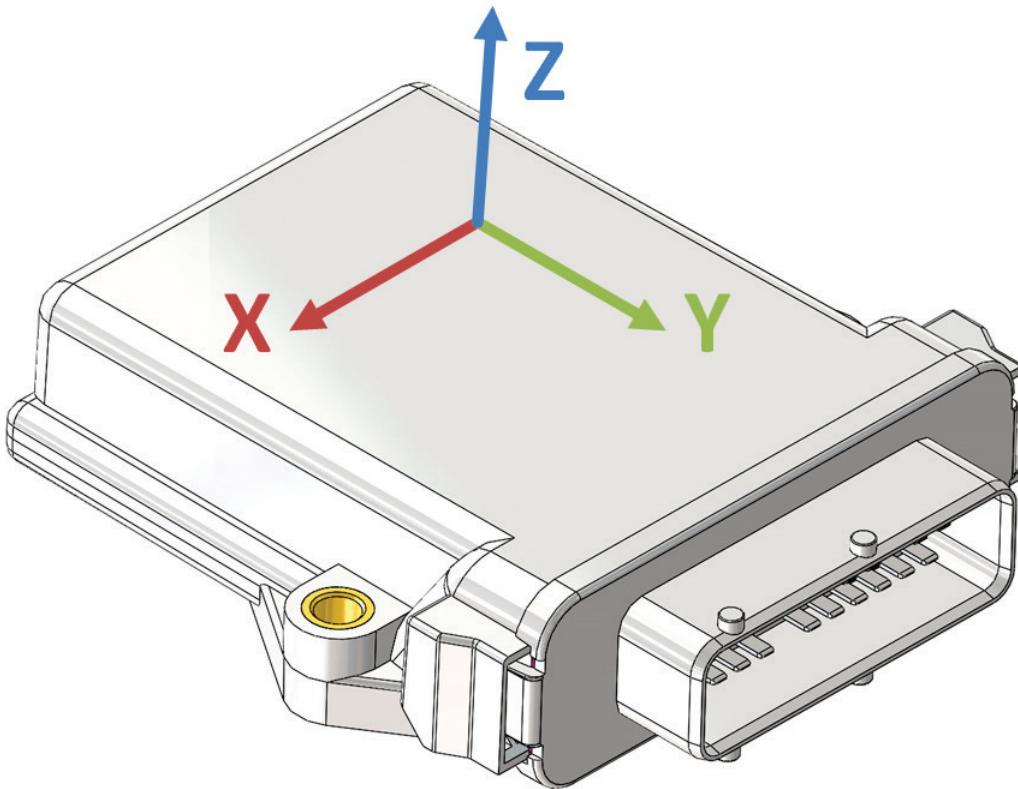
6.3.3 Motion Sensor

Low Level Hardware Access

The motion sensor (LIS3DH) has a scale of: $\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$ (dynamically selectable fullscale)

Path	Input information	Output information	Description
/sys/bus/i2c/devices/0-0018/x	-	Acceleration of the X-Axis in [mg]	This user space interface gives the acceleration of the X-Axis in mg (1000mg ~ 9,81 m/s ²)
/sys/bus/i2c/devices/0-0018/y	-	Acceleration of the Y-Axis in [mg]	This user space interface gives the acceleration of the Y-Axis in mg (1000mg ~ 9,81 m/s ²)
/sys/bus/i2c/devices/0-0018/z	-	Acceleration of the Z-Axis in [mg]	This user space interface gives the acceleration of the Z-Axis in mg (1000mg ~ 9,81 m/s ²)
/sys/bus/i2c/devices/0-0018/xyz	-	Acceleration of the XYZ-Axis in [mg]	This user space interface gives the acceleration of the XYZ-Axis in mg (1000mg ~ 9,81 m/s ²)

TC3G with 3-Axis of motion sensor



Example: How to access value of the y-Axis

Read out a single axis.

```
# cat /sys/bus/i2c/devices/0-0018/y
-1004
```

Calculation of the result

$g = -1004mg \sim 1g$

Example: Change the scale of the motion sensor

Change scale from 2g to 8g

```
# echo scale=8g > /sys/bus/i2c/devices/0-0018/control
```

Verify change

```
# cat /sys/bus/i2c/devices/0-0018/control
power=on                [on/off]
powermode=normal        [normal/low_power]
fifomode=off            [off/fifo/stream/bypass]
datarate=100Hz          [1Hz/10Hz/25Hz/50Hz/100Hz/200Hz/400Hz/1.25kHz/1.6kHz/5kHz]
scale=8g                [2g/4g/8g/16g]
setting=default         [wakeup/default]
wakeup_threshold=05     (default=10 [05 .. 101])
```

6.4 Network handling

For network handling use the network daemon that handles network interfaces automatically.

Precondition:

The network daemon must be started. The network daemon is not started during the boot up automatically.

How to activate the network daemon see network daemon (see "[Network daemon](#)" on page 107).

Library

Function	Brief
ynetworkd_get_connected_interface (see " ynetworkd_get_connected_interface " on page 251)	The TAF library function returns the connection status.
ynetworkd_start_connection (see " ynetworkd_start_connection " on page 253)	The TAF library function turns the network daemon into connectivity mode.
ynetworkd_stop_connection (see " ynetworkd_stop_connection " on page 254)	The TAF library function disconnects the current network connection.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/network	p1	<p>Interfaces file</p> <p>/etc/network/interfaces includes the following network options:</p> <ul style="list-style-type: none"> • dhcp .. get IP via DHCP • lan_static .. static IP from etc/init.d/rc.conf • lan_static_networkd .. static IP used only by the ynetworkd • wlan_static .. static IP from etc/init.d/rc.conf 	<p>This shell script creates the temporary file /tmp/interfaces and initializes the ethernet (eth0) and the CAN (can0/1) depending on the network options from /etc/init.d/rc.conf. It is also started by the boot up sequence (see "Boot up / Shut down sequence" on page 42).</p> <p>p1:</p> <p>< start > / < restart ></p> <ul style="list-style-type: none"> • creates the interface file and initializes eth0 and can0/1 depending on /etc/init.d/rc.conf

6.4.1 GSM

The default setup of the GSM modem provides functionalities to handle more than one GSM service at a time.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/gsm	p1	-	<p>This shell script starts or stops the GSM modem and loads APN settings automatically from database /etc/apns-conf-xml depending of the /etc/init.d/rc.conf settings. Make sure before start that a SIM-Card is inserted. Script is started automatically by the boot up sequence (see "Boot up / Shut down sequence" on page 42).</p> <p>p1: defines what action should be done</p> <p>< start ></p> <ul style="list-style-type: none"> switches the modem on, set the SIM-PIN if necessary, create MUX devices and start APN detection <p>< stop ></p> <ul style="list-style-type: none"> switches the modem off <p>< restart ></p> <ul style="list-style-type: none"> calls stop calls start

Interface at rc.conf: Set GSM parameters

Set the GSM parameters to etc/init.d/rc.conf:

```
# GSM baudrate
# Supported bit rates see table below.
export GSM_BAUDRATE="460800"
# PIN for the SIM card
export GSM_PIN=""
```

Example: Use picocom for GSM modem communication (AT commands)

The picocom binary was designed to serve as a simple, manual, modem configuration, testing and debugging tool.

```
# picocom -b 460800 /dev/mux0
```

The key combination to execute picocom is CTRL+A+Q.

Low Level Hardware Access

The modem interfaces are accessible over the symbolic link files /dev/mux0 and /dev/mux1:

Path	Input information	Output information	Description
/dev/mux0	-	-	<p>This device is created per default.</p> <p>Send/Receive AT-commands to GSM modem e.g. via</p>

Path	Input information	Output information	Description
			picocom
/dev/mux1	-	-	This device is created per default. Send/Receive AT-commands to GSM modem e.g. via picocom


WARNING:

If one of these interfaces mux0/1 are used in data mode (GPRS connection is established), no AT commands can be sent to the modem by using tools like picocom.

Path	Input information	Output information	Description
/proc/device-tree/stw_gsm@00/device	-	Name of the modem e.g. PHS8	This user space interface includes the name of the modem.
/proc/device-tree/stw_gsm@00/name	-	Interface name e.g. stw_gsm	This user space interface includes the name of the modem interface.
/proc/device-tree/stw_gsm@00/port	-	Path to real interface e.g. /dev/ttyACM0	This user space interface includes the real path to the modem serial interface.
/proc/stw_gsm/gsm_power	p1	-	This user space interface could be used in a sequence depending on modem type to switch the GSM modem on/off. p1: < 0 > <ul style="list-style-type: none"> triggers pin to low level < 1 > <ul style="list-style-type: none"> triggers pin to high level
/proc/stw_gsm/gsm_port	-	-	Symlink for tty-device (/dev/ttyACM0) the modem is connected to. Send/Receive AT-Commands depending on GSM modem type.

6.4.1.1 GPRS / Peer-To-Peer

The easiest way to establish an internet connection via the GPRS interface is to activate and configure the network daemon.

Network daemon configuration file (see "[Network daemon](#)" on page 107)

High Level Hardware Access

This chapter describes the user space interfaces for the Peer-To-Peer connection that can be established over the GPRS modem.

Linux provides the Point-to-Point Protocol daemon (pppd) to manage internet links that can be established over dial-up modems, DSL connections, and other types of point-to-point links.

Automatically detection of the correct provider settings.

After starting the modem, the TC3G tries to detect the correct provider settings automatically. In order to do this, the mobile country code (MCC) and the mobile network code (MNC) are requested from the modem. The combination of both results in a unique number which identifies your mobile provider.

In the next step, the daemon searches for the received provider identification number in the `/etc/apns-conf.xml` file. This file contains a list of well known mobile providers around the world. If the corresponding provider is found in the list, then the configuration data are written to `/tmp/apn.setting`. The modem is ready to use now.

Since network providers change their access point configuration from time to time, the list might not always have the latest configuration information. In that case, the generated `/tmp/apn.setting` file provides an incorrect configuration too and the ppp connection fails.

There are two options to handle that issue:

Set GPRS parameters manually in the file `/etc/init.d/rc.conf`

If you never want to change your mobile provider, than set the necessary provider informations. This mechanism disables the automatically detection of the provider settings.

Disadvantage: After changing the SIM card, for example you use a different provider, the settings from the file `/etc/init.d/rc.conf` do not fit and the ppp connection will fail.

```
# Access Point Name from provider
export GSM_APN="ProviderAPN"
# User name from provider
export GSM_User="ProviderUser"
# Password from provider
export GSM_Password="ProviderPass"
# Service Center Address, i.e. your SIM card number from provider
export GSM_SCA="00491710760000"
# DNS you want to connect to from provider, if needed.
export GSM_DNS=""
```

Create your own apns-conf.xml file in the NAND dataflash

The much more convenient way is to create your own `apns-conf.xml` file and store it in the NAND dataflash. If the NAND dataflash is mounted (typically behavior of the TC3G), then the daemon checks if the `/mnt/dataflash/stw/modem` directory structure is available.

Place your own `"apns-conf.xml"` file into the `/mnt/dataflash/stw/modem` directory. Also place a `"apn-default.xml"` file into this directory.

Search mechanism of the daemon:

- The daemon searches the received provider identification number in the customer xml file `"/mnt/dataflash/stw/modem/apns-conf.xml"`.
- If the correct ID is not found, then the daemon parses the `"etc/apns-conf.xml"` file. If there is also no entry found that contains to the received ID, the daemon looks for the `"mnt/dataflash/stw/modem/apn-default.xml"` file.

If the apn-default.xml file is available, the daemon uses the the first entry of the file whether this provider identification number fits or not.

The apn-default.xml file has nearly the same behavior as the manually set GPRS parameters mechanism of the daemon. But here is the benefit that the automatically detection will be used first.

Typically user defined apns.conf.xml file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<apns version="7">
  <apn carrier="T-Mobile Internet"
    mcc="262"
    mnc="01"
    apn="internet.telekom"
    user="tm"
    password="tm"
    type="default,supl"
  />
</apns>
```

Path	Input information	Output information	Description
/etc/ppp/ppp-start	-	<p>The following configuration files will be created:</p> <ul style="list-style-type: none"> /etc/ppp/pap-secrets .. PAP secrets for pppd /etc/ppp/gprs-options_gsm .. option file for pppd 	This shell script starts the /usr/sbin/pppd (PPP connection) to establish a GPRS connection. It uses /dev/mux0 and if the /etc/init.d/rc.conf GSM settings are NOT activated then the automatically created APN settings will be used.
/etc/ppp/ppp-stop	-	-	This shell script kills the /usr/sbin/pppd to disconnect the GPRS connection.

6.4.1.2 SMS

The easiest way to handle the Short Message Service, is to activate and configure the SMS daemon.

SMS daemon configuration file

Precondition:

To apply the changes, the system needs to be rebooted.

Library

Function	Brief
ysmsd_send_sms	The TAF library function sends a SMS.
ysmsd_send_sms_urgent	The TAF library function sends a urgent SMS.
ysmsd_request_sms_fetch_urgent	The TAF library function sends a request SMS fetch urgent message.

How to send a SMS using AT commands:

1. Connect to the modem via picocom

```
# picocom -b 115200 /dev/mux1
picocom v1.6

port is      : /dev/mux1
flowcontrol  : none
baudrate is  : 115200
parity is    : none
databits are : 8
escape is    : C-a
local echo is : no
noinit is    : no
noreset is   : no
nolock is    : no
send_cmd is  : sz -vv
receive_cmd is : rz -vv
imap is      :
omap is      :
emap is      : crcrlf,delbs,

Terminal ready
```

2. Put the modem in SMS mode

```
AT+CMGF=1          <ENTER>
OK
```

3. Send the message

```
AT+CMGS="+49160XXXXXXX"  <ENTER>
> This is a sample Text.  <ENTER>
> It was send from the TC3G. <CTRL-Z>

+CMGS: 55

OK
```

4. Disconnect from the modem

```
<CTRL-A-Q>
Thanks for using picocom
#
```

6.4.2 Ethernet

The easiest way to establish a internet connection via the ethernet interface is to activate and configure the network daemon.

Network daemon configuration file (see "[Network daemon](#)" on page 107)

High Level Hardware Access

Get local information

It is possible to read out the local settings of the eth0 device, such as MAC address, IP address, Gateway, Netmask,....

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1D:48:20:F6:11
          inet addr:172.25.230.17  Bcast:172.25.230.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3819 errors:0 dropped:0 overruns:0 frame:0
          TX packets:295 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:507786 (495.8 KiB)  TX bytes:23816 (23.2 KiB)
          Interrupt:133 Base address:0x3000
```

How to set static IP for eth0

1. Set the IP settings to "STATIC" in /etc/init.d/rc.conf

```
# ----- NETWORK ETH0:

# DHCP settings for eth0
# if ETH0_CONF is set to "DHCP" then DHCP will be used
# in case we do not get any DHCP configuration from a DHCP server, we will automatically
# fall back to mode "STATIC".
# if ETH0_CONF is set to "STATIC" then the static settings below will be used
# if ETH0_CONF is set to "UBOOT" then the settings will not be touched
# Note: also adapt the setting in /sbin/dhclient for the gateway entries
export ETH0_CONF="STATIC"

# Static settings for eth0 (if ETH0_CONF="STATIC")
# Note: ETH0_GW is also important if you use ETH0_CONF="UBOOT"
# IP of eth0 if ETH0_CONF="STATIC"
export ETH0_IP="192.168.200.1"
# Submask of eth0 if ETH0_CONF="STATIC"
export ETH0_SUBMASK="255.255.255.0"
# Broadcast IP of eth0 if ETH0_CONF="STATIC"
export ETH0_BR="192.168.200.255"
# Settings for eth0 (if ETH0_CONF="STATIC" or "UBOOT")
# Gateway of eth0
export ETH0_GW="192.168.200.1"
# Domain of eth0
export ETH0_DOMAIN=""
# DNS which eth0 uses for communication
export ETH0_DNS="192.168.1.8"
```

2. Reboot the system:

```
# reboot
```

Example: C-Code

```
# How to TCP connection via sockets
```

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

sint16 s16_Return;
sint16 s16_Sockfd = socket(AF_INET, SOCK_STREAM, 0); // Create socket

struct sockaddr_in t_Srv;

memset(&t_Srv, 0, sizeof(struct sockaddr_in));

t_Srv.sin_family = AF_INET;
inet_pton(AF_INET, "1.2.3.4", &t_Srv.sin_addr); // Set server IP to struct sockaddr_in
t_Srv.sin_port = htons(1234); // Set port in Network-Byte-Order (Big Endian) to sin_port

s16_Return = connect(s16_Sockfd, (struct sockaddr*)&t_Srv, sizeof(struct sockaddr_in)); //
Try to connect
if(s16_Return == 0)
{
// TCP connection is established. Now write() and read() could be used to to write or read
sockets

[...] // Data handling

shutdown(s16_Sockfd, SHUT_WR); // Send a EOF-Byte to the server, so that on the next read()
we will get a 0-return and we could close the connection

close(s16_Sockfd);
```

6.4.3 WLAN

The easiest way to establish a internet connection via the WLAN interface is to activate and configure the network daemon.

Network daemon configuration file (see "[Network daemon](#)" on page 107)

The TC3G WLAN module allows two modes:

- Managed-Mode
- Access-Point-Mode (AP mode)

The managed mode can be started by wpa_supplicant and wpa_cli, the access-point mode can be started by hostapd.

Which one is started depends on the selected mode within the /etc/init.d/rc.conf configuration file.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/wlan	p1		<p>Only available when a hardware WLAN module exists on the used TC3G.</p> <p>This shell script loads the WLAN module in the kernel during system boot up.</p> <p>p1:</p> <p>< start ></p> <ul style="list-style-type: none"> • checks if MAC address is set • loads the specific WLAN modules to the kernel • starts AP or managed mode <p>< stop ></p> <ul style="list-style-type: none"> • stops started daemons • unloads specific WLAN modules from kernel <p>< restart ></p> <ul style="list-style-type: none"> • calls stop • calls start

Get local information

It is possible to read out the local settings of the wlan0 device, such as MAC address, IP address, Gateway, Netmask,....

```
# ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr 00:00:B1:6B:00:B5
            inet addr:192.168.201.1  Bcast:192.168.201.255  Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:960 errors:0 dropped:9 overruns:0 frame:0
            TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:1000
RX bytes:71280 (69.6 KiB) TX bytes:14777 (14.4 KiB)
```

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/stw_com_board/wlan_enable	p1	-	<p>Directly shuts down or starts the WLAN module.</p> <p>p1:</p> <p>< 0 ></p> <ul style="list-style-type: none"> triggers pin to low level stops the WLAN module <p>< 1 ></p> <ul style="list-style-type: none"> triggers pin to high level starts the WLAN module



WARNING:

The wlan_enable pin must not be toggled by the user. This pin is automatically managed by the kernel.

6.4.3.1 Access-Point-Mode

Connection: Access-Point-Mode (AP mode)

The AP mode is the default mode of the TC3G WLAN module.

Interface rc.conf: Setup IP parameters for startup in AP mode

Setup the IP handling strategy in etc/init.d/rc.conf

```
# ----- NETWORK WLAN0:
# WLAN0_CONF="AP" sets the WLAN interface into AP mode.
export WLAN0_CONF="AP"
# IP of wlan0
export WLAN0_IP="192.168.201.1"
# Submask of wlan0
export WLAN0_SUBMASK="255.255.255.0"
# Broadcast address of wlan0
export WLAN_BR="192.168.201.255"
# Gateway of wlan0
export WLAN_GW="192.168.201.1"
# DNS for wlan0
export WLAN0_DNS="192.168.201.1"
```



NOTE:

To apply the changes in rc.conf, the system needs to be rebooted.

Setup wpa_supplicant.conf

Setup the access-point mode in /etc/wlan/hostapd.conf

```
##### hostapd configuration file #####

interface=wlan0
# Driver interface type
driver=nl80211

# Interface for separate control program. If this is specified, hostapd
# will create this directory and a UNIX domain socket for listening to requests
# from external programs (CLI/GUI, etc.) for status information and
# configuration. The socket file will be named based on the interface name, so
# multiple hostapd processes/interfaces can be run at the same time if more
# than one interface is used.
# /var/run/hostapd is the recommended directory for sockets and by default,
# hostapd_cli will use it when trying to connect with hostapd.
ctrl_interface=/var/run/hostapd

# Access control for the control interface can be configured by setting the
# directory to allow only members of a group to use sockets. This way, it is
# possible to run hostapd as root (since it needs to change network
# configuration and open raw sockets) and still allow GUI/CLI components to be
# run as non-root users. However, since the control interface can be used to
# change the network configuration, this access needs to be protected in many
# cases. By default, hostapd is configured to use gid 0 (root). If you
# want to allow non-root users to use the control interface, add a new group
# and change this value to match with that group. Add users that should have
# control interface access to this group.
#
# This variable can be a group name or gid.
ctrl_interface_group=0

##### IEEE 802.11 related configuration #####

# SSID to be used in IEEE 802.11 management frames
# Will be set automatically to hostname
#ssid=tc3g

# Country code (ISO/IEC 3166-1). Used to set regulatory domain.
# Set as needed to indicate country in which device is operating.
# This can limit available channels and transmit power.
# Default is the minimum subset of all available restrictions (world 00)
#country_code=DE

# Enable IEEE 802.11d. This advertises the country_code and the set of allowed
# channels and transmit power levels based on the regulatory limits. The
# country_code setting must be configured with the correct country for
# IEEE 802.11d functions.
# (default: 0 = disabled)
#ieee80211d=1
# Enable IEEE 802.11h. This enables radar detection and DFS support if
# available. DFS support is required on outdoor 5 GHz channels in most countries
# of the world. This can be used only with ieee80211d=1.
# (default: 0 = disabled)
#ieee80211h=1
# ieee80211n: Whether IEEE 802.11n (HT) is enabled
# 0 = disabled (default)
# 1 = enabled
# Note: You will also need to enable WMM for full HT functionality.
#ieee80211n=1

# Operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g,
# ad = IEEE 802.11ad (60 GHz); a/g options are used with IEEE 802.11n, too, to
```



```
# specify band)
# Default: IEEE 802.11b
hw_mode=g

# Channel number (IEEE 802.11)
# (default: 0, i.e., not set)
# Please note that some drivers do not use this value from hostapd and the
# channel will need to be configured separately with iwconfig.
#
# If CONFIG_ACS build option is enabled, the channel can be selected
# automatically at run time by setting channel=acs_survey or channel=0, both of
# which will enable the ACS survey based algorithm.
channel=6

# for 802.11a or 802.11g networks
# These parameters are sent to WMM clients when they associate.
# The parameters will be used by WMM clients for frames transmitted to the
# access point.
#
# note - txop_limit is in units of 32microseconds
# note - acm is admission control mandatory flag. 0 = admission control not
# required, 1 = mandatory
# note - here cwMin and cmMax are in exponent form. the actual cw value used
# will be (2^n)-1 where n is the value given here
#
wmm_enabled=1

##### WPA/IEEE 802.11i configuration #####

# Enable WPA. Setting this variable configures the AP to require WPA (either
# WPA-PSK or WPA-RADIUS/EAP based on other configuration). For WPA-PSK, either
# wpa_psk or wpa_passphrase must be set and wpa_key_mgmt must include WPA-PSK.
# Instead of wpa_psk / wpa_passphrase, wpa_psk_radius might suffice.
# For WPA-RADIUS/EAP, ieee8021x must be set (but without dynamic WEP keys),
# RADIUS authentication server must be configured, and WPA-EAP must be included
# in wpa_key_mgmt.
# This field is a bit field that can be used to enable WPA (IEEE 802.11i/D3.0)
# and/or WPA2 (full IEEE 802.11i/RSN):
# bit0 = WPA
# bit1 = IEEE 802.11i/RSN (WPA2) (dot11RSNAEnabled)
wpa=2

# WPA pre-shared keys for WPA-PSK. This can be either entered as a 256-bit
# secret in hex format (64 hex digits), wpa_psk, or as an ASCII passphrase
# (8..63 characters) that will be converted to PSK. This conversion uses SSID
# so the PSK changes when ASCII passphrase is used and the SSID is changed.
# wpa_psk (dot11RSNAConfigPSKValue)
# wpa_passphrase (dot11RSNAConfigPSKPassPhrase)
#wpa_psk=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
wpa_passphrase=darth_vader

# Set of accepted cipher suites (encryption algorithms) for pairwise keys
# (unicast packets). This is a space separated list of algorithms:
# CCMP = AES in Counter mode with CBC-MAC [RFC 3610, IEEE 802.11i/D7.0]
# TKIP = Temporal Key Integrity Protocol [IEEE 802.11i/D7.0]
# Group cipher suite (encryption algorithm for broadcast and multicast frames)
# is automatically selected based on this configuration. If only CCMP is
# allowed as the pairwise cipher, group cipher will also be CCMP. Otherwise,
# TKIP will be used as the group cipher.
# (dot11RSNAConfigPairwiseCiphersTable)
# Pairwise cipher for WPA (v1) (default: TKIP)
#wpa_pairwise=TKIP CCMP

# Pairwise cipher for RSN/WPA2 (default: use wpa_pairwise value)
#rsn_pairwise=CCMP
```

How to switch from 2.4 GHz to 5 GHz Access-Point-Mode

The switching can be realized by adapting three configuration values of the /etc/wlan/hostapd.conf-file. Refer to Examples of Country Codes (see "[Examples of Country Codes](#)" on page 63) for country codes.

1. Country Code: country_code:

Adapt the "country_code". The default setting is '00' --> 'world'.

The world setting represents the combination of all available Wi-Fi restrictions. In some countries 5 GHz channels are not available.

In Germany 5 GHz is allowed. Setting the 'country_code' to Germany would be done by this configuration:

```
country_code=DE
```

2. Hardware mode: hw_mode:

In the hostapd.conf-file the parameter "hw_mode" must be adapted. In order to use 5 GHz it needs to be set to 'a':

```
hw_mode=a
```

3. Wi-Fi channel: channel

Set the Wi-Fi channel to a free channel. To retrieve available channels use the "iw list" command. A free channel in Germany for example can be '44':

```
channel=44
```

After setting the correct value, the TC3G needs to be restarted. The last parameter that needs to be checked is the transmit power. It can be read with:

```
# iwconfig wlan0
wlan0 IEEE 802.11abgn Mode:Master Tx-Power=20 dBm
      Retry long limit:7   RTS thr:off   Fragment thr:off
      Power Management:on
```

The transmit power in this example is 20 dBm that matches with 100 mW. To reduce the transmit power to e.g. 10 dBm, the following command can be used:

```
# iwconfig wlan0 txpower 10
```

6.4.3.2 Managed-Mode

Connection: Managed-Mode



NOTE:

To use the TC3G WLAN module in Managed-Mode, the ynetwork daemon has to be activated.

Interface rc.conf: Setup IP parameters for startup in Managed-Mode

Setup the IP handling strategy in etc/init.d/rc.conf

```
# ----- NETWORK WLAN0:
# WLAN0_CONF="CLIENT" sets the WLAN in managed mode. In this mode the ynetworkd
# connects the TC3G WLAN module to other networks.
export WLAN0_CONF="CLIENT"
```

```
# IP of wlan0, for managed mode this parameter is not needed
export WLAN0_IP="192.168.201.1"
# Submask of wlan0, for managed mode this parameter is not needed
export WLAN0_SUBMASK="255.255.255.0"
# Broadcast address of wlan0, for managed mode this parameter is not needed
export WLAN_BR="192.168.201.255"
# Gateway of wlan0, for managed mode this parameter is not needed
export WLAN_GW="192.168.201.1"
# DNS for wlan0, for managed mode this parameter is not needed
export WLAN0_DNS="192.168.201.1"
```


NOTE:

To apply the changes in rc.conf, reboot the system.

Setup wpa_supplicant.conf

Setup the managed mode in /etc/wlan/wpa_supplicant.conf

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1

# Country code (ISO/IEC 3166-1, see also Examples of Country Codes (see "Examples of Country Codes" on page 63)).
# The Country code is used to set regulatory domain.
# Set as needed to indicate the country in which the device is operating.
# This can limit available channels and transmission power.
# Default is the minimum subset off all available restrictions (world 00)
# country=DE

# The scan_ssid parameter is needed for networks with hidden SSIDs.
# It needs to be placed directly below the 'ssid=' line.
# scan_ssid=1

network={
    ssid="test_net"
    psk="12345678"
}
```

6.4.3.3 Examples of Country Codes

Country Code/ Country	802.11 Bands for hw_mode	Allowed Wi-Fi Channels	Maximum Transmit Power (Radio Tx + Antenna Gain = EIRP)	Frequency Range (GHz)
AT / Austria	a	36, 40, 44, 48	60 mW EIRP	5.15-5.25
	b/g	1 - 11	100 mW EIRP	2.4-2.4835
DE / Germany	a	36, 40, 44, 48 52, 56, 60, 64 104, 108, 112, 116, 120, 124, 128, 132, 140	200 mW EIRP 200 mW EIRP 1 W EIRP	5.15-5.25 5.25-5.35 5.47-5.725

Country Code/ Country	802.11 Bands for hw_mode	Allowed Wi-Fi Channels	Maximum Transmit Power (Radio Tx + Antenna Gain = EIRP)	Frequency Range (GHz)
	b/g	1 - 11	100 mW EIRP	2.4-2.4835
FR / France	a	36, 40, 44, 48 52, 56, 60, 64	200 mW EIRP 200 mW EIRP	5.15-5.25 5.25-5.35
	b/g	1 - 7 8 - 11	100 mW EIRP 100 mW EIRP	2.4-2.4835 2.4-2.454
GB / United Kingdom	a	36, 40, 44, 48 52, 56, 60, 64 104, 108, 112, 116, 120, 124, 128, 132, 140	200 mW EIRP 200 mW EIRP 1 W EIRP	5.15-5.25 5.25-5.35 5.47-5.725
	b/g	1 - 11	100 mW EIRP	2.4-2.4835
IT / Italy	a	36, 40, 44, 48 52, 56, 60, 64 104, 108, 112, 116, 120, 124, 128, 132, 140	200 mW EIRP 200 mW EIRP 1 W EIRP	5.15-5.25 5.25-5.35 5.47-5.725
	b/g	1 - 11	100 mW EIRP	2.4-2.4835
US / United States of America	a	36, 40, 44, 48 52, 56, 60, 64	50 mW+6 dBi=200 mW 250 mW+6 dBi=1 W	5.15-5.25 5.25-5.35
	b/g	1 - 11	1 W Conducted Output	2.4-2.4835

6.4.4 IP handling

This chapter describes how

- DNSMASQ
- IP forwarding
- NAT

can be handled on the TC3G system.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/dnsmasq	p1	-	<p>This shell script starts the DHCP and DNS services for the selected interface.</p> <p>p1:</p> <ul style="list-style-type: none"> < start > starts service for eth0 / wlan0 if rc.conf eth0 configuration is static / wlan0 configuration is set to "AP" and DNS = "ON", else no DNSMASQ service is set for eth0 / wlan0 < stop > stops DHCP and DNS service for eth0 / wlan0 < restart > processes stop and start
/etc/init.d/scripts/ip_forwarding	p1	-	<p>This shell script starts the IP Forwarding and NAT services for the selected interface.</p> <p>p1:</p> <ul style="list-style-type: none"> < start > starts service for eth0 / wlan0 / ppp0 if service is set to "ON" in rc.conf and NAT, else IP Forwarding is switched off anyway. NAT for ppp0 is set by default. < stop > stops IP Forwarding anyway < restart reload > processes stop and start

Interface at rc.conf: Set DNSMASQ, IP forwarding and NAT parameters

```
# Handling those settings in etc/init.d/rc.conf

# ----- DNSMASQ / IP FORWARDING / NAT
# Start the dnsmasq service on eth0 (works only with ETH0_CONF=static or uboot)
# (uses /etc/dnsmasq_lan.conf) by setting export to "ON", else service will be deactivated.
export ETH0_DNSMASQ="OFF"
# Start the dnsmasq service on wlan0 (works only with WLAN0_CONF=AP)
# (uses /etc/dnsmasq_wlan.conf) by setting export to "ON", else service will be
deactivated.
export WLAN0_DNSMASQ="ON"
# Activate service by setting export to "ON", else disables IP forwarding for eth0 / wlan0/
ppp0.
export IP_FORWARDING="OFF"
# NAT options will be used in case IP forwarding is activated.
# Deactivate service by setting export to "OFF", else activates the NAT for network
interface (wlan0 / eth0).
export ETH0_NAT="OFF"
export WLAN0_NAT="OFF"
```

6.4.5 CAN

High Level Hardware Access

The Socket CAN package is an implementation of CAN (Controller Area Network) protocols for Linux. CAN is a networking technology which has wide-spread use in automation, embedded devices, and automotive fields. While there have been other CAN implementations for Linux based devices, Socket CAN uses the Berkeley socket API, the Linux network stack and implements the CAN device drivers as network interfaces. The CAN socket API has been designed as similar as possible to the TCP/IP protocols to allow programmers, who are familiar with network programming, easily to learn how to use CAN sockets.

Get local information

It is possible to read out the local settings of the can0 and can1 devices, such as RX bytes, TX bytes,....

```
# ifconfig can0; ifconfig can1
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:780 (780.0 B)
          Interrupt:145

can1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:780 (780.0 B)  TX bytes:0 (0.0 B)
          Interrupt:146
```

Change the Transmit Queue Length

It is possible to vary the maximal number of CAN messages in the transmit queue of e.g. can0.

```
# ifconfig can0 txqueuelen 1000
# ifconfig can0
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:145
```

Interface at rc.conf: Activate / Deactivate CAN 0/1 and set bitrate

```
# CAN 0 "y" activate, else deactivate
export SYSCFG_CAN0="y"
# Baudrate of can0
export SYSCFG_CAN0_BAUD="500000"
# CAN 1 "y" activate, else deactivate
export SYSCFG_CAN1="y"
# Baudrate of can1
export SYSCFG_CAN1_BAUD="500000"
```

Example: Change bitrate of can0

Handling the user space interface can0 by using /sbin/ifconfig provided by the BusyBox multi-call binary and /sbin/ip

Stop can0, change bitrate to 250000 and restart can0.

```
# ifconfig can0 down
# /bin/ip link set can0 up type can bitrate 250000
# ifconfig can0 up
```

Example: Socket CAN for C programming

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/if.h>

#include <linux/can.h>
#include <linux/can/raw.h>
#include <string.h>

#include "stwtypes.h"

/* At time of writing, these constants are not defined in the headers */
/* ... */

/* Somewhere in your application */

/* Create the socket */
sint16 sl6_Socket = socket( PF_CAN, SOCK_RAW, CAN_RAW);

/* Locate the interface you wish to use */
struct ifreq t_Ifr;
strcpy(t_Ifr.ifr_name, "can0");
ioctl(sl6_Socket, SIOCGIFINDEX, &t_Ifr); /* Ifr.ifr_ifindex gets filled with that
device's index*/

/* Select that CAN interface, and bind the socket to it.*/
struct sockaddr_can t_Addr;
t_Addr.can_family = AF_CAN;
t_Addr.can_ifindex = t_Ifr.ifr_ifindex;
bind( sl6_Socket, (struct sockaddr*)&t_Addr, sizeof(t_Addr));

/* Send a message to the CAN bus */
struct can_frame t_Frame;
t_Frame.can_id = 0x123;
strcpy( t_Frame.data, "foo");
t_Frame.can_dlc = strlen( t_Frame.data);
sint16 sl6_BytesSent = write( sl6_Socket, &t_Frame, sizeof(t_Frame));

/* Read a message back from the CAN bus */
sint16 sl6_BytesRead = read( sl6_Socket, &t_Frame, sizeof(t_Frame));
```

6.4.6 Bluetooth

This chapter describes the Bluetooth interface on the TC3G and how a Bluetooth connection can be established to other systems.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/bt	p1	-	This shell script handles the Bluetooth module and is automatically started by the boot up sequence

Path	Input information	Output information	Description
			<p>(see "Boot up / Shut down sequence" on page 42).</p> <p>p1: defines what action should be done</p> <p>< start ></p> <ul style="list-style-type: none"> initializes bluetooth port starts bluetooth module starts the bluetooth daemon starts bluetooth services (serial port service, obex ftp service) starts bt_connect script <p>< stop ></p> <ul style="list-style-type: none"> stops bt_connect script stops bluetooth services stops bluetooth daemon stops bluetooth module <p>< restart ></p> <ul style="list-style-type: none"> calls stop calls start
/etc/init.d/scripts/bt_connect	-	-	This shell script registers the Bluetooth address of every device that is ready for pairing. It also registers a password (default "0000").

Get local information

It is possible to read out the local settings of the bluetooth device, such as address, name, class, hci version,....

```
# hciconfig -a
hci0:  Type: BR/EDR  Bus: UART
      BD Address: 84:DD:20:BD:77:3F  ACL MTU: 1021:4  SCO MTU: 180:4
      UP RUNNING PSCAN ISCAN
      RX bytes:907 acl:0 sco:0 events:40 errors:0
      TX bytes:722 acl:0 sco:0 commands:36 errors:0
      Features: 0xff 0xfe 0x2d 0xfe 0xdb 0xff 0x7b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF
      Link mode: SLAVE ACCEPT
      Name: 'TC3G-141846121005'
      Class: 0x100110
      Service Classes: Object Transfer
      Device Class: Computer, Handheld
      HCI Version: 4.0 (0x6)  Revision: 0x0
      LMP Version: 4.0 (0x6)  Subversion: 0x1f40
      Manufacturer: Texas Instruments Inc. (13)
```

Get environmental information

Scan the environment for other devices


```
# hcitool scan
Scanning ...
    00:15:83:41:79:8D      TEST-PC
```

Achieving a bluetooth connection to the TC3G from different operating systems

Connect to TC3G from windows based systems over serial port

Windows 7 or newer operating system based devices can connect to the TC1. The device has to be added with paring code "0000".

Check the properties of the device to identify the used serial port (e.g. COM13).

The serial port can be selected in a terminal program (e.g. TeraTerm) to connect to the TC3G.



NOTE:

The default configuration of the TC3G does not need to be changed.

Connect to TC3G from windows based systems for file sharing

Windows 7 or newer operating system based devices can connect to the TC1. The device has to be added with paring code "0000".

In order to send a file via Obex FTP it is possible to add the TC3G as a storage medium.



NOTE:

The file sharing folder on the TC3G is /tmp/.

The default configuration of the TC3G does not need to be changed.

Connect to TC3G from android based systems for file sharing

In order to connect to the TC3G from android based systems, a similar APP like "Bluetooth File Transfer" has to be used.



NOTE:

The file sharing folder on the TC3G is /tmp/.

The default configuration of the TC3G does not need to be changed.

Connect to TC3G from linux based systems for file sharing

Configure the device for file sharing from a linux based system by setting up a specific communication device in /etc/bluetooth/rfcomm.conf.

```
# RFCOMM configuration file.

rfcomm0 {
    # Automatically bind the device at startup
    bind yes;

    # Bluetooth address of the device
    device 00:15:83:41:79:8D;

    # RFCOMM channel for the connection
    channel 7;

    # Description of the connection
```

```
}      comment "Linux VM";
```

It is necessary to gain root privileges. Use the obexfs tool to connect to the TC1 and mount the connection to bluetooth_test

```
root@pc-name:# obexfs -b 00:07:80:97:E7:BB /home/bluetooth_test/
```

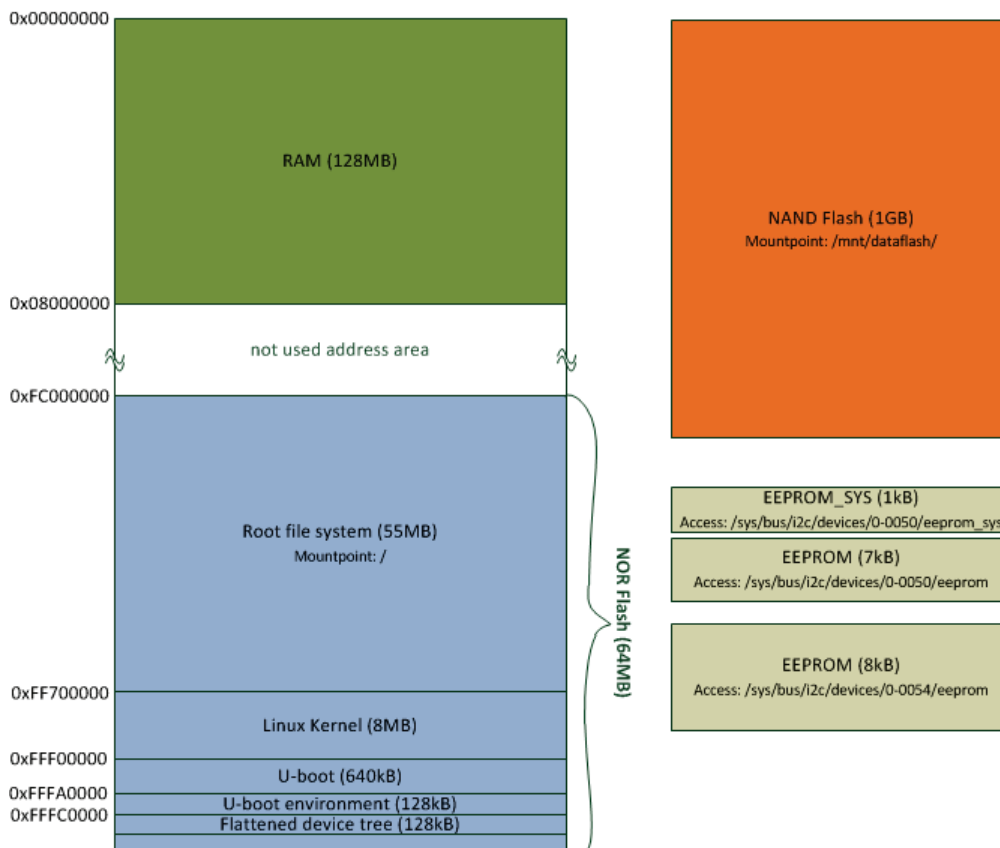
6.5 Memory handling

The memory system of the device includes (see also Technical Data):

- RAM
- NOR
- NAND
- EEPROM

Overview of the memory mapping.

- Root file system image: Unsorted Block File System (UBIFS)
- Kernel image: Includes the linux kernel
- U-Boot: Bootloader which initializes and loads the linux kernel
- U-Boot environment: U-Boot specific environment variables
- Flattened device tree: Used by the U-Boot to identify the hardware
- NAND Flash: Path for the user data
- EEPROM: User area



High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/filesystem	p1		This user space interface includes the device and

Path	Input information	Output information	Description
			<p>partition information of the system.</p> <p>p1:</p> <p>< start ></p> <ul style="list-style-type: none"> copies data from NOR to RAM and remounts the temporarily file system, mounts USB device to /mnt/usbdisk, prepare NAND

Example at rc.conf: Use shared memory

Useful for other applications that need shared memory.

1. Check export variable TMPFS in /etc/init.d/rc.conf

```
export TMPFS="tmpfs"
```

2. Enable shared memory section in /etc/init.d/scripts/filesystem

```
# Shared memory, in case a application would need it:
if [ "$TMPFS" = "tmpfs" ]
then
mount -t $TMPFS shm /dev/shm
fi
# Shared memory end
```

Example: Read file system usage statistics by using /bin/df

The dataflash is the only partition that is not mounted to RAM. All other have a partition size of 8MB exclude the rootfs itself which has a size of 48 MB.

After the kernel has been loaded by the U-Boot from RAM, the system partitions are:

```
# df
Filesystem      Size      Used Available Use% Mounted on
ubi0:rootfs     48.6M    39.7M      8.9M   82% /
rwfs            7.9M    312.0K      7.6M    4% /mnt/rwfs
rwfs            7.9M    312.0K      7.6M    4% /tmp
rwfs            7.9M    312.0K      7.6M    4% /var
rwfs            7.9M    312.0K      7.6M    4% /dev
rwfs           926.3M     14.2M    907.4M    2% /etc/ppp
rwfs            7.9M    312.0K      7.6M    4% /mnt/usbdisk
ubil:dataflash  926.3M     14.2M    907.4M    2% /mnt/dataflash
unionfs         926.3M     14.2M    907.4M    2% /usr
unionfs         926.3M     14.2M    907.4M    2% /etc
unionfs         926.3M     14.2M    907.4M    2% /opt
unionfs         926.3M     14.2M    907.4M    2% /lib
unionfs         926.3M     14.2M    907.4M    2% /home
unionfs         926.3M     14.2M    907.4M    2% /root
```



NOTE:

In case of writing log files to e.g. var directory, it is important to keep in mind that the allocated size is 8MB. If the log file reaches the size of 8MB, not further data can be stored.

Example: Read out EEPROM

Displays in two byte hexadecimal by using /usr/bin/hexdump.

```
# hexdump -x /sys/bus/i2c/devices/0-0054/eeprom
00000000  ffff  ffff  ffff  ffff  ffff  ffff  ffff  ffff
..
00020000
```

Display canonical hex+ASCII, 16 byte per line.

```
# hexdump -C /sys/bus/i2c/devices/0-0054/eeprom
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff  |.....|
..
00002000
```

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/mtd	-	List of devices to partition parameters	This user space interface includes the device and partition information of the system.
/proc/cmdline	-	-	This user space interface includes the command line parameters which the kernel itself was booted.

Example:

Read device and partition information

```
# cat /proc/mtd
dev:      size  erasesize  name
mtd0: 03700000 00020000 "rootfs"
mtd1: 00800000 00020000 "kernel"
mtd2: 00040000 00020000 "u-boot"
mtd3: 00020000 00020000 "u-boot-env"
mtd4: 000a0000 00020000 "fdt"
mtd5: 000a0000 00020000 "reserved"
mtd6: 40000000 00020000 "nand-flash"
```

Get command line parameters. Parameter ubi.mtd=0 indicates to which mtd block the file system is mounted to.

```
# cat /proc/cmdline
console=ttyPSC3,115200 ubi.mtd=0 root=ubi0:rootfs rw rootfstype=ubifs quiet
ip=192.168.4.253:192.168.4.204::255.255.255.0::eth0:off panic=1
```

6.5.1 NOR Flash

Low Level Hardware Access

Path	Input information	Output information	Description
/dev/mtd0	-	-	This character device includes the interface to the root file system "(root) /".
/dev/mtd1	-	-	This character device includes the interface to the kernel image.
/dev/mtd2	-	-	This character device includes the interface to the U-Boot.
/dev/mtd3	-	-	This character device includes the interface to the U-Boot environment variables.
/dev/mtd4	-	-	This character device includes the interface to the flattened device tree (FDT).
/dev/mtd5	-	-	This character device includes the interface to a reserved part of the FDT (/dev/mtd4).
/dev/mtd6	-	-	This character device includes the interface to the nand-flash.

Example: Read memory information

Read memory information from /dev/mtd0 by using /usr/sbin/mtdinfo

```
# mtdinfo /dev/mtd0
mtd0
Name:                rootfs
Type:                nor
Eraseblock size:     131072 bytes, 128.0 KiB
Amount of eraseblocks: 440 (57671680 bytes, 55.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:       1 byte
Character device major/minor: 90:0
Bad blocks are allowed: false
Device is writable:  true
```



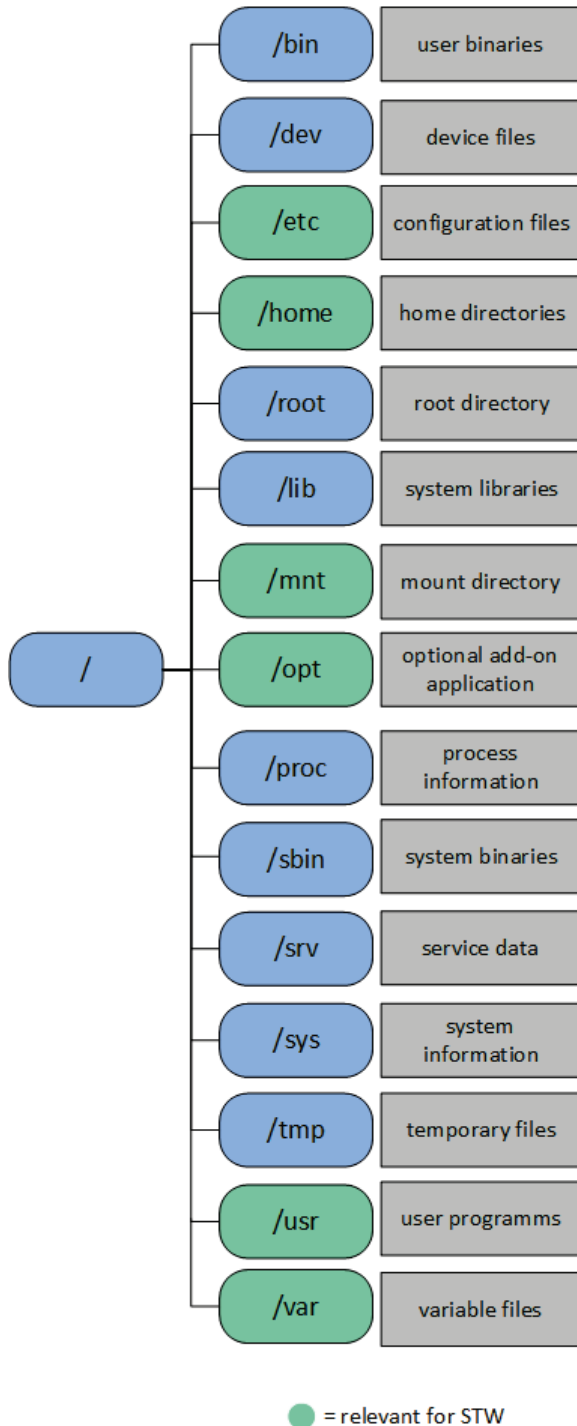
WARNING:

Do not use the NOR flash for data storage.

Due to the limited number of writing cycles, it is highly recommended to use the NOR flash either in read-only mode or activate the overlay mechanism to gain the NOR flash's lifetime.

6.5.1.1 Linux Folder Structure

The following chapter will explain all folders of the Linux directory structure that is created in the NORFlash.



/root

- This is the root users's home directory. Here are saved all personal data and terminal configurations of the root user.
- Only the root user has the writing permission for this directory.



NOTE:

Do not get confused between "/" and /root. "/" is the main folder from that all directories start.

/bin – User Binaries

This directory contains:

- Binary executables
- Common linux commands needed to use in single-user modes
- Commands used by all users of the system. For example: ps, ls, ping, grep, cp.

/dev – Device Files

- Contains device files.
- Examples: Terminal devices, USB, or any device attached to the system like /dev/tty1, /dev/usbdev1.1

/etc – Configuration Files

- Contains configuration files that are required for all programs.
- This also contains startup and shutdown shell scripts used to start and stop single programs. For example: /etc/resolv.conf, /etc/init.d/rc.conf

/home – Home Directories

- This directories is for all users to store their personal files. For example: /home/default

/lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*, so*

/mnt – Mount Directory

- Temporary directory, where system administrators can mount filesystems.

/opt – Optional add-on Applications

- Contains add-on applications from individual vendors.
- Add-on applications should be installed under either /opt/ or /opt/ sub-directory.

/proc – Process Information

- Contains information about system processes.
- This pseudo filesystem provides a file-system like interface to the kernel. This allows applications and users to fetch information from the kernel using normal filesystem I/O operation. For example: /proc/{pid} directory contains information about the process with that particular PID.
- Contains text information about system resources. For example: /proc/uptime

/sbin – System Binaries

- Contains binary executables like /bin.

- The linux commands located under this directory are used typically by the system administrator for system maintenance purpose. For example: iptables, reboot, fdisk, ifconfig

/srv – Service Data

Contains server specific and services related data.

/sys – System information

- Modern Linux distributions include a /sys directory as a pseudo filesystem (sysfs, comparable to /proc), which stores and allows modification of the devices connected to the system.
- Many traditional Unix and Unix-like operating systems use /sys as a symbolic link to the kernel source tree.

/tmp – Temporary Files

- Contains temporary files created by system and users.
- Files in this directory are deleted when system is rebooted.

/usr – User Programs

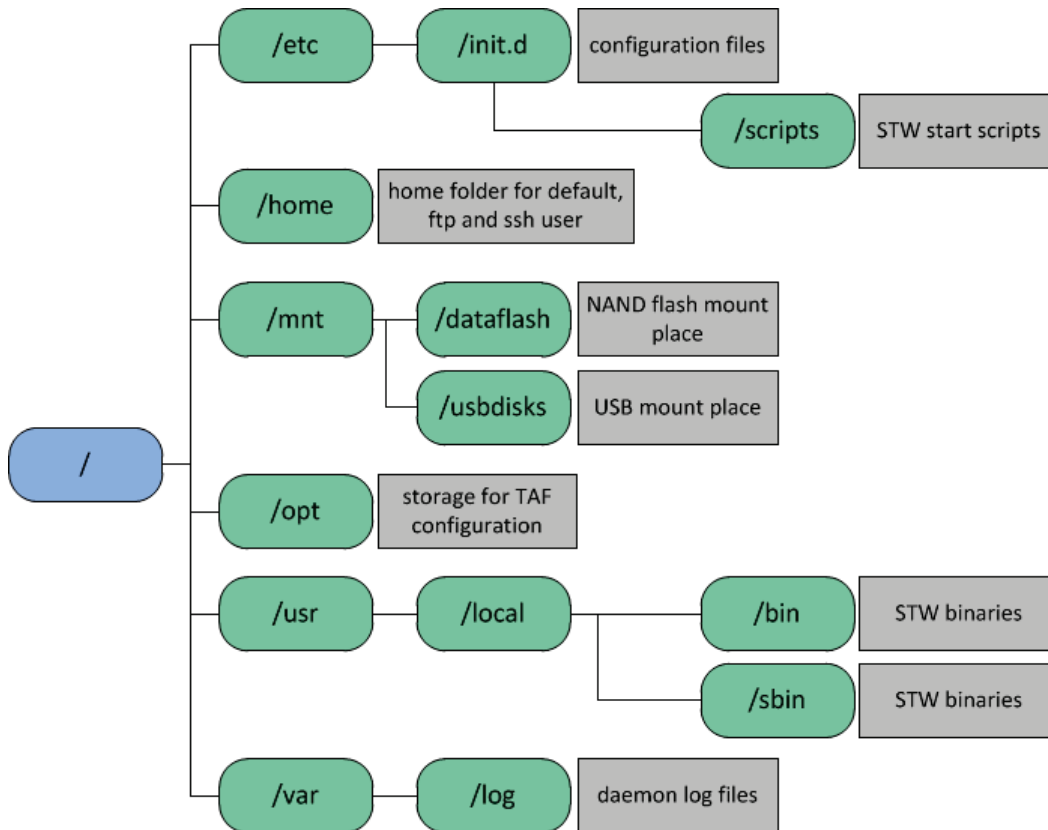
- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If a user binary cannot be found under /bin, it might be under /usr/bin. For example: at, awk, chat, less
- /usr/sbin contains binary files for system administrators. If a system binary cannot be found under /sbin, it might be under /usr/sbin. For example: ftpd, httpd
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains user programs that are installed from source.

/var – Variable Files

- Contains files with expected increasing size (content).
- Example: System log files (/var/log); packages and database files (/var/lib); lock files (/var/lock); temp files needed across reboots (/var/tmp);

6.5.1.2 STW Folder Structure

Software from STW uses the following folders:



/etc/init.d

Contains the configuration files of all TAF daemons. Only if the configuration file is in the folder, the corresponding daemon will start. The folder also includes the user startup script rc.local and the global startup script rcS.

/etc/init.d/scripts

Contains all boot scripts which are necessary for starting up the system.

/home

Contains the home folder of the default, ftp and ssh user.

/mnt/dataflash

Contains the NAND when it is mounted.

Mount the NAND in order to store files into it. Otherwise the files will be stored into the NOR flash.

/mnt/usbdisks

All USB storage devices will be mounted to /mnt/usbdisks/.

For example:

- /mnt/usbdisks/sda1
- /mnt/usbdisks/sda2

/mnt/opt

Directory to store data logger configurations or server configurations.

/usr/local/bin

Contains STW specific binaries and scripts like cangen or nand_mount_sh.

/usr/local/sbin

Contains STW specific scripts.

/var/log

Contains the log files of the current activated TAF components, e.g. files with the syntax <Name of TAF component>.log.

6.5.1.3 Overlay Filesystem

Overlay filesystem

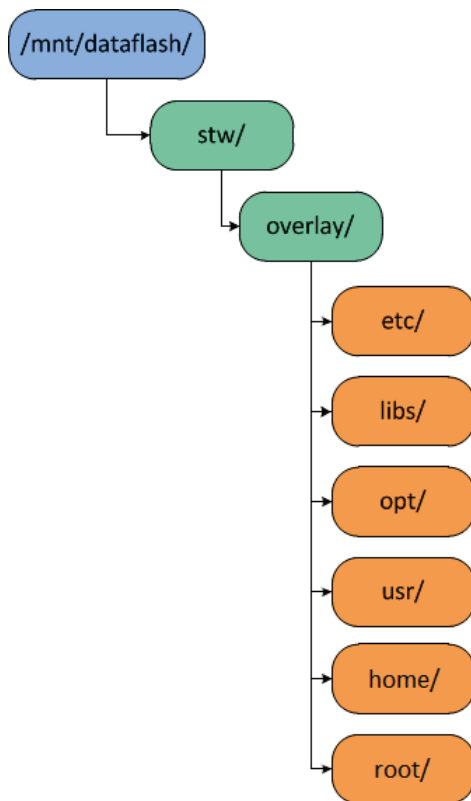
The overlay filesystem is a feature that uses the mechanism of the UnionFS filesystem of the Linux kernel. For details see <http://unionfs.filesystems.org/> (see Link to UnionFS homepage - <http://unionfs.filesystems.org/>).

If the system is updated with the latest BSP, all modifications will be overwritten. The root filesystem must be customized again, which is an unnecessary workload and can cause wrong configuration of the devices that are installed in the vehicle.

The overlay filesystem prevents data from being overwritten when an update to the latest BSP must be performed. It also supports the procedure to restore user software and script settings after an update.

How the mechanism of the overlay filesystem works

In addition to the root filesystem partition, which should be always mounted with read-only access, a data partition "/mnt/dataflash" is available. The following directory structure can be found on that partition:



At startup of the TC3G, the /etc, /usr, /libs, /home, /opt and /root directories of the root filesystem are automatically overlaid with the content of directory /mnt/dataflash/stw/overlay.

For example the /etc directory is overlaid with /mnt/dataflash/stw/overlay/etc.

If a user changes or deletes a file in one of these root filesystem directories, the changes will be stored in the corresponding /mnt/dataflash/stw/overlay directory.

Activate the overlay filesystem in the U-Boot

1. Connect the TC3G to the PC over RS232
2. Press and hold ESC while booting the TC3G

```

u-boot 2012.10-svn9064 (Jun 01 2016 - 10:58:25) tc3_b STW-V2.05r1

CPU:   MPC5200B v2.2, Core v1.4 at 396 MHz
       Bus 132 MHz, IPB 132 MHz, PCI 66 MHz
Board: STW TC3
I2C:   343 kHz, ready
DRAM:  128 MiB
Flash: 64 MiB
NAND:  1024 MiB
In:     serial
Out:    serial
Err:    serial
Net:    FEC

Hit 3 * ESC key to stop autoboot...
=>
  
```

"=>" this arrow means you are now in the U-Boot.

Type '?' or 'help' in case you need it

To activate the overlay filesystem, add the parameter "overlay_on" to the U-Boot bootargs:

```
setenv ubifsargs setenv bootargs console=ttyPSC3,115200 ubi.mtd=0 root=ubi0:rootfs rw
rootfstype=ubifs quiet overlay_on
```

After saving the U-Boot variables and booting the TC3G the overlay feature can be used.

Activate the overlay filesystem in the user space

To activate the overlay filesystem, add the parameter "overlay_on" to the user space:

```
fw_setenv ubifsargs setenv bootargs console=ttyPSC3,115200 ubi.mtd=0 root=ubi0:rootfs rw
rootfstype=ubifs quiet overlay_on
```

After restarting the TC3G the overlay feature can be used.

Examples

Example1:

If the user changes the /etc/init.d/rc.conf file, the changes will be stored to /mnt/dataflash/stw/overlay/etc/init.d/rc.conf.

The original root filesystem remains unchanged.

If the user deletes the /mnt/dataflash/stw/overlay/etc/init.d/rc.conf file, the original /etc/init.d/rc.conf file from the rootfs will be restored.

Example2:

If the user deletes the /etc/init.d/scripts/bt script in the root filesystem, then the TC3G behaves like the file is gone. But the file is not really deleted. Only a flag in the corresponding overlay directory is set. In this case the flag is set in the /mnt/dataflash/stw/overlay/etc/init.d/script directory.

```
# ls -al /mnt/dataflash/stw/overlay/etc/init.d/scripts/
total 3
drwxrwxr-x  2 root    root          288 Nov 24 15:57 .
drwxrwxr-x  3 root    root          360 Nov 19 15:41 ..
-----  1 root    root              0 Nov 24 15:57 .wh.bt
```

After removing the flag or deactivating the overlay, the file is restored in the root filesystem.

6.5.2 NAND Flash

The NAND flash is mounted to the path /mnt/dataflash.



WARNING:

It is recommended to store all the user data to /mnt/dataflash to avoid data loss.

High Level Hardware Access

Path	Input information	Output information	Description
/usr/local/bin/nand_prepare.sh	-	-	<p>This shell script prepares the NAND flash to use it with UBIFS.</p> <p>Assumptions:</p> <ul style="list-style-type: none"> the NAND is organized in one MDT block with size 1GByte flash will be attached to UBI device /dev/ubi0 <p>Following steps are done:</p> <ul style="list-style-type: none"> formatting the whole NAND flash to use it with the UBI driver <hr/> <p>WARNING:</p> <p>All data on the flash are deleted.</p> <hr/> <ul style="list-style-type: none"> creating one volume named "dataflash" on the flash using all available flash memory
/usr/local/bin/nand_mount.sh	-	-	<p>This shell script attaches and mounts the volume "dataflash" of the NAND flash to /mnt/dataflash.</p> <p>Assumption:</p> <ul style="list-style-type: none"> the NAND flash contains a volume called "dataflash" (e.g. use nand_prepare.sh to format flash and create the volume) mount point /mnt/dataflash must exist
/usr/local/bin/nand_ismounted.sh	-	p1	<p>This shell script checks if the NAND flash is mounted to /mnt/dataflash.</p> <p>p1:</p> <ul style="list-style-type: none"> < 0 > NAND flash is mounted < 1 > NAND flash is NOT mounted
/usr/local/bin/nand_umount.sh	-	-	<p>This shell script unmounts and detaches the volume "dataflash" of the NAND flash on</p>

Path	Input information	Output information	Description
			/mnt/dataflash.

Low Level Hardware Access

Path	Input information	Output information	Description
/dev/mtd6	-	-	This device includes the entry into the NAND flash.
/mnt/dataflash	-	-	The 1 GB NAND flash is mounted here.

Example: Read memory information

Read memory information of /dev/mtd6 using /usr/sbin/mtdinfo.

<pre># mtdinfo /dev/mtd6 mtd6 Name: nand-flash Type: nand Eraseblock size: 131072 bytes, 128.0 KiB Amount of eraseblocks: 8192 (1073741824 bytes, 1024.0 MiB) Minimum input/output unit size: 2048 bytes Sub-page size: 512 bytes OOB size: 64 bytes Character device major/minor: 90:12 Bad blocks are allowed: true Device is writable: true</pre>			
--	--	--	--

6.5.3 EEPROM

Low Level Hardware Access

Path	Input information	Output information	Description
/sys/bus/i2c/devices/0-0050/eeprom	Write 1 .. 7168 bytes	Read out 1 .. 7168bytes	This user space interface represents the EXTERN 7kB EEPROM. The user can handle it by using a shell script and / or C-Code.
/sys/bus/i2c/devices/0-0050/lock	p1	p2	System file to switch EEPROM between read only and read/write mode. p1: < lock > .. sets EEPROM to read only < kcol > .. sets EEPROM to read / write p2: < locked > .. EEPROM is read only < unlocked > .. EEPROM is readable / writable

Path	Input information	Output information	Description
/sys/bus/i2c/devices/0-0054/eeprom	Write 1 .. 8192 bytes	Read out 1 .. 8192 bytes	This user space interface represents the EXTERN 8kB EEPROM. The user can handle it by using a shell script and / or C-Code
/sys/bus/i2c/devices/0-0054/lock	p1	p2	System file to switch EEPROM between read only and read/write mode p1: < lock > .. sets EEPROM to read only < kcol > .. sets EEPROM to read / write p2: < locked > .. EEPROM is read only < unlocked > .. EEPROM is readable / writable

Example: Read system parameters from EEPROM

```
# stw_eep -r variant
Y_TC3G
```



WARNING:

Don't unlock the eeprom_sys STW eeprom area. Overwriting this area will cause an unexpected system behavior!

Path	Input information	Output information	Description
/sys/bus/i2c/devices/0-0050/eeprom_sys	-	-	This user space interface represents the internal 1kB EEPROM only used by STW.
/sys/bus/i2c/devices/0-0050/lock_sys	p1	p2	System file to switch eeprom_sys between read only and read / write mode. p1: < lock > .. sets EEPROM to read only < kcol > .. sets EEPROM to read / write p2: < locked > .. EEPROM is read only < unlocked > .. EEPROM is readable / writable

6.6 Watchdog

Library functions for the watchdog

The watchdog can be configured with the functions of the system daemon (see "[System daemon](#)" on page 95).

Function	Brief description
ysysd_register_watch_dog (see "ysysd_register_watch_dog" on page 166)	The TAF library function registers the application on system daemon.
ysysd_trigger_watch_dog (see "ysysd_trigger_watch_dog" on page 167)	The TAF library function sets the interval time to the system daemon.
ysysd_cancel_watch_dog (see "ysysd_cancel_watch_dog" on page 168)	The TAF library function deregisters the supervising from the system daemon.

High Level Hardware Access

Path	Input information	Output information	Description
/dev/watchdog	-	-	This device is created by the linux kernel that provides the interface of the software watchdog. The watchdog has to be triggered each 60 seconds.

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/stw_wd/trigger	p1	-	This user space interface handles the hardware watchdog of the system. This has to be toggled each second. p1: < 0 > = sets signal to low < 1 > = sets signal to high < t > = toggles the signal



NOTE:

This user interface has to be toggled within less than one second. Otherwise the system will reboot.

6.7 Temperature Sensor

Low Level Hardware Access

Path	Input information	Output information	Description
/sys/bus/spi/devices/spi32766.1/temp1_input	-	p1	This user space interface reads the current temperature of the temperature sensor in degrees celsius. p1: < 0..n > = Temperature in in degrees celsius multiplied by 1000

Example: Read and calculate the temperature

1. Read out the temperature

```
# cat /sys/bus/spi/devices/spi32766.1/temp1_input
54750
```

2. Calculate the temperature in degrees celsius manually

54750 / 1000 = 54,750 °C

6.8 GPS

For GPS handling use the GPS daemon that handles the GPS interface automatically.

Precondition:

The GPS daemon must be started. The GPS daemon is not started during the boot up automatically.

How to activate the network daemon see GPS daemon (see "[GPS daemon](#)" on page 101)

Library functions for the GPS

Function	Brief
ygpsd_get_gps_data (see " ygpsd_get_gps_data " on page 243)	The TAF library function receives the data from the GPS module.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/gps	p1	-	This shell script starts and stops the internal GPS receiver connected via /proc/stw_gps/port. It is already started by the boot up sequence (see " Boot up / Shut down sequence " on page 42).

Path	Input information	Output information	Description
			p1: <ul style="list-style-type: none"> < start restart > starts the internal GPS receiver

Low Level Hardware Access

Path	Input information	Output information	Description
/dev/ttyPSC2	-	NMEA string from GPS module	This user space interface reads out the NMEA string from the GPS module.
/proc/stw_gps/port	-	NMEA string from GPS module	Symbolic link for the tty-device (/dev/ttyPSC2) the GPS module is connected to.

6.9 Serial



NOTE:

The serial interface is already started during the boot up sequence. It is not necessary to start it manually.

Low Level Hardware Access

Path	Input information	Output information	Description
/dev/ttyPSC6	Transmit characters one by one.	Receives characters one by one.	This device created by the linux kernel provides the serial interface (RS232) of the system. The serial interface is already started by the boot up sequence (see " Boot up / Shut down sequence " on page 42).

6.10 Digital Input / Output

High Level Hardware Access

Example: C - Code Read / Write on PIN1

```
// This example describes how to read / write on /proc/stw_io/PIN1
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <fcntl.h>
#include "stwtypes.h"
```

```
void mv_ToggleOutput (void)
{
    charn cn_Status = 0;
    fd_set t_Readfds;
    struct timeval t_Timeout;
    sint16 s16_Fd; // File descriptor

    // Set timeout to zero --> no wait time
    t_Timeout.tv_sec = 0;
    t_Timeout.tv_usec = 0;
    s16_Fd = open ("/proc/stw_io/PIN1", O_RDWR); // Open PIN1
    if (s16_Fd >= 0)
    {
        // Clear fds
        FD_ZERO (&t_Readfds);

        // Set
        FD_SET (s16_Fd, &t_Readfds);
        if (select (s16_Fd + 1, &t_Readfds, NULL, NULL, &t_Timeout) > 0)
        {
            // Check if the corresponding bit is set in the fds
            if (FD_ISSET (s16_Fd, &t_Readfds))
            {
                // Read from /proc/stw_io/PIN1
                (void)read (s16_Fd, &cn_Status, sizeof (charn));
                // If cn_Status is not Zero 0 == 0x30 than set cn_Status to zero,
                // else set it to 1
                cn_Status = (cn_Status != 0x30)? 0x30: 0x31;
                // Write to /proc/stw_io/PIN1
                (void) write (s16_Fd, &cn_Status, sizeof (charn));
            }
        }
        (void) close(s16_Fd);
    }
}
```

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/stw_io/PIN1	p1	p2	<p>This user space interface reads and writes on digital input / output PIN.</p> <p>p1:</p> <p>< 0 > .. sets digital value to low level</p> <p>< 1..n > .. sets digital value to high level</p> <p>p2:</p> <p>< 0 > .. digital value is low</p> <p>< 1..n > .. digital value is high</p>
/proc/stw_io/PIN2	p1	-	<p>This user space interface reads on digital output PIN.</p> <p>p1:</p> <p>< 0 > .. sets digital value to low level</p> <p>< 1..n > .. sets digital value to high level</p>

6.11 Universal Serial Bus

This chapter describes the possibilities of how to access the universal serial bus (USB) interface of the TC3G.

High Level Hardware Access

Path	Input information	Output information	Description
/etc/init.d/scripts/mdev	-	-	This is a shell script that adds the mdev as a hotplug to the kernel. The kernel uses the script to scan for USB devices during startup.
/usr/local/sbin/usb_automount.sh	p1 p2	-	This script is called by /sbin/mdev if a storage device is created or removed. Storage devices are directly mounted by the usb_automount.sh to /mnt/usbdisk/sd[a..z][1..9] e.g. /mnt/usbdisk/sda1 p1: < device name > e.g. /dev/sd[a-z][1-9] p2: < mount path > . e.g. /mnt/usbdisk/<device_name>
/etc/init.d/scripts/usb_user_action	-	-	Use this shell script for actions that must be executed after a storage device is plugged in.



NOTE:

A plugged in USB storage devices will be automatically mounted to /mnt/usbdisk/<device_name>. However, not each USB to Serial adapter is recognized by the linux kernel. The used USB storage devices must be FAT32 formatted.

Low Level Hardware Access

Path	Input information	Output information	Description
/dev/sd[a..z][1..9]	-	-	The Kernel creates the interface after a USB storage was hotplugged. The user can handle it using a shell script and / or C-Code.
/dev/ttyUSB[0..9]	Write characters to serial out to another device e.g. via picocom	Read characters from serial out (use e.g. picocom) transmitted by another device	The Kernel creates this device after a USB to Serial Adapter was hotplugged. The user can handle it using a shell script, C-Code or picocom.

6.12 Beeper

High Level Hardware Access

A summary of common signals could be found under: /usr/local/bin/

Low Level Hardware Access

Path	Input information	Output information	Description
/proc/stw_beep/beep	p1	Read note table	This user space interface gives the opportunity to create sound signals. p1: < 0 ... 16000 > Frequency in Hz

Note table: Recommended by STW

Note	Hertz	Note	Hertz	Note	Hertz
C'	264	C''	528	C'''	1056
D flat'	282	D flat''	564		
D'	297	D''	594		
E flat'	316	E flat''	632		
E'	330	E''	660		
F'	352	F''	704		
G flat'	396	G flat''	742		
G'	422	G''	792		
A flat'	440	A flat''	844		
A'	440	A''	880		
B flat'	475	B flat''	950		
B'	495	B''	990		

Example:

Set C' note to beeper

```
# sudo echo 264 > /proc/stw_beep/beep
```

Set beeper off

```
# sudo echo 0 > /proc/stw_beep/beep
```

6.13 LED

The signal daemon signalizes the current status of the TC3G via LED signal.

Signal daemon configuration file (see "[Signal daemon](#)" on page 116)



NOTE:

The signal daemon is already started during boot up sequence. It is not necessary to start it manually.



NOTE:

The LED can be controlled manually. To be able to control the LED manually, deactivate the signal daemon.

Low Level Hardware Access

Path	Input information	Output information	Description
/sys/class/leds/status_led/brightness	p1	p1	<p>This file is used to switch on the LED in a certain color</p> <p>p1: defines the color of the LED</p> <p>< 0 > = led off < 1 > = red < 2 > = green</p> <p>< 3 > = yellow < 4 > = blue</p> <p>< 5 > = pink</p> <p>< 6 > = cyan < 7 > = white</p>
/sys/class/leds/status_led/trigger	p1	The active mode is marked with squared brackets [p1]	<p>This file is used to define the trigger-mode of the LED</p> <p>p1: defines the trigger mode</p> <p>< none > = led is permanently switched on</p> <p>< timer > = led is switched ON and OFF for a variable time</p> <p>< heartbeat > = led signalizes a heartbeat</p> <p>< default-on > = led is permanently switched on</p>
/sys/class/leds/status_led/delay_on	p1	p1	<p>This file occurs when the timer-mode is active</p> <p>p1: defines how many milliseconds the led is switched ON in the timer-mode</p> <p>default: 1000 ms</p>

Path	Input information	Output information	Description
/sys/class/leds/status_led/delay_off	p1	p1	<p>This file occurs when the timer-mode is active</p> <p>p1: defines how many milliseconds the led is switched OFF in the timer-mode</p> <p>default: 400 ms</p>

Example: Deactivate signal daemon to be started by boot up sequence

1. Deactivating the ysignalld means to rename his configuration file like the following

```
# mv /etc/init.d/_ysignalld.config /etc/init.d/ysignalld.config
```

2. Reboot the system

```
# reboot
```

Example: How to use the low level interface

```
# echo "none" > /sys/class/leds/status_led/trigger
# echo "2" > /sys/class/leds/status_led/brightness
```



A cyan led signals flashing similar to a heartbeat can be achieved with the following configuration:

```
# echo "heartbeat" > /sys/class/leds/status_led/trigger
# echo "6" > /sys/class/leds/status_led/brightness
```



A blue led signals flashing with the default timer frequency can be achieved with the following configuration:

```
# echo "timer" > /sys/class/leds/status_led/trigger (default ON: 1000 ms, default OFF: 400 ms)
# echo "4" > /sys/class/leds/status_led/brightness
```



A pink led signals flashing with a frequency of 3.33 Hz can be achieved with the following configuration:

```
# echo "timer" > /sys/class/leds/status_led/trigger
# echo "300" > /sys/class/leds/status_led/delay_on
# echo "300" > /sys/class/leds/status_led/delay_off
# echo "4" > /sys/class/leds/status_led/brightness
```



7 Teleservice Application Framework

7.1 Overview

STW provides two different ways for application development on the module. The first way is simply for the developer to write applications from scratch, using the board support package and the sample code that is provided. Also as any open source applications or tools that make sense for that project can be used. The linux operating system provides a very flexible environment for this type of approach and the developer can choose any language or set of tools in the wide world of linux development.

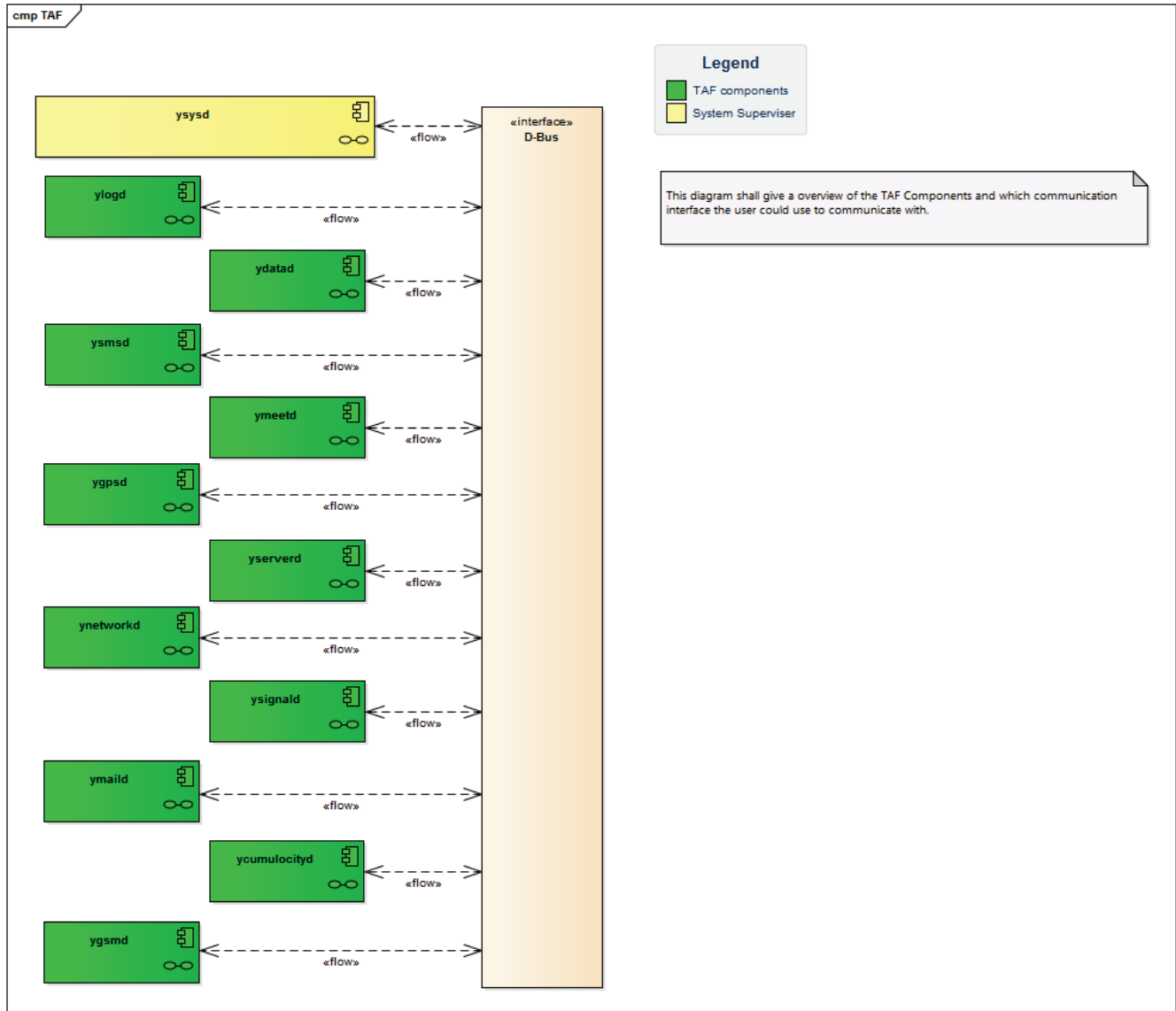
A second way for development is provided through a hierarchy of components that STW has developed and will continue developing around the Teleservice concept. The foundation of this hierarchy is the Telematics Application Framework (TAF). This framework consists of a set of linux daemons and utilities that provide useful functions for the developer.

A typical teleservice application consists of a number of system components that are part of the dedicated teleservice module (e.g. TC3G) on the one hand and external components (eg. ESX, remote computers) on the other hand. See the following examples:

- System components within the teleservice module: RTC, I/Os, GSM-module, GPS-module, filesystem / datalogger, bluetooth, network stacks etc.
- External system components: vehicle controller (ESX, IOX, vehicle sensors and actuators; controllers and components of different brands), remote controllers, servers and PCs.
- Each component can be represented by a software module that provides certain data and services. In order to enable applications to make use of those services an inter-process mechanism is required.

7.2 TAF Components

TAF Components:



7.2.1 System daemon

The system daemon has a special role among the daemons.

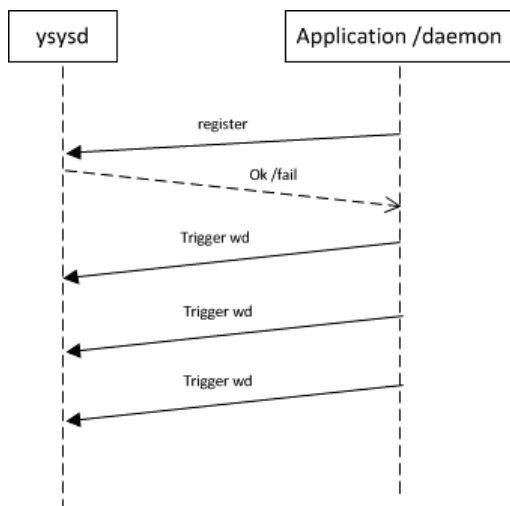
The system daemon is already activated per default.

Daemon location	Configuration file
/usr/local/bin/ysysd	/etc/init.d/ysysd.conf

ysysd

The system daemon's task is to

- trigger the kernel watchdog (which triggers the hardware watchdog).
- offer watchdog functionality to applications.
- applications can register, trigger and cancel the watchdog service via D-Bus calls.
- trigger, supervise and cancel the system shutdown.
- start and stop the shutdown depending on the D+ pin.
- delay the shutdown process according to the settings in the /tmp/sdruntime file.
- send ignition event signals.



Basic configuration of the daemon could be handled in its own configuration file (ysysd.conf).

Relevant sections of the configuration file:

Section: Log file path

```

# The logging information can be written to a log file at the following path
# By default, the path is deactivated because the corresponding line is commented out.
# Max. path length is 255
# If the path is deactivated or no path is set, an error message will be put on stdout
Log_File /var/log/ysysd.log

```

Section: Ignition interface path

```
# The ysysd watches the ignition pin and would start the shutdown process if it is switched off.
# Max. path length is 256
# If the path is deactivated or no path is set, an error message will be put on stdout.
dplus_path /proc/stw_shutdown/dplus
```

Section: Watchdog handling

```
# The hardware watchdog path ensures that the user space scheduling is running properly.
# Max. path length is 255
# If the path is deactivated or no path is set, an error message will be put on stdout.
watchdog_path /dev/watchdog
# Watchdog interval is set in seconds. Min. 0 / max. 0xFFFFFFFF, recommended min. 30s / max. is 100s, default interval is 30s
# If no interval is set, then the default interval of 30s will be set.
watchdog_interval 30
```

Section: Shutdown run time path

```
# The shutdown run time path includes the run time in seconds after shutdown (ignition off). When the ignition pins
# goes off, the controller (TC3G can keep on running for a certain amount of time. User space applications can also
# set the desired time.
# Max. path length 255
# If the path is deactivated or no path is set, an error message will be put on stdout.
runtimefile_path /tmp/sdruntime
```



NOTE:

This file will be set by the boot up script at boot time, but it can be overwritten by applications anytime before the ignition goes off.

Section: Run time interval

```
# The run time interval is in seconds. Min. 0, Max. recommended 10s, default interval 5s
# If no interval is set, the default interval will be set.
run_interval 5
```

Section: DPlus inversion

```
# On some devices it is necessary to invert the dplus pin. Per default, the
# inversion of the pin of off (default value : 0) To activate the inversion,
# set the value to 1
dplus_inv 0
```

7.2.2 Data daemon

Daemon location	Configuration file
/usr/local/bin/ydatad	/etc/init.d/_ydatad.config

How to activate the configuration file of the data daemon

Activate the configuration file of the data daemon, so the data daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ydatad the following way:

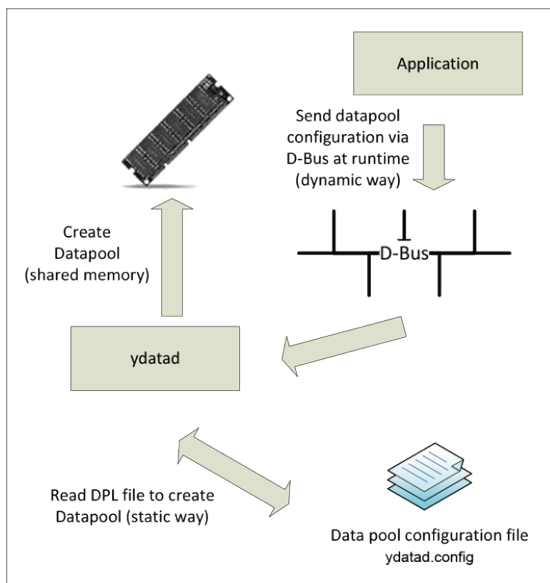
```
# mv /etc/init.d/_ydatad.config /etc/init.d/ydatad.config
```

2. Reboot the system

```
# reboot
```

ydatad

The data daemon provides access to common data pools. It gives applications and processes the opportunity to define, create and delete variable lists via D-Bus commands. In order to achieve quick response times and a generally good performance when working with those variables, writing and reading will be handled by the "shared memory" mechanism.



Basic configuration of the daemon can be handled in its own configuration file (ydatad.config).

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default path is deactivated.
# Max. path length is 255
```

```
# If the path is deactivated or no path is set, an error message will be put on stdout.
Log_File /var/log/ydatad.log
```

Section: Supervising the data daemon

```
# The data daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. interval is 0, that deactivates the watch dog, max. interval is 0x7FFFFFFF,
# recommended min. 120s / max. interval is 1000s
# If no interval is set, the watchdog will be deactivated (set to 0).
watchdog_interval 120
# The system reboot path, the maximum path length is 255. After the time has elapsed
# without triggering the watchdog,
# the ysysd kills the ydatad and executes the "CMDOnWatchdog" command.
# If no path is set, the default 'signal01.beep' will be set.
CMDOnWatchdog /sbin/reboot
```

Section: Variable lists path

```
# The data daemon creates temporary variable list files during operation in the following
# path.
# If no path is set, the default path "/var/run/taf/Datapools/" will be always set by the
# daemon.
# Max. path length 255
BaseFolderTmp /var/run/taf/Datapools/
```

Section: Data pools path

```
# The data daemon provides also the possibility to configure data pools automatically at
# startup.
# In that case all data pools which shall be configured at startup must be placed into the
# following path.
# If no path is set, default path "/opt/taf/Datapools/" will be always set by the daemon.
# Max. path length 255
BaseFolderStatic /opt/taf/Datapools/
```

Section: Set data pool configuration mode

```
# 1 / TRUE: Data pool configuration at startup (static data pools) + dynamic data pool
# configuration
# 0 / FALSE: Only dynamic data pool configuration is possible
# If no value is set, default (FALSE) will be always set by the daemon.
StartUpConfiguration 1
```

7.2.3 Logger daemon

Daemon location	Configuration file
/usr/local/bin/ylogd	/etc/init.d/_ylogd.config

How to activate the configuration file of the logger daemon

Activate the configuration file of the logger daemon, so the logger daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ydatad the following way:

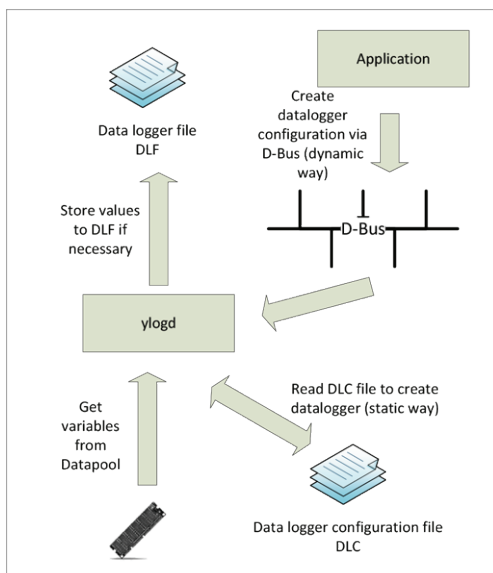
```
# mv /etc/init.d/_ylogd.config /etc/init.d/ylogd.config
```

2. Reboot the system

```
# reboot
```

ylogd

The data logging daemon "ylogd" offers the opportunity to create dynamic or static logging jobs. Applications can add variables to the job, which should be logged and define trigger conditions on those variables that make the daemon write a new data record into the log-file.



Basic configuration of the daemon can be handled in its own configuration file (ylogd.config).

Relevant sections of the configuration file:

Section: Log-file path

```
# The logging information can be written to a log-file at the following path
# By default, the path is deactivated because the corresponding line is commented out.
# Max. path length is 255
# If the path is deactivated or no path is set, an error message will be put on stdout.
Log_File /var/log/ylogd.log
```

Section: Supervising the data daemon

```
# The data logging daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. interval is 0 that deactivates the watch dog, max. interval is 0x7FFFFFFF,
# recommended min. 60s / max. interval is 1000s
# If no interval is set, the watch dog will be deactivated (set to 0).
watchdog_interval 120
# After the time has elapsed without triggering the watchdog, the ysysd kills the ylogd and
# executes the "CMDOnWatchdog" command.
# If no path is set, the default 'signal01.beep' will be set.
CMDOnWatchdog /sbin/reboot
```

Section: Data logger configuration files path

```
# The data logging daemon creates temporary data logger configuration files during
# operation in the following path.
# Max. path length 255
# If no path is set, the default path "/var/run/taf/DLC/" will be set.
DlcFolderTmp /var/run/taf/DLC/
```

Section: Data logger path

```
# The data logging daemon provides also the possibility to configure data logger
# automatically at startup.
# In that case all data logger which shall be configured at startup must be placed in the
# following path.
# Max. path length 255
# If no path is set, the default path "/opt/taf/DLC/" will be set.
DlcFolderStatic /opt/taf/DLC/
```

Section: Set data logger configuration mode

```
# 1 / TRUE: Data pool configuration at startup (static data pools) + dynamic data pool
# configuration
# 0 / FALSE: Only dynamic data pool configuration is possible
# If no value is set, the default value (FALSE) will be set by the daemon.
StartUpConfiguration 1
```

Section: Result file path

```
# Result path for the logs of the data log daemon. Result file extension is ".dlf".
# Max. path length 255
# If no path is set, the default path "/mnt/ dataflash/dlf/" will be set and an error
# message will be put on stdout.
DlfFolder /mnt/dataflash/dlf/
```

Section: Synchronization interval

```
# Interval for writing the logged values from RAM to file.
# Min. and default interval is 500ms, max. interval is 0xFFFFFFFF, recommended max.
# interval is 30000ms
# If no interval is set, the default interval of 500ms will be set.
SyncInterval 30000
```


Section: Full dataset interval

```
# Interval for writing a complete dataset to the datalogger file. The interval value is set
in bytes
# Per default, the mechanism is disabled because it's only necessary in combination with
the yserverd
# If "0" is set, the mechanism is disabled (default)
# If the value is > 0 && < 10000, the value will be set to 10000
# If the value is > 1048576 bytes (1024 * 1024 --> 1 MB), the vale will be set to 1 MB
FullDatasetInt 0
```

7.2.4 GPS daemon

Daemon location	Configuration file
/usr/local/bin/ygpsd	/etc/init.d/_ygpsd.config

How to activate the configuration file of the GPS daemon

Activate the configuration file of the GPS daemon, so the GPS daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ygpsd the following way:

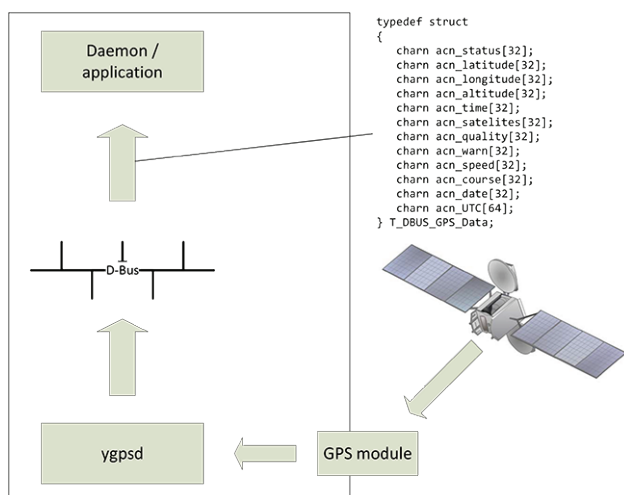
```
# mv /etc/init.d/_ygpsd.config /etc/init.d/ygpsd.config
```

2. Reboot the system

```
# reboot
```

ygpsd

The GPS daemon provides position, date, and time information from the GPS module.



Basic configuration of the daemon could be handled in its own configuration file (ygpsd.config).

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default, the path is deactivated because the corresponding line is commented out.
# Max. path length is 255
# If the path is deactivated or no path is set, an error message will be put on stdout.
Log_File /var/log/ygpsd.log
```

Section: Supervising the GPS daemon

```
# The GPS daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. interval is 0 that deactivates the watch dog, max. interval is 0x7FFFFFFF,
recommended min. 120s / max. interval is 1000s
# If no interval is set, watchdog will be deactivated (set to 0).
watchdog_interval 120
# System reboot path, max. path length 255. After the time has elapsed without triggering
the watchdog,
# the ysysd kills the ygpsd and executes the "CMDOnWatchdog" command.
# If no path is set, the default signal01.beep will be set.
CMDOnWatchdog /sbin/reboot
```

Section: GPS receiver interface path

```
# TCP port for gpsd usage, min. > 0, max. 65535.
# If port is > max, the variable will be set to 0 and default port of gpsd will be set
gps_tcp_port 5894
```

Section: Synchronize system time / date

```
# The GPS daemon can set the linux (system time) and / or the hardware clock (RTC chip)
automatically
# when ever it gets its valid time from the GPS satellites.
# Activate the two functions by uncommenting the following line / deactivate this by
commenting or deleting the line.
auto_set_system_time
auto_set_rtc_time

# "RTC_path" let you select the device that shall be used as the hwclock
# Default path is "/dev/rtc0"
RTC_path /dev/rtc0

# The ygpsd updates the hwclock in an interval of "auto_set_rtc_time_interval" [seconds]
# Default is 21600 (= 6 hours.)
auto_set_rtc_time_interval 21600
```



NOTE:

Many applications cannot handle abrupt time shifts while they are running. Therefore, the system time is only set in case, when the time is invalid. A invalid time is assumed, when the year is < 2010. In this case, the TC3G is restarted to get the real time again.

7.2.5 GSM daemon

Daemon location	Configuration file
/usr/local/bin/ygsmd	/etc/init.d/ygsmd.config

How to activate/deactivate the GSM daemon:

The GSM daemon is active by default. In order to deactivate the daemon:

1. Rename the configuration file the following way:

```
# mv /etc/init.d/ygsmd.config /etc/init.d/_ygsmd.config
```

2. Reboot the system:

```
# reboot
```

ygsmd

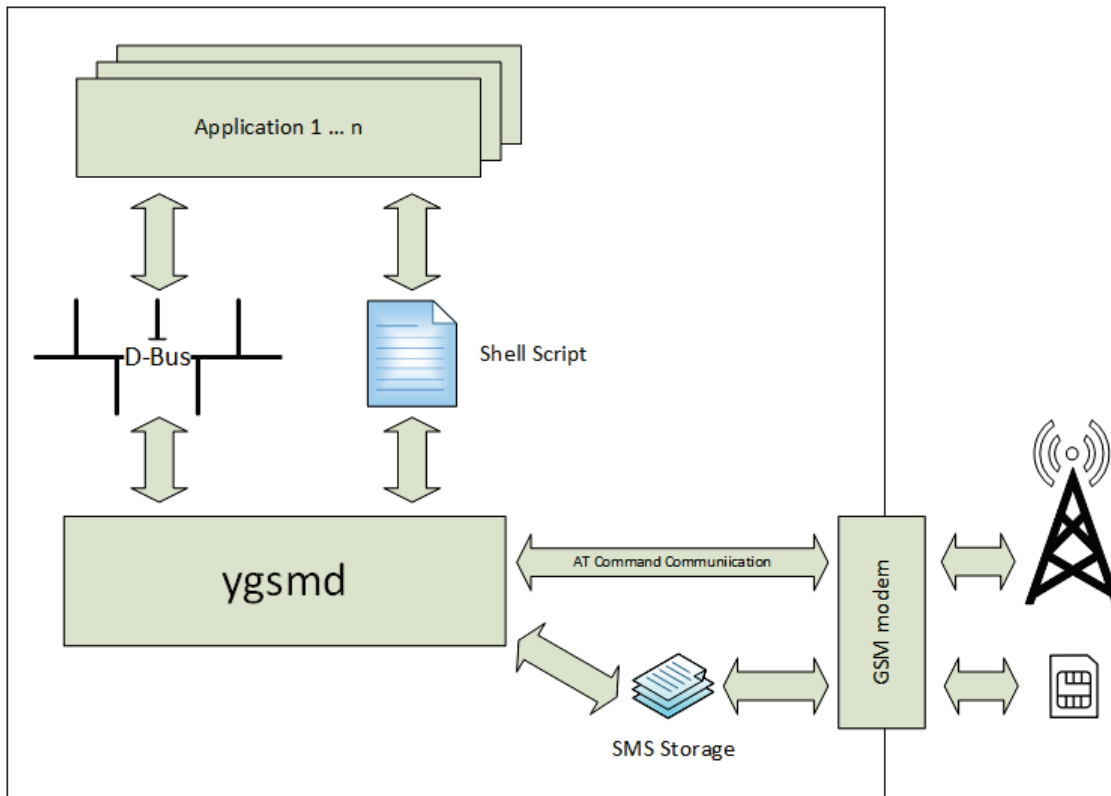
The GSM daemon is part of the STW Teleservice Application Framework. Its purpose is to communicate with the internal modem of the device.

The daemon provides all relevant information of the modem, SIM card and provider (e.g. signal quality, IMEI, IMSI, APN,...).



NOTE:

No other application should be talking to the modem of the device.



For other applications to access the data, the daemon deploys a GSM structure via shell script and via D-Bus.

T_DBUS_GSM_Data

```

typedef struct
{
    charn acn_PhoneNo[32];           // Phone number of the SIM card
    charn acn_SIMSerial[32];         // Serial number of the SIM card
    charn acn_SIMState[32];          // SIM card status
    charn acn_SIMMCC[32];            // Mobile Country Code of the SIM card
    charn acn_SIMMNC[32];            // Mobile Network Code of the SIM card
    charn acn_SIMCarrier[32];         // Provider of the SIM card
    charn acn_SIMAPN[32];            // Access Point Name of the SIM card
    charn acn_SIMUser[32];           // SIM user name
    charn acn_SIMPasswd[32];         // SIM password
    charn acn_IMEI[32];              // International Mobile Equipment Identity (IMEI)
    charn acn_IMSI[32];              // International Mobile Subscriber Identity (IMEI)
    charn acn_RegStatus[32];          // Network Registration Status
    charn acn_SigQuality[32];         // Signal Quality
    charn acn_SigQualitydBm[32];     // Signal Quality in dBm
    charn acn_AccessTec[32];          // Access Technology (e.g.3G)
    charn acn_MCC[32];               // Mobile Country Code
    charn acn_MNC[32];               // Mobile Network Code
    charn acn_LAC[32];               // Location Area Code
    charn acn_CellID[32];            // Cell ID
} T_DBUS_GSM_Data;
  
```

Shell Script /tmp/GSM.info

```

# GSM info structure:

PhoneNo="+4X1X5X0X4X1X"
SIMSerial="8X4X0X0X0X6X1X5X5X"
SIMState="5"
  
```

```
SIMCarrier="T-Mobile Internet"
SIMMCC="262"
SIMMNC="01"
SIMAPN="internet.xxxxxxxxxx.de"
SIMUser="xxx"
SIMPasswd="xxx"
IMEI="3X9X9X0X4X3X0X8"
IMSI="2X2X1X2X6X8X0X8"
RegState="1"
SigQuality="OK"
SigQualitydBm="-87"
AccessTec="3G"
MCC="262"
MNC="01"
LAC="1X9X1"
CellID="6X0X8"
LastUpdate="0X.1X.1X_12:06:45"
```

Basic configuration of the daemon could be handled in its own configuration file (ygsmd.config).

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default path is deactivated by commented the line out.
# Max. path length is 256
# If no path is set or deactivated, error message will be put on stdout.
LOG_FILE /var/log/ygsmd.log
```

Section: Debug mode

```
# The daemon is able to print detailed debug messages when requested.
# Depending on the LOG_FILE variable, the messages will either be printed to
# stdout or directly into the specified log file.
# 0 = deactivate extended debug messages
# 1 = activate extended debug messages
DEBUG_MODE 1
```

Section: Watchdog interval

```
# Signal daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "WD INTERVAL".
# Min. interval is 120s, max. interval is 0x7FFFFFFF,
# recommended min. 120s / max. interval is 1000s
# If no interval is set, watchdog will be set to 120s.
WD_INTERVAL 120
```

Section: Command on Watchdog

```
# System reboot path, max. path length 256. After the time is left without
# triggering the watchdog, the ysysd would kill the daemon and execute the
# "CMD ON WD" command.
# If no path is set, default (signal01.beep) will be set.
CMD_ON_WD /sbin/reboot
```

Section: Modem path

```
# Modem path - select modem interface the daemon should use
# default: /dev/mux1
```

```
MODEM_PATH /dev/mux1
```

Section: Update interval

```
# Update interval - select when the AT commands should be updated
# MIN = 30 seconds, MAX = 65536 seconds, default: 300 seconds
UPDATE_INTERVAL 300
```

Section: User XML file path

```
# Custom XML file path - select path for customer specific APN settings
# default /mnt/dataflash/stw/modem/apns-conf.xml
USER_XML_FILE /mnt/dataflash/stw/modem/apns-conf.xml
```

Section: SMS outbox path

```
# In case the system shuts down before we could put all SMS in the outgoing folder,
# save the out going SMS to this file.
# This file will be loaded at the next system start.
# Max. path length 255
# If no path is set, default is used: "/mnt/dataflash/stw/modem/sms.outbox"
SMS_OUTBOX /mnt/dataflash/stw/modem/sms.outbox
```

Section: SMS buffer size

```
# Received SMS are stored in the GSM modem when received.
# Set the number of SMS slots that we should supervise through the daemon.
# Min. and default buffer size is 10, max. buffer size 0x7FFFFFFF, recommended max. is 10.
# If buffer size is less than 10, default size will be set and an error message will be
# put on stdout.
SMS_BUFFER_SIZE 10
```

7.2.6 Network daemon

Daemon location	Configuration file
/usr/local/bin/ynetworkd	/etc/init.d/_ynetworkd.config

How to activate the configuration file of the network daemon:

Activate the configuration file of the network daemon, so the network daemon is started during the boot up sequence automatically.

1. Rename the configuration file ynetworkd the following way:

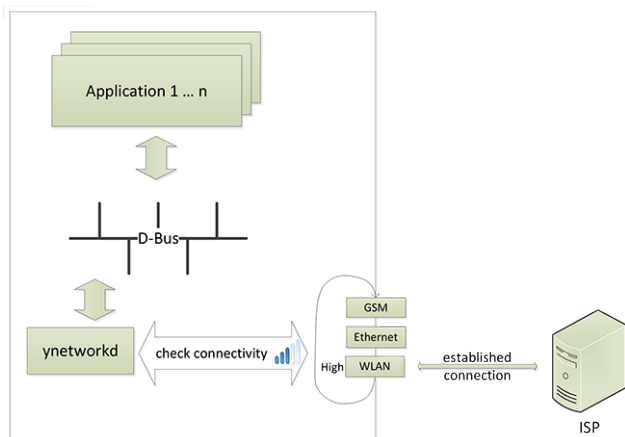
```
# mv /etc/init.d/_ynetworkd.config /etc/init.d/ynetworkd.config
```

2. Reboot the system:

```
# reboot
```

ynetworkd

The ynetworkd daemon provides functions to manage the internet connectivity automatically. Via D-Bus the ynetworkd can be switched from "enabled connectivity" to "disabled connectivity". It is also possible to get the actual connected interface.



To use the ynetworkd, it is necessary to configure the configuration file of the ynetworkd (ynetworkd.config). This file configures the behavior of the network daemon, which is responsible for monitoring all network interfaces, such as ETH, WLAN, and PPP (via BT or GSM). The daemon tries to connect to the internet, depending on the priority of the interface. If it is possible to connect to a higher-prior interface, then the current connection is cut and a connection to an interface with the higher priority is established.

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default, the path is deactivated because the corresponding line is commented out.
# Max. path length is 255
# If no path is set or deactivated, error message will be put on stdout.
Log_File /var/log/ynetworkd.log
```

Section: Supervising the network daemon

```
# Network daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. interval and default is 120s, max. interval is 0x7FFFFFFFs, recommended max. is 1000s
# If no interval is set, watchdog will be set to default (120s).
watchdog_interval 120
# System reboot path, max. path length 255. After the time has elapsed without triggering
the watchdog,
# the ysysd would kill the ynetworkd and execute the "CMDOnWatchdog" command.
# If no path is set, default (signal01.beep) will be set.
CMDOnWatchdog /sbin/reboot
```

Section: Connectivity

```
# Activate 1 / deactivate 0 connectivity at startup, default is 0.
# If the value is set to zero, the connectivity can be activated via D-Bus signal, see TAF
library.
# If value unequal 0 / 1 or not set, default will be set and an error message will be put
on stdout.
activate_connectivity 1
```

Section: Matrix of supervised interfaces

```
# With this option you can select the interfaces that shall be supervised by ynetworkd.
# Wrong interfaces will be ignored.
# Binary coded value:
# Ethernet 0100 --> 4
# WLAN      0010 --> 2
# PPP       0001 --> 1
# For example, if you want to activate all interfaces, your value is 7 (4 + 2 + 1).
# If you want to activate ETH and PPP, your value is 5 (4 + 1).
# Min. value is 1, max. value and default is 7
# If no value is set, a line is deleted, or the value is not in range, then the default
will be set and an error message
# will be put on stdout.
interface_matrix 5
```

Section: Matrix of priority

```
# The daemon will automatically try to connect with the highest prior interface.
# If it isn't possible, the daemon will switch to the next lower prior interface and so on
...
# H = High, M = Middle, L = LOW
# Min. value and default is 1, max. value is 6
# If no value is set, line is deleted or value not in range, default will be set and an
error message
# will be put on stdout.
# ETH   WLAN   PPP   VALUE
# H     M      L     1
# H     L      M     2
# M     H      L     3
# L     H      M     4
# M     L      H     5
# L     M      H     6
priority_matrix 1
```

Section: Network verification

```
# The ynetworkd daemon supervises the desired interface status periodically.
# Every "verify_interface_status_interval" seconds.
# Min. interval and default 5s, max. interval is 0x7FFFFFFFs, recommended max. is 5000s
```



```
# If interval is less than 5s, default will be set.
verify_interface_status_interval 5

# Select type of verification
# 1 --> Ping
# 2 --> Check if specific file is available (get via http)
# If no verification type is selected, an error message will be put on stdout. Daemon will
not be started!
# If value is not in range, 2 will be set.
verify_interface_method 2

# ----- Ping -----
# In order to check if a connection to the network is established,
# the ynetworkd does ping to server that can be set here in dotted notation
(123.123.123.123) or in URL notation (www.google.de)
# Max. URL length 255
# If no server address is set, an error message will be put on stdout. Daemon will not be
started!
PingServerAddr www.google.de
# Ping timeout in seconds until ping succeed.
# Min. timeout and default is 15s, max. timeout 0xFFFFs)
# If no timeout is set, default will be set.
ping_timeout 15

# ----- File supervision -----
# For file supervision modus it's necessary to set the server and the file name
# Max. URL length 255
# If no path is set, an error message will be put to stdout. Daemon will not be started!
FileServerAddr www.google.de
# File name of the file to verify.
# Max. file name length 255
# If no file name is set, an error message will be put to stdout. Daemon will not be
started!
FileName index.html
# Connection port number of the FileServerAddr
# Min. port number 0, max. port number 0xFFFF
# If no port number is set, an error message will be put to stdout. Daemon will not be
started!
FilePort 80
# Timeout until file request exits.
# Min. timeout and default 10s, max. timeout 0xFFFFs
# If no timeout is set, default will be set.
FileTimeout 10
```

Section: GSM settings

```
# To start and stop connections via GPRS, the ynetworkd uses standard linux tools.
# Set the scripts, which have to run to establish and to cut a GPRS connection.
# Start GPRS connection
# Max. path length 255
# If no path is set, an error message will be put to stdout. Daemon will not be started!
pppd_start_script /etc/ppp/ppp-start
# Stop GPRS connection
# Max. path length 255
# If no path is set, default command "killall pppd" will be set and an error message will
be put to stdout.
# Daemon will not be started!
pppd_stop_script /etc/ppp/ppp-stop

# Mux_mode_enabled activates the usage of the gsmMuxd within the ynetworkd
# Per default the "use gsmMuxd modus" is deactivated and so the value is set to 0.
# In case of using the modem with the gsmMuxd, the mux mode has to be enabled (1).
# In that case one mux channel is used for data traffic
# and <gsm_tty_path> is used for sending AT commands to the modem.
# Every value greater than 1 will be automatically set to the default value (0)
mux_mode_enabled 0
```


WARNING:

ygsmd daemon has to be inactive when mux_mode_enabled=1

```
# The virtual port of the GSM modem which is reserved for AT command communication.
# Max. path length 255
# If no path is set, the default path "/dev/mux1" will be used!
gsm_tty_path /dev/mux1

# In case the modem is no longer accessible there must be a way to reset the modem.
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
modem_restart_script /etc/init.d/scripts/gsm restart

# Time which the ynetworkd waits before it verifies the connection.
# Note: Time will be doubled internally
# Min. search time and default is 5, max. search time 0xFFFFs
# If search time is less than 5, default search time will be set.
ppp_net_search_time 10

# After trying NO_NETWORK_REG_COUNTER times, the GSM modem will restart.
# Min. number of tries and default is 3, max. number of tries is 0xFFFF
# If the tries number is less than 3, default tries number will be set.
NO_NETWORK_REG_COUNTER 3
```

Section: WLAN settings

```
# To start and stop connections via WLAN, the ynetworkd uses standard linux tools.
# Set the scripts that have to run to establish and to finish a connection via WLAN.
# Start WLAN connection, i.e. scan for hosts and clients
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
wlan_start_script /etc/init.d/scripts/wlan_daemon
# Connect to a host or client
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
wlan_connect_script /etc/init.d/scripts/wlan_daemon
# Disconnect WLAN connection from host or client
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
wlan_disconnect_script /etc/init.d/scripts/wlan_daemon

# Time that the ynetworkd waits before it verifies the connection.
# Note: Time will be doubled internally.
# Min. search time and default is 5s, max. search time 0xFFFFs
# If search time is less than 5s, default will be set.
wlan_net_search_time 5

# WLAN tx power:
# 1: 10 dBm --> 10 mW
# 2: 15 dBm --> 31 mW
# 3: 20 dBm --> 100 mW
# >3 and default is 15 dBm
wlan_tx_power 2
```

Section: Ethernet settings

```
# To start and stop connections via ETH, the ynetworkd uses standard linux tools.
# Set the scripts that have to run to establish and to finish a connection via ETH.
# For static usage:
```

```
# Start eth0 connection with lan_static_networkd
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
eth_start_script /sbin/ifup eth0=lan_static_networkd
# Stop eth0 connection with lan_static_networkd
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
eth_stop_script /sbin/ifdown eth0=lan_static_networkd
# For dynamic usage:
# Start eth0 connection with dhcp
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
eth_start_script /sbin/ifup eth0=dhcp
# Stop eth0 connection with dhcp
# Max. path length 255
# If no path is set, an error message will be put on stdout. Daemon will not be started!
eth_stop_script /sbin/ifdown eth0=dhcp

# Time which the ynetworkd waits before it verifies the connection.
# Note: Time will be doubled internally.
# Min. search time and default is 2s, max. search path is 0xFFFFs
# If search time is less 2s, default will be set.
eth_net_search_time 3
```

7.2.7 Server daemon

Daemon location	Configuration file
/usr/local/bin/yserverd	/etc/init.d/_yserverd.config

How to activate the configuration file of the server daemon

Activate the configuration file of the server daemon, so the server daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ydatad the following way:

```
# mv /etc/init.d/_yserverd.config /etc/init.d/yserverd.config
```

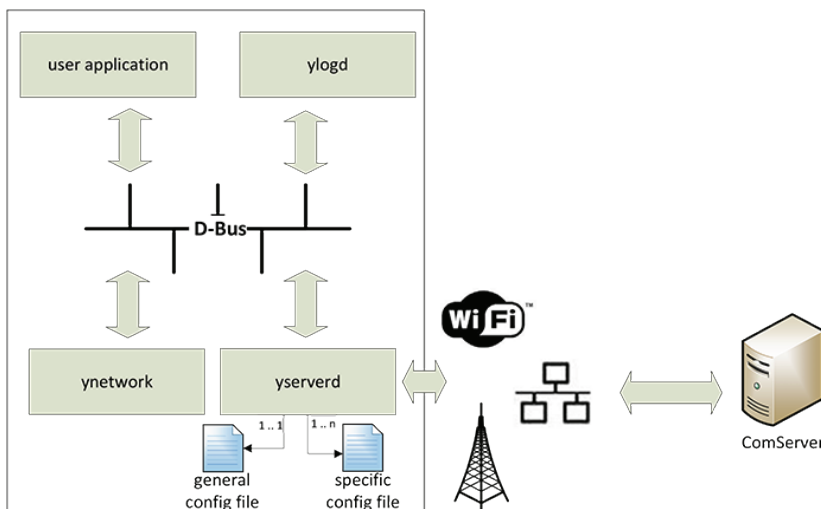
2. Reboot the system

```
# reboot
```

yserverd

The server daemon provides functions to manage the communication with an external server automatically. The server daemon communicates with an external server (like ComServer).

The yserverd uses one of the interfaces selected in the ynetworkd. In a periodic interval, the server daemon requests a specified file from the server. This file can be configured in the yserverd specific configuration file. Via HTTP the file is requested and afterwards interpreted. All known commands are send to the corresponding daemons. After the command was completed, it's also possible to send a reply to the external server.



The server daemon is configured with two configuration files:

- The general configuration file (/etc/init.d/yserverd.config) is used for all started daemon instances. It includes the default paths like watchdog interval, reboot command and path to specific configuration files.
- The specific configuration file (/opt/yserverd/configuration/<CONFIG_NAME>.config) that includes communication specific settings, like communication URL, port, protocol. Properties for the specific configuration file:

- Syntax for the name CONFIG_NAME: [a..z][A..Z][0..9][_ -]
- Limitation of the configuration file: The memory size of the system memory where the configuration file will be saved limits the size of the configuration files.



NOTE:

For each specific configuration file the server daemon creates a new identical instance of himself parametrized by it.

Basic configuration of the daemon could be handled in its own general configuration file.

Relevant sections of the general configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# The path is deactivated by default
# Max. path length is 512
# If the path is deactivated or no path is set, an error message will be put on stdout.
Log_File /var/log/yserverd.log
```

Section: Supervising the data daemon

```
# Server daemon can be supervised by a watchdog (provided by the ysys daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. and default watchdog interval is 100s, max. watchdog interval is 0x7FFFFFFFs,
# recommended max. is 1000s
# If no watchdog is set, default will be set (100s).
watchdog_interval 100
# System reboot path, max. path length 512. After the time has elapsed without triggering
# the watchdog,
# the ysysd would kill the yserverd and execute the "CMDOnWatchdog" command.
# If no path is set, default (signal01.beep) will be set.
CMDOnWatchdog /sbin/reboot
```

Section: Specific configuration files path

```
# Path to the specific configuration files
# Max. path length is 512
# If no path is set, parent will still work!
ConfigFileFolderPath /opt/yserverd/configuration
```

Relevant sections of the specific configuration file:

Section: Exchange directory for server communication

```
# Server exchange directory path. The yserverd copies the *.dlf to the given directory.
# Max. path length 255
# If no path is set, default "/mnt/ dataflash/dlf/" will be set and an error message will
# be put on stdout.
ServerExchangeFolder /mnt/dataflash/yserverd/
```

Section: Communication server URL

```
# Server info: URL, Port, path to server command file on server
# Example: com_server_url 192.168.2.14
# Max. path is 512
# If no path is set, an error message will be put on stdout. Specific connection will not
```

```
be started!
com_server_url teleservice.specific-url.de
```

Section: Communication protocol download path

```
# Path for Nextjob.job download on the ComServer
# Use the string <client_id> as a placeholder for the real client ID. The daemon will
# replace the string with the real ID.
# Max. path length 512
# If no path is set, an error message will be put on stdout. The specified connection will
# not be started!
com_server_path_get ec2c/<client_id>/ToDevice/
```

Section: Communication protocol upload path

```
# Path for Nextjob.job upload on the ComServer
# Use the string <client_id> as a placeholder for the real client ID. The daemon will
# replace the string with the real ID.
# Max. path length 512
# If no path is set, an error message will be put on stdout. The specified connection will
# not be started!
com_server_path_put ec2c/<client_id>
```

Section: Communication port number

```
# Port number
# Min. port number is 0, max port number is 0xFFFF, default port number is 80
# If no port number is set, default port will be set (80).
port_no 8080
```



WARNING:

Use only characters [0..9] for the port. Using other characters forces a system fault.

Section: Basic http authentication

```
# User name and corresponding password for the HTTP basic authentication is necessary to
# access the server
# The maximum length for the user name and also for the password is 30 characters.
http_user test
http_passwd testtest
```

Section: Protocol file name from server

```
# Command file name from Com-Server
# Min. command length is 1, max. command length is 512
# If no command is set, an error message will be put on stdout. The specific connection
# will not be started!
srv_cmd_file NextJob.job
```

Section: User file name from device

```
# Command file name from device
# Min. command length is 1, max. command length is 512
# If no command is set, an error message will be put on stdout. The specific connection
```

```
will not be started!
usr_cmd_file Result
```

Section: TC3G device name

```
# Client_ID: Optional <hostname> will be set by the yserverd, e.g. TC3G-<serial_num>_0...n
# Min. client id length is 1, max. client id length is 512, default client name is the TC3G
hostname
# If no client name is set, default will be set (hostname).
client_id 101011421008
```

Section: Default interval for Nextjob.job checking

```
# Trigger interval for the NextJob.job file sets the connection_cycle_timeout, be
overwritten by the interval of the NextJob.job file,
# in case there is one set.
# Min. timeout is 0s, max. timeout is 0xFFFFFFFFs
# If no timeout is set, an error message will be put on stdout. Specific connection will
not be started!
connection_cycle_timeout 360
```

Section: Data logger transmission size

```
# This section sets the maximum data logger transmission size in bytes before the '.end'
file is created.
# The default value is 0, this means that the '.end' file is created after the complete
transmission of the data logger file.
# Min. and default size is 0, max. size is 0xFFFFFFFF
# If no size is set, the default value will be set.
DLTS 0
```

Section: Maximum result file size

```
# Sets the maximum size in bytes for one single result file.
# After reaching the max. size of a result file, a new one is created --> result.001,
result.002 ...
# Min. size is 0, max. size is 0xFFFFFFFF
# The size will be set to 4096, if value of the size is less than 4096.
# If no size will be set, 8192 will be set.
max_result_file_size 8192
```



NOTE:

For more information about the NextJob.job mechanism contact the Sensor-Technik Wiedemann GmbH support.

7.2.8 Signal daemon

Daemon location	Configuration file
/usr/local/bin/ysignald	/etc/init.d/ysignald.config



NOTE:

The signal daemon can only be used, when a LED is available.

How to activate the configuration file of the signal daemon

Activate the configuration file of the signal daemon, so the signal daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ysignald the following way:

```
# mv /etc/init.d/_ysignald.config /etc/init.d/ysignald.config
```

2. Reboot the system

```
# reboot
```

ysignald

The signal daemon signalizes the current status of the module over led signal.



The following states can be displayed:

State of the module	LED color	Description
Power ON	yellow	This state is active when the TC3G has booted and none of the other states are active.
GPS	blue	This state is active when ygpsd is activated and a valid NMEA string is received and the internet-state is not active.
Internet	pink	This state is active when ynetworkd is activated and a valid connection to the internet exists and the GPS-state is not active. This is also verified by a 'ping' signal or a file download.
GPS + Internet	green	This state is active when the 'GPS' state and the 'Internet' state are both active.
CAN traffic	flashing LED, without changing its color	This state is active when there is traffic on CAN0 or CAN1.
Error	blink code initialized by a 'red' LED signal	This state is active when a 'Goodbye' signal from a daemon is received or when a valid entry in the /tmp/LED_Error file exists.
User	blink code initialized by a pause of the LED signal	This state is active when a valid entry in the /tmp/LED_User file exists.



NOTE:

The 'CAN traffic' state will not be signaled when the 'Error' state is active.
The 'User' state is signaled parallel to all other states.

The signal daemon checks if the variant supports:

- GPS
- GSM
- WLAN
- BT

If the variant does not support GPS, then the 'GPS + Internet' state will never be reached. Therefore, the color of the 'Internet' state is set to 'green'.



NOTE:

If the LED signal is 'green', all possible states, depending on the variant, are reached.

Low Level Hardware Access

LED	Path	Input information	Output information	Description
User LED	/tmp/LED_User	p1 p2	-	<p>This file includes the parameters of the 'User' states</p> <p>p1: defines the color of the LED</p> <ul style="list-style-type: none"> • < 1 > = red • < 2 > = green • < 3 > = yellow • < 4 > = blue • < 5 > = pink • < 6 > = cyan • < 7 > = white <p>p2: defines how often the signal should flash</p> <ul style="list-style-type: none"> • < 0 > = ON for 2 seconds • < 1 > = flash 1 time • < x > = flash x times
STW LED	/tmp/LED_Error	p1 p2	-	<p>This file includes the parameters for the 'Error' states</p> <p>p1: defines the color of the LED</p> <ul style="list-style-type: none"> • < 2 > = green • < 3 > = yellow • < 4 > = blue • < 5 > = pink • < 6 > = cyan • < 7 > = white <p>p2: defines how often the signal should flash</p> <ul style="list-style-type: none"> • < 1 > = flash 1 time • < x > = flash x times


NOTE:

If the parameters of the LED_User/LED_Error file deviate from the description above, the file will not be taken into account.


WARNING:

The 'LED_Error' file must not be modified by the user. Otherwise, the error codes defined by STW are no longer valid.

These blink codes can speed up the support through STW in case of an error.

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default, the path is deactivated because the corresponding line is commented out.
# Max. path length is 256
# If no path is set or deactivated, an error message will be put on stdout.
Log_File /var/log/ysignald.log
```

Information that can be found in the log file is:

- Does the variant support WLAN, GSM, GPS or BT?
- Which daemon is on the D-BUS?
- Which daemon is no longer on the D-BUS?
- Is there traffic on the CAN?
- Is there a connection to the internet?
- Is a valid GPS signal received?

All information is stored with a time-stamp.

Section: Supervising the signal daemon

```
# Signal daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "watchdog_interval".
# Min. interval is 120s, max. interval is 0x7FFFFFFF,
# recommended min. 120s / max. interval is 1000s
# If no interval is set, watchdog will be set to 120s.
watchdog_interval 120
# System reboot path, max. path length 256. After the time has elapsed without triggering
the watchdog,
# the ysysd would kill the ynetworkd and execute the "CMDOnWatchdog" command.
# If no path is set, default (signal01.beep) will be set.
CMDOnWatchdog /sbin/reboot
```

Section: Using the signal daemon without the TAF components ynetwork and/or ygpsd

```
# This setting makes the 'internet state' independent of the ynetworkd.
# ysignald will only listen to the D-Bus signal: Internet_Connection_State (link (see
"ysignald\_internet\_con\_state" on page 265))
# true = use external internet D-Bus signal
UseExternalInternetState true

# This setting makes the 'GPS state' independent of the ygpsd.
# ysignald will only listen to the D-Bus signal: GPS_Connection_State (link (see
"ysignald\_GPS\_con\_state" on page 267))
# true = use external GPS D-Bus signal
UseExternalGPSState true
```

Section: Monitor the other daemons but do not set/signalize an error if they get killed

```
# The surveillance of single daemons can be deactivated if necessary.
# The signal daemon will still keep track of the other daemons in his
```

```
# log file but there will not be an error set/signalized by the daemon.
# In order to deactivate for example ydatad, ylogd and ymsd use:
DeactivateDaemon ydatad ylogd ymsd
```

Example Signals for Normal Operating Mode

The signal below starts with the "maximum reachable" state (e.g. GPS + Internet). After one second, there is traffic on the CAN bus and the LED starts flashing.



If the GPS signal is lost, the signal switches to the color of the 'Internet' state. When there is no more traffic on the CAN bus the LED stops flashing as well.



Example Signals for the User Mode

If there is a valid entry in the 'LED_User' file, the signal daemon starts showing the 'User' state and the state of the module alternately.

The signal below shows alternating the 'GPS + Internet' state and the 'User' state.

The matching entry in the 'LED_User' file would be: 4 3

```
{ echo "4 3" > /tmp/LED_User }
```



The signal below shows alternating the 'GPS' state and the 'User' state.

The matching entry in the 'LED_User' file would be: 2 2

```
{ echo "2 2" > /tmp/LED_User }
```



This signal shows alternating the 'Error' state and the 'User' state.

The matching entry in the 'LED_Error' file would be: 5 3

The matching entry in the 'LED_User' file would be: 6 0



Example Error Signals

If there is a valid entry in the 'LED_Error' file, the signal daemon switches to the 'Error' state.

The signal below indicates that the error has something to do with the 'GPS' state because it flashes 'blue'.

The matching entry in the 'LED_Error' file would be: 4 2



The signal below indicates that there is something wrong with the 'Internet' state because it flashes 'pink'.

The matching entry in the 'LED_Error' file would be: 5 3



The signal below indicates that there is something wrong and it has nothing to do with the 'Internet' or the 'GPS' state.

The matching entry in the 'LED_Error' file would be: 3 4



NOTE:

It is possible that the module is running in areas without the possibility to receive internet or GPS signals. That is considered as a normal operating condition and not as an error.

Error Signals and Description

LED color	Flashing Frequency	Description
blue	1	The ygpsd daemon is no longer on the D-BUS.
pink	1	The ynetworkd daemon is no longer on the D-BUS.
pink	2	Communication with the modem of the TC3G failed OR creating APN settings failed OR reading APN settings failed.
pink	3	APN settings not valid. (e.g. TC3G could not interpret provider or country code)
yellow	1	One of the other daemons (ysysd, ydatad, ylogd, ysmsd, ymeetd or yserverd) is no longer on the D-BUS.

7.2.9 Mail daemon

Daemon location	Configuration file
/usr/local/bin/ymaild	/etc/init.d/_ymaild.config

How to activate the configuration file of the mail daemon

Activate the configuration file of the mail daemon, so the mail daemon is started during the boot up sequence automatically.

1. Rename the configuration file of the ymaild the following way:

```
# mv /etc/init.d/_ymaild.config /etc/init.d/ymaild.config
```

2. Reboot the system

```
# reboot
```

ymaild

In contrast to other daemons of STW the ymail daemon is written as a shell script. It is based on the mailx e-mail client and grants the developer easy access to the TC3G in order to perform simple tasks.

The ymail daemon has no access to the D-Bus and does not register itself to the ysysd.

If you are using the ymaild the first time, then refer to Setup ymaild for the first time (see "[Setup ymaild for the first time](#)" on page 125).

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# The path is deactivated by default.
# Max. path length is 255
# If no path is set or deactivated, error messages will be put to stdout.
Log_File /var/log/ymaild.log
```

Section: Select e-mail account

```
# Set the mail account that should be used by the daemon
# (has to be defined in mail.account-file)
Active_Account master_account
```

Section: Verify internet connection

```
# Verify a valid internet connection using the network daemon
# true = do verify (default)
# false = do not care
Verify_Internet_Connection true
```

Section: Select mail interval

```
# Interval check for new mails in minutes
# default: 5 min
Check_Mails_Interval 5
```

Section: Select mailbox path

```
# Set path for mailbox
# default path: /mnt/dataflash/stw/mbox
Mail_Box /mnt/dataflash/stw/mbox
```



NOTE:

The mailbox should be located in the NAND flash.

Tasks of the ymaild

get a file from the TC3G

After the user has sent an e-mail to the TC3G requesting a file, the TC3G will send a response mail with the requested file in attachment.

The ymaild will search for the file(s) under the specified directories. All files found will be attached to and described in the response mail.

Syntax: file_name;folder_path

Example: get the apn-setting and the log-file of ysysd

```
<get_file>
apn_setting;/tmp
ysysd.log;/var/log
</get_file>
```

replace a file on the TC3G

The user has the possibility to replace existing files on the TC3G. Those files have to be in the attachment of the e-mail sent to the TC3G.

The ymaild will replace the file(s) in the specified directories. The replaced files will keep the same permissions as before.

Syntax: file_name;folder_path

Example: replace the configuration file of ysignald and ymaild

```
<replace_file>
ysignald.config;/etc
ymaild.config;/etc
</replace_file>
```

add a file to the TC3G

The user has the possibility to add files to the TC3G. Those files have to be in the attachment of the e-mail sent to the TC3G.

The ymaild will add the file(s) to the specified directory with the specified file permissions. The directory must not already exist on the TC3G.

Syntax: file_name;file_permissions;folder_path

Example: add new files to the /tmp directory

```
<add_file>
test.txt;777;/tmp/new_folder/
text.file;111;/tmp/
</add_file>
```

add a user to the mail.list

The user can add other users to the mailing list. Every user on that list is able to get mail from the TC3G.

Syntax: email_address

Example: add mail addresses to mail.list

```
<add_user>
test.user@123.de
sample.email@address.com
</add_user>
```

delete a user from the mail.list

The user can delete other users from the mailing list.

Syntax: email_address

Example: delete mail addresses from mail.list

```
<del_user>
test.user@123.de
sample.email@address.com
</del_user>
```

Important



NOTE:

The subject of the e-mail needs to include the serial number of the device. For example the subject of the mail is TC3G-XXXXXXXXXXXX

Encryption needs to be activated before sending the mail.

The format of the e-mail should be plain text.

Example

This example describes how to get all log files from the /var directory. Also this mail must be sent to a colleague who is not on the mail.list yet..

Mail content:

```
<get_file>
*.log;/var
</get_file>

<add_user>
colleague.email@address.com
</add_user>
```

It is important to choose Option -> Encryption before sending the e-mail.



NOTE:

If another user should get the mail from the TC3G as well, it is necessary to put that user on the CC list when sending the mail to the TC3G.

After sending the mail, the TC3G will respond with an encrypted mail and the desired files attached to it. The content of that mail will be similar to:

```
> <get_file>
> *.log;/var
> </get_file>
>
> <add_user>
> college.email@address.com
> </add_user>
>

Task: <get_file>
File(s) to be searched: *.log.
File(s) found and attached to this email:
    /var/log/ysignald.log
    /var/log/ysysd.log

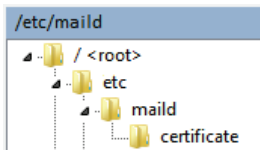
Task: <add_user>
User(s) that should be added: college.email@address.com.
    user college.email@address.com was added to user list

sent from my TC3G
```

7.2.9.1 Setup ymail for the first time

Home Directory - maild

Before using the ymaild, it has to be configured. All but one configuration files can be found in the home directory /etc/maild.



mail.account - file

The configuration of the e-mail account is handled in the file mail.account.

The following setting are for the google mail account: example@gmail.com with the password: 123456.

```
account master_account {
    set folder=imaps://example@imap.gmail.com/INBOX
    set password-example@imap.gmail.com="123456"
    set imap-use-starttls
    set from="TCx <example@gmail.com>"
    set replyto="example@gmail.com"
    set sender="example@gmail.com"
    set smtp-use-starttls
    set ssl-verify=ignore
    set smtp="smtp://smtp.gmail.com:587"
    set smtp-auth="login"
    set smtp-auth-user=example@gmail.com
    set smtp-auth-password="123456"

    # using a signature is not mandatory
    set signature=/etc/maild/signature

    # encryption
    set smime-force-encryption
```

```
set smime-sign-cert=/etc/mailed/certificate/TCxpubl.priv.pem

# include mailing list
set NAIL_EXTRA_RC=/etc/mailed/mail.list
}
```



NOTE:

Renaming the .pem file needs to be handled with care and should be avoided.

mail.list - file

The allowed e-mail addresses are handled in the file mail.list. See description below.

The first user needs to be added by hand. Therefore, the blue highlighted part needs to be replaced.

```
set smime-encrypt-example.address@sensor-technik.de=/etc/mailed/certificate/PCpubl.pem
```

After that, the handling could be performed via e-mail and the tags <add_user> and <delete_user>.

signature - file

The text in the signature file will be added to the end of every mail send by the TC3G.

certificate - folder

The certificate folder includes a configuration file and an executable script.

The objective is to generate certificates for two e-mail accounts, one for the account of the TC3G and the other one for the mail account of the developer.



NOTE:

The ymaild can only handle two certificates. Otherwise the automated mechanisms add user and delete user will not work.

cer.config - file

Is a configuration file and needs to be adapted.

```
# config file for the creation of certificates
countryName=DE
stateName=Some State
localityName=
organizationName=TCx Ltd
organizationalUnitName=
commonName=
emailAddressPC=needs.to@be.set
emailAddressTCx=needs.to@be.set
validDays=9999
keySize=2048
```

create_cert.sh - file

This is the script that needs to be executed in order to generate the necessary certificate files. It is recommended to select a password.

Those files are:

- PCpubl.pem - Public key of the PC that wants to communicate with the TC3G, needs to stay on the device for encryption

- PCpubl.priv.pfx - Public and private key of the PC, needs to be imported under Windows for decryption
- TCxpubl.cer - Public key of the TC3G, needs to be imported under Windows for encryption
- TCxpubl.priv.pem - Public and private key of the TC3G, needs to stay on the device for decryption

Continue with Certificate Handling (see "[Certificate Handling](#)" on page 127).

7.2.9.2 Certificate Handling

After creating the certificates as explained in Setup ymaild for the first time (see "[Setup ymaild for the first time](#)" on page 125), the certificates need to be imported to Windows.

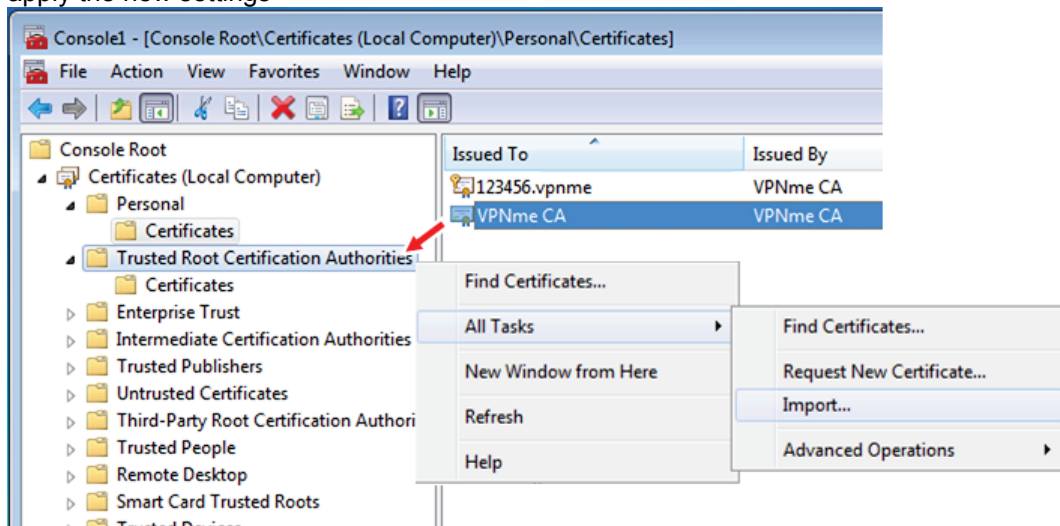
Two e-mail addresses are embedded in these certificates are two e-mail addresses embedded:

1. E-mail address for communicating: The user wants to send and receive encrypted e-mails from the TC1
Best Practice example for a team of developers: Create an e-mail account especially for encrypted communication with the TC1. Everyone will be able to communicate with the TC1 via this particular mail account.
2. E-mail address for observing: Each user, who wants to observe the communication to the TC1, is put on Carbon Copy.
Alternatively, the developer communicating with the TC1 put another developer on CC.

Import the certificates to Windows 7

1. Copy the files PCpubl.priv.pfx and TCxpubl.cer on the windows PC.
2. Execute the PFX file:
 - Enter the password
 - Add the certificate to 'personal certificates'
 - The procedure is finished here, if the user wants only to observe the communication. Continue with step 3, if communication is required
3. Import both certificates to 'trusted root certificates':

- Click Start
 - Click Start Search
 - Type mmc
 - Press ENTER
 - Click Add/Remove Snap-in on the file menu
 - Click certificate Under available snap-ins
 - Click Add
 - Select the computer whose local Group Policy object (GPO) you want to edit, and then click Finish
 - If you have no more snap-ins to add to the console, click OK
4. Under Root certificate stores: Import the root CAs that the local computer can trust, and then click OK to apply the new settings



5. The PFX file needs to be imported into Outlook, in order to decrypt the messages from the TC3G, as well.
- Open Outlook 2013
 - Go to Options | Trust Center | Settings of the Trust Center | E-Mail-Security | Settings
 - Choose the certificate and click OK
6. The CER file needs to be added to the Outlook contact of the TC3G, in order to encrypt outgoing messages.
- Open / Create TC3G contact information
 - Go to contact | certificate | import
 - Import the CER file and click OK and save the settings

7.2.10 machines.cloud daemon

Daemon location	Configuration file
/usr/local/bin/ycumulocityd	/etc/init.d/ycumulocityd.config

How to activate/deactivate the ycumulocitydaemon:

The ycumulocity daemon is deactivated by default. In order to activate the daemon:

1. Rename the configuration file the following way:

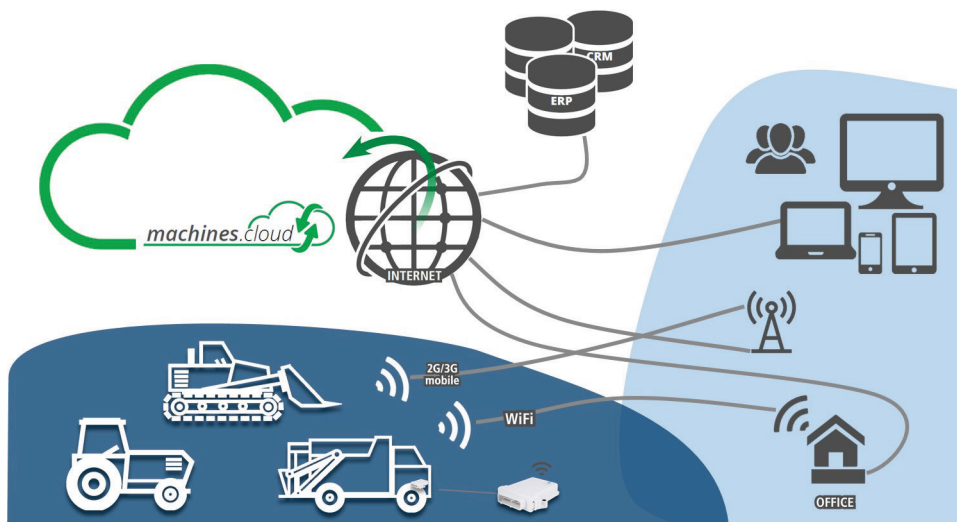
```
# mv /etc/init.d/_ycumulocityd.config /etc/init.d/ycumulocityd.config
```

2. Reboot the system:

```
# reboot
```

ycumulocityd

The ycumulocity daemon is part of the STW Teleservice Application Framework. Its purpose is to connect the device to machines.cloud or to compatible cloud platforms.



Basic configuration of the daemon could be handled in its own configuration file (ycumulocityd.config).

Relevant sections of the configuration file:

Section: Log file path

```
# The logging information can be written to a log file at the following path
# By default path is deactivated by commented the line out.
# Max. path length is 256
# If no path is set or deactivated, error message will be put on stdout.
#LOG_FILE /var/log/ycumulocityd.log
```

Section: Debug

```
# The daemon is able to print detailed debug messages when requested.
# Depending on the LOG_FILE variable, the messages will either be printed to
```

```
# stdout or directly into the specified log file.
# 0 = deactivate extended debug messages
# 1 = activate extended debug messages
#DEBUG_MODE 1
```

Section: Supervising the ycumulocityd

```
# Signal daemon can be supervised by a watchdog (provided by the ysysd daemon).
# The time when the watchdog shall be triggered is set by "WD INTERVAL".
# Min. interval is 120s, max. interval is 0x7FFFFFFF,
# recommended min. 120s / max. interval is 1000s
# If no interval is set, watchdog will be set to 120s.
WD_INTERVAL 120

# System reboot path, max. path length 256. After the time is left without
# triggering the watchdog, the ysysd would kill the ycumulocityd and execute the
# "CMD ON WD" command.
# If no path is set, default (signal01.beep) will be set.
CMD_ON_WD /sbin/reboot
```

Section: Handling log files

```
# Directory of the logger files which shall be send to cloud
# If the ycumulocityd is used in combination with the VDS system, the directory path
# must be equal to the Online transfer directory of the VDS system
DATA_LOG_DIR /mnt/dataflash/logger
```

Section: Rest interface specific parameters

```
# Default interval to communicate with the cloud REST interface in seconds
# Min Value is 1 second, max value is 3600s (1hour)
# Default value is 300 seconds
REST_COM_INTERVAL 300

# Maximal number of dataset in one post request
# Default value is 100, minimum is 1, maximum is 1000
REST_MAX_NUM_DATASET 100

# Real Time backchannel timeout in seconds. After x seconds without a request from
# server the channel will timeout and immediately reconnect. Decreasing the value will
# increase the overall accessibility in case of a bad internet connection but on the
# downside it may result in higher data traffic.
# MIN: 60 seconds, MAX: 3600 seconds, DEFAULT: 180 seconds
#RTN_TIMEOUT 180
```

Section: Cloud specific parameters

```
# URL to the REST proxy server in HTTP or HTTPS
# Default value is https://developer.machines.cloud
#BOOTSTRAP_URL https://management.ram.m2m.telekom.com
#BOOTSTRAP_URL http://developer.cumulocity.com

# There are default bootstrap user credentials set up in the code already
# Typically you do not have to change the options below
#BOOTSTRAP_USER <user_name>
#BOOTSTRAP_PASSWORD <password>

# The daemon itself supports a number of operations that will activate certain
# functionality on the machines.cloud, such as shell and tracking.
# In case the user wants to extend that list and support additional operations
# he can do that by adding them here. In order to use the widgets 'Relay Control'
# and 'Message Sending' apply the following operations:
#SUPPORTED_OPERATIONS "c8y_Relay; c8y_Message"
```

Section: Device shell settings



NOTE:
Handle these settings with care!

```
# There is the possibility to prohibit certain characters. Those cannot be executed
# in the device shell. If PROHIBITED_CHARS is not set, ";|&><" are forbidden.
#PROHIBITED_CHARS ';|&><'

# There is a possibility to extend the commands that are allowed to be executed in
# the device shell.
# In order to allow the 'cmd1' and the 'cmd2' command use:
#ALLOWED_SHELL_CMDS 'cmd1, cmd2'
```

Section: Software components surveillance

```
# The list of software components that is shown under device management -> software
# could be extended. Type the command that needs to be executed to get the software
# version. The whole path and quotes are needed. See below an example for the
# stw_GetGPS and the stw_SendSMS tool.
#SOFTWARE_COMPONENTS "/usr/local/bin/stw_GetGPS -v; /usr/local/bin/stw_SendSMS -v"
```

Section: Live channel settings

```
# Default path to the temporary data logger configuration file directory.
# The path should be equivalent to the path set in the ylogd.config file
# Default path is "/var/run/taf/DLC"
#DLC_BASE_PATH /var/run/raf/DLC
```

Section: Event Settings

```
# It is possible to disable the automatic generation of location update events.
# TRUE = The device will not send location update events to the cloud.
# FALSE = The device will send location update events to the cloud (default).
#DISABLE_LOCATION_EVENT TRUE

# It is possible to disable the automatic upload of mobile data information.
# TRUE = The device will not send mobile data updates to the cloud.
# FALSE = The device will send mobile data updates to the cloud (default).
#DISABLE_MOBILE_DATA_EVENT TRUE
```

Section: Inbox / Outbox settings

```
# INBOX - The STW Inbox functionality allows the user to send any JSON content
# via operation to the device. The operation is stored into a file.
# The name of the file could be specified within the operation itself, keyword
# is 'file_name'. This file name is extended by the current time stamp
# (e.g. YYYYMMDDhhmmss_file_name). The directory is /mnt/dataflash/stw/cloud/INBOX.
# TRUE = Activate the inbox
# FALSE = Deactivate the inbox (default)
ACTIVATE_INBOX TRUE

# There is a possibility to extend the tags that are allowed to be send to the
# inbox. The default tag is 'stw_Inbox', it is always active when the inbox is
# activated. In order to allow 'c8y_Message' and e.g. the 'c8y_Relay' tag use:
#INBOX_TAGS "c8y_Message; c8y_Relay"

# The interval in which the OUTBOX is checked for files to be send to the cloud
# can be adjusted.
# MIN: 1 seconds, MAX: 3600 seconds, DEFAULT: 10 seconds
#OUTBOX_INTERVAL 10

# OUTBOX - The STW Outbox is a very powerful mechanism, that allows the user to
# directly push alarms, events, measurements and inventory entries to the machines.cloud.
# For further information, access the inline help: 'ycumulocityd -o'.
# TRUE = Activate the outbox
# FALSE = Deactivate the outbox (default)
ACTIVATE_OUTBOX TRUE
```

7.2.10.1 INBOX

The INBOX functionality allows the user to send any JSON content via operation to the TC3G.

Each operation is stored into a separate file. The name of the file could be specified within the operation itself.

The keyword is 'file_name'. This file name is extended by the current time stamp (e.g. YYYYMMDDhhmmss_file_name).

If no file name is specified the default 'InboxOperation' is used.

Directory of the INBOX

```
/mnt/dataflash/stw/cloud/INBOX
```

Activate the INBOX (/etc/init.d/ycumulocityd.config)

```
ACTIVATE_INBOX TRUE
```

With STW's backchannel-widget it is possible to send operations to the INBOX.

The widget configuration:

- TITLE = Title of the widget on the dashboard
- BUTTON TEXT = Text displayed on the button
- DESCRIPTION = Text that will appear in the control log of the device management application

- MESSAGE = JSON conform content that will appear in the file in the INBOX on the TC3G.

EDIT WIDGET

WIDGET

Backchannel ▼

TITLE

Backchannel - Test

TARGET ASSETS OR DEVICES

✓ **TC3G-150000001004**

BUTTON TEXT

send command

DESCRIPTION

Inbox test operation

MESSAGE ⓘ

```
{
  "file_name": "InboxTestOperation",
  "test": "inbox",
  "key": "value",
  "object": {}
}
```

SAVE

CANCEL

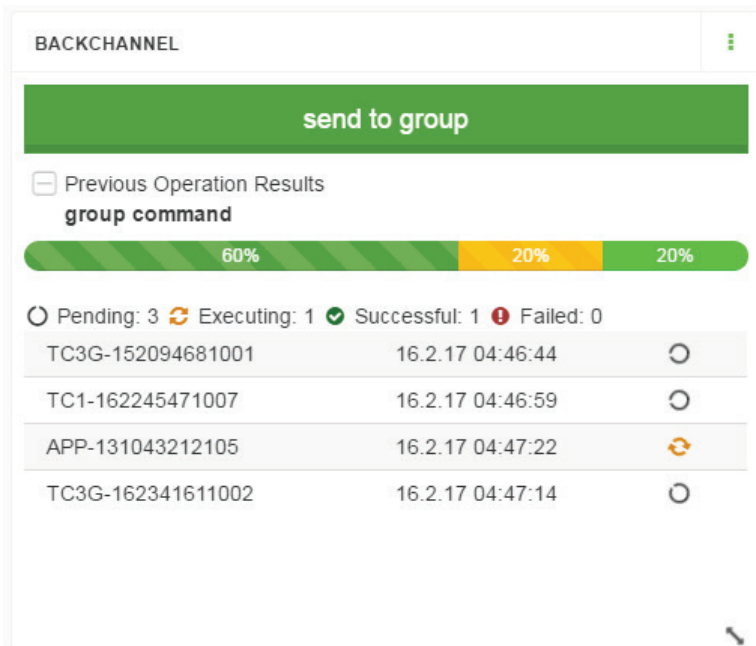
Resulting widget on the dashboard:

BACKCHANNEL - TEST
⋮

send command

TC3G-150000001004	16.2.17 03:33:29	✓
-------------------	------------------	---

The operation could be executed on a whole group as well:



The widget informs the about the state of every device. Here:

- 3 devices are pending
- 1 device is executing the operation
- 1 device has successfully executed the operation

As long as the device has not worked of the command successfully they will appear in the list below.

In case the customer want the TC3G to react on a different operation, the allowed INBOX tag needs to be extended.

Extend the allowed INBOX tag

There is a possibility to extend the tags that are allowed to be send to the inbox.

The default tag is 'stw_Inbox', it is always active when the inbox is activated. In order to allow 'tag1' and e.g. the 'c8y_Relay' tag use:

```
INBOX_TAGS "tag1; c8y_Relay"
```

With the 'c8y_Relay' tag allowed it is possible to receive an operation send from the widget 'RELAY CONTROL'.

```
Closing the relay switch
# cat /mnt/dataflash/stw/cloud/INBOX/20170216152956_InboxOperation
{
  "relayState":"CLOSED"
}
Opening the relay switch
# cat /mnt/dataflash/stw/cloud/INBOX/20170216153007_InboxOperation
{
  "relayState":"OPEN"
}
```

7.2.10.2 OUTBOX

The OUTBOX is a very powerful mechanism, that allows the user to directly push alarms, events and measurements to the machines.cloud.

The OUTBOX could be activated / deactivated in the configuration file of the daemon.

ALARM

The ALARM handling allows the user to create / update alarms on the cloud directly from the device. It is possible to update that alarm in order to e.g. escalate or acknowledge or clear it.

An alarm could be trigger, by placing an alarm-config file into the OUTBOX directory.

When the alarm could be worked off (internet connection, device activated,...) the configuration file is deleted automatically.

The response from the cloud is saved in the RESPONSE directory on the device.

Alarm configuration directory path

```
/mnt/dataflash/stw/cloud/OUTBOX/alarms
```

File extension

```
.alarm
```

Response directory path: (contains e.g. the ID of a created alarm)

```
/mnt/dataflash/stw/cloud/RESPONSE
```

The response file name is equal to the alarm config file name.
In case there was a problem, the file extension will be '.fail',
in case there was no error, the file extension will be '.ok'.

Configuration file structure

The configuration File needs to content certain parameters.

Parameter	create Alarm	update Alarm
ALARM_ID	not allowed	mandatory
TYPE	mandatory	no effect
TEXT	mandatory	optional
SEVERITY	mandatory	optional
STATUS	mandatory	optional
TIME	optional	no effect
(When no time is specified, the local time is used.)		

Parameter values

```
ALARM_ID <id of the alarm that should be updated>
TYPE <name the type of the alarm>
TEXT <text is shown when alarm occurs>
SEVERITY [ CRITICAL | MAJOR | MINOR | WARNING ]
STATUS [ ACTIVE | ACKNOWLEDGE | CLEARED ]
TIME year-month-dayThour:minutes:seconds.milliseconds+utc
e.g. 2017-02-07T12:26:59.000+00:00
```

Example

```
create alarm
```

```
# cat /mnt/dataflash/stw/cloud/OUTBOX/alarms/example.alarm
TYPE TestAlarm_1
TEXT this is a test alarm
SEVERITY CRITICAL
STATUS ACTIVE

update alarm - deescalate to warning
# cat /mnt/dataflash/stw/cloud/OUTBOX/alarms/example.alarm
ALARM_ID 26708916
SEVERITY WARNING
```

EVENT

The EVENT handling allows the user to send an event to the cloud directly from the device. In case of an error, the response from the cloud is saved in the RESPONSE directory on the device.

Event directory configuration path

```
/mnt/dataflash/stw/cloud/OUTBOX/events
```

File extension

```
.event
```

Response directory path (used in case of an error)

```
/mnt/dataflash/stw/cloud/RESPONSE
The response file name is equal to the event config file name.
(The file extension '.fail' will be added.)
```

Configuration file structure

```
The configuration File needs to be JSON conform.
The minimal key / value pairs are:
{
  "type": "<type>",      <- mandatory, name of the type of the event
  "text": "<text>",       <- mandatory, text is shown when event occurs
  "time": "<time>"       <- optional, when no time is specified, the local time is
used
  .                      <- could be extended by further objects
  .
  .
}
```

Example

```
# cat /mnt/dataflash/stw/cloud/OUTBOX/events/example.event
{
  "type": "TestEvent",
  "text": "Test event was triggered"
}
```

MEASUREMENT

The MEASUREMENT handling allows the user to send one or more measurement values to the cloud directly from the device. That way the values do not need to be part of the data pool.

They e.g. could be the result of a calculation performed by an application. In case of an error, the response from the cloud is saved in the RESPONSE directory on the device.

Measurement directory configuration path

```
/mnt/dataflash/stw/cloud/OUTBOX/measurements
```

File extension

```
.meas
```

Response directory path (used in case of an error)

```
/mnt/dataflash/stw/cloud/RESPONSE
The response file name is equal to the measurement config file name.
(The file extension '.fail' will be added.)
```

Configuration file structure

The configuration File needs to be JSON conform.

```
{
  "measurements":
  [
    {
      "<c8y_TestMeasurement>": {          <- name of the measurement (one per file!)
        "variable 1": {                  <- name of the 1st variable
          "value": <value>,              <- value of the 1st variable
          "unit": "<unit>" },            <- unit of the 1st variable
        "variable 2": {                  <- name of the 2nd variable
          "value": <value>,              <- value of the 2nd variable
          "unit": "<unit>" },            <- unit of the 2nd variable
          .                               <- could be extended by X variables
          .
          .
        },
        "type": "<c8y_TestMeasurement>"  <- name of the measurement
        "time": "<time>"                 <- optional, when no time is specified, the
local time is used
      }
    ]
  }
```

Example

```
# cat /mnt/dataflash/stw/cloud/OUTBOX/measurements/example.meas
{
  "measurements":
  [
    {
      "c8y_VehicleMeasurement": {
        "velocity": {
          "value": 100,
          "unit": "km/h" },
        "speed": {
          "value": 3000,
          "unit": "rpm" },
        "battery": {
          "value": 14,
          "unit": "V" }
      },
      "type": "c8y_VehicleMeasurement"
```

```
}
]
}
```

INVENTORY

The INVENTORY handling allows the user to send a management object to the cloud directly from the device. In case of an error, the response from the cloud is saved in the RESPONSE directory on the device.

This mechanism is quite powerful and needs to be handled with care!

Inventory directory configuration path

```
/mnt/dataflash/stw/cloud/OUTBOX/inventory
```

File extension

```
.inv
```

Response directory path (used in case of an error)

```
/mnt/dataflash/stw/cloud/RESPONSE
The response file name is equal to the inventory config file name.
(The file extension '.fail' will be added.)
```

Configuration file structure

```
The configuration File needs to be JSON conform.
There are no further limitations, the content will be transmitted to the cloud.
{
  "key": "<value>",      <- no limitation
  "object": {}           <- no limitation
  .                      <- could be extended by further objects, etc.
  .
  .
}
```

Example

```
It could also be used in order to transmit predefined structures to the cloud.
# cat /mnt/dataflash/stw/cloud/OUTBOX/inventory/example.inv
{
  "c8y_CellInfo": {
    "radioType": "gsm",
    "cellTowers": [{
      "mobileCountryCode": 240,
      "mobileNetworkCode": 1,
      "locationAreaCode": 3012,
      "cellId": 11950
    }]
  }
}
```

7.3 TAF Library

7.3.1 Introduction

The TAF library (libtaf) provides the functionality of all components on the TC3G which are necessary for a teleservice application.



NOTE:

The following description contains all possible features. Whether a single feature is available or not depends on variant of the device

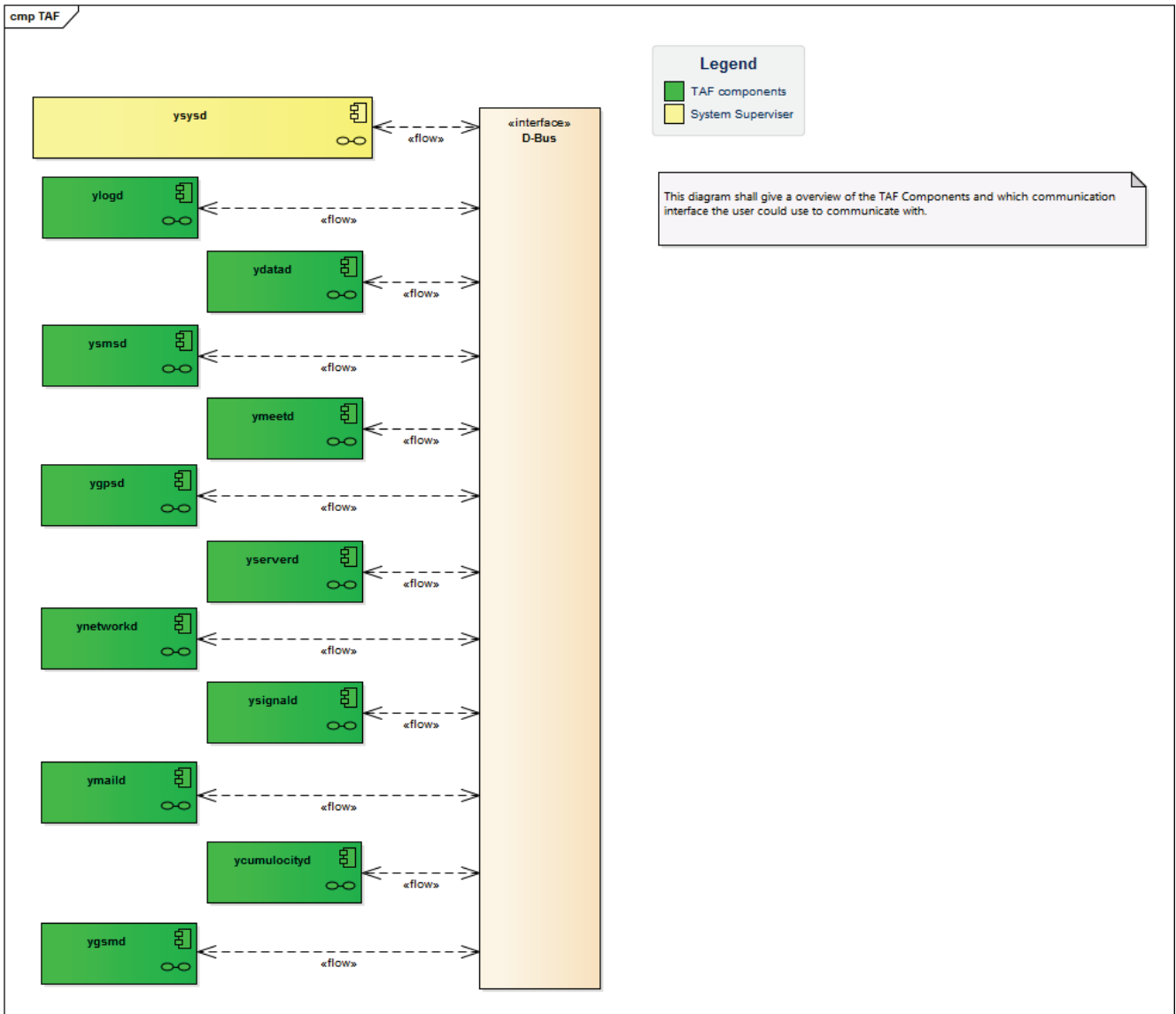
Each mentioned daemon in the TAF overview diagram will be started automatically if the following conditions are met:

- The daemon binary file has to be located within the directory "/usr/local/bin"
- Every daemon needs a configuration file
 - The configuration file has to be in the directory "/etc/init.d/"
 - The configuration file name is equal to the corresponding daemon name and the extension ".config"

A daemon can be started manually like in the following example:

```
ylogd ylogd.config &>/dev/null &
```

TAF overview



7.3.2 Notation

The following gives you a quick overview about how a function description in this document looks like.

7.3.2.1 Types and Prefixes

Header file: "stwtypes.h"

STW type definitions

Used types like 'word', 'long' or 'dword' are mistakable. They can have a different meaning, depending on the platform where they are used.

The meaning of 'word' means 16 bits at a 16-bit controller like the Infineon C167 or the Freescale 68k. At a 32-bit controller like the used TriCore processor 'word' means 32 bits. On TriCore hardware 16 bits are called a 'halfword' instead.

To avoid confusions STW introduced clear types like 'uint16'.

Recommended clear type definitions and prefixes

STW type	Prefix	Size	Range (hexadecimal - HEX)	Range (decimal - DEC)	C equivalent
uint8	u8_	8 bit	0x00 .. 0xFF	0 .. 255	unsigned char
sint8	s8_	8 bit	0x80 .. 0x7F	-128 .. +127	signed char
uint16	u16_	16 bit	0x0000 .. 0xFFFF	0 .. 65535	unsigned short int
sint16	s16_	16 bit	0x8000 .. 0x7FFF	-32768 .. +32767	signed short int
uint32	u32_	32 bit	0x00000000 .. 0xFFFFFFFF	0 .. 4294967295	unsigned long int
sint32	s32_	32 bit	0x80000000 .. 0x7FFFFFFF	- 2147483648 .. +2147483647	signed long int
uint64	u64_	64 bit	0x0000000000000000 .. 0xFFFFFFFFFFFFFFFF	0 .. 18446744073709551615	unsigned long long int
sint64	s64_	64 bit	0x8000000000000000 .. 0x7FFFFFFFFFFFFFFF	- 9223372036854775808 .. +9223372036854775807	signed long long int
float32	f32_	32 bit	[$\approx \pm 2^{-126} .. \pm 2^{127}$]	0, $\approx \pm 1.18E-38 .. \pm 3.4E38$ (decimal places 6..7)	float (FPU: IEEE-754) single precision
float64	f64_	64 bit	[$\approx \pm 2^{-1022} .. \pm 2^{1023}$]	0, $\approx \pm 2.225E-308 .. \pm 1.798E+308$ (decimal	double (SW: IEEE-754)

STW type	Prefix	Size	Range (hexadecimal - HEX)	Range (decimal - DEC)	C equivalent
				places 15..16)	double precision


NOTE:

For information about using floating point arithmetic see programming hints.

NATIVE type definitions

STW type	Prefix	Size	Range (hexadecimal - HEX)	Range (decimal - DEC)	C equivalent
charn	cn_	8 bit	0x80 .. 0x7F	-128 .. +127	char
uintn	un_	32 bit	0x00000000 .. 0xFFFFFFFF	0 .. 4294967295	unsigned int
sintn	sn_	32 bit	0x80000000 .. 0x7FFFFFFF	- 2147483648 .. +2147483647	signed int


NOTE:

The width of native data types (charn, uintn, and sintn) depend on the used platforms. On 16-bit platforms an integer has a width of 16-bits - at 32-bit platform it is 32-bit wide. The API uses this behavior for definition of platform comprehensive functions.

Not recommended mistakable COMPATIBILITY type definitions

STW type	Prefix	Size	Range (hexadecimal - HEX)	Range (decimal - DEC)	C equivalent
boolean	q_	8 bit	0x00:FALSE / (!=0x00):TRUE	0:FALSE / (!=0):TRUE	unsigned char

STW also defines the data type boolean to store binary data. A boolean data type can help to increase the performance. However boolean is target specific and should be used with care. On some platforms a boolean variable takes 1 bit of memory. Such a variable can only be 0 or 1 but can not be referenced by pointers or members of a structure. Other platforms define boolean as an integer variable. In this case it is not guaranteed, that the value of this variable is 0 or 1 - it could also be greater than 1. If a platform defines a boolean as an integer, the variable can be referenced by pointers or member of a structure.


WARNING:

Note:

The usage of boolean can lead to unportable code.

Type definition (typedef) prefixes

Prefix	Example	Description
T_	<code>typedef struct { ... } T_Struct;</code>	type definition for structure (or bitfield)
U_	<code>typedef union { ... } U_Union;</code>	type definition for union
E_	<code>typedef enum { ... } E_Enum;</code>	type definition for enum
PR_	<code>typedef void (*PR_Function)(const uint8 ou8_Parameter);</code>	type definition for function pointer

Definition prefixes

Prefix	Example	Description
t_	<code>T_Struct t_Struct;</code>	structure (or bitfield)
u_	<code>U_Union u_Union;</code>	union
e_	<code>E_Enum e_Enum;</code>	enum
pr_	<code>PR_Function pr_FunctionPointer;</code>	function pointer
pv_	<code>void *pv_VoidPointer;</code>	void pointer
s_	<code>char s_Text[3]="12";</code>	zero ('\0') terminated string

Modifier prefixes

Prefix	Example	Description
p<type prefix>	<code>uint8 *pu8_Example;</code>	pointer of type
a<type prefix>	<code>uint8 au8_Example[4];</code>	array of type

Area of validity prefixes

Prefix	Example	Description
g<type prefix>	<code>uint8 gu8_Global;</code>	global
m<type prefix>	<code>static uint8 mu8_ModuleGlobal;</code>	module global (static inside file)
h<type prefix>	<code>static uint8 hu8_Example;</code>	local static (static inside function - hold)

Prefix	Example	Description
o<type prefix>	void Callback(const uint8 ou8_Parameter);	function parameter (operand)

Examples

```
//Prefix examples

/* local variable sint16 */
sint16 s16_VariableName;

/* function internal array of pointers to uint8 with MAX_MESSAGES elements */
uint8 *apu8_Messages[MAX_MESSAGES];

/* Module global variable of type uint32 */
static uint32 mu32_Index;

/* typedef for structure */
typedef struct
{
    uint8 u8_Element1;
    uint32 *pu32_Element2;
} T_StructType;

/* modul globale pointer to structure defined by typedef */
static T_StructType *mpt_PointerToMyStructType;

/* function internal used structure */
T_StructType t_MyStructObject;

/* module global structure variable */
static T_StructType mt_MyStruct;

/* typedef for function pointer for a module 'DigIN' */
typedef void (*PR_DIN_CallBack)(const uint8 ou8_Parameter);

/* global function pointer in module 'DigIN' */
PR_DIN_CallBack gpr_DIN_CallBack;
```

7.3.2.2 Function descriptions

The following gives you a quick overview about how a function description in this document looks like:

- First there is a function description (see "[Function descriptions](#)" on page 144) that contains the function prototype and a detailed description.
- The information flow (see "[Function descriptions](#)" on page 144) contains information about the input/output parameters (operands) and the returning values (error codes).
- Next there is an example code (see "[Function descriptions](#)" on page 144) that contains a little demonstration code.



NOTE:

The examples are meant to show the principle of how to use the function and are not necessarily compilable.


WARNING:

DO NOT USE undocumented BIOS functions!

The availability and functionality of undocumented BIOS functions CANNOT be assured.

Function Description

```
sint16 x_function_name (const uint8 ou8_VarName1, const uint32 ou32_VarName2, uint16 * const
opu16_PointerName )
```

Detailed function description...

Information Flow

Input Information

Parameter	Range	Description
ou8_VarName1	0 .. 63	8-bit input parameter
ou32_VarName2	0x00000000 .. 0xFFFFFFFF (no limitation)	32-bit input parameter

Output Information

Result	Range	Description
opu16_PointerName	0x0000 .. 0xFFFF (no limitation)	Pointer to 16-bit output parameter variable

Return Value	Description
C_NO_ERR	function executed without error
C_RANGE	parameter ou8_VarName1 out of range

Example code:

```
// call of x_function_name()
void main(void)
{
    sint16 s16_Result = C_NO_ERR;
    uint16 u16_Value = 0;

    s16_Result = x_function_name(15, 0x20000, &u16_Value);
    if (s16_Result != C_NO_ERR) // check if an error occurred!
    {
        u16_Value = 0;
    }
    else
    {
        u16_Value *= 2;
    }
}
```

```

:
}

```

7.3.2.3 Error Codes

The error codes used by the C-BIOS API functions are handled by using macros.


NOTE:

The corresponding values here are informal only (e.g. for debugging). Values/Constants should not be used direct for coding.

The following macros are defined:

Macro	Value	Description
C_NO_ERR	0	Function executed without error
C_UNKNOWN_ERR	-1	Unknown error
C_WARN	-2	Warning error
C_DEFAULT	-3	Default error
C_BUSY	-4	Busy error
C_RANGE	-5	Range error
C_OVERFLOW	-6	Overflow error
C_RD_WR	-7	Read/Write error
C_NOACT	-8	No actual value (available) error
C_COM	-9	Communication error
C_CONFIG	-10	Configuration error
C_CHECKSUM	-11	Checksum error
C_TIMEOUT	-12	Timeout occurred
C_IN_PROGRESS	-13	Asynchronous operation still in progress

Example

```
// Error code handling example

sint16 s16_Result;
s16_Result = x_sys_stay_alive(X_ON);

if (s16_Result == C_NO_ERR)
{
    ; // if reached then this means that function x_sys_stay_alive(..)
    // has been executed successfully!
}
```

7.3.3 D-Bus Utils

Header file: "dbus-utils.h"

What is D-Bus?

D-Bus is a message bus system that provides a simple way for applications to talk to one another. In addition to interprocess communication, D-Bus helps coordinate the process life cycle. It makes it simple and reliable to code a "single instance" application or daemon, and to launch applications and daemons on demand. [freedesktop.org]

7.3.3.1 Introduction

The TAF library provides useful D-Bus functionality to use TAF functionality at runtime. Almost every TAF function uses the structure T_DBUS_Util.

Input and output parameters of the structure T_DBUS_Util:

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn          acn_myNameString[128];   // Name of the application, Don't use any
special                                         // characters, white spaces or new lines!
    charn          acn_myVersion[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];       // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];      // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;      // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```


7.3.3.2 Initialize the D-Bus

How to initialize the D-Bus:

Before the D-Bus can be used it must be initialized.

Example on how to initialize the D-Bus

```

/* -- Module Global Variables ----- */
T_DBUS_Util mt_DBUS_util;
/* -- Implementation ----- */
int main(int argc, char** argv)
{
    sint32 s32_Retval;

    /* Basic initialization */

    // Application information
    sprintf(mt_DBUS_util.acn_myNameString, "TAF example");
    sprintf(mt_DBUS_util.acn_myVersion, "vX.XXrX");
    sprintf(mt_DBUS_util.acn_myStatus, "Run");
    sprintf(mt_DBUS_util.acn_myAddInfo, "Example application making use of TAF
components");
    // Trigger interval that the ysysd will expect a trigger signal from us in seconds
    mt_DBUS_util.s32_myTriginterval = 8;
    // In case the watchdog goes off, make the ysysd execute the following bash command for
us
    // Ensure to use the entry path to the bash-cmd and redirect its outputs (>/dev/null)
    sprintf(mt_DBUS_util.acn_myCMDOnWatchdog, "/usr/local/bin/signal02.beep >/dev/null");

    // Creates connection and registers application on D-Bus
    s32_Retval = dbus_get_on_the_bus(&mt_DBUS_util);
    if(s32_Retval != C_NO_ERR)
    {
        // Error handling
        return C_UNKNOWN_ERR;
    }
    // ...
    // Add rules for which messages have to be seen on the D-Bus
    dbus_bus_add_match (mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.broadcast'", NULL);
    dbus_connection_flush (mt_dbus_util.pt_dbus_conn);
    dbus_bus_add_match (mt_dbus_util.pt_dbus_conn,
"type='method',interface='stw.taf.ysysd'", NULL);
    dbus_connection_flush (mt_dbus_util.pt_dbus_conn);
    // ...

    /* More initialization steps */

    // ...

    /* Start main loop */
    while (1)
    {
        // Trigger the watchdog every 1 seconds
...
        (void) usleep (1000);

    }

    return 0;
}

```

7.3.3.3 dbus_get_on_the_bus

Function Description

```
sint32 dbus_get_on_the_bus (T_DBUS_Util * const opt_DBusInstance)
```

Creates a connection to the D-Bus and registers the application. This function finishes the initialization of the opt_DBusInstance instance. Make sure that all members of this structure are properly set before calling this function.



NOTE:

This is the only function where the opt_DBusInstance is used as a input / output parameter.



WARNING:

T_DBUS_Util: Don not use any special characters, whitespaces or new lines.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all information which is used to generate a D-Bus connection instance

Output Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util ->pt_dbus_conn	includes the connection instance which is generated by the D-Bus

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn          acn_myNameString[128];  // Name of the application, Don't use any
    special                                               // characters, white spaces or new lines!
    charn          acn_myVersion[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];     // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;     // Time interval when the ysysd
                                               // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                               // by the ysysd in case no
                                               // trigger signal occurred in time.
} T_DBUS_Util;
```

Return Value	Description
C_NO_ERR	Function executed without error. The connection to the D-Bus was successful
C_COM	Communication error, parameter pt_dbus_conn of T_DBUS_Util is set to NULL
C_CONFIG	Conflict with configured D-Bus name, acn_myNameString of T_DBUS_Util
C_RANGE	Parameter opt_DBusInstance is a NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
// Basic initialization (see "Introduction" on page 148) for TAF communication
// ...
```

7.3.3.4 dbus_close

Function Description

```
sint32 dbus_close (T_DBUS_Util * const opt_DBusInstance)
```

Unregister from the D-Bus and close connection.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all information regarding the D-Bus connection instance

Output Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util ->pt_dbus_conn	includes the connection instance which is generated by the D-Bus

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                                  // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                  // by the ysysd in case no
                                                  // trigger signal occurred in time.
} T_DBUS_Util;
```

Return Value	Description
C_NO_ERR	Function executed without error.
C_CONFIG	Could not unregister from D-Bus
C_RANGE	Parameter opt_DBusInstance is a NULL pointer

7.3.3.5 dbus_send_hallo_signal

Function Description

```
sint32 dbus_send_hallo_signal (const T_DBUS_Util * const opt_DBusInstance)
```

Call this function to inform other applications on the D-Bus about your application. It creates a signal (HELLO signal) on the D-Bus to that other TAF applications can listen. The use of the HELLO signal is not mandatory for applications but could be used to ensure the presence of cooperating applications.



NOTE:

All TAF daemons send a HELLO signal on startup and on request (=dbus_who_is_there_signal).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!

    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];   // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error. "Hello" signal was successful sent
C_UNKNOWN_ERR	D-Bus signal sent, out of memory error
C_CONFIG	D-Bus message append argument error
C_RANGE	opt_DBusInstance is a NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
// Basic initialization (see "Introduction" on page 148) for TAF communication
// ...
s32_Retval = dbus_send_hello_signal (&mt_DBUS_Util);
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.3.6 dbus_send_goodbye_signal

Function Description

```
sint32 dbus_send_goodbye_signal(const T_DBUS_Util * const opt_DBusInstance)
```

This function informs other application over D-Bus that your application will be closed.

The function creates a goodbye signal and sends it over the D-Bus so other applications can listen.

Use this function to ensure the presence of cooperating applications. The use of the goodbye signal is not mandatory.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all information regarding the D-Bus connection instance

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error.
C_UNKNOWN_ERR	DBus signal send, but a end of memory error occurred
C_CONFIG	An arguments error is appended to the D-BUS message

Return Value	Description
C_RANGE	Parameter opt_DBusInstance is a NULL pointer

7.3.3.7 dbus_who_is_there_signal

Function Description

```
sint32 dbus_who_is_there_signal (const T_DBUS_Util * const opt_DBusInstance)
```

All TAF daemons send a HELLO signal on startup and on request. This function creates a WholsThere signal on the D-Bus which then triggers all TAF daemons and the applications that are reacting on it, to answer with HELLO signals. This mechanism can be handy when your applications relies on the presence and services of certain TAF daemons or applications. It can be used to make sure that all required daemons are running before your applications tries to use them.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed with no error. The WholsThere signal was sent on the D-Bus
C_UNKNOWN_ERR	Could not create the WholsThere signal

Return Value	Description
C_WARN	Could not add the WholsThere signal on outgoing D-Bus queue
C_RANGE	opt_DBusInstance is a NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = dbus_who_is_there_signal (&mt_DBUS_Util);
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.3.8 dbus_call_method

Function Description

```
sint32 dbus_call_method (const T_DBUS_Util * const opt_DBusInstance, const charn * const
opcn_DBusTarget, const charn * const opcn_MethodName, const charn * const
opcn_MethodArgument, charn * const opcn_Answer, const sint32 os32_BufferSize)
```

This function is a generic D-Bus utility function that makes a method call on the D-Bus . It calls a method on D-Bus target with one string argument and receives a one string answer.



NOTE:

This function is used from TAF internal functions and it is not recommended to use it from user applications. Use the corresponding daemon functions.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_DBusTarget	maximal 240	contains target on D-Bus e.g. "ysysd"
opcn_MethodName	maximum the name length we expect to call	pointer to the method from target e.g. "GetGSMMMode"
opcn_MethodArgument	maximum the name length we expect to call	pointer to the method argument e.g. "12345"
os32_BufferSize	1 .. answer length we expect	size of the reply answer

Output Information

Result	Range	Description
opcn_Answer	os32_BufferSize	reply of the method call

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                                  // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                  // by the ysysd in case no
                                                  // trigger signal occurred in time.
} T_DBUS_Util;
```

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call
C_COM	Error receiving the method call answer
C_NOACT	Answer string received but it starts with "ERROR"
C_OVERFLOW	Answer string is larger than the buffer
C_RANGE	Some of the function arguments are NULL pointer or os32_BufferSize < 1

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
charn acn_Value[256];
sint32 s32_Retval;
s32_Retval = dbus_call_methode (&mt_DBUS_Util, "ysysd", "GetGSMMMode", "", acn_Value,
sizeof(acn_Value));
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.3.9 dbus_call_signal

Function Description

```
sint32 dbus_call_signal (const T_DBUS_Util * const opt_DBusInstance, const charn * const
opcn_DBusTarget, const charn * const opcn_SignalName, const charn * const
opcn_SignalArgument);
```

This generic D-Bus utility function creates a signal call on the D-Bus with one string argument.



NOTE:

This function is used from TAF internal functions and it is not recommended to use it from user applications. Use the corresponding daemon functions.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_DBusTarget	maximal 240	contains target on D-Bus e.g. "ysysd"
opcn_SignalName	maximum the name length we expect to call	pointer to the signal from target e.g. "GetGSMMode"
opcn_SignalArgument	maximum the name length we expect to call	pointer to the signal argument e.g. "12345"

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error.
C_UNKNOWN_ERR	Error sending the signal call
C_WARN	Error sending the signal call -> out of memory
C_RANGE	Some of the function arguments are NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = dbus_call_signal(&mt_DBUS_Util, "my_daemon", "SetTimer", "1000");
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.3.10 dbus_initialize_request_callbacks

Function Description

```
void dbus_initialization_request_callbacks (const PR_LIBTAF_GET_SMS opr_SMSCallback, const
PR_LIBTAF_GET_HELLO opr_HelloCallback,
const PR_LIBTAF_NETWORK_NOTIFICATION opr_NetworkCallback, const
PR_LIBTAF_GET_GOODBYE opr_GoodbyeCallback,
const PR_LIBTAF_GET_NRTI opr_NetworkResponseTimeIndicator, const
PR_LIBTAF_WHOISTHERE_NOTIFICATION opr_WhoIsThereCallback,
const PR_LIBTAF_USER_MESSAGE_NOTIFICATION opr_UserMessageCallback)
```

The TAF informs your application about certain events regarding the GSM communication. Register your own callback functions with the TAF with this function.

The user message callback can be used to receive user defined messages in an application. It is possible to use one or more other callbacks in the same application.

Information Flow

Input Information

Parameter	Range	Description
opr_SMSCallback		callback function for SMS
opr_HelloCallback		callback function for hello signal

Parameter	Range	Description
opr_NetworkCallback		callback function for network notification service
opr_GoodbyeCallback		callback function for goodbye signal
opr_NetworkResponseTimeIndicator		callback function for NetworkResponseTimeIndicator
opr_WholsThereCallback		callback function for WholsThere signal
opr_UserMessageCallback		callback function for a message defined by the user

Callback function prototypes

SMS:

```
typedef void (* PR_LIBTAF_GET_SMS) (const charn * const opcn_PhoneNumber, const charn * const opcn_Message);
```

Output Information

Parameter	Range	Description
opcn_PhoneNumber	< 20 characters	Phone number of the subscriber which has sent the received SMS
opcn_Message	0 - 160 characters	Received SMS

Hello:

```
typedef void (* PR_LIBTAF_GET_HELLO) (const charn * const opcn_DaemonName);
```

Output Information

Parameter	Range	Description
opcn_DaemonName	0 - 128 characters examples for typical daemon names: <ul style="list-style-type: none"> • ylogd • ydatad • ymsd 	Name of the daemon which has sent the "hello" signal

Network notification service:

```
typedef void (* PR_LIBTAF_GET_NETWORK_NOTIFICATION) (const charn * const opcn_InterfaceName, const charn * const opcn_InterfaceStatus);
```

Output Information

Parameter	Range	Description
opcn_InterfaceName	<ul style="list-style-type: none"> • ETH • WLAN • PPP 	Contains the short name of the interface which is currently used, e.g. ETH...Ethernet
opcn_InterfaceStatus	<ul style="list-style-type: none"> • Down • UP 	Signalizes the interface connection to the internet, if UP or Down.

Goodbye:

```
typedef void (* PR_LIBTAF_GET_GOODBYE) (const charn * const opcn_DaemonName);
```

Output Information

Parameter	Range	Description
opcn_DaemonName	0 - 128 characters examples for typical daemon names: <ul style="list-style-type: none"> • ydatad • ygpsd • ylogd 	Name of the daemon which has sent the "goodbye" signal

NRTI (NetworkResponseTimeIndicator):

```
typedef void (* PR_LIBTAF_GET_NRTI) (const charn * const opcn_NetworkTimeIndicator);
```

Output Information

Parameter	Range	Description
opcn_NetworkTimeIndicator	0 - 128 characters	network response time indicator in milliseconds (only an approximate value)

WholsThere:

```
typedef void (* PR_LIBTAF_WHOISTHERE_NOTIFICATION) (const charn * const opcn_DaemonName);
```

Output Information

Parameter	Range	Description
opcn_DaemonName	0 - 128 characters examples for typical daemon names: <ul style="list-style-type: none"> • ydatad • ygpsd 	Name of the daemon which has sent the "WholsThere" signal

Parameter	Range	Description
	<ul style="list-style-type: none"> ylogd 	

User_Message:

```
typedef void (* PR_LIBTAF_USER_MESSAGE_NOTIFICATION) (DBusMessage * const opt_msg);
```

Output Information

Parameter	Range	Description
opt_msg	DBus_Message	Object representing a message received from or to be sent to another application.

For receiving all information, add the following dbus rules to your source code:

```
dbus_bus_add_match(mt_dbus_util.pt_dbus_conn, "type='signal',interface='stw.taf.ysmsd'",
NULL);
dbus_bus_add_match(mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.ynetworkd'", NULL);
dbus_bus_add_match(mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.broadcast'", NULL);
```

Example

```
// Global module
static void mv_SMS_Callback (const charn * const opcn_PhoneNumber, const charn * const
opcn_Message)
{
    // TODO
}

static void mv_Hello_Callback (const charn * const opcn_DaemonName)
{
    // TODO
}

static void mv_NetworkNotification_Callback (const charn * const opcn_InterfaceName, const
charn * const opcn_InterfaceStatus)
{
    // TODO
}

static void mv_Goodbye_Callback (const charn * const opcn_DaemonName)
{
    // TODO
}

static void mv_NRTI_Callback (const charn * const opcn_NetworkTimeIndicator)
{
    // TODO
}

static void mv_WhoIsThere_Callback (const charn * const opcn_DaemonName)
{
    // TODO
}

static void mv_UserDBusMessage (DBusMessage * const opt_msg)
{
}
```

```

// TODO
}

// module global variables
static T_DBUS_Util mt_dbus_util;
// main function
int main(int argc, char** argv)
{
    // Init stuff ...
    ...
    // Add rules for which messages must be seen on the DBUS
    dbus_bus_add_match(mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.broadcast'", NULL);
    dbus_connection_flush(mt_dbus_util.pt_dbus_conn);
    dbus_bus_add_match(mt_dbus_util.pt_dbus_conn, "type='signal',interface='stw.taf.ysmsd'",
NULL);
    dbus_connection_flush(mt_dbus_util.pt_dbus_conn);
    dbus_bus_add_match(mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.ynetworkd'", NULL);
    dbus_connection_flush(mt_dbus_util.pt_dbus_conn);
    // Install the callback functions
    dbus_initialize_request_callbacks (&mv_SMS_Callback, &mv_Hello_Callback,
&mv_NetworkNotification_Callback, &mv_Goodbye_Callback,
&mv_NRTI_Callback, &mv_WhoIsThere_Callback,
&mv_UserDBusMessage);
    ...
}

```

Example: UserMessageCallback

The example below shows how the UserMessageCallback is used by the ysignal daemon.

```

.
.
.
int main(int osn_argc, char** oppcn_argv)
{
    ...

    // Initialize the global D-BUS structure T_DBUS_Util
    (void)sprintf(mt_dbus_util.acn_myNameString , MY_DBUS_NAME);
    (void)sprintf(mt_dbus_util.acn_myVersion   , PROG_VERSION);
    (void)sprintf(mt_dbus_util.acn_myStatus    , "unknown");
    (void)sprintf(mt_dbus_util.acn_myAddInfo   , MY_DBUS_ADDINFO);

    // Register with DBUS
    if(dbus_get_on_the_bus(&mt_dbus_util) != 0)
    {
        (void)utils_log_print("Register ysignald on DBUS failed");
        return(0);
    }

    // Add rules for which messages we want to see on the DBUS
    dbus_bus_add_match (mt_dbus_util.pt_dbus_conn,
"type='signal',interface='stw.taf.ysignald'", NULL);
    dbus_connection_flush (mt_dbus_util.pt_dbus_conn);

    // Install the callback functions Hello_Callback, Goodbye_Callback, and
    UserMessage_Callback
    dbus_initialize_request_callbacks (NULL, &mv_Hello_Callback, NULL,
&mv_Goodbye_Callback, NULL, NULL,
&mv_UserMessage_Callback);
    ...

    while(true)
    {

```

```

...
    // process D-BUS requests
    s32_retval = dbus_process_requests(&mt_dbus_util);
    if(s32_retval != C_NO_ERR)
    {
        (void)printf("dbus_process_request went wrong!\n");
        mv_Exit();
    }
...
}
.
.
.

static void mv_UserMessage_Callback(DBusMessage * const opt_msg)
{
    DBusMessageIter t_args;
    ...
    if(opt_msg != NULL)
    {
        // check if the message is "Internet_Connection_State"
        if (dbus_message_is_signal(opt_msg, "stw.taf.ysignald", "Internet_Connection_State")
== TRUE)
        {
            // Get the message arguments
            if (!dbus_message_iter_init(opt_msg, &t_args))
            {
                (void)printf("ProcessSignalArgument: DBUS message has no arguments\n");
            }
            dbus_message_iter_get_basic(&t_args, &pcn_Argument);
            (void)utils_strncpy(acn_SignalState, pcn_Argument, sizeof( acn_SignalState));
            (void)printf("Internet_Connection_State: %s\n", acn_SignalState);
        }
    }
    ...
}
.
.
.

```

7.3.3.11 dbus_process_requests

Function Description

```
void dbus_process_requests (const T_DBUS_Util * const opt_DBusInstance)
```

This function needs to be called periodically in order to allow the TAF library to check for signals on the D-Bus and to call your registered callback functions.



WARNING:

The function must be called frequently!

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!

    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error.
C_COM	If any error occurs while communicating via D-Bus
C_CONFIG	D-Bus instance is not correct, an instance of type T_DBUS_Util is expected

Example

```
// Global Module
sint32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
while (1)
{
    s32_Retval = dbus_process_requests (&mt_DBUS_Util);
    if (s32_Retval != C_NO_ERR)
    {
        // Handle error cases
    }
}
```

7.3.4 System

Header file: "System_handler.h"

7.3.4.1 Introduction

The TAF library system component uses the ysys daemon to provide all system relevant functions like register, trigger or cancel the watchdog.

7.3.4.2 ysysd_register_watch_dog

Function Description

```
sint32 ysysd_register_watch_dog (const T_DBUS_Util * const opt_DBusInstance)
```

Requests a watchdog service from the ysysd over the D-Bus, which sets the parameters for future observations of this application. The parameter opt_DBusInstance must contain all necessary parameters like the watchdog trigger interval etc. before this function is called.

It is important to cancel (unregister) this service at the end of your application.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
    special                                                // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                                // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                // by the ysysd in case no
                                                // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error.

Return Value	Description
C_WARN	Error registering a watchdog service from the ysysd (out of memory)
C_BUSY	Error registering a watchdog service from the ysysd (pending call)
C_UNKNOWN_ERR	Error registering a watchdog service from the ysysd (stealing reply)
C_CONFIG	Error registering a watchdog service from the ysysd (Message has no arguments)
C_COM	Error registering a watchdog service from the ysysd (Answer is not a string).

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ysysd_register_watch_dog (&mt_DBUS_Util);
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.4.3 ysysd_trigger_watch_dog

Function Description

```
sint32 ysysd_trigger_watch_dog (const T_DBUS_Util * const opt_DBusInstance)
```

Sends a trigger over the D-Bus to the system daemon to prevent it to kill the application. This functions needs to be called within the registered time interval (set during the ysysd_register_watch_dog function call in the parameter opt_DBusInstance).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
}
```

```

// by the ysysd in case no
// trigger signal occurred in time.
} T_DBUS_Util;

```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Could not sent trigger message
C_RANGE	Parameter opt_DBusInstance is a NULL pointer

Example

```

// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ysysd_trigger_watch_dog (&mt_DBUS_Util);
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}

```

7.3.4.4 ysysd_cancel_watch_dog

Function Description

```
sint32 ysysd_cancel_watch_dog (const T_DBUS_Util * const opt_DBusInstance)
```

This functions calls a method of the system daemon over the D-Bus that cancels the supervision of our application. Make sure that this function is called whenever your applications ends, even by kill or ctrl+c. Otherwise the system daemon will execute the acn_myCMDOnWatchdog command after the trigger time has elapsed.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```

typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special

```

```

charn          acn_myVersion[128];    // characters, white spaces or new lines!
charn          acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
charn          acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
sint32         s32_myTriginterval;    // Time interval when the ysysd
                                     // expects to be triggered
charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                     // by the ysysd in case no
                                     // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, the watchdog stopped successfully
C_UNKNOWN_ERR	Could not cancel the watchdog service from ysysd
C_RANGE	Parameter opt_DBusInstance is a NULL pointer
C_COM	The application was never registered with ysysd

Example

```

// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ysysd_cancel_watch_dog (&mt_DBUS_Util);
if (s32_Retval != C_NO_ERR)
{
    // Handle error cases
}
```

7.3.4.5 ysysd_get_ignition_status

Function Description

```
sint32 ysysd_get_ignition_status (const T_DBUS_Util * const opt_DBusInstance)
```

Returns the current status of the ignition pin from the system daemon. It calls a method of the system daemon via the D-Bus, requesting the current status of the ignition pin.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special          // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
    // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
    // by the ysysd in case no
    // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
IGN_OFF	Ignition pin is low (off)
IGN_ON	Ignition pin is high (on)
C_UNKNOWN_ERR	Could not retrieve the ignition status from system daemon
C_RANGE	Parameter opt_DBusInstance is a NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ysysd_get_ignition_status (&mt_DBUS_Util);
if (s32_Retval == IGN_OFF)
{
    // Ignition is off
}
else if (s32_Retval == IGN_ON)
{
    // Ignition is on
}
else
{
    // Error reading the ignition status
}
```

7.3.4.6 ysysd_request_stay_alive

Function Description

```
sint32 ysysd_request_stay_alive (const T_DBUS_Util * const opt_DBusInstance, const uint32
ou32_DurationSec)
```

With this function the system daemon receives over the D-Bus the command not to shutdown the system, for at least the demanded amount of time in seconds.

If the state of the ignition pin switches to low and the configured time has elapsed, the system shuts down.

If your application is in a crucial task that must not be interrupted (e.g. saving data to flash, transferring data to the server), then use this function so that the TC3G stays alive for the time specified with `ou32_DurationSec`, until this crucial task is finished.

Information Flow

Input Information

Parameter	Range	Description
<code>opt_DBusInstance</code>	<code>T_DBUS_Util</code>	holds all D-Bus information
<code>ou32_DurationSec</code>	minimum 1	time in seconds, the system should stay alive from now on

Structure `T_DBUS_Util`

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
    special                                                // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                                // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                // by the ysysd in case no
                                                // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
<code>C_NO_ERR</code>	system daemon accepted the command
<code>C_UNKNOWN_ERR</code>	Could not sent the stay alive request to the system daemon (right ysysd version, daemon running?)
<code>C_RANGE</code>	requested duration is 0 or <code>opt_DBusInstance</code> is a NULL pointer

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ysysd_request_stay_alive (&mt_DBUS_Util, 10);
if (s32_Retval != C_NO_ERR)
{
    // Don't start any of the critical work
}
```

```
// here because we can not trust the system
// to keep running.
}
// Do the critical work here, the system keeps running
```


7.3.5 Datapool

Header file: "DP_handler.h" and "DP_dbus_handler.h"

These header files provide utility functions to handle the data pool daemon.

7.3.5.1 Introduction

The data pool functionality of the TAF library uses the ydata daemon to provide access to common data pools. It provides applications and processes the opportunity to define, create and delete variable lists via D-Bus commands. In order to achieve quick response times and a good performance when working with these variables, writing and reading will not be handled via D-Bus but by the "shared memory" mechanism.

A data pool can be generated in two different ways:

- Dynamic mode: The data pool will be generated at runtime.
- Static mode: The data pool is created by a user and contains 1..n data pool lists.

The used mode is determined in the ydatad configuration file.

7.3.5.2 Dynamic Mode

Header file: "DP_dbus_handler.h"

This header provides utility functions to use the dbus and data pool daemon in dynamic mode.

7.3.5.2.1 ydatad_create_variable_list

Function Description

```
sint32 ydatad_create_variable_list (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_Datapool, const charn *const opcn_List, const charn * const opcn_Description,
const charn * const opcn_Creator)
```

The function creates a variable list opcn_List in the data pool opcn_Datapool. To describe the variables of the list use opcn_Description. The name of the creator of the list can be added to the list with the parameter opcn_Creator.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the name of the data pool
opcn_List	maximum MAX_DP_NAME_LENGTH	pointer to the name of the variable that is listed in the data pool.
opcn_Description	maximum MAX_DP_METANAME_LENGTH	pointer to the description for the variable list
opcn_Creator	maximum MAX_DP_METANAME_LENGTH	pointer to the name of the creator e.g. tool or user

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn          acn_myNameString[128];   // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn          acn_myVersion[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];       // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];      // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;      // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call or null pointer
C_COM	Error receiving the method call answer
C_NOACT	Error occurs because data pool is already initialized or variable list already exists
C_RANGE	Parameters are out of range

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ydatad_create_variable_list (&mt_DBUS_Util, "MyDatapool", "GPSDates",
"Description", "MyApplication");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.2 ydatad_add_variable_to_list

Function Description

```
sint32 ydatad_add_variable_to_list (const T_DBUS_Util const *const opt_DBusInstance, const
charn * const opcn_Datapool, const charn *const opcn_List, const charn * const opcn_Name,
const charn * const opcn_Type, const charn * const opcn_Size, const charn * const
opcn_Unit, const charn * const opcn_Comment)
```

The function adds a variable with the name `opcn_Name` of the type `opcn_Type`, the size `opcn_Size` to the specified list `opcn_List` within the data pool `opcn_Datapool`. It's also possible to add a unit `opcn_Unit` and a comment `opcn_Comment` to the variable declaration.



NOTE:

It's not possible to add variables to already initialized lists or initialized data pools.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the data pool name

Parameter	Range	Description
opcn_List	maximum MAX_DP_NAME_LENGTH	pointer to the name of the variable list
opcn_Name	maximum MAX_DP_NAME_LENGTH	pointer to the name of the variable
opcn_Type	maximum MAX_DP_TYPENAME_LENGTH, allowed types: "UINT8" "SINT8" "UINT16" "SINT16" "UINT32" "SINT32" "FLOAT32" "AOBYTE" "STRING"	pointer to the type of the variable
opcn_Size	maximum MAX_DP_SIZE_LENGTH, allowed type size: "UINT8" = 1 "SINT8" = 1 "UINT16" = 2 "SINT16" = 2 "UINT32" = 4 "SINT32" = 4 "FLOAT32" = 4 "AOBYTE" = sizeof (au8_byte) "STRING" = strlen (acn_string)	pointer to the size in byte of the type
opcn_Unit	maximum MAX_DP_METANAME_LENGTH	pointer to the user defined unit of the variable
opcn_Comment	maximum MAX_DP_METANAME_LENGTH	pointer to the variable description

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                                  // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                  // by the ysysd in case no
                                                  // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call
C_COM	Error receiving the method call answer
C_RANGE	Error miss match in variable type and size
C_NOACT	Error occurs because data pool or variable list already initialized, variable list does not exist or could not open variable list file

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ydatad_add_variable_to_list (&mt_DBUS_Util, "MyDatapool", "GPSDates",
"Longitude", "FLOAT_32","4", "degree", "Der Laengengrad");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.3 ydatad_init_variable_list

Function Description

```
sint32 ydatad_init_variable_list (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_Datapool , const charn *const opcn_List)
```

This function creates and initializes the shared memory segment for the named variable list opcn_List within the data pool opcn_Datapool.


NOTE:

After initialization, it isn't possible to add more variables to the initialized list!

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the name of the data pool
opcn_List	maximum MAX_DP_NAME_LENGTH	pointer to the name of the variable list

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128]; // Name of the application, Don't use any
    special                                                // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                                // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                // by the ysysd in case no
                                                // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call or NULL pointer
C_COM	Error receiving the method call answer
C_NOACT	Error occurred because the data pool or variable list was already initialized, variable list does not exist, determinate shared memory size failed or allocating of a shared memory segment failed
C_RANGE	Parameters are out of range

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ydatad_init_variable_list (&mt_DBUS_Util, "MyDatapool", "GPSDates");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.4 ydatad_delete_variable_list

Function Description

```
sint32 ydatad_delete_variable_list (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_Datapool , const charn *const opcn_List)
```

The function deletes the shared memory segment for the named variable list ocn_List and the according variable list file in the data pool ocn_Datapool.



NOTE:

The variable list can only be deleted if it was initialized before.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the name of the data pool
opcn_List	maximum MAX_DP_NAME_LENGTH	pointer to the name of the variable list

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
```

```
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, variable list was successfully deleted
C_UNKNOWN_ERR	Error sending the method call or NULL pointer error
C_COM	Error receiving the method call answer
C_NOACT	Error occurred because the variable list did not exist, getting shared memory segment failed or deleting shared memory segment failed
C_RANGE	Parameters are out of range

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ydatad_delete_variable_list (&mt_DBUS_Util, "MyDatapool", "GPSDates");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.5 ydatad_get_datapool_base_path

Function Description

```
sint32 ydatad_get_datapool_base_path (const T_DBUS_Util *const opt_DBusInstance, charn *
const opcn_BasePath, const sint32 os32_Size)
```

This function returns the base path of all data pools on the system, as configured in the ydatad configuration file. The maximum length of opcn_BasePath is defined with the variable os32_Size.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
os32_Size	minimum MAX_DP_BASEFOLDER LENGHT	includes the buffer size of the base path

Output Information

Parameter	Range	Description
opcn_BasePath	os32_Size	pointer to base path

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                                  // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                  // by the ysysd in case no
                                                  // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, base path was returned successful
C_UNKNOWN_ERR	Error sending the method call or NULL pointer
C_COM	Error receiving the method call answer
C_NOACT	Reply error from bus, can't get data pool base path
C_RANGE	Size of base path array is to small

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
charn acn_Value[256];
s32_Retval = ydatad_get_datapool_base_path (&mt_DBUS_Util, acn_Value, sizeof (acn_Value));
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.6 ydatad_init_datapool

Function Description

```
sint32 ydatad_init_datapool (const T_DBUS_Util *const opt_DBusInstance, charn * const
opcn_Datapool)
```

The function initializes the data pool ocn_Datapool. Before initializing the data pool, all lists within the data pool have to be initialized. After initialization, it isn't possible to add more lists to the initialized data pool. The ydata daemon creates a file in the named data pool subdirectory with the extension ".dpi". This file contains the information about the created data pool.



NOTE:

The data pool can only be initialized if minimum one variable was add to the variable list.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the name of the data pool

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, data pool was initialized successful
C_UNKNOWN_ERR	Error sending the method call or NULL pointer
C_COM	Error receiving the method call answer

Return Value	Description
C_NOACT	Error occurred because the data pool sub directory did not exist or the dpi-file already existed
C_RANGE	Parameter out of range

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ydatad_init_datapool (&mt_DBUS_Util, "MyDatapool");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.2.7 ydatad_delete_datapool

Function Description

```
sint32 ydatad_delete_datapool (const T_DBUS_Util *const opt_DBusInstance, charn * const
opcn_Datapool)
```

This function delete the data pool ocn_Datapool. The data pool is deleted and all resources are released, regardless whether the data pool was initialized before or not



WARNING:

Deleting a data pool will have impact on all programs using this data pool. Handle this function with care!

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_Datapool	maximum MAX_DP_NAME_LENGTH	pointer to the name of the data pool

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special
```

```

    charn          acn_myVersion[128];      // characters, white spaces or new lines!
    charn          acn_myStatus[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];      // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;      // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;

```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call or NULL pointer
C_COM	Error receiving the method call answer
C_NOACT	Data pool does not exist
C_RANGE	Parameter out of range

7.3.5.2.8 ydatad_validate_variable

Function Description

`E_DataType ydatad_validate_variable (const charn * const opcn_Type, const charn * const opcn_Size)`

The function checks whether the type of variable exists and if the size is correct. If both input values fit, the function returns the corresponding "variable type index".

Information Flow

Input Information

Parameter	Range	Description
opcn_Type	allowed types: "UINT8" "SINT8" "UINT16" "SINT16" "UINT32" "SINT32" "FLOAT32"	pointer to the variable type

Parameter	Range	Description
	"AOBYTE" "STRING"	
opcn_Size	allowed type size: "UINT8" = 1 "SINT8" = 1 "UINT16" = 2 "SINT16" = 2 "UINT32" = 4 "SINT32" = 4 "FLOAT32" = 4 "AOBYTE" = sizeof (au8_byte) "STRING" = strlen (acn_string)	pointer to the size of variable type

Enum E_DataType

```
typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
} E_DataType;
```

Output Information

Return Value	Description
eNULL	Type is unknown
else	Type and size is found. Returns the variable type index. See E_DataType.

7.3.5.3 Static Mode

7.3.5.3.1 Basic folder structure

A data pool is only a folder with a specific name. Every data pool contains 1...n data pool lists files (DPL). A data pool list file contains 1...n variables. Every variable in the DPL file is added to the shared memory.

Example

The static data pool is stored in the folder /opt/taf/.

The name of the data pool is "TestDatapool".

The data pool contains the DPL file "Engine.dpl".

```
ls -al /opt/taf
drw-r--r--  2 root  root 232 Dec 5 08:21 TestDatapool
ls -al /opt/taf/TestDatapool/
-rwxr-x-r-x 1 root root 3306 Dec 5 08:21 Engine.dpl
```

7.3.5.3.2 Basic DPL file structure

Every data pool list file has the following basic components:

Necessary DPL file objects

Parameter	Description
[CONFIG]	Start tag of the DPL file
NUMOFVARS	Number of variables in the DPL file (is set by ydatad)
LISTNAME	Name of the data pool list file
DESCRIPTION	Small description of the DPL file
CREATING_PRGM	Is set by application or in case of a user defined DPL file free text

Example

```
[CONFIG]
NUMOFVARS=0
LISTNAME=ENGINE
DESCRIPTION=Test List
CREATING_PRGM=JOHNE DOE
```

7.3.5.3.3 Add variable to DPL file

The following components have to be added to the DPL file in order to add a variable:

Variable objects

Parameter	Description
[VARIABLE1-n]	Start tag which contains the actual number of the variable
NAME	Name of the variable
ADDRESS	The address starts at 0, if you are adding the first variable with the size of 4 byte, the next variable starts at address number 4.
SIZE	Size of the variable in byte
TYPE	Type of the variable: UINT8 : 1 byte SINT8 : 1 byte UINT16 : 2 byte SINT16 : 2 byte UINT32 : 4 byte SINT32 : 4 byte FLOAT32 : 4 byte AOBYTE : UINT8 * SIZE STRING: SINT8 * SIZE
UNIT	Unit of the variable
COMMENT	Specific comment for the variable

Example

```
[CONFIG]
NUMOFVARS=0
LISTNAME=ENGINE
DESCRIPTION=Test List
CREATING_PRGM=JOHN DOE

[VARIABLE1]
NAME=EngineSpeed
ADDRESS=0
SIZE=4
TYPE=UINT32
UNIT=rpm
COMMENT=EngineSpeed
[VARIABLE2]
NAME=OIL
ADDRESS=4
SIZE=2
TYPE=UINT16
UNIT=bar
COMMENT=Pressure
[VARIABLE3]
NAME=EnginePower
ADDRESS=6
SIZE=4
TYPE=UINT32
UNIT=%
```

```
COMMENT=EnginePower
```

7.3.5.4 Access Data Pool

Header file: "DP_handler.h"

This header file provides utility functions to use the data pool daemon.

7.3.5.4.1 ydatad_load_datapool

Function Description

```
sint32 ydatad_load_datapool (const sint32 os32_DatapoolIndex, const charn * const opcn_Path)
```

This function parses the sub folder of the indexed database ou32_DatapoolIndex and creates a data structure for each valid data list. Afterwards, it's possible to use the indexed data pool.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the index of the data pool
opcn_Path	maximum MAX_DP_BASEFOLDER LENGHT	pointer to the base folder of the data pool

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, data pool was loaded successful
C_UNKNOWN_ERR	NULL pointer
C_RANGE	Index or path of data pool does not exist, isn't valid, the variable list file couldn't be opened or parameter opcn_Path is out of range

Example

```
// Global module
T_DATA_DATAPOOL_INFO *pt_Datapool;
sint32 s32_Retval = C_NO_ERR;
charn acn_HelpString[MAX_DP_BASEFOLDER LENGHT];
uint32 u32_NumberOfDatapools
u32_NumberOfDatapools = ydatad_get_size_loaded_datapools ();
for (uint32 u32_i = 0; u32_i < s32_NumberOfDatapools; u32_i++)
{
    s32_Retval = ydatad_get_datapool_info (u32_i, &pt_Datapool);
    if (s32_Retval != C_NO_ERR)
    {
        // Do error handling
    }
    if (strstr (pt_Datapool_Info->acn_DatapoolName, "MyDatapool") != NULL)
```



```

{
    s32_Retval = ydatad_load_datapool (u32_i, acn_HelpString);
    if (s32_Retval != C_NO_ERR)
    {
        // Error handling
    }
}
}

```

7.3.5.4.2 ydatad_get_number_of_loaded_datapools

Function Description

sint32 ydatad_get_number_of_loaded_datapools(void)

The function returns the number of loaded data pools.



WARNING:

First call the function ydatad_load_datapool_names before calling this function.

Information Flow

Output Information

Return Value	Description
> 0	Number of loaded data pools
0	No data pools initialized or function ydatad_load_datapool_names was not called before

Example

```

// Global module
uint32 u32_NumberOfDatapools;
u32_NumberOfDatapools = ydatad_get_number_loaded_datapools ();

```

7.3.5.4.3 ydatad_load_datapool_names

Function Description

sint32 ydatad_load_datapool_names (const charn * const opcn_Path)

This command skims through the base folder and creates a data pool structure for each subfolder that contains a valid data pool.

Information Flow

Input Information

Parameter	Range	Description
opcn_Path	maximum MAX_DP_BASEFOLDER_LENHT	pointer to the base folder of the data pools

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	NULL pointer
C_RANGE	No data pools available or path size error

Example

```
// Global module
sint32 s32_Retval;
T_DBUS_Util mt_DBUS_Util;
charn acn_HelpString[MAX_DP_BASEFOLDER LENGHT];
s32_Retval = ydatad_get_datapool_base_path (&mt_DBUS_Util, acn_HelpString,
sizeof(acn_HelpString));
if (s32_Retval != C_NO_ERR)
{
    // Error handling
}
s32_Retval = ydatad_load_datapool_names (acn_HelpString)
if (s32_Retval != C_NO_ERR)
{
    // Error handling
}
```

7.3.5.4.4 ydatad_get_datapool_info

Function Description

```
sint32 ydatad_get_datapool_info (const sint32 os32_DatapoolIndex, T_DATA_DATAPOOL_INFO **
oppt_Datapool)
```

The function gets the data pool info structure from the corresponding data pool index.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes a data pool index from data pool list

Output Information

Parameter	Range	Description
oppt_Datapool	T_DATA_DATAPOOL_INFO	pointer to the data pool structure

Structure T_DATA_DATAPOOL_INFO

```
typedef struct
{
    charn          acn_DatapoolName[MAX_DP_NAME_LENGTH]; // Data pool name, max size 32
    sint32         s32_NbOfLists; // Number of data pool lists (T_DATA_LIST_INFO)
    T_DATA_LIST_INFO* pt_lists;
```

```

}T_DATA_DATAPOOL_INFO;

typedef struct
{
    charn          acn_ListName[MAX_DP_NAME_LENGTH]; // Name of the list, max size 32
    charn          acn_Description[MAX_DP_METANAME_LENGTH]; // Description of list,
max size of 64
    charn          acn_CreatingPRGM[MAX_DP_METANAME_LENGTH]; // Creator or program
name, max size of 64
    sint32         s32_NbOfVars; // Number of variables in the list
(T_DATA_VARIABLE_INFO)
    sint32         s32_SHM_ID;
    sint32         s32_Sem_ID;
    void*          pv_SHM_Address;
    T_DATA_VARIABLE_INFO* pt_Vars;
} T_DATA_LIST_INFO;

typedef struct
{
    charn          acn_VarName[MAX_DP_NAME_LENGTH]; // Variable name, max size of 32
    charn          acn_Type[MAX_DP_TYPENAME_LENGTH]; // Type of variable: max size of 10
// "UINT8"
// "SINT8"
// "UINT16"
// "SINT16"
// "UINT32"
// "SINT32"
// "FLOAT32"
// "AOBYTE"
// "STRING"

    E_DataType e_TypeIndex;
    sint32       s32_Size; // Size of the variable
    charn        acn_Unit[MAX_DP_METANAME_LENGTH]; // Unit of the variable, max size of 64
    charn        acn_Comment[MAX_DP_METANAME_LENGTH]; // Comment of the variable, max size of
64
    void*        pv_SHM_Address;
    sint32       s32_Address_Offset;
} T_DATA_VARIABLE_INFO;

typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
}E_DataType;

```

Return Value	Description
C_NO_ERR	Function executed without error, the pool info was received successfully
C_RANGE	Selected data pool index isn't valid

Example

```
// Global module
T_DATA_DATAPOOL_INFO *pt_Datapool;
sint32 s32_Retval;
uint32 u32_NumberOfDatapools
u32_NumberOfDatapools = ydatad_get_size_loaded_datapools ();
for (uint32 u32_i = 0; u32_i < s32_NumberOfDatapools; u32_i++)
{
    s32_Retval = ydatad_get_datapool_info (u32_i, &pt_Datapool);
    if (s32_Retval != C_NO_ERR)
    {
        // Error handling
    }
}
```

7.3.5.4.5 ydatad_get_datapool_index

Function Description

sint32 ydatad_get_datapool_index (const charn * const opcn_DatapoolName)

The function returns the data pool index number of the expected data pool name opcn_DatapoolName.



WARNING:

The data pool names have to be loaded first!

Information Flow

Input Information

Parameter	Range	Description
opcn_DatapoolName	1 .. MAX_DP_NAME_LENGTH	pointer to the name of the expected data pool

Return Value	Description
>=0	Returns the index number of the data pool with the expected name
C_UNKNOWN_ERR	NULL pointer
C_RANGE	Data pool does not exist or data pool name length out of range

Structure T_DATA_VARIABLE_INFO

```
typedef struct
{
    charn    acn_VarName[MAX_DP_NAME_LENGTH]; // Variable name, max size of 32
    charn    acn_Type[MAX_DP_TYPENAME_LENGTH]; // Type of variable: max size of 10
                                                // "UINT8"
                                                // "SINT8"
```

```

// "UINT16"
// "SINT16"
// "UINT32"
// "SINT32"
// "FLOAT32"
// "AOBYTE"
// "STRING"

E_DataType e_TypeIndex;
sint32      s32_Size; // Size of the variable
charn      acn_Unit[MAX_DP_METANAME_LENGTH]; // Unit of the variable, max size of 64
charn      acn_Comment[MAX_DP_METANAME_LENGTH]; // Comment of the variable, max size of
64
void*      pv_SHM_Address;
sint32      s32_Address_Offset;
} T_DATA_VARIABLE_INFO;

typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
} E_DataType;

```

```

// Global module
sint32 s32_DataPoolIndex;
s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")

```

7.3.5.4.6 ydatad_get_variable_index

Function Description

```

sint32 ydatad_get_variable_index (const sint32 ops32_DatapoolIndex, const sint32
ops32_VariableListIndex, const charn * const opcn_VariableName)

```

The function returns the index number of the variable name opcn_VariableName.



WARNING:

The data pool names, data pools and lists have to be loaded first.

Information Flow

Input Information

Parameter	Range	Description
ops32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
ops32_VariableListIndex	0 .. maximum of sint32	includes the variable list index

Parameter	Range	Description
opcn_VariableName	1 .. MAX_DP_NAME_LENGTH	pointer to the variable name

Output Information

Return Value	Description
>= 0	Returns the variable index number of the named variable list and data pool
C_UNKNOWN_ERR	NULL pointer
C_RANGE	Invalid data pool, list or variable or variable name out of range

Structure T_DATA_VARIABLE_INFO

```
typedef struct
{
    charn      acn_VarName[MAX_DP_NAME_LENGTH]; // Variable name, max size of 32
    charn      acn_Type[MAX_DP_TYPENAME_LENGTH]; // Type of variable: max size of 10
                                                    // "UINT8"
                                                    // "SINT8"
                                                    // "UINT16"
                                                    // "SINT16"
                                                    // "UINT32"
                                                    // "SINT32"
                                                    // "FLOAT32"
                                                    // "AOBYTE"
                                                    // "STRING"

    E_DataType e_TypeIndex;
    sint32     s32_Size; // Size of the variable
    charn      acn_Unit[MAX_DP_METANAME_LENGTH]; // Unit of the variable, max size of 64
    charn      acn_Comment[MAX_DP_METANAME_LENGTH]; // Comment of the variable, max size of
64
    void*      pv_SHM_Address;
    sint32     s32_Address_Offset;
} T_DATA_VARIABLE_INFO;

typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
} E_DataType;
```

Example

```
// Global module
sint32 s32_DataPoolIndex;
sint32 s32_ListIndex;
```

```
sint32 s32_VarIndex;
T_DATA_VARIABLE_INFO t_DataVariableInfo;
sint32 s32_Retval;
s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")
s32_ListIndex = ydatad_get_list_index (s32_DataPoolIndex , "GPSDates");
s32_VarIndex = ydatad_get_variable_index (s32_DataPoolIndex, s32_ListIndex , "Logitude");
```

7.3.5.4.7 ydatad_get_list_index

Function Description

```
sint32 ydatad_get_list_index (const sint32 ops32_DatapoolIndex, const charn * const
opc_n_VariableListName)
```

The function returns the variable list index number ops32_DatapoolIndex, of the variable list name opc_n_VariableListName.



WARNING:

The data pool names and the data pool itself have to be loaded first.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the index of the data pool
opc_n_VariableListName	1 .. MAX_DP_NAME_LENGTH	pointer to the variable list name

Output Information

Return Value	Description
>= 0	Returns the data pool index number of the named list
C_UNKNOWN_ERR	NULL pointer
C_RANGE	Data pool not initialized or data pool index invalid or variable list name out of range

Structure T_DATA_VARIABLE_INFO

```
typedef struct
{
    charn      acn_VarName[MAX_DP_NAME_LENGTH]; // Variable name, max size of 32
    charn      acn_Type[MAX_DP_TYPENAME_LENGTH]; // Type of variable: max size of 10
                                                    // "UINT8"
                                                    // "SINT8"
                                                    // "UINT16"
                                                    // "SINT16"
                                                    // "UINT32"
                                                    // "SINT32"
                                                    // "FLOAT32"
                                                    // "AOBYTE"
                                                    // "STRING"
```

```

E_DataType e_TypeIndex;
sint32      s32_Size; // Size of the variable
charn       acn_Unit[MAX_DP_METANAME_LENGTH]; // Unit of the variable, max size of 64
charn       acn_Comment[MAX_DP_METANAME_LENGTH]; // Comment of the variable, max size of
64
void*       pv_SHM_Address;
sint32      s32_Address_Offset;
} T_DATA_VARIABLE_INFO;

typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
}E_DataType;

```

Example

```

// Global module
sint32 s32_DataPoolIndex;
sint32 s32_ListIndex;
s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")
s32_ListIndex = ydatad_get_list_index (s32_DataPoolIndex , "GPSDates");

```

7.3.5.4.8 ydatad_set_variable

Function Description

```

sint32 ydatad_set_variable (const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex, const sint32 os32_VariableIndex, const void * const opv_Buffer, const
uint32 ou32_NumberOfBytesToWrite)

```

This function verifies whether the indexed variable exists and that it is not locked by a write operation of another process. The function writes the number of bytes from the buffer to the variable.



WARNING:

The variable list and data pool have to be initialized before. In addition the data pool has to be loaded!

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index

Parameter	Range	Description
os32_VariableIndex	0 .. maximum of sint32	includes the variable index
opv_Buffer	ou32_NumberOfBytesToWrite	pointer to the value for the variable
ou32_NumberOfBytesToWrite	depends on the variable size	includes the size of byte that must be written to

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, value was set successful to the variable
C_RANGE	Data pool, list or variable does not exist

Example

```
// Global module
sint32 s32_DataPoolIndex;
sint32 s32_ListIndex;
sint32 s32_VarIndex;
float32 f32_Value = 5;
sint32 s32_Retval;

s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")
s32_ListIndex = ydatad_get_list_index (s32_DataPoolIndex , "GPSDates");
s32_VarIndex = ydatad_get_variable_index (s32_DataPoolIndex, s32_ListIndex , "Longitude");
s32_Retval = ydatad_set_variable (s32_DataPoolIndex, s32_ListIndex, s32_VarIndex,
&f32_Value, sizeof(float32));
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.4.9 ydatad_get_variable

Function Description

```
sint32 ydatad_get_variable (const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex, const sint32 os32_VariableIndex, const void * const opv_Buffer, uint32
const ou32_SizeOfBuffer)
```

The function verifies, whether the indexed variable exists and reads the maximum number of bytes into the buffer of the variable memory of opv_Buffer.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index

Parameter	Range	Description
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index
os32_VariableIndex	0 .. maximum of sint32	includes the variable index
ou32_SizeOfBuffer	depends on variable size	includes the size of the buffer in bytes for the buffer for the variable ou32_SizeOfBuffer

Output Information

Parameter	Range	Description
opv_Buffer	ou32_SizeOfBuffer	pointer to the value of the variable

Return Value	Description
C_NO_ERR	Function executed without error, read value and set it into given buffer was successful
C_RANGE	Data pool, list or variable does not exist

Example

```
// Global module
sint32 s32_DataPoolIndex;
sint32 s32_ListIndex;
sint32 s32_VarIndex;
float32 f32_Value;
sint32 s32_Retval;
s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")
s32_ListIndex = ydatad_get_list_index (s32_DataPoolIndex , "GPSDates");
s32_VarIndex = ydatad_get_variable_index (s32_DataPoolIndex, s32_ListIndex , "Longitude");
s32_Retval = ydatad_get_variable (s32_DataPoolIndex, s32_ListIndex, s32_VarIndex,
&f32_Value, sizeof(float32));
if (s32_Retval == C_NO_ERR)
{
    // Go on
    printf("Value: %f\n", f32_Value);
}
```

7.3.5.4.10 ydatad_get_variable_info Function Description

```
sint32 ydatad_get_variable_info (const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex, const sint32 os32_VariableIndex, T_DATA_VARIABLE_INFO * const
opt_DataVariableInfo)
```

The function verifies whether the indexed variable exists and returns a pointer to the variable information structure opt_DataVariableInfo.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index
os32_VariableIndex	0 .. maximum of sint32	includes the variable index

Output Information

Parameter	Range	Description
opt_DataVariableInfo	T_DATA_VARIABLE_INFO	includes the data variable info

Structure T_DATA_VARIABLE_INFO

```
typedef struct
{
    charn      acn_VarName[MAX_DP_NAME_LENGTH]; // Variable name, max size of 32
    charn      acn_Type[MAX_DP_TYPENAME_LENGTH]; // Type of variable: max size of 10
                                                    // "UINT8"
                                                    // "SINT8"
                                                    // "UINT16"
                                                    // "SINT16"
                                                    // "UINT32"
                                                    // "SINT32"
                                                    // "FLOAT32"
                                                    // "AOBYTE"
                                                    // "STRING"

    E_DataType e_TypeIndex;
    sint32     s32_Size; // Size of the variable
    charn      acn_Unit[MAX_DP_METANAME_LENGTH]; // Unit of the variable, max size of 64
    charn      acn_Comment[MAX_DP_METANAME_LENGTH]; // Comment of the variable, max size of
64
    void*      pv_SHM_Address;
    sint32     s32_Address_Offset;
} T_DATA_VARIABLE_INFO;

typedef enum
{
    eUNKNOWN = 0,
    eUINT8,
    eSINT8,
    eUINT16,
    eSINT16,
    eUINT32,
    eSINT32,
    eFLOAT32,
    eAOBYTE,
    eSTRING,
    eNULL
} E_DataType;
```

Return Value	Description
C_NO_ERR	Function executed without error, read data pool info runs successful
C_RANGE	Data pool, list or variable does not exist, or NULL pointer

Example

```
//Global module
sint32 s32_DataPoolIndex;
sint32 s32_ListIndex;
sint32 s32_VarIndex;
T_DATA_VARIABLE_INFO t_DataVariableInfo;
sint32 s32_Retval;
s32_DataPoolIndex = ydatad_get_datapool_index ("MyDatapool")
s32_ListIndex = ydatad_get_list_index (s32_DataPoolIndex , "GPSDates");
s32_VarIndex = ydatad_get_variable_index (s32_DataPoolIndex, s32_ListIndex , "Logitude");
s32_Retval = ydatad_get_variable_info (s32_DataPoolIndex, s32_ListIndex ,s32_VarIndex
,&t_DataVariableInfo);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.5.4.11 ydatad_check_update_list

Function Description

```
sint32 ydatad_check_update_list(const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex)
```

The function checks if at least one entry of the data pool list specified by os32_DatapoolIndex and os32_VariableListIndex has been updated since the last call of this function.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index

Output Information

Return Value	Description
C_NO_ERR	The list has been updated
C_NOACT	The list has not been updated
C_RANGE	Data pool or variable list does not exist

7.3.5.4.12 ydatad_check_update_variable

Function Description

```
sint32 ydatad_check_update_variable(const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex, const sint32 os32_VariableIndex)
```

The function checks if the variable specified by os32_DatapoolIndex, os32_VariableListIndex and os32_VariableIndex has been updated since the last call of ydatad_get_variable.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index
os32_VariableIndex	0 .. maximum of sint32	includes the variable index

Output Information

Return Value	Description
C_NO_ERR	The list has been updated
C_NOACT	The list has not been updated
C_RANGE	Data pool, variable list or variable does not exist

7.3.5.4.13 ydatad_get_updated_variable

Function Description

```
sint32 ydatad_get_updated_variable(const sint32 os32_DatapoolIndex, const sint32
os32_VariableListIndex, sint32* const ops32_VariableIndex)
```

The function gets the variable index of the first updated variable in the data pool list specified by os32_DatapoolIndex and os32_VariableListIndex.

Information Flow

Input Information

Parameter	Range	Description
os32_DatapoolIndex	0 .. maximum of sint32	includes the data pool index
os32_VariableListIndex	0 .. maximum of sint32	includes the variable list index
os32_VariableIndex	0 .. maximum of sint32	includes the variable index

Output Information

Parameter	Range	Description
ops32_VariableIndex		returned variable index

Return Value	Description
C_NO_ERR	Updated variable found
C_NOACT	No variable updated in list
C_RANGE	Data pool, or variable list does not exist, or NULL pointer

7.3.6 Datalogger

Header file: "DL_LogFile_handler.h" and "DL_dbus_handler.h"

These header files provide utility functions to handle the logger daemon.

7.3.6.1 Introduction

The datalogger TAF library functionality uses the ylog daemon to provide the opportunity to create static or dynamic logging jobs.

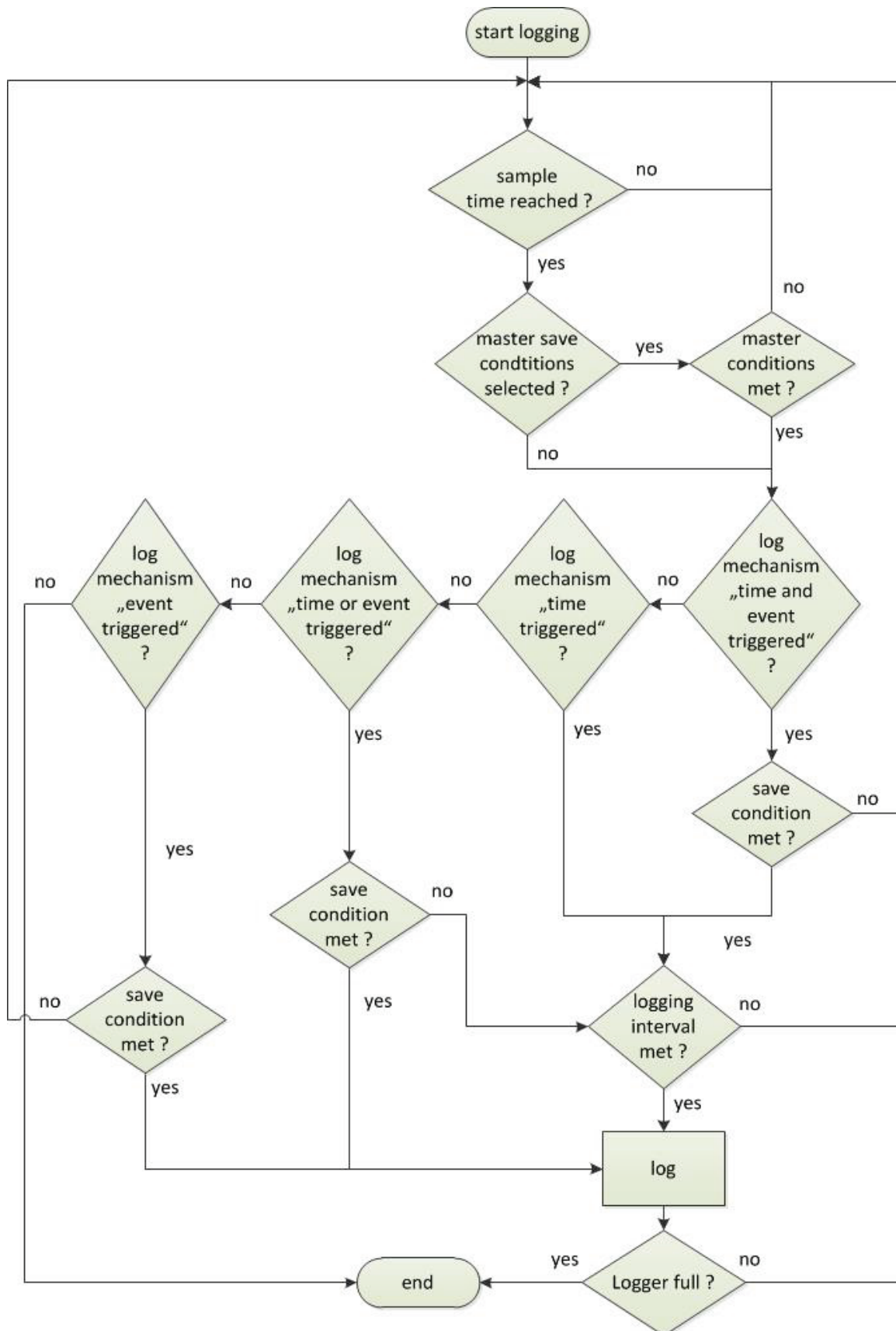
Dynamic logging mode:

Each application can add variables that should be logged from the logging job. An application can define trigger conditions for these variables that make the daemon write a new data record into the log-file.

Static logging mode:

A user created data logging configuration file (dlc) will be used. This file will be parsed by the ylogd. If no error occurs while parsing, the data logger is ready for use. The used mode is determined over the ylogd configuration file.

How the logger mechanism works



7.3.6.2 Dynamic mode

Header file: "DL_dbus_handler.h"

This header provides utility functions to use the dbus and the logger daemon in dynamic mode.

7.3.6.2.1 ylogd_create_log_job

Function Description

```
sint32 ylogd_create_log_job (const T_DBUS_Util *const opt_DBusInstance, const charn * const
opcn_LogJobName ,const uint32 ou32_LogFileFormat, const uint32 ou32_JobSize, const uint32
ou32_Timestamp, const charn * const opcn_Comment, const uint32 ou32_SampleTime)
```

The function creates a new log job. It creates the data logger configuration (DLC) sub-folder under the base path and also the basic logger job file (.dlc). The logger which should be created will log the values in the ou32_LogJobFormat format. The maximum size of the logger is defined by ou32_LogJobSize. At the beginning of every logjob dataset entry, a time stamp will be written if ou32_Timestamp is set to TRUE (1). Every ou32_SampleTime milliseconds, the logger checks all values if they have changed. If they have changed, the values will be updated to an internal structure. After the trigger time has elapsed, the internal structure will be written into the logging file opcn_LogJobName.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the name of the data log job
ou32_LogFileFormat	1 .. 5	Format option for log file 1 : ASCII_STD 2 : ASCII_TC1 3 : ASCII_CSV 4 : BINARY_C2C_LE 5 : BINARY_TC3_LE
ou32_JobSize	internal: 0 .. 524287999 (< 500 MB) external: 0 .. 4 GByte	maximum size of the log job in bytes (internal / external e.g. USB storage)
ou32_Timestamp	0 or 1	time stamp On or Off (0 1)
opcn_Comment	maximum MAX_DL_METANAME_LENGTH	pointer to the comment buffer
ou32_SampleTime	1 .. uint32	interval in milliseconds between variable update and trigger condition checks

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                           characters, white spaces or new lines!

    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, job successfully created
C_UNKNOWN_ERR	Error occurs during sending the method call or NULL pointer
C_COM	Error occurs during receiving the method call answer
C_NOACT	Error occurs because logger job name or log file format invalid, basic sample interval too short or logger job configuration already exists
C_RANGE	Parameters are out of range

Logfile format description

ASCII_STD

```
Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;1
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp;1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART
```

Time stamp	;	Index	;	VAR Name	;	Value	;	Index	;	VAR Name	;	Value	;	...	;
------------	---	-------	---	----------	---	-------	---	-------	---	----------	---	-------	---	-----	---

23.11.10_10:00:51:159;[0x000003];Height;688;[0x000005];Course;327;[0x000100];Temperature;25; // First line
23.11.10_10:00:52:150;[0x000100];Temperature;25; // Second line

ASCII_TC1

Log File header: ----- LOGFILE_HEADER LogFile;Test_Logger LogFileFormat;2 Comment;Test log job for Test MaxSize;10485760 Date+Time;22.11.11_13:30:53 FileInfo: Timestamp:1 DP :00;Test Lst:00;Engine Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:% DATASTART

\$	Time stamp in sec since 2000	;	Value	;	Value	;	Value	;	Value	;	...	;
----	------------------------------	---	-------	---	-------	---	-------	---	-------	---	-----	---

\$540000;28;755;Test;28;56332; // First line
\$540001;;;Test2;29;;; // Second line

ASCII_CSV

Log File header: ----- TIMESTAMP;<DP name> > <DPL name> > <Var name> [Unit];<DP name> > <DPL name> > <Var name> [Unit]; ...

\$	Time stamp	;	Value	;	Value	;	Value	;	Value	;	...	;
----	------------	---	-------	---	-------	---	-------	---	-------	---	-----	---

23.11.10_10:00:51:159;28;34;230; // First Line
23.11.10_10:00:52:150;;;232; // First Line

Binary_C2C_LE

Log File header: ----- LOGFILE_HEADER LogFile;Test_Logger LogFileFormat;4 Comment;Test log job for Test MaxSize;10485760 Date+Time;22.11.11_13:30:53 FileInfo: Timestamp:1 DP :00;Test Lst:00;Engine Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:% DATASTART

START_TAG (uint16)	BlockLength (uint16)	Datainfo (uint16)	Datum in sec since 1970 (uint32)	DP List ID (uint8)	DP List Var ID (uint8)	Datum (Datum Length * uint8)
-----------------------	-------------------------	----------------------	--	--------------------------	------------------------------	------------------------------------

Datainfo:

BIT 0 1

0 = time triggered
1 = event triggered
2 = time or event triggered

BIT 2 3

0 = no timestamp
1 = RTC
2 = user defined timestamp

BIT 4

0 = C2C Format

Binary_TC3_LE

```
Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;5
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp:1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART
```

START_TAG (uint16)	BlockLength (uint16)	Datainfo (uint16)	Datum in sec since 1970 (uint32)	msec (uint16)	not used	DP ID (uint8)	DP List ID (uint8)	DP List Var ID (uint8)	Datum (Datum Length * uint8)
-----------------------	-------------------------	----------------------	--	------------------	-------------	------------------	--------------------------	---------------------------------	---------------------------------------

Datainfo:

BIT 0 1

0 = time triggered
1 = event triggered
2 = time or event triggered

BIT 2 3

0 = no timestamp
1 = RTC
2 = user defined timestamp

BIT 4
1 = TC3 Format

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_create_log_job (&mt_DBUS_Util, "TC3_Logger", 1, 1024*1024, 1, "Test log
job", 500);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.2 ylogd_add_log_variable

Function Description

```
sint32 ylogd_add_log_variable (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_LogJobName , const charn * const opcn_DatapoolName, const charn * const
opcn_ListName, const charn * const opcn_VariableName, const uint32 ou32_Hysteresis)
```

The function adds a variable to be logged as part of a data logging record of the data logger job opcn_LogJobName. The value will only be written to the log file in case it differs more than +/- ou32_Hysteresis from its previous record.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to log job name
opcn_DatapoolName	maximum MAX_DP_NAME_LENGTH	pointer to the data pool name
opcn_ListName	maximum MAX_DP_NAME_LENGTH	pointer to the list name
opcn_VariableName	maximum MAX_DP_NAME_LENGTH	pointer to the variable name for logging
ou32_Hysteresis	uint32	includes the logging hysteresis

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
```

```

    charn          acn_myNameString[128]; // Name of the application, Don't use any
special           // characters, white spaces or new lines!

    charn          acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error variable successfully added to the logger file (.dlc)
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error or wrong parameters
C_NOACT	Error occurs because logger job, data pool, variable list or variable name is invalid, received values are corrupt, data logger configuration does not exist or is corrupt or logger already initialized
C_RANGE	Parameters are out of range

Example

```

// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_add_log_variable (&mt_DBUS_Util, "TC3_Logger", "MyDatapool", "GPSDates",
"Longitude", 5);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.3 ylogd_set_variable_change_cmd

Function Description

```

sint32 ylogd_set_variable_change_cmd (const T_DBUS_Util * const opt_DBusInstance,
                                     const charn * const opcn_LogJobName,
                                     const charn * const opcn_DatapoolName,
                                     const charn * const opcn_ListName,
```

```
const charn * const opcn_VariableName,
const charn * const opcn_Command)
```

The function adds a new variable event command to expected log job opcn_LogJobName.

If a command for that variable already exists, it will be replaced.

When the given variable logging due to reaching its hysteresis threshold, the given command will be executed. If the command contains the string defined by FORMATTER_OLDVALUE or FORMATTER_NEWVALUE, then those portions of the command will be replaced by the previously logged value and the currently logged value, respectively.



NOTE:

This command should only be used when a hysteresis > 0 is used, and when the frequency of the change is expected to be low. Otherwise, the log job may not be able to maintain the sample rate.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to log job name
opcn_DatapoolName	maximum MAX_DP_NAME_LENGTH	pointer to the data pool name
opcn_ListName	maximum MAX_DP_NAME_LENGTH	pointer to the list name
opcn_VariableName	maximum MAX_DP_NAME_LENGTH	pointer to the variable name for logging
opcn_Command	uint32, maximum size of MAX_DL_COMMAND_LENGTH	pointer to command string to execute

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
```

```

charn          acn_myCMDOnWatchdog[2048]; // expects to be triggered
// Bash cmd that must be executed
// by the ysysd in case no
// trigger signal occurred in time.
} T_DBUS_Util;

```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error event condition was successfully set
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error or wrong parameters
C_NOACT	Error occurs because: <ul style="list-style-type: none"> log job name invalid log job does not exist log job active, variable does not exist in data pool cannot update log job
C_RANGE	Parameters are out of range

7.3.6.2.4 ylogd_set_log_mechanism

Function Description

```

sint32 ylogd_set_log_mechanism (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_LogJobName, const uint8 ou8_LogMechanism)

```

The function sets the log mechanism for the selected logger specified in name opcn_LogJobName.



NOTE:

For an illustration of the logging sequence see How the logger mechanism works (see "[Introduction](#)" on page 203).

It's possible to select one of the following log mechanisms:

TIME_TRIGGERED:

Every ou32_LoggingInterval which can be set via ylogd_set_trigger_time (see "[ylogd_set_trigger_time](#)" on page 214) function, a dataset will be written to the datalogger log file. In TIME_TRIGGERED mode the ou32_SampleTime, which can be set via the function ylogd_create_log_job (see "[ylogd_create_log_job](#)" on page 205), shall be set to ou32_LoggingInterval time.

TIME_AND_EVENT_TRIGGERED:

A dataset will be written to the datalogger log file, when the selected trigger time ou32_LoggingInterval is reached and the chosen event condition(s) is(are) true. The trigger time can be set via the function ylogd_set_trigger_time (see "[ylogd_set_trigger_time](#)" on page 214) and a condition can be added via the function ylogd_set_event_condition (see "[ylogd_set_save_condition](#)" on page 220).

TIME_OR_EVENT_TRIGGERED:

A dataset will be written to the datalogger log file, when the selected trigger time ou32_LoggingInterval is reached or the chosen event condition(s) is(are) true. If the event condition(s) is(are) true, the trigger time value will be set to ou32_SampleTime until the event condition(s) is(are) false. The trigger time can be set via the function ylogd_set_trigger_time (see "[ylogd_set_trigger_time](#)" on page 214), a condition can be added via the function ylogd_set_event_condition (see "[ylogd_set_save_condition](#)" on page 220) and the sample time can be set via the function ylogd_create_log_job (see "[ylogd_create_log_job](#)" on page 205).

EVENT_TRIGGERED:

In EVENT_TRIGGERED mode, a dataset will be written, when the condition(s) is(are) true. In EVENT_TRIGGERED mode, the trigger time will be automatically set to ou32_SampleTime. A condition can be added via the function ylogd_set_event_condition (see "[ylogd_set_save_condition](#)" on page 220) and the sample time can be set via the function ylogd_create_log_job (see "[ylogd_create_log_job](#)" on page 205).



NOTE:

If no log mechanism is set, the default behavior is "TIME_TRIGGERED".

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the data log job name
ou8_LogMechanism	-TIME_TRIGGERED (1) -TIME_AND_EVENT_TRIGGERED(2) -TIME_OR_EVENT_TRIGGERED (3) -EVENT_TRIGGERED (4)	Value for the chosen logging mechanism. Default value is "TIME_TRIGGERED"

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn          acn_myNameString[128];  // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn          acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];     // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;     // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, trigger time was activated successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error
C_RANGE	ou8_LogMechanism is out of range
C_NOACT	Error occurred because the logger job name was invalid, data logger configuration does not exist, DLC file is corrupt or the logger already exists

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_set_log_mechanism (&mt_DBUS_Util, "TC3_Logger", TIME_TRIGGERED);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.5 ylogd_set_trigger_time Function Description

```
sint32 ylogd_set_trigger_time (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_LogJobName, const uint32 ou32_LoggingInterval)
```

The function sets the time interval between two data logging records. The logging interval ou32_Logging has to be greater than or equal to the sample time. The sample time is set in the function ylogd_create_log_job (see ["ylogd_create_log_job"](#) on page 205).



WARNING:

If the trigger time is not set, the log mechanism can not be started. The only exception for this rule is simple event triggered logging. In that case the trigger time will be automatically set to the sample time.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Parameter	Range	Description
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the data log job name
ou32_LoggingInterval	1 .. uint32	<ul style="list-style-type: none"> includes trigger time in ms value has to be >= sample time parameter from ylogd_create_log_job (see "ylogd create log job" on page 205)

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, trigger time was activated successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error
C_RANGE	opcn_LogJobName is out of range
C_NOACT	Error occurred because the logger job name was invalid, trigger time interval is too short, data logger configuration does not exist, DLC file is corrupt or logger already exists

Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
```

```
s32_Retval = ylogd_set_trigger_time (&mt_DBUS_Util, "TC3_Logger", 500);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.6 ylogd_set_buffered_logging

Function Description

```
sint32 ylogd_set_buffered_logging (const T_DBUS_Util * const opt_DBusInstance, const charn
* const opcn_LogJobName, const uint32 ou32_LoggingBufferSize)
```

The function sets the number of records to buffer. All logging data will be sent to this buffer.

If the log is triggered with ylogd_trigger_bufferd_log (see "[ylogd_trigger_buffered_log](#)" on page 241), the buffered data will be placed in the log file, and then normal logging will continue.

When log job is activated, instead of creating a file, entries are stored in a ring buffer, replacing the oldest entry when buffer is full. All normal log entry rules are in effect.



NOTE:

Log job activation will fail if the application is unable to allocate buffer memory.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to log job name
ou32_LoggingBufferSize	maximum MAX_DL_BUFFER_SIZE	0 means buffer disabled Otherwise number of log entries to buffer

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error buffered trigger was successfully set
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error or wrong parameters
C_NOACT	Error occurs because: <ul style="list-style-type: none"> log job name is invalid trigger time interval to short data logger configuration does not exist, DLC file is corrupt log job does not exists log job is active and cannot be updated
C_RANGE	Parameters are out of range

7.3.6.2.7 ylogd_set_master_save_condition

Function Description

```
sint32 ylogd_set_master_save_condition (const T_DBUS_Util *const opt_DBusInstance, const
charn * const opcn_LogJobName, const charn * const opcn_DatapoolName, const charn * const
```

```
opcn_ListName, const charn * const opcn_VariableName, const charn * const opcn_Logic,
const uint32 ou32_Hysteresis, const uint32 ou32_Threshold, const charn * const
opcn_Concatenation)
```

This function adds a new master save condition to the log job opcn_LogJobName:

```
if (VARIABLE ('<=' or '>=' or '==' or '!=') (os32_Threshold +/- os32_Hysteresis))
then
condition is true, value will be logged.
```

opcn_VariableName is the value of opcn_Datapoolname.opcn_ListName.opcn_VariableName.

When a master save condition is defined it will always be concatenated with the previous master save conditions result. The variable to be tested is defined by: opcn_DatapoolName, opcn_ListName and opcn_VariableName

The testing logic is defined by: opcn_Logic which can be '!=', '==', '<=', '>='

The variable is checked against the following parameters: os32_Threshold +/- os32_Hysteresis

If there are more than one conditions to be tested, then the result is logically linked with the previous result by opcn_Concatenation ('AND', 'OR' and 'EXOR').



WARNING:

If a master save condition is chosen, the condition has to be true before the regular save condition will be checked.



NOTE:

For an illustration of the logging sequence see How the logger mechanism works (see "[Introduction](#)" on page 203).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the log job name
opcn_DatapoolName	maximum MAX_DP_NAME_LENGTH	pointer to the data pool name
opcn_ListName	maximum MAX_DP_NAME_LENGTH	pointer to the variable list name
opcn_VariableName	maximum MAX_DP_NAME_LENGTH	pointer to the variable name
opcn_Logic	maximum MAX_DL_LOGIC_LENGTH	pointer to the logical operator like '<=', '>=', '!=', '=='

Parameter	Range	Description
ou32_Hysteresis	uint32	pointer to the hysteresis of the variable we expect to log
ou32_Threshold	uint32	pointer to the logging threshold for the variable we expect to log
opcn_Concatenation	maximum MAX_DL_CONC_LENGTH	pointer to concatenation options like 'AND', 'OR' and 'EXOR'

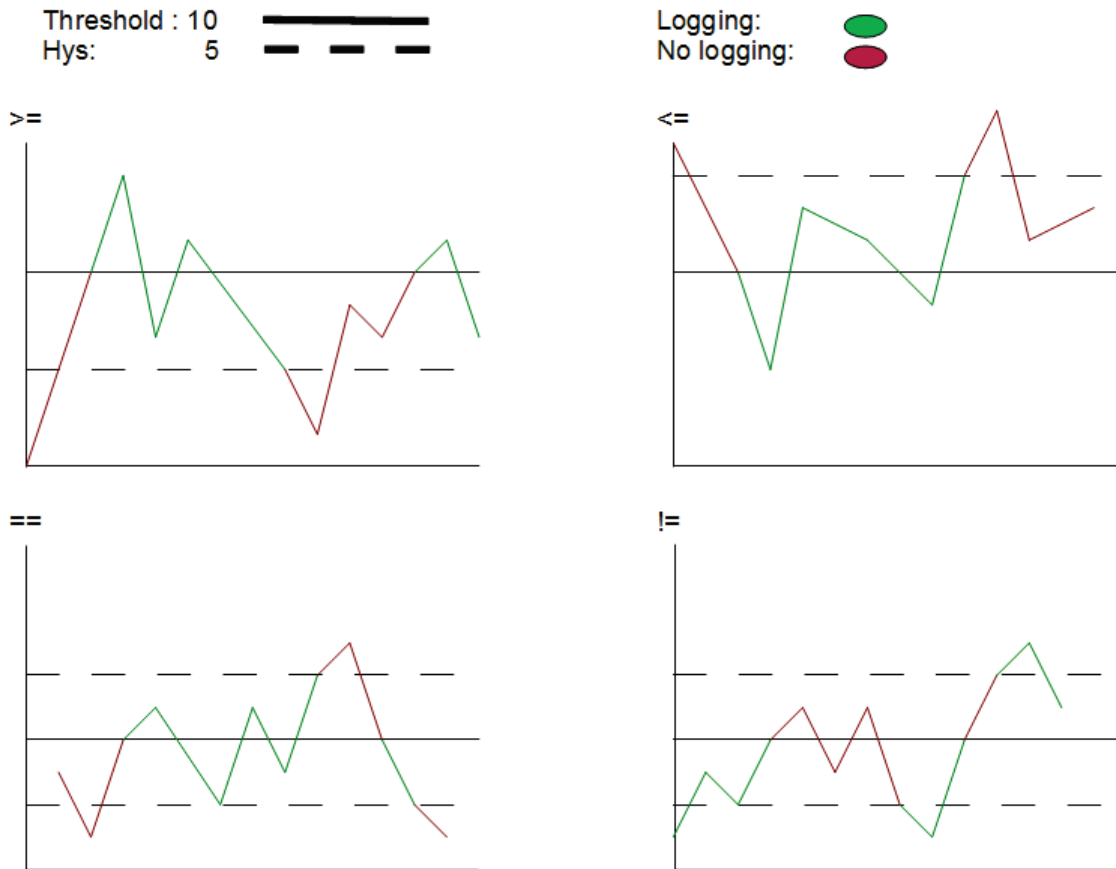
Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];   // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;   // Time interval when the ysysd
                                          // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                          // by the ysysd in case no
                                          // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, event condition was set successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	Wrong or existing job, dp, list, var name or wrong parameter for concatenation, logic or NULL pointer
C_NOACT	Error occurred because the logger job, data pool, variable list or variable name invalid, logic, or concatenation operator was invalid, received values corrupt, data logger configuration does not exist, DLC file corrupt or logger already initialized
C_RANGE	Parameters are out of range

Association between logic, hysteresis and threshold



Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_set_master_save_condition (&mt_DBUS_Util, "TC3_Logger", "MyDatapool",
"GPSDates", "Longitude", ">=", 5, 10, "AND");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.8 ylogd_set_save_condition

Function Description

```
sint32 ylogd_set_save_condition (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_LogJobName, const charn * const opcn_DatapoolName, const charn * const
opcn_ListName, const charn * const opcn_VariableName, const charn * const opcn_Logic,
const uint32 ou32_Hysteresis, const uint32 ou32_Threshold, const charn * const
opcn_Concatenation)
```

This function adds a new save condition to the log job opcn_LogJobName:

```
if (VARIABLE ('<=' or '>=' or '==' or '!=') (os32_Threshold +/- os32_Hysteresis))
then
condition is true, value will be logged.
```


When a save condition is already defined the actual save condition will always be concatenated with the previous save conditions result. The Variable which shall be used for the condition has to be set by opcn_DatapoolName, opcn_ListName and opcn_VariableName

The testing logic is defined by: opcn_Logic which can be '!=', '==', '<=', '>='

The variable is checked against the following parameters: os32_Threshold +/- os32_Hysteresis

If there are more than one condition set, then each condition result will be logically linked with his previous result by opcn_Concatenation ('AND', 'OR' and 'EXOR')).


NOTE:

For an illustration of the logging sequence see How the logger mechanism works (see "[Introduction](#)" on page 203).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the log job name
opcn_DatapoolName	maximum MAX_DP_NAME_LENGTH	pointer to the data pool name
opcn_ListName	maximum MAX_DP_NAME_LENGTH	pointer to the variable list name
opcn_VariableName	maximum MAX_DP_NAME_LENGTH	pointer to the variable name
opcn_Logic	maximum MAX_DL_LOGIC_LENGTH	pointer to logical operator like '<=', '>=', '!=', '=='
ou32_Hysteresis	uint32	pointer to the hysteresis of the variable we expect to log
ou32_Threshold	uint32	pointer to the logging threshold for the variable we expect to log
opcn_Concatenation	maximum MAX_DL_CONC_LENGTH	pointer to concatenation options like 'AND', 'OR' and 'EXOR'

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn           acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
}
```

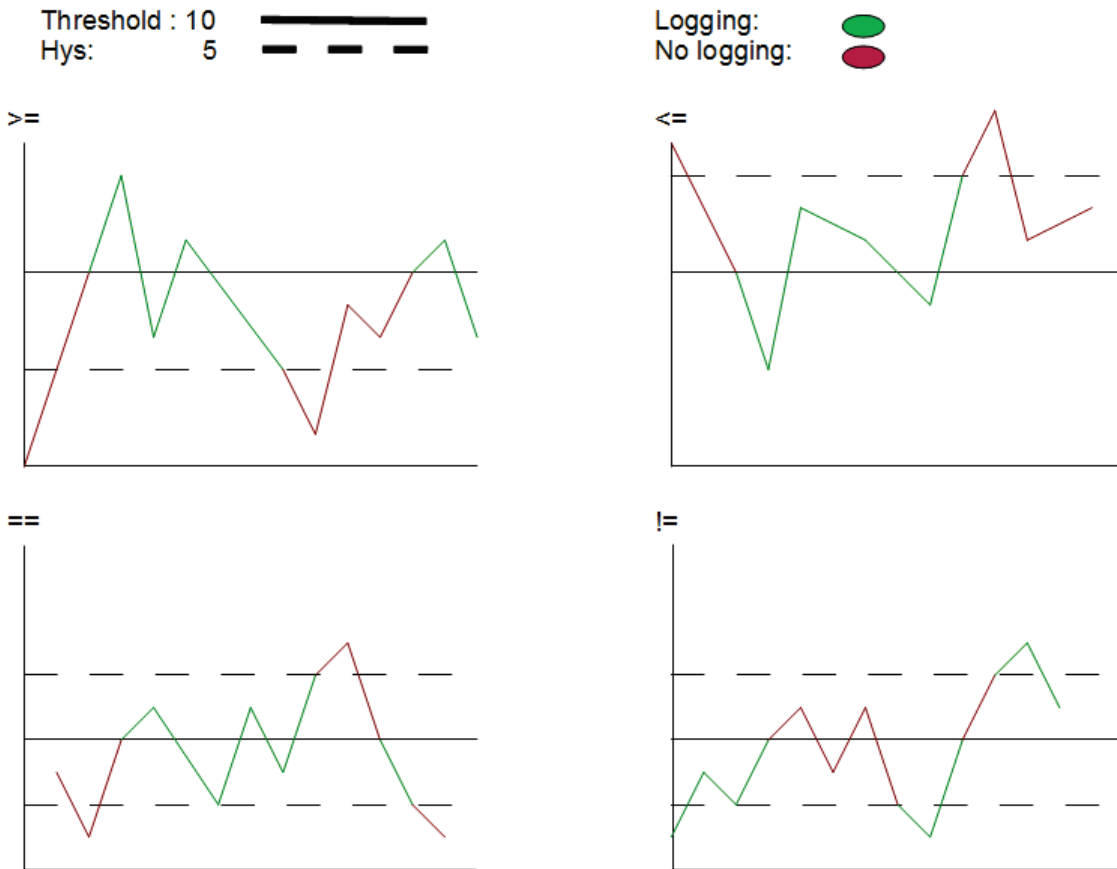
```

charn      acn_myStatus[128];      // Additional value (for HELLO) signal (opt.)
charn      acn_myAddInfo[512];     // Additional value (for HELLO) signal (opt.)
sint32     s32_myTriginterval;     // Time interval when the ysysd
                                   // expects to be triggered
charn      acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                   // by the ysysd in case no
                                   // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, event condition was set successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	Wrong or existing job, dp, list, var name or wrong parameter for concatenation, logic or NULL pointer
C_NOACT	Error occurred because logger job, data pool, variable list, or variable name was invalid, logic or concatenation operator invalid, received values corrupt, data logger configuration does not exist, DLC file corrupt or logger already initialized
C_RANGE	Parameters are out of range

Association between logic, hysteresis and threshold



Example

```
// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_set_save_condition (&mt_DBUS_Util, "TC3_Logger", "MyDatapool",
"GPSDates", "Longitude", ">=", 5, 10, "AND");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.6.2.9 ylogd_set_log_management

Function Description

```
sint32 ylogd_set_log_management(const T_DBUS_Util * const opt_DBusInstance,
                                const charn * const opcn_LogJobName,
                                const charn * const opcn_LogDestination,
                                const uint32 ou32_LogDuration,
                                const uint32 ou32_LogCompression,
                                const uint32 ou32_LogFolderSize,
                                const charn * const opcn_LogFileExtension);
```

The function configures the log management settings of a log job.

When the log job terminates or a file limit is reached, a resulting log file will be placed in the destination directory. The file name will be "LOGJOB_START_STOP.EXT[.gz]".

- LOGJOB - opcn_LogJobName
- START – time of file start YYYYMMDDHHmmss
- STOP – time of file stop YYYYMMDDHHmmss
- EXT – ".dlf" or opcn_LogFileExtension
- [.gz] – optional extension if compression is used

If the folder size limit is reached, the oldest log files in the folder will be removed until the folder size limit is maintained.

If the destination folder does not exist, the system will attempt to create it.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to log job name that shall be activated
opcn_LogDestination	maximum MAX_DL_NAME_LENGTH	destination folder, logs will be copied to this folder when the LogDuration or Jobsizes is reached. If folder is " ", then no copy will occur.
ou32_LogDuration		minutes in size of log file, 0 means new file at midnight
ou32_LogCompression		0 no compression, 1 gzip compression

Parameter	Range	Description
ou32_LogFolderSize		if folder size reaches this level, older logs will be deleted
opcn_LogFileExtension	maximum MAX_DL_NAME_LENGTH	Extension of the logger file.

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128]; // Name of the application, Don't use any
    special                                                // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                                // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                // by the ysysd in case no
                                                // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error activating log job successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	Wrong value or NULL pointer
C_NOACT	Error occurs because: <ul style="list-style-type: none"> • logger job name • start/stop flag invalid, • data logger configuration does not exist • no data logger running, • DLC file corrupt • logger already initialized • can not initialize data pool configuration

Return Value	Description
	<ul style="list-style-type: none"> can not read data logger configuration, can not allocate space can not start child process can not get on the bus
C_RANGE	opcn_LogJobName is out of range

7.3.6.2.10 ylogd_set_log_file_cmd

Function Description

```
sint32 ylogd_set_log_file_cmd (const T_DBUS_Util * const opt_DBusInstance,
                               const charn * const opcn_LogJobName,
                               const charn * const opcn_Command)
```

The function the log file completion command.

When a log file is created due to log file management, the given command will be executed. If the command contains the string defined by FORMATTER_FILENAME, then that portion of the command will be replaced by the full path name of the created log file.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to log job name
opcn_Command	maximum MAX_DL_COMMAND_LENGTH	pointer to the command to execute

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
```

```

    charn          acn_myCMDOnWatchdog[2048]; // expects to be triggered
                                                    // Bash cmd that must be executed
                                                    // by the ysysd in case no
                                                    // trigger signal occurred in time.
} T_DBUS_Util;

```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error buffered trigger was successfully set
C_COM	D-Bus communication error
C_UNKNOWN_ERR	NULL pointer error or wrong parameters
C_NOACT	Error occurs because: <ul style="list-style-type: none"> logger job name is invalid log management has not been configured yet
C_RANGE	opcn_LogJobName is out of range

7.3.6.2.11 ylogd_activate_log_job

Function Description

```

sint32 ylogd_activate_log_job (const T_DBUS_Util *const opt_DBusInstance, const charn *
const opcn_LogJobName, const uint32 ou32_OnOff)

```

The function starts (ou8_OnOff != 0) and stops (ou8_OnOff == 0) the log job opcn_LogJobName.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_LogJobName	maximum MAX_DL_NAME_LENGTH	pointer to the log job name which shall be activated
ou32_OnOff	0 = Off , 1 = On	includes the OnOff flag (0 = off 1 = on)

Structure T_DBUS_Util

```

typedef struct
{
    DBusConnection* pt_dbus_conn; // D-BUS connection instance

```

```

    charn          acn_myNameString[128]; // Name of the application, Don't use any
special           // characters, white spaces or new lines!

    charn          acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;

```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, log job activate successfully
C_COM	D-Bus communication error
C_UNKNOWN_ERR	Wrong or not existing job
C_NOACT	Error occurs because logger job name or start/stop flag invalid, data logger configuration does not exist, no data logger running, DLC file corrupt, logger already initialized, cannot initialize data pool configuration, cannot read data logger configuration, can not allocate space, can not start child process, cannot get on the bus
C_RANGE	opcn_LogJobName is out of range

Example

```

// Global module
T_DBUS_Util mt_DBUS_Util;
sint32 s32_Retval;
s32_Retval = ylogd_activate_log_job (&mt_DBUS_Util, "TC3_Logger", 1);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}

```


7.3.6.3 Static mode

7.3.6.3.1 Basic DLC file structure

Every datalogger configuration file has the following basic components:

Necessary DLC file objects

Parameter	Description
[JOBOPTIONS]	Start tag of the DLC file
ACTIVE	Active flag will be automatically set to 1 by ylogd after correct file parsing
LFJNAME	Name of the log file job
LFFORMAT	Format of the data logger file: 1 : ASCII_STD 2 : ASCII_TC1 3 : ASCII_CSV 4 : BINARY_C2C_LE 5 : BINARY_TC3_LE
MEMSIZE	Maximum size of the data logger file in bytes (Maximum value is 500 MB)
BASIC_SAMPLE_TIME	Time interval in which is checked whether the values have changed
TIMESTAMP	Time stamp at the beginning of every new logged line in the data logger file
[FILEHEADER]	Start tag of the log-file
COMMENT	Comment
[SAMPLECONDITIONS]	Start tag of the values which shall be logged chapter
NUMCONDITIONS	Number of values which shall be logged
[MASTERSAVECONDITIONS]	Start of the chapter for master save conditions. See also How the logger mechanism works (see " Introduction " on page 203).
NUMMASTERSAVECONDITIONS	Number of master save conditions which must be valid before the "normal" save conditions are checked
[SAVECONDITIONS]	Start of the chapter for save conditions. See also How the logger mechanism works (see " Introduction " on page 203)
LOG_MECHANISM	Log mechanism modes:

Parameter	Description
	<ul style="list-style-type: none"> TIME_TRIGGERED (1) Every TRIGGERTIME, a dataset will be written to the datalogger log file. In TIME_TRIGGERED mode, the BASIC_SAMPLE_TIME shall be equal to TRIGGERTIME TIME_AND_EVENT_TRIGGERED (2) A dataset will be written to the datalogger log file when the selected TRIGGERTIME is reached and the chosen save condition(s) is(are) true. TIME_OR_EVENT_TRIGGERED (3) A dataset will be written to the datalogger log file when the selected TRIGGERTIME is reached or the chosen save condition(s) is(are) true. If the save condition(s) is(are) true, the TRIGGERTIME value will be set to BASIC_SAMPLE_TIME until the save condition(s) is(are) false. EVENT_TRIGGERED (4) In EVENT_TRIGGERED mode, a dataset will be written when the condition(s) is(are) true. In EVENT_TRIGGERED mode, the TRIGGERTIME will be automatically set to BASIC_SAMPLE_TIME. <hr/> <p>In case of a missing LOG_MECHANISM entry, the default behavior is "TIME_TRIGGERED"</p>
TRIGGERTIME	Time interval in ms, after a new line to the data logger file shall be written if the save conditions are valid
NUMSAVECONDITIONS	Number of save conditions which must be valid before a line shall be written to the data logger file

Logfile format description

ASCII_STD

```

Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;1
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp:1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART

```

Time stamp	;	Index	;	VAR Name	;	Value	;	Index	;	VAR Name	;	Value	;	...	;
23.11.10_10:00:51:159;[0x000003];Height;688;[0x000005];Course;327;[0x000100];Temperature;25; // First line															
23.11.10_10:00:52:150;[0x000100];Temperature;25; // Second line															

ASCII_TC1

```
Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;2
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp:1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART
```

\$	Time stamp in sec since 2000	;	Value	;	Value	;	Value	;	Value	;	...	;
----	------------------------------	---	-------	---	-------	---	-------	---	-------	---	-----	---

\$540000;28;755;Test;28;56332; // First line

\$540001;;;Test2;29;;; // Second line

ASCII_CSV

```
Log File header:
-----
TIMESTAMP;<DP name> > <DPL name> > <Var name> [Unit];<DP name> > <DPL name> > <Var name>
[Unit]; ...
```

\$	Time stamp	;	Value	;	Value	;	Value	;	Value	;	...	;
----	------------	---	-------	---	-------	---	-------	---	-------	---	-----	---

23.11.10_10:00:51:159;28;34;230; // First Line

23.11.10_10:00:52:150;;;232; // First Line

Binary_C2C_LE

```
Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;4
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp:1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART
```

START_TAG (uint16)	BlockLength (uint16)	Datainfo (uint16)	Datum in sec since 1970 (uint32)	DP List ID (uint8)	DP List Var ID (uint8)	Datum (Datum Length * uint8)
-----------------------	-------------------------	----------------------	--	--------------------------	------------------------------	------------------------------------

Datainfo:

BIT 0 1

0 = time triggered

1 = event triggered

2 = time or event triggered

BIT 2 3

0 = no timestamp

1 = RTC

2 = user defined timestamp

BIT 4

0 = C2C Format

Binary_TC3_LE

```
Log File header:
-----
LOGFILE_HEADER
LogFile;Test_Logger
LogFileFormat;5
Comment;Test log job for Test
MaxSize;10485760
Date+Time;22.11.11_13:30:53
FileInfo:
Timestamp:1
DP :00;Test
Lst:00;Engine
Var:00;EngineSpeed;Type:SINT32;Size:4;Unit:rpm
Var:01;InjectionQuantity;Type:SINT32;Size:4;Unit:mm3/H
Var:02;EnginePowerReserve;Type:SINT32;Size:4;UNIT:%
DATASTART
```

START_TAG (uint16)	BlockLength (uint16)	Datainfo (uint16)	Datum in sec since 1970 (uint32)	msec (uint16)	not used	DP ID (uint8)	DP List ID (uint8)	DP List Var ID (uint8)	Datum (Datum Length * uint8)
-----------------------	-------------------------	----------------------	--	------------------	-------------	------------------	--------------------------	---------------------------------	---------------------------------------

Datainfo:

BIT 0 1

0 = time triggered

1 = event triggered

2 = time or event triggered

BIT 2 3

0 = no timestamp

1 = RTC

2 = user defined timestamp

BIT 4

1 = TC3 Format

Example

```
[JOBOPTIONS]
ACTIVE=0
LFJNAME=Logger1
LFFORMAT=1
MEMSIZE=1048576
BASIC_SAMPLE_TIME=10
TIMESTAMP=0
[FILEHEADER]
COMMENT=Test log job
[SAMPLECONDITIONS]
NUMCONDITIONS=0
[MASTERSAVECONDITIONS]
```

```
NUMMASTERSAVECONDITIONS=0
[SAVECONDITIONS]
LOG_MECHANISM=1
TRIGGERTIME=0
NUMSAVECONDITIONS=0
```

7.3.6.3.2 Add variable to DLC file

The following variable objects have to be added to the DLC file in order to add a variable:

Variable objects

Parameter	Description
[SAMPLECONDITION1-n]	Start tag which contains the actual number of the variable which shall be logged
NAME	Name of the variable
DATAPOOL	Name of the data pool, which contains the list and the variable name
LIST	Name of the list which contains the variable name
SAVEHYSTERESIS	The variable is only written to the datalogger if the value changes since it has been stored for the last time. The first dataset in a new file and the first dataset after starting up the device will always contain all variables

Example

```
[JOBOPTIONS]
ACTIVE=0
LFJNAME=Logger1
LFFORMAT=1
MEMSIZE=1048576
BASIC_SAMPLE_TIME=10
TIMESTAMP=0
[FILEHEADER]
COMMENT=Test log job
[SAMPLECONDITIONS]
NUMCONDITIONS=2
[SAMPLECONDITION1]
NAME=Logitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[SAMPLECONDITION2]
NAME=Latitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[MASTERSAVECONDITIONS]
NUMMASTERSAVECONDITIONS=0
[SAVECONDITIONS]
LOG_MECHANISM=1
TRIGGERTIME=0
NUMSAVECONDITIONS=0
```

7.3.6.3.3 Add master save condition to DLC file

The following variable objects have to be added to the DLC file in order to add a master save condition:

Variable objects

Parameter	Description
[MASTERSAVECONDITION1-n]	Start tag which contains the actual number of the save condition
VARIABLENAME	Name of the master save condition variable
DATAPOOL	Name of the data pool, which contains the list and the master save condition variable name
LIST	Name of the list which contains the master save condition variable name
LOGIC	<=, >=, ==, !=
HYSTERESIS	Hysteresis
THRESHOLD	Threshold
CONCATENATION	AND, OR, XOR

Explanation

This means that a master save condition is defined that is concatenated with the previous event conditions result. A variable to be tested is defined by: DATAPOOL, LIST and VARIABLENAME.

The testing logic is defined by LOGIC which can be "!=" , "==" , "<=" , ">="

The value the variable is tested against is defined by:

THRESHOLD+/- HYSTERESIS

If there are more than one conditions to be tested, then it's result is logically linked with its previous result by CONCATENATION.

CONCATENATION can be "AND", "OR", XOR".



WARNING:

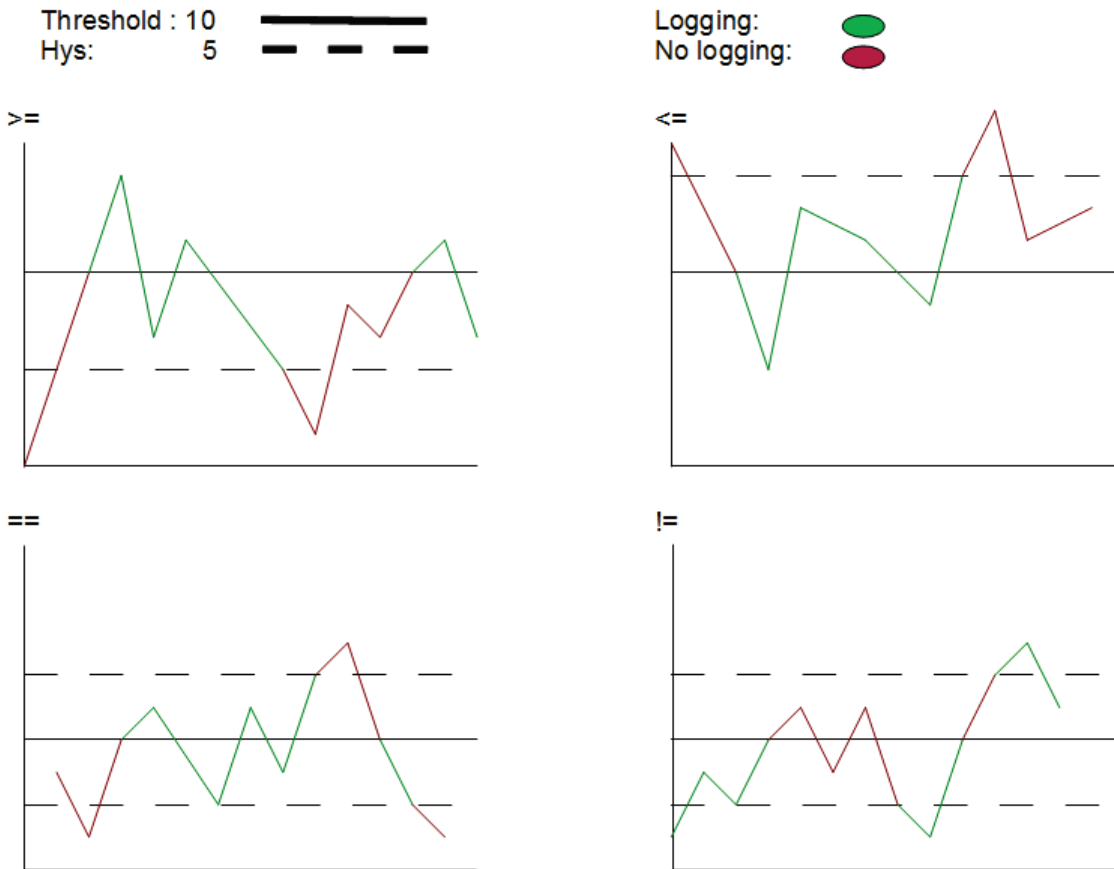
If a master save condition is chosen, the condition has to be true before the regular save condition will be checked.



NOTE:

See also How the logger mechanism works (see "[Introduction](#)" on page 203).

Association between logic, hysteresis and threshold



Example DLC file

```
[JOBOPTIONS]
ACTIVE=0
LFJNAME=Logger1
LFFORMAT=1
MEMSIZE=1048576
BASIC_SAMPLE_TIME=10
TIMESTAMP=0
[FILEHEADER]
COMMENT=Test log job
[SAMPLECONDITIONS]
NUMCONDITIONS=2
[SAMPLECONDITION1]
NAME=Logitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[SAMPLECONDITION2]
NAME=Latitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[MASTERSAVECONDITIONS]
NUMMASTERSAVECONDITIONS=1
[MASTERSAVECONDITION1]
VARIABLENAME=EngineSpeed
DATAPOOL=MyDatapool
LIST=Engine
LOGIC=>=
HYSTERESIS=50
```

```
THRESHOLD=1000  
CONCATENATION=OR  
[SAVECONDITIONS]  
LOG_MECHANISM=1  
TRIGGERTIME=0  
NUMSAVECONDITIONS=0
```


7.3.6.3.4 Add save condition to DLC file

The following variable objects have to be added to the DLC file in order to add a save condition:

Variable objects

Parameter	Description
[SAVECONDITION1-n]	Start tag which contains the actual number of the save condition
VARIABLENAME	Name of the save condition variable
DATAPOOL	Name of the data pool, which contains the list and the save condition variable name
LIST	Name of the list which contains the save condition variable name
LOGIC	<=, >=, ==, !=
HYSTERESIS	Hysteresis
THRESHOLD	Threshold
CONCATENATION	AND, OR, XOR

Explanation

This means that a save condition is defined that is concatenated with the previous event conditions result. A variable to be tested is defined by: DATAPOOL, LIST and VARIABLENAME.

The testing logic is defined by LOGIC which can be "!=" , "==" , "<=" , ">="

The value the variable is tested against is defined by:

THRESHOLD+/- HYSTERESIS

If there are more than one conditions to be tested, then it's result is logically linked with its previous result by CONCATENATION.

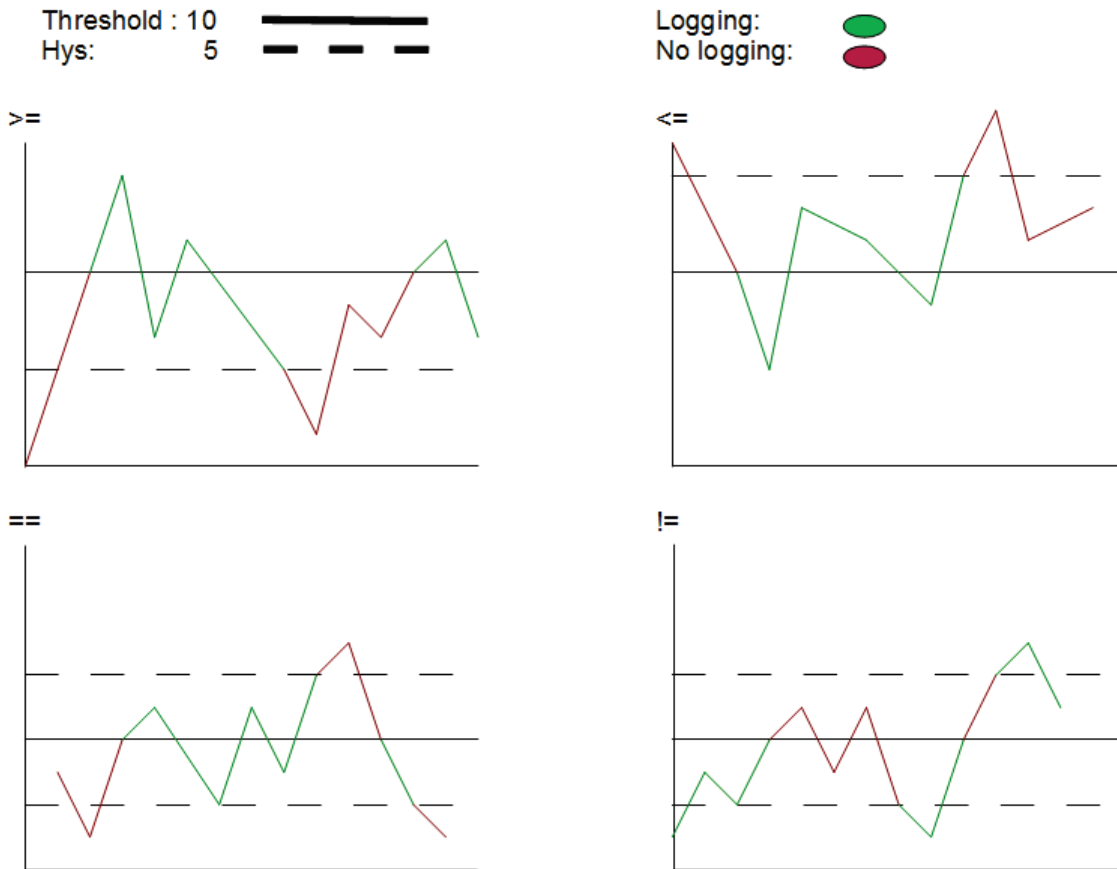
CONCATENATION can be "AND", "OR", XOR".



NOTE:

See also How the logger mechanism works (see "[Introduction](#)" on page 203).

Association between logic, hysteresis and threshold



Example DLC file

```
[JOBOPTIONS]
ACTIVE=0
LFJNAME=Logger1
LFFORMAT=1
MEMSIZE=1048576
BASIC_SAMPLE_TIME=10
TIMESTAMP=0
[FILEHEADER]
COMMENT=Test log job
[SAMPLECONDITIONS]
NUMCONDITIONS=2
[SAMPLECONDITION1]
NAME=Logitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[SAMPLECONDITION2]
NAME=Latitude
DATAPOOL=MyDatapool
LIST=GPSDates
SAVEHYSTERESIS=10
[MASTERSAVECONDITIONS]
NUMMASTERSAVECONDITIONS=0
[SAVECONDITIONS]
LOG_MECHANISM=1
TRIGGERTIME=1000
NUMSAVECONDITIONS=1
[SAVECONDITION1]
VARIABLENAME=EngineSpeed
```

```
DATAPOOL=MyDatapool
LIST=Engine
LOGIC=>=
HYSTERESIS=50
THRESHOLD=1000
CONCATENATION=OR
```

7.3.6.4 Handle Logger File

Header file: "DL_LogFile_handler.h"

This header provides utility functions to use the logger daemon to handle logger files.

7.3.6.4.1 ylogd_request_data_logger_file

Function Description

```
sint32 ylogd_request_data_logger_file (const T_DBUS_Util *const opt_DBusInstance, const
char * const opcn_DataLoggerName, char * const opcn_DataLoggerPath, const uint16
ou16_BufferSize)
```

The function requests the datalogger file opcn_DataLoggerName. If the logger file exists, the path to a copy (tmp file) will be responded via the variable opcn_DataLoggerPath. The maximal length of the path to the datalogger copy is defined via ou16_BufferSize.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_DataLoggerName	maximum MAX_DL_NAME_LENGTH	pointer to the name of the datalogger
ou16_BufferSize	minimum MAX_DL_BASEFOLDER_LENGTH	maximum buffer size of datalogger path

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Parameter	Range	Description
opcn_DataLoggerPath	ou16_BufferSize	contains the path to the datalogger file copy

Return Value	Description
C_NO_ERR	file was found, logger name fits and file pointer was set successfully
C_CONFIG	no file found or logger name is incorrect
C_UNKNOWN_ERR	NULL pointer error
C_RANGE	Parameters out of range

Example

```
sint32 s32_Retval = C_NO_ERR;
charn acn_DLPath[500];
s32_Retval = ylogd_request_data_logger_file (&gtgt_dbus_util_child, opcn_DLName,
acn_DLPath[500], sizeof(acn_DLPath));
if (s32_Return == C_NO_ERR)
{
    // Go on
}
```

7.3.6.4.2 ylogd_delete_log_file

Function Description

```
sint32 ylogd_delete_log_file (const T_DBUS_Util *const opt_DBusInstance, const charn * const
opcn_DataLoggerName, const uint8 ou8_DeleteAll)
```

The function deletes the logger opcn_DataLoggerName. The flag ou8_DeleteAll indicates if only the tmp file shall be deleted (tmp file is the file which can be used for copying to server or to a extern flash via USB) or as well the file in which the data will be actually logged.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_DataLoggerName	maximum MAX_DL_NAME_LENGTH	pointer to the name of the logger file
ou8_DeleteAll	0 ...1	0 ... deletes just the temporary file 1 ... deletes both files, temporary an current data logger file

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!

    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	file / files was / were found and deleted
C_CONFIG	flag parameter not correct
C_UNKNOWN_ERR	NULL pointer
C_RANGE	opcn_DataLoggerName out of range

Example

```
sint32 s32_Retval = C_NO_ERR;
s32_Retval = ylogd_delete_log_file (&gt_dbus_util_child, acn_LogFileName, 1);
if(s32_Return != C_NO_ERR)
{
    / Error handling
}
```

7.3.6.4.3 ylogd_trigger_buffered_log

Function Description

```
sint32 ylogd_trigger_buffered_log(const T_DBUS_Util * const opt_DBusInstance, const charn *
const opcn_DataLoggerName)
```

The function causes an activated, buffered log job to flush its ring buffer to the log file and then continue operation as normal.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_DataLoggerName	maximum MAX_DL_NAME_LENGTH	pointer to the name of the logger file

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn           acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	file / files was / were found and deleted
C_CONFIG	flag parameter not correct, log job not buffered
C_UNKNOWN_ERR	NULL pointer
C_RANGE	opcn_DataLoggerName out of range

7.3.7 GPS

Header file: "GPS_handler.h"

7.3.7.1 Introduction

The GPS functionality from the TAF library uses the ygpsd daemon to provide position, date, time, height and actual speed from the integrated GPS receiver.

7.3.7.2 ygpsd_get_gps_data

Function Description

```
sint32 ygpsd_get_gps_data (const T_DBUS_UTIL * const opt_DBusInstance, T_DBUS_GPS_Data *
const opt_DBusGpsData)
```

The function sends a method call on the D-Bus to the ygpsd opt_DBusInstance to receive GPS data. This received GPS data are stored in the structure opt_DBusGpsData.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                           // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Result	Range	Description
opt_DBusGpusData	T_DBUS_GPS_Data	holds the GPS data information

Structure T_DBUS_GPS_Data



NOTE:

This structure is only valid in combination with ygpsd version 4.00r0 or newer

```
typedef struct
{
    charn acn_status[32];          // NONE = fix not available,
                                   // GPS  = GPS fix,
                                   // DGPS = Differential GPS fix,
                                   // PPS  = PPS fix,
                                   // RTK  = Real Time Kinematic,
                                   // FRTK = Float RTK,
                                   // NMEA = acn_quatily > 5
    charn acn_latitude[32];        // Degrees notation: DD.dddddd
    charn acn_longitude[32];       // Degrees notation: DD.dddddd
    charn acn_altitude[32];        // Above/below mean-sea-level in meters
    charn acn_time[32];            // hhmmss
    charn acn_satellites[32];      // number of satellites used in solution
    charn acn_quality[32];         // 0 = fix not available,
                                   // 1 = GPS fix,
                                   // 2 = Differential GPS fix
                                   // 3 = PPS fix
                                   // 4 = Real Time Kinematic
                                   // 5 = Float RTK
                                   // 6 = estimated (dead reckoning)
                                   // 7 = Manual input mode
                                   // 8 = Simulation mode
    charn acn_warn[32];            // V = GPS position is not valid
                                   // A = GPS position is valid
    charn acn_speed[32];           // Speed over ground km/h XXX.YYYYYY
    charn acn_course[32];          // Degrees (0..360) XXX.YYYYYY
    charn acn_date[32];            // ddmmyy
    charn acn.UTC[64];             // MMDDhhmmYYYY
    charn acn_HDOP[32];            // Horizontal dilution of precision
    charn acn_AgeOfDGPS[32];       // Age of differential GPS data in seconds
} T_DBUS_GPS_Data;
```

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call to the daemon
C_COM	Error receiving the method call answer from the daemon
C_RANGE	One of the arguments are NULL pointers

Example

```
static sint32 ms32_RequestGPSData(const T_DBUS_Util* const opt_DBusInstance)
{
    T_DBUS_GPS_Data t_DBus_GPS_Data;
    // Call dbus util function
    if(ygpsd_get_gps_data (opt_DBusInstance, &t_DBus_GPS_Data) != 0)
    {
        printf("Error fetching GPS data!\n");
        return C_COM;
    }
    // To something with the received values
    return C_NO_ERR;
}
```

7.3.8 SMS

Header file: "SMS_handler.h"

7.3.8.1 Introduction

The SMS functionality from the TAF library uses the ygsmd daemon to send and receive text messages

SMS can be sent or received either after standard time intervals or immediately. The standard time interval and other parameters can be configured in the ygsmd configuration file (see GSM daemon).

For receiving SMS messages automatically in your application, use the TAF library function `dbus_initialize_request_callbacks` (see "[dbus_initialize_request_callbacks](#)" on page 159).

7.3.8.2 ygsmd_send_sms

Function Description

```
sint32 ygsmd_send_sms (const T_DBUS_Util *const opt_DBusInstance, const charn * const
opcn_PhoneNumber, const charn *const opcn_Message)
```

The function sends a SMS to a subscriber with the number `opcn_PhoneNumber`. The SMS is sent after the standard time interval that is set in the configuration file of the SMS daemon. The maximum message size is 160 characters

Information Flow

Input Information

Parameter	Range	Description
<code>opt_DBusInstance</code>	<code>T_DBUS_Util</code>	holds all D-Bus information
<code>opcn_PhoneNumber</code>	maximal <code>MAX_SMS_NUMBER_LENGTH</code>	pointer to the phone number buffer
<code>opcn_Message</code>	<code>0 ..MAX_SMS_LENGTH</code>	includes the message

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	NULL pointer or memory error
C_COM	D-Bus communication error
C_RANGE	Invalid SMS or phone number length

Example

```
sint32 s32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
s32_Retval = ygsmd_send_sms(&mt_DBUS_Util, "017612341234", "Hello World");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.8.3 ygsmd_send_sms_urgent

Function Description

```
sint32 ygsmd_send_sms_urgent (const T_DBUS_Util *const opt_DBusInstance, const charn * const
opcn_PhoneNumber , const charn *const opcn_Message)
```

The function sends an urgent SMS to the given phone number opcn_PhoneNumber. The standard time interval for sending/receiving SMS, which is set in configuration file, is ignored and the SMS will be sent immediately.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
opcn_PhoneNumber	maximal MAX_SMS_NUMBER_LENGTH	pointer to the phone number buffer
opcn_Message	0 ..MAX_SMS_LENGTH	includes the message

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!

    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered

    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	NULL pointer or memory error
C_COM	D-Bus communication error
C_RANGE	Invalid SMS or phone number length

Example

```
sint32 s32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
s32_Retval = ygsmc_send_sms_urgent(&mt_DBUS_Util, "017612341234", "Hello World");
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.8.4 ygsmd_request_sms_fetch_urgent

Function Description

```
sint32 ygsmd_request_sms_fetch_urgent (const T_DBUS_Util *const opt_DBusInstance)
```

The function triggers the request mode for text message immediately, i.e. it looks immediately if a text message is received.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	NULL pointer or memory error
C_COM	D-Bus communication error
C_NOACT	No activity

Example

```
sint32 s32 Retval = C_NO_ERR;
T_DBUS_Util mt_DBus_Util;
s32_Retval = ygsmd_request_sms_fetch_urgent(&mt_DBus_Util);
```

```
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.9 Network

Header file: "Network_handler.h"

7.3.9.1 Introduction

The ynetwork daemon provides functions to manage the internet connectivity automatically. Via D-Bus the ynetworkd can be switched from connected to disconnected. It is also possible to get the actual connected interface.

The basic behavior of the ynetworkd is configurable via the ynetworkd configuration file. In the configuration file it's possible to select network interfaces which shall be used to get online. There is also the opportunity to assign priorities to the interfaces. Interfaces with higher priority can't be interrupted from interfaces with a lower priority. For the chosen interfaces, timings and general behaviors can be configured. It's also possible to switch between different online verification mechanisms.

For receiving network notifications automatically in your application, refer to the TAF library function `dbus_initialize_request_callbacks` (see "[dbus_initialize_request_callbacks](#)" on page 159).

7.3.9.2 ynetworkd_get_connected_interface

Function Description

```
sint32 ynetworkd_get_connected_interface (const T_DBUS_Util *const opt_DBusInstance, charn *
const opcn_ConnectedInterface, const uint8 ou8_BufferSize)
```

The function returns the name of the actual connected interface.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information
ou8_BufferSize	MIN_NETWORK_CON_INTERFACE .. uint8	includes the buffer size of the connected interface

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn          acn_myNameString[128];   // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn          acn_myVersion[128];      // Additional value (for HELLO) signal (opt.)
    charn          acn_myStatus[128];       // Additional value (for HELLO) signal (opt.)
    charn          acn_myAddInfo[512];      // Additional value (for HELLO) signal (opt.)
    sint32         s32_myTriginterval;      // Time interval when the ysysd
                                           // expects to be triggered
    charn          acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Parameter	Range	Description
opcn_ConnectedInterface	ou8_BufferSize	name of the actual connected interface, "ETH", "WLAN", "PPP", "NotConnected"

Return Value	Description
C_NO_ERR	Function executed without error
C_COM	D-Bus communication error
C_CONFIG	Invalid interface length or NULL pointer

Example

```
sint32 s32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
uint8 au8_Name[20] = {0};
s32_Retval = ynetworkd_get_connected_interface (&mt_DBUS_Util, au8_Name, sizeof(au8_Name));
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```


7.3.9.3 ynetworkd_start_connection

Function Description

```
sint32 ynetworkd_start_connection (const T_DBUS_Util *const opt_DBusInstance, const uint32
ou32_ConTime)
```

The function turns ynetworkd into connectivity mode for ou32_ConTime seconds. After ou32_ConTime seconds without a recall of the function, the ynetworkd automatically turns back into not connected mode.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
ou32_ConTime	uint32	Time in seconds to stay in connectivity mode

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];   // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	No error, function runs successful
C_COM	D-BUS communication error
C_NOACT	Error occurs because of invalid arguments or connected mode time is >= 120 sec (minimal connectivity time)
C_UNKNOWN_ERR	NULL pointer error

Example

```
sint32 s32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
s32_Retval = ynetworkd_start_connection (&mt_DBUS_Util, 120);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.9.4 ynetworkd_stop_connection

Function Description

```
sint32 ynetworkd_stop_connection (const T_DBUS_Util *const opt_DBusInstance)
```

The function turns the ynetworkd into not connected mode. If a connection is established, the connection will be automatically closed.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
    special                                                  // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                                  // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                  // by the ysysd in case no
                                                  // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	Function executed without error

Return Value	Description
C_COM	D-Bus communication error
C_NOACT	Error occurs because of invalid arguments
C_UNKNOWN_ERR	NULL pointer error

Example

```
sint32 s32_Retval = C_NO_ERR;
T_DBUS_Util mt_DBUS_Util;
s32_Retval = ynetworkd_stop_connection (&mt_DBUS_Util);
if (s32_Retval == C_NO_ERR)
{
    // Go on
}
```

7.3.10 Server

Header file: "Server_handler.h"

7.3.10.1 Introduction

The yserver daemon of the TAF library is a component that provides functionality for immediate NextJob.job file request.



NOTE:

For more information about the NextJob.job mechanism contact Sensor-Technik Wiedemann GmbH Support.

7.3.10.2 eGPRComServer specific commands

The eGPRComServer specific commands manage the data exchange between a TC3G and a GPRS communication server (GPRS ComServer). If not included in a data section, the parameters are stored in ASCII format in the NextJob.job file.

FWSW – Set wakeup cycle time

Syntax of command:

FWSW;<wakeup_cycle>

Syntax of response:

FWSW;<error_code>

Description:

Set a new wakeup cycle time. At this interval the TC3G requests a new NextJob.job from the ComServer.

Parameter:

- Input:

Variable	Variable Type	Description
wakeup_cycle	uint32	Wakeup cycle time in seconds

- Output:

- Output:

Variable	Variable Type	Description
error_code	uint16	0: no error occurred, else command failed

DLGN – Get data logger log file by name

Syntax of command:

```
DLGN;<name>;<format>;<clear>;<first_dataset>
```

Syntax of response:

```
DLGN;<error_code>
```

Description:

Returns the content of the specified data logger log file. Starts with first_dataset.
Log file data can be deleted after reading is finished.

Parameter:

- Input:

Variable	Variable Type	Description
name	string	Name of data logger (max 16 char)
format	string	If 'H': get info header 1 & 2 + comment + data If 'N': get log file name + data
clear	string	If 'C': clear data logger content
first_dataset	uint32	First data set number to be read

- Output:

Variable	Variable Type	Description
error_code	uint16	0: no error occurred, else command failed
<name>.dlf	file	dlf-file containing the logger data

DLCN – Clear data logger log file by name

Syntax of command:

```
DLCN;<name>
```

Syntax of response:

```
DLCN;<error_code>
```

Description:

Clears the specified data logger log file (Only deletes logger data, does not delete the log file).

Parameter:

<ul style="list-style-type: none"> Input: 		
Variable	Variable Type	Description
name	string	Name of data logger

<ul style="list-style-type: none"> Output: 		
Variable	Variable Type	Description
error_code	uint16	0: no error occurred, else command failed

KFGV - Get variables

Syntax of command:

```
KFGV;<name>;<var_index_1>;<var_index_2>;...;<var_index_n> *
```

```
KFGV;DAT[VL1<u16_block_len><as8_name[16]><u16_var_index_1>  
<u16_var_index_2>...<u16_var_index_n><0x0000>] **
```

Syntax of response:

```
KFGV;<error_code>
```

```
*: ASCII mode
```

```
** : Binary mode
```

Description:

Reads a list of variables. Two formats of the command are supported. In ASCII mode var_index_n is ASCII coded. In binary mode the variable index are transferred binary in a data section. This allows to reduce the traffic volume compared to the ASCII format. The raw data returned from the device is parsed and a DLF file containing the variable values is generated. The name of the DLF file is specified by parameter <name>. If reading a variable fails, the command execution is aborted and an error code is returned. The DLF file only contains valid data. All data types are supported. The sizes of the variables are specified by their types.

Parameter:

<ul style="list-style-type: none"> Input: 		
Variable	Variable Type	Description
block_len	uint16	Length of data section in bytes 16 ((for name) + (No. of index * 2))
name	string	File name for returned data (max 16 char)
var_index_n	uint16	Index of variables to be read

<ul style="list-style-type: none"> Output: 		
Variable	Variable Type	Description
error_code	uint16	0: no error occurs, else command failed
<name>.dlf	file	DLF file containing the read variables

KFSV – Set variables

Syntax of command:

```
KFSV;<var_index_1>;<value_1>;...;<var_index_n>;<value_n> *
KFSV;DAT[VL2<u16_block_len_1><u16_var_index_1><au8_value[m]>...
<u16_block_len_n><u16_var_index_n><au8_value_n[m]>
<0x0000>]**
```

Syntax of response:

```
KFSV;<error_code>
*: ASCII mode
**: Binary mode
```

Description:

Writes a list of variables. Two formats are supported.
In ASCII mode var_index_n and value_n are ASCII coded. Only int8, int16 and int32 types are supported.
The length is not included in the command of this format. It is determined by the variable index.
In binary mode var_index_n and value_n are transferred binary in a data section. All data types are supported.
The sizes of the variables are specified by their types
Additionally, the size of each variable (+2 bytes for the variable index) is included in <block_len_n >.
The command response is the same for both formats. If writing of a variable fails, the command breaks at this variable and returns the error code of the write operation.

Parameter:

<ul style="list-style-type: none"> Input: 		
Variable	Variable Type	Description
block_len_n	uint16	Size of variable + 2 bytes for index (for binary format only)
var_index_n	uint16	Index of variables to be written
value_n		Value of variable. Size (m) specified by var_index (via KSP file) and block_len_n-2

<ul style="list-style-type: none"> Output: 		
Variable	Variable Type	Description
error_code	uint16	0: no error occurred, else command failed

DLTS – Datalogger transmission size
Syntax of command:

DLTS;<transmission size in bytes>

Syntax of response:

DLTS;<error_code>

Description:

Set the maximum transmission size before a result.xxx.end file is created. The default value is 0. In that case, the logger will transmit all logged data before a *.end file is created.

Parameter:

<ul style="list-style-type: none"> Input: 		
Variable	Variable Type	Description
transmission size in bytes	uint32	transmission size in bytes

- Output:

Variable	Variable Type	Description
error_code	uint16	0: no error occurred, else command failed

7.3.10.3 Server independent commands

User specific commands can be send via yserverd on the D-Bus. To use this mechanism, the following syntax must be used:

Send to D-Bus

DBUS;<response>;<destination>;<data_length>;<payload><CR>

Parameter	Description
DBUS	Tag, don't edit
response	0: Don't wait for D-Bus response, always answer with "OK" in the "job" file 1: Wait for D-Bus answer with a maximum wait time of 5 seconds, After timeout, the "job" file will be automatically answered with an error.
destination	D-Bus name of the application which receives the message. If the destination address is empty, the message is automatically send as a broadcast message with no wait for response. In this case, it does not matter whether the response flag is set or not.
data_length	Length of the payload
payload	Data which shall be send via D-Bus

Example:

```
DBUS;0;yserverd;10;WhoIsThere
```

7.3.10.4 yserverd_request_nextjob

Function Description

```
sint32 yserverd_request_nextjob (const T_DBUS_Util * const opt_DBusInstance)
```

The function sends a D-Bus message for immediate request of a NextJob.job file. All running yserverd instances will receive the message. All instances of the yserverd check if there is a new Nextjob.job file available on the server.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;     // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_CO_ERR	Function executed without error
C_UNKNOWN_ERR	NULL pointer error or D-Bus communication error

7.3.11 GSM

Header file: "GSM_handler.h"

7.3.11.1 Introduction

The GSM functionality from the TAF library uses the ygsm daemon to provide modem, SIM card and provider information from the integrated GSM module.

7.3.11.2 ygsm_get_gsm_data

Function Description

```
sint32 ygsm_get_gsm_data (const T_DBUS_UTIL * const opt_DBusInstance, T_DBUS_GPS_Data *
const opt_DBusGsmData)
```

The function sends a method call on the D-Bus to the ygsm opt_DBusInstance to receive GSM data. This received GSM data is stored in the structure opt_DBusGsmData.

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-Bus information

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Result	Range	Description
opt_DBusGsmData	T_DBUS_GSM_Data	holds the GSM data information

Structure T_DBUS_GSM_Data

```
typedef struct
{
    charn acn_PhoneNo[32];           // Phone number of the SIM card
    charn acn_SIMSerial[32];         // Serial number of the SIM card
    charn acn_SIMState[32];          // SIM card status
    charn acn_SIMMCC[32];            // Mobile Country Code of the SIM card
    charn acn_SIMMNC[32];            // Mobile Network Code of the SIM card
    charn acn_SIMCarrier[32];        // Provider of the SIM card
    charn acn_SIMAPN[32];            // Access Point Name of the SIM card
    charn acn_SIMUser[32];           // SIM user name
    charn acn_SIMPasswd[32];         // SIM password
    charn acn_IMEI[32];              // International Mobile Equipment Identity (IMEI)
    charn acn_IMSI[32];              // International Mobile Subscriber Identity (IMEI)
    charn acn_RegStatus[32];         // Network Registration Status
    charn acn_SigQuality[32];        // Signal Quality
    charn acn_SigQualitydBm[32];     // Signal Quality in dBm
    charn acn_AccessTec[32];         // Access Technology (e.g.3G)
    charn acn_MCC[32];               // Mobile Country Code
    charn acn_MNC[32];               // Mobile Network Code
    charn acn_LAC[32];               // Location Area Code
    charn acn_CellID[32];            // Cell ID
} T_DBUS_GSM_Data;
```

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	Error sending the method call to the daemon
C_COM	Error receiving the method call answer from the daemon
C_RANGE	One of the arguments are NULL pointers

Example

```
static sint32 ms32_RequestGSMData(const T_DBUS_Util* const opt_DBusInstance)
{
    T_DBUS_GSM_Data t_DBus_GSM_Data;
    // Call dbus util function
    if(ygsmd_get_gsm_data(opt_DBusInstance, &t_DBus_GSM_Data) != 0)
    {
        printf("Error fetching GSM data!\n");
        return C_COM;
    }
    // To something with the received values
    return C_NO_ERR;
```

```
}

```

7.3.12 Signal

Header file: "Signal_handler.h"



NOTE:

The signal daemon can only be used, when a LED is available on the used module.

7.3.12.1 Introduction

The ysignal daemon signalizes the current state of the module (for example Power ON) according to the D-Bus signals of the ynetworkd and ygpsd.

The ysignal daemon can be used without the TAF components ynetworkd and ygpsd. If the daemon is configured that way, it will listen to the D-Bus for signals that characterizes the current state of the module.



NOTE:

In order to make the ysignal daemon independent of ynetworkd and ygpsd the according settings in the ysignal.config file need to be adjusted. For further information see ysignal daemon (see "[Signal daemon](#)" on page 116).

7.3.12.2 ysignald_internet_con_state

Function Description

```
sint32 ysignald_internet_con_state (const T_DBUS_Util * const opt_DBusInstance, const charn
* const opcn_SignalArgument);
```

This function sends the desired internet connection state via D-Bus.

To be able to use this function the UseExternalInternetState must be set to true.

UseExternalInternetState can be configured in the configuration file ysignal.config of the ysignal daemon (see "[Signal daemon](#)" on page 116).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_SignalArgument	"ETH" "WLAN" "PPP" "NotConnected"	signalizes connected to the internet via ethernet signalizes connected to the internet via wireless lan signalizes connected to the internet via ppp signalizes lost connection to the internet

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn           acn_myNameString[128];  // Name of the application, Don't use any
    special                                                // characters, white spaces or new lines!
    charn           acn_myVersion[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn           acn_myAddInfo[512];     // Additional value (for HELLO) signal (opt.)
    sint32          s32_myTriginterval;     // Time interval when the ysysd
                                                // expects to be triggered
    charn           acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                                // by the ysysd in case no
                                                // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	No error, function runs successful
C_COM	Communicated wrong signal state
C_RANGE	Some of the function arguments are NULL pointer

Example

```
.
.
.
/* Register with D-BUS */
if(dbus_get_on_the_bus(&mt_dbus_util) != 0)
{
    // Register application on D-BUS failed
    return(1);
}

/* send dbus signal */
s32_retval = ysignal_d_internet_con_state(&mt_dbus_util, "ETH");
if(s32_retval != C_NO_ERR)
{
    // Sending 'Internet_Connection_State' signal with state 'ETH' failed
    return(1);
}
.
.
.
```

7.3.12.3 ysignald_GPS_con_state

Function Description

```
sint32 ysignald_GPS_con_state (const T_DBUS_Util * const opt_DBusInstance, const charn *
const opcn_SignalArgument);
```

This function sends the desired GPS state via D-BUS.

To be able to use this function the UseExternalGPSState must be set to true.

UseExternalGPSState can be configured in the configuration file ysignald.config of the ysignal daemon (see "[Signal daemon](#)" on page 116).

Information Flow

Input Information

Parameter	Range	Description
opt_DBusInstance	T_DBUS_Util	holds all D-BUS information
opcn_SignalArgument	"Valid" "Invalid"	signalizes, received valid GPS data signalizes, no valid GPS data received

Structure T_DBUS_Util

```
typedef struct
{
    DBusConnection* pt_dbus_conn;           // D-BUS connection instance
    charn            acn_myNameString[128]; // Name of the application, Don't use any
special                                     // characters, white spaces or new lines!
    charn            acn_myVersion[128];    // Additional value (for HELLO) signal (opt.)
    charn            acn_myStatus[128];     // Additional value (for HELLO) signal (opt.)
    charn            acn_myAddInfo[512];    // Additional value (for HELLO) signal (opt.)
    sint32           s32_myTriginterval;    // Time interval when the ysysd
                                           // expects to be triggered
    charn            acn_myCMDOnWatchdog[2048]; // Bash cmd that must be executed
                                           // by the ysysd in case no
                                           // trigger signal occurred in time.
} T_DBUS_Util;
```

Output Information

Return Value	Description
C_NO_ERR	No error, function runs successful
C_COM	Communicated wrong signal state
C_RANGE	Some of the function arguments are NULL pointer

Example

```
.
.
.
/* Register with D-BUS */
if(dbus_get_on_the_bus(&mt_dbus_util) != 0)
{
    // Register application on D-BUS failed */
    return(1);
}

/* send dbus signal */
s32_retval = ysignald_GPS_con_state(&mt_dbus_util, "Valid");
if(s32_retval != C_NO_ERR)
{
    // Sending 'GPS_Connection_State' signal with state 'Valid' failed
    return(1);
}
.
.
.
```


7.3.13 Utils

Header file: "utils.h"

7.3.13.1 utils_create_directory

Function Description

```
sint32 utils_create_directory (const charn * const opcn_Path, const mode_t oun_Mode)
```

The function creates a directory, in case the directory does not already exist.

Information Flow

Input Information

Parameter	Range	Description
opcn_Path		pointer to the name of the directory we want to create
ocn_Mode	mode_t	<p>S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP, S_IROTH, S_IWOTH, S_IXOTH, S_ISUID, S_ISGID, S_ISVTX</p> <p>S_IRWXU is the bitwise OR of S_IRUSR, S_IWUSR and S_IXUSR.</p> <p>S_IRWXG is the bitwise OR of S_IRGRP, S_IWGRP and S_IXGRP.</p> <p>S_IRWXO is the bitwise OR of S_IROTH, S_IWOTH and S_IXOTH.</p>

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_CONFIG	NULL pointer
C_NOACT	can not create the directory

7.3.13.2 utils_remove_directory

Function Description

```
sint32 utils_remove_directory (const charn * const opcn_DirPath)
```

The function removes a directory, in case the directory exists.

Information Flow

Input Information

Parameter	Range	Description
opcn_DirPath		pointer to the name of the directory

Output Information

Return Value	Description
C_NO_ERR	Function executed without error. The directory was removed successfully! (if directory is incorrect nothing happen)

7.3.13.3 utils_file_exists

Function Description

```
sint32 utils_file_exists (const charn * const opcn_FileName)
```

The function verifies if the file opcn_FileName exists in the current folder.

Information Flow

Input Information

Parameter	Range	Description
opcn_FileName		pointer to file name we expect to find

Output Information

Return Value	Description
1	The file exists
0	The file does not exist

7.3.13.4 utils_reg_file_exists

Function Description

```
sint32 utils_reg_file_exists (const charn * const opcn_FileName)
```

The function verifies if the regular file opcn_FileName exists in the current folder.

Information Flow

Input Information

Parameter	Range	Description
opcn_FileName		pointer to regular file name we expect to find

Output Information

Return Value	Description
1	The regular file exists
0	The regular file does not exist

7.3.13.5 utils_directory_exists

Function Description

```
sint32 utils_directory_exists (const charn * const opcn_DirName)
```

The function verifies if the directory opcn_DirName exists.

Information Flow

Input Information

Parameter	Range	Description
opcn_DirName		pointer to the directory path

Output Information

Return Value	Description
1	The directory exists
0	The directory does not exist

7.3.13.6 utils_link_exists

Function Description

```
sint32 utils_link_exists (const charn * const opcn_LinkName)
```

The function verifies if the link opcn_LinkName exists in the current folder.

Information Flow

Input Information

Parameter	Range	Description
opcn_LinkName		pointer to name of the link to check

Output Information

Return Value	Description
1	The link exists
0	The link does not exist

7.3.13.7 utils_socket_exists

Function Description

```
sint32 utils_socket_exists (const charn * const opcn_SocketName)
```

The function verifies if the socket opcn_SocketName exists in the current folder.

Information Flow

Input Information

Parameter	Range	Description
opcn_SocketName		pointer to name of the socket to check

Output Information

Return Value	Description
1	The socket exists
0	The socket does not exist

7.3.13.8 utils_fifo_exists

Function Description

```
sint32 s32_FifoExists(charn const * const opcn_FifoName)
```

The function verifies if the FIFO opcn_FifoName exists in the current folder.

Information Flow

Input Information

Parameter	Range	Description
opcn_FifoName		pointer to FIFO name

Output Information

Return Value	Description
1	The FIFO exists
0	The FIFO does not exist

7.3.13.9 utils_strlcpy

Function Description

```
sint32 s32_strlcpy (charn * opcn_Destiny, const charn *opcn_Source, sint32 os32_Size)
```

The function copies opcn_Source to opcn_Destiny. The maximum length of the opcn_Source is os32_Size.

Information Flow

Input Information

Parameter	Range	Description
opcn_Source	ou32_Size	contains the pointer to source
os32_Size	uint32	contains the size of the source in byte

Output Information

Parameter	Range	Description
opcn_Destiny	...	contains the pointer to the expected destiny

Return Value	Description
sint32	number of copied bytes

7.3.13.10 utils_search_first_value

Function Description

```
sint32 utils_search_first_value (FILE * opt_File, const charn * const opcn_SearchString, charn * opcn_ValueTarget, const uint32 ou32_ValueTargetBufferSize)
```

The function searches for the string opcn_SearchString, from the beginning in the file opt_File. If opcn_SearchString is located in the file, the rest of the line is copied into opcn_ValueTarget. The maximal length of opcn_ValueTarget is defined by ou32_ValueTargetBufferSize in byte.

Information Flow

Input Information

Parameter	Range	Description
opt_File	FILE	pointing to file in that the search sting must be searched
opcn_SearchString		pointer to the search string
opcn_ValueTarget		pointer to the value string
opcn_ValueTargetBufferSize		pointer to the buffer size of the value

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, the search string is found and the value is written to the buffer
C_UNKNOWN_ERR	No search string was found, the buffer is too small or can not allocate the buffer

7.3.13.11 utils_search_next_value

Function Description

```
sint32 utils_search_next_value (FILE * opt_File, charn const * const opcn_SearchString, const charn * opcn_ValueTarget, const uint32 ou32_ValueTargetBufferSize)
```

The function searches for the string opcn_SearchString from the current file pointer position in file opt_File. If opcn_SearchString is located in the file, the rest of the line is copied into opcn_ValueTarget. The maximum length of opcn_ValueTarget is defined by ou32_ValueTargetBufferSize in byte.

Information Flow

Input Information

Parameter	Range	Description
opt_File	FILE	pointing to file in that the search sting must be searched

Parameter	Range	Description
opcn_SearchString		pointer to the search string
opcn_ValueTarget		pointer to the value string
opcn_ValueTargetBufferSize		pointer to the buffer size of the value

Output Information

Return Value	Description
C_NO_ERR	search string is found and value is written to buffer
C_UNKNOWN_ERR	no search string found, buffer too small or couldn't allocate buffer

7.3.13.12 utils_open_log_file

Function Description

```
sint32 utils_open_log_file (charn * opcn_FileName)
```

The function opens or creates the log file opcn_FileName.

Information Flow

Input Information

Parameter	Range	Description
opcn_FileName		pointer to file name

Output Information

Return Value	Description
C_NO_ERR	The file was opened or created
C_UNKNOWN_ERR	It was not possible to open or create the file

Example

```
sint32 s32_Retval;
s32_Retval = utils_open_log_file ("/var/run/log")
if (s32_Retval == C_NO_ERR)
{
    (void) utils_log_print ("Test");
    (void) utils_log_close ();
}
```

7.3.13.13 utils_log_print

Function Description

```
sint32 utils_log_print (charn * opcn_LogString)
```

The function prints the string opcn_LogString in the log file that was already opened. The function adds a time stamp to the string opcn_LogString.

If the string could not be added to the log file, it is printed to stdout.

Information Flow

Input Information

Parameter	Range	Description
opcn_LogString		includes the log string

Output Information

Return Value	Description
C_NO_ERR	Function executed without error.

Example

```
if (utils_open_log_file ("/var/run/log") == C_NO_ERR)
{
    (void) utils_log_print ("Test");
    (void) utils_log_close ();
}
```

7.3.13.14 utils_log_close

Function Description

```
sint32 utils_log_close (void)
```

The function closes the log file.

Information Flow

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_UNKNOWN_ERR	No open log file found

Example

```
if (utils_open_log_file ("/var/run/log") == C_NO_ERR)
{
    (void) utils_log_print ("Test");
    (void) utils_log_close ();
}
```


7.3.13.15 utils_remove_file

Function Description

```
sint32 utils_remove_file (const charn * const opcn_FileName)
```

The function removes the file opcn_FileName. Before the file will be removed, the content will be deleted. Recovering of the content is not possible.

Information Flow

Input Information

Parameter	Range	Description
opcn_FileName		includes file name

Output Information

Return Value	Description
C_NO_ERR	the file was deleted
C_CONFIG	Null pointer
C_UNKNOWN_ERR	the file can not be deleted

7.3.13.16 utils_init_semaphore

Function Description

```
sint32 utils_init_semaphore (key_t const osn_SHMKey, const uint32 ou32_SemCounter, sint32 * const ops32_SemID)
```

The function initializes the semaphore with the ID ops32_SemID. If the semaphore already exists, the semaphore ID of this existing semaphore will be returned.

Information Flow

Input Information

Parameter	Range	Description
osn_SHMKey		Specifies either the IPC_PRIVATE value or an IPC key constructed by the ftok subroutine
ou32_SemCounter		Specifies the number of semaphores in the set

Output Information

Parameter	Range	Description
ops32_SemID		includes the semaphore ID

Return Value	Description
C_NO_ERR	Function executed without error, a semaphore ID was successful generated
C_CONFIG	Semaphore counter is smaller than or equal to zero
C_NOACT	An error occurred when generating a new semaphore ID, or getting the semaphore ID that was already initialized

7.3.13.17 utils_semaphore_remove

Function Description

```
sint32 utils_semaphore_remove (const sint32 os32_SemID)
```

The function removes the semaphore specified by the Id os32_SemID.

Information Flow

Input Information

Parameter	Range	Description
os32_SemID		semaphore ID

Output Information

Return Value	Description
C_NO_ERR	Function executed without error, a semaphore ID was successful removed
C_CONFIG	Cannot remove semaphore

7.3.13.18 utils_semaphore_getid

Function Description

```
sint32 utils_semaphore_getid(key_t const osn_SHMKey, sint32 * const ops32_SemID)
```

The function gets the semaphore Id that is associated with osn_SHMKey.

Information Flow

Input Information

Parameter	Range	Description
osn_SHMKey		Specifies either the IPC_PRIVATE value or an IPC key constructed by the ftok subroutine

Output Information

Parameter	Range	Description
ops32_SemID		includes the semaphore ID

Return Value	Description
C_NO_ERR	Function executed without error, a semaphore ID was successful generated
C_CONFIG	invalid semaphore key

7.3.13.19 utils_semaphore_lock

Function Description

```
sint32 utils_semaphore_lock (const sint32 os32_SemID, const uint8 ou8_Mode)
```

The function locks the created semaphore with the ID os32_SemID. Use this function before entering a critical section.

Information Flow

Input Information

Parameter	Range	Description
os32_SemID		includes the semaphore ID
ou8_Mode	0 ... 1	0: Wait until it is possible to access the critical section 1: No waiting. If it is not possible to enter the section, return with an error

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_CONFIG	The semaphore ID is smaller than zero
C_NOACT	The function can not lock the semaphore with the given ID

7.3.13.20 utils_semaphore_unlock

Function Description

```
sint32 utils_semaphore_unlock (const sint32 os32_SemID, const uint8 ou8_Mode)
```

The function unlocks the semaphore `os32_SemID` that was locked before. Use function after leaving a critical section.

Information Flow

Input Information

Parameter	Range	Description
os32_SemID		includes the semaphore ID
ou8_Mode	0 ... 1	0: Wait until it is possible to access the critical section 1: No waiting. If it is not possible to enter the section, return with an error

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_CONFIG	The semaphore ID is smaller than zero
C_NOACT	The function can not unlock the semaphore with the given ID <code>os32_SemID</code>

7.3.13.21 utils_get_hostname

Function Description

```
sint32 utils_get_hostname (charn * const opcn_Hostname)
```

The function gets the host name opcn_Hostname of the device.

Information Flow

Output Information

Parameter	Range	Description
opcn_Hostname		includes the host name of the device

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
else	"TC3G-GET_HOSTNAME_ERR" will be set to the output

7.3.13.22 utils_write_string_value

Function Description

```
sint32 utils_write_string_value (FILE * opt_File, const charn * const opcn_String)
```

The function write the string opcn_String to the end of the file opt_File.

Information Flow

Input Information

Parameter	Range	Description
opt_File		contains the file name to that the string must be written to
opcn_String		contains the string that must be added to the file

Output Information

Return Value	Description
C_NO_ERR	Function executed without error
C_CONFIG	The string could not be written to the end of the file

7.3.13.23 utils_get_time_sig

Function Description

```
sint32 utils_get_time_sig (charn * const opcn_Time)
```

The function returns the current time string opcn_Time. The data type of the time string is signed integer.

Information Flow

Input Information

Parameter	Range	Description
opcn_Time		contains the time string, the time string contains the current date and time

Output Information

Return Value	Description
>0	time format as integer (seconds since 1970)
C_CONFIG	The time string is corrupted

7.3.13.24 utils_get_time

Function Description

```
charn * utils_get_time (charn * opcn_Time)
```

The function writes the current time and date as a string into opcn_Time.

Information Flow

Output Information

Parameter	Range	Description
opcn_Time		contains local time

Return Value	Description
charn *	pointer to opcn_Time

7.3.13.25 utils_get_GM_time

Function Description

```
charn * utils_get_GM_time (charn * const opcn_Time)
```

The function writes the current UTC time and date as a string into opcn_Time.

Information Flow

Output Information

Parameter	Range	Description
opcn_Time		contains the GM Time

Return Value	Description
charn *	pointer to opcn_Time

7.3.13.26 utils_get_time_ms

Funcon Description

```
uint32 utils_get_time_ms (void)
```

The function returns the current time that is elapsed since the TC3G was initialized.

The elapsed time is scaled in milliseconds.

Information Flow

Output Information

Return Value	Description
uint32	Up time in milliseconds

7.3.13.27 utils_open_dir

Function Description

```
void* utils_open_dir(const char* const opcn_Path)
```

The function opens a directory for reading.

Information Flow

Input Information

Parameter	Range	Description
opcn_Path		Directory path

Output Information

Return Value	Description
!= NULL	Directory handle
NULL	Failed to open a directory

7.3.13.28 utils_close_dir

Function Description

```
void utils_close_dir(void* const opv_DirHandle)
```

The function closes a directory handle.

Information Flow

Input Information

Parameter	Range	Description
opv_DirHandle		Directory handle

7.3.13.29 utils_read_dir

Function Description

```
sint32 utils_read_dir(void* const opv_DirHandle, const char* const opcn_Match, char*
const opcn_FileName)
```

The function returns the next file `opcn_FileName` that contains the string `opcn_Match` from the directory `opv_DirHandle`.

Information Flow

Input Information

Parameter	Range	Description
<code>opv_DirHandle</code>		Directory handle
<code>opcn_Match</code>		Match pattern (NULL or "": all files)

Output Information

Return Value	Description
<code>C_NO_ERR</code>	Function executed without error, a file was found
<code>C_NOACT</code>	Failed to find file, no files found
<code>C_RANGE</code>	Directory handle or file name parameter is a NULL pointer

8 Development Tools

8.1 Create Own Root File System

STW provides a "Board Support Package" for the TC3G. Within this BSP the customer has the possibility to create its own root file system.

The BSP uses Buildroot (<http://buildroot.uclibc.org/>) BR to create the root file system.

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation.

For how to work with the customer BR version of STW see chapter:

Download, extract and test setup (see "[Download, extract and test setup](#)" on page 288)

This customized root file skeleton can be increased, for example with customer scripts or customer applications.

The root file skeleton can be easily modified and increased with customer scripts or customer applications.

If you want to extend the skeleton by an own application, just copy the files and binaries in the desired folder (e.g. /usr/local/bin).

To update the TC3G, use the result file, for further information see chapter Update the Device (see "[Update the Device](#)" on page 321).

For how to adapt the root file skeleton with own scripts and applications see chapter:

Adapt the root file skeleton (see "[Adapt the root file skeleton](#)" on page 289)

For how to configure the BR packages see chapter:

Enable or Disable Buildroot Packages (see "[Enable or Disable Buildroot Packages](#)" on page 289)

For how to extend the customer BR with other packages see chapter:

Extend BR by adding packages (see "[Extend BR by adding packages](#)" on page 293)

For an explanation of the STW build scripts see chapter:

STW Build Scripts (see "[STW Build Scripts](#)" on page 293)

How to build a root file system

1. Go to the base folder of buildroot target project
2. Build the rootfs with the default configuration:
./bat/buildscripts/do_make_all

Or

Change the buildroot package configuration:

1. Go to the base folder of buildroot target project
2. Execute ./bat/buildscripts/do_defconfig to load the default configuration
3. Execute ./bat/buildscripts/do_menuconfig to open the Buildroot target configuration
4. Change the configuration as needed, see chapter Enable or Disable Buildroot Packages (see "[Enable or Disable Buildroot Packages](#)" on page 289)
5. Update the STW rootfs version (menuconfig->Set STW Version)
6. Save and exit menuconfig

7. Build the rootfs with the changed configuration:
`./bat/buildscripts/do_make`
8. Test the new rootfs by updating the TC3G (see chapter Update the Device (see "[Update the Device](#)" on page 321))
9. Set the new configuration to be the default configuration:
`./bat/buildscripts/do_savedefconfig`

Root file system version management

The STW rootfs version is configured by option BR2_STW_VERSION in the defconfig file. It can be changed either by editing the defconfig file or via the menuconfig->Set STW Version. The version of the final rootfs can be read in file `/etc/br_version`.

8.1.1 Download, extract and test setup

For each TC3G variant exists a specific BSP with its own customer Buildroot (BR).

1. Download the customer BR from the STW FTP server
ftp://esx-tc3.de/ (see ftp://esx-tc3.de/ - <ftp://esx-tc3.de/>)
2. Prepare your operation system
The following packages needs to be installed under Ubuntu 14.04 LTS:

```
sudo apt-get install g++
sudo apt-get install git
sudo apt-get install bison
sudo apt-get install flex
sudo apt-get install gettext
sudo apt-get install texinfo
sudo apt-get install subversion
sudo apt-get install mtd-utils
sudo apt-get install libssl-dev
sudo apt-get install libncurses5-dev
for the 64bit version additionally install:
sudo apt-get install lib32stdc++6
sudo apt-get install lib32z1
```

3. Extract the customer BR
The following commands unzips the customer BR into a folder:

```
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg$ ls
br_tc3g.tar.gz
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg$ mkdir unzip
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg$ ls
br_tc3g.tar.gz unzip
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg$ tar xvfz br_tc3g.tar.gz -C unzip/
./
./src/
./src/tc3g_defconfig
./src/udimage.cfg
./src/buildscripts_target_config
```

4. Test your setup
In order to work with the customer BR the setup should be tested. The easiest way to do that, is to build the BR with default settings: (do_make_all)

```
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip$ ./bat/buildscripts/do_make_all
make: Entering directory `/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip/libs/br_pj
```

Acknowledge that you have updated the version number with 'y'

```
#
make: Leaving directory `/projects/y/linux/mpc5200/rootfs/targets/t
Current STW RootFs version: "STW-V1.xrx"
Have you updated the version? (y/n)?y
Make rfs ...
```

5. The following screen output signalizes that your setup is correct and that you successfully built the root file system with default settings:

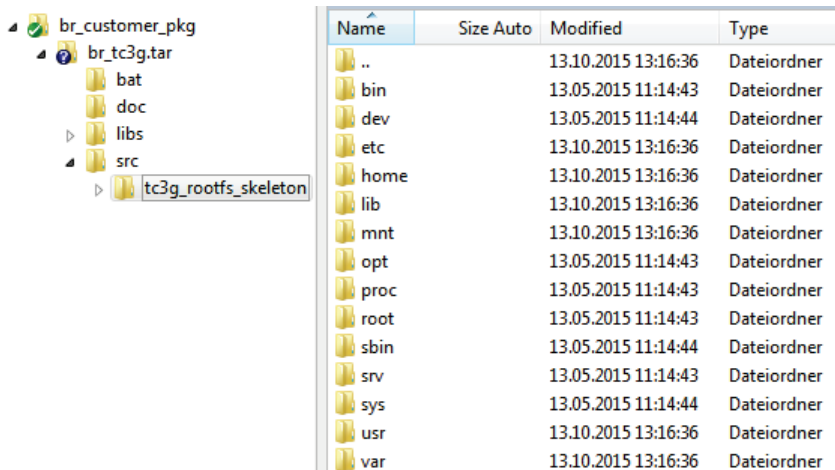
```
Done!
----- Build Finished (Build Time: 00h:30m:48s) -----
```

The 'rootfs.ubi' can be found in the result folder. In order to update the TC3G see chapter Linux Updater to update the Root File System.

8.1.2 Adapt the root file skeleton

The root file skeleton could easily be modified and extended by e.g. several scripts or applications of the customer.

The skeleton could be found in the 'src' folder. The content of the "tc3g_rootfs_skeleton" folder is the file system that will later be available on the TC3G.



Name	Size	Auto	Modified	Type
..			13.10.2015 13:16:36	Dateiordner
bin			13.05.2015 11:14:43	Dateiordner
dev			13.05.2015 11:14:44	Dateiordner
etc			13.10.2015 13:16:36	Dateiordner
home			13.10.2015 13:16:36	Dateiordner
lib			13.10.2015 13:16:36	Dateiordner
mnt			13.10.2015 13:16:36	Dateiordner
opt			13.05.2015 11:14:43	Dateiordner
proc			13.05.2015 11:14:43	Dateiordner
root			13.05.2015 11:14:43	Dateiordner
sbin			13.05.2015 11:14:44	Dateiordner
srv			13.05.2015 11:14:43	Dateiordner
sys			13.05.2015 11:14:44	Dateiordner
usr			13.10.2015 13:16:36	Dateiordner
var			13.10.2015 13:16:36	Dateiordner

If you want to extend the skeleton by an own application, just copy the files and binaries in the desired folder (for example /usr/local/bin).

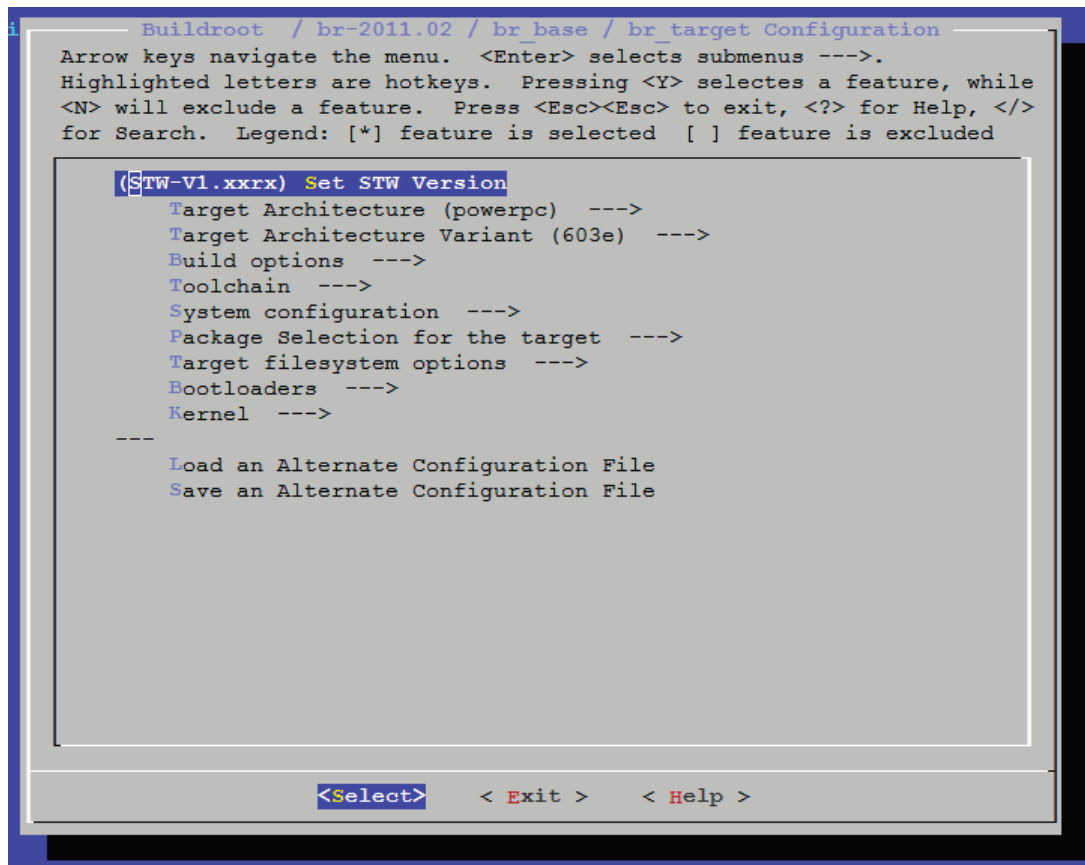
In order to build the root file system with the modifications, the 'do_make_all' script needs to be executed.

8.1.3 Enable or Disable Buildroot Packages

In order to change the configuration of the root file system e.g. enable or disable certain packages, it should be built with default settings first (do_make_all).

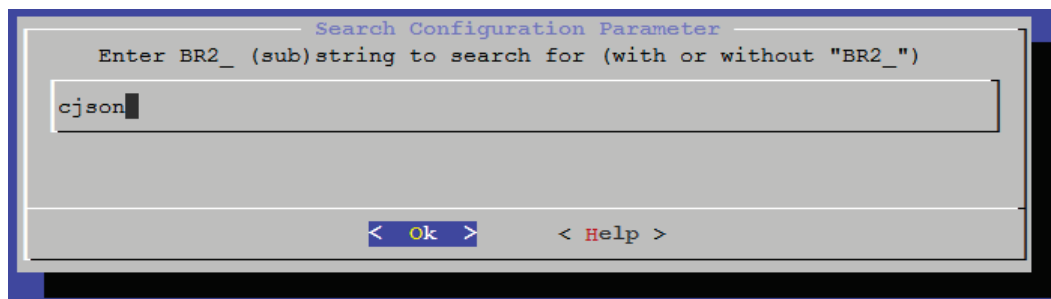
Use the STW build script 'do_menuconfig' under 'bat/buildscripts' to enable or disable a package. For further information see chapter STW Build Scripts (see "[STW Build Scripts](#)" on page 293).

Enter the Buildroot configuration menu via `./bat/buildscripts/do_menuconfig`.

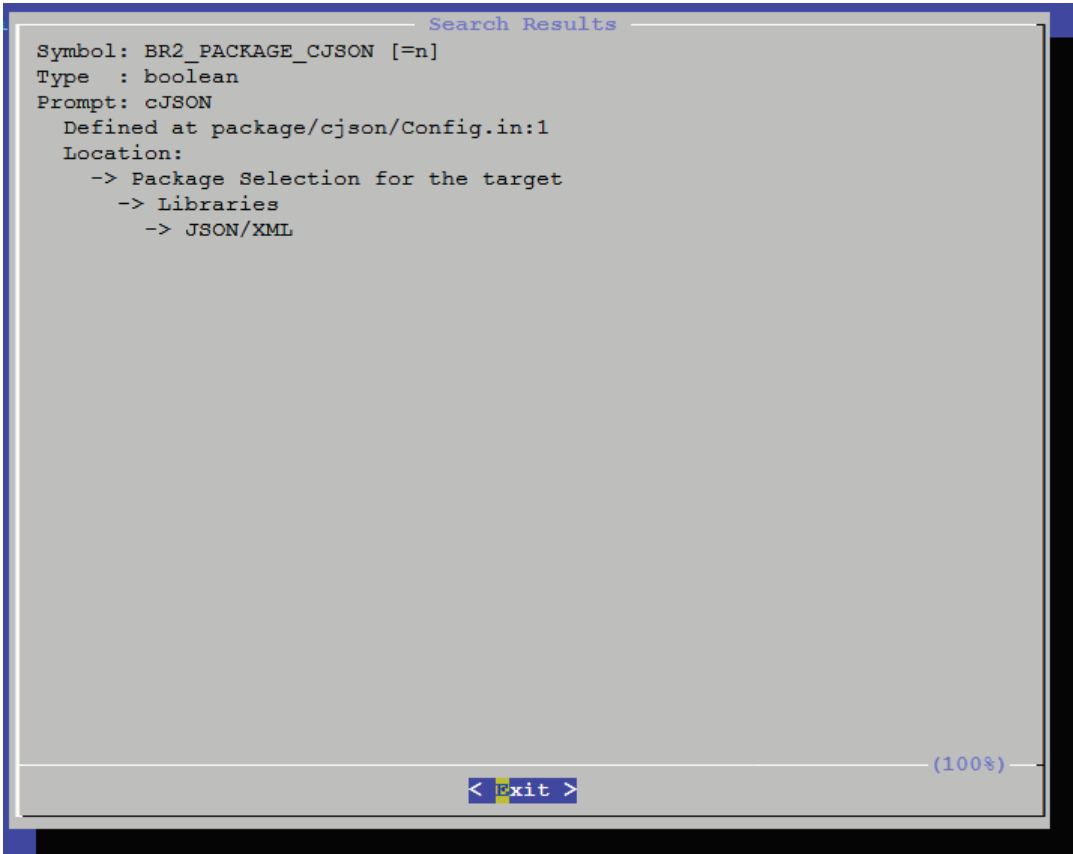


Inside the configuration menu it is possible to navigate through the sub menus and activate or deactivate the desired packages.

Use the search function, if you know the name of the package. Open the search mask with '/' (like vi), insert the package name, and press ENTER.



The results will be displayed.



```

Search Results
Symbol: BR2_PACKAGE_CJSON [=n]
Type  : boolean
Prompt: cJSON
Defined at package/cjson/Config.in:1
Location:
  -> Package Selection for the target
    -> Libraries
      -> JSON/XML
(100%)
< Exit >

```

'Location' shows the path to the searched package. In this case:

Top level -> Package Selection for the target -> Libraries -> JSON/XML

After navigation to this location, the searched package can be activated or deactivated.

```

JSON/XML
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> selectes a feature, while
<N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] feature is selected [ ] feature is excluded

[ ] cJSON
[ ] expat
[ ] ezxml
[*] json-c
-*- libxml2
-*- libxslt
[ ] xerces-c++

<Select> < Exit > < Help >

```

After the configuration is done, go to the top level and save the changes when leaving the menuconfig.

```

Do you wish to save your new configuration? <ESC><ESC>
to continue.

< Yes > < No >

```

When the build script 'do_make' is executed next, it will create a root file system with the new configurations.



NOTE:

'do_make_all' will overwrite the buildroot and busybox configuration with the target defconfig. To store the menuconfig settings permanently, use 'do_savedefconfig'. After that the 'do_make_all' script will use the updated configuration.

8.1.4 Extend BR by adding packages


NOTE:

Adding new packages to the Buildroot is only recommended for experienced Buildroot users.

The packages need to be added to the './libs/br_pjt/package' directory.

When a desired package is not available in the STW customer package selection, there is a fair chance, that this package exists in a newer version of Buildroot. In order to find out, the latest BR source tree should be searched for this package.

In case the package could be found in the newer version of Buildroot, it only needs to be adapted.

In case the package could not be found, it needs to be created.


NOTE:

In both cases the buildroot documentation is highly recommended '\libs\br_pjt\docs'.

8.1.5 STW Build Scripts

The STW build scripts for Buildroot could be found under 'bat/buildscripts':

```
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip$ ls
bat doc libs src
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip$ cd bat/buildsc
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip/bat/buildscripts$ ls
do_busybox_menuconfig  do_distclean  do_make_customer_buildroot  do_savedefconfig
do_busybox_savedefconfig  do_make      do_make_fitimage           do_zip_buildroot
do_defconfig           do_make_all  do_menuconfig              target_config
~/projects/y/linux/mpc5200/rootfs/targets/tc3g/orig_trunk/result/br_customer_pkg/unzip/bat/buildscripts$
```

Folder 'buildscripts' contains STW scripts to configure and build the rootfs. All steps could also be done directly in the buildroot project folder (libs/br_pjt) by using the Linux make command.

do_defconfig

Call buildroot "make <defconfig file>" to apply the buildroot configuration. The script overwrites the current .config file of the buildroot.

do_menuconfig

The script opens the menuconfig setup menu to change the buildroot configuration.

do_savedefconfig

The script saves the current buildroot configuration to the target defconfig file.

do_make

The script builds the rootfs with the current configuration. The results of the new built rootfs are stored in the folder result.

do_make_all

The script builds the default configuration of the rootfs.

The do_make_all script calls other build scripts in the following order:

```
do_distclean
do_defconfig
do_make
```

do_make_customer_buildroot
do_make_fitimage

**NOTE:**

The current buildroot and busybox configuration will be overwritten by the target defconfig!
To store the menuconfig settings permanently, use 'do_savedefconfig'. After that the 'do_make_all' script will use the updated configuration

do_make_customer_buildroot

The script creates the buildroot packages that can be shipped to the customer. All packages that are marked as "closed_source" packages (see menuconfig) are added to the customer BR version, without the source code.

do_make_fitimage

The script creates a FIT image from the rootfs.ubi result file. This image is located in the result folder and can be used to update the TC3G via the STW U-Boot update mechanism.

do_distclean

The script cleans the buildroot folder after a rootfs is build. The script deletes all files generated during the build process.

do_busybox_menuconfig

The script opens the busybox menuconfig setup menu to change its configuration. This only works after the rootfs has been built. The busybox package must have been unzipped into the output folder.

do_busybox_savedefconfig

The script saves the current busybox configuration to the target busybox defconfig file.

target_config

The script sets up the configuration files (target defconfig) and the target directory structure.

**WARNING:**

Change this file only when the scripts are ported to a new target.

8.2 Create Own Application

Precondition:

To be able to create an application, the toolchain must be installed on your platform that are you using for application development. Refer therefor to Toolchain (see "[Toolchain](#)" on page 302).

Compile a program:

There are basically two ways to compile a program:

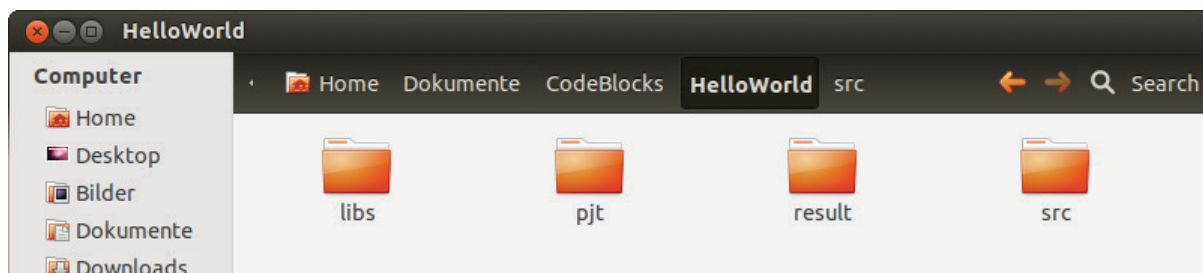
- Using the IDE: This is a comfortable way that uses the CodeBlocks IDE.
- Using the shell: This is the classic way to compile a program. It uses the shell and a make file or directly the cross compiler gcc.

In this chapter both paths are described.

8.2.1 Create New Project

Create a practical directory structure for the new project. Folder for source files, for the project file, for libraries and for the result are needed.

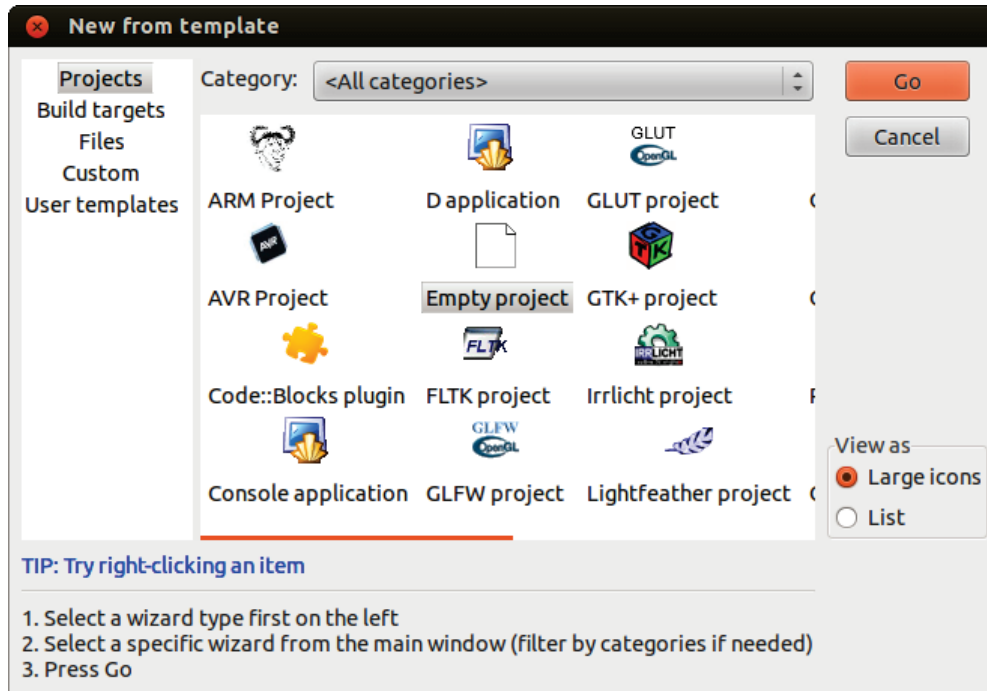
Example for the directory structure:



Start the CodeBlocks IDE.

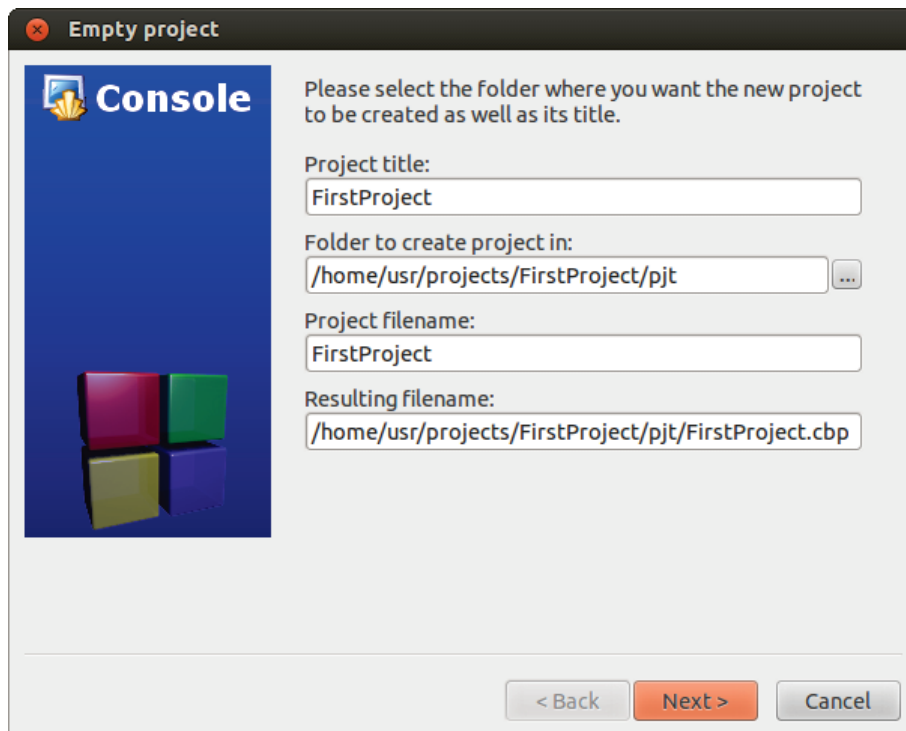
Menu: File | new | project

Pick the empty project.



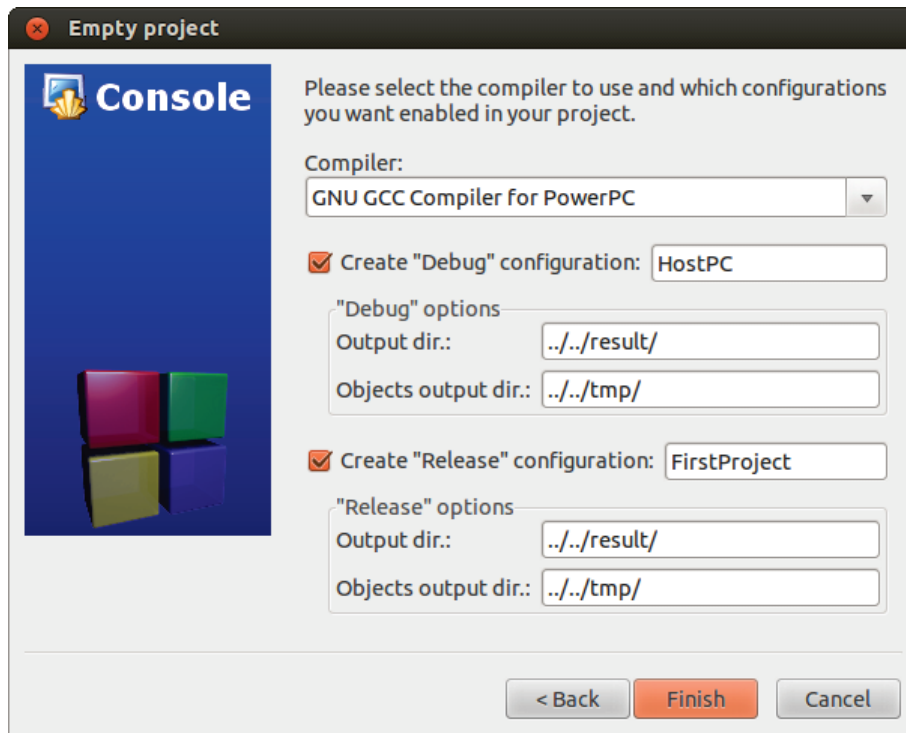
Click Go. The empty project wizard will pop up. Click Next.

Specify the name of the project, the directory for the project file and the name of the project file.



Click Next.

Select the cross compiler and the output folder.



Click "Finish"

The new project will be created with the following project tree:

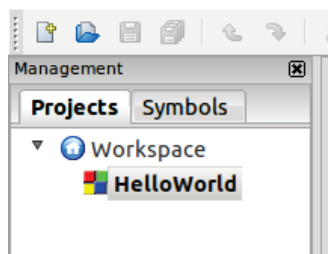
Project directory "test":

- test | pjt: This folder contains the project files
- test | result | TC3G: This folder contains the compiled output file

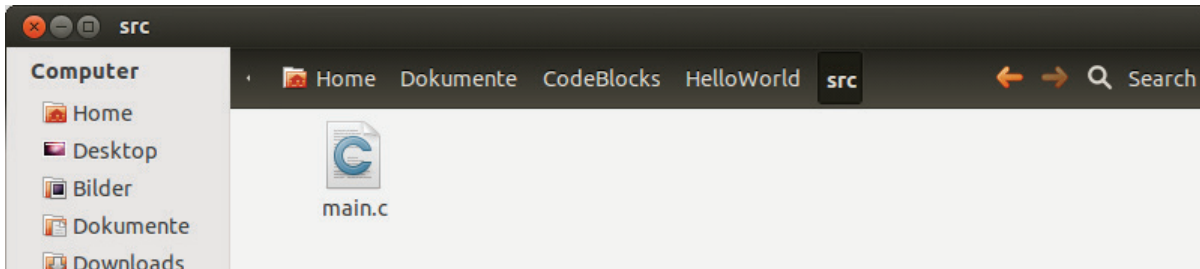
The new project shall be displayed in the "Project Management area".

8.2.2 Code Blocks

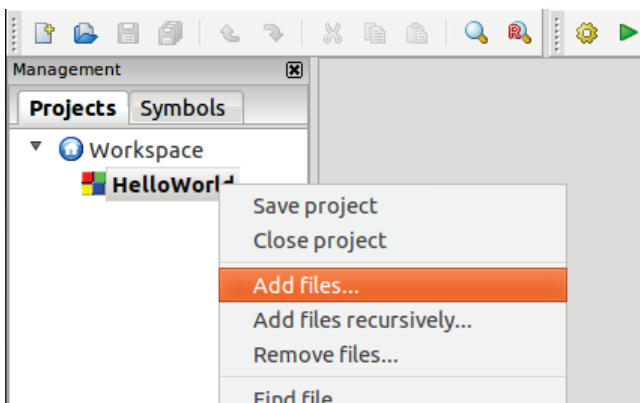
After a New Project (see ["Create New Project"](#) on page 295) is created the "Workspace" is available:



Create a source file. Switch to the src directory of the project and create a new file "main.c":



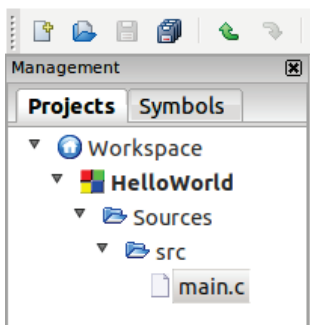
Add the file to the project. Right mouse click on the project symbol in the project management area in CodeBlocks and click on "Add files ...":



Select the "main.c" file in the source directory and click on "open".

Now it is possible to fold up the directory structure of the Hello World project in the management area.

Double click on the "main.c" file. The edit window pops up.



Now add the following source snip:

```
#include <stdio.h>

void main (void)
{
    printf ("Hello World\r\n");

    return 0;
}
```

Your file appears as follows in the IDE:

```

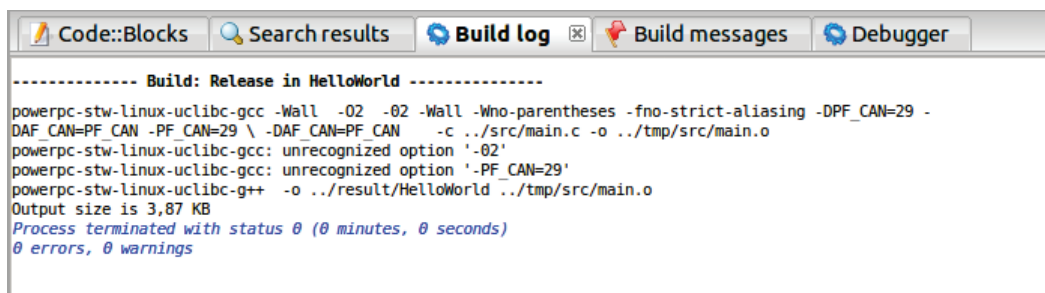
1  #include <stdio.h>
2
3  int main (void)
4  {
5      printf("Hello World\r\n");
6
7      return 0;
8  }
9

```

Click Rebuild:

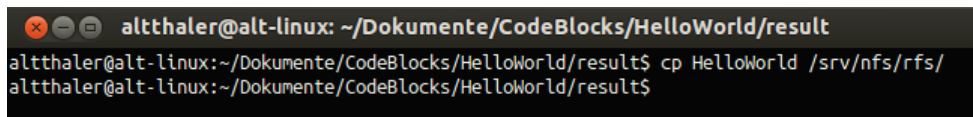


Code::Blocks compiles and links all necessary components and creates an executable file in the output directory.



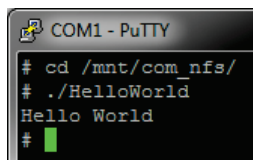
Copy the file to the device (for example over NFS (see "[NFS](#)" on page 336)):

Copy file via NFS



Note: Don't forget to mount the directories on Host PC and TC3G!

After executing the application, you should get the following output:



8.2.3 Command Line

If you only have one or two source files, the classic way using the gcc compiler directly is faster than using the Code::Blocks IDE.

Create a source file that shall be compiled:

```
altthaler@alt-linux: ~/Dokumente/HelloWorld
altthaler@alt-linux:~$ cd Dokumente/
altthaler@alt-linux:~/Dokumente$ mkdir HelloWorld
altthaler@alt-linux:~/Dokumente$ cd HelloWorld/
altthaler@alt-linux:~/Dokumente/HelloWorld$ touch hello.c
altthaler@alt-linux:~/Dokumente/HelloWorld$ gedit hello.c
```

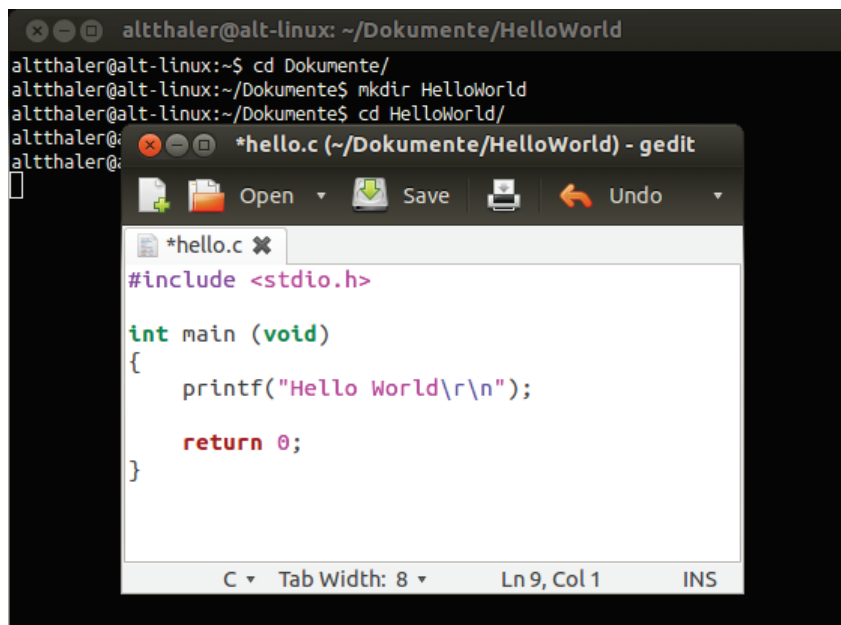
Add following source snip:

```
#include <stdio.h>

int main (void)
{
    printf ("Hello World\r\n");

    return 0;
}
```

```
altthaler@alt-linux: ~/Dokumente/HelloWorld
altthaler@alt-linux:~$ cd Dokumente/
altthaler@alt-linux:~/Dokumente$ mkdir HelloWorld
altthaler@alt-linux:~/Dokumente$ cd HelloWorld/
altthaler@alt-linux:~/Dokumente/HelloWorld$ gedit
altthaler@alt-linux:~/Dokumente/HelloWorld$
```



Now save the file and create an executable file. Therefor use the STW cross compiler.

**NOTE:**

Compiling in command line is possible after installing the DevKit (see "[Install the Toolchain](#)" on page 302)

The first parameter is the source file, the second one is the output file.

```
altthaler@alt-linux: ~/Dokumente/HelloWorld
altthaler@alt-linux:~$ cd Dokumente/
altthaler@alt-linux:~/Dokumente$ mkdir HelloWorld
altthaler@alt-linux:~/Dokumente$ cd HelloWorld/
altthaler@alt-linux:~/Dokumente/HelloWorld$ touch hello.c
altthaler@alt-linux:~/Dokumente/HelloWorld$ gedit hello.c
altthaler@alt-linux:~/Dokumente/HelloWorld$ powerpc-stw-linux-uclibc-gcc hello.c -o hello
altthaler@alt-linux:~/Dokumente/HelloWorld$ ls
hello hello.c
altthaler@alt-linux:~/Dokumente/HelloWorld$ cp hello /srv/nfs/rfs/
altthaler@alt-linux:~/Dokumente/HelloWorld$
```

Copy the output file via USB stick or NFS (see "[NFS](#)" on page 336) to device. After executing the application, you should get the following output:

```
COM1 - PuTTY
# cd /mnt/com_nfs/
# ./hello
Hello World
#
```

8.3 Toolchain

8.3.1 Linux



NOTE:

STW supports and recommends Ubuntu 14.04 LTS for developing. This operating system has been tested and is 100 % compatible with the TC3G development kit. All other linux based operating systems are neither recommended nor supported for TC3G development.

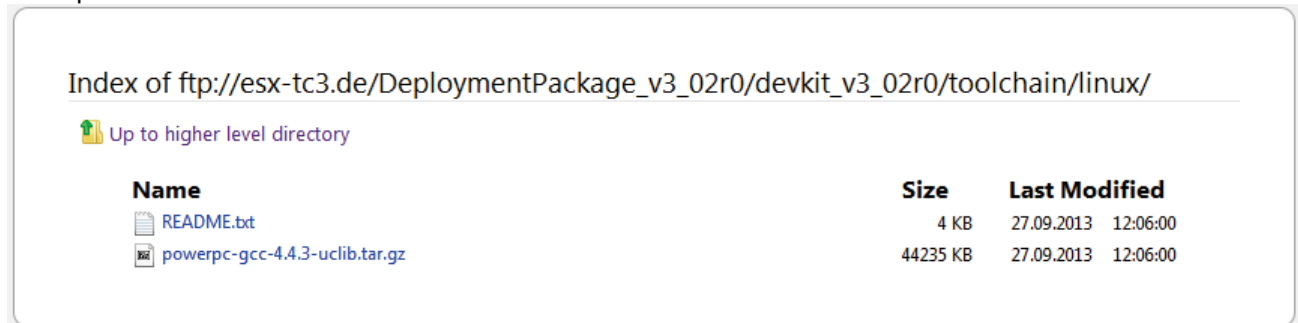
8.3.1.1 Install the Toolchain

This chapter gives you a step by step tutorial to install the toolchain on a 32-/64-Bit Ubuntu 14.04 LTS System.

How to install the toolchain

1. Download the toolchain "powerpc-gcc-4.4.3-uclib.tar.gz" from STW FTP - 32-/64-Bit
URL: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>)
TC3G/DeploymentPackage_vX_XXrX/devkit_vX_XXrX/toolchain/linux/ (see example image below)

Example:

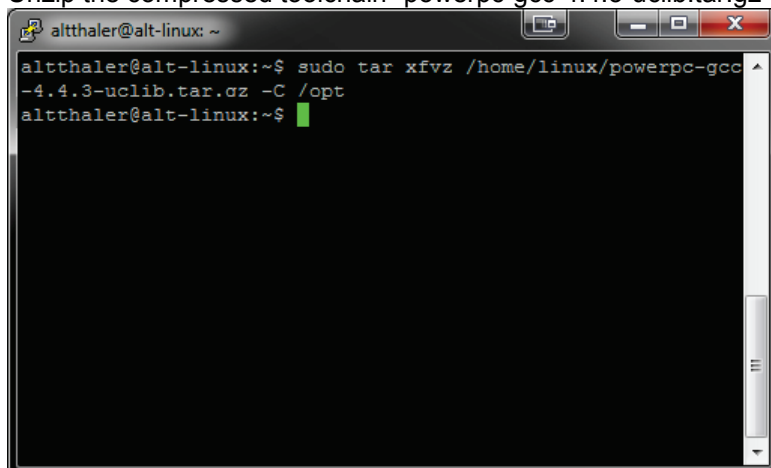


Index of ftp://esx-tc3.de/DeploymentPackage_v3_02r0/devkit_v3_02r0/toolchain/linux/

Up to higher level directory

Name	Size	Last Modified
README.txt	4 KB	27.09.2013 12:06:00
powerpc-gcc-4.4.3-uclib.tar.gz	44235 KB	27.09.2013 12:06:00

2. Unzip the compressed toolchain "powerpc-gcc-4.4.3-uclib.tar.gz" - 32-/64-Bit



3. Configure shell to use the toolchain - 32-/64-Bit

To use the toolchain under the shell, set the environmental variables. Open the ".bashrc" file
`~$ sudo gedit ~/.bashrc`

4. Add the following lines at the end of file:

```
# Export the Environment variable for the ppc
export CROSS_COMPILE=powerpc-stw-linux-uclibc-
export PATH="/opt/powerpc-gcc-4.4.3-uclib/bin/":"/opt/powerpc-gcc-4.4.3-
uclib/bin/:${PATH}
export ARCH=powerpc
```

Save the changes and close the terminal. After reopening the cross compiler is available.

5. Optional step, when a 64 bit operating system is used:

Activate an i386 architecture on the 64 bit system with
`sudo apt-get install libc6-i386`

6. Find the compiler in the shell

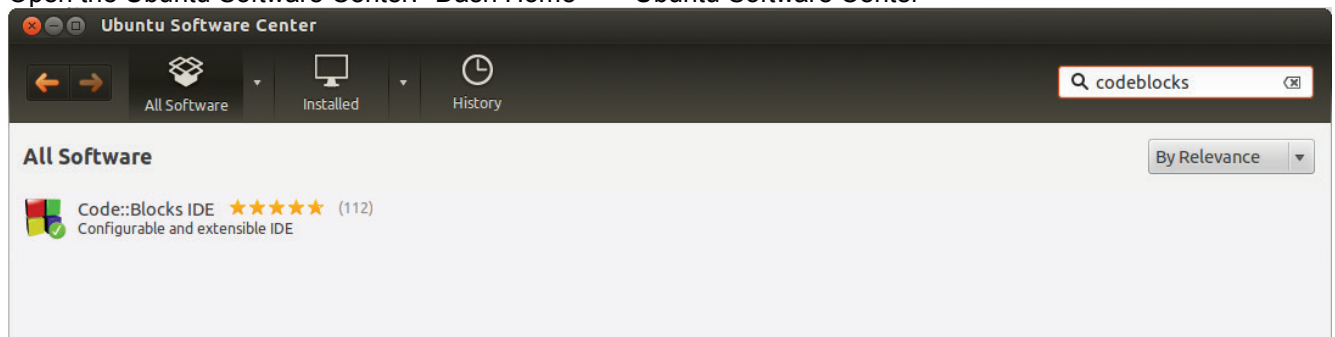
Find the installed compiler, type 'powerpc-stw-linux-uclibc' and press TAB

The shell will return the amount of compiler that are currently installed.

8.3.1.2 Install the Code::Blocks IDE

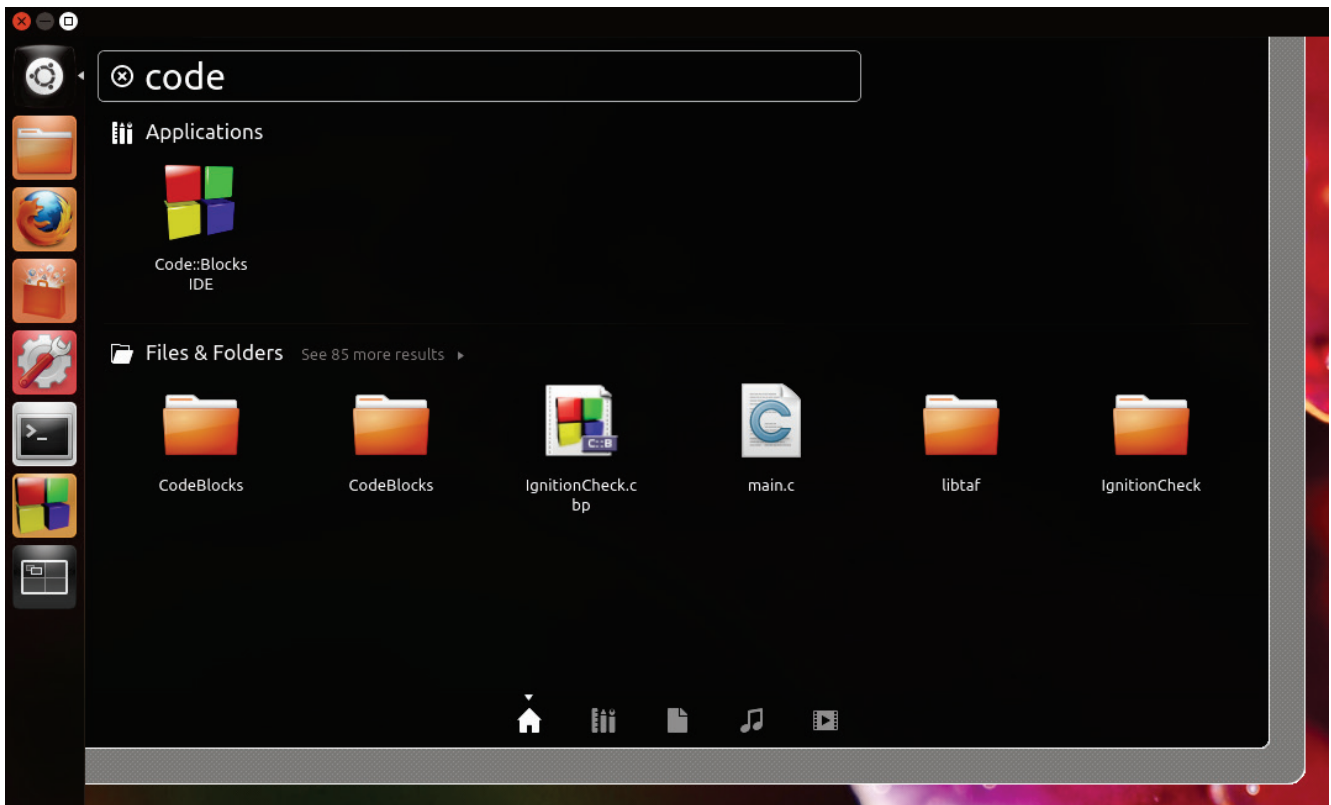
Code::Blocks can be used to compile ready to go executables for the TC3G. Especially for larger projects, Code::Blocks is recommended, since it provides a good overview over your software modules to be used. If you want to install Code::Blocks, go to the synaptic package manager of Linux Ubuntu.

1. Open the Ubuntu Software Center: "Dash Home" -> "Ubuntu Software Center"



2. Click Install and follow the instructions on the screen.

3. Open the Code::Blocks IDE from the application drop down menu at the top of your Ubuntu desktop. Type the first letters of the program name.

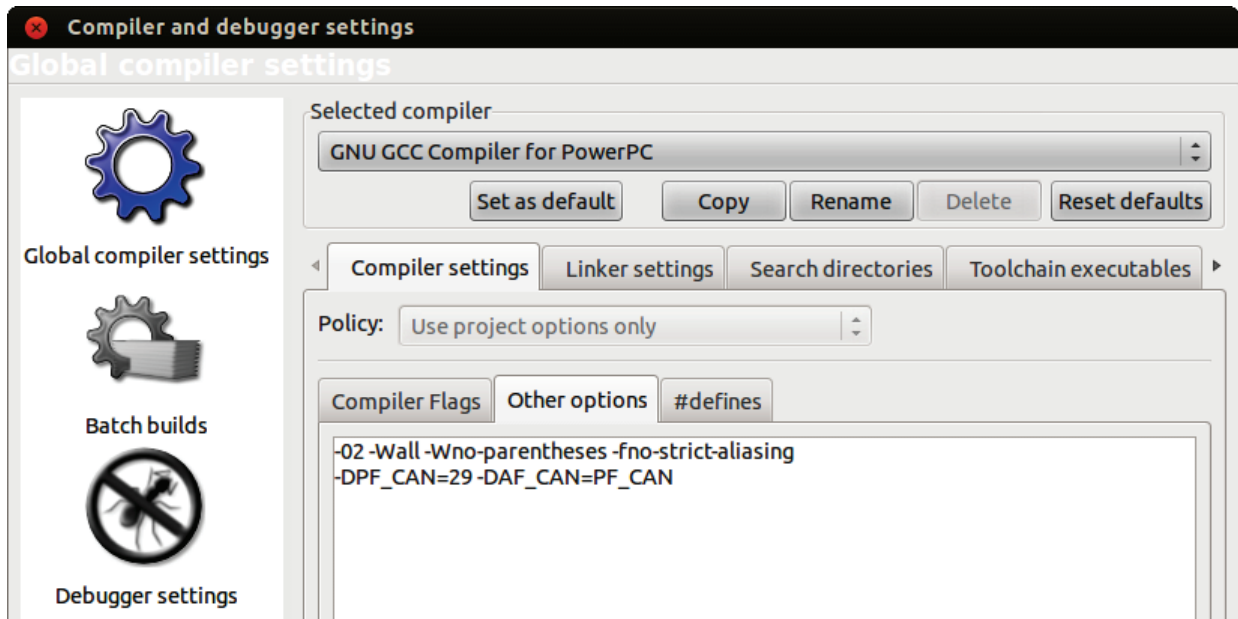


8.3.1.3 Setup the Code::Blocks IDE

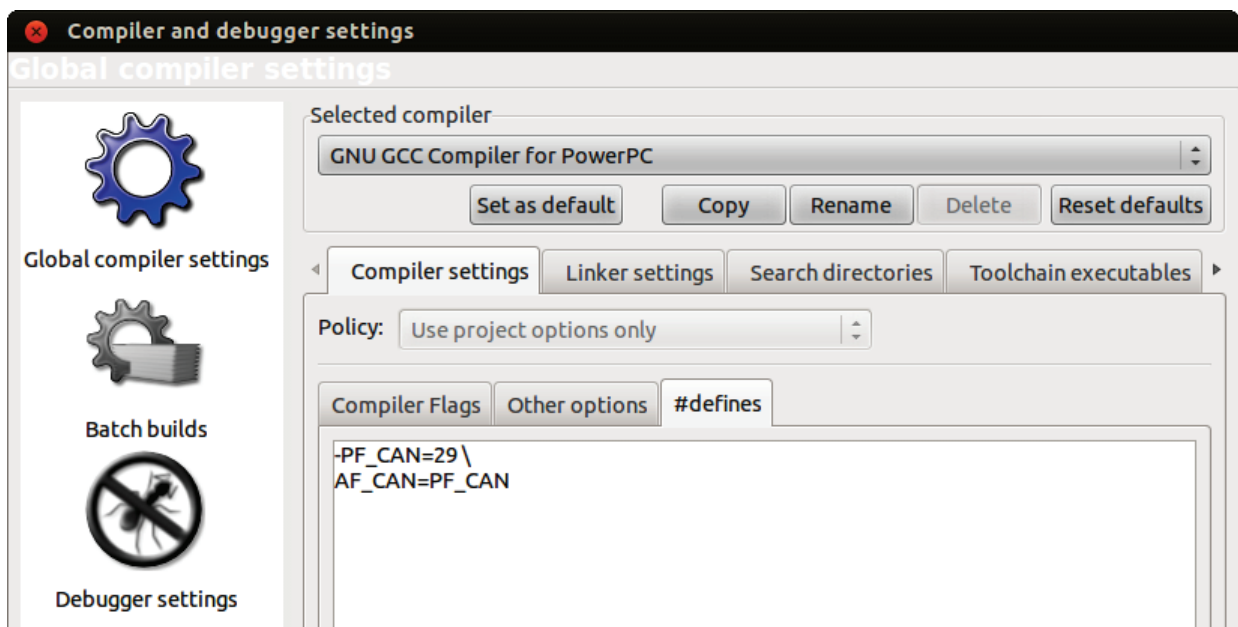
After Code::Blocks has been installed on your PC, it must be configured so that it can be used with the TC3G.

1. Start Code::Blocks
2. In the main menu select "Settings | Compiler and Debugger..."
3. Select the compiler: "GNU GCC Compiler for PowerPC"

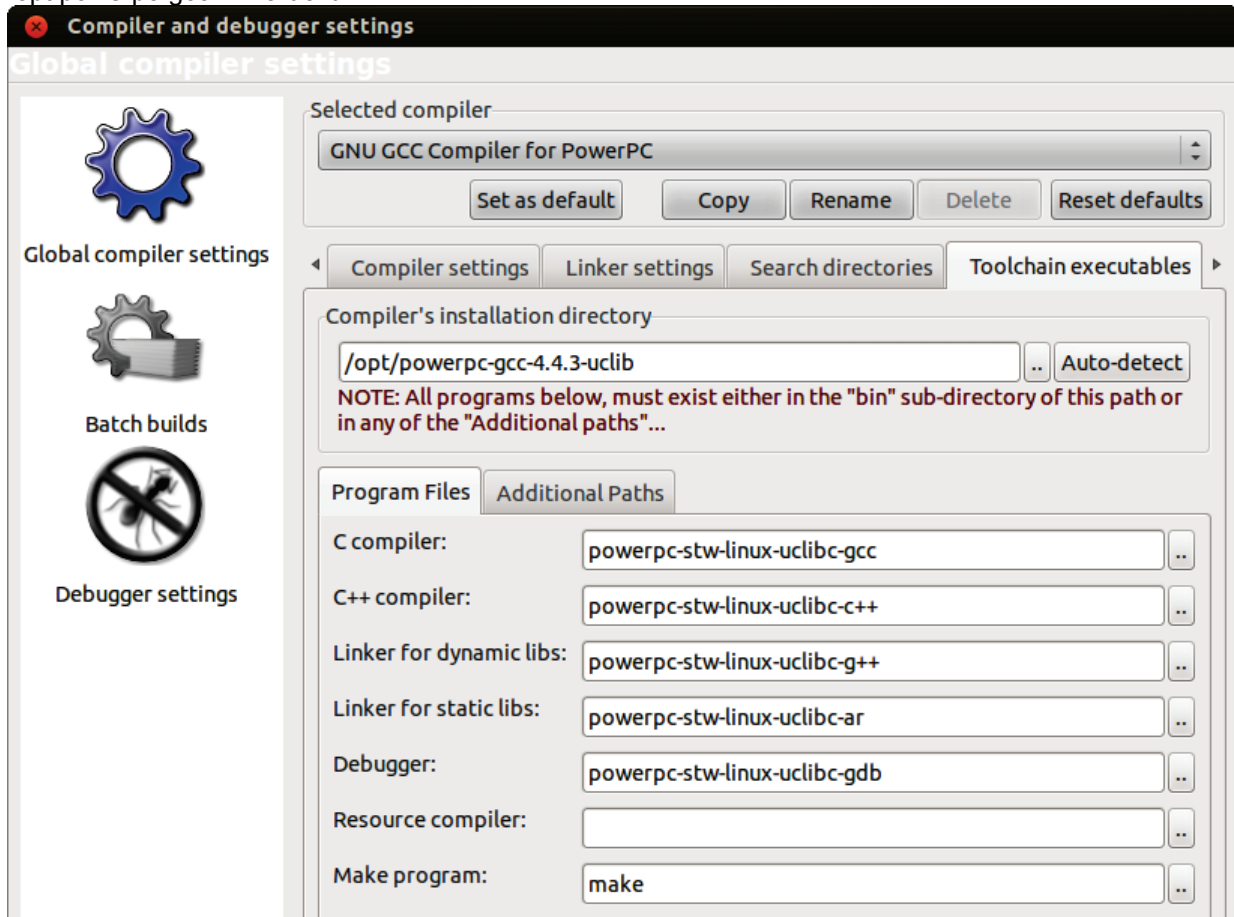
4. Navigate to the tab "Compiler settings | Other option":



5. Enter in the tab "Other options" the following lines:
`-O2 -Wall -Wno-parentheses -fno-strict-aliasing`
`-DPF_CAN=29 -DAF_CAN=PF_CAN`
6. Since many applications need access to the CAN Bus, add the following defines to your build options. Select the tab "#defines" in the tab "Compiler settings". Enter following line and click "OK":
`PF_CAN=29 \`
`AF_CAN=PF_CAN`

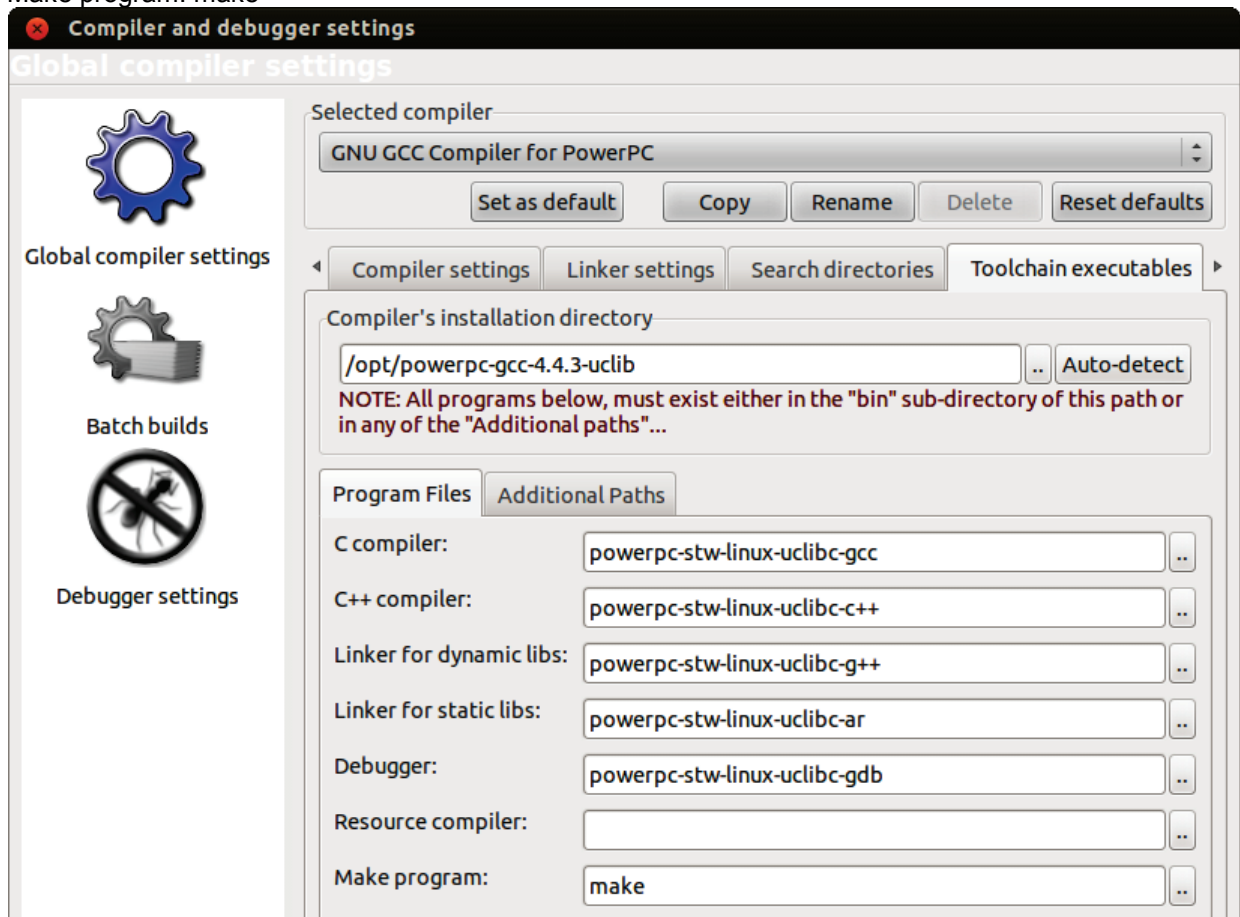


7. Navigate to the tab "Toolchain executables" and enter in "Compilers's installation directory" the following line:
/opt/powerpc-gcc-4.4.3-uclib/



8. Navigate to the tab "Program Files" enter the following settings:

- C compiler: powerpc-stw-linux-uclibc-gcc
- C++ compiler: powerpc-stw-linux-uclibc-c++
- Linker for dynamic libs: powerpc-stw-linux-uclibc-g++
- Linker for static libs: powerpc-stw-linux-uclibc-ar
- Debugger: powerpc-stw-linux-uclibc-gdb
- Make program: make



9. Click "Set As Default" to set the settings for the default settings.

10. Click "OK" to accept the changes.

8.3.1.4 Setup the Compiler for Debugging

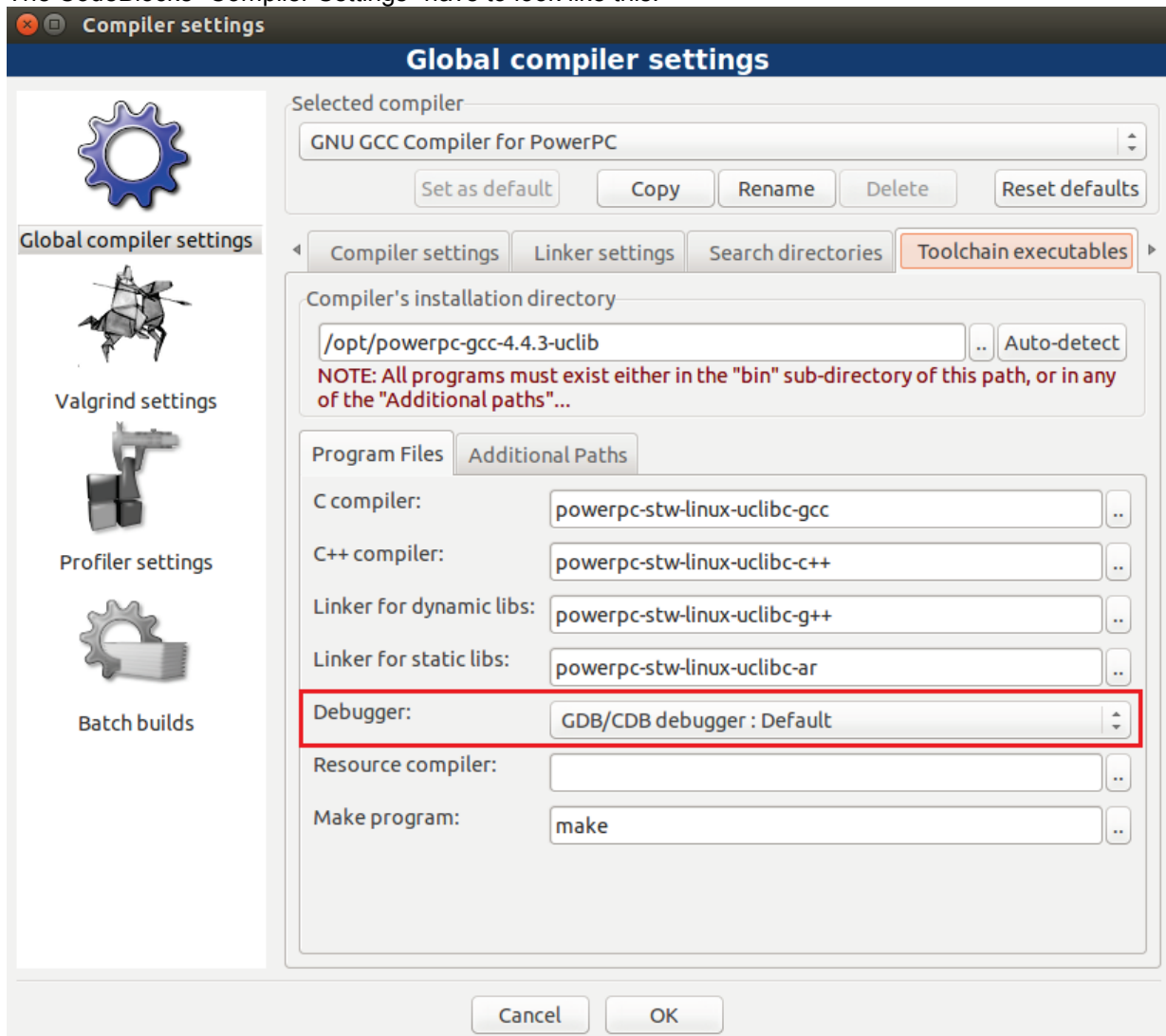
It is possible to use the GNU Debugger to debug applications under Linux. The description of the GNU Debugger (GDB) is in combination with Code::Blocks. No additional software is needed.

How to prepare debugging

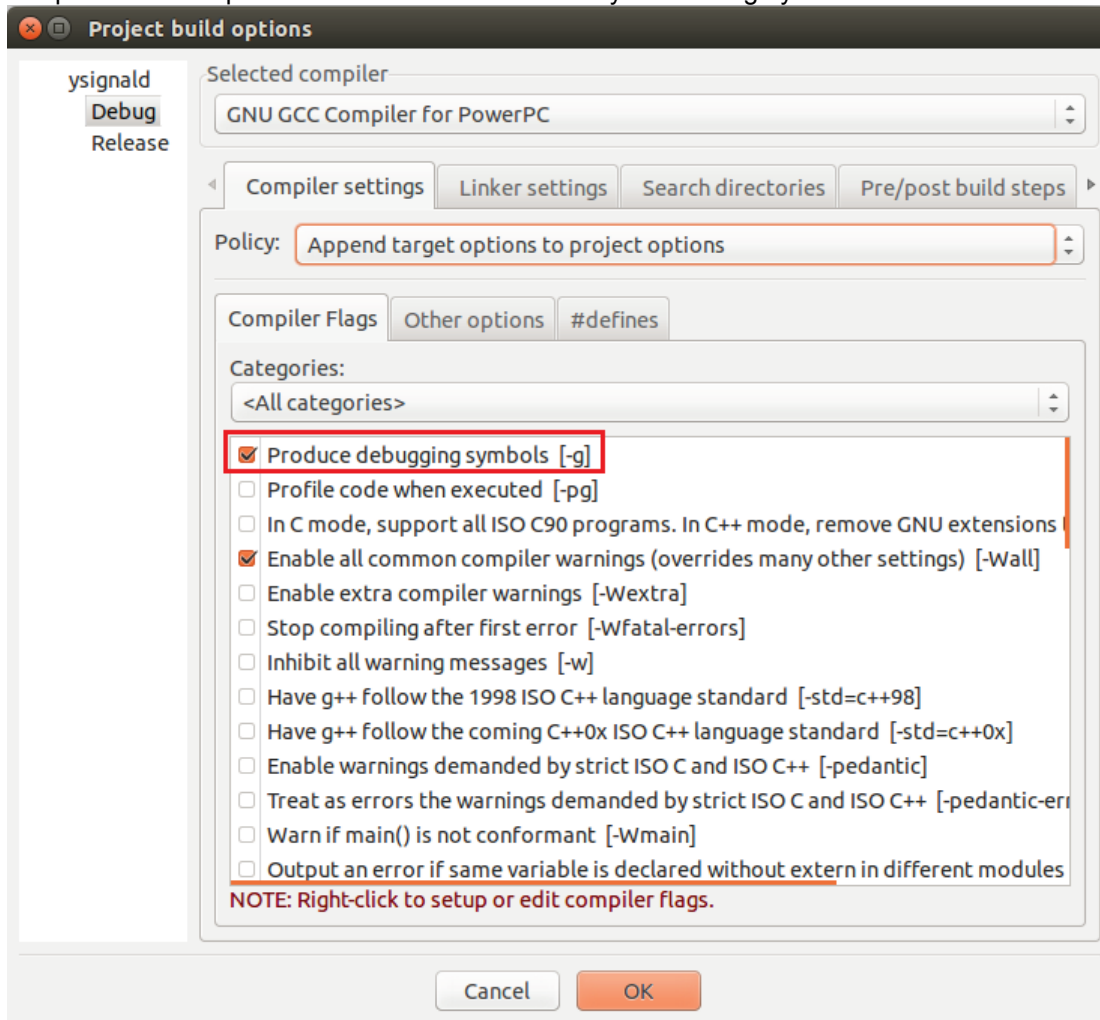
Precondition: The toolchain and the IDE CodeBlocks is installed and set up

1. In case of a 64-bit operation system, you need to install 'lib32ncurses5'. For 32-bit operation systems this step can be skipped.
To install the library via command line use:
`sudo apt-get install lib32ncurses5`

2. Check the Code::Blocks Compiler Settings.
The CodeBlocks "Compiler Settings" have to look like this:

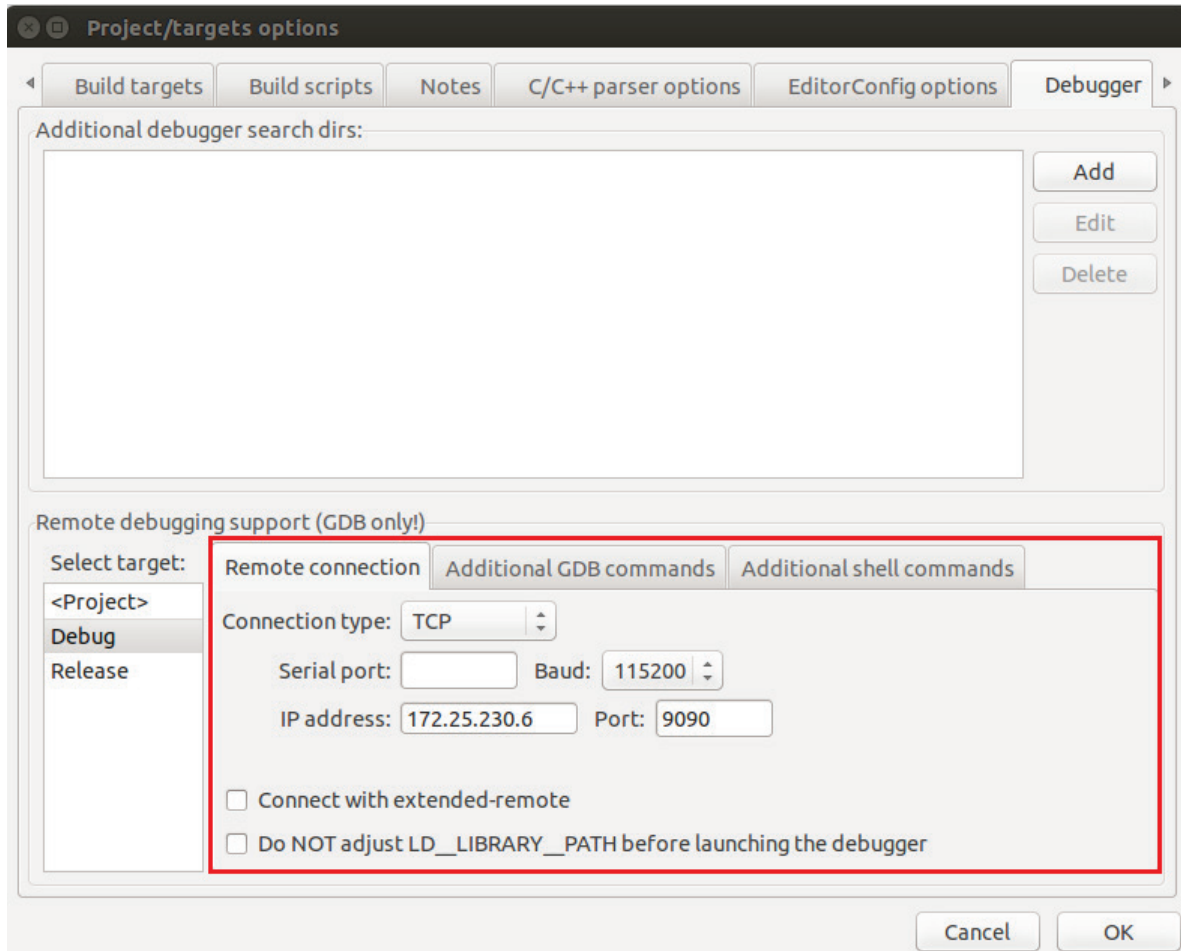


3. Adapt the "Build Options" in order to build the binary with debug symbols:



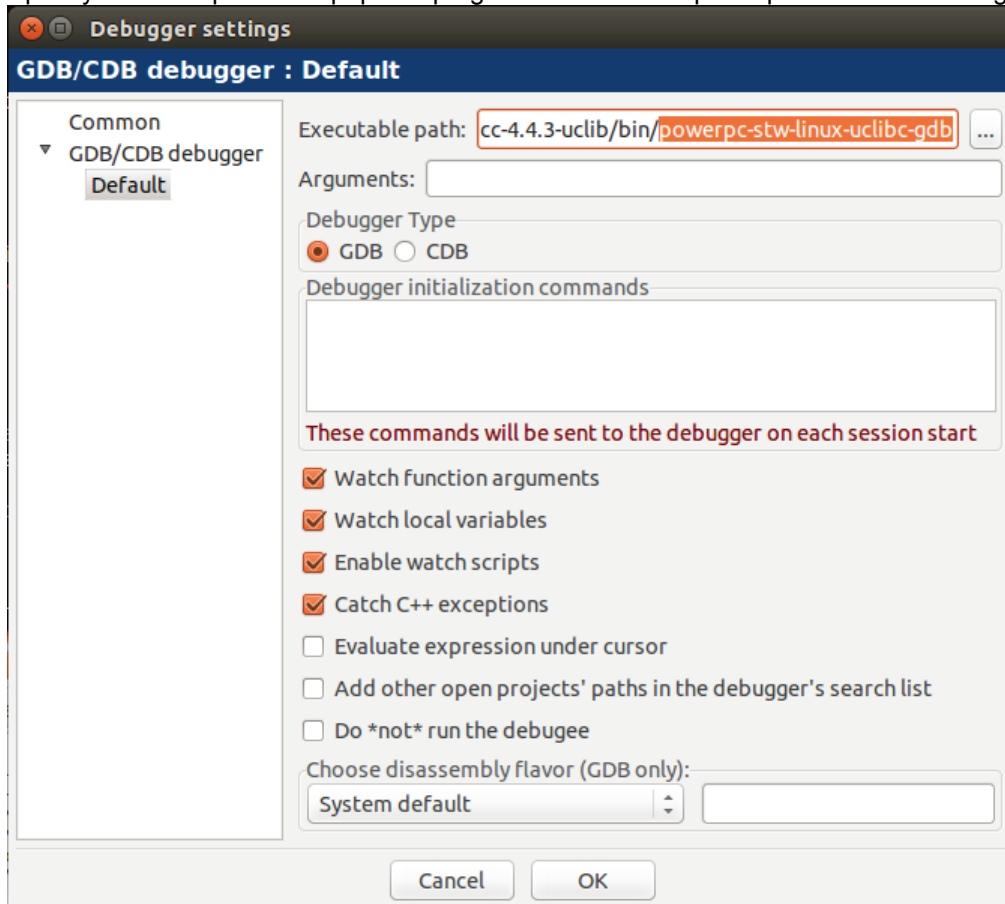
4. In the main menu select "Project | Properties... | Debugger" and set up the remote connection settings:

- Serial port: stays empty
- Baud: 115200
- IP address: the IP adresse of your TC3G
- Port: 9090



5. In the main menu select "Settings | Debugger..." and go to Default

6. Specify the GDB path to: /opt/powerpc-gcc-4.4.3-ucLib/bin/powerpc-stw-linux-ucLibc-gdb



7. Make the built binary accessible from the TC3G: Use for example the NFS (see "[NFS](#)" on page 336), or if available use an USB drive.
8. Start the gdbserver on the TC3G (with your application e.g. ysignal):
 - # gdbserver :9090 ysignal
 - Process ysignal created; pid = 1436
 - Listening on port 9090
9. Start the debugging in Code::Blocks:
 - Debug -> Target's default
 - Debug -> Start / Continue (F8).

```
#
# gdbserver :9090 ysignal /etc/init.d/ysignal.config
Process ysignal created; pid = 1277
Listening on port 9090
Remote debugging from host XXX.XXX.XXX.XXX
Error in config file. No valid Log_File found
No log file entry found in config file -> send log msg to stdout
29.10.15 07:37:37 This TC3G variant does support GPS
```



Continue with Create Own Application (see "[Create Own Application](#)" on page 295)

8.3.2 Windows



8.3.2.1 Install the Toolchain

This chapter provides a step by step tutorial to install the toolchain on a 32-/64-Bit Windows 7 System.

1. Download the toolchain installer from the STW FTP server:
URL: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>) ESX-TC3G/DeploymentPackage_vX_XXrX/devkit_vX_XXrX/toolchain/windows/gcc_4_4_3/:

Index von ftp://tc3@esx-tc3.de/ESX-TC3G/DeploymentPackage_v1_00r0/devkit_v1_00r0/toolchain/windows/gcc_4_4_3/			
 In den übergeordneten Ordner wechseln			
Name	Größe	Zuletzt verändert	
 Setup.exe	31919 KB	17.09.2014	00:00:00

2. Download the add-on installer from STW FTP server
URL: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>) ESX-TC3G/DeploymentPackage_vX_XXrX/devkit_vX_XXrX/toolchain/windows/gcc_4_4_3_Add-ons/:

Index von ftp://tc3@esx-tc3.de/ESX-TC3G/DeploymentPackage_v1_00r0/devkit_v1_00r0/toolchain/windows/gcc_4_4_3_Add-ons/			
 In den übergeordneten Ordner wechseln			
Name	Größe	Zuletzt verändert	
 Setup.exe	21978 KB	17.09.2014	00:00:00

3. Execute the installers:



NOTE:

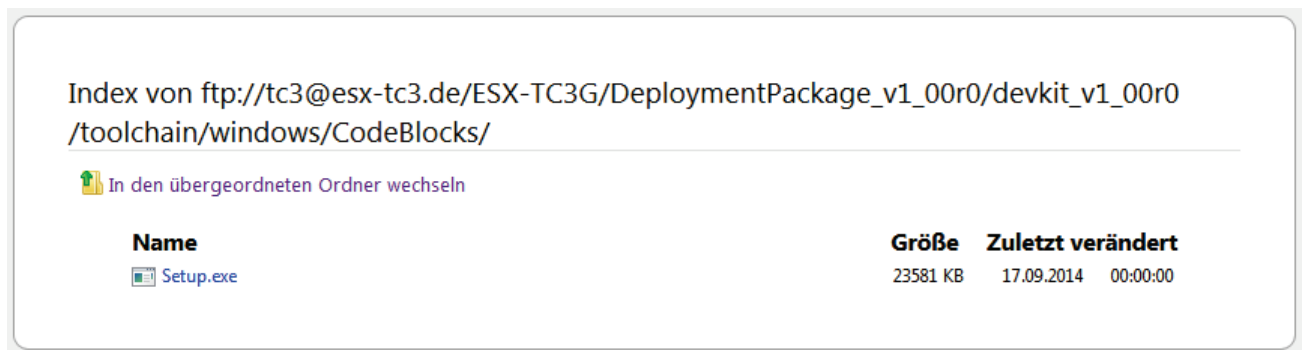
For Windows 10: Make sure to execute the installers as administrator.

- a. Execute the toolchain installer (Setup.exe binary) and follow the instructions on the screen.
- b. Execute next the add-on installer (Setup.exe binary) and follow the instructions on the screen.

8.3.2.2 Install the Code::Blocks IDE

Code::Blocks can be used to compile ready to go executables for the TC1. Especially for larger projects, CodeBlocks is recommended, since it provides a good overview over your software modules to be used.

1. Download the Code::Blocks IDE from STW FTP server:
URL: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>) ESX-TC3G/DeploymentPackage_vX_XXrX/devkit_vX_XXrX/toolchain/windows/CodeBlocks/:



2. Execute the IDE installer (Setup.exe binary) and follow the instructions on the screen.



NOTE:

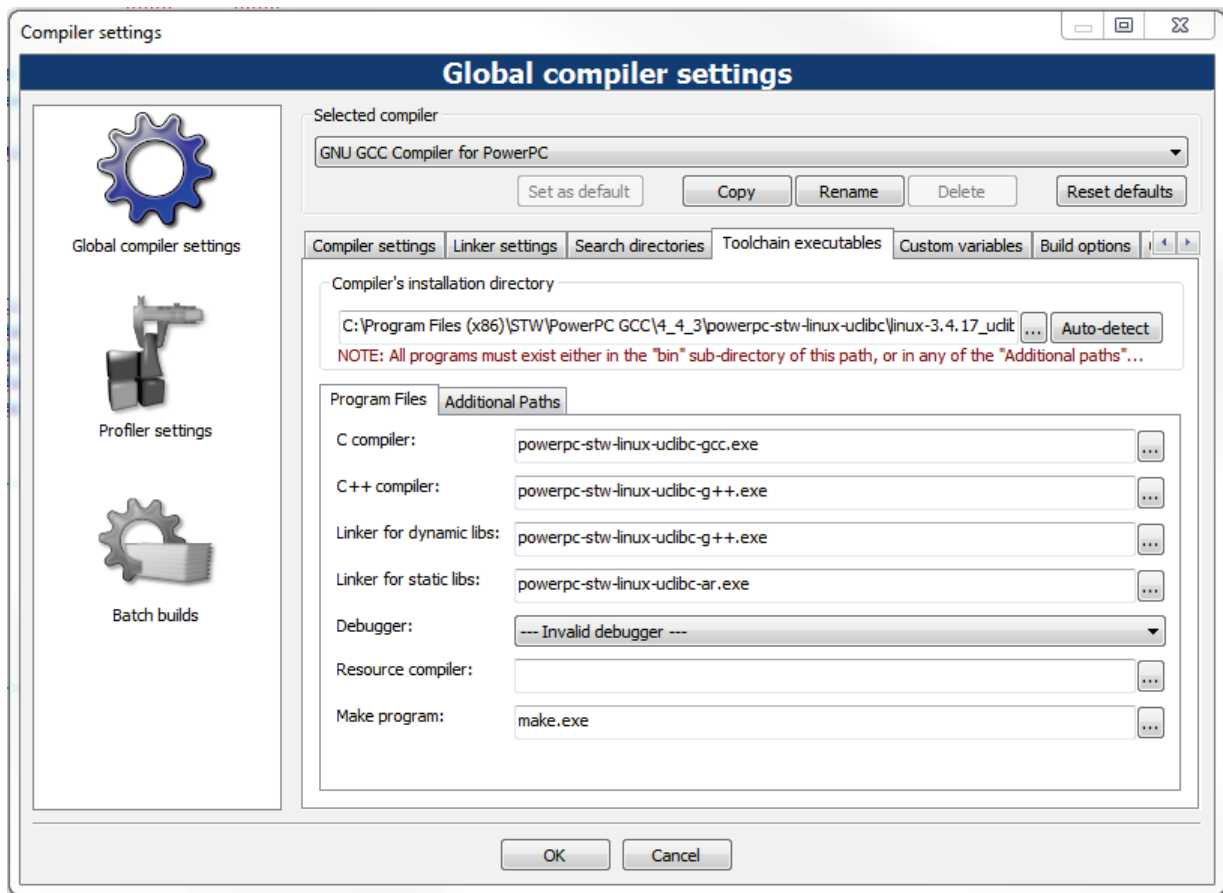
For Windows 10: Make sure to execute the installer as administrator.

8.3.2.3 Setup the Code::Blocks IDE

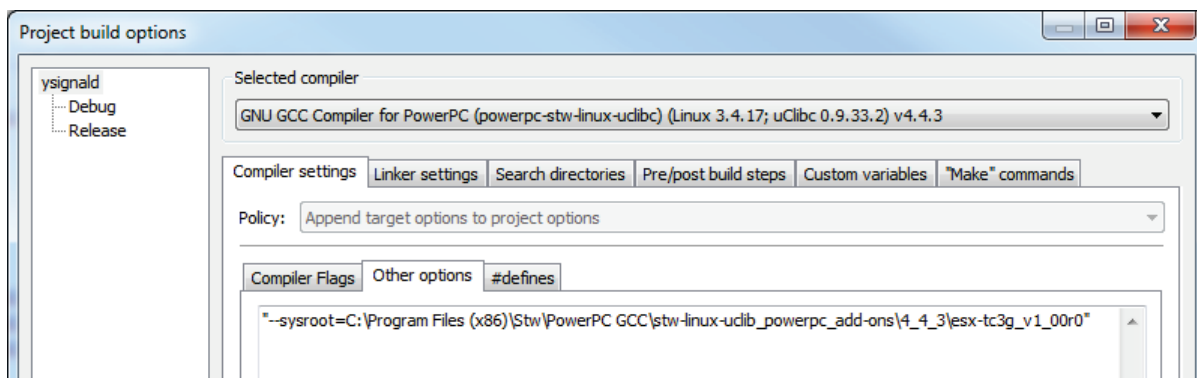
After Code::Blocks has been installed on your PC, it must be configured so that it can be used with the TC3G.

1. Start Code::Blocks
2. In the main menu select "Settings | Compiler and Debugger..."
3. Select the compiler: "GNU GCC Compiler for PowerPC"
4. Navigate to the Tab "Toolchain executables"
5. Select the "Compiler's installation directory". Set the path depending where Code::Bloccks is installed.
On a 64 bit system, Code::BLoccks is installed in "Program Files (x86)"

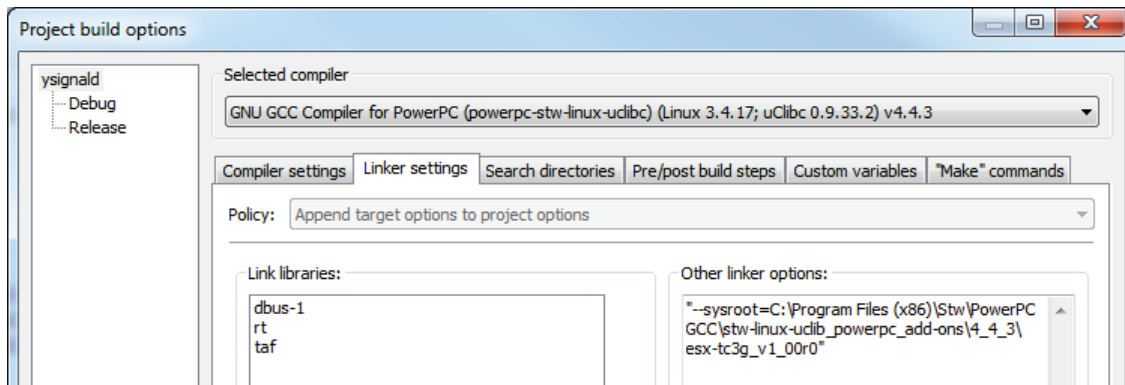
6. Set the files in the tab "Program Files" like in the following compiler settings:



7. Click "OK" to accept the changes.
8. In the main menu select "Project | Build options..."
9. Go to the tab "Compiler settings" | "Other options" and add the following line:
For a 32 bit system: `--sysroot=C:\Program Files\STW\PowerPC GCC\stw-linux-udlibc_powerpc_add-ons\4_4_3\esx-tc3g_v1_00r0`
For a 64 bit system: `--sysroot=C:\Program Files (x86)\Stw\PowerPC GCC\stw-linux-udlibc_powerpc_add-ons\4_4_3\esx-tc3g_v1_00r0`



10. Go to the tab "Linker settings"



11. Enter in the field "Link libraries:"

- dbus-1: This links the D-Bus library
- rt: This links the run time library
- taf: This links the Teleservice Application Framework library;

12. Enter in the field "Other linker options:" the following line:

For a 32 bit system: --sysroot=C:\Program Files\STW\PowerPC GCC\stw-linux-uclibc_powerpc_add-ons\4_4_3\esx-tc3g_v1_00r0

For a 64 bit system: --sysroot=C:\Program Files (x86)\Stw\PowerPC GCC\stw-linux-uclibc_powerpc_add-ons\4_4_3\esx-tc3g_v1_00r0

13. Click "OK" to accept the changes.

8.3.2.4 Setup the Compiler for Debugging

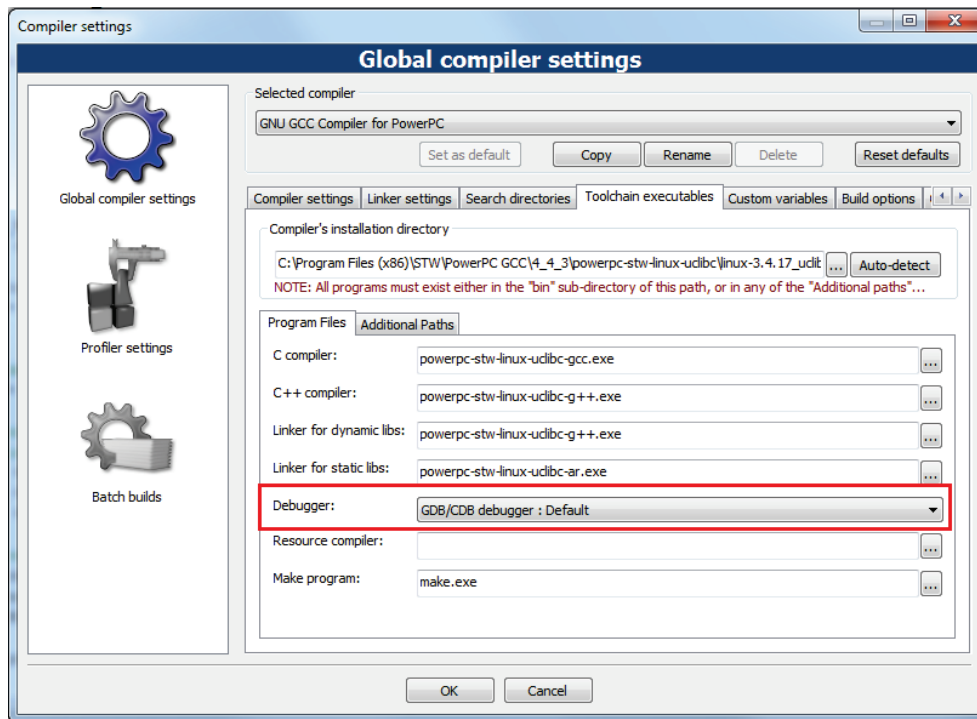
Use the GNU Debugger to debug applications under Windows. The description of the GNU Debugger (GDB) is in combination with the Code::Blocks IDE. No additional software is needed.

Precondition:

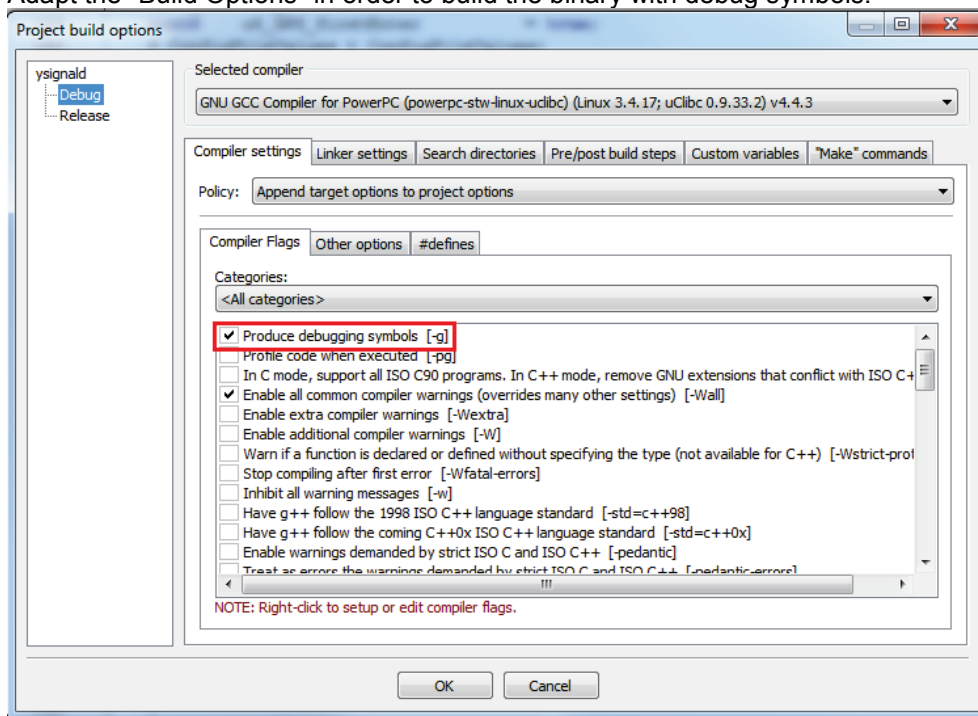
The toolchain and the Code::Blocks IDE is installed and configured.

How to prepare debugging

1. Check the Code::Blocks Compiler Settings.
The CodeBlocks "Compiler Settings" have to look like this:

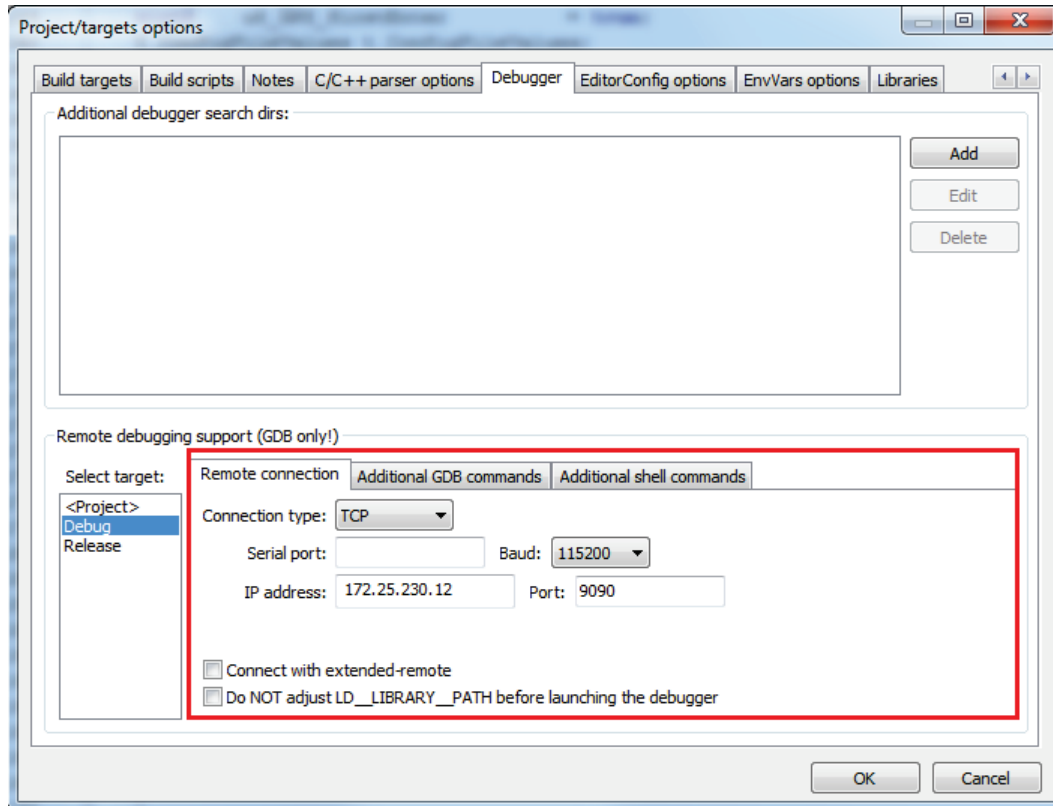


2. Adapt the "Build Options" in order to build the binary with debug symbols:



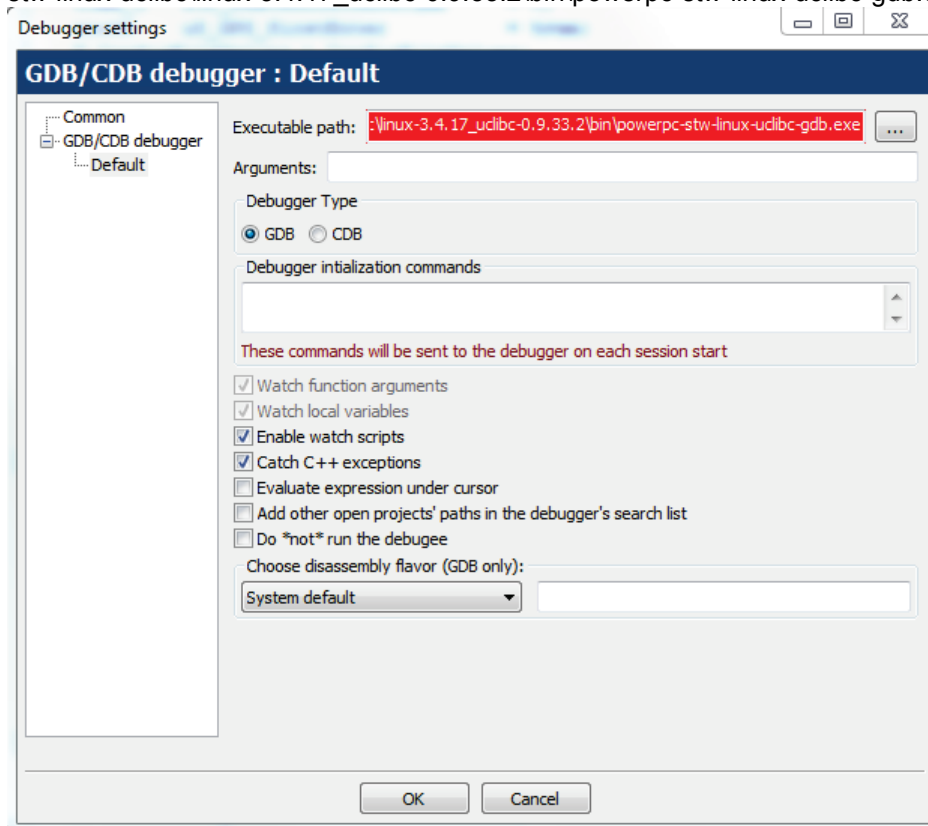
3. In the main menu select "Project | Properties... | Debugger" and set up the remote connection settings:

- Serial port: stays empty
- Baud: 115200
- IP address: the IP adresse of your TC3G
- Port: 9090



4. In the main menu select "Settings | Debugger..." and go to Default

5. In "Executable path" specify the GDB path to : C:\Program Files (x86)\Stw\PowerPC GCC\4_4_3\powerpc-stw-linux-uclibc\linux-3.4.17_uclibc-0.9.33.2\bin\powerpc-stw-linux-uclibc-gdb.exe

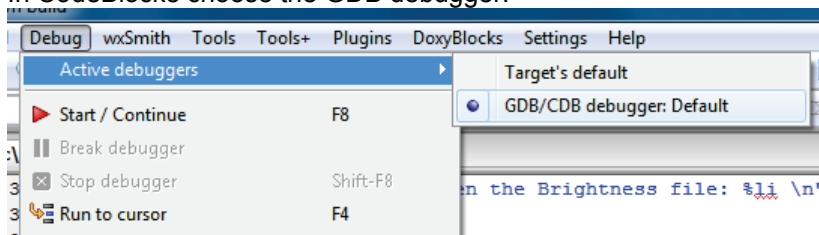


6. Make the built binary accessible from the TC3G: Use for example the NFS (see "[NFS](#)" on page 336), or if available use an USB drive.

7. Start the gdbserver on the TC3G (with your application e.g. ysignal):

- # gdbserver :9090 ysignal
- Process ysignal created; pid = 1436
- Listening on port 9090

8. In CodeBlocks choose the GDB debugger:



9. Click "Debug | Start" the debugging:

```
#
# gdbserver :9090 ysignalld /etc/init.d/ysignalld.config
Process ysignalld created; pid = 1277
Listening on port 9090
Remote debugging from host XXX.XXX.XXX.XXX
Error in config file. No valid Log_File found
No log file entry found in config file -> send log msg to stdout
29.10.15 07:37:37 This TC3G variant does support GPS
```

Continue with Create Own Application (see "[Create Own Application](#)" on page 295)

8.3.3 Examples

The example folder DeploymentPackage v3_00r1/devkit v3_00r2/examples contains a basic Code::Blocks project.

This example project describes how to use:

- CAN
- beeper
- motion
- temperature sensor.

Folder structure of this Code::Blocks example project:

Folder name	Definition of files
doc	Additional information to the project
libs	External libraries which are necessary to compile the project or sub projects
pjt	Contains the CodeBlock project
result	Executable output files
src	Contains all header and source files for the project

Make sure to crete the same folder structure for your application.

8.3.4 Libraries

STW provides the following libraries for development:

TAF

The Teleservice Application Framework (TAF) is the core of each teleservice application. It contains all necessary daemons and the corresponding library to interact with the daemons.

With the TAF it is possible to:

- create data loggers
- manage the network
- send or receive SMS
- communicate with external servers

HTTP Tiny

The http tiny library extends the functionality of your application with the possibility to perform HTTP queries like post and get. For further information please refer to the `http_lib.h`.

CANopen

The libraries provide an easy way to access and communicate with slave nodes like I/O extension modules.

STW devices

- DIOS library: This library provides higher level functions to interact with ESX-DIOS and ESX-DIOM modules.
- IOX library: This library provides higher level functions to interact with ESX-IOX modules.

Non-STW devices

COL2 library: This CANopen Layer 2 library can be used for communication with Non-STW CANopen devices

J1939

The library provides an easy way to access j1939 nodes over CAN.

9 Update the Device

The Board Support Packages (BSP) includes all components which are required for a boot able system. In order to update the TC3G to the latest BSP (see "[BSP Components](#)" on page 321), use the BSP updater.

The BSP updater for each TC3G variant contains the following folder structure:

Folder name	Description
data	Contains the data files for the system (U-Boot binary, Kernel binary, Root file system image, Flattened device tree blob)
make_FIT	Includes executable to generate a FIT image (for Linux)
scripts	Contains TeraTerm and shell scripts
tftp	Contains the Tftpd32 program (a open source tftp server for Windows)
tterm	Contains the TeraTerm program (a open source terminal program for Windows)
tc3g_updater_for_linux	Executable which starts the Linux-update-process
tc3g_updater_for_windows.bat	Executable which starts the Windows-update-process
tc3g_updater.log	Log-file of the update procedure. (Appears after an update.)
config.ttl	General settings for the update process
readme	Help file

How to update the TC3G

To perform an update, follow the steps:

1. Customize the configuration file (config.ttl)
2. Execute the start script, depending on the operating system:
 - Windows: Run the start script = tc3g_updater_for_windos.bat
 - Linux: Check if the file permission '-x' is already set for the start script = tc3g_updater_for_linux and run the start script
3. Follow the instructions on the screen.

9.1 BSP Components

The Board Support Packages (BSP) includes all components which are required for a bootable system. For each TC3G variant exists one specific BSP. These BSPs can be found on the STW FTP server.

BSP path on STW FTP server: <ftp://esx-tc3.de/> (see <ftp://esx-tc3.de/> - <ftp://esx-tc3.de/>)

U-Boot (u-boot.bin)

The universal bootloader starts the system. The CPU of the system can only execute program code that is located in the ROM (read only memory) or in the RAM (random access memory) of the TC3G. The Linux operating system, the root file system and all user applications are stored in the NOR flash. The NOR flash is a nonvolatile data storage. At startup, the U-Boot copies all files to the RAM that are necessary for running the system.

U-Boot environment variables (u-boot_env.txt)

The environment variables of the bootloader is a set of defines. These defines describe different startup routines or the IP configuration. All settings are necessary for a trouble free system start.

Root File System (rootfs.ubi)

The Root File System is located on the same partition as the root directory is located on. On this filesystem all the other filesystems are mounted, for example: logically attached to, when the system is booted up, for example during start up.

A filesystem is a hierarchy of directories (also referred to as a directory tree) that is used to organize files on a computer system. On Linux and other Unix-like operating systems, the directories start with the root directory, which contains a number of subdirectories, which can contain more subdirectories.

Flattened Device Tree (stw5200b.dtb)

The Linux kernel expects certain information about the hardware that it is running on. This information includes the Flattened Device Tree (FDT).

Kernel (vmlinux.img)

The Linux kernel is the core of the operating system.

9.2 Windows Updater

The Windows Updater is a software to update the TC3G from a Windows OS.



NOTE:

To be able to perform the update super user permissions are mandatory.

How to use the Windows Updater

The update procedure can be divided into the following steps:

1. Customize the configuration file config.ttl
2. Connect the TC3G via RS232 to the host PC from that the update has to be executed
3. Execute the tc3g_updater_for_windows.bat
4. Follow the instructions on the screen

How to configure the Updater file config.ttl:

The configuration file config.ttl includes a section called "User Settings".

In this section the user can select the components of the BSP (see "[BSP Components](#)" on page 321) that have to be updated.

```
;*****
; START OF USER SETTINGS.                                     ***
; Do not make any changes to this file except in this section!!! ***
;*****
```

1. Open the config.ttl with an editor of your choice.
2. Select the COM port for the Windows Updater, where the TC3G is connected to the PC:

```
;*****
; Select COM port for the Windows Updater
;*****
VarComPort      = '/C=1' ;COM1
;VarComPort      = '/C=2' ;COM2
;VarComPort      = '/C=3' ;COM3
;VarComPort      = '/C=4' ;COM4
```

3. Set the update source:

```
;*****
; Network settings for TFTP data download
;*****
VarTftpDownload = 1 ; 1.. Update via TFTP
                  ; 0.. Update via RS232 (only possible for the Windows Updater)
                  ; 0.. Update via USB (only possible for the Linux Updater)

VarServerIP     = '' ; TFTP server ip (ip of host PC)
VarClientIP     = '' ; ip of TC3G
VarNetMask      = '' ; network mask
VarGatewayIP    = '' ; gateway ip (not required when ServerIP and ClientIP are in same subnet)
```

- a. Update via TFTP download: Set VarTftpDownload = 1
In case of an update via TFTP, the network settings of the config.ttl file needs to be completed (for example: set the IP address, set the network mask).
 - b. Update via RS232 download: Set VarTftpDownload = 0. The Update via RS232 is not recommended, because it will take a very long time.
4. Select the components that shall be updated:

```
;*****
; Components to be installed
;*****
VarComponentUboot      = 1 ; install uboot
VarComponentUbootEnvVars = 1 ; update uboot environment variables
VarComponentKernel     = 1 ; install kernel
VarComponentFdt         = 1 ; install flattened device tree
VarComponentRootfs      = 1 ; install root file system
VarPrepareNandFlash     = 1 ; prepare (format) the NAND data flash
```

- a. Go to the sub section "Components to be installed". By default all components are set to 1 and will be updated.
 - b. Set all components that must be updated to 1. Set each component to 0 when it is not needed to be updated.
5. Save your settings and close the file.

The complete update procedure will be logged in a file. The default name of this log file is tc3g_updater.log.

How to test the RS232 connection

1. Open a terminal program on the host PC (for example Hyperterm, or the TeraTerm that is supplied in the tterm folder)
2. Setup the serial port where the device is connected to the following settings:
 - a. 115200 baud
 - b. 1 stop bit
 - c. no parity bit
 - d. no handshake.
3. Switch on the power supply of the TC3G controller.

You should see the boot messages of the U-Boot and Linux kernel in the terminal.

How to start the Updater



WARNING:

Do not interrupt the update process!

If the update process is interrupted during the update of the U-Boot component, it is not possible to access the board from an external interface any more.

In this case the controller has to be sent back to STW for reprogramming.

The U-Boot is updated in one of the last update steps.

Interrupting the update process during any other step is not too critical, since it can just be restarted.

1. Switch off the device.
2. Start the batch file tc3g_updater_for_windows.bat. The batch file opens the Tftpd32 and the TeraTerm programs and starts the update script.
3. When the script prompts to switch on the device, then do it.

9.3 Linux Updater

The Linux updater is a software tool to update the TC3G from a Linux OS. It can be found in the Deployment Package v1.00r1 or newer.

Over the GUI the components for the updated can be selected. The settings for the update are stored in the configuration file config.ttl.

The Linux updater provides the possibility to update over USB, when a USB device is available on the board.



NOTE:

To be able to perform the update super user permissions are mandatory.

It is only possible to update an U-Boot 2012.10 U-Boot. The U-Boot 2008.10 is no longer supported.

For the first update over TFTP download from a PC, a connection to the internet is required.

On the first update, a TFTP tool will be automatically installed to your PC.

This TFTP tool performs the update.

How to use the Linux Updater

The update procedure can be divided into the following steps:

1. Connect the TC3G via RS232 to the host PC from that the update has to be executed
2. Execute the start script `tc3g_updater_for_linux`
3. Follow the instructions on the screen
(For some steps during the update process super user permissions are mandatory. Therefore, the updater will ask for the sudo password.)

How to test the RS232 connection

1. Open a terminal program on the host PC (for example Hyperterm, or the TeraTerm that is supplied in the `tterm` folder)
2. Setup the serial port where the device is connected to the following settings:
 - a. 115200 baud
 - b. 1 stop bit
 - c. no parity bit
 - d. no handshake.
3. Switch on the power supply of the TC3G controller.

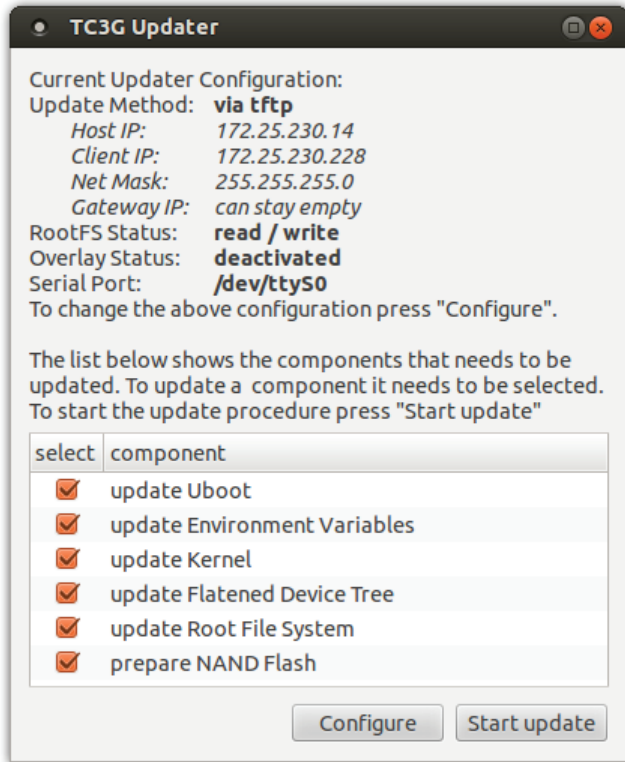
You should see the boot messages of the U-Boot and Linux kernel in the terminal.

How to configure the Updater over the GUI:

The update settings are stored in the configuration file `config.ttl`.

1. Execute the start script `tc3g_updater_for_linux`

2. Over the GUI select the components that must be updated:



How to configure the Updater file config.ttl without using the GUI:

The configuration file config.ttl includes a section called "User Settings". In this section the user can select the components of the BSP (see ["BSP Components"](#) on page 321) that have to be updated.

The start script tc3g_updater_for_linux can be started without the GUI, by using parameter "-n".

In this case the config.ttl has to be changed with an editor of your choice.

1. Open the config.ttl with an editor of your choice.
2. Set the update source:
 - a. Update via TFTP download: Set VarTftpDownload = 1
In case of an update via TFTP, the network settings of the config.ttl file needs to be completed (for example: set the IP address, set the network mask).
 - b. Update via USB download: Set VarTftpDownload = 0. For updating the device via USB you need a USB storage device with a minimum size of 128MB (FAT32 format).
3. Configure the status of the root file system: Set VarRootFileSystemStatus to 1 for read and write or to 0 for read only
4. Select the status of the overlay filesystem (see ["Overlay Filesystem"](#) on page 79) (unionFS): Set VarOverlayStatus to 1 to be activated or to 0 for being deactivated
5. Select the components that shall be updated:
 - a. Go to the sub section "Components to be installed". By default all components are set to 1 and will be updated.
 - b. Set all components that must be updated to 1. Set each component to 0 when it is not needed to be updated.
6. Save your settings and close the file.

The complete update procedure will be logged in a file. The default name of this log file is tc3g_updater.log.

How to start the Updater

**WARNING:**

Do not interrupt the update process!

If the update process is interrupted during the update of the U-Boot component, it is not possible to access the board from an external interface any more.

In this case the controller has to be sent back to STW for reprogramming.

The U-Boot is updated in one of the last update steps.

Interrupting the update process during any other step is not too critical, since it can just be restarted.

**WARNING:**

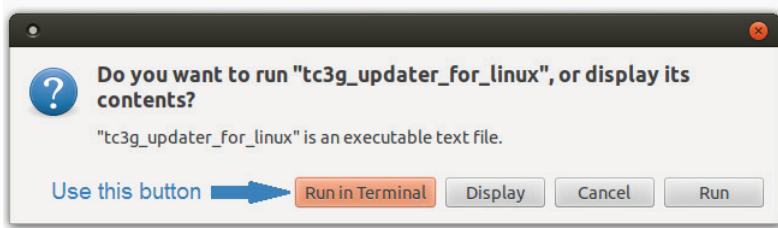
Do not use an U-Boot .bin file from an older BSP. Downgrading from U-Boot 2012.10 to U-Boot 2008.10 will destroy the U-Boot. After that, the controller has to be sent back to STW for reprogramming.

1. Set the file permissions of the start script `tc3g_updater_for_linux` of the updater to "executable":

```
user@pc-name:~$ chmod +x tc3g_updater_for_linux
# (has to be executed only once)
```

2. Execute the start script:

- a. Execute the script via mouse click:



- b. Or, via terminal:

```
user@pc-name:~$ ./tc3g_updater_for_linux
```

When starting the updater via terminal, it is possible to use command line parameters:

```
user@pc-name:~$ ./tc3g_updater_for_linux -n
# starts the updater without the GUI, uses the configuration from the config.ttl file

user@pc-name:~$ ./tc3g_updater_for_linux -h
# prints a short help showing all possible parameters
```

9.4 STW Update Mechanism

9.4.1 STW Update Procedure

General

The UBoot bootloader of the STW Linux devices provides an update procedure to update all firmware components of the target. There are three methods to provide the update image (file that contains the update data):

Update via USB:

The update image is provided by a USB flash disk.

Update via Dataflash (internal NAND Flash):

The update image is stored in the internal data flash. A user space application (Linux application) is responsible to receive the image file from any source and to write it to the data flash.

Update via TFTP Download:

Before an update via TFTP download could be performed, some preliminary steps need to be taken:

- A TFTP server must be installed and configured on the host PC for example TFTPd.
- The update image has to be moved into the working folder of the TFTP server.

After these settings, the update image could be downloaded from the TFTP server via the ethernet interface. Other interfaces like WLAN or GSM are not supported.

Update Procedure

During system startup, the UBoot checks for each of the supported update methods if it is enabled and if an update image file is available. If no update is available, a regular system start is performed. If an update is available, the image file is loaded into the RAM and the update process is started. See flow chart diagrams (see "[STW Update Procedure Flow Charts](#)" on page 333) for details.

Update Image File Format

The format of the update image file is the UBoot FIT-Image format (Flatted Image Tree). For details see the UBoot documentation (folder "doc/uImage.FIT" in UBoot source code, available e.g. here: <http://git.denx.de/?p=u-boot.git;a=summary> (<http://git.denx.de/?p=u-boot.git;a=summary>)).

The file can contain several data nodes. Two types of data nodes are supported:

- Update Data Image: Data to be programmed to the NOR flash of the target. The address to which the data will be programmed is provided by the update image file (data load address)
- UBoot Script Image: Sequence of commands that are executed by the UBoot (as if the command were entered in the UBoot shell, e.g. setenv x y, saveenv).

The data nodes are processed sequentially. If an error occurs, the update process is terminated.

Password Protection

The first node in the update image file is the password node. It provides a password that is compared to the password stored in the UBoot environment variables. If they do not match, the update process is terminated. This allows a simple security mechanism to avoid flashing the wrong application to the device (e.g. flashing application "coffee machine" to a device installed on a concrete mixer).

The aim of the password mechanism is to avoid unintended changes of the target device. It is not designed to provide security against intended manipulation.

System Configuration Options

There are a number of configuration options available to control the update procedure.

UBoot Environment Variables

General settings like enabling / disabling of an update method are controlled by UBoot environment variables.

Environment Variable	Description
update_en_usb	Update via USB flash disk: <ul style="list-style-type: none"> 0: disabled 1: enabled
update_en_df	Update via internal dataflash (NAND flash): <ul style="list-style-type: none"> 0: disabled 1: enabled
update_en_tftp	Update via TFTP download: <ul style="list-style-type: none"> 0: disabled 1: enabled
update_file	File name of update image. Path to sub-folder is allowed, use "/" (slash) for separation
update_pswd	Password to be compared with password in update image
update_loadaddr	RAM address for update image. The update image is loaded to this RAM location before the update process is started. This variable is optional. If not defined, the load address defaults to 0x100000.

EEPROM Entries

For each of the three update methods there is an "update request" flag in the STW area of the EEPROM (system EEPROM). With this flags the update mechanism can be controlled. The meaning of the flags depends on the update method.

A Linux user space tool is provided to access the EEPROM data, see next chapter.

EEPROM Entry	Description
udrequsb	Enable update via USB flash disk. If set, the UBoot tries to load an update image from USB flash disk (if connected) and updates the device. This is done independently from the value of UBoot environment variable <code>update_en_usb</code> . The flag is not reseted by the UBoot (neither in case an update was found nor in case none was found).
udreqdf	Enable update via internal dataflash (NAND flash). If set and UBoot environment variable <code>update_en_df=1</code> , the UBoot tries to load a update image from the dataflash and updates the device. The flag is reseted after the update has been processed (independent from its result).
udreqtftp	This flag is currently not used. Changing its value has no effect.

When the update process is finished its status and result is stored in the EEPROM.

Update Status Entry	Description
Method	Update method of last update: <ul style="list-style-type: none"> • none: no update • USB: update via USB flash disk • DF: update via internal dataflash (NAND flash) • TFTP: update via tftp download
Number of Images	Number of data images that have been processed during last update
Global Status	Result status of last update: <ul style="list-style-type: none"> • 1: update processed successfully • !=1: an error occurred
	For each data image that has been processed the following information is stored (X is the image number from 1 to "Number of Images"):
Image X Name	Name of image as defined during update image compilation
Image X Status	Update result status of this image <ul style="list-style-type: none"> • 100: image programmed to flash • 101: data in flash is up to date, no update required • 110: UBoot script executed successfully • 111: UBoot script execution failed • other: some further error codes

The update status data structure in the EEPROM is overwritten by the UBoot. Thus a previously stored update status will be lost. To recognize from Linux user space if a new update process has been executed, the update status structure should be cleared after reading.

Tool to access the EEPROM entries: stw_eep

Besides some other features this Linux user space tool can be used to read and write the EEPROM data to control the STW update mechanism in UBoot.

For an overview of the supported features see the built-in help text of the tool:

```
# stw_eep -h
```

Set one of the update request flags:

```
# stw_eep -w udrequsb 1
# stw_eep -w udreqdf 1
```

Read back one of the update request flags:

```
# stw_eep -r udrequsb
# stw_eep -r udreqdf
```

Read the update status:

```
# stw_eep -r udstat
```

Clear the update status:

```
# stw_eep -w udstat 0
```

Update Image File generation

An update image file can be generated under a Linux environment (e.g. Ubuntu) by using the tool "make_udimg" (provided by STW):

```
# ./make_udimg
```

The content of the update image file is defined in the configuration file "udimage.cfg". Here is a list of the configuration options:

Config Option	Requirement	Description
result_file	mandatory	File name of the resulting update image
image_description	optional	Some text to describe the image, not used during the update
password	mandatory	Password to check against password in UBoot environment variables
imgX_name	mandatory	Name of the data image (e.g. kernel_v1.23r4) Caution: max. length is 15 characters
imgX_description	optional	Some text to describe the image data, not used during the update
imgX_file	mandatory	Path / Filename to image source data (e.g. to kernel image)
imgX_type	optional	Type of image Set to "script" for UBoot script image. For a data image set to "firmware". If not specified, type "firmware" is used as default => entry imgX_type can be omitted for data images in the configuration file.
imgX_addr	mandatory / optional	Flash address for image data; not required for UBoot script image
imgX_size	optional	Size of flash area to be erased (in bytes) If not specified, only as many flash sectors as required to store the image are erased (default behavior). In some cases this is not sufficient. E.g. for the rootfs (UBIFS format) it is necessary to erase not only the space required for the image but to erase the whole rootfs flash area. This is because the rootfs image auto-expands to the complete rootfs flash area (whole mtd block). This fails if the flash sectors are not blank. Specifying this parameter makes sure that a defined flash area is erased, independent from the image size.

imgX_ must be replaced by the image number, starting at 1. Up to 10 images can be compiled into a update image file. The UBoot updater can process all images but only the result for the first 5 images is stored in the update status structure in the EEPROM.

The source file for an UBoot script can be a plain text file containing the UBoot commands or an UBoot script image file (see UBoot documentation for details, e.g. here: <http://git.denx.de/?p=u-boot.git;a=summary>).

A configuration file to update the kernel and device tree blob and execute an UBoot script (e.g. to modify UBoot

environment variables) afterwards could look like this (source data files are expected to be stored in folder <make_udimg folder>/data):

```
# -----
#   file           udimage.cfg
#   brief          configuration file for the STW uboot updater (for TC3/EB07)
#
#   This file defines the components (kernel image, rootfs image...) that will
#   be assembled into the update image.
#
#   The syntax of the configuration options must match shell script variable
#   definition rules
#
#   created        17.01.2013   STW/G.Waibel (georg.waibel@sensor-technik.de)
# -----

# Result file name (FIT image name)
result_file="image.fit"

# Update image header definitions
image_description="STW update image file"
password="skywalker"

# First image: data image
img1_name="kernel"
img1_description="This is the kernel image"
img1_file="./data/eb07_vmlinux.img"
img1_addr="0xFF700000"

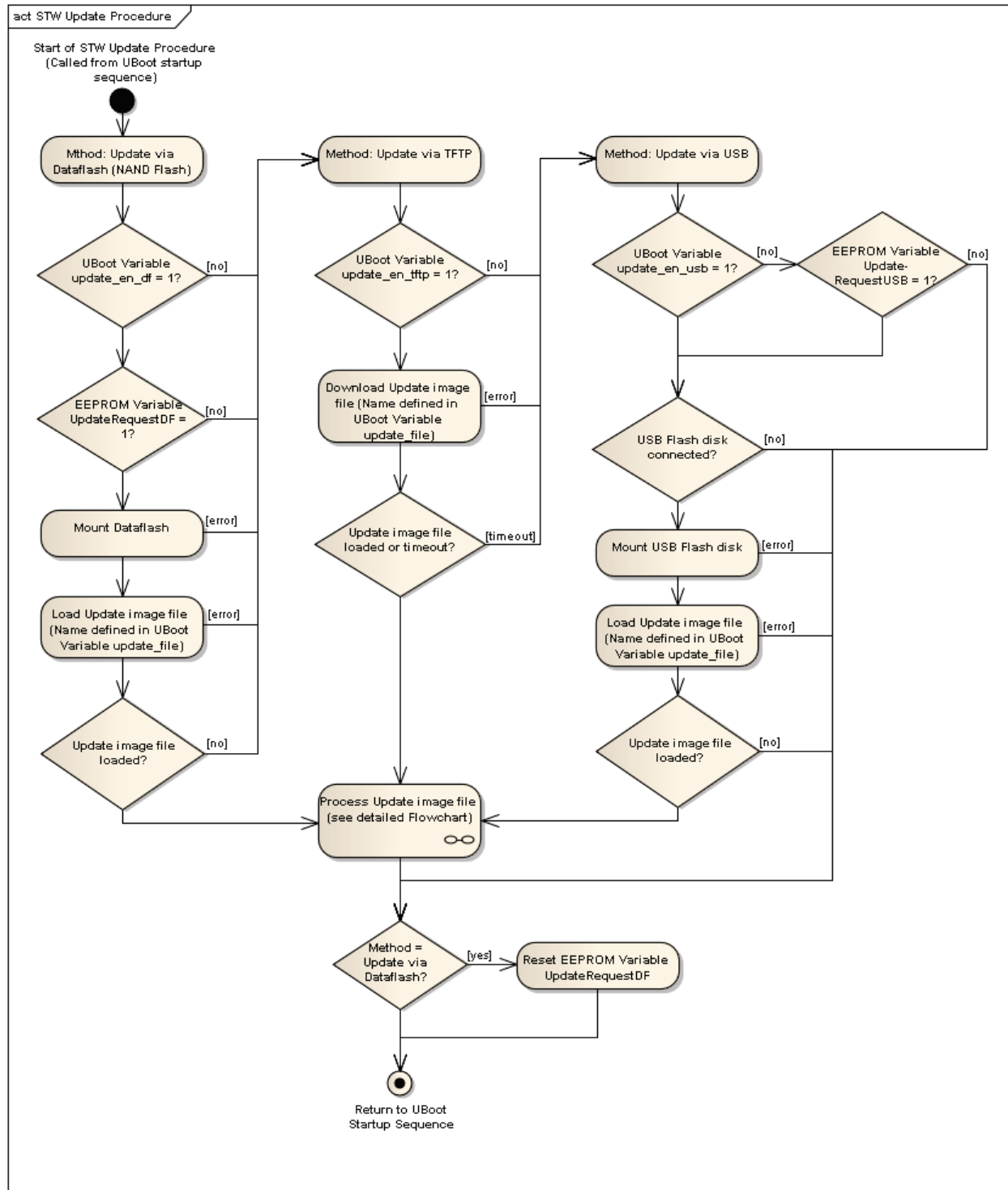
# Second image: data image
img2_name="dtb"
img2_description="This is the device tree blob image"
img2_file="./data/eb07.dtb"
img2_addr="0xFFFC0000"

# Third image: data image
img3_name="rootfs"
img3_description="This is the rootfs image"
img3_file="./data/rootfs.ubi"
img3_addr="0xFC000000"
# for eb07:
img3_size="0x2700000"
# for tc3:
# img3_size="0x3700000"

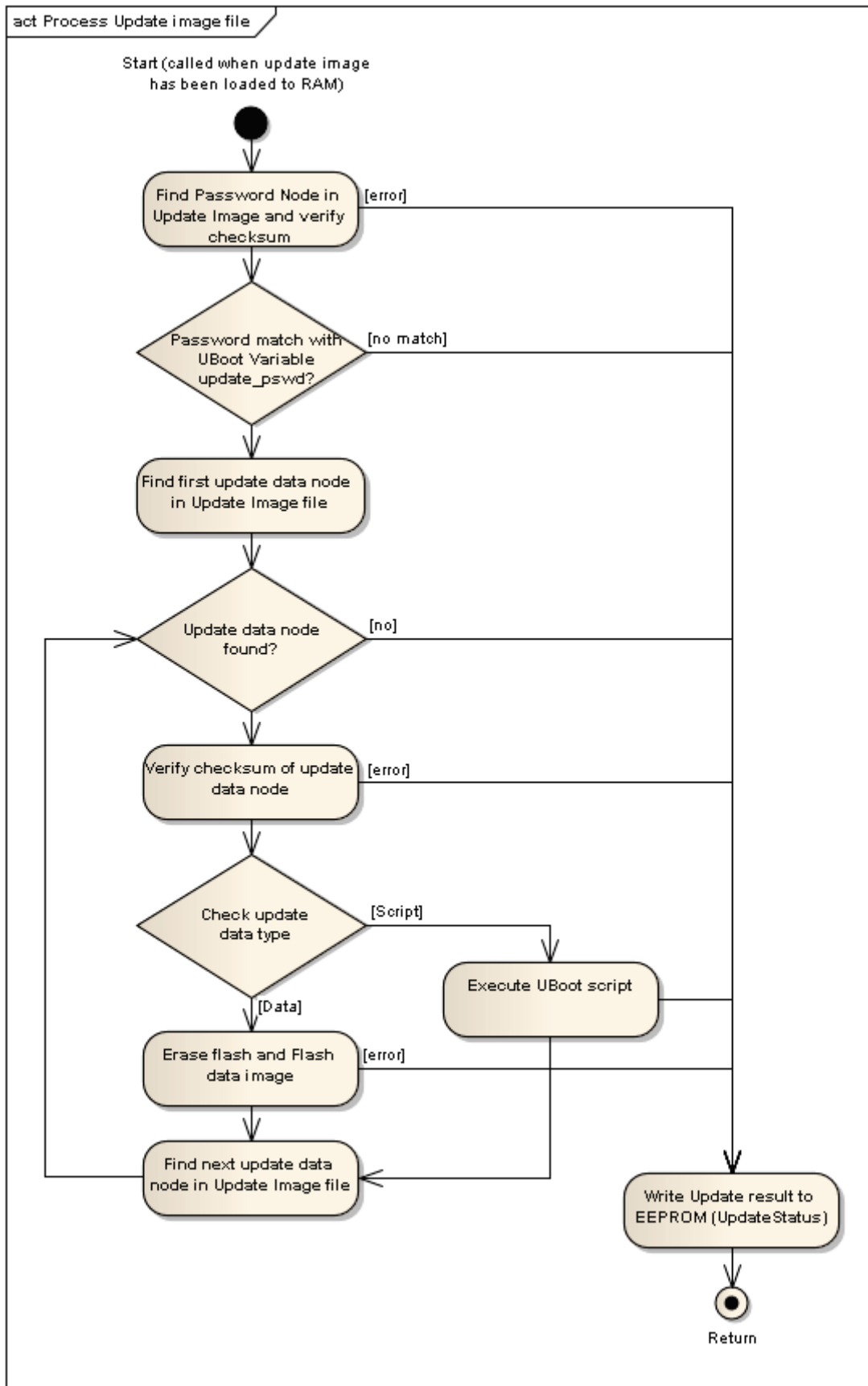
# Forth image: uboot script
img4_name="script"
img4_description="This is a uboot script file"
img4_file="./data/uboot.script"
img4_type="script"
```


9.4.2 STW Update Procedure Flow Charts

During startup of the system, the U-Boot bootloader executes the STW Update Process:



When a Update image file has been loaded to RAM, the update procedure is executed to process the data:



10 Application Notes

10.1 Communication Interfaces

10.1.1 Setting up the Serial Interface

Terminal program for the serial interface

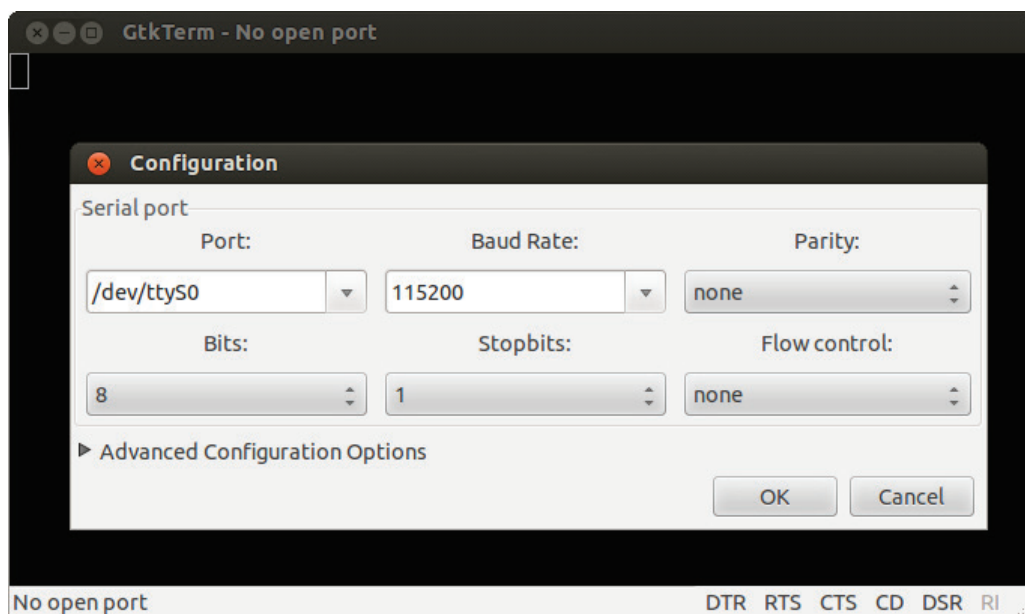
Use a serial straight-through cable to connect the TC3G with your PC.

Depending on the used operating system on the used PC for development, a RS232 terminal must be installed.

Recommended programs for a RS232 terminal:

- GtTerm for computers using a Linux operating system
- Tera Term for Windows desktop PCs

GtTerm used for RS232 terminal program:



Settings for the serial port

Property	Value/Description
Port	state here your port
Baud rate	115200
Data	8bit
Parity	none
Stop	1 bit

Property	Value/Description
Flow control	none

10.1.2 TFTP

- Install "tftpd" and related packages on your host computer

Open a terminal and type: (root password required)

```
$ sudo apt-get install xinetd tftpd tftp
```

- Create /etc/xinetd.d/tftp and add the following:

```
service tftp
{
    protocol          = udp
    port              = 69
    socket_type       = dgram
    wait              = yes
    user               = root
    server             = /usr/sbin/in.tftpd
    server_args       = /tftpboot
    disable            = no
}
```

- Make /tftpboot directory

```
$ sudo mkdir /tftpboot
$ sudo chmod -R 777 /tftpboot
$ sudo chown -R root /tftpboot
```

- Start tftpd through xinetd

```
$ sudo /etc/init.d/xinetd restart
```

10.1.3 NFS

The root-file system of the TC3G as well as data transfer of your own application program, will be performed by using NFS.

Host Computer

Install NFS:

```
$ sudo apt-get install portmap nfs-kernel-server
```

Enter Shares:

Open the file "/etc/exports" with the following command:

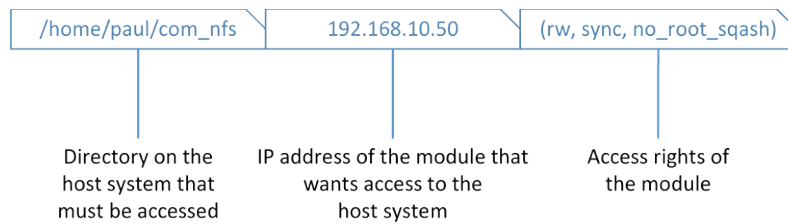
```
$ sudo gedit /etc/exports
```

And add the shares:

Example:

```
/home/<user>/com_nfs 192.168.10.50(rw,sync,no_root_squash)
/home/paul/projects 192.168.10.50(rw,sync,no_root_squash)
```

This example shares two directories with one client with a fixed IP -address. The “rw” indicates that read and write access is allowed.



After setting up `/etc/exports`, export the shares: (this is necessary every time the exports file is edited)

```
$ sudo exportfs -ra
```

Restart Services:

The NFS Kernel Server requires a restart:

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

On the device module:

Mount folder on TC3G:

```
$ mount.nfs 192.168.10.101:/home/<user>/com_nfs /mnt/com_nfs/
```

Now the folder with all files and subdirectories of the host PC ("`/home/<user>/com_nfs`") is mounted to "`/mnt/com_nfs/`" on the TC3G device.

192.168.10.101 is the IP address of the host computer.

Copy files onto TC3G

- Put the file that shall be copied to the TC3G (for example "test.txt") , to the folder: `/home/<user>/com_nfs`
- Type in the command:

```
$ cd /mnt/com_nfs
$ ls
test.txt
```

Now you can see the file (for example "test.txt") on your TC3G.

10.1.4 Telnet

Telnet is another way of communicating with your TC3G. However, Telnet uses the Ethernet instead of RS232.

Once the TC3G has been setup and is running, you don't need the RS232 terminal anymore to just log in to the TC3G or to apply changes. You can simply use TELNET (Telnet is supported by Linux, Macintosh, even Microsoft)

In order to login to the TC3G, start a terminal on any computer (Macintosh, Linux, MS), which is connected via Ethernet to the same network as the TC3G.

Switch on the TC3G and wait for it approximately 10 to 30 seconds to fully boot up.

Type:

```
telnet <tc3g - IP address>
```

Example:

```
telnet 192.168.10.3
```

You should then see the login screen of the TC3G.

```
altthaler@alt-linux: ~
# telnet ????.????.????.??? <TC3 IP-Address>

Entering character mode
Escape character is '^]'.

+++++++ Welcome to ++++++
TC3-?? <- S/N -> (Var tc3_???)
Have a lot of fun...
+++++++
TC3-?? <- S/N -> login: root
#
```

To log in, type: root

```
altthaler@alt-linux: ~
# telnet ????.????.????.??? <TC3 IP-Address>

Entering character mode
Escape character is '^]'.

+++++++ Welcome to ++++++
TC3-?? <- S/N -> (Var tc3_???)
Have a lot of fun...
+++++++
TC3-?? <- S/N -> login: root
#
# exit
Connection closed by foreign host
#
```

10.1.5 I/O Pin's

The TC3G comes with 1 digital output and 1 digital input.

Digital Output:

This output is designed to provide power to an attached device, e.g. an external GPS receiver with up to 400 mA at DC 12 V. It is equipped with an internal 10 kOhm pull down resistor. To prevent damage to the TC3G, an external fuse must be applied to make sure the current does not exceed 400 mA.

The output can be switched ON / OFF over the driver interface with the following command:

```
echo 1 > /proc/stw_io/PIN1//To turn ON the output use command:
echo 0 > /proc/stw_io/PIN1//To turn OFF the output use command:
cat /proc/stw_io/PIN1 //To read back this output:
```

Digital Input:

This digital input has a built in 10kOhm pull down resistor and a threshold Voltage of ~3.5V. Signals below this Voltage will result in "0", signals above this Voltage will result in "1".

This input can be queried through the driver interface with the command:

```
/proc/stw_io/Pin2 //e.g.: "cat /proc/stw_io/Pin2"
```

10.1.6 GSM

GSM settings:

The GSM settings are located in:

```
/etc/rc.d/rc.conf
```

```
# GSM Settings
export GSM_BAUDRATE="460800"
export GSM_PIN="1234"
export GSM_SCA="0012063140005"
export GSM_APN="epc.tmobile.com"
```

GSM Manual control:

The GSM service of the TC3G built in GSM/GPRS Modem can be manually controlled via the following scripts:

- ppp-start
- ppp-stop

These scripts reside in:

```
/etc/ppp/
```

10.1.6.1 GSM SIM Card



NOTE:

To be able to use the TC3G in a cellphone network, a SIM card is necessary.

Recommended is a SIM card with unlimited data plan.

Use a SIM card from a cellphone network provider of your choice, for example T-Mobile or AT&T.



How to insert the SIM card:

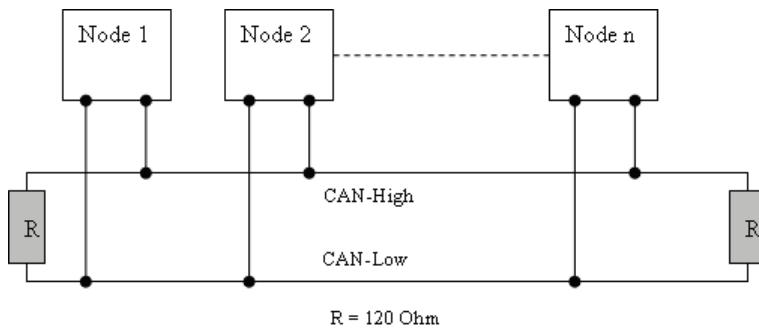
1. Disconnect your TC3G.
2. Pull out the rubber gasket from the SIM card slot.
3. Do not use any sharp instruments that can damage the green yellow button. Use a pointed pin like a pen to push the green-yellow button next to the tray to unlock the SIM card tray.
4. Take out the SIM card tray and insert your SIM card.
5. Inserting the SIM card tray with the inserted SIM card into the slot. Make sure to insert the SIM card tray running rails.
6. Insert the rubber gasket. Make sure the rubber gasket fits properly.

10.1.7 CAN

CAN-Termination

According to the CAN specification, every end of a CAN bus trunk-line must be terminated with a 120 Ohm Resistor.

Example of a terminated CAN bus:



CAN-Bus length (Trunk-line length)

The CAN bus length (Trunk-line length) is limited and depends on the Baud-rate, the CAN bus is operated with. The following table shall give an overview of maximum CAN bus length according to the used Baud rate.

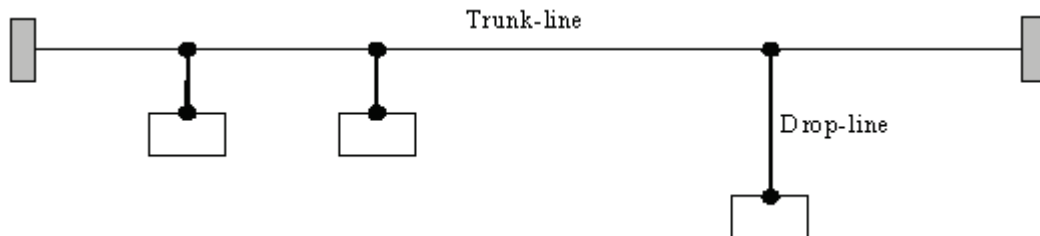
Baud-Rate [kbit/s]	CAN-bus length
1000 (1MBit/s)	< 20m
500	< 100
250	< 250
125	< 500
50	< 1000
20	< 2500
10	< 5000

Drop-line length

Drop-lines are allowed, but should be avoided, as they typically create signal echo. Drop-lines must not be terminated with 120 Ohm resistors! The following table shows the maximum lengths of drop-lines to be used:

Baud-Rate [kBit/s]	Drop-line length [m]	Total length of all drop-lines [m]
1000 (1Mb/s)	< 1	< 5
500	< 5	< 25
250	< 10	< 50

Baud-Rate [kBit/s]	Drop-line length [m]	Total length of all drop-lines [m]
125	< 20	< 100
50	< 50	< 250



10.1.7.1 CAN for ISOBUS

To run an ISOBUS application, the used CAN bus interface must be specified according to the ISOBUS standard 11783-2.

How to configure a CAN bus interface for an ISOBUS application

Set the used CAN interface (can0 or can1) to the following specifications:

1. Stop the used CAN bus
2. Set the bitrate to 250000
3. Set the sample-point to 0.80
4. Restart the used CAN bus.

Example: Change bitrate of CAN 0 for ISOBUS application

```
# ifconfig can0 down
# /sbin/ip link set can0 up type can bitrate 250000 sample-point 0.80
# ifconfig can0 up
```

10.1.8 Bluetooth M2M

The file describes how to easily create a connection between a master TC3G module to a slave TC3G module.

Precondition: Both modules must have a up and discoverable Bluetooth device.

How to check if the Bluetooth interface is up:

After booting up the devices, check if the Bluetooth interface is up with the hciconfig command:

```
# hciconfig
hci0:   Type: BR/EDR   Bus: USB
        BD Address: 00:07:80:43:5A:EE   ACL MTU: 310:10   SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
indicates that the Bluetooth interface is
discoverd.
        RX bytes:1339895 acl:7679 sco:0 events:1891 errors:0
        TX bytes:31463 acl:1568 sco:0 commands:152 errors:0
#
```

On success:

The line 'UP RUNNING PSCAN ISCAN' indicates that the interface is up and discoverable.

On failure:

Open and initialize the Bluetooth interface:

```
# hciconfig hci0 up
```

Enable page and inquiry scan of the Bluetooth interface:

```
# hciconfig hci0 piscan
```

Serial Connection via Bluetooth

Scan for other Bluetooth devices to retrieve the bdaddr for the needed device, execute the command hcitool with the parameter scan:

```
# hcitool scan
Scanning ...
        00:07:80:57:BF:A6      TC3G-141231231003   //This is the module the connection
shall be established with
        00:15:83:41:79:8D      Notebook_1
        CC:52:AF:05:BF:0D      Smartphone_xyz
#
```

In this example the the module TC3G-141231231003 was found and is used to establish a connection with.

The bdaddr of the TC3G-141231231003 is 00:07:80:57:BF:A6. This bdaddr is used in this example, to establish a connection via Bluetooth.

Use the rfcomm command to establish a serial port connection via Bluetooth. The first parameter is "bind". With the parameter bind, the rfcomm command binds the chosen RFCOMM device to a remote Bluetooth device. The command does not establish a connection to the remote device, it only creates the binding. The second parameter is the RFCOMM device parameter. /dev/rfcomm0 is already occupied for incoming serial connections and /dev/rfcomm1 will be used for obexftp.

In this case use the rfcomm2 command. The last parameter is the Bluetooth address bdaddr of the device that must be connected:

```
# rfcomm bind rfcomm2 00:07:80:57:BF:A6
```

Now the device rfcomm2 can be seen in the /dev directory:

```
# ls -al /dev | grep rfcomm2
crw-rw----  1 root    root      216,   2 Jan 12 02:28 rfcomm2
```

Open the connection via picocom:

```
# picocom -b 115200 /dev/rfcomm2
```

In case of success, after some seconds you can see the terminal prompt of the connected device:

```
picocom v1.6
port is      : /dev/rfcomm2
flowcontrol  : none
baudrate is  : 115200
parity is    : none
databits are : 8
escape is    : C-a
local echo is : no
noinit is    : no
noreset is   : no
nolock is    : no
send_cmd is  : sz -vv
receive_cmd is : rz -vv
imap is      :
omap is      :
emap is      : crcrlf,delbs,

picocom -b 115200 /dev/rfcomm2Terminal ready

+++++++      Welcome to      ++++++
TC3G-141231231003 (Var tc3_A_J)
Have a lot of fun...
+++++++
TC3G-141231231003 login:
```

Serial Connection via Bluetooth using a proprietary protocol:

For a productive system disable the terminal prompt that is forwarded from the rfcomm device.

Edit the /etc/init.d/scripts/bt start script:

```
# vi /etc/init.d/scripts/bt
```

Search the following line:

```
rfcomm watch rfcomm0 1 &>/dev/null getty 460800 /dev/rfcomm0 &
```

and remove the getty command:

```
rfcomm watch rfcomm0 1 &>/dev/null &
```

Change the file on both devices. After that, restart and continue with:

```
# hcitool scan
Scanning ...
    00:07:80:57:BF:A6      TC3G-141231231003
    00:15:83:41:79:8D      Notebook_1
    CC:52:AF:05:BF:0D      Smartphone_xyz
# rfcomm bind rfcomm2 00:07:80:57:BF:A6
# picocom -b 115200 /dev/rfcomm2
```

On the "slave" TC3G open also the /dev/rfcomm0 via picocom.

```
# picocom -b 115200 /dev/rfcomm0
```

Result: Now characters can be sent or received.

It is also possible to connect more than only two devices. For every additional device that must be added to the Bluetooth network, start one more time the rfcomm device via the rfcomm command and add it to the /etc/init.d/scripts/bt file

File Transfer via Bluetooth between two TC3G

A frequent application case is to transfer files from one TC3G to another over Bluetooth.

Scan for the needed Bluetooth device:

```
# hcitool scan
Scanning ...
    00:07:80:57:BF:A6      TC3G-141231231003
    00:15:83:41:79:8D      Notebook_1
    CC:52:AF:05:BF:0D      Smartphone_xyz
#
```

In this example the the module TC3G-141231231003 was found and is used to send or receive files via ftp.

Check, if it is possible to send or receive files via ftp. Use the command sdptool to perform the check.

The command sdptool provides the interface for performing "Service Discovery Protocol" queries on Bluetooth devices.

To search for the needed device that support the file transfer protocol, execute the sdptool command and search for the Bluetooth address of the TC3G-141231231003:

```
# sdptool search ftp
Inquiring ...
----- output of the inquiry -----
Searching for ftp on 00:07:80:57:BF:A6 ...
Service Name: OBEX File Transfer
Service RecHandle: 0x10006
Service Class ID List:
  "OBEX File Transfer" (0x1106)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 10
  "OBEX" (0x0008)
Profile Descriptor List:
  "OBEX File Transfer" (0x1106)
    Version: 0x0100
----- output of the inquiry -----
#
```

In the output of the inquiry the channel number 10 is received. This channel number is used for sending or receiving files between the devices.

For sending a file use the obexftp command. The first parameter specifies the Bluetooth address of the remote device and the second parameter specifies the Bluetooth channel. The third parameter is the path to the file that must be sent:

```
# obexftp -b 00:07:80:57:BF:A6 -B 10 -p /etc/apns-conf.xml
Connecting...done
Tried to connect for 825ms
Sending "/etc/apns-conf.xml"...|done
Disconnecting...done
#
```

For receiving a file use also the obexftp command. The only difference is the last parameter.

Use the list parameter of the obexftp command, to get the available files:

```
# obexftp -b 00:07:80:57:BF:A6 -B 10 -l
Connecting...done
Tried to connect for 562ms
Receiving "(null)"...\<?xml version="1.0"?>
<!DOCTYPE folder-listing SYSTEM "obex-folder-listing.dtd">
<folder-listing version="1.0">
<file name="apns-conf.xml" size="250493" user-perm="RWD" modified="19700105T033647Z"
created="19700105T033647Z" accessed="19700105T033720Z" />
<file name="apn.setting" size="96" user-perm="RWD" modified="19700105T032939Z"
created="19700105T032939Z" accessed="19700105T032939Z" />
<file name="sdruntime" size="2" user-perm="RWD" modified="19700105T032926Z"
created="19700105T032926Z" accessed="19700105T032926Z" />
<file name="wakeup" size="4" user-perm="RWD" modified="19700105T032926Z"
created="19700105T032926Z" accessed="19700105T032939Z" />
<file name="interfaces" size="510" user-perm="RWD" modified="19700105T032925Z"
created="19700105T032925Z" accessed="19700105T032925Z" />
<file name="issue" size="164" user-perm="RWD" modified="19700105T032925Z"
created="19700105T032925Z" accessed="19700105T032940Z" />
<file name="hostname" size="17" user-perm="RWD" modified="19700105T032925Z"
created="19700105T032925Z" accessed="19700105T032925Z" />
<folder name="dbus" size="60" user-perm="RWD" modified="20130422T053845Z"
created="19700105T032922Z" accessed="20130422T053845Z" />
</folder-listing>
done
Disconnecting...done
#
```

To download, for example, the hostname file use the get parameter of the obexftp command in the following way:

```
# obexftp -b 00:07:80:57:BF:A6 -B 10 -g hostname
Connecting...done
Tried to connect for 569ms
Receiving "hostname"... done
Disconnecting...done
#
```

10.1.9 E-Mail

The TC3G is equipped with an e-mail client. The name of the client is 'mailx'. With that client it is possible to send an e-mail from the TC3G.

The account settings have to be added to a configuration file.

/root/.mailrc - file

The configuration of the e-mail account is handled in the file .mailrc in the /root directory.

The following settings are for the google mail account: example@gmail.com with the password: 123456.

```
account gmail_account {
    set folder=imaps://example@imap.gmail.com/INBOX
    set password-example@imap.gmail.com="123456"
    set imap-use-starttls
    set from="TC3G <example@gmail.com>"
    set replyto="example@gmail.com"
    set sender="example@gmail.com"
    set smtp-use-starttls
    set ssl-verify=ignore
    set smtp="smtp://smtp.gmail.com:587"
    set smtp-auth="login"
    set smtp-auth-user=example@gmail.com
```

```
set smtp-auth-password="123456"
}

# address book
alias developer1 developer.number1@test.de
```

Sending an E-Mail

After the account is configured, it is possible to send an e-mail considering the following syntax:

mailx [-s subject] [-a attachment] [-c cc-addr] [-b bcc-addr] [-A account] to-addr . . .

Example

```
echo "Please find the apn-settings in the attachment." | mailx -A gmail_account -s "apn-
settings of the TC3G" -a /tmp/apn.setting developer1
```

11 Utilities Tools

Description

This chapter provides an overview, including a short description, about the utilities of STW available on the TC3G.

11.1 stw_dptool

NAME

```
stw_dptool - tool shows the content of the datapool
```

SYNOPSIS

```
stw_dptool [-h] [-v] [-m] [-a] [-x] [-z]
           [-d <dp_name> <dp_list> <dp_var>]
           [-s <dp_name> <dp_list> <dp_var> <value>]
```

DESCRIPTION

This tool can show the content of the datapool. It allows the user to print all datapools or only a subset of lists or variables of a specific datapool.

OPTIONS

```
-v    Version number of the tool.

-h    Usage and options (help).

-a <datapool_path>
    Print a list of all datapools found under <datapool_path>.
    Calling without a path will use default directory: /var/run/taf/Datapools.

-p <dp_name> <dp_list> <dp_var>
    Print only a specific datapool, list or variable.

-s <dp_name> <dp_list> <dp_var> <value> <datapool_path>
    Set specific <value> to <dp_var>, <datapool_path> is optional.

-x    Print a list of all datapools with detailed information.

-z    Print a list of all datapools with more dedetailed information.
```

EXAMPLES

```
Print all datapools found under /var/run/taf/Datapools:
    stw_dptool -a
Print all datapools found under /mnt/dataflash/Datapools:
    stw_dptool -a /mnt/dataflash/Datapools
Print only the datapool 'gpsDPL':
    stw_dptool -p gpsDPL
Print only the list 'gpsList' of the datapool 'gpsDPL':
    stw_dptool -p gpsDPL gpsList
Print only the variable 'gps_lat' of the list 'gpsList' of the datapool 'gpsDPL':
    stw_dptool -p gpsDPL gpsList gps_lat
Set the variable 'gps_qual' of the list 'gpsList' of the datapool 'gpsDPL' to '4'
(using the default datapool path: /var/run/taf/Datapools):
    stw_dptool -s gpsDPL gpsList gps_qual 4
```


11.2 stw_GetGPS

NAME

```
stw_GetGPS - cmd line tool to fetch GPS data
```

SYNOPSIS

```
stw_GetGPS [-hvmja]
```

DESCRIPTION

This application asks the GPS data structure via dbus from the ygpsd.

Structure T_DBUS_GPS_Data



NOTE:

This structure is only valid in combination with ygpsd version 4.00r0 or newer

```
typedef struct
{
    charn acn_status[32];          // NONE = fix not available,
                                   // GPS = GPS fix,
                                   // DGPS = Differential GPS fix,
                                   // PPS = PPS fix,
                                   // RTK = Real Time Kinematic,
                                   // FRTK = Float RTK,
                                   // NMEA = acn_quatily > 5
    charn acn_latitude[32];        // Degrees notation: DD.dddddd
    charn acn_longitude[32];       // Degrees notation: DD.dddddd
    charn acn_altitude[32];        // Above/below mean-sea-level in meters
    charn acn_time[32];            // hhmmss
    charn acn_satellites[32];      // number of satellites used in solution
    charn acn_quality[32];         // 0 = fix not available,
                                   // 1 = GPS fix,
                                   // 2 = Differential GPS fix
                                   // 3 = PPS fix
                                   // 4 = Real Time Kinematic
                                   // 5 = Float RTK
                                   // 6 = estimated (dead reckoning)
                                   // 7 = Manual input mode
                                   // 8 = Simulation mode
    charn acn_warn[32];            // V = GPS position is not valid
                                   // A = GPS position is valid
    charn acn_speed[32];           // Speed over ground km/h XXX.YYYYYY
    charn acn_course[32];          // Degrees (0..360) XXX.YYYYYY
    charn acn_date[32];            // ddmmyy
    charn acn.UTC[64];             // MMDDhhmmYYYY
    charn acn_HDOP[32];            // Horizontal dilution of precision
    charn acn_AgeOfDGPS[32];       // Age of differential GPS data in seconds
} T_DBUS_GPS_Data;
```

Therefore, the ygpsd needs to be up and running.

The application parses the information from the NMEA0183 strings GGA and RMC.

For further information, see the TC3G User Manual.

OPTIONS

```
-v, --version
    Version number of the daemon.
```

```
-h, --help
    Usage and options (help).

-a, --array
    print a single line array of the values

-j, --json
    print the data in JSON format

<NONE>
    print the dbus_gps_data structure
```

EXAMPLES

```
stw_GetGPS
Status: NMEA
Latitude: 47.862628
Longitude: 10.631297
Altitude: 696
Time: 114713
Satellites: 3
Quality: 1
Warn: A
Speed: 0.00000
Course: 184.39000
Date: 210716
UTC String: 072111472016

stw_GetGPS -a
NMEA 47.862653 10.631265 696 114854 3 1 A 0.03600 41.50000 210716 072111482016

stw_GetGPS -j
{'status':'NMEA','lat':'47.862653','long':'10.631233','alt':'692',
 'time':'114934','sat':'3','quality':'1','warn':'A','speed':'0.01800',
 'course':'135.99000','date':'210716','utc':'072111492016'}
```

11.3 stw_ReadACC

NAME

```
stw_ReadACC - Reads the acceleration values of the internal motion sensor
```

SYNOPSIS

```
stw_ReadACC [-h] [-v] [-m] [-d] [-q <NUMBER>]
```

DESCRIPTION

This application reads the acceleration values of the internal motion sensor of the device.

It is necessary for the tool to work, that the FIFO mode is enabled.

The motion sensor could be adapted via its config file 'control'.

```
'cat /sys/bus/i2c/devices/0-0018/control'
power=on                [on/off]
powermode=normal        [normal/low_power]
fifomode=stream         [off/fifo/stream/bypass]
datarate=100Hz          [1Hz/10Hz/25Hz/50Hz/100Hz/200Hz/400Hz/1.25kHz/1.6kHz/5kHz]
scale=2g                [2g/4g/8g/16g]
setting=default         [wakeUp/default]
wakeUp_threshold=05     (default=10 [05 .. 101])
```

In order to adapt a parameter of the config file, it needs be overwritten.

Example: set 'scale' to '8g'

```
'echo "scale=8g" > /sys/bus/i2c/devices/0-0018/control'
```

For further information, see the User Manual of the device.

OPTIONS

```
-v, --version
    Version number of the daemon.

-h, --help
    Usage and options (help).

-d, --debug
    Print debug messages and the current motion sensor configuration.

-q <NUMBER>, --quantity <NUMBER>
    Print specific <NUMBER> of acceleration values.

<NONE>
    Print endless acceleration values.
```

EXAMPLES

```
stw_ReadACC -q 10
    8;   -312;   1008
    8;   -296;   1016
    8;   -312;   1012
    4;   -312;   1012
    4;   -304;   1016
    4;   -308;   1008
    12;  -316;   1008
    4;   -300;   1008
    8;   -292;   1000
    24;  -320;   1012

stw_ReadACC -q 10 -d
Config file '/sys/bus/i2c/devices/0-0018/control':
power=on                [on/off]
powermode=normal         [normal/low_power]
fifomode=stream          [off/fifo/stream/bypass]
datarate=100Hz           [1Hz/10Hz/25Hz/50Hz/100Hz/200Hz/400Hz/1.25kHz/1.6kHz/5kHz]
scale=2g                 [2g/4g/8g/16g]
setting=default          [wakeup/default]
wakeup_threshold=05      (default=10 [05 .. 101])

Sample quantity:        10
Sleeptime (us):         16666

-- X --|-- Y --|-- Z --
-4;   -312;   1004
0;    -308;   1008
4;    -304;   1004
4;    -304;   1004
8;    -312;   1008
4;    -320;   1012
4;    -304;   1004
8;    -292;   1028
4;    -316;   1032
-4;   -288;   1040
```

11.4 stw_acc2can

NAME

```
acc2can - acceleration sensor to can
```

SYNOPSIS

```
acc2can [-h] [-v] [-m] [-d] [-c] <CAN_BUS> --msg_id <msg id> [--msg_int] [--sample_rate] [--agg_type] --angle <path-to-config-file>
```

DESCRIPTION

The application acc2can is part of the teleservice application framework utilities. Its purpose is to send the acceleration values of the internal sensor on CAN bus.

For further information, see the "User Manual" of the device.

OPTIONS

```
-v, --version
    Version number of the application.

-h, --help
    Usage and options (help).

-m, --man
    Print the manpage.

-d, --debug
    Print debug messages.

-b, --big
    Set byte order to big endian
    Default: little endian

-c, --can
    CAN bus to use
    Default value:    can0
    Allowed values:  can0
                    can1
                    vcan0
                    vcan1..X

--msg_id  Message ID in decimal or hex to use for sending the ACC values
          Byte order is per default little endian
          ACC_X --> BYTE 0,1
          ACC_Y --> BYTE 2,3
          ACC_Z --> BYTE 4,5
          Allowed values:  <= 0x1FFFFFFF

--msg_int  Interval of CAN Message to send in msec
          Default value:    1000
          Allowed values:  >= 10

--sample_rate  ACC sample rate in msec
              Shall be greater or equal <msg_int>
              Default value:    500
              Allowed value:    >=10 and <= 1000

--agg_type  Aggregation to use
          If aggregation is selected, only the maximum, absolute value is sent via
          CAN. The aggregation time is defined via the CAN message send interval.

The sample rate
          defines the update rate of the acceleration values within the chosen
          aggregation time.
```

```

Default value:    0: No Aggregation
Allowed values:   0: No Aggregation
                  1: MAX Aggregation

```

-a, --angle Calculate zero terminated tilt angle between the main xyz-axis and driving plane.

The result angle will be given in degrees multiplied with 10.
The result will be written to the end of the can message (BYTE 6,7).
A path to a configuration file must be given with this parameter.
The configuration file can have following parameters:

```

PLANE_REFERENCE_X <value>
PLANE_REFERENCE_Y <value>
PLANE_REFERENCE_Z <value>

```

If one or more of these parameters will not be given, default values (zeros) will be used.

Example of normal positioning:

```

PLANE_REFERENCE_X 0
PLANE_REFERENCE_Y 0
PLANE_REFERENCE_Z -1000

```

If the configuration file does not exist, the first calculated values will be the reference for

zero terminating, that means that the calculated angle will be zero terminated to the started position

EXAMPLES

```

stw_acc2can -c can0 --msg_id 0x100 --agg_type 1
    send message ID 0x100 every 1000ms. Acc sample time is 500 ms, aggregation is set to
max
stw_acc2can -c can0 --msg_id 0x100 --angle /etc/init.d/stw_acc2can.config
    send message ID 0x100 and pass the angle between the driving plane and xyz axis to
the can message end

```

11.5 stw_show_gps

Name

```
stw_show_gps
```

Description

The application stw_show_gps parses NMEA0183 strings from a port of the device.
The tool will work only, if the GPS daemon ygpsd is not running.

Synopsis

```

# stw_show_gps
Usage: show_gps <dev_file> [-lat] [-lon] [-alt] [-tim] [-sat] [-qua] [-war] [-spe] [-cou]
[-dat] [-sta] [-all] [-p]

```

Example

```

# stw_show_gps /proc/stw_gps/port -all
$GPGGA,133935.000,4751.7594,N,01037.8708,E,1,6,1.86,688.1,M,47.9,M,,*5C
GPGGA==GPGGA 5c==5c
0:$GPGGA
1:133935.000
2:4751.7594
3:N
4:01037.8708
5:E
6:1
7:6
8:1.86
9:688.1

```

```

10:M
11:47.9
12:M
13:
14:*5C

$GPRMC,133935.000,A,4751.7594,N,01037.8708,E,2.42,221.84,190716,,,A*6B
GPRMC==GPRMC 6b==6b
0:$GPRMC
1:133935.000
2:A
3:4751.7594
4:N
5:01037.8708
6:E
7:2.42
8:221.84
9:190716
10:
11:
12:A*6B

Latitude: 47.862657
Longitude: 10.631180
Altitude: 688
Time: 133935
Satellites: 6
Quality: 1
Warn: A
Speed: 2.420000
Couse: 221.840000
Date: 190716
NMEA: received

```

11.6 stw_flash_client

Description

This application supports the functionality of the STW Kefex Tool WinFlash.

Example

```

# stw_flash_client -h
Usage: flash_client <options> <ini_file> <ini_file_section>
  -h --help           Displays this usage information
  -w --wakeup         Triggers a wakeup on specified node
                      e.g. flash_client -w -i can0 WinFlash.ini CONFIG1
  -s --serial         Set wakeup mode to serial number of the ECU
                      Setup up serial number in config file e.g. WinFlash.ini
parameter SRN
  -v --version        Output version information
  -a --flash_fin_act action Set what to do after flashing has finished
                      With action in:
                      NODE_RETURN 1: start application
                      NODE_RESET  2: reset node
                      NODE_SLEEP  3: go back to sleep mode
                      NET_START   4: start all nodes
                      NET_RESET   5: reset all nodes
                      NONE        6: no action (DEFAULT)
  -c --can index      Set which CAN interface to use (default is 0)
                      DEPRECATED option, prefer -i instead
  -i <CAN interface> Specify CAN interface name (e.g. can0)
-----
Example (WinFlash.ini):

```

```
[CONFIG1]
BITRATE=125
STARTTIME=3
SENDID=81
XTDID=0
COMPANYID=Y1
LOCALID=13
FILENAME=flashTest.hex
PROGTYPE=5
SNR=121312112020
SENDRESETRQ=0
DEV_ID_CHECK=1
DEV_ID_CHECK_GET_ID_FAIL=2
DEV_ID_CHECK_MATCH_ID_FAIL=2

Start FLASH process by default local id: flash_client -a NONE -i can0    WinFlash.ini
CONFIG1
Start FLASH process by serial number    : flash_client -a NONE -i can0 -s WinFlash.ini
CONFIG1
-----
```

11.7 stw_SendSMS

NAME

```
stw_SendSMS - Sends a SMS via the TAF and the internal modem.
```

SYNOPSIS

```
stw_SendSMS <PhoneNo> <Text> [-h] [-v] [-m]
```

DESCRIPTION

This application allows you to send a text message right away. This is done via DBUS and the ysm sd.

For the application to work, ysm sd needs to be active and the device needs to be equipped with a SIM card.

For further information, see the User Manual of the device.

OPTIONS

```
-v    Version number of the daemon.
```

```
-h    Usage and options (help).
```

```
<PhoneNo> <Text>    send <Text> to <PhoneNo> in form of a SMS
```

EXAMPLES

```
stw_SendSMS "016012345678" "This could be your text"
    The text 'This could be your text' is send to '016012345678' in form of a SMS.
```

11.8 stw_RecvSMS

NAME

```
stw_RecvSMS - Reads the next SMS via the TAF from the internal modem.
```

SYNOPSIS

```
stw_RecvSMS <WaitTimeSec> [single] [-h] [-v] [-m]
```

DESCRIPTION

This application can read SMS from the internal modem via dbus and ysmsd. In order to use this application, the ysmsd needs to be active.

For further information, see the User Manual of the device.

OPTIONS

```
-v    Version number of the daemon.

-h    Usage and options (help).

<WaitTimeSec>    Receive SMS for the next <WaitTimeSec> seconds. Default: 30 s.

<WaitTimeSec> single    Stop application after the first SMS is received.
```

EXAMPLES

```
stw_RecvSMS single"
    Application will wait until the first SMS is received.
```

11.9 stw_CANsnapshot

NAME

```
stw_CANsnapshot - shell script for getting a snapshot of 'raw' CAN data
```

SYNOPSIS

```
stw_CANsnapshot [-vhmd] -f <file> -p <parameter> -t <timeout>
```

DESCRIPTION

stw_CANsnapshot is a shell script that allows it user to get a snapshot of raw values from any CAN interface of the device. Using stw_CANsnapshot options allow a very good configuration of the 'candump' binary, which is used for the generation of the CAN snapshot.

The script is mainly built for remote accessing the device, where 'ctrl-c' cannot be used to abort the binary.

OPTIONS

```
-f, --file <file_name>
    configure file name for the snapshot results
    default: 'snapshot.txt'

-p, --parameter <parameters>
    configure candump parameters, this string is handed over to the binary
    default: 'can0 -tA'
    use parameter 'help' for candump help

-t, --timeout <timeout>
    configure length of CAN snapshot in seconds
    range: 0 - 30 seconds
    default: 5 seconds

-d, --debug
    activate debug messages

-h, --help
    usage and options (this help)

-v, --version
```



```
version number of the daemon
```

EXAMPLES

```
# print CAN snapshot with default parameter (can0, 5 seconds)
./stw_CANsnapshot
(2016-12-19 09:53:53.279812)  can0  4E9   [3]  16 D9 ED
(2016-12-19 09:53:54.881546)  can0  1CA   [1]   67
(2016-12-19 09:53:55.081841)  can0  6B2   [7]  2C F8 1B 83 53 69 79
(2016-12-19 09:53:56.683675)  can0  75B   [6]  1D AD 89 66 74 87
(2016-12-19 09:53:57.684727)  can0  5BA   [8]  7D 94 56 CD 73 8E 50 7D
(2016-12-19 09:53:58.085107)  can0  3AA   [8]  05 A7 54 D6 29 31 4B 05

# print CAN snapshot of can1 with timeout of 1 seconds and debug messages active
./stw_CANsnapshot -d -t 1 -p 'can1 -tA'
***debug is acivated***
/usr/bin/candump can1 -tA > /mnt/dataflash/stwErrFiles/snapshot.txt &
cat /mnt/dataflash/stwErrFiles/snapshot.txt
(2016-12-19 09:53:05.427515)  can1  577   [6]  28 A9 A7 2E 5E 0F
(2016-12-19 09:53:05.627706)  can1  553   [6]  5D AC 2A D7 55 4D
(2016-12-19 09:53:05.827868)  can1  416   [4]  2F 5C 02 9E
(2016-12-19 09:53:06.028071)  can1  30C   [5]  11 9B 09 21 50
(2016-12-19 09:53:06.228374)  can1  5CE   [8]  42 80 6F 90 2B F9 89 6E

# print 'candump' help with all possible options and parameters
./stw_CANsnapshot -p help
```

11.10 stw_GetGSMinfo

NAME

```
stw_GetGSMinfo - shell script for getting GSM information
```

SYNOPSIS

```
stw_GetGSMinfo [-vhmdaj]
```

DESCRIPTION

stw_GetGSMinfo is a shell script that allows its user to get all GSM information, that the ygsmd provides.
Using stw_GetGSMinfo options will print the GSM information in the desired format.

OPTIONS

```
-h, --help
    usage and options

-v, --version
    version number of the daemon

-a, --array
```

```
print a single line array of the values
```

```
-j, --json
    print the data in JSON format

-d, --debug
    activate debug messages
```

EXAMPLES

```
# print GSM information
./stw_GetGSMInfo
PhoneNo: +123456789098
SIMSerial: 12345678909876543211
SIMState: 5
SIMCarrier: T-Mobile Internet
SIMMCC: 123
SIMMNC: 01
SIMAPN: internet.t-mobile
SIMUser: user
SIMPasswd: passwd
IMEI: 123456789098765
IMSI: 123456789098765
RegState: 1
SigQuality: Good
SigQualitydBm: -77
AccessTec:3G
MCC: 123
MNC: 01
LAC: 12345
CellID: 12345
LastUpdate: X0.0X.1X_12:18:52

# print GSM information in JSON format
./stw_GetGSMInfo -j
{
  "phoneno":"+123456789098",
  "simserial":"12345678909876543211",
  "simstate":"5",
  "simcarrier":"T-Mobile Internet",
  "simmcc":"123",
  "simmnc":"01",
  "simapn":"internet.t-mobile",
  "simuser":"user",
  "simpasswd":"passwd",
  "imei":"123456789098765",
  "imsi":"123456789098765",
  "regstate":"1",
  "sigquality":"Good",
  "accesstec":"3G",
  "mcc":"123",
  "mnc":"01",
  "lac":"12345",
  "cellid":"12345"
}
```

11.11 stw_RetentionRule

NAME

```
stw_RetentionRule - removing files after size limit is reached
```

SYNOPSIS

```
stw_RetentionRule [-vhmd] -f <folder> -l <limit> -t <loop_time>
```

DESCRIPTION

stw_RetentionRule is a script that checks a desired directory for its size, in case that size reaches a certain limit, the tool starts to delete files from a specified directory (the oldest file first).

OPTIONS

```
-h, --help
    usage and options (this help)

-v, --version
    version number

-d, --debug
    activate debug messages

-l, --limit [MB]
    when limit is reached, files will be deleted

-t, --time [sec]
    every <time> interval it is checked if the <limit> is reached

-f, --folder
    set directory that will be checked for size limit,
    same directory will be used to delete files in
```

USER DEFINED VARIABLES

```
Directory that will be checked for size limit
->/mnt/dataflash/blackbox
When limit x is reached, files will be deleted
->15 MB
Directory in which files will be deleted
->/mnt/dataflash/blackbox
Interval to check if limit is reached
->120 seconds
Printing debug messages when 'true'
->>false
```

11.12 stw_TrafficMonitor

NAME

```
stw_TrafficMonitor - shell script for getting network traffic for given interface
```

SYNOPSIS

```
stw_TrafficMonitor [-vhmdipo]
```

DESCRIPTION

stw_TrafficMonitor is a shell script that allows its user to monitor network traffic for a given device.

OPTIONS

```
-i, --interval <seconds>
    upload interval
    default: '600'

-p, --parameter <interface>
    set interface
    default: 'ppp0'

-o, --outbox <outbox>
    select outbox directory
    default: /mnt/dataflash/stw/cloud/OUTBOX

-d, --debug
    activate debug messages

-h, --help
    usage and options (this help)

-m, --manual
    manual page

-v, --version
    version number of the daemon
```

EXAMPLES

```
# Example output
{
  "measurements": [
    {
      "eth0_NetworkTraffic": {
        "tx_bytes": {
          "value": 14033752,
          "unit" : "bytes"
        },
        "tx_bytes_delta": {
          "value": 7542,
          "unit" : "bytes"
        },
        "rx_bytes": {
          "value": 33052679,
          "unit": "bytes"
        }
      }
    }
  ]
}
```

```

        },
        "rx_bytes_delta": {
            "value": 12468,
            "unit": "bytes"
        }
    },
    "type": "eth0_NetworkTraffic"
}
]
}

```

11.13 can_bridge - CAN interface bridging tool

Description

The `can_bridge` is a command line tool to bridge CAN traffic between two interfaces / sockets. Supported interfaces / socket types:

- CAN interface (Socket CAN)
- TCP Client
- TCP Server
- UNIX Socket Client

Synopsis

`can_bridge <option1> <option2>`

Options

`-h`
Print help information

`-v`
Print version information

`--can NAME`
Open a socket on the the CAN interface specified by NAME (e.g. `can0`).

`--tcp DESTINATION`
Open a TCP client socket and connect to the server specified by DESTINATION. The format of DESTINATION is `<Ip:Port>`, (e.g. `172.20.230.131:30000`).
Format of the CAN messages is the STW TCP-DLL format.

`--tcp-listen PORT`
Open a listening TCP socket on port number PORT and wait for a incoming connection.
Format of the CAN messages is the STW TCP-DLL format.

`--unix NAME`
Open a UNIX client socket and connect to the server specified by NAME. NAME is the path and file name of the local UNIX socket (e.g. `/var/run/service_stream1`).
Format of the CAN messages is the STW TCP-DLL format.

Exit Status

On error the exit status is set to 1 and an error messages is printed to `stderr`.
On success the exit status is set to 0.

STW TCP-DLL CAN message format

```

au8_Data[0]      'C';
au8_Data[1..4]    CAN identifier ([1]:LSB, [4]:MSB)
au8_Data[5]      Flags (TCP_CAN_FRAME_RTR_FLAG, TCP_CAN_FRAME_XTD_FLAG)
au8_Data[6]      Message DLC
au8_Data[7..14]  Message data

```

Examples

```

# can_bridge --can can0 --can can1
# can_bridge --can can0 --tcp 172.20.230.10:30000
# can_bridge --can can0 --tcp-listen 30000
# can_bridge --can can0 --unix /var/run/service_stream1

```

Notes

Supported platforms:

- x86 Linux-PC (e.g. Ubuntu) (experimental)
- PowerPC embedded Linux (EB07, TC3, TC1)
- ARM-Cortex embedded Linux (TC4P (i.MX6)) (experimental)

11.14 kefex_client - STW KEFEX console client

Description

The `kefex_client` is a command line tool for the STW KEFEX toolchain. It can handle KEFEX project files and communicate with a ECU via the STW KEFEX protocol. It implements the client side of the KEFEX communication protocol, e.g. like the RamView PC tool.

Current version: V1.06r0



WARNING:

Despite the tool provides commands to safely read and write data from / to a ECU it is not allowed to use it in a safety application. This would require a tool qualification which is not available yet!

Synopsis

```
kefex_client [options] [ListName [.VarName]]
```

Options

-h
Print help information

-v
Print version information

Config Options

-c PROTOCOL

Specify the communication protocol. Supported values for PROTOCOL are KFX (KEFEX protocol), SIP11 (SHIP-IP protocol with 11-bit CAN-ID) and SIP29 (SHIP-IP protocol with 29-bit CAN-ID).

Default value if option is omitted is KFX.

-d LEVEL

Specify the verbosity level to control the amount of status information printed to the console (stderr). Supported values for LEVEL range from 0 (none (default)) to 4 (very verbose).

-f FORMAT

Specify the output format of variable values when printing to stdout. Supported values for FORMAT are DEC (Decimal) and HEX (Hexadecimal).

Default value if option is omitted is DEC.

-i NAME

Specify the CAN interface for communication with a ECU.

Linux: NAME specifies the interface name of the hosts CAN bus (e.g. can0). The interface shall be up and running before invoking the tool.

Windows: NAME specifies the name of the STW CAN-DLL (e.g. stwpeak2.dll). The interface settings shall be defined in the DLL's INI file.

-p FILE

Specify the KEFEX project file. Both, the .DEF and .KSP project file formats are supported.

--eeprom

Specify if the invoked command shall deal with all variable lists or EEPROM lists only.

If specified, only EEPROM lists are considered. Read or write commands will access the EEPROM values directly.

If omitted, all lists are considered. Read or write commands will access the RAM values (RAM mirrors for EEPROM lists).

Command Options

--show_type

Print some variable information:

Get the list name, variable name, type, location and access rights of all variables / variable list ListName / single variable ListName.VarName from the KEFEX project and print to stdout.

Mandatory config options: -p. Optional config options: --eeprom.

--show_type2

Print some more variable information:

Get the list name, variable name, type, byte size, location, access rights, scaling factor/digits, unit, min. and max value of all variables / variable list ListName / single variable ListName.VarName from the KEFEX project and print to stdout.

Mandatory config options: -p. Optional config options: --eeprom.

--show_type2csv

Print same information as --show_type2, but print it in semicolon-separated CSV format. This format might be better machine-readable.

--show_crc

Print the KEFEX project CRC to stdout.

Mandatory config options: -p. Optional config options: none.

-r, --show_value

Read the values of all variables / variable list ListName / single variable ListName.VarName from the ECU and print to stdout.

Mandatory config options: -i, -p. Optional config options: -c, -f, --eeprom.

-w VALUE, --write_value VALUE

Write VALUE to variable ListName.VarName.

Mandatory config options: -i, -p. Optional config options: -c, --eeprom.

--dump FILE

Read the values of all variables / variable list ListName from the ECU and write it to FILE. The file extension specifies the file format. .RVI specifies the ASCII file format, all other extensions (e.g. .DAT) specify the binary data file format. Both formats are compatible with the RamView tool.

Mandatory config options: -i, -p. Optional config options: -c, --eeprom.

--flush FILE

Write all variable values specified in FILE to the ECU. ASCII and binary data file formats are supported. Both

formats are compatible with the RamView tool.

Mandatory config options: -i, -p. Optional config options: -c, --eeprom.

--sil_dump FILE

Safely read the values of all variables / variable list ListName from the ECU and write it to FILE. This command only reads EEPROM variable lists (direct EEPROM access, no RAM mirrors) regardless if option --eeprom is specified or not. A safe communication protocol is used. The data are stored to FILE in the .KDX format. The file format is compatible with the RamView tool.

Mandatory config options: -i, -p. Optional config options: none.

--sil_flush FILE

Safely write all variable values specified in FILE to the ECU. The safe.KDX file format is supported. The format is compatible with the RamView tool.

Mandatory config options: -i, -p. Optional config options: none.

--dp_create

Create a TAF (Teleservice application framework) data pool containing all variables of the specified KEFEX project. If config option --eeprom is specified only EEPROM variable lists are added to the TAF data pool. The name of the created data pool is identical to the KEFEX project name.

This command requires a running TAF ydatad daemon on the system

Mandatory config options: -p. Optional config options: --eeprom.

--dp_remove

Remove a TAF data pool. Only a data pool that has previously been created by command --dp_create can be removed. The TAF data pool to be removed is identified by the name of the KEFEX project specified by config option -p.

This command requires a running TAF ydatad daemon on the system

Mandatory config options: -p. Optional config options: none.

--dp_fill FILE

Read KEFEX variable values from a ECU and write it to the corresponding variable in a TAF data pool. The TAF data pool shall have been created previously by command --dp_create. The content of FILE specifies a set of variables from the KEFEX project that shall be read and its read cycle time. Additionally the command checks the TAF datapool for updated values and writes this values to the ECU if access is granted by the KEFEX project configuration and the settings in FILE. See command --dp_defcfg.

Mandatory config options: -i, -p. Optional config options: none.

--dp_defcfg FILE

Create a template configuration file to be used for command --dp_fill. It contains the list and variable name of all variables / variable list ListName / single variable ListName.VarName and a default cycle time of -1 for each variable. Additionally, the access right for each variable is added to the file. This setting is evaluated when a variable shall be written from TAF datapool to the ECU and allows to prohibit write access to variables even if the KEFEX project settings would allow this.

Format of FILE: One line per variable: <list name>.<variable name>;<cycle time>;<access right>.

Values for <cycle time>: -1: don't read variable; 0: read variable once; >0: read variable cyclically, cycle time in milliseconds.

Values for <access right>: RW: Read/Write, RO: Read-only, WO: Write-only, NV: Not visible.

Mandatory config options: -p. Optional config options: --eeprom.

--make_xml FILE

Create a XML file containing the configuration of all variables / variable list ListName / single variable ListName.VarName. The file is in the eiCab format.

Mandatory config options: -p. Optional config options: --eeprom.

--make_xdd FILE

Create a openPowerlink XDD file containing all variables / variable list ListName / single variable ListName.VarName. The file is in the openPowerlink XDD format.

Mandatory config options: -p. Optional config options: --eeprom.

Exit Status

In case of an error the exit status is set to 1 and an error messages is printed to stderr.

In case of success the exit status is set to 0.

Examples

All following examples assume a KEFEX project sample.ksp and a ECU connected to CAN interface can0. The CAN interface is assumed to be up and set to the correct bitrate.

Print list name, variable name and type of all variables in KEFEX project to stdout:

```
# kefex_client -p sample.ksp --show_type
```

Read command

Read RAM variable values of list System from ECU and print to stdout:

```
# kefex_client -p sample.ksp -i can0 -r System
```

Read RAM variable value of variable System.Id from ECU and print to stdout:

```
# kefex_client -p sample.ksp -i can0 -r System.Id
```

Read EEPROM variable values of list System from ECU and print to stdout:

```
# kefex_client -p sample.ksp -i can0 --eeprom -r System
```

Write command

Write value 99 to variable System.Id on ECU (write to RAM, don't touch EEPROM value):

```
# kefex_client -p sample.ksp -i can0 -w 99 System.Id
```

Write value 99 to variable System.Id on ECU (write to EEPROM, don't touch RAM value):

```
# kefex_client -p sample.ksp -i can0 --eeprom -w 99 System.Id
```

Dump/Flush command

Dump (read) EEPROM values of list System from ECU and store it to file system.rvi:

```
# kefex_client -p sample.ksp -i can0 --eeprom --dump system.rvi System
```

Flush (write) variable values from file system.rvi to EEPROM of ECU:

```
# kefex_client -p sample.ksp -i can0 --eeprom --flush system.rvi
```

TAF data pool commands

Create TAF data pool sample containing all lists and variables of KEFEX project sample.ksp:

```
# kefex_client -p sample.ksp --dp_create
```

Create a template config file var.cfg containing all lists and variables of KEFEX project sample.ksp. Edit the file to select the variables that shall be read once or cyclically:

```
# kefex_client -p sample.ksp --dp_defcfg var.cfg
```

Read values of variables specified in file var.cfg and write it to TAF data pool sample. Reading stops on program termination:

```
# kefex_client -p sample.ksp -i can0 --dp_fill var.cfg
```

Notes

Supported platforms:

- x86 Windows-PC (experimental)
- x86 Linux-PC (e.g. Ubuntu) (experimental)
- PowerPC embedded Linux (EB07, TC3, TC1)
- ARM-Cortex embedded Linux (TC4P (i.MX6)) (experimental)

11.15 Lighttpd Webserver

Lighttpd is a light weight http server adequate to run on an embedded system.

Requirements

- lighttpd must be added to the rootfs of the module (enable option in buildroot)

Configuration

- Create a configuration file, e.g /mnt/usrflash/etc/lighttpd.conf with following content:

```
server.document-root = "/mnt/dataflash/webserver/pages/"

server.port = 80

#server.username = "www"
#server.groupname = "www"

mimetype.assign = (
    ".html" => "text/html",
    ".htm"  => "text/html",
    ".txt"  => "text/plain",
    ".jpg"  => "image/jpeg",
    ".png"  => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )
```

- Copy your html pages to the path specified under server.document-root

Starting the server

- To test the syntax of the configuration file:

```
# lighttpd -t -f /mnt/usrflash/etc/lighttpd.conf
```

- To start the server:

```
# lighttpd -D -f /mnt/usrflash/etc/lighttpd.conf &
```

Useful links

<http://redmine.lighttpd.net/wiki/1/TutorialConfiguration> <http://redmine.lighttpd.net/wiki/1/TutorialConfiguration>

12 Open Source Licenses

This product contains Free Software or Open Source Software. For license details see the components below. To obtain a copy of the source code of all Free Software or Open Source Software components please contact STW:

Sensor-Technik Wiedemann GmbH
Am Bärenwald 6
87600 Kaufbeuren
Germany

Phone: +49 8341 9505-0
FAX: +49 8341 9505-55
E-Mail: info@sensor-technik.de
Web: www.sensor-technik.de

12.1 Linux Kernel License

NOTE! This copyright does **not** cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does **not** fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.

Also note that the only valid version of the GPL as far as the kernel is concerned is this particular version of the license (ie v2, not v2.2 or v3.x or whatever), unless explicitly otherwise stated.

Linus Torvalds

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it

in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary

form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in

certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
```

the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program 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 General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General
Public License instead of this License.

12.2 U-Boot License

U-Boot is Free Software. It is copyrighted by Wolfgang Denk and many others who contributed code (see the actual source code for details). You can redistribute U-Boot and/or modify it under the terms of version 2 of the GNU General Public License as published by the Free Software Foundation. Most of it can also be distributed, at your option, under any later version of the GNU General Public License -- see individual files for exceptions.

NOTE! This license does *not* cover the so-called "standalone" applications that use U-Boot services by means of the jump table provided by U-Boot exactly for this purpose - this is merely considered normal use of U-Boot, and does *not* fall under the heading of "derived work".

The header files "include/image.h" and "include/asm-*/u-boot.h" define interfaces to U-Boot. Including these (unmodified) header files in another file is considered normal use of U-Boot, and does *not* fall under the heading of "derived work".

Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the U-Boot source code) is copyrighted by me and others who actually wrote it.
-- Wolfgang Denk

=====

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an

announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt

otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

12.3 busybox License

--- A note on GPL versions

BusyBox is distributed under version 2 of the General Public License (included in its entirety, below). Version 2 is the only version of this license which this version of BusyBox (or modified versions derived from this one) may be distributed under.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This

General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the

notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is

allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is

implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program 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 General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year  name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

12.4 uClibc License

uClibc is licensed under the LGPL v2.1.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it. You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,
not price. Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights. These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you. You must make sure that they, too, receive or can get the source
code. If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that
there is no warranty for the free library. Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

Finally, software patents pose a constant threat to the existence of
any free program. We wish to make sure that a company cannot

effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work

which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of

this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is

safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
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., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:


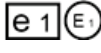
```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

13 Qualification Tests

13.1 Compliance Information

Standard		Test description	Parameter
		Conformity	see Declaration of Conformity (see "Declaration of Conformity" on page 16)
KBA (Kraftfahrt-Bundesamt)		Certification see Connecting Guidelines (see "Connecting Guidelines" on page 36)	EMC corresponds 2004/104/EG; 2009/19/EG ff.
DIN EN 62311		Assessment of electronic and electrical equipment related to human exposure restrictions for electromagnetic fields (0 Hz - 300 GHz)	
		AT&T Approval	
		PTCRB Certified	

13.2 Electrical Safety

Standard	Test description	Parameter
DIN EN 60204-1	Safety of machinery - Electrical equipment of machines - Part 1: General requirements	Compliance with air and creeping distances (Covered by software "Integra")
STW Company Standard	Supply voltage	Operation at minimum and maximum supply voltage (9-32) V Duration: 60 minutes
STW Company Standard	Overvoltage / Undervoltage	Operation at 32.96 V and 5.53 V (maximum 3 % above maximum supply voltage and below minimum supply voltage)
STW Company Standard	Cable breakage supply lines	Disconnect each single supply line at minimum supply voltage and maximum supply voltage
STW Company Standard	Short circuit strength of signal and communication lines	Short circuit test of each type of Input and Output against GND and UB at minimum supply voltage and maximum supply

Standard	Test description	Parameter
		<p>voltage.</p> <p>Short circuit test of PWM outputs and digital outputs to low resistance loads against GND.</p>
STW Company Standard	Polarity protection	Interchanged battery terminals
STW Company Standard	Current consumption	<p>Current consumption without D+ (9 V UB): 1.42 mA</p> <p>Current consumption without D+ (16 V UB): 1.77 mA</p> <p>Current consumption without D+ (32 V UB): 2.56 mA</p> <p>Current consumption with D+ (9 V UB): 361 mA</p> <p>Current consumption with D+ (16 V UB): 209 mA</p> <p>Current consumption with D+ (32 V UB): 126 mA</p>
STW Company Standard	Data retention	<p>Non-volatile memory keep data when supply voltage is disconnected</p> <p>NOR-Flash: 20 years</p> <p>NAND-Flash: 10 years</p>
STW Company Standard	Load test	<p>Operate 48 hours at minimum temperature with minimum voltage and maximum current. (-40 °C, 9V, 350 mA)</p> <p>Afterwards operate 48 hours at maximum temperature with maximum voltage and maximum current. (+70 °C, 32V, 350 mA)</p>
ISO 16750-2	Direct current supply voltage	Operation at minimum and maximum supply voltage
ISO 16750-2	Overvoltage - Systems with 12V / 24V nominal voltage	24 V system: Apply 36 V for 60 minutes at a temperature that is 20 °C below the maximum operating temperature (=50 °C).
ISO 16750-2	Superimposed alternating voltage	<p>Maximum supply voltage = 16 V (12 V system) and</p> <p>Maximum supply voltage = 32 V (24 V system)</p> <p>Sweep duration: 120 seconds</p> <p>Number of sweeps: 5</p> <p>UN: 24 V: Severity 2: Peak to peak voltage 4 V</p> <p>UN: 12 V: Severity 4: Peak to peak</p>

Standard	Test description	Parameter
		voltage 2 V
ISO 16750-2	Slow decrease and increase of supply voltage	Decrease the supply voltage from the minimum supply voltage to 0 V, then increase it from 0 V to minimum supply voltage applying a change rate of (0.5 ± 0.1) V/min linear, or in equal steps of not more than 25 mV.
ISO 16750-2	Discontinuities in supply voltage - Momentary drop in supply voltage	12 V system: Drop to 4.5 V for a duration ≤ 100 ms 24 V system: Drop to 9 V for a duration ≤ 100 ms
ISO 16750-2	Discontinuities in supply voltage - Reset behavior at voltage drop	Decrease supply voltage in steps of 5 %
ISO 16750-2	Discontinuities in supply voltage - Starting profile	12 V, Code C: <ul style="list-style-type: none"> • minimum supply voltage 9 V, • maximum supply voltage 16 V • Level I, Level II, Level III, Level IV 24 V, Code E: <ul style="list-style-type: none"> • minimum supply voltage 10 V, • maximum supply voltage 32 V • Level I, Level II, Level III
ISO 16750-2	Discontinuities in supply voltage - Load dump	Test B: 10 pulses
ISO 16750-2	Reversed voltage	Case 2 12 V system: Test voltage 14 V 24 V system: Test voltage 28 V Reversed polarity to all relevant inputs (terminals) for a duration of (60 ± 6) seconds
ISO 16750-2	Open circuit tests - Single line interruption	Interruption of each single Output for (10 ± 1) seconds
ISO 16750-2	Open circuit tests - Multiple line interruption	Disconnect the device under test (DUT) for a duration of (10 ± 1) seconds, then restore the connection.
ISO 16750-2	Short circuit protection - signal circuits	Connect every in- and output to maximum supply voltage and GND for 1 minute
ISO 16750-2	Short circuit protection - load circuits	This test is applicable only for systems/components with load circuits.

Standard	Test description	Parameter
ISO 16750-2	Withstand voltage	Perform a "damp heat cyclic" test in accordance with ISO 16750-4, then apply 500 V rms (50/60 Hz) for a duration of 60 seconds.
ISO 16750-2	Insulation resistance	Perform a "damp heat cyclic" test in accordance with ISO 16750-4, then apply 500 V d.c. (50/60 Hz) for a duration of 60 seconds.

13.3 Electromagnetic Compatibility e1/E1

Standard	Test description	Parameter	
2006/28/EG (CISPR25, DIN EN 55025)	Radiated emissions from components - ALSE method (emissions antenna)	150 kHz to 3 GHz, 1m, 120 kHz bandwidth	
2006/28/EG (ISO 11452-5/-2)	Radiated immunity	Stripline: 10 kHz - 400 MHz, 200V/m, 80 % AM Antenna, ALSE: 200 MHz - 3 GHz, 200 V/m, PM	
ISO 7637-2	Voltage transient emissions test		
ISO 7637-2	Electrical transient conduction along supply lines only	Pulse 1 (12 V): Pulse 1 (24 V):	- 100 V, 5000 pulses, 10 Ω - 600 V, 5000 pulses, 50 Ω
		Pulse 2a (12 V + 24 V): Pulse 2b (12 V): Pulse 2b (24 V):	+ 50 V, 5000 pulses, 2 Ω + 10 V, 10 pulses + 20 V, 10 pulses
		Pulse 3a (12 V + 24 V): Pulse 3b (12 V + 24 V):	- 200 V, 1 hour + 200 V, 1 hour
		Pulse 4 (12 V): Pulse 4 (24 V):	- 7 V, 2 pulses - 16 V, 2 pulses
		Pulse 5b (24 V):	US* 23 V, td 200 ms, 4 Ω

Standard	Test description	Parameter
ISO 7637-3	Test pulse generator - Fast transient test pulses a and b	24V, Level IV, 60 minutes pulse a: - 80 V pulse b: + 80 V
ISO 10605	ESD - Component immunity test method (powered-up test)	Contact discharge ± 2 kV / ± 4 kV / ± 8 kV Air discharge ± 4 kV / ± 8 kV / ± 15 kV 2000 Ω / 330 pF, 2000 Ω / 150 pF
ISO 10605	ESD - Packaging and handling (unpowered test)	Contact discharge ± 2 kV / ± 4 kV Air discharge ± 4 kV / ± 8 kV / ± 25 kV

13.4 Electromagnetic Compatibility CE

Standard	Test description	Parameter
DIN EN 61000-6-3	Emission standard for residential, commercial and light-industrial environments	150 kHz to 30 MHz conducted, 30 MHz to 1 GHz antenna
DIN EN 61000-6-2	Immunity for industrial environments - Electrostatic discharge immunity test	DIN EN 61000-4-2 330 Ω / 150 pF contact discharge ± 4 kV air discharge ± 8 kV
DIN EN 61000-6-2	Immunity for industrial environments - Radiated, radio-frequency, electromagnetic field immunity test	DIN EN 61000-4-3 80 MHz to 1.0 GHz, 10 V/m 1.4 GHz to 2.0 GHz, 3 V/m 2.0 GHz to 2.7 GHz, 1 V/m 3 m, horizontal and vertical
DIN EN 61000-6-2	Immunity for industrial environments - Immunity to conducted disturbances, induced by radio-frequency fields	DIN EN 61000-4-6 150 kHz to 80 MHz, 10V, 80 % AM, sinus at 1 kHz

13.5 EMC Radio and Telecommunications Terminal Equipment

Standard	Test description	Parameter
1999/5/EC	(R&TTE) Radio equipment and telecommunications terminal equipment	
ETSI EN 301489 - 1	ElectroMagnetic Compatibility (EMC) standard for radio equipment and services; Part 1: Common technical requirements	GPS

Standard	Test description	Parameter
ETSI EN 301489 - 7	ElectroMagnetic Compatibility (EMC) standard for radio equipment and services; Part 7: Specific conditions for mobile and portable radio and ancillary equipment of digital cellular radio	GSM & DCS
ETSI EN 301489 - 17	ElectroMagnetic Compatibility (EMC) standard for radio equipment and services; Part 17: Specific conditions for Broadband Data Transmission Systems	Bluetooth & WLAN

13.6 Environmental Influences

Standard	Test description	Parameter
ISO 16750-3	Free fall	3 devices, 2 falls every device on the opposite side of the housing. Drop height: 1 m to concrete ground or steel plate
DIN EN 60068-2-6	Vibration (sinusoidal)	10 Hz to 2 kHz, 1 oct./min., 5 g, 10 cycles, bidirectional
DIN EN 60068-2-27	Shock	50 g, 11 ms, sinus, 3 shocks/axis
DIN EN 60068-2-27	Bump	Acceleration: 30 g, time: 6 ms, sinus, 1000 shocks/axis
ISO 16750-4	Tests at constant temperature - low temperature - storage	24 hours at -40 °C
ISO 16750-4	Tests at constant temperature - low temperature - operation	24 hours at -40 °C
ISO 16750-4	Tests at constant temperature - high temperature - storage	48 hours at 85 °C
ISO 16750-4	Tests at constant temperature - high temperature - operation	96 hours at 70 °C
ISO 16750-4	Temperature step test	Decrease the temperature in steps of 5 °C from 20 °C to -40 °C, then increase the temperature in steps of 5 °C from -40 °C to +70 °C

Standard	Test description	Parameter
ISO 16750-4 IEC 60068-2-14, Test Nb	Temperature cycling test	30 cycles each 8 hours, from -40 °C to +70 °C
ISO 16750-4 IEC 60068-2-14, Test Na	Temperature cycling test - rapid change of temperature	100 cycles, from -40 °C to +70 °C transfer time ≤ 30 seconds dwell time: 60 minutes
ISO 16750-4 IEC 60068-2-52, Test Kb	Salt spray tests - corrosion test	Severity level 5
ISO 16750-4 IEC 60068-2-38, Test Z/AD	Humid heat, cyclic test - Test 2: Composite temperature / humidity cyclic test	10 cycles each 24 hours, 5 of them with frost phase (-10 °C) upper temperature: +65 °C, 93 % relative humidity
ISO 16750-4 IEC 60068-2-78	Damp heat, steady-state test	Severity: (40 ± 2) °C and (85 ± 3) % humidity test duration: 21 days - 20 days and 23 hours without operation, the last hour in typical operation mode
ISO 16750-4 IEC 60068-2-60, Test Ke, method 4	Corrosion test with flow of mixed gas	Test duration: 10 or 21 days (SO ₂ , H ₂ S, NO ₂ , CL ₂)
IEC 60529	IP code	IP6x, IPx7, IPx9k
ISO 16750-5	Chemical loads	Sulfuric acid 37 %, brake fluid, engine oil, hydraulic oil, petrol/gasoline unleaded, urea, diesel fuel cold cleaning agent

14 Index

A

Access Data Pool • 189
 Access-Point-Mode • 59
 Adapt the root file skeleton • 287, 290
 Add master save condition to DLC file • 235
 Add save condition to DLC file • 238
 Add variable to DLC file • 234
 Add variable to DPL file • 188
 Additional Available Supplies • 22, 23
 Antenna • 38
 Application Notes • 336

B

Basic DLC file structure • 230
 Basic DPL file structure • 187
 Basic folder structure • 187
 Beeper • 90
 Block Diagram • 27
 Bluetooth • 67
 Bluetooth M2M • 344
 Boot up / Shut down sequence • 42, 50, 51, 68, 87
 BSP Components • 322, 323, 324, 327
 busybox License • 378

C

CAN • 66, 342
 CAN for ISOBUS • 343
 can_bridge - CAN interface bridging tool • 362
 Certificate Handling • 128
 Code Blocks • 298
 Command Line • 301
 Communication Interfaces • 29, 336
 Compliance Information • 392
 Connect TC3G • 23
 Connect to Development Box • 22, 23
 Connecting Guidelines • 36, 392

Connector • 22, 31, 32, 36, 38
 Contact • 14
 Copyright • 15
 Create New Project • 296, 298
 Create Own Application • 296, 312, 320
 Create Own Root File System • 21, 287

D

Data daemon • 98
 Datalogger • 204
 Datapool • 174
 D-Bus Utils • 149
 dbus_call_method • 157
 dbus_call_signal • 159
 dbus_close • 152
 dbus_get_on_the_bus • 151
 dbus_initialize_request_callbacks • 21, 160, 247, 252
 dbus_process_requests • 165
 dbus_send_goodbye_signal • 155
 dbus_send_hallo_signal • 153
 dbus_who_is_there_signal • 156
 Declaration of Conformity • 16, 392
 Development Tools • 287
 Digital Input / Output • 88
 Disposal • 19
 Documents • 17
 Download, extract and test setup • 20, 287, 289
 Dynamic mode • 206
 Dynamic Mode • 175

E

EEPROM • 83
 eGPRSComServer specific commands • 257
 Electrical Safety • 392
 Electromagnetic Compatibility CE • 396
 Electromagnetic Compatibility e1/E1 • 395
 E-Mail • 21, 347

EMC Radio and Telecommunications Terminal Equipment • 396
 Enable or Disable Buildroot Packages • 287, 290
 Environmental Influences • 397
 Error Codes • 147
 Ethernet • 56
 Examples • 320
 Examples of Country Codes • 62, 63
 Extend BR by adding packages • 287, 294
F
 Function descriptions • 145
G
 General Information • 14
 Getting started with TC3G • 22
 GPRS / Peer-To-Peer • 53
 GPS • 86, 244
 GPS daemon • 86, 102
 GSM • 20, 51, 264, 341
 GSM daemon • 20, 104, 247
 GSM SIM Card • 341
H
 Handle Logger File • 240
 Hardware • 27
 Housing • 36
 How to Secure the System • 26
I
 I/O Pin's • 340
 Ignition • 45
 INBOX • 133
 Index of Abbreviations • 6
 Initialize the D-Bus • 150
 Inputs and Outputs • 30
 Insert SIM Card • 22
 Install the Code
 Blocks IDE • 304
 Blocks IDE • 314

Install the Toolchain • 301, 303, 313
 Introduction • 13, 140, 149, 152, 154, 167, 174, 204, 213, 219, 222, 230, 235, 238, 244, 247, 252, 257, 266
 Introducton • 264
 IP handling • 64
K
 kefex_client - STW KEFEX console client • 363
L
 LED • 91
 Libraries • 321
 Lighttpd Webserver • 367
 Linux • 303
 Linux Folder Structure • 75
 Linux Kernel License • 368
 Linux Updater • 325
 Logger daemon • 100
M
 machines.cloud daemon • 20, 130
 Mail daemon • 21, 123
 Managed-Mode • 62
 Memory handling • 71
 Motion Sensor • 48
 Mounting Guidelines • 33
N
 NAND Flash • 82
 Network • 252
 Network daemon • 50, 53, 56, 58, 108
 Network handling • 50
 NFS • 300, 302, 312, 319, 337
 NOR Flash • 74
 Notation • 142
O
 Open Source Licenses • 21, 368
 OUTBOX • 136
 Overlay Filesystem • 20, 21, 79, 327

Overview • 94

P

Pin Assignment • 23, 32

Power Supply • 28

Power up Device • 25

Processor and System Memory • 28

Q

Qualification Tests • 20, 392

R

Real Time Clock • 46

S

Serial • 87

Server • 257

Server daemon • 113

Server independent commands • 262

Setting up the Serial Interface • 24, 336

Setup the Code

Blocks IDE • 305

Blocks IDE • 314

Setup the Compiler for Debugging • 20, 309, 316

Setup ymail for the first time • 123, 126, 128

Signal • 21, 266

Signal daemon • 21, 91, 117, 266, 268

SMS • 20, 54, 247

Software • 40

Static mode • 230

Static Mode • 187

STW Build Scripts • 287, 290, 294

STW Folder Structure • 78

STW Update Mechanism • 329

STW Update Procedure • 329

STW Update Procedure Flow Charts • 329, 334

stw_acc2can • 353

stw_CANsnapshot • 20, 357

stw_dptool • 349

stw_flash_client • 355

stw_GetGPS • 350

stw_GetGSMInfo • 20, 358

stw_ReadACC • 351

stw_RecvSMS • 356

stw_RetentionRule • 20, 360

stw_SendSMS • 356

stw_show_gps • 354

stw_TrafficMonitor • 20, 361

System • 167

System daemon • 45, 85, 96

System Data • 31

System Information • 40

T

TAF Components • 95

TAF Library • 140

Target Group • 17

TC3G_History • 20

Technical Data • 28

Teleservice Application Framework • 94

Telnet • 338

Temperature Sensor • 86

TFTP • 337

Toolchain • 296, 303

Types and Prefixes • 142

U

U-Boot License • 374

uClibc License • 384

Universal Serial Bus • 89

Update the Device • 287, 288, 322

Used Symbols and Formats • 18

Utilities Tools • 20, 349

Utils • 270

utils_close_dir • 285

utils_create_directory • 270

utils_directory_exists • 272

utils_fifo_exists • 274

utils_file_exists • 271

utils_get_GM_time • 284

utils_get_hostname • 282

utils_get_time • 283

utils_get_time_ms • 284

utils_get_time_sig • 283

utils_init_semaphore • 278

utils_link_exists • 273

utils_log_close • 277

utils_log_print • 277

utils_open_dir • 285

utils_open_log_file • 276

utils_read_dir • 286

utils_reg_file_exists • 272

utils_remove_directory • 271

utils_remove_file • 278

utils_search_first_value • 275

utils_search_next_value • 275

utils_semaphore_getid • 280

utils_semaphore_lock • 280

utils_semaphore_remove • 279

utils_semaphore_unlock • 281

utils_socket_exists • 273

utils_strncpy • 274

utils_write_string_value • 282

W

Wakeup • 44

Warranty • 17

Watchdog • 85

Windows • 313

Windows Updater • 323

WLAN • 21, 58

Y

ydatad_add_variable_to_list • 176

ydatad_check_update_list • 201

ydatad_check_update_variable • 202

ydatad_create_variable_list • 175

ydatad_delete_datapool • 184

ydatad_delete_variable_list • 180

ydatad_get_datapool_base_path • 181

ydatad_get_datapool_index • 193

ydatad_get_datapool_info • 191

ydatad_get_list_index • 196

ydatad_get_number_of_loaded_datapools • 190

ydatad_get_updated_variable • 202

ydatad_get_variable • 198

ydatad_get_variable_index • 194

ydatad_get_variable_info • 200

ydatad_init_datapool • 183

ydatad_init_variable_list • 178

ydatad_load_datapool • 189

ydatad_load_datapool_names • 190

ydatad_set_variable • 197

ydatad_validate_variable • 185

ygpsd_get_gps_data • 20, 86, 244

- ygsmd_get_gsm_data • 264
- ygsmd_request_sms_fetch_urgent • 250
- ygsmd_send_sms • 247
- ygsmd_send_sms_urgent • 248
- ylogd_activate_log_job • 228
- ylogd_add_log_variable • 210
- ylogd_create_log_job • 206, 213, 214, 215, 216
- ylogd_delete_log_file • 241
- ylogd_request_data_logger_file • 240
- ylogd_set_buffered_logging • 217
- ylogd_set_log_file_cmd • 227
- ylogd_set_log_management • 225
- ylogd_set_log_mechanism • 213
- ylogd_set_master_save_condition • 219
- ylogd_set_save_condition • 214, 221
- ylogd_set_trigger_time • 213, 214, 215
- ylogd_set_variable_change_cmd • 211
- ylogd_trigger_buffered_log • 217, 242
- ynetworkd_get_connected_interface • 50, 252
- ynetworkd_start_connection • 50, 254
- ynetworkd_stop_connection • 50, 255
- yserverd_request_nextjob • 262
- ysignald_GPS_con_state • 120, 268
- ysignald_internet_con_state • 120, 266
- ysysd_cancel_watch_dog • 85, 169
- ysysd_get_ignition_status • 45, 170
- ysysd_register_watch_dog • 85, 167
- ysysd_request_stay_alive • 171
- ysysd_trigger_watch_dog • 85, 168