Teststand of Electrical Drives, Acquisition of Stationary and Transient Quantities Using a Personal Computer

Richard Leeb, Wolfgang Kurt Mörtl
Institute for Electrical Engineering, University Leoben

19th September 2001

„Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.“

“I declare on oath that I have done this diploma-theses independently and without help of others with no use of any other sources and aids than the indicated and that I have indicated all literally and texual taken passages from used sources.”

Unterschrift des Diplomanden
Diplomand’s Signature

Abstract

In order to keep track with technical development in electrical drive engineering, the Institute for Electrical Engineering enforced the development of low power traction drive systems for vehicles up to 20 [kw] maximum Power used for e.g. scooter golf mobiles, and zero-emission vehicles for short distances. The improvement of such systems determined the build up of an automated test stand. This test stand monitors the whole drive. A personal computer (PC) and measurement devices linked with several interfaces meet the demands regarding the data acquisition. Additionally reliability and security of the system is ensured with a programmable logic controller (PLC). The software implementation features a modular concept, a standardized measurement protocol and a user friendly interface. The modular concept allows expanding the Test-Stand for future development. A standardized measurement protocol ensures easy data processing. A user friendly interface keeps training periods short.
Furthermore this measurement system is used for teaching LabView and electrical drive engineering. Future development will be a fully automated test stand control with access via internet.

Abstrakt

Sensorlose Regelungsverfahren kombiniert mit einer hoch ausgenützten, optimal dimensionierten (hohes Anfahrmoment und groSSer Drehzahlbereich) pulswechelsrichtergespeisten Asynchronmaschine ergeben eine neue, hocheffiziente, robuste, warungsfreie und zugleich kostengünstige Generation von Traktionsantrieben. Eine Verbesserung solcher Traktionsantriebe und deren Antriebsysteme bedingte den Aufbau eines automatisierten Prüfstandes.
Zu den Aufgaben eines solchen Prüfstandes zählt die Aufnahme von Meßdaten und die Überwachung des Betriebzustandes von Testantrieb und dessen Versorgung. Die Datenaufnahme wird von einem Personal Computer (PC) und mehreren Meßgeräten übernommen. Die Verbindung zwischen den Geräten wird durch verschieden Bussysteme realisiert. Mit Hilfe eines PCt's und einer speicherprogrammierbaren Steuerung (SPS) wird Für Zuverläss igkeit und Sicherheit während der Messung garantiert.
Die eigens entwickelte Software zeichnet sich durch einen modularen Aufbau, ein standardisiertes Meßprotokoll und durch eine benutzerfreundliche Bedienoberfläche aus. Das modulare Konzept der Software erlaubt es auch neue Meßgeräte in den Prüfstand zu integrieren. Ein standardisiertes Meßprotokoll soll eine leichte und nachvollziehbare Meßdatenauswertung gewährleisten, und mit einer benutzerfreundlichen Bedienoberfläche soll die Einschulungszeiten neuer Benutzer kurzgehalten werden.
Neben Forschung und Entwicklung findet der Prüfstand auch in der Lehre Verwendung finden. Studenten wird die Antriebstechnik und die Programmiersprache G (LabView) näher gebracht werden.

Inhaltsverzeichnis

Kapitel 1

Introduction

1.1 Challenge

The Institute of Electrical Engineering's main scientific work focuses on power electronics. This includes the development of whole drive (motor design, inverter, control, electronics ...) systems as well as the development of large power conversion facilities.

When developing traction drive systems a lot of measurement has to be done. Data acquisition itself is a very structured work to do. It is often time consuming with little variety. To release highly educated workforce for other more challenging tasks and also to decrease the costs, the build up of an automated test stand was decided.

The two main motivations are saving time and saving money. Besides these the test stand should also facilitate the analysis and improve safety matters and the reliability of the data.

1.2 The main theme of interest

Improvement and development includes the whole traction drive system. The drive system encloses the electrical motor, the inverter, the control of the drive, mechanical parts and the source of power. The drive system may also enclose any mechanical components like the shaft or the coupling.

Interesting quantities for examination of the traction drive system are shown in Table 1.1 on page 2. If these quantities are known, the motor performance can be calculated.

Further targets for examination are

- the motor with its parameter and their dependencies, its behavior while steady state or transient state;

Quantities	*Affected part of the system*
current RMS	electrical motor
voltage RMS	electrical motor
power	electrical motor
power factor	electrical motor
frequency	supply
speed	electrical motor
torque	electrical motor
temperature	electrical motor, housing, coupling

Tabelle 1.1: Examined Quantities

- the battery, its capacity and its charging properties;
- the inverter and the Digital Signal Processor (DSP) and
- the interaction between the parts of the drive (estimated efficiency, travelling range . . .).

Therefore an Automated Test Stand should allow to look at the whole drive system development with a more widespread view.

1.3 Demands

The following demands are made on the test stand:

- full automatization
- timesaving (easy use)
- standardized measurement protocol (easy evaluation)
  in detail:
  - time;
  - current, voltage (L1, L2, L3 to induction motor , line);
  - temperatures:
    - cooling sink
    - induction motor
    - load
    - environment
    - shaft
  - torque

  - rotating speed

- testing of the whole drive system from batteries to inverter;
- defined quality assurance;
- safe operation(software, hardware, environment);
- flexibility for further extensions

1.4 General Framework

The Test-Stand can be logically divided into *Test-Object* and *Test-Bench*. The Test-Object is the target of testing, the Test-Bench is everthing needed for testing the Test-Object. Test-Bench consists of *Measurement-System* and *Control-System*. All parts needed for data acquisition are referred to Measurement-System. Parts including controlling tasks are counted among Control-System. A possible carrying out of the Test-Stand is shown in figure 1.1 on page 4

1.4.1 Test-Bench

Measurement-System

The Measurement-System consists of an Intel based Personal Computer (PC) with a built in Data AcQuisition (DAQ) board and a General Purpose Interface Bus Board (GPIB) both from National Instruments. A Poweranalyzer by LEM Norma a Dewerack-16 with several signal conditioning plug-in modules (Pads) by Dewetron and several temperature-, torque- and rotating-speed-sensors are the measurement devices.

Control-System

The Control-System consists of a load system by Lenze (inverter and motor), a programmable logical controller (PLC) S7-S212 by Siemens.

The task is to combine these parts logically and physically to a Test-Bench. Logically with a suited program written in G (Labview) and physically with the appropriate connections. While testing the Test-Object this Test-Bench has to meet all the demands mentioned above.

1.4.2 Test-Object

Whole traction drive systems up to 13 kW rated power and 22 kW maximum power (short time operation) are the target of testing.

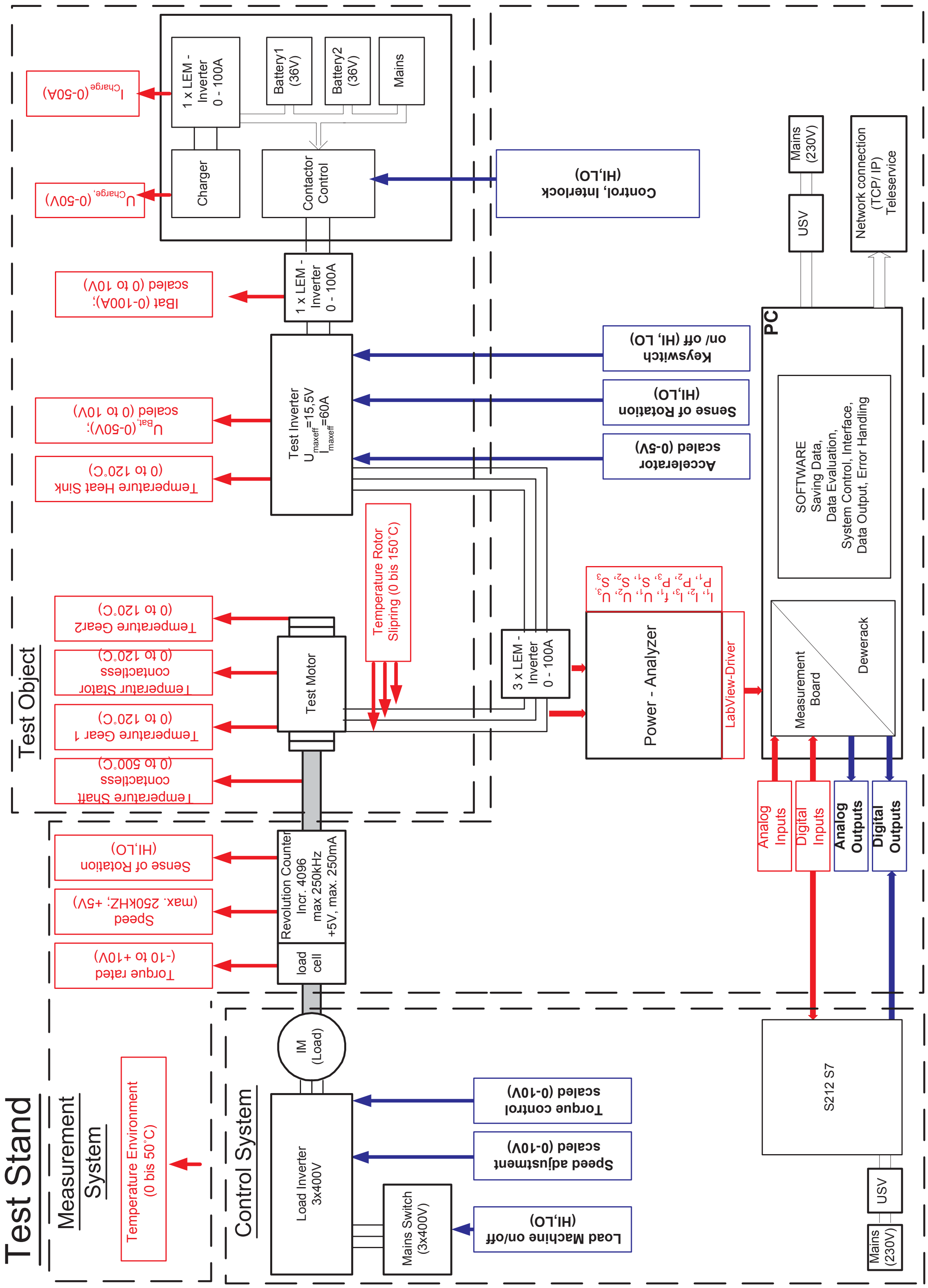


Abbildung 1.1: Possible Measurement Concept

Kapitel 2

The Test Stand

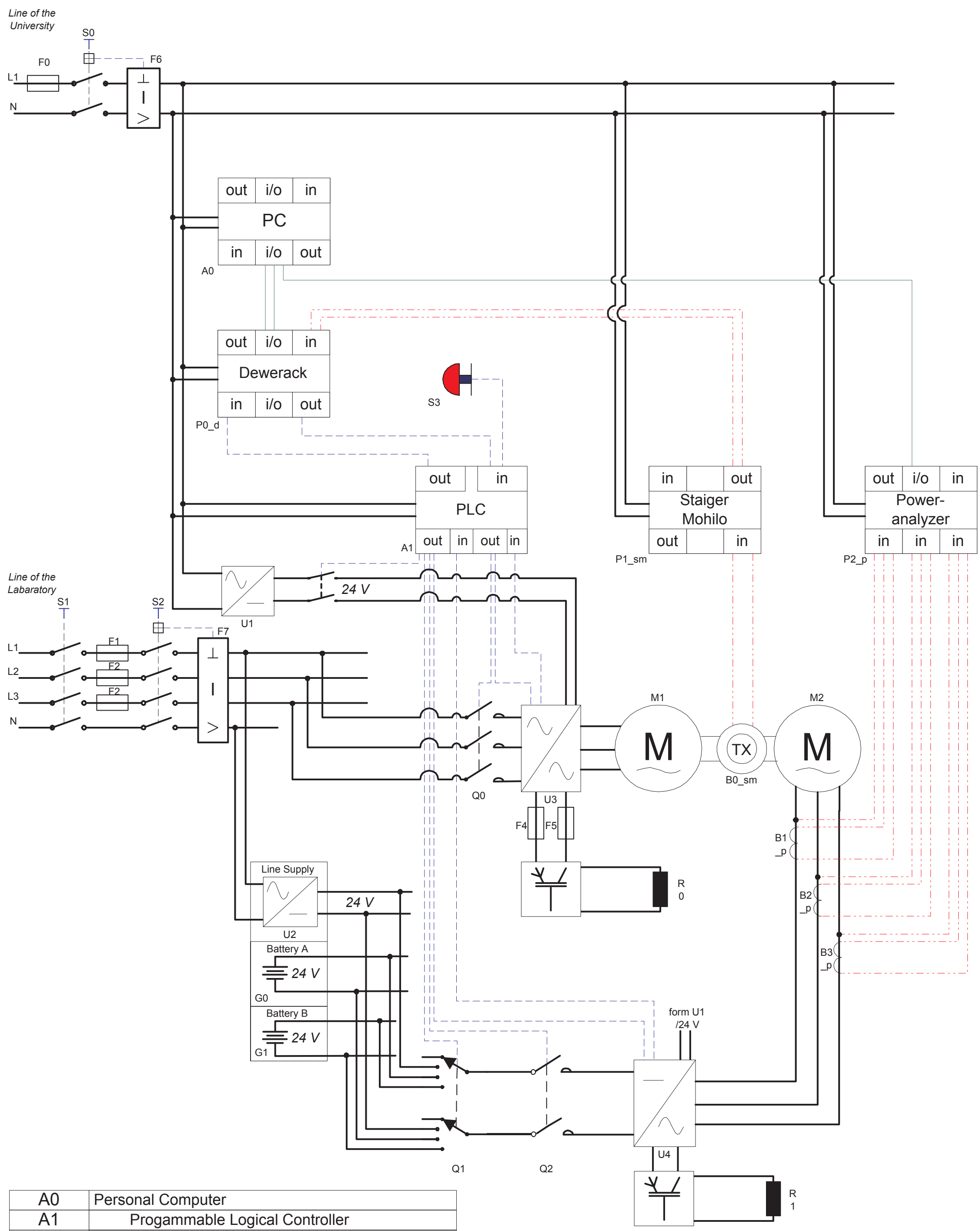
2.1 Structure

The general system structure is shown in picture 2.1 on page 6.

2.2 Test Bench - Measurement System

The measurement system consists of:

1. personal computer with
   - DAQ device
   - GPIB device
   - RS232 device
2. Poweranalyzer with
   - GPIB interface
3. Dewerack with
   - RS232 interface
   - DAQ interface
   - PAD-modules and DAQ-modules
   - attached sensors:
     - speed
     - torque
     - temperature



A0	Personal Computer
A1	Progammable Logical Controller
B0_sm	Speedcounter, Inductive Angular Positioning
B1-3_p	3 Phase Current Sensor Module
F1-F5	Fuses
F6-F7	Earth-Leakage Circuit-Breaker
G0-G1	Batteries 24 V
M0	Load Machine
M1	Test Machine
P0_d	Dewerack 16 spezial

P1_sm	Torque and Speed Measurement System
P2_p	Poweranalyzer
Q0-Q3	Contactors
R0-R1	Breaking resitor
S0-S3	Switches (S3 Emergency button)
U0-U4	Inverters

Abbildung 2.1: System Wiring Diagram

Personal Computer Resources

Internal System Resources			*Additional Information*	
Processor Speed	500	MHz	Intel Pentium III	
Memory	128	MB	DIMM-RAM-ECC	
Cache	512	kB		
Hard disc capacity	20	GB		
Internal System Interfaces			*Additional Information*	*Connection to*
Printer Interface	1 x		EPP	
RS232 Interfaces	2 x		RS232c	Dewerack, Load Inverter
PS/2 Interfaces	1 x			
USB Interfaces	1 x			
Plug-in Boards			*Additional Information*	*Connection to*
Measurement Board	PCI -	6024E	NI Multi I/O Board	more than one option
GPIB	PCI -	GPIB	NI-448.2, X2 cable	Poweranalyzer
Ethernet board	PCI		Kingston fast Ethernet	Local Area Network
Graphic board	AGP		Matrox G400	Monitor

Tabelle 2.1: List of Resources

2.2.1 Personal Computer

To give an idea of the available hardware capacity, system specification are shown in Table 2.1 on page 7.

In the following devices will be described in a more detailed way.

RS232 Interfaces

This kind of serial interface can be found in almost every IBM-compatible PC. The data transfer rate in this system is set to 9600 bits per second (bps). A higher rate can be achieved if necessary.

There are two Recommended Standard 232 (RS232) Interfaces and both are in use. The first COM port is used to realize the communication between PC and the Dewerack. There are four pads which are affected from this communication link. Two of these pads DE-PAD-TH8-K and DE-PAD-V8 are used for signal conditioning of all measured temperature values. These values are taken by seven thermoelements type K and two infrared temperature sensors. Two pads are of type PAD-D07. They have seven digital outputs each and are used for communication with a PLC S7-212.

The second COM port is used to communicate with the load inverter. In the future this task can be realized via CAN-bus.

PCI 6024 Low Cost Multifunctional I/O Board

There are 16 analog inputs, two analog outputs, eight digital in- and outputs and two counters. If only one input channel is used, the board will offer the ability to scan it

200 000 times per second. If more than one channel is used the scan rate decreases accordingly to the number of the channels used. The resolution of a measurement point is 12 bit.

The measurement board is connected with the Dewerack. Several channels are combined with pads installed in the Dewerack or only passed through to connectors to the chassis of the Dewerack. Six pads type DE-DAQ-V-D are connected with the first six analog inputs of the board. Two pads type DE-DAQ-Out are connected to the two existing analog outputs. One counter input channel is assigned to one BNC connector of the Dewerack. One of the digital input/ output (I/O) channel is connected to a BNC connector to control the rotary direction, the rest of the digital I/O channels are passed through to a sub-D connector located at the back of the Dewerack.

Two analog inputs are in use. They are used to survey torque and rotation speed.

GPIB PCI plug-in board

This board is needed to realize an interconnection between the PC and the Power-analyzer via IEEE 488.2, also called the General Purpose Interface Bus (GPIB).

The IEEE 488 interface is a general purpose digital interface system that can be used to transfer data between two or more devices.

Some of its key features are: Up to 15 devices may be connected to one bus. Total bus length may be up to 20m and the distance between devices may be up to 2m. Communication is digital and messages are sent one byte (8 bits) at a time. Message transactions are hardware handshaked. Data rates may be up to 1 Mbyte/sec.

Ethernet board

The Kingston Ethernet board connects the local PC to the local area network of the department. This provides the possibility to future extensions like online publication in the internet or online control.

2.2.2 Dewerack-16 Serial with Special Features

Dewerack-16 in combination with installed plug-in pads is a signal conditioning rack. Possible input signals are set according to the used DEWE-DAQ, DEWE-PAD modules and the connectors on the rear (see manuals [8, 9]).

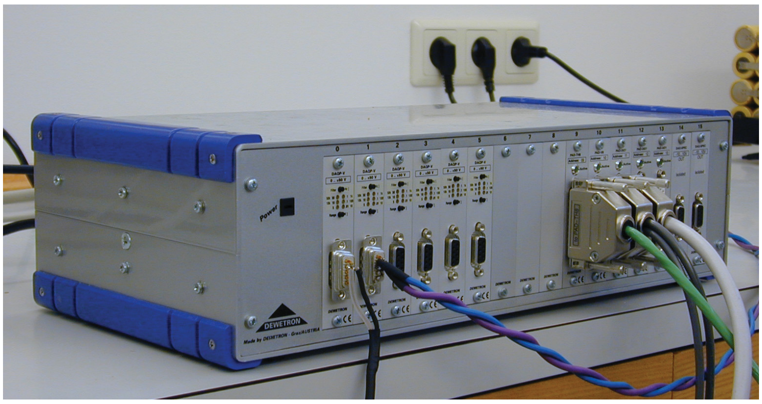


Abbildung 2.2: Dewrack-16 Serial Special

The use of the avaialble 16 slots is shown in the following:

- slot zero to five: DAQP-V
- slot six to eight: unused
- slot nine: PAD-TH8
- slot ten: PAD-V8
- slot 11 to 12: PAD-D07 (out)
- slot 13: PAD-A01 (out)
- slot 14 to 15: DAQ-SPEC (out)

DAQxx- Modules

DAQP-V modules are voltage isolation amplifiers, standardized for voltages from $\pm\, 10\, mV$ up to $\pm\, 50\, V$. Using a shunt resistor currents also can be measured. Signal conditioning is done by selectable low pass filters. The selection ranges from 10 Hz to 10 kHz. There are two ways to change measurement range and input filters. One is realized with button selection, the selection will be stored to an electrical erasable programmable read only memory (EEPROM). The second is a temporary setting via RS-232/485 interface. The analog digital (A/D) conversion is realized external by the measurement board in the PC.

DAQ-SPEC modules are output modules. They provide an isolated voltage output from -10 to +10 V. The actual output depends on the DAC-output-voltage of the measurement board. The proportion is 1:1.

PADxx- modules

All PAD- modules are to program via a RS232 interface. Data operations (read, write operations), configuration setting and calibration are done via this interface. The maximum transfer speed concerning the modules specification is 9600 bps.

PAD-TH8 modules are eight channel thermocouple amplifiers. Eight thermocouple type K can be connected to this module (seven are currently installed). These channels are differential with an resolution of 16 bit. The input signals are converted by an integrated A/D converter and transferred via RS232 interface.

PAD-V8 modules are eight channel amplifiers for voltage and current measurement. These input channels are differential with a resolution of 16 bit. Voltage ranges are available from ± 10 V to ± 150 mV, for a current range of 20 mA an external 125 Ω shunt resistor is necessary. The input signals are converted by an integrated A/D converter and transferred via RS232 interface. Both infrared temperature sensors are connected to this module.

PAD-D07 modules are seven relay output modules with maximum load 0.5 A using 60 VAC or 1 A using 24 VDC featuring a high isolation voltage of 1000 V. The relays itself are form 'A' type relay SPST N.O. Relay on time accounts to 5 ms. Their state can be set via the RS232 interface. Five of the 14 relays are used for the communication between the PC and the PLC S7.

PAD-A01 modules are one channel output (digital to analog) modules. The output resolution is 12 bit and the range of the output signal lasts from 0 to 10 V or from 0 to 20 mA respectively from 4 to 20 mA. The kind of the output and its value is set via the RS232 interface.

Rear Connectors

In this special configuration three BNC connectors and a 37 pin female Sub-d connector are mounted to the rear panel of the Dewerack-16.

The first BNC connector (DI0) is wired to "Digital I/O 0" of the measurement board, the second (Counter0) and the third (Counter1) to "PFI8/GPCTR0_Source" respectively to PFI3/GPCTR1_SOURCE both are counter inputs of the measurement board.

Via the Sub-d connector the remaining digital in- and outputs of the measurement board "Digital I/O_"(one to seven) can be wired.

2.2.3 Poweranalyzer

Abbildung 2.3: Poweranalyzer D4000

Figure 2.3 shows the Poweranalyzer NORMA D4000 which is a true root means square (TRMS) measurement device for three phase power measurement. Measurement categories are voltage, current, frequency and derived values like power and powerfactor (see manual [10]).

A three phase current sensor module increases the possible measurable current to 100 ampere.

In addition to the voltage and current inputs four analogue inputs and four analogue outputs are implemented and combined to a 25 pin female Sub-d connector.

Voltage

There are 8 voltage ranges selectable, from 0,3 V up to 1000 V. Values to be measured can be TRMS, rectified mean value, averaged value, peak to peak value, minimum and maximum value, peak factor and form factor.

Current

The measurable current depend on the input used. There are two inputs per line, a five ampere input with possible selectable ranges lasting from 15 mA to 5 A and a 15 ampere input with ranges from 1 A to 30 A. Values to be measured can be TRMS, rectified mean value, averaged value, peak to peak value, minimum and maximum value, peak factor and form factor.

Power

There are 80 ranges according to the product of voltage and current. Values to be measured are real power, complex power and reactive power.

Frequency

The measurable frequency ranges from direct current (DC) to 300 kHz. NORMA D4000 can calculate Fast Fourier-Transformation (FFT)in a range from 2 Hz to 100 kHz referring to the actual current, voltage or real power.

External Communication

This measurement device is also capable to communicate with other devices. The interfaces built in are IEEE 488.2, RS232 and Centronics. IEEE 488.2 is connected with GPIB PCI plug-in board of the PC. For further detail please refer to the manual of the Poweranalyzer Norma D4000 [10].

2.2.4 Sensors

In the measurement system implemented sensors are: Nine temperature sensors, three hall sensors, and a metering shaft for torque and speed detection.

Temperature Sensors

Seven of the nine temperature sensors are thermocouples of the type K the measurement range lasts from - 50 °C to 1300 °C. They are connected via PAD-TH8 to the Dewerack 16.

The remaining two senors are infrared (IR) thermometers. They allow a non-contact temperature measurement up to 1371 °C. They are connected via PAD-V8 to the Dewerack 16.

Hall Senors

The three hall senors are part of a three phase current sensor module. This module is used to measure currents up to 100 ampere in a frequency range from DC to 10 kHz.

The hall sensors are supplied by the analogue outputs of the Poweranalyzer and connected to the five ampere input of the same.

Network Filter Characteristics

Type	*Design Value*		*Spark Suppression Rating*	
	Rated Current	*Inductivity*	*A*	*B*
9328	42 A	0,8 mH		o

Tabelle 2.2: Network Filter Characteristics

Metering Shaft- Torque

The torque is proportional to the angle of torsion. The angle of torsion of the shaft between load machine and traction machine is measured using an inductive angular position measuring system. The measurement system converts the angle to a proportional signal. The torque is derived from this signal.

The signal is conditioned and amplified by a torque and speed measurement system by Staiger & Mohilo and passed on to a DAQP-V module of the Dewerack.

Metering Shaft - Speed

The speed is measured with an optical speed counter. The optical speed counter is realized as speed measurement device with optical scanning of a raster disc.

The signal is conditioned and amplified by a torque and speed measurement system by Staiger & Mohilo (see manual [6, 7]) and passed on to 'Counter0' of the Dewerack.

2.3 Test Bench - Control System

2.3.1 Load

The load consists of a network filter, an inverter, a brake chopper, an external braking resistor and an induction machine. All parts are purchased from the same producer, 'Lenze Antriebstechnik Ges.m.b.H' and are phased on each other. The system is designed for a power range up to 13,2 kW rated power with an overload capacity of 22 kw (see manuals [2, 3, 4, 5]).

Network Filter

For an absence of feedback concerning the public mains a network filter is installed. The network filter also contains anti-interference components. Its characteristics are shown in table 2.2 on page 13 see [3] (page 3-7).

Load Inverter Characteristics

Type			*9328*
Servo Inverter			EVS9328-ES
Mains Voltage	U_N	[V]	320-528
Mains Frequency	F_N	[Hz]	45-65
Alternative DC Supply	U_G	[V]	460-740
Motor Rating (4. pol ASM)	P_N	[KW]	22,0
Output Current (8 kHz)	I_{N8}	[A]	47
Output Current (16 kHz)	I_{N16}	[A]	30,6
Output Rating	S_N	[kVA]	32,6
Max Output Current (8 kHz)	I_{N8}	[A]	70,5
Max Output Current (16 kHz)	I_N	[A]	44,0
Mains Current (U_{mains} 400 V)	I_{N16}	[A]	30,6
Motor Voltage	U_M	[V]	$3 \sim 0 \cdot\cdot U_{mains}$
Dissipation (U_{mains} 400 V)	P_D	[W]	640
Power Reduction (40 °C - 50 °C)		$\%/K$	2
Power Reduction (1000m - 4000m)		$\%/km$	5

Tabelle 2.3: Load Inverter Characteristics

Load Inverter

The load inverter is one of the vector control frequency inverters from the 9300 series of Lenze. It is a servo inverter with technology and PLC functions.

These technology functions mean that additional external controls or mechanical speed controllers can be omitted. It is capable of speed, torque and angular control of servo motors including synchronous motors and induction motors. Integrated PLC functionality in the servo inverter means that the control and regulating functions can be wired as required. These functions can be internally adapted to the tasks with a programming language from automation engineering (IEC 1131-3) or by means of function block wiring.

There are several ways to access the inverter to change system parameter or set regulating functions. It can be done via a console or by "Global Drive Control" a software for a PC. The connection between PC and inverter can be realized with an additional interbus module, a profibus module, or a RS232/485 module.

The power consumption is covered by the 3AC/400V/50Hz line of the laboratory. The inverter control is supplied by external 24 V. Table 2.3 on page 14 is an overview of the main characteristics. Taken from catalogue [2] page 22.

Brake Chopper with External Braking Resistor

Dynamic braking leads to an inverse energy flow. To cope with this energy one can either recover it to the main or consume it by a braking resistor. In this system the

Break Chopper Characteristics

Break Chopper	Type EMB9352-E		
Supply Voltage	U_S	[V]	270-780
Max Current (DC)	I_{max}	[A]	42
Continuous Rating	P_c	[KW]	19
Peak Power (max 60 sek)	P_p	[KW]	32
Smallest Resistor	R	[Ω]	18
Operating Point (U_{mains} 400 V)	U_{op}	[V]	725 (DC)
Operating Point (U_{mains} 460 V)	U_{op}	[V]	725 (DC)
Operating Point (U_{mains} 480 V)	U_{op}	[V]	765 (DC)

Tabelle 2.4: Break Chopper Characteristics

Break Resistor Characteristics

Break Resistor	Type ERBD022R03k0		
Resistance	R	[Ω]	22
Peak Power	P_p	[KW]	26
Continuous Rating	P_c	[W]	3000
Heat Capacity	C_p	[kJ/(kg K)]	450
Mass	m	[kg]	10,6

Tabelle 2.5: Break Resistor Characteristics

second possibility is applied. Table 2.4 on page 15 and Table 2.5 on page 15 indicate the characteristics of the brake chopper and the resistor. Taken from manual [4] page 3-3.

Load Machine

The load machine is a squirrel-cage asynchronous servo motor with a rated power of 13,2 [kW]. Its nominal speed is 3510 RPM and it has a field weakening range up to a maximal speed of 8000 RPM. Protection for overheating is realized by an integrated KTY temperature sensor and the cooling is done by a separate driven fan. Table 2.6 at page 16. Taken from manual [5] page 12.

2.3.2 Usage of PLC S7 212

Main objectives:

- Control tasks
  - of the PC and software are operating
  - of Hardware operating
  - settings made by the PC and software are allowed

Servo Motor Characteristics

Servo Motor	Type MDFKA 100-22,120		
Axis height	h	[mm]	96
Speed (rated)	n_r	$[min^{-1}]$	3510
Torque (rated)	M_r	[Nm]	36.0
Power (rated)	P_r	[kW]	13.2
Voltage (rated)	V_{r3}	[V]	390
Current (rated)	I_r	[A]	28.7
Torque max	M_{max}	[Nm]	180
Speed max	n_{max}	$[min^{-1}]$	8000
Mass moment of inertia	J	$[kg \cdot cm^2]$	72
Weight	m	[kg]	48.2

Tabelle 2.6: Servo Motor Characteristics

Properties S7-212

SPS S7 212				*Expansion EM223*	
Memory		*Connectors*		*Connectors*	
EEPROM	512 words	Inputs	8 DC 24V	Inputs	8 DC 24V
User Data	512 words	Outputs	6 relays	Outputs	8 Relays
Internal Marker	128				

Tabelle 2.7: Properties PLC

- the hardware has carried out settings made by the PC and software

- start-up and shut-down tasks
  - Start up of all hardware components in a defined order (the measurement PC is not regarded)
  - Shut down of all hardware components in a defined order (the measurement PC is not regarded)

Properties and Possibilities of the PLC S7 212

Table 2.7 and figure 2.4 give a small review over the properties of the PLC S7-212 (S7). The stored program can link the values at the inputs and set the outputs accordingly. The programming language contains commands to define sequences, subprograms, boolean operations, delays, counters, and timers. For detailed information refer to the system guide of the S7 [13].

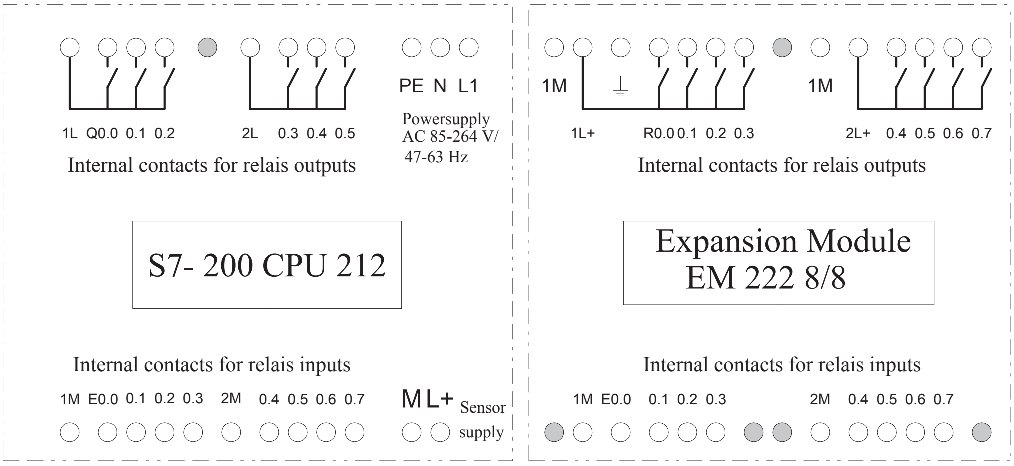


Abbildung 2.4: Terminal Diagram SPS S7 212

2.4 Test Object

Every electrical drive with power up to 22 kw and speed up to 10 000 RPM can be tested with the above described test bench. Problems which have to be solved are the physical adaptation of a changed mechanical coupling and also the adaptation of the supply of the test motor.

The Test-Stand was planned with a test object consisting of induction motor, an inverter, two batteries and a main supply and a charging unit. With this approach every part of a common drive system can be considered, all aspect concerning the power supply, a possible power conversion and the drive itself. A possible measurement concept with this mentioned test object is shown above in figure 1.1 on page 4.

The projects aim was the development of an extensible test stand. Extendability means also the ability to test different kind of electrical drives. It should be pointed out here that the Test-Stand can test more than induction engines but all kind of electrical drives.

Kapitel 3

LOLA– Low Observing Laboratory Application

3.1 Demands

The software controlling the test stand is called LOLA. This is a synonym for **L**ow **o**bserving **L**aboratory **A**pplication. LOLA handles communication between the user, the Test-Object, the Measurement-System and the Control-System. It enables the user to handle all the test stand's functionality from one place. Therefore it is an low observing application.

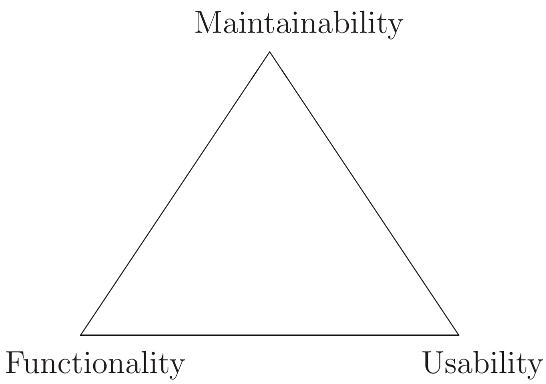


Abbildung 3.1: Demands

Figure 3.1 shows the area of conflict developing software for the proposed Test-Stand. There are three main objectives: Functionality, usability and maintainability. Functionality includes all possibilities of the system. Usability means how easy the system is to use. Maintainability means how easy it is to adapt and improve the system.

3.1.1 Functionality

Providing a Software with good functionality is not easy. In order to manage the size of LOLA its structure had to be planed carefully. Data sheets, flow sheets, drafts and name conventions for variables and subVIs helped to develop, improve, and adapt the software. Logically independent parts were programmed as independent modules to ensure separate testing and easy replacement by improved VIs.

LOLA's Functionality covers the following functions:

- Acquiring Data,
- Processing Data,
- Control of the Test-Stand,
- Well Designed User-Interface,
- Quality assurance.

Basically LOLA has to acquire data which result from signals at the used sensors. For our purpose a *signal* is defined as any physical quantity that varies with time [21]. The physical quantity has to be transformed into an electrical signal. The electrical signals are generated by sensors. Next the signal is standardized, sampled and quantized. The resulting values with according *Time-Stamps* are the *Measurement-Data* (see section 3.2.1).

The Measurement-Data is processed in the PC. Processing Measurement-Data includes transferring, rearranging, displaying and saving. A standardized data format for saving is essential for automated data analysis.

Information of every system part's actual state is important for LOLA to keep control of the Test-Stand. With this information LOLA is able to derive actions that ensure a secure operation. This information is called *Control-Data* (see section 3.2.2).

A well-designed User Interface is vital to an easy use of the program.

Quality assurance is an important issue in the world of industry. The reliability of data is one of the property of the Test-Stand (see chapter 6).

3.1.2 Maintainability

A good maintainability means that enhancing, improving and changing is easy. There are some cases of such:

- Investigating different Test-Objects.
- Expansion, replacement or removal of system components.
- Changing processing of Measurement-Data.

In order to meet the demands of maintainability LOLA is constructed in a modular way. If the Test-Object changes, parameters can be changed easily. A measurement device with a different system bus can also be added, replaced or removed easily.

3.1.3 Usability

The Usability focuses on things concerning the user. This means

- short time to learn,
- operational reliability,
- self-explaining user interface.

These mentioned facts are discussed more detailed in chapter 5.

3.2 Data

3.2.1 Measurement Data

Measurement-Data includes measured values. These values are completed with their time of acquisition the *Time-Stamp* (see chapter 6). Time-Stamp is the difference between PC-system-time on start of LOLA and PC-system-time at time of acquisition. Measurement-Data is organized in two-dimensional arrays. First column contains Time-Stamp and the further columns measurement values. An array of column-names declared in the settings is used to identify the columns (see chapter 4).

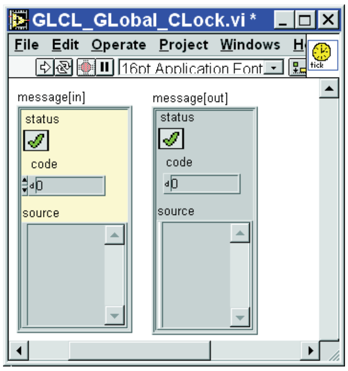
3.2.2 Control Data

As mentioned above LOLA needs information to control the Test-Stand. This information together with the derived commands for the Test-Stand is the Control-Data. It includes the state of every subVI and the state of the Measurement- and Control-System. The structure of Control-Data is described more detailed in chapter 4.

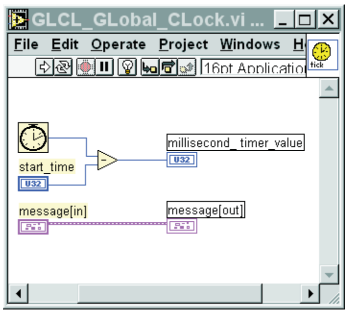
3.3 Basics of LabView

3.3.1 Introduction to G

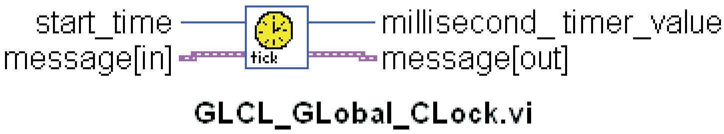
LOLA is developed with LabView. Therefore a short introduction to LabView is given in this section.



(a) Front Panel



(b) Block Diagram



(c) Connector Pane

Abbildung 3.2: Make-up of a LabView Application

LabView, with its programming language G, is a fully featured programming language produced by National Instruments. It is a general-purpose graphical programming language quite different to popular text-based programming languages. There is no text based code as such, but a diagrammatic view of data flow through the program.

G programs are called virtual instruments (VIs) which are similar to functions of conventional programming languages.

A VI consists of an interactive user interface, the *front panel*, a dataflow diagram, the *block diagram* and icon connections, the *connector pane*. The make-up is visualized at Figure 3.2 on page and subsequently described more detailed. Expressions are taken from G Reference Manual [15].

- The *front panel* contains controls and indicators like buttons, graphs, etc. Its appearance was designed to simulate real instruments. See figure 3.2(a).
- The *block diagram* is the pictorial solution to the programming problem. A VI handles data according to its block diagram. See figure 3.2(b). There are objects and connections. The objects can be controls, operators or subVIs. Data ‘flows’ along connections. Objects are sequentially executed the way data flows.
- The *connector pane* is a symbol for the VI. It can have inputs and outputs called connectors. Via these connectors several VI’s can interchange data. Its usage could be described as a graphical parameter list. See figure 3.2(c).

For further information see [14, 16, 17] and [15] all manuals provided by National Instruments (`www.ni.com/manuals`) especially 'LabView, User Manual' [14] is recommended.

3.3.2 Independent Data Exchange

This section will introduce possibilities of exchanging data between subVIs running parallel at the same time. In LabView there are three such possibilities:

1. *Global Variables* can be accessed in every part of an LabView application. They can not be adapted for special purposes. Debugging is difficult.

2. *Direct Data Exchange (DDE) or Transmission Control Protocol / Internet Protocol (TCP/IP)*: This system establishes a client-server structure. To ensure a lossless data transfer several control structures have to be added. Furthermore there is big overhead.

3. *Temporary-Storages* are non-reentrant subVIs which can theoretically be called everywhere in the program. In LOLA each of them is called twice: Once for writing and once for reading. So the connection is a point to point connection. The detailed construction can be found in chapter 4.

   Originally this method had to be used with LabView 4 as there were no global variables.

LOLA uses the third possibility. The advantage of the Temporary-Storages is that many different values can be buffered with one call, the storages can be easily adapted for different purposes and they can even get a kind of "intelligence".

3.4 Structure

3.4.1 Principle

In principle the main program LOLA is a common VI containing several subVIs. The differences between a workaday G solution and LOLA are first the grown complex structure, second the claim of an well designed user interface, third a secure operation for the user and the equipment, and fourth the independent execution of the subVIs while maintaining a steady data flow to the hard-disk and the user interface.

This last difference is mainly dictating the structure of LOLA. On the one hand LOLA has to process data from different locations on different scan rates using multi-threading capability of Microsoft Windows NT^{TM}, on the other hand LOLA has to send the gathered data to two different sinks – the userinterface and the hard-disk. If data is not processed independently one source has to wait for the other. Different

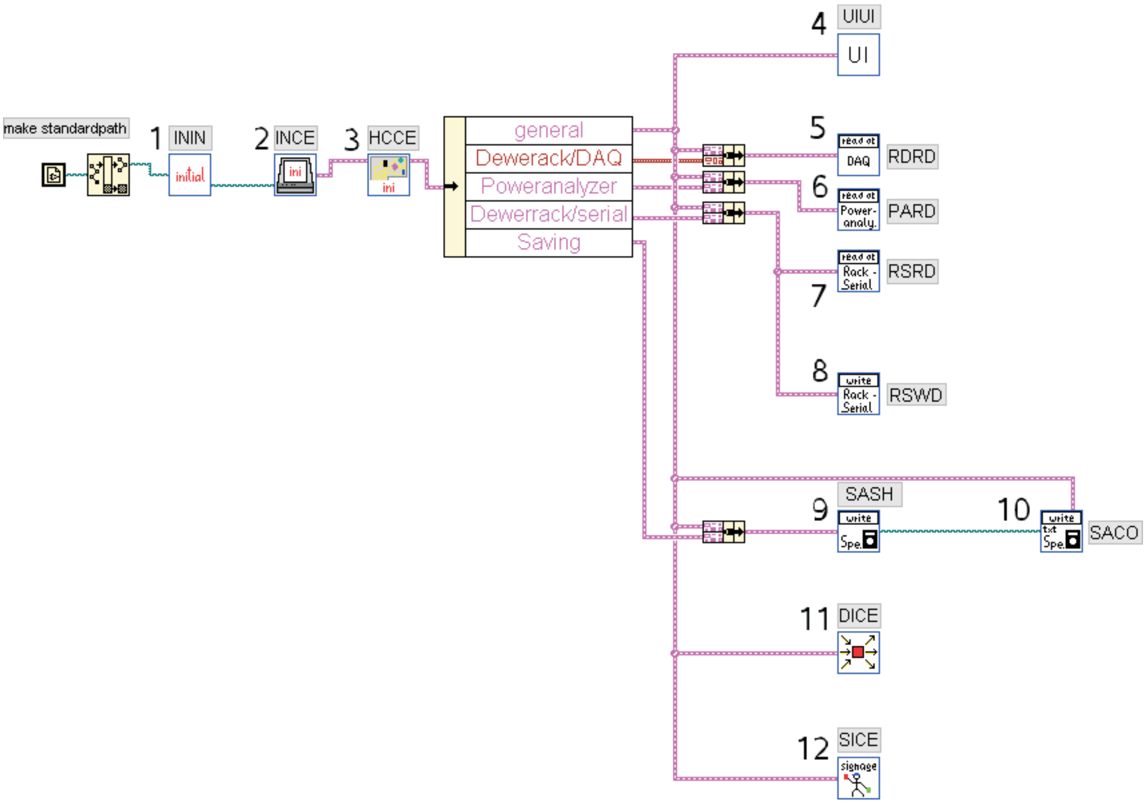


Abbildung 3.3: LOLA: Diagram of Main Program

scan rates on different location can easily produce data jams like buffer overflow and resulting loss of data.

Each communication connection to a measurement device is implemented by an independent subVI. This subVI holds the driver and transfers the data to a buffer. This buffer is the *Temporary-Storage*. It stores the data until the *Distribution-Center* reads it out.

The Distribution-Center's duty is to collect all Measurement-Data of the system by reading out the Temporary-Storages and distributes it to target Temporary-Storages for further processing. Further processing means displaying for the user, saving on hard-disk and controlling the data. Detailed description is given in chapter 4. SubVIs for generating- and processing data are called *Modules*.

3.4.2 Parts

The main programm contains all main parts (VIs) of the program see figure 3.3 on page 23.

Initialization subVIs

The VIs 1,2 and 3 are for initializing the system: (1) ININ_INitialization_INitialization.vi (ININ) resets all parts of LOLA. (2) INCE_INitialization_CEnter.vi (INCE) enables the user to change settings all parts of LOLA. (3) HCCE_Hardware_Configuration_CEnter.vi (HCCE) starts the Measurement- and Control-System.

Control SubVIs

Measurement- and Control-Data distribution from their sources to their predefined sinks is performed by (10) DICE_DIstribution_CEnter.vi (DICE). The control of the system's status is done by (11) SICE_SIgnage_CEnter.vi (SICE).

Data Read SubVIs

(5) RDRD_RackDAQ_ReadData.vi gets measurement data from Dewerack using the DAQ-board. (6) PARD_PowerAnalyzer_ReadData.vi gets measurement data from LEM Poweranalyzer. (7) RSRD_RackSerial_ReadData.vi gets measurement data from Dewerack using RS232-interface.

Data Write SubVIs

(8) RSWS_RackSerial_WriteS7.vi sends Commands to the PLC S7 S212.

Processing SubVIs

(9) SASH_SAve_SHot.vi saves Data on demand. (4) UIUL_UserInterface_UserInterface.vi enables the user to interact with the system. (12) SACO_SAve_COnversion.vi (SACO) is executed after the data acquisition has been terminated. It converts the saved data file from the LabView intern format to ASCII-text format

The following chapters include more detailed information on the Modules.

3.4.3 Summary

LOLA contains a main VI with several subVIs called Modules, the diagram is shown in 3.3. Apart from the VIs (1,2,3) and (10) the modules execute parallel without waiting for data from any other VI. The steady dataflow is guaranteed by the Distribution-Center in concert with Temporary-Storages.

Kapitel 4

Data Processing

As mentioned in the previous chapter there is Measurement- an Control-Data. How these data is composed and transported between the Modules is explained in this chapter.

Figure 4.1 on page 26 gives an overview of the connection of the parallel running Modules while operation. These can be sorted into groups:

- Data Read subVIs
- Data Write subVIs
- Control subVIs
- Processing subVIs

4.1 Measurement-Data

Measurement-Data is organized in two-dimensional arrays. In LOLA there are three sources for Measurement- Data the VIs (5) RDRD, (6) PARD, (7) RSRD (see figure 4.1 on page 26). For every source a two-dimensional array is created. First column contains time and the further columns measurement values. The number of columns with measurement values depends on the number and kind of commands set in the user setting while the initialization. The header of each column is also declared in the settings while the initialization (see chapter 5). For example the in table 4.1 listed settings were used while testing LOLA.

4.2 Control-Data

Before discussing the way Control-Data is processed, the structure of Control-Data and the error handling in LabView in general have to be explained.

In LabView standard error clusters and *General Error Handler* (GEH) are used to cope with errors which may occur in executing VIs. Standard error cluster consists of *Status* (boolean), *Code* (large integer) and *Source* (string).

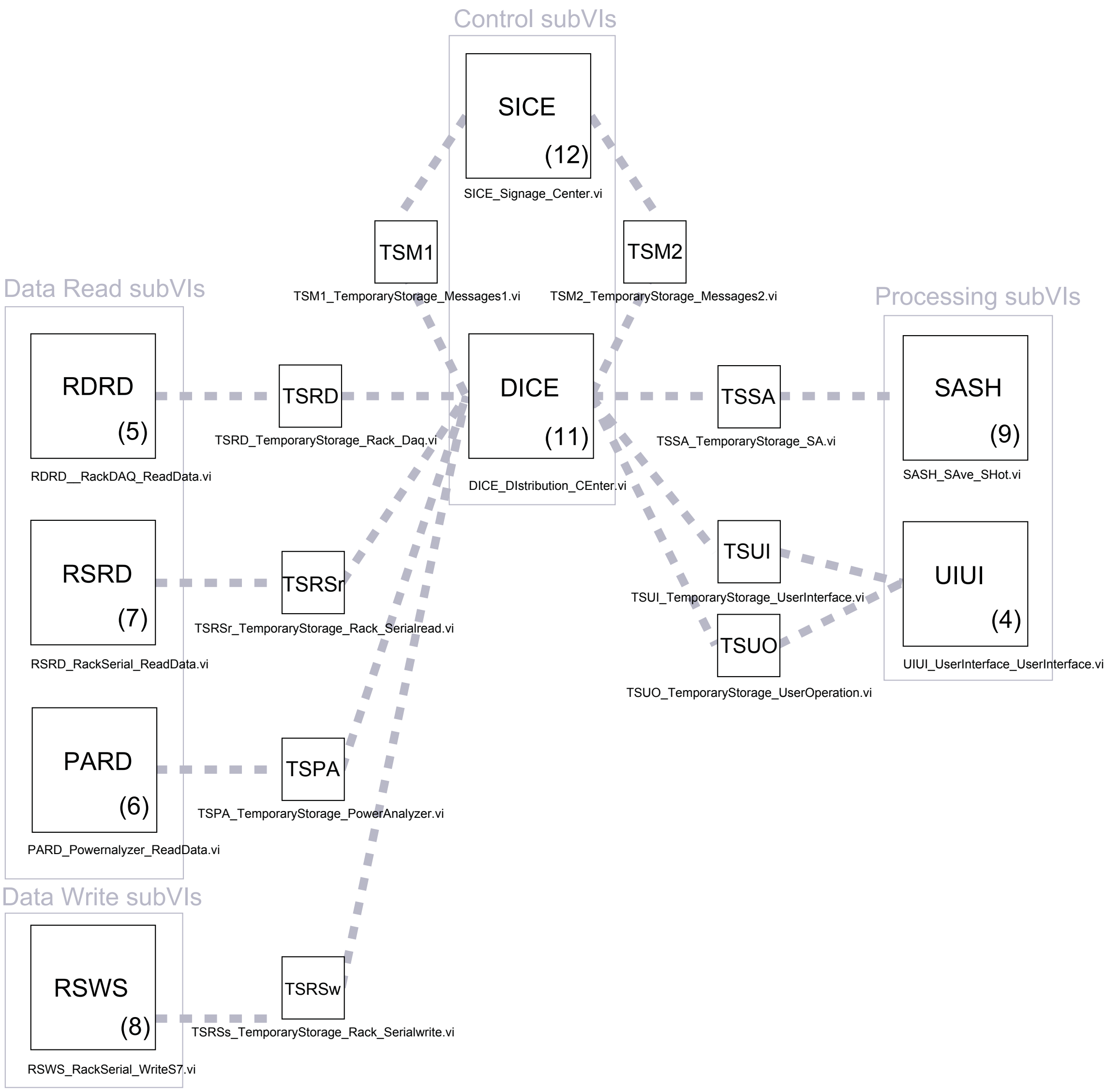


Abbildung 4.1: Synopsis of DDS and MGS
Numbers in brackets according to figure 3.3 on page 23.

PowerAnalyzer_ReadData										
	L1			L2			L3			All
time	V_1	I_1	f_1	V_2	I_2	f_2	V_2	I_3	f_3	φ
...	...	...	...	...	...	...	...	...	...	...
RackSerial_ReadData										
	K- Elements							Infrared Sens.		
time	ϑ_{ke0}	ϑ_{ke1}	ϑ_{ke2}	ϑ_{ke3}	ϑ_{ke4}	ϑ_{ke5}	ϑ_{ke6}	ϑ_{is_0}	ϑ_{is_1}	
...	...	...	...	...	...	...	...	...	...	
RackDaq_ReadData										
	Metering Shaft									
time	n	T_{Torque}								
...	...									

Tabelle 4.1: Measured Values

V ... voltage, I ... current, f ... frequency,
φ ... power-factor angle, n ... speed, T ... torque.

The standardized error handling differs between No Error, Warning and Error. In case of No Error (default) Status is False, Code and Source are empty. If a warning occurs Status is unchanged, but code is set according to the warning and the name of the VI is written to source. If an error occurs in the executing VI Status is set to TRUE, Code is set according to the occurred error and the name of the VI is written to Source.

In LabView there are two ways to get error or warning information about the execution of the running VI. The first way is to wire the error-connector of a LabView-standard VI with a GEH. The GEH stops the execution of the VI and replaces the string of Source with an explanation of the occurred error and displays it via a pop-up menu. The second way to get error or warning information is to define user errors. Own routines have to decide wether the error is triggered or not. The codes and explanations of the user-defined error codes have to be added to the GEH codes. LOLA uses both ways.

Unfortunately the standardized error handling is too strict to implement the sum of Control-Data. To circumvent these restrictions *Messages* and SICE_SInage_Center.vi are introduced to LOLA. Messages represent all Control-Data needed in LOLA and SICE is the core to process them. In general Messages are LabView standard error clusters. The exception are Reports.

Messages in turn can be Condition, Command, Confirmation, Log, and Report. Messages have the following tasks:

- *Condition*: Information about the condition of each Module, can be a Warning or an Error.
- *Command*: Information sent to modules to control their behaviour. The mea-

ning can be: termination, halt for the the Control-System, or save-operation to name only a few.

- *Confirmation*: An answer of each module to a given *Command*.
- *Log*: Information consisting of concentrated and reformatted Conditions. This information is saved into a log-file and displayed at the userinterface in the Messages-Panel.
- *Report*: All information generated in the modules except the Measurement-Data. Its data format does not allow an integration with the standard error cluster but they are also used to control and monitor status of the Test-Stand.

4.3 Data Delivery System

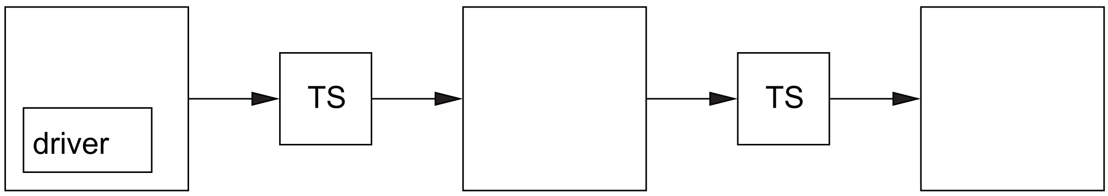


Abbildung 4.2: Data Delivery System (DDS)

In order to make Measurement-Data processing possible some communication links between the participating parts of the system have to be established. The way this is done is called *Data Delivery System (DDS)*. In principle the DDS ensures that Measurement-Data gets from the driver-subVI to the process subVIs without affecting the independent execution of the VIs concerned.

The physical links have been described in chapter 2. Figure 4.2 shows the principle of the software solution of the communication link. It shows a block diagram of two exemplary Modules and their communication. Driver-subVIs do the communication between hardware and software which is described more detailed in Chapter 7. *Temporary-Storages* (TS) buffer the Measurement-Data so that other parts get it independently. The *Distribution-Center* collects and distributes the data. These two steps will be explained later on. The process-subVIs display, save and control the Measurement-Data. They will be described in the chapters 5 and 6.

4.4 Message Gathering System

Parallel to the DDS a *Message Gathering System (MGS)* is implemented to gather all relevant Messages.

The way Messages are collected and distributed can be seen in figure 4.3. The MGS ensures a proper communication between the Modules of LOLA. It owns all functions of a bus-system except data transfer which is realized by the DDS as said above.

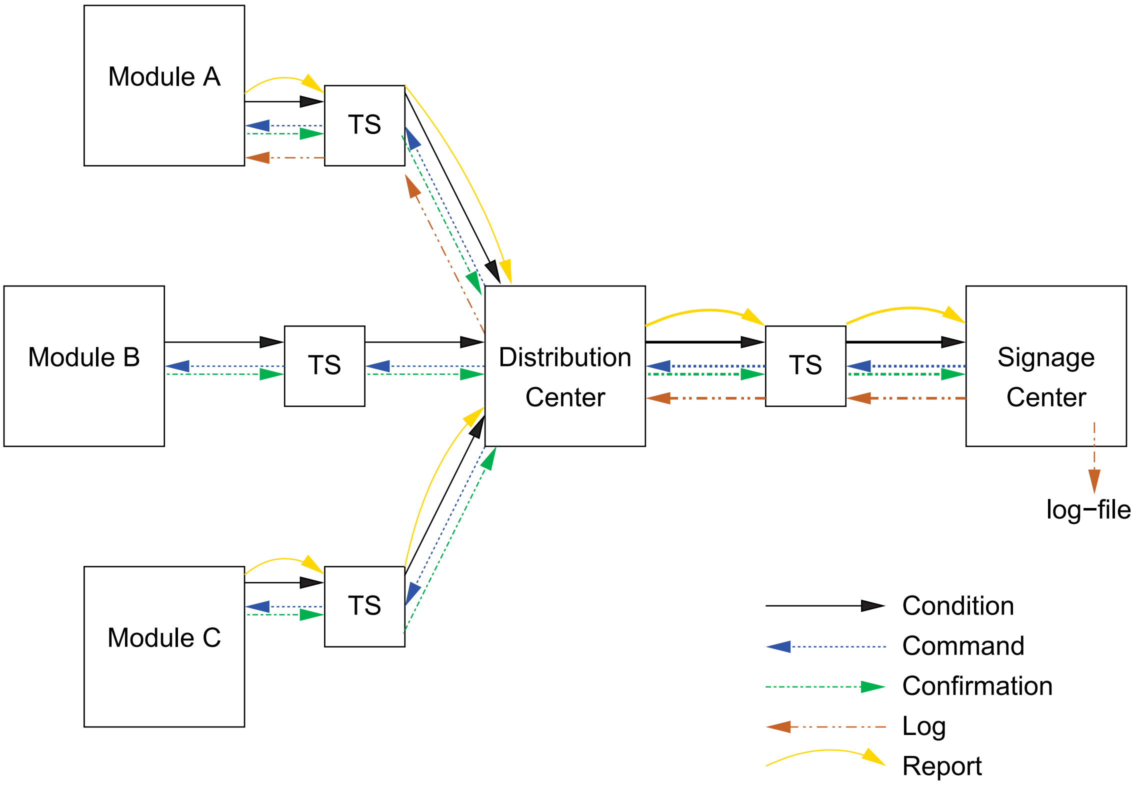


Abbildung 4.3: Message Gathering System (MGS)

The Modules have to notify their Condition so the system can react accordingly. Conditions from all Modules are passed on through Temporary-Storages to the Distribution-Center. The collected Conditions are forwarded to the Signage-Center. There proper reactions have to be initiated (See Section 4.5). For the fact that Reports are different formatted Conditions the same procedures are applied.

The proper reaction results in Commands. These are sent to the Modules via Temporary-Storages and Distribution-Center. In each subVI rules are defined to react to these Commands. After executing a Command the Module generates a Confirmation.

The Confirmation is sent back so that the Signage-Center knows whether the Commands were executed properly or not.

Additionally there is Log. Log are all conditions and reports reformatted for the user. It is displayed at the User-Interface and stored to a log-file. This log-file is usually called log.lok, located in subdirectory `misc`.

4.5 Important Parts of DDS and MGS

4.5.1 Temporary-Storages

Temporary-Storages are part of every Module except for Initialization-Modules. Basically Temporary-Storages consist of a while loop with a shift register. The while loop is executed only once and data is stored in the shift register. The stored data can be read again later. In some of the Temporary-Storages there are also If-Then instructions included. This means that some Temporary-Storages decide, following the terms of the If-Then instructions, whether new data is added to the referring storage or not.

So Temporary-Storages are buffers, which can be either read or written. In terms of Measurement-Data *read* means take the buffered Measurement-Data and emptying the buffer. *Write* means adding measurement data to the buffer. Table 4.2 shows a list of existing Temporary-Storages in LOLA and which modules they connect.

Temporary Storage	*Communication between DICE and*
TSM1_TemporaryStorage_Messages1.vi	SICE_SInage_CEnter.vi
TSM2_TemporaryStorage_Messages2.vi	SICE_SInage_CEnter.vi
TSPA_TemporaryStorage_PowerAnalyzer.vi	PARD_PowerAnalyzer_ReadData.vi
TSRD_TemporaryStorage_Rack_Daq.vi	RDRD_RackDaq_REadData.vi
TSRSr_TemporaryStorage_Rack_Serialread.vi	RSRD_RackSerial_ReadData.vi
TSRSw_TemporaryStorage_Rack_Serialwrite.vi	RSWS_RackSerial_WriteS7.vi
TSSA_TemporaryStorage_SAve.vi	SASH_SAveSHot.vi
TSUI_TemporaryStorage_UserInterface.vi	UIUI_UserInterface_UserInterface.vi
TSUO_TemporaryStorage_UserOperation.vi	UIUI_UserInterface_UserInterface.vi

Tabelle 4.2: List of Temporary-Storages

Messages are handled differently compared to Measurement-Data. Which Temporary-Storage handles which kind of message as well as the action done can be seen in table 4.3 on page 34.

4.5.2 DICE_DIstribution_CEnter.vi

The *DICE_Distribution_CEnter.vi (DICE)* collects and distributes data via Temporary-Storages listed in table 4.2.

In figure 4.4 the Block Diagram of DICE_DIstribution_CEnter.vi can be seen. As usual data flows from left to right. DICE does the following:

- reading data from Temporary-Storages;
- organize Messages in order to
  - terminate zero Messages (code=0, status=FALSE),
  - terminate double Messages so that each Message appears only once,

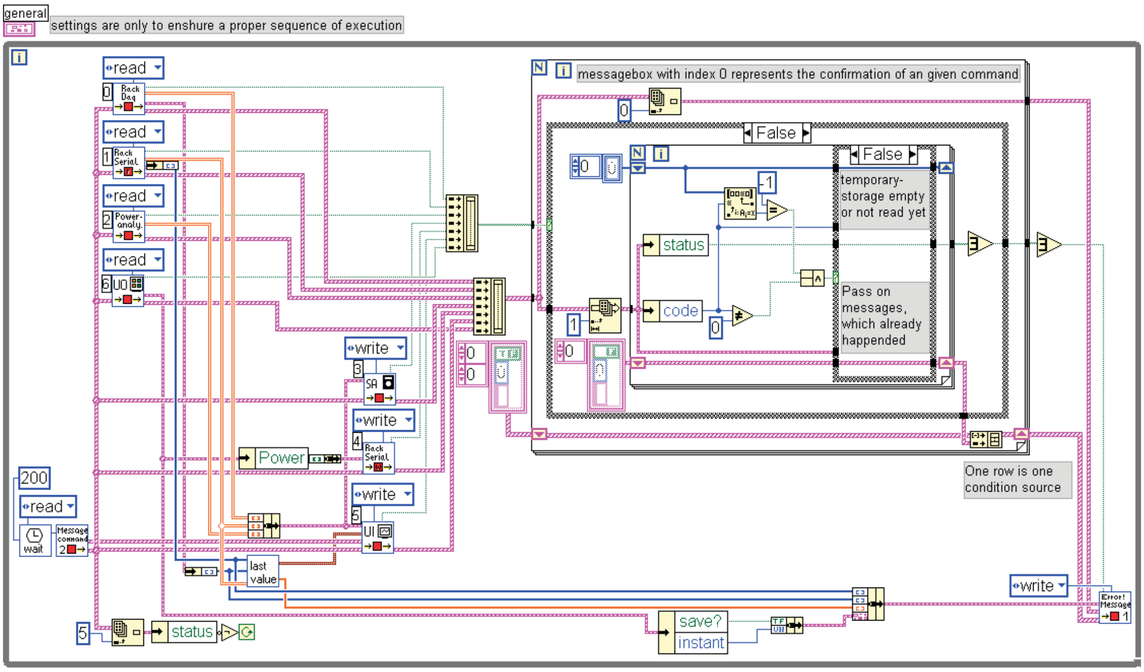


Abbildung 4.4: Block Diagram DICE

- sort message so that all modules get the data and only the data they need for proper operation;

- write data to Temporary-Storages.

The detailed sequence can be seen in Flowchart DICE in appendix A.

4.5.3 SICE_Sinage_Center.vi

The core of the software control implemented in LOLA is the subVI *SICE_SInage_-CEnter.vi (SICE)*. Its block diagram is shown in figure 4.5 and the detailed sequence can be seen in Flowchart SICE in appendix A.

As explained above in section 4.4 this VI gathers all relevant messages (Conditions, Confirmations and Reports). Those are sent by DICE via the Temporary-Storage TSM1_Temporary_Storage_Messages1.vi. SICE has four objectives:

- derive Commands from Conditions,

- derive Commands from actual reports,

- check if Commands are accomplished,

- save Logs in log.lok for debugging.

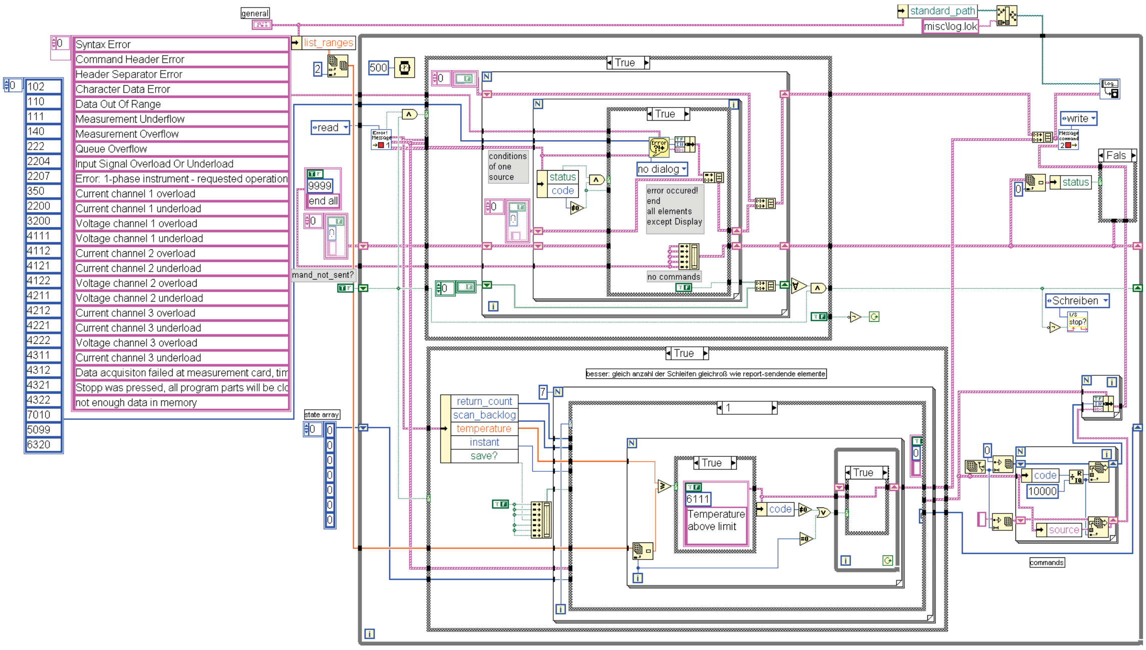


Abbildung 4.5: Block Diagram SICE

Condition Evaluation

The idea of SICE was to create a special Command for related Conditions. Therefore SICE has to recognize the Condition and derive the proper Command. The recognition of a Condition is realized by a unique Code and its Status. This action is carried out by one adapted GEH. It uses the LabView internal error codes listed in the online help of the program and the user-defined error codes, which are listed and explained more detailed in table C.1 in chapter 8.

In this stage of the program SICE only distinguishes between the main two groups of conditions, Errors and Warnings. If any error occurs, the termination Command is sent to all operating subVIs. The hardware is sent into the wait-state and the operator is informed of the program termination and the reason why. If any warning occurs it is only reformatted and displayed as log so the user is informed but the program continues. The structure (1) in the diagram is executing the Condition evaluation.

Report Evaluation

For the fact that Reports are system information in other format than the error-cluster the course of action is the same as Condition evaluation. All reports have to be evaluated and a proper reaction has to be defined. The difference between Condition evaluation and Report evaluation is that no GEH is available. Therefore a check routine is implemented for each different kind of Report.

Structure (2) realizes the check routine. It can be described as state machine. Every

Module (e.g. UIUI, RDRD, RSRD ...) has its own states and their state variable defines the actual state. The states are realized as encapsulated case structures and the state variables are stored in a shift register. During every execution of the while-loop all reports are checked. This can cause a change of the state of the referring Module. The change of state results in a Command or at least in a Warning.

Commands

At this stage only the termination command is explained. The number and kind of the other commands used in LOLA is described more detailed in chapter 8.

If an executing Module receives the termination command it stops the execution of the main while loop and executes possible remaining operations located at the right side of the while loop (data flows from the left to the right). The structure which enables the Module to recognize the termination signal is the subVI GLCM_GLobal_CheckMessages.vi. This subVI looks in the array of incoming messages for the termination command. Is it found, the output *stop?* is set to TRUE. This value is sent to the termination operator of the main while loop.

Confirmation

Once a Command is sent, SICE has to check if the Command was executed by the target-VI(s). If the Command is executed by the target-Module, it will send back the same Command as Confirmation. While waiting for the Confirmation SICE stops sending new Commands. An exception of this rule is the termination Command. This Command has the highest priority and can be sent at any time.

Log saving

After the incoming Messages were evaluated a short version Log of the Condition and Reports is saved to hard disk and sent to the User-Interface. This short version consists of the name of the Condition or exceptional Report and the Location of Occurrence.

Action	TSM1	TSM2	TSPA	TSRD	TSRSr	TSRSw	TSSA	TSUI	TSUO
READ									
measurement_data_array[out], *report_array[out]* are read and the referring storage emptied.			√	√	√		√		
measurement_data_array[out], *log_array[out]* are read and the referring storage emptied.								√	
message_array[out] is read and replaced by *message_array[in]*.			√	√	√				√
Command_array[out] and *log_array[out]* are read.		√							
confirmation_array[out] and *condition_array[out]* are read and the referring storage emptied.	√								
report[out]* is read and the referring storage emptied.	√					√			√
if *not_read?* = FALSE and *message_array[out]* contains no error then *message_array[out]* is replaced by *message_array[in]*.						√	√		
if *not_read?* = TRUE then *message_array[out]* is read and replaced by *message_array[in]*.								√	
empty? := TRUE.			√	√	√				√
new_messages? := FALSE.	√								
not_read? := FALSE.						√	√	√	
WRITE									
measurement_data_array[in] is added to *measurement_data_array[out]*			√	√	√		√	√	
message_array[out] is read and replaced by *message_array[in]*.						√	√	√	
Command_array[out] and *log_array[out]* are replaced by *Command_array[in]* and *log_array[in]*.		√							
confirmation_array[in] and *condition_array[in]* are added to *confirmation_array[out]* and *condition_array[out]*.	√								
report[in]* is added to *report*[out]*	√			√	√	√	√		
log_array[in] is added to *log_array[out]*								√	
if last *reports[in].useroperation.save* = FALSE then *reports[in].useroperation.save* is added to *reports[out]*	√								
if last *reports[in].useroperation.save* = TRUE then *reports[out].useroperation* is replaced by *reports[in].useroperation* else the referring storage is emptied.									√
if *new_messages?[out]* = FALSE then *new_messages?[out]* is replaced by *new_messages?[in]*.	√								
if *empty?* = FALSE and *message_array[out]* contains no error then *message_array[out]* is replaced by *message_array[in]*.			√	√	√				√
empty? := FALSE.			√	√	√				√
not_read? := TRUE.						√	√	√	

Tabelle 4.3: Functionality of Temporary-Strorages

Kapitel 5

User Interaction

This chapter copes with aspects of the User-Interface, design strategies and their implementation. A spot on Strategies for effective human-computer interaction and ideas that mainly affected the design of the User-Interface is included (see [20]). The LabView VIs are described and results of a first evaluation is added.

5.1 Design Strategies

5.1.1 Object-Action Interface Model

The "Object-Action Interface Model (OAI)" (see [20] page 61ff) is a very helpful tool for structuring the design process. User-interface developing with OAI starts with understanding the users tasks. They include the universe of real-world objects with which users work to accomplish their intentions and actions that they want to apply to those objects. Once these tasks are identified the metaphoric representations of the interface objects and actions can be created.

LOLA's User-Interface was basically created using the main ideas of the OAI.

5.1.2 The Design Process

Development of a simple and clear user-interface demands several stages of work:

1. The first step is to specify tasks and subjects that must be carried out.
2. A second step is ensuring system reliability. This means that the system works in a proper way.
3. The third step is ensuring consistency. This leads to common action sequences, terms, units, layouts, color, typography and so on.

4. In [20] a fourth step is mentioned: Schedules and Budgets which is not considered within this work.

If all these steps are done sufficiently the attention can be focused on the design and testing process.

For a proper design the user has to be characterized as precise and complete as possible. According to this characterization the style of interaction and the actions and objects can be created.

5.1.3 Golden Rules of Interface Design

Shneiderman gives “Eight Golden Rules of Interface Design” (see [20] page 74f). Five of them are vital to the basic ideas of LOLA’s User-Interface:

1. *Strive for consistency.* “This rule is the most frequently violated one, but following it can be tricky because there are many forms of consistency. Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus and help screens; and consistent color, layout, capitalization, fonts, and so on should be employed throughout. Exceptions, such as no echoing of passwords or confirmation of the delete command, should be comprehensible and limited in number.”
2. *Offer informative feedback.* “For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. Visual presentation of the objects of interest provides a convenient environment for showing changes explicitly.”
3. *Offer error prevention and simple error handling.* “As much as possible, design the system such that users cannot make a serious error; for example, prefer menu selection to form filling and do not allow alphabetic characters in numeric entry fields. If users make an error, the system should detect the error and offer simple, constructive and specific instructions for recovery. Erroneous actions should leave the system state unchanged, or the system should give instructions about restoring the state.
4. *Support internal locus of control.* Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Surprising system actions, tedious sequences of data entries inability or difficulty in obtaining necessary information and inability to produce the action desired all build anxiety and dissatisfaction.”
5. *Reduce short-term memory load.* “The limitation of human information processing in short-term memory (the rule of thumb is that humans can remember “seven-plus or minus-two chunks” of information) requires that displays be kept

simple, multiple page displays be consolidated, window motion frequency be reduced and sufficient training time be allotted for codes, mnemonics and sequences of actions. Where appropriate, online access to command-syntax forms, abbreviations, codes and other information should be provided."

5.2 Implementation

This section describes the basic ideas of LOLA's User-Interface taking into account what was mentioned above. LOLA enables the user to interact with the system twice. First for initialization. Second for operation. Both use direct manipulation interaction style (see [20]). This means that a visual representation of the real world is presented on screen.

In the following sections the implementation of the above mentioned ideas is explained. Objects and actions are developed from task according to OAI. System reliability is discussed in the previous chapters. Intensive efforts were made to strive for consistency. Further improvement can be done if necessary.

5.2.1 Characterization of the User

The User of LOLA and the Test-Stand, has to be considered as an engineer that knows the basics of LabView and therefore has well understood the principle of LOLA's implementation (DDS, MGS, ...). The user knows the functionality of all devices an their communication, basic principles of digital signal processing and is capable of the English language.

5.2.2 User-Interface for Initialization

Tasks

The users tasks are: changing settings, saving and loading settings to and from hard-disk and telling the system being ready.

Objects

The whole Test-Stand is the main object. So there are settings for nearly every part. The user-Interface for initialization shows a structure of the Test-Stands parts. Each part is an object. Objects are combined to groups indicated with a box.

Furthermore there are two buttons for saving and loading settings and one for getting to the User-Interface for Operation.

The User-Interface for initialization is programmed in *INCE_INitialization_CEnter.vi (INCE)* (see section 5.3.1).

Actions

Moving the cursor on a group activates lines representing “communication connections” between the referring objects (see figure 5.1 on page 39). Moving the cursors on an object and pressing the left mouse-button activates a popup menu. In this menu the user can change referring object’s settings.

Additionally there is an Status-Panel which displays the actual settings of the object as well as the current `.ini`-file.

For more experienced users there is a object called “expert settings”. Clicking on it opens a pop-up menu with all settings possible.

As there was not enough time yet to test this user-interface for a extended practical use most pop-up menus are not complete. A frequency-of-use-classification of each setting is necessary to decide whether it is only an expert setting or not.

5.2.3 User-Interface for Operation

Tasks

The user wants to explore the characteristics of a Test-Object. Therefore the state of the system can be change by changing rotary speed or torque. LOLA supports the user to save measured quantities for different states of the Test-Object to hard-disk. Further analysis can be done with pertinent data analysis software.

This main task can be split into several sub-tasks as listed in the following:

- Trigger the save-action.
- Set speed or torque at load-inverter.
- Turn on and off devices (supply, load- and test-inverter, ...).
- Watch quantities on Graph-Panel and digital display.
- Set color of quantity graphs.
- Change graph-properties (zoom in and out, ...).
- Control warning- and alarm-situations.
- Watch Messages.
- Find out where warning- and alarm-levels are exceeded.

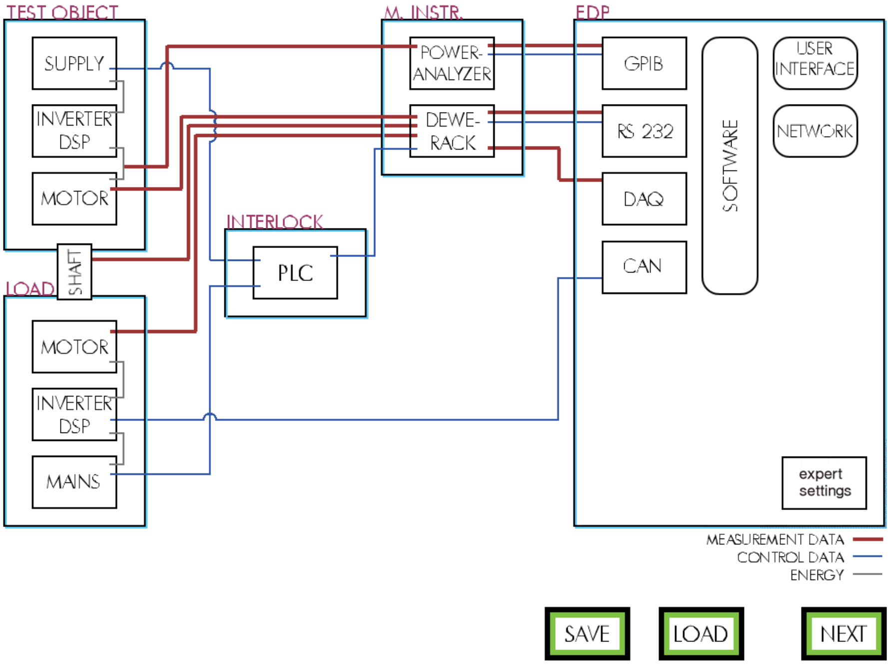


Abbildung 5.1: Front Panel INCE_INitialization_CEnter.vi

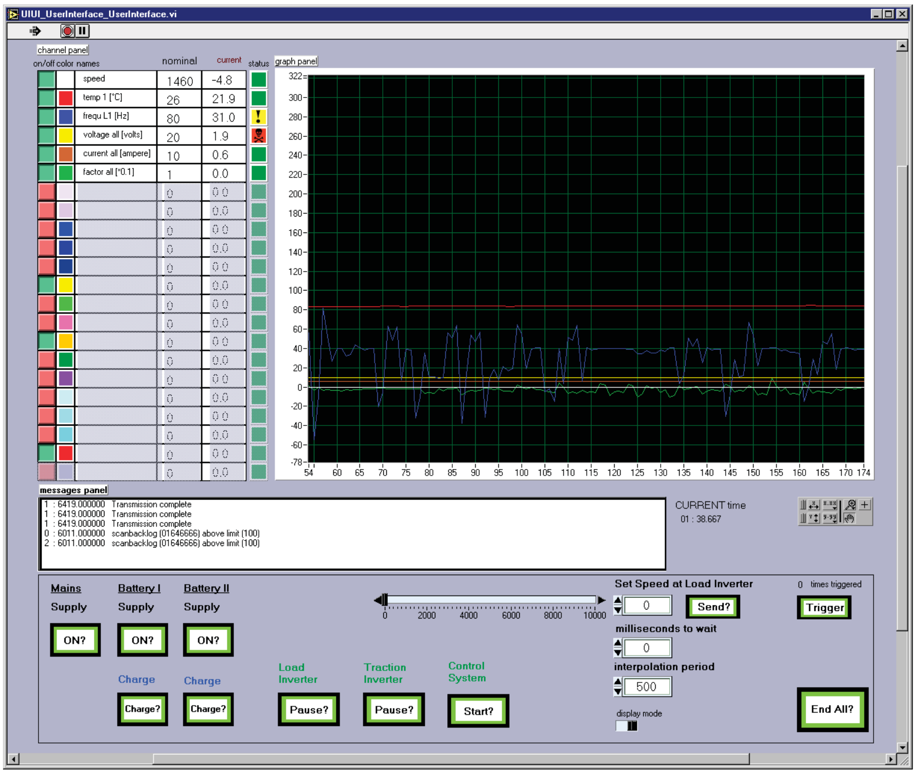


Abbildung 5.2: Front Panel UIUI_UserInterface_UserInterface.vi

Objects

Control facilities generally consist of objects like switches, numeric displays, and so on. With LabView these objects are easily realized on screen. The user can click on objects with a pointing device. So desired actions like turning a device on and off or change the color of a certain graph may be performed.

The User-Interface for Operation is divided into *panels* (see figure 5.2 on page 39). Each panel consists of objects. The panels are described in the following:

Graph-Panel The Graph-Panel consists of one single object: a two-dimensional LabView graph. This graph displays measurement data graphs. One for each quantity.

Channel-Panel The Channel-Panel lists all quantities by name. The lines of the list are called *channels*. Each line consist of: A control to turn the graph on and off, a control to select the color of the graph, the graphs name and unit, the nominal value, the current value and an indicator indicating whether an alarm- or warning-level is exceeded. The alarm and warning-level indicator is normally green and changes to yellow if the warning- and red if the alarm-level is exceeded.

Messages-Panel The Messages-Panel consists of a text-field for displaying messages.

Control-Panel The Control-Panel consists of functions the user needs to control the Test-Stand. There are buttons, switches and bars.

Status-Panel The Status-Panel is a stylized picture of the Test-Bench and Test-Object with squares to indicate each part's status.

Actions

Graph-Panel In order to bring data from different devices with different scan-rate into one graph the data is transformed: The target is to get measurement data with values from all quantities connected to one time. In addition a constant time gap between two adjacent values is . This is done via linear interpolation of all incoming data-streams. So the time axes does not need to be the same scale.

Figure 5.3 shows how the interpolation is done. The axis of abscissae is the time-axis and the ordinate is the value-axis of one measured point. The area under the new displayed points (red line) has the same size as the area under the interpolated original measured points (blue line) between $(t_a|y_a)$ and $(t_b|y_b)$.

The graphs represent relative values in percent (%) of the rated values.

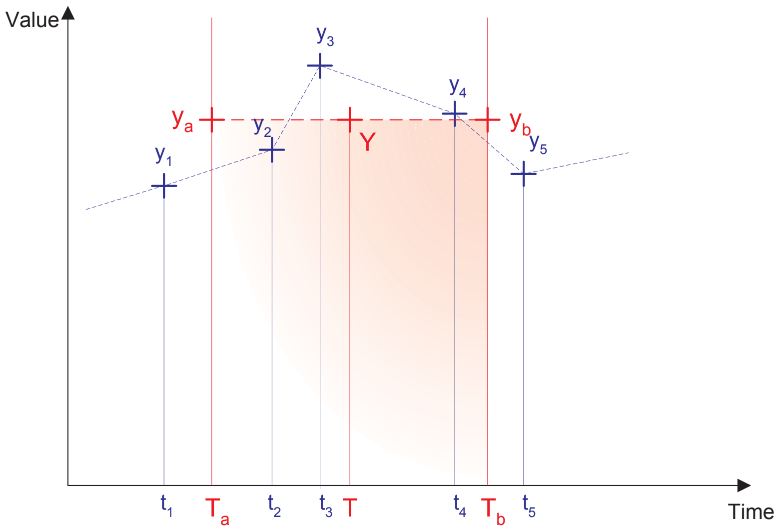


Abbildung 5.3: Interpolation for Graph-Panel
$(t_{1...n}|y_{1...n})$ ... original measured point, $(T|Y)$ ... average new point, $(t_{a,b}|y_{a,b})$ ... new displayed points

Channel-Panel The Channel-Panel enables the user to turn on and off graphs, watch digital values of each channel, watch warning and alarm-levels and change graph colors.

Messages-Panel Messages-Panel displays the content of Log [1]. This informs the user about the systems status.

Control-Panel The Control-Panel enables the user to control the system. The following actions can be done:

- Program operation can be stopped. When LOLA is fully stopped an pop-up menu appears to inform the user about the stop.
- System parameters as load characteristics (T,n) can be changed.
- Devices can be turned on and off.
- Saving of quantities can be triggered.

Status-Panel Status-Panel indicates the location of warning- and alarm-level exceed.

[1]See chapter 4.

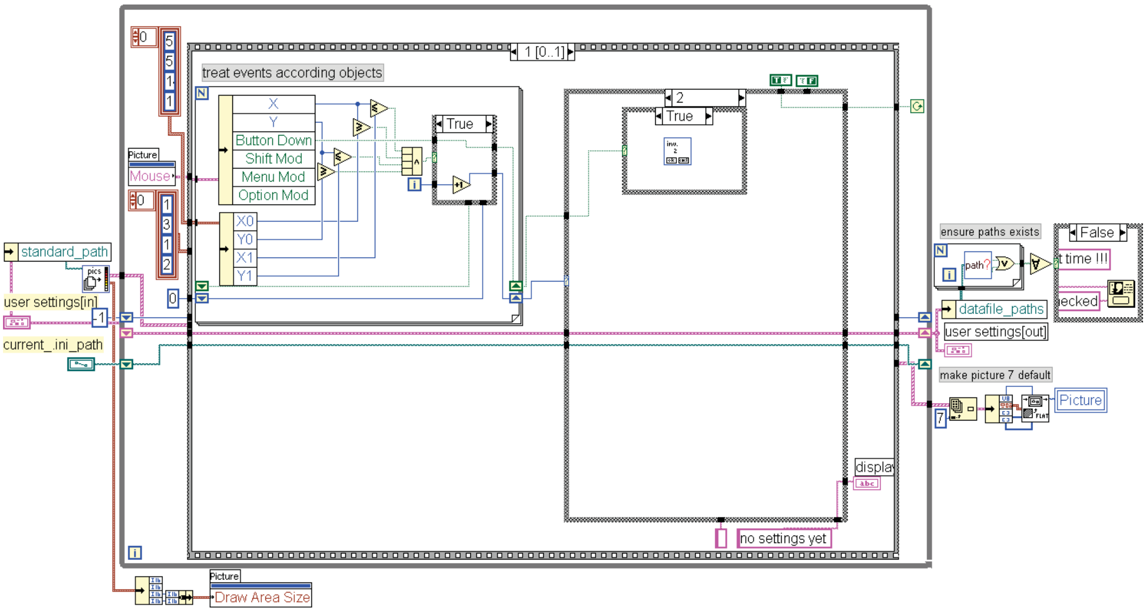


Abbildung 5.4: Block Diagram INCE_INitialization_CEnter.vi

5.3 LabView VIs

5.3.1 INCE_INitialization_CEnter.vi

Figure 5.1 shows the front panel of *INCE_INitialization_CEnter.vi (INCE)*. It contains the mentioned map of the system showing all connection lines.

Figure 5.4 on page 42 shows the block diagram *INCE_INitialization_CEnter.vi (INCE)*. The idea behind this user-interface is not to use LabView controls but a picture environment. The picture environment recognizes the position of the mouse cursor and the mouse click. This can be seen in sequence (1) in figure 5.4on page 42. Sequence (2) in figure 5.4 on page 42 activates the according pop-up menu to the action.

5.3.2 INIF_INitialization_Ini_File.vi

Figure 5.5 on page 43 shows the front panel of *INIF_INitialization_Ini_File.vi (INIF)*. It contains all available settings that can be changed.

Figure 5.6 on page 44 and 5.7 on page 45 give a short explanation of the most important settings.

5.3.3 UIUI_UserInterface_UserInterface.vi

The panels of the user-interface for operation except the Control-Panel can be seen in figure 5.2 on page 39 which shows the front panel of the UIUI_UserInterface_-

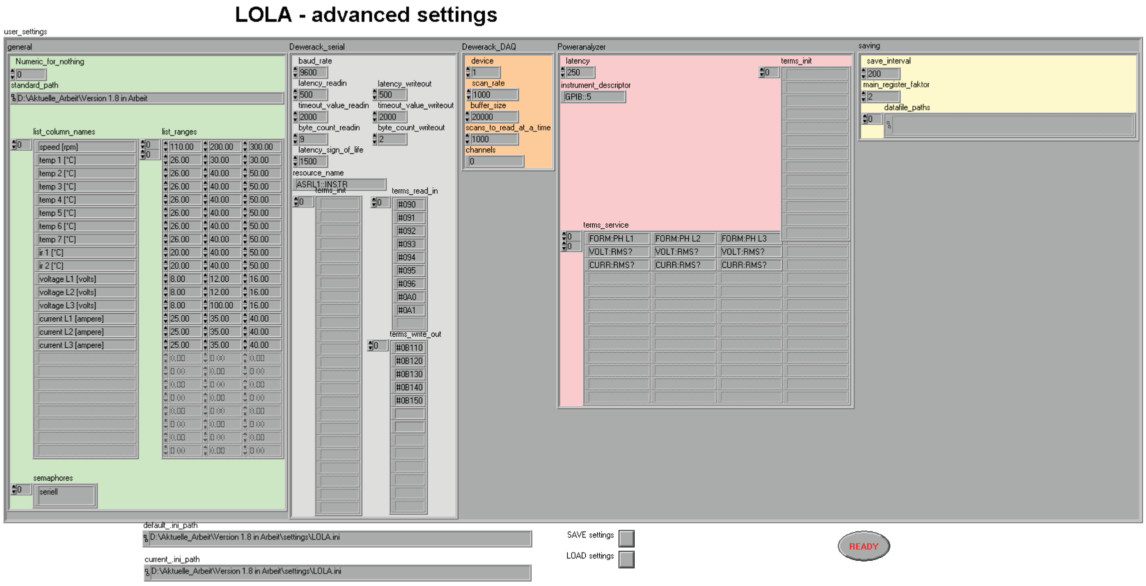


Abbildung 5.5: Front Panel INIF_INitialization_FIle.vi

UserInterface.vi (UIUI). The Control-Panel can be activated via the “display mode” switch.

Figure 5.8 on page 46 shows the block diagram of UIUI:

(1) While loop containing all elements that display data and information.

(1a) Case structure containing sub VIs for interpolation for the Graph-Panel.

(1b) For loop containing preparation of messages for the Messages-Panel.

(1c) Structures belonging to DDS ensuring proper termination of UIUI.

(2) While loop containing all elements for user interaction.

(2a) Triggering the save action.

(2b) Controlling the load-inverter speed.

(2c) Cases controlling the states of buttons and switches.

(2d) Structures ensuring proper termination of UIUI.

general

standard_path is the path where the program is located.

list_column_names is an array containing the titles of the measurement categories.
The names' order must match to the commands' order.

name_of_channel is displayed in the user interface, it consists of identifier and unit.

list_ranges is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*.

semaphores is an array of names of all existing semaphores.

Dewerack_serial

latency_readin is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop.

latency_writeout is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop.

byte_count_readin is the number of bytes to be read.

byte_count_writeout is the number of bytes to be written.

latency_sign_of_life is the time-period between two alternating signals sent to the PLC S7 200.

resource_name is the instrument descriptor of the interface addressing the Dewerack for VISA driver.

terms_init is an array of commands for initialization of the Dewerack

terms_read_in is an array of addresses respectively commands for the read request at a certain port.

terms_write_out is an array of addresses respectively commands for the write request at a certain port.

Dewerack_DAQ

device is the device number of the DAQ board.

scan_rate is the number of scans performed per second.

buffer_size buffer size is the number of scans each buffer should hold.

scans_to_read_at_a_time number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.

Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.

channels specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Abbildung 5.6: List of most important Settings (1)

Poweranalyzer

latency is the time in milliseconds for PARD_PowerAnalyzer_ReadData.vi to wait between two executions of the while loop.

instrument_descriptor is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver.

terms_init is an array of commands (called terms here) for the initialization of the Poweranalyzer.

terms_service is an array of commands (called terms here) for the read request.

The structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases.

saving

save_interval determines the amount of measurement-data saved.

main_register_faktor determines the amount of measurement-data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*.

datafile_paths name and location of files where the measured data is saved.

Abbildung 5.7: List of most important Settings (2)

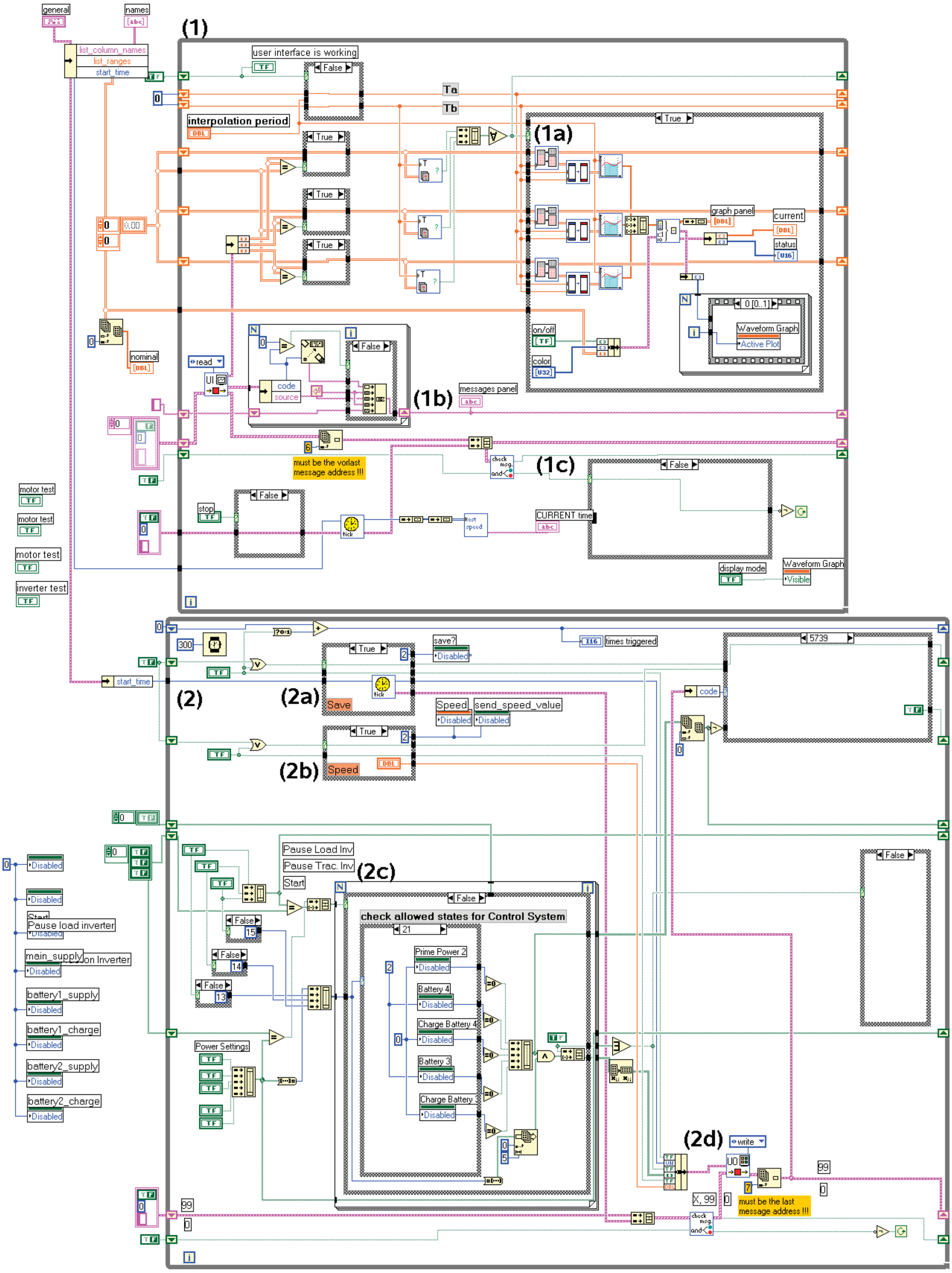


Abbildung 5.8: Block Diagram UIUI_UserInterface_UserInterface.vi

5.4 First Evaluation of User-Interface

A first evaluation should ensure that the basic challenge of LOLA's design meets all demands. Some evaluation aspects additionally to the already mentioned aspects is introduced before the tasks are evaluated.

5.4.1 Aspects

Ben Shneiderman lists five measurable factors which are central to evaluation (see [20] page 15):

- *Time to learn*: How long does it take for typical members of the user community to learn how to use the commands relevant to a set of tasks?
- *Speed of performance*: How long does it take to carry out the benchmark tasks? How fast are the tasks processed?
- *Rate of errors by users*: How many and what kinds of errors do people make in carrying out the benchmark tasks? Although time to make and correct errors might be incorporate into the speed of performance, error handling is such a critical component of system usage that it deserves extensive study.
- *Retention over time*: How well do users maintain their knowledge after an hour, a day, or a week? Retention over time may be linked closely to time to learn, and frequent of use plays an important role.
- *Subjective satisfaction*: How much did users like using various aspects of the system? The answer can be ascertained by interview or by written surveys that include satisfaction scales and space for free-form comments.

This test is less concerned in these five factors, they are only to be kept in mind. Emphasis is laid on a basic evaluation of functionality and design.

5.4.2 Result

The following lists all evaluated aspects with proposed improvements. Some of the improvements were implemented immediately. This evaluation lacks surveys of an independent user. The feedback of an independent user who is using the Test-Stand extensively will be interesting for future development.

User-Interface for Initialization

Pop-up menus are incomplete.	Implement the pop-up menus so that settings can be done as easy as possible.
User has to wait long until interaction is possible.	
Auto-creation of paths cannot be cancelled.	Correct the INM15_Initialization-_Menue15.vi.

User-Interface for Operation

Current states of devices cannot be recognized sufficiently.	Use switches instead of buttons.
Messages in the Message-Panel change to fast.	Split Message-Panel into tree fields for different messages rising in importance.
Status-Panel does not work properly.	Make user surveys to find out which information is reasonable in the Status-Panel.
Display of digits is not sufficient for all channels.	Enable different number of digits for each channel

Stress must be laid on the fact that this ‘test’ is very intuitive. Further evaluations must be prepared carefully. A scientific approach including the clear definition of a target and a precise measurement problem would be the basis for such an evaluation.

Kapitel 6

Data Saving

6.1 SASH_SAve_SHot.vi

Before explaining the process of saving something has to be said about time: *GLCL_-GLobal_CLock.vi (GLCL)* gives the time from initialization up to point of execution in milliseconds. This value of time is called 'time-stamp' in ININ_INitialisation_INitialisation.vi GLCL is initialized. This means that a initial system time is stored.

Saving is done in *SASH_SAve_SHot.vi (SASH).* In a first realization saving can be triggered by the user by clicking on the 'trigger' button. When clicking on the 'trigger' button, UIUI sends a message and the time-stamp (report) of the trigger action via the MGS to the SASH. Once the save action is triggered a certain amount of data is saved to hard-disk Each saved quantity is an average value of the certain amount of Measurement-Data.

The following explains the Block-Diagram of SASH (see figure 6.1):

(1) Settings are handed over:

- 'datafile_paths': Location where measurement data is saved on hard-disk.
- 'save_interval': The amount of data for averaging is determined by 'save_-interval' (see figure 6.2 on page 51). SASH buffers a more data than the 'save_interval' determines. If the 'trigger' message comes from UIUI, SASH derives average values for each quantity. Which values are averaged is determined by the reported 'trigger' time and the save interval.
- 'main_register_factor': SASH buffers an amount of Measurement-Data. 'save_interval' multiplied with 'main_register_factor' specifies the amount of data in the buffer.

(2) Time-stamp row is extracted from the measurement-data.

(3) Structures belonging to DDS ensuring proper termination of SASH.

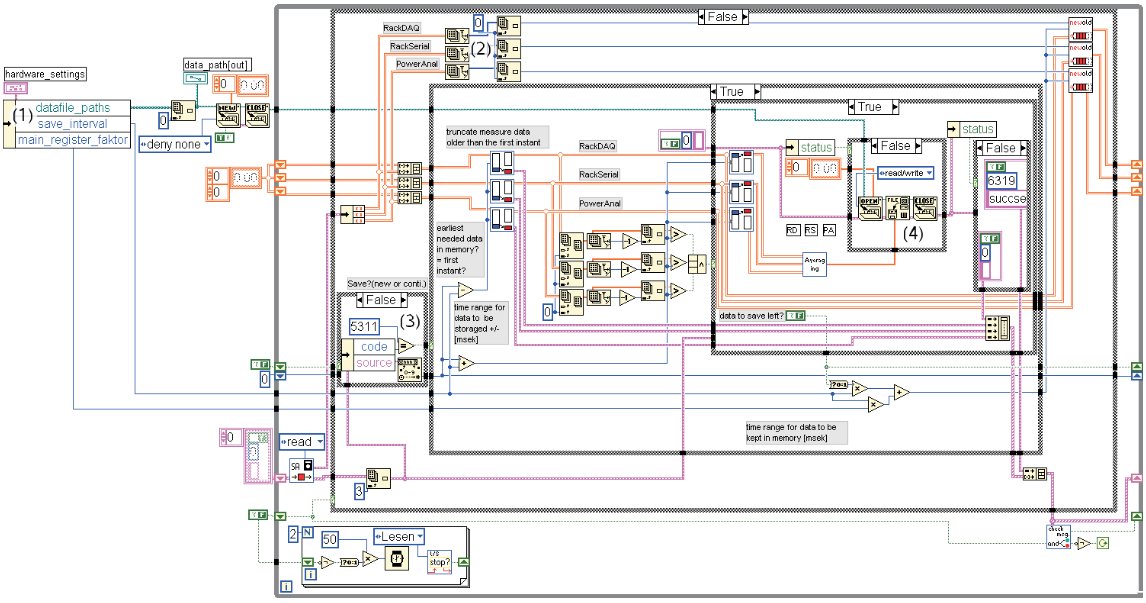


Abbildung 6.1: Block-Diagram SASH_SAve_SHot.vi

(4) Case structure handles the actual process of saving data to hard-disk. Here the values of the 'save_interval' are averaged so that only one value has to be saved to disk.

6.2 Data Format

In order make a flexible use of the measured data possible measurement data is saved to an ASCII-formatted file. The structure of this file can be seen in figure 6.3 on page 51 which shows the result of an example measurement.

The file consists of rows and columns. The first row contains are the names of quantities. Remaining rows contain measurement values. The first collum contains time-stamps. Remaining columns contain averaged values.

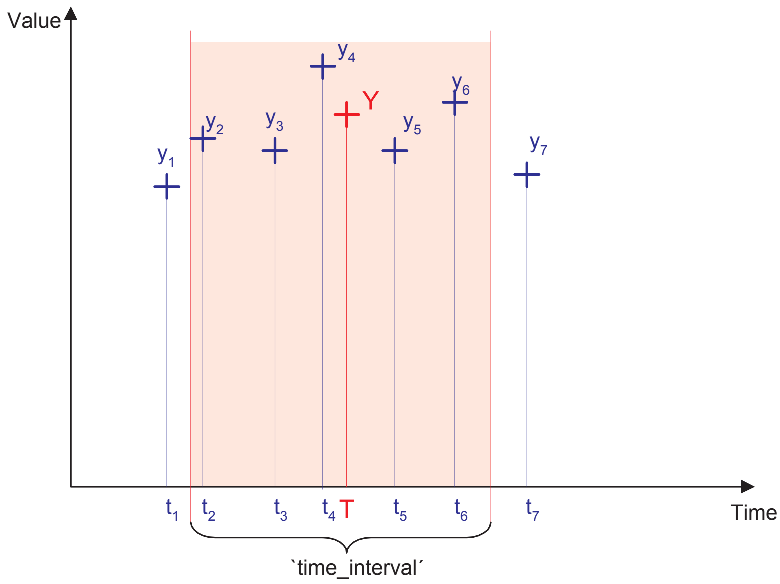


Abbildung 6.2: Process of Saving

$y_n \ldots$ measured values, $t_n \ldots$ time-stamps of measured values,

$Y = \frac{\sum_{n=2}^{6} y_n}{n} \ldots$ average value, $T \ldots$ 'trigger' time

Time [ms]	speed [RPM]	temp 1 [°C]	...	voltage L1 [volts]	voltage L2 [volts]	...
1070796.875	1494.104	32.600	...	2.456	2.749	...
1109556.000	1499.615	31.800	...	4.283	4.606	...
1132578.000	1499.468	31.600	...	4.130	4.568	...

Abbildung 6.3: Example Save-File

Kapitel 7

Hardware Communication

7.1 Devices

LOLA communicates as mentioned above with three devices:

- Dewerack-16,
- Poweranalyzer,
- Measurement board.

The first device is the Dewerack 16. Via the RS232 interface the temperature values of the nine temperature sensors are gathered and the communication with the PLC is realized. The second device is the Poweranalyzer. LOLA receives the measurement values of current and voltage from the Poweranalyzer via the GPIB-interface. The third device is the built-in measurement card which sends the measured values of torque and speed via the PCI-bus to LOLA.

7.2 Realization in G

The realization of the communication with these devices is done by using a application programming interface (API) called *Virtual Instrumentation Software Architecture* (VISA) and by using advanced LabView standard I/O-VIs. Whereas VISA is used for communication via GPIB and RS232 the I/O-VIs are used only for the data acquisition with the DAQ-board.

7.2.1 Concept of VISA

VISA is a standard I/O API for instrumentation programming. VISA by itself does not provide programming capability to realize communication with interfaces. Visa is a high-level API that calls into lower level drivers. The hierarchy of NI_VISA is shown in the figure 7.1 below.

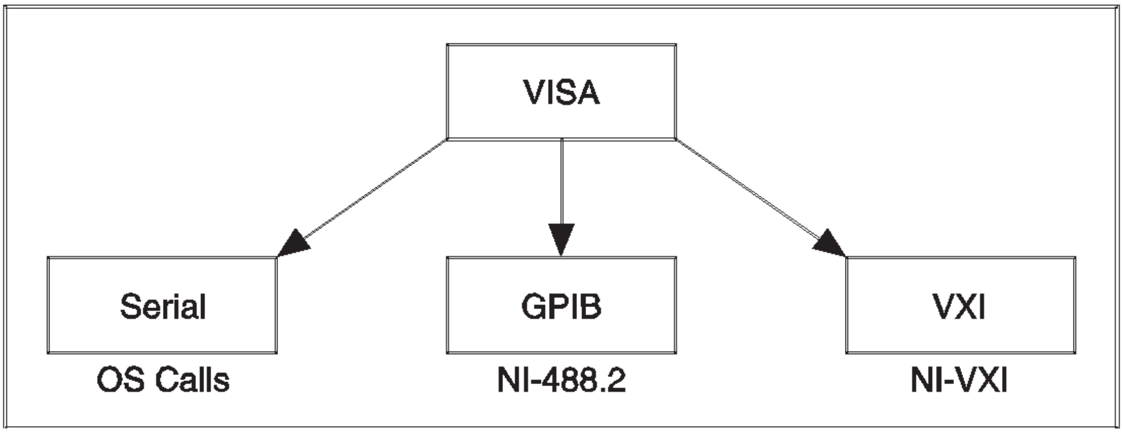


Abbildung 7.1: hierarchy of NI-VISA

VISA can control three bus systems, Vme eXtension for Instrumentation (VXI) (not used in the Test-Stand), GPIB or RS232, making the appropriate driver calls depending on the type of instrument being used. VISA uses the same operations to communicate with instruments regardless of the interface type. For example, the VISA command to write an ASCII string to a message-based instrument is the same wether the instrument is serial, GPIB or VXI. While the way the ASCII string is sent is independent of the device, the commands are not. Usually every device has its own set of commands to initialize, request data and write data. Detailed information of the concept of VISA, respectively NI-Visa is to be found in the user manual of LabView [14]

7.2.2 Advanced I/O VIs

For the data acquisition at the measurement board all subVI origin from the tool palette 'Data Acquisition'. This tool palette contains VIs to configure the measurement board, to receive and send data.

7.3 Communication VIs

7.3.1 Initialization

The settings for the communication origin from ININ_INitialization_INitialization.vi. They are used in HCCE_ HardwareConfiguration_CEnter.vi to set all parameters for the communication between the three devices in LOLA. The parameter are set in the according configuration-VI which are LabView own VIs. They can be found in the function palette in section *Data Acquisition* and *Instrument I/O*.

7.3.2 Communication Dewerack-16

In the VI RSRD_RackSerial_ReadData.vi (RSRD) VISA subVIs are used to read the temperatures values at PAD-TH8 and PAD-V8 from the Dewerack. In the VI RSWS_RackSerial_WriteS7.vi (RSWS)VISA subVIs are used to write through PAD-D07 to the PLC.

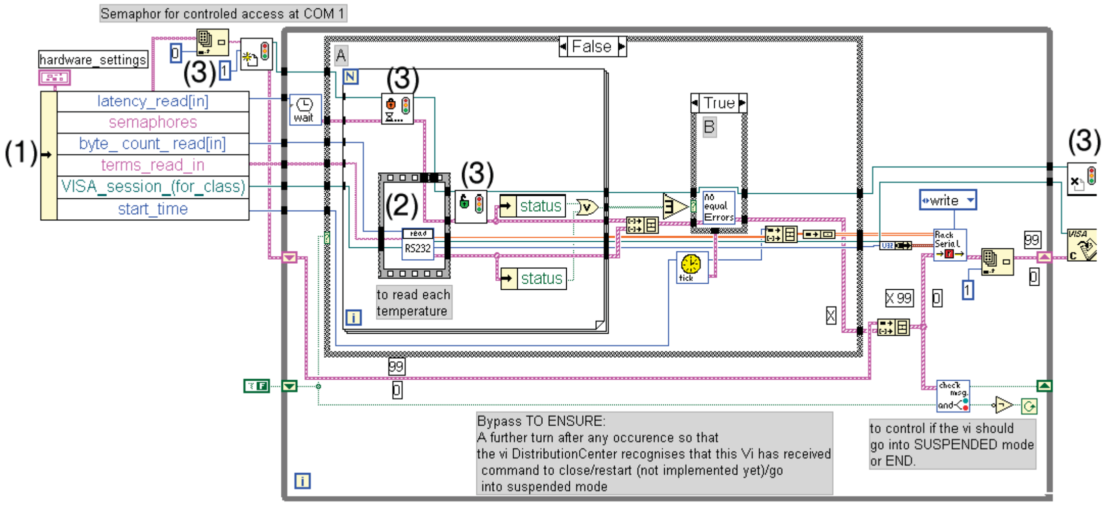


Abbildung 7.2: Read at Dewerack-16

RSRD_RackSerial_ReadData.vi

To receive a temperature value from a sensor a message string has to be sent to the Dewerack-16. Such a message string includes the address of the pad and the port where the sensor is plugged in. In order to receive all temperature values a whole set of message strings have to be sent sequentially. In figure 7.2 the set of message strings is found at (1) (terms_read_in). The single string is sent via VI-SA_write.vi to the Dewerack-16 and the according temperature value is passed back with VISA_read.vi. Both VIs are LabView own VIs and in RSRD they are located in RSRI_RackSerial_Read_dIrect.vi (2). Label (3) in figure 7.2 marks a special of this VI. The structures marked are VIs to manage semaphores. in RSRD a semaphore is used to control the access to the RS232 communication session. The access has to be controlled because the RS232 session is used by two VIs, RSRD and RSWS.

RSWS_RackSerial_WriteS7.vi

While RSRD is used to receive Measurement-Data, RSWS is used to write Commands to the PLC via digital outputs in PAD-D07. The procedure of writing is the same as reading. The difference lies only within the message string (1)(terms_write_out) in figure 7.3 on page 55.

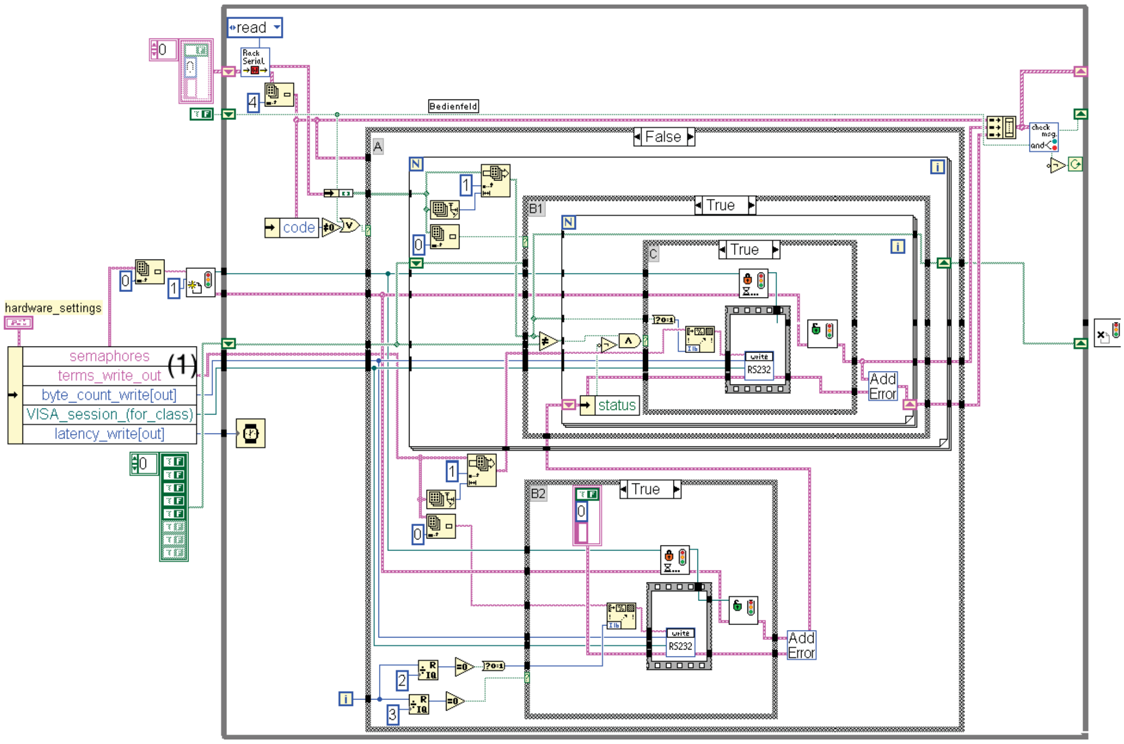


Abbildung 7.3: Write at Dewerack-16

7.3.3 Communication Poweranalyzer

For the fact that VISA is also used to communicate with the Poweranalyzer, the procedure is similar to the one in RSRD. The set of message strings (1) (terms_service) is also transferred via Visa_write.vi (2) and the Measurement-Data is received at (3) VISA_read.vi (figure 7.4).

7.3.4 Communication Measurement Board

The structure of RDRD is mainly derived from the example 'Acquire N Scans using Multiple Asynchronous Occurrences' which is one of the examples added to the online help. A Temporary-Storage, a GLCM_GLobal_CheckMessages.vi (1), a GLCL_GLobal _CLock.vi (2) and an error generating routine (3) are combined with the example (figure 7.5). The changes made to the example VI insure the compatibility with the MGS and the DDS.

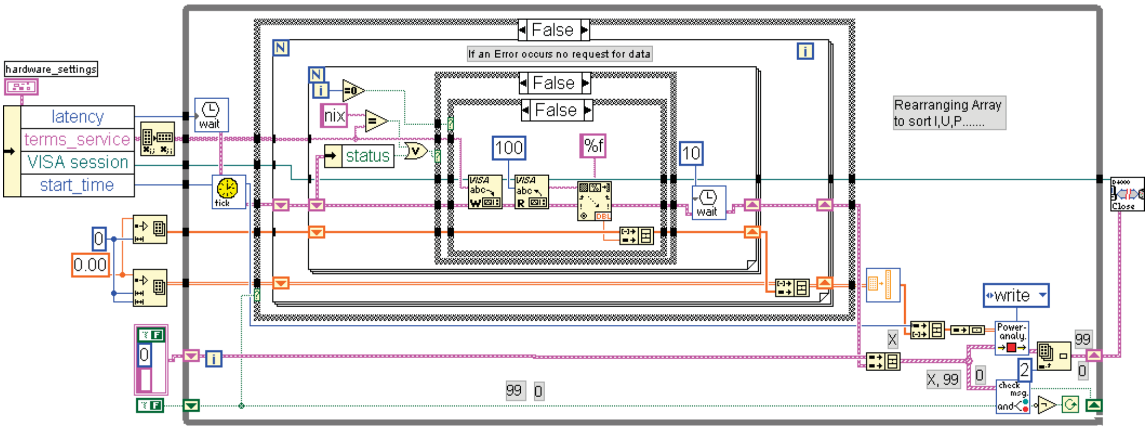


Abbildung 7.4: Read at Poweranalyzer

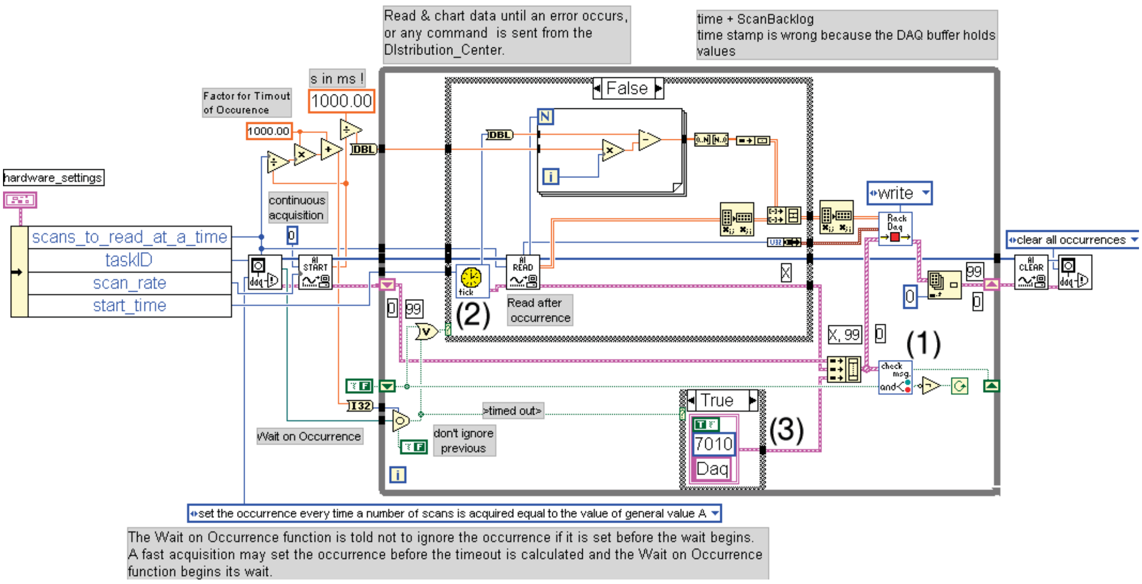


Abbildung 7.5: Read at Measurement Board

Kapitel 8

Security Concept

8.1 Overview

The global aim of the security system is to protect the operator and the expensive equipment. In the concept following possible threats are concerned:

- Overvoltage,
- Overcurrent,
- Destructive Heat,
- Exceeded Speed,
- PC Breakdown.

The concept is based on four domains, which are mentioned subsequently in figure 8.1 ordered by their reaction time with the fastest at first.

In this diagram more emphasis is put on the physical connections of the PLC with the surrounding system.

Figure 8.3 shows the general relations between the hardware devices, the PLC, the PC and the operator. As indicated each device offers information on its current status for different receivers. While each device is capable of protecting itself within certain limits (intrinsic safety), it is necessary to think about a strategy for all devices if a failure occurs. The PLC is the central device to gather all information and to decide if the system is to be set in a safe condition (Stop, Off). The safe condition is regarded as off-circuit except of the 24 V supply for the control equipment (PLC, contactors and control units of the inverters). Processes are deduced as mentioned from considered threats but were also deduced from termination commands given by the operator via the emergency break button or the PC.

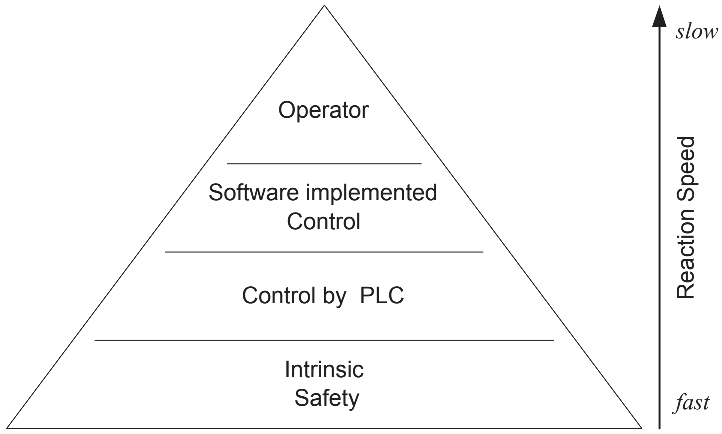


Abbildung 8.1: Security Domains

8.2 Intrinsic Safety

The intrinsic safety of the devices is regarded as the fastest and primary protection. It includes protection against damaging the device and endangering persons. This protection can be active like fuses or earth- leakage circuit breaker or passive like a high tolerable input range and a protective earth (PE) conductor. The following gives information of the intrinsic safety appliance in the system.

8.2.1 Power Supply

The line of the university (230 V line voltage) is secured by a ten- ampere quick-acting fuse and a current-operating earth-leakage circuit breaker with a maximum leakage current of 30 mA.

The 380 V line of the laboratory is secured by 25-ampere quick-acting fuse and current-operating earth-leakage circuit breaker with a maximum leakage current of 30 mA.

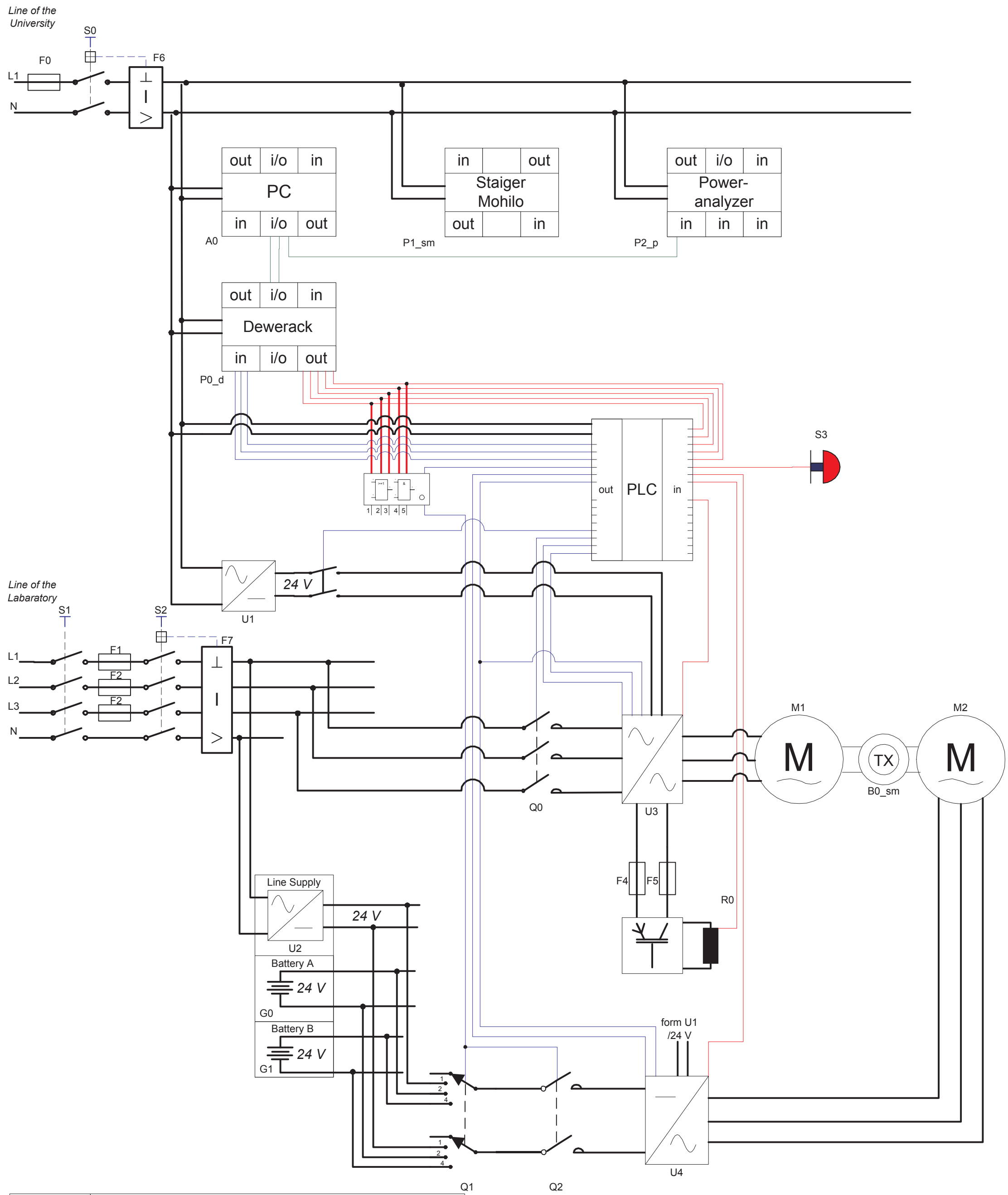
The 230 V line of the laboratory is secured by six-ampere quick-acting fuse and a current-operating earth-leakage circuit breaker with a maximum leakage current of 30 mA.

8.2.2 Measurement System

Dewerack 16

The line supply connector is equipped with a PE-conductor.

Except of the PAD-A01 all used modules are isolated between the input and the system. In general the isolation amounts to 1 kV. Therefore no direct current can



A0	Personal Computer
A1	Progammable Logical Controller
B0_sm	Speedcounter, Inductive Angular Positioning
B1-3_p	3 Phase Current Sensor Module
F1-F5	Fuses
F6-F7	Earth-Leakage Circuit-Breaker
G0-G1	Batteries 24 V
M0	Load Machine
M1	Test Machine
P0_d	Dewerack 16 spezial

P1_sm	Torque and Speed Measurement System
P2_p	Poweranalyzer
Q0-Q3	Contactors
R0-R1	Breaking resitor
S0-S3	Switches (S3 Emergency button)
U0-U4	Inverters

Abbildung 8.2: PLC Connection Plan

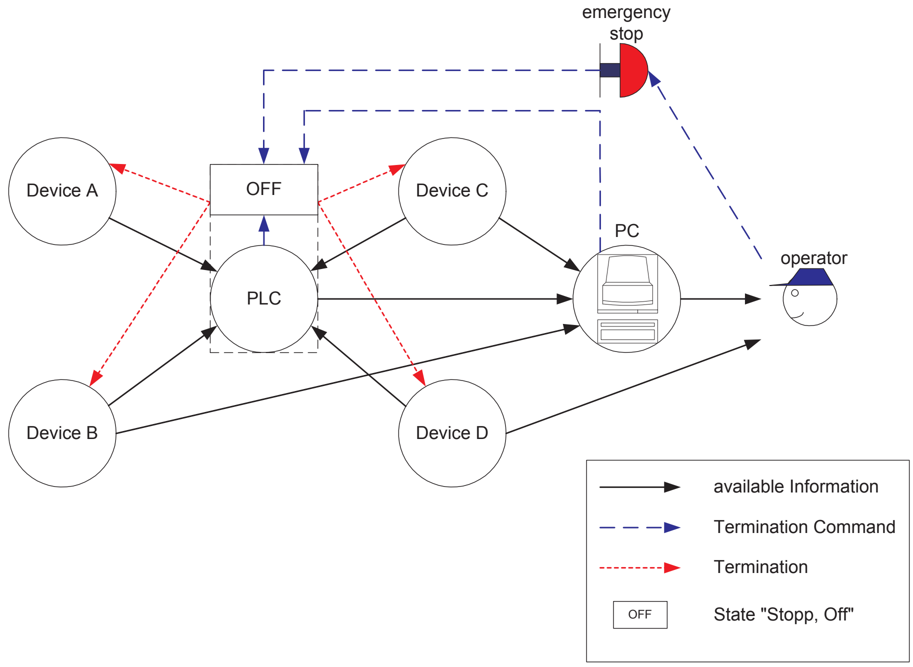


Abbildung 8.3: Security Concept – general Relationship

flow from the supply to the inputs if the voltage at the isolation remains below. The applied voltages and currents should be handled with care, because there is no tolerance according to the input ranges of the modules (see manual [8]).

Following standards are applied:

- EN50081 Part 1 and 2
- EN50082 Part 1 and 2
- general safety test according CE regulations

Poweranalyzer

The line supply connector is equipped with a PE-conductor and secured by a two-ampere quick-acting fuse. The connected voltage of 230 V is allowed to vary ±15 percent and its frequency may range from 45 up to 65 Hz.

The voltage to earth applied to the voltage and current inputs is limited to 600 V. This complies to the over-voltage category two. The five-ampere current input is secured against a short-term overload of 15 A and the thirty-ampere current input is secured against short- term overload of 100 A. The short term may last 20 msec.

The disruptive strength from housing to inputs amounts 2,2 kV/ 50 Hz/ 1 min, from input to input 4 kV/ 50Hz/ 1 min and from supply line to housing 1,5 kV/ 50 Hz/ 1 min (see manual [10]).

Following standards are applied:

- IEC 61010-1
- EN 61010-1
- IEC 61010-2-031
- EN 61010-2-031

Poweranalyzer is classified as class of protection I, disruptive strength class II and pollution severity class II.

Staiger & Mohilo

The line supply connector is equipped with a PE-conductor and secured by a two-ampere quick-acting fuse (see [6, 7]).

8.2.3 Load System

Load Inverter

The load inverter is protected by the security devices of the 380 V line supply. The task of upstream line filter is not to protect the inverter but to condition the supply.

For a save operation the load inverter offers the possibility to monitor several condition of the load system. The monitored conditions are listed below.

- temperature load motor
- temperature cooling sink
- internal temperature of the inverter
- external error
- resolver
- contouring error
- phase-angle sensor
- CAN bus
- speed

For each condition critical values and proximate reactions can be defined. If one of the monitored conditions exceeds its critical value, the system will start the proximate reaction. The name of these reactions and their triggered processes are described in figure 8.4 on page 62.

IF

- *TRIP* (highest priority) is set
  - the power outputs U,W,V are switched in high-impedance states until TRIP-reset
  - the drive spins (no controller active)
  - after TRIP-reset is set the drive returns to its set point following a predefined ramp
- *MELDUNG* is set
  - the power outputs U,W,V are switched in high-impedance states as long the interference is active.
  - short term interference ($\leq 0,5sec$)
    - the drive spins (no controller active)
    - after the interference has ended the drive returns to its set point with maximum torque.
  - long term interference ($\geq 0,5sec$)
    - the drive spins (no controller active)
    - reset of the controller
    - position setpoint = actual position
- *FAIL-QSP* is set
  - operation of the drive is stopped within a given time period
  - speed is decelerated following a defined ramp until standstill
- *WARNUNG* is set
  - drive operates controlled
- *AUS* (lowest priority)
  - monitoring deactivated

Abbildung 8.4: Load Inverter – Reactions and their Processes

Load Resistor

The load resistor is protected by means of a 50-ampere fuse for each line. A Temperature monitor is built in. If the resistort's temperature exceeds the permissible limit the inverter has to be cut off from the supply and the controller inhibit has to be set. Therefore the external monitoring circuit is connected to the temperature monitor.

Load Motor

The load motor is also equipped with an internal temperature sensor (type KTY). The sensor is connected via the resolver to the inverter which carries out the monitoring task.

Following *standards* are applied by all parts of the load system:

- EN50178,
- EN60146,
- EN60439,
- General safety test according CE regulations.

For further detail please refer to the manuals [2], [3] ,[4], [5].

8.3 PLC Control

8.3.1 Overview

The PLC with its expansion has 16 Inputs (E0.0-E1.7) and 14 outputs (A0.0-A0.5, A1.0-1.7) .These In- and Outputs enable the PLC to interact with the PC, the inverters and the contactors (further referred to as *System*). Figure 8.2 and 8.5 gives an overview of the connector assignment.

The task of the PLC are:

- reaction to errors at both inverters
- reaction to activation of the emergency button
- monitoring if the PC is online
- receiving and processing user-inputs
- control if the supply and charge status of the traction drive is allowed

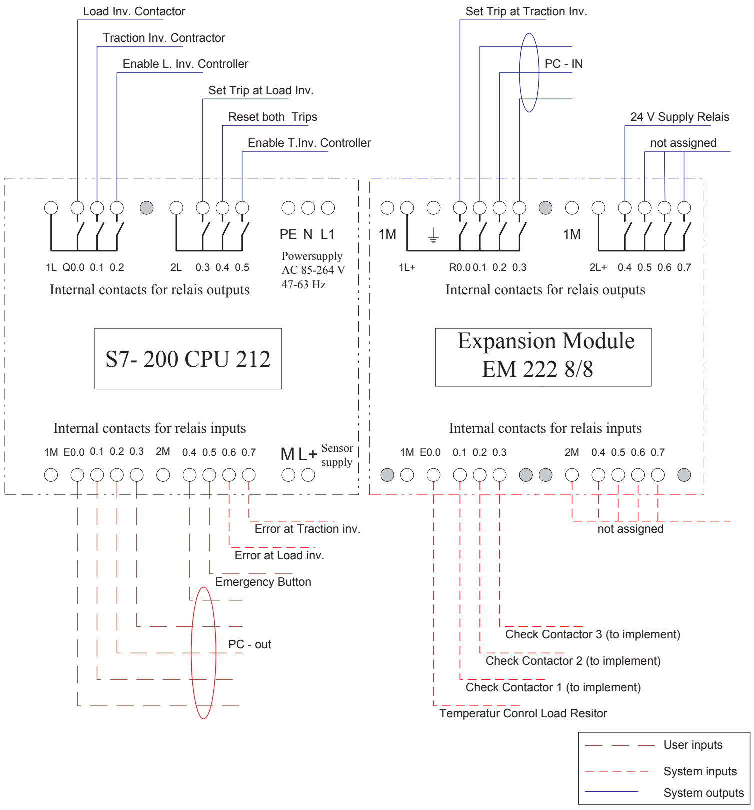


Abbildung 8.5: PLC assigned connectors

Reaction according to errors at both inverters and reaction to the activation of the emergency button is switching the System in a safe condition. A safe condition as mentioned above is regarded as off-circuit. Except of the 24 V supply for the control equipment (PLC, contactors and control units of the inverters) every power source is drained off. This is accomplished by operating the contactors.

The PC sends an alternating signal to the PLC. If the signal stops alternating the PLC regards the PC as off-line and pauses the system until the PC is online again or the system is shut down.

The operator has several possibilities to interact with the system. The System can be started, paused, stopped, and single inverters can be paused while the other is still operating.

To ensure that the operator does not mistakenly choose the wrong combination of supplying and charging, the settings for the two batteries and the line supply are controlled.

The implemented programming realize the mentioned tasks. Its developing Steps are listed below:

- identification of all inputs,
- identification of all outputs,

- identification of the states,
- design of a state diagram,
- definition of the transition conditions,
- declaration of all outputs each state.

The resulting program to meet the requirements above is realized as ladder diagram or statement list. The program itself is added to the appendix E.

8.3.2 Identification of all inputs

In table B.1 in appendix B all considered inputs with their assigned connection are shown. The inputs are divided in user-inputs and system-inputs. User inputs origin by an user-action (pressed button, pulled level ...) and system-inputs are generated by the hardware (sensor input, error bit ...).

Except of the emergency button all user-inputs are set by the user via LOLA. These user-inputs (2-8) are coded with 5 bits to save connectors and to be more flexibly if further user-inputs are to be added. In table B.2 in appendix B all used codes are listed.

8.3.3 Identification of All Outputs

In table B.3 in appendix B all considered outputs with their assigned connection are shown. There are only system outputs. The communication to the Pc is realized with three connectors. The combination of the three reflects the actual state the PLC program is in. All used combinations are listed In table B.4 in appendix B.

8.3.4 Identification of the States

This section deals with the question which states should be implemented in the PLC program to control the system. States are defined conditions of the System and there are no other conditions as the defined. The System can only be in one state at a time. Following states are implemented in the PLC program.

- S0 Initialization,
- S1 Wait,
- S2 Load Inverter Online,
- S3 Traction Inverter Online,
- S4 Both Inverter Online,

- S5 Stop,
- S6 Close Session,

The first state (S0) Initialization resets all used outputs and variables at System start. This has to be done so that the start condition is defined and every time the same.

(S1) Wait has the purpose to ”pause”the system. ”Pause”means that the System is halted without an error occurred. The system can be set online at any time.

State (S2) Load Inverter Online indicates that the load inverter is online and ready to receive the set point value for speed or torque. In this state the traction inverter is not online. It might be paused by the operator, or the supply is not proper set yet.

State (S3) Traction Inverter is online is the same state as S2 but the traction inverter is online and the load inverter is not.

(S4) Both Inverter Online represents the state where both inverter are online and ready to receive the set point value for speed or torque.

Only if an error at the inverters or the emergency button is pressed the System goes into (S5) Stop. The System is switched to a safe condition.

(S6) Close Session powers down the System and the 24V supply for the control equipment. After powering down the PLC and the Measurement-System can be shut down.

8.3.5 Design of the State Diagram

In figure 8.6 the state diagram is shown. It shows the way the System can change its state. A change in state can be accomplished along the drawn connections. The diagram also gives an preview of the transition conditions.

8.3.6 Definition of the Transition Conditions

The transition conditions are shown in detail in table B.6 in appendix B. If the transition condition is met, the System changes its state. As an example the transition condition for the transition from the state S2 Load Inverter Online to S1 Wait is explained:

The state is changed if no system error *and* one of the following occurred: Close-, Stop- or Pause Load Inverter- command *or* PC off-line.

8.3.7 Declaration of All Outputs

After identification of all states and the definition of transition conditions, each output has to be set in each state accordingly. The values of the outputs in every

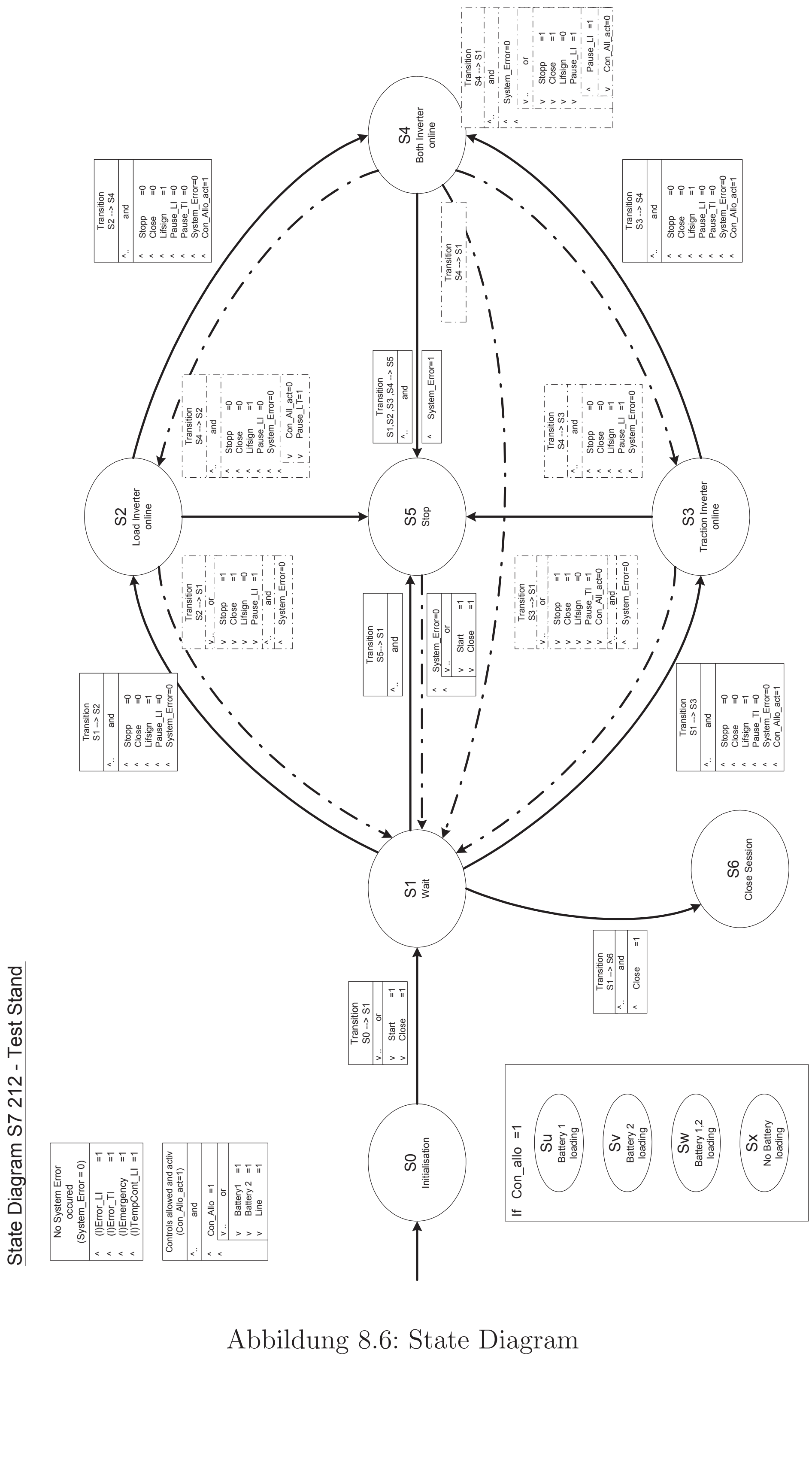


Abbildung 8.6: State Diagram

state is listed in table B.7 in appendix B.

8.4 Software Control

The theme software control encloses all security tasks handled by LOLA. These tasks are

- ensuring a proper programm execution,
- a lossless data acquisition as possible,
- processing of user-inputs.

These task are performed by MGS and SICE.

8.4.1 (MGS) Message Gathering System

The MGS is the transport system for all Control-Data. Its principle is explained in chapter 4.

8.4.2 (SICE) Signage-Center

The core of the software control implemented in LOLA is the subVI SICE_SInage_CEnter.vi (SICE). The principles of SICE (fig 8.7), Condition evaluation (1) and Report evaluation (2) are explained in chapter 4. In this subsection more detailed information about the user-defined error codes and the used state machine in the Report evaluation is given.

User-defined Error Codes

In LabView a wide range of Codes is used. Codes lasting from 5000 to 10000 are reserved for user-defined codes. Table C.1 and table C.2in appendix C is a list of all commonly used error codes. For an easier use the range of user-defined codes are split up in three parts. In LOLA the range from 5000 to 5999 is reserved for Commands. The range from 6000 to 6999 is reserved for Warnings and the remaining codes 7000 to 10000 are used for Errors. Table C.3 presents the detailed allocation of the user-defined error codes.

Report Evaluation - State Machine

Table 8.1 is a summary of the main functions of the state machine used for the Report evaluation. In the first column all concerned variables are listed. They are

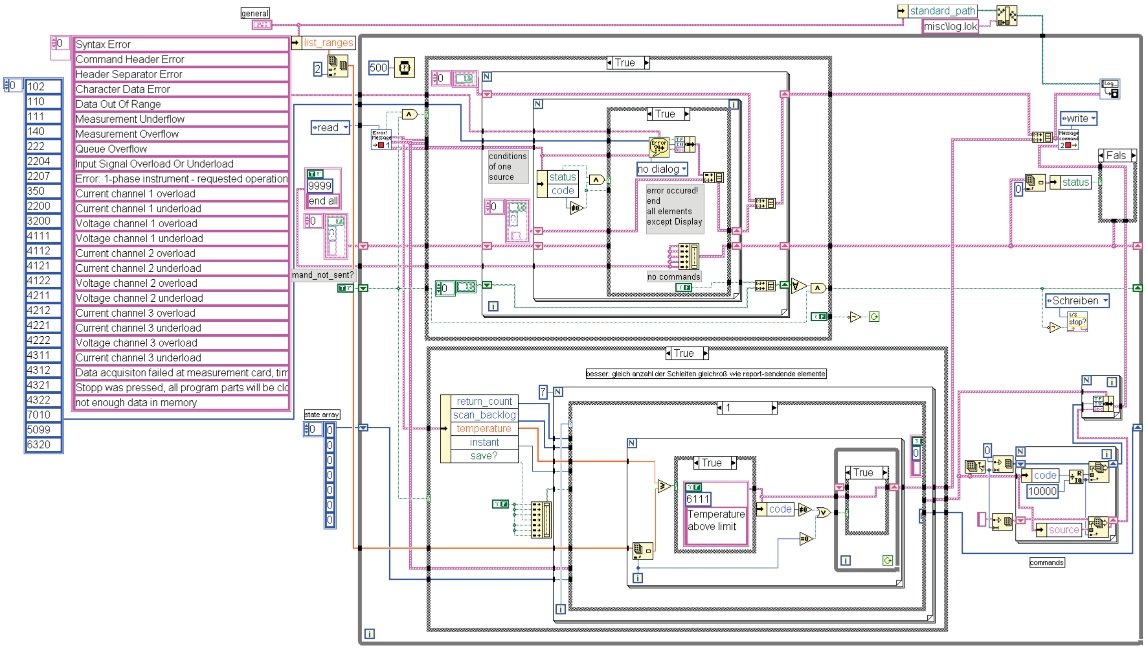


Abbildung 8.7: Block Diagram SICE

checked every turn and are compared with the evaluation criteria listed in the next column. The elements of the third column represent the requirement for a resulting action. The actions themselves are listed in the last column.

Example SSave is pressed”

To give an idea of the order of events the result of pressing the Save Button is described more detailed.

Without the Save-Button pressed the value of the state variable of Shot (the saving VI) is equal to zero. If the Save-Button is pressed at the user-interface, the button will be disabled, so no further activation can take place. The report of the activation containing a boolean variable (*save?= TRUE*) and the time of activation (*instant*) is sent via the Temporary-Storage TSUO, the distributing VI DICE and the Temporary-Storage TSM1 to the Message evaluating VI SICE. The Report evaluation registers the activation and sets the state variable of SHOT to one. In the same turn it sends the Command 5311 (User has pressed Save-Button) via TSM2, DICE and TSSA to SHOT. This Command will be sent until SHOT sends back the Warning 6311 (Saving in progress) or 6319 (Saving finished). If the sent Warning is 6311 the state variable of SHOT will be increased by one to two. The Report evaluation checks the incoming Messages for the Warning 6319 but the Command 5311 will not be sent anymore. If the sent Warning is 6319 the state variable is set to four. The Command 5619 (Enable the Save-Button) is sent to the user-interface the same way back which the report had taken. Afterwards the state variable is set back to zero. When Command is received in UIUI, the Save-Button is enabled again.

From State	*Evaluation Criteria*			*Command Sent*
Scanrate (S)	Warning Level	W_S	$W_S \leq S \leq E_S$	display
	Error Level	E_S	$S \geq E_S$	terminate
Temperatures ($T_1 - T_9$)	Warning Level	W_{T1-T9}	$W_{T1-T9} \leq (T_1 - T_9) \leq E_{T1-T9}$	display
	Error Level	E_{T1-T9}	$(T_1 - T_9) \geq E_{t1-T2}$	terminate
Return Count Read (RCR)	Byte read	9	$RCR \leq 9$	display
Return Count Write (RCW)	Byte written	2	$RCW \neq 2$	terminate
Save Button	Action	Pressed	=TRUE	start saving
			=FALSE	nothing
Settings Buttons	Action	Pressed	=TRUE	send new setting
			=FALSE	nothing

Tabelle 8.1: Evaluation Criteria

This procedure is used for every variable in table 8.1. The given example is a more complex one. In general there are less steps used.

8.5 Operator

The operator himself is also a part of the security concept. Operating the Test-Stand at this stage requires a schooled operator. He must be aware of all risks of his actions. Therefore the knowledge of all user-manuals of the used equipment is a requirement.

Kapitel 9

Measurement Example

This chapter is a report of a first test-circle of the Test-Stand. Measured and calculated values shall exemplify proper function of the test stand. They are of no significance to any real Test-Object. The theoretical background is taken from [22] page 399ff.

9.1 Problem Definition

The Test-Object is an four-pole induction motor.

Figure 9.1 shows the per-phase equivalent circuit of an induction motor. In the following the resistance of the stator R_s, the , the magnetizing inductance L_m, the the leakage inductance of the stator L_{ls} and the resister and the leakage inductance of the per-phase equivalent rotor winding R_r and L_{lr} will be evaluated from measured values, where $\vec{V}_s$ is the per-phase voltage and $\vec{E}_{ag}$ is the air gap voltage. R_s is the resistance of the stator winding and L_{ls} is the leakage inductance of the stator winding. $\vec{I}_m$ is the magnetizing component of the stator current $\vec{I}_s$.

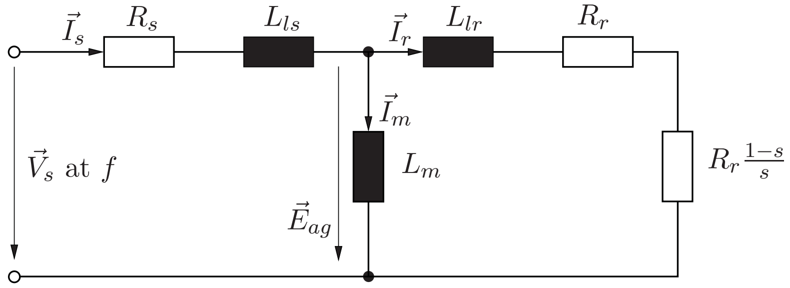


Abbildung 9.1: Per-phase Equivalent Circuit

9.2 Procedure

The synchronous speed in p-pole motor, supplied by frequency f, can be obtained as

$$\omega_s = \frac{2\pi/(p/2)}{1/f} = \frac{2}{p}(2\pi f) = \frac{2}{p}\omega \qquad \text{(rad/s)}$$

In terms of revolutions per minute (rpm), the synchronous speed is

$$n_s = 60 \times \frac{\omega_s}{2\pi} = \frac{120}{p} f$$

The slip s is

$$s = \frac{\text{slip speed}}{\text{synchronous speed}} = \frac{\omega_s - \omega_r}{\omega_s} = \frac{f_s - f_r}{f_s}$$

where ω_r is the speed of the rotor. The Slip speed ω_{sl}

$$\omega_{sl} = \omega_s - \omega_r = s\,\omega_s$$

is the speed of the air gap flux. Equations result from the per-phase equivalent circuit in figure 9.1:

$$\vec{E}_{ag} = \omega_s \frac{R_r}{\omega_{sl}} \vec{I}_r \, + \, j\omega_s L_{ir} \vec{I}_r$$
$$\vec{I}_s = \vec{i}m + \vec{I}_r$$
$$\vec{V}_s = \vec{E}_{ag} + (R_S + j\omega_s L_{ls})\vec{I}_s$$

No-load In no-load operation $\omega_r = \omega_s$ therefore $s = 0$. Thus

$$R_r \frac{1-s}{s} = \infty$$

and the per-phase equivalent circuit reduces to one circuit with R_s, L_{ls} and L_m.

To summarize:

$$\vec{E}_{ag} = j\omega_s L_m \vec{I}_s$$

$$\boxed{\vec{V}_s = [R_s + j\omega_s(L_{ls} + L_m)]\vec{I}_s}$$

Standstill In standstill operation $\omega_r = 0$ therefore $s = 1$. Thus

$$R_r \frac{1-s}{s} = 0.$$

As the voltages are low in order to keep I_s in the allowed boundaries the magnetizing inductance

$$L_m \simeq 0$$

and the per-phase equivalent circuit reduces to one circuit with R_s, L_{ls}, L_{lr} and R_r.

quantity	f	$V_{meas.}$	$I_{meas.}$	$\cos(\varphi_{meas.})$
units	Hz	volts	ampere	
no-load 1	86.45	5.28	10.67	0.927
no-load 2	86.37	8.95	18.41	0.909
no-load 3	86.29	11.37	24.03	0.910
standstill 1	74.17	2.01	16.81	0.584
standstill 2	74.18	2.148	24.18	0.688
standstill 3	74.069	2.151	24.18	0.691

Tabelle 9.1: Measured Values

To summarize:

$$\vec{E}_{ag} = [R_r + j\omega_s L_m]\vec{I}_s$$

$$\boxed{\vec{V}_s = [R_s + R_r + j\omega_s(L_{ls} + L_l r)]\vec{I}_s}$$

Table 9.1 shows the values measured with LOLA and the Test-Stand in the mentioned Example.

As the tested induction motor is delta connected and the measured voltages and currents are measured connected in star $V_{meas.}$, $I_{meas.}$ and $\cos(\varphi_{meas.})$ have to be transformed as follows:

$$V_s = V_{meas.}\sqrt{3},$$

$$I_s = \frac{I_{meas.}}{\sqrt{3}},$$

$$\cos\varphi_s = \cos(\varphi_{meas} + 30°).$$

Table 9.2 shows the results of this transformation.

quantity	ω	V_s	I_s	$\varphi_{meas.}$	φ_s	$\cos(\varphi_s)$
units	$sec.^{-1}$	volts	ampere	degree	degree	
no-load 1	543.2	9.14	6.16	22.03	52.03	0.615
no-load 2	542.7	15.50	10.63	24.63	54.63	0.579
no-load 3	543.1	19.69	13.87	24.50	54.50	0.581
standstill 1	466.0	3.478	9.71	54.27	84.27	0.100
standstill 2	466.1	3.72	13.96	46.53	76.53	0.233
standstill 3	465.4	3.73	13.96	46.30	76.30	0.237

Tabelle 9.2: Transformated Values

After transformation the per-phase equivalent circuit can be calculated. First the no-load operation is calculated.

$$\underline{R_s} = \frac{|V_s|}{|I_s|}\cos(\varphi)$$

$$\underline{L_m + L_{ls}} = \frac{1}{\omega_s} \cdot \frac{|V_s|}{|I_s|} \sin(\varphi)$$

Second the standstill operation is calculated.

$$\underline{R_r} = \frac{|V_s|}{|I_s|} \cos(\varphi) - R_s$$

$$\underline{L_{ls} + L_{lr}} = \frac{1}{\omega_s} \cdot \frac{|V_s|}{|I_s|} \sin(\varphi)$$

$$\underline{L_{ls} = L_{lr}}$$

9.3 Result

Figure 9.3 shows the Results of the calculation.

quantity	R_s	$L_m + L_{ls}$	
units	ohm	henry	
no-load 1	0.913	$2.154 \cdot 10^{-03}$	
no-load 2	0.844	$2.192 \cdot 10^{-03}$	
no-load 3	0.825	$2.132 \cdot 10^{-03}$	
average	**0.861**	$2.159 \cdot 10^{-03}$	
quantity	R_r	$L_r = L_{ls}$	L_m
units	ohm	henry	henry
standstill 1	0.897	$7.650 \cdot 10^{-04}$	$1.394 \cdot 10^{-03}$
standstill 2	0.923	$5.562 \cdot 10^{-04}$	$1.603 \cdot 10^{-03}$
standstill 3	0.924	$5.570 \cdot 10^{-04}$	$1.602 \cdot 10^{-03}$
average	**0.914**	$\mathbf{6.261 \cdot 10^{-04}}$	$\mathbf{1.533 \cdot 10^{-03}}$

Tabelle 9.3: Transformated Values

For an overview of the result the calculated per-phase equivalent circuit is summarized:

$$\underline{\underline{R_s = 0.861\,\Omega}}$$

$$\underline{\underline{R_m = 0.914\,\Omega}}$$

$$\underline{\underline{L_r = L_{ls} = 0.626\,\text{mH}}}$$

$$\underline{\underline{L_m = 1.53\,\text{mH}}}$$

Kapitel 10

Conclusion

The primary goal of our work was to build up an automated test stand. Our task was to create a concept and a program in G (LabView) which combines the given parts to a unit. The following list summaries our work done:

- Development of a general concept concerning:
  - Measurement Hardware;
  - Data-Acquisition;
  - Control Hardware and
  - Security
- Programming LOLA;
- Review of the concept.

The demands on the resulting Test-Stand are flexibility, an easy usage, reliable data acquisition and security concerning its operation.

The requirement of flexibility is met by LOLA. Due to its modular concept it can be improved or extended. With a specific measurement task LOLA can be 'easily' adapted and put in service. Unfortunately the word 'easily' has to be seen in relation to good programming skills in the programming language G. Therefore only skilled personal should adapt LOLA for further purposes.

Further requirements are met theoretically. The concept has been completed, but further tests and practical experience are necessary to proof its capability.

10.1 Further Goals

LOLA was never intended to be complete at all. Demand on the Test-Stand will change. Therefore demands on LOLA will change. This section should provide ideas and suggestions for successors concerning expansions and improvements of the Test-Stand and LOLA.

10.1.1 Evaluation of the Measurement System

There was no evaluation whether the measured values are reproducible or not. Therefore the system should be tested carefully to ensure proper measurement functionality.

10.1.2 Improvement for Test-Stand

Several user operations and the evaluation of the acquired data have not been automatized yet. This full automatization combined with a certain given application is still a great challenge.

10.1.3 Expansions for LOLA

During developing there were several ideas according additional capabilities of LOLA. To mention only the most important: First there is improvement of automation, second there predefined load diagrams, third there is TCP/IP connectivity and last but not least there is perfection of the security concept.

The modular construction of the software establishes the basis for more automatization of the system. The target is to have test-cycles that last several days or even weeks without any user operation. Such long-time tests can help to improve traction drive systems for vehicles.

Long-time tests are reasonable if there are predefined load diagrams. Load diagrams are simulations of driving situations which a vehicle has to cope with when it is used. This means that a load diagram is an alternating load, simulating a certain road profile (e.g. rush-hour traffic, mountain trips). This makes realistic testing possible.

A TCP/IP connectivity enables offering data online in the local area network of the department and worldwide through the internet. Furthermore such a connection makes control of LOLA from possible from everywhere.

Last but not least the realization and improvement of the security concept has also be counted to future goals, because at the moment only the concept is elaborated.

Improvements for the programming of LOLA

There are some aspects of LOLA which can be improved in further versions. The proposal for improvement concern the Message Gathering System (MGS) and Sinage_Center.vi (SICE), saving and the initialization user-interface.

The proposed change to SICE is to standardize the message evaluation by integrating the condition evaluation in the state machine of the report evaluation. The structure would become easier to handle.

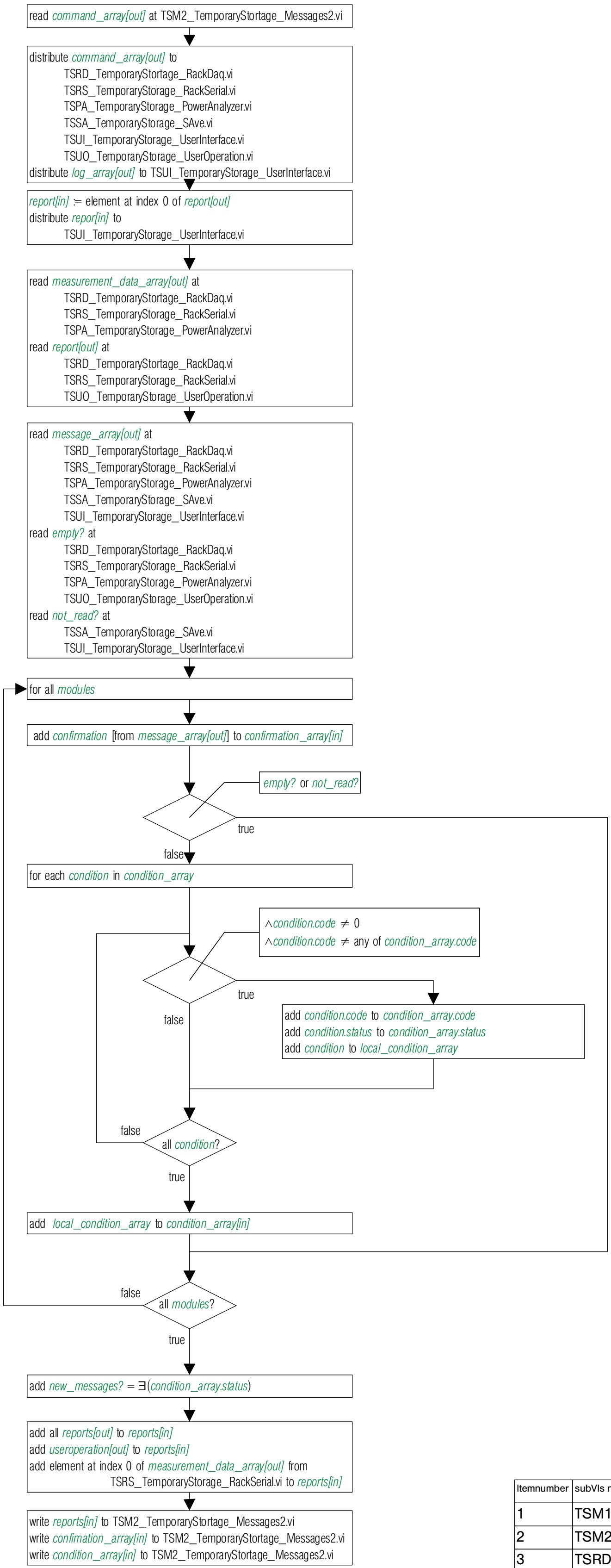
The influence of MGS can be extended over the whole program. At the moment it is implemented in the simultaneously executing VIs. It will allow reactivating and re-initializing LOLA. Therefore it must not be terminated after one execution.

The saving of data to disk can be improved regarding to statistic and mathematical operations and its degree of automatization can be increased too. This will ease the Measurement-Data evaluation and improve also the data reliability.

The proposal for improvement of the user-interface for initialization concern the adaption to an specific operator. At the moment there is a excellent shell. But it could not been tested for all incidence.

Anhang A

Flowcharts



Variables:

form TSM2_TemporaryStortage_Messages2.vi:
- *messages2 (command_array, log_array)*
- *command*
- *command_array*
- *log*
- *log_array*

to TSM1_TEmporaryStortage_MEssages1.vi:
- *new_messages?*
- *messages1 (condition_array, confirmation_array, report_array)*
- *confirmation*
- *confirmation_array*
- *condition*
  - *condition.code*
  - *condition.status*
- *condition_array*

from TemproraryStorage:
measurement_data
- *message_array [index0: confirmation, index1..: condition_array]*
- *empty?*
- *not_read?*

to TemproraryStorage:
- *report*

local:
- *modules*
- *condition_array-status*
- *local_condition_array*

Itemnumber	subVIs name
1	TSM1_TemporaryStorage_Messages1.vi
2	TSM2_TemporaryStortage_Messages2.vi
3	TSRD_TemporaryStortage_RackDaq.vi
4	TSRS_TemporaryStortage_RackSerial.vi
5	TSPA_TemporaryStortage_PoweranAlyzer.vi
6	TSSA_TemporaryStortage_SAve.vi
7	TSUI_TemporaryStortage_UserInterface.vi
8	TSUO_TemporaryStortage_UserOperation.vi

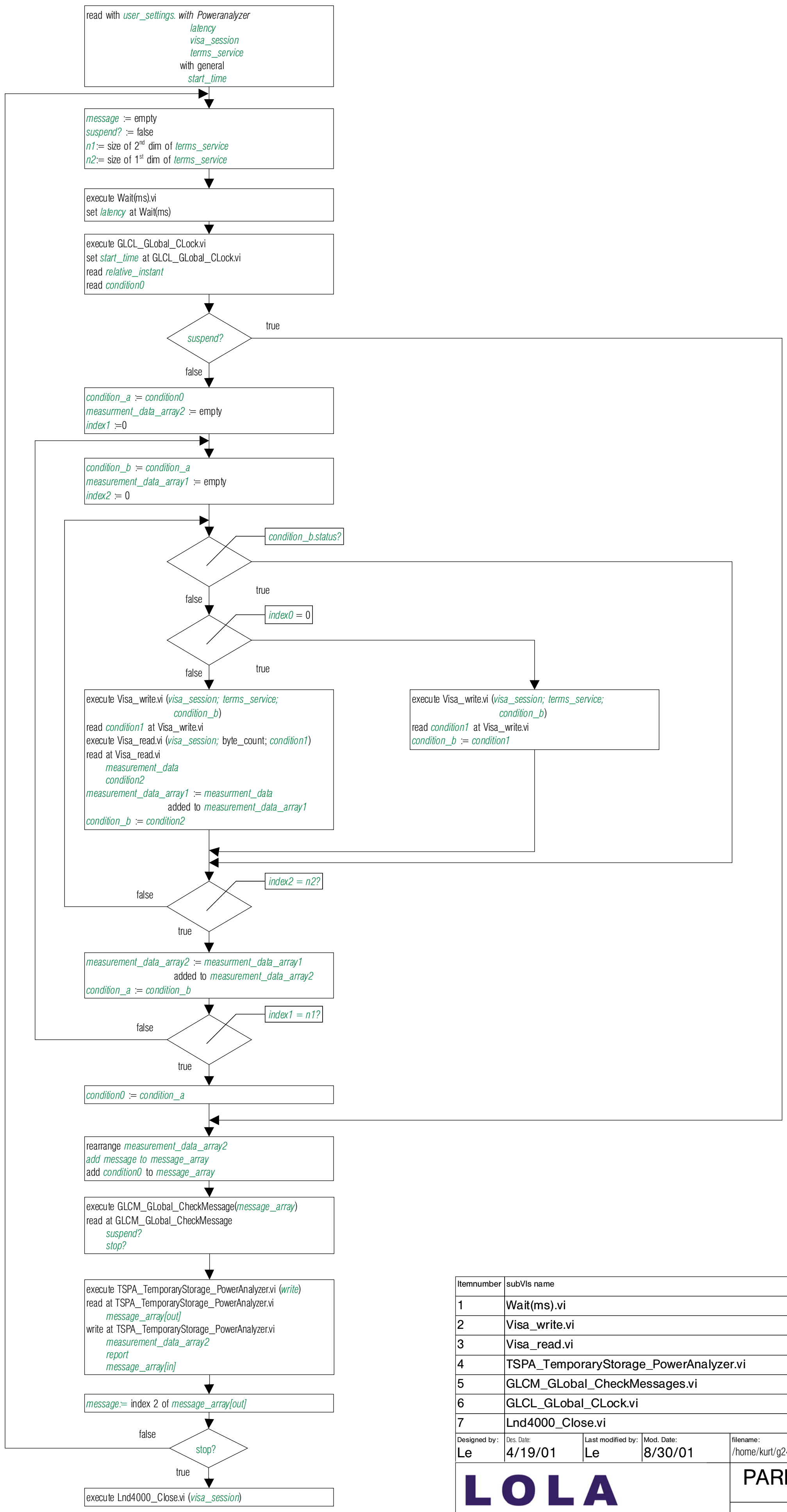
Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Mö	3/29/01	LE	8/30/01	/home/kurt/g2415m/Flieádiag/fl-DICE_flowchart-DIstributionCEnter.sda

LOLA
Institut für Elektrotechnik 2001

DICE_DIstributionCEnter.vi

LOLA: Version 1.9

Page 1 of 1



Variables:

from user_settings (Input):
with Poweranalyzer
latency
visa_session
terms_service
with general
start_time
from GLCL_GLobal_CLock.vi
relative_instant
condition0
from Visa_write.vi
condition1
to Visa_read.vi
condition1
byte_count:= 100
from Visa_read.vi
measurement_data
condition2
to TSPA_TemporaryStorage_PowerAnalyzer.vi
measurement_data_array2
message_array (internal)
from TSRD_TemporaryStorage_PowerAnalyzer.vi
message_array (ex)
from GLCM_GLobal_CheckMessages.vi
suspend?
stop?
local:
message
measurement_data_array1
condition_a
condition_b
n1
n2
index1
index2

Itemnumber	subVIs name
1	Wait(ms).vi
2	Visa_write.vi
3	Visa_read.vi
4	TSPA_TemporaryStorage_PowerAnalyzer.vi
5	GLCM_GLobal_CheckMessages.vi
6	GLCL_GLobal_CLock.vi
7	Lnd4000_Close.vi

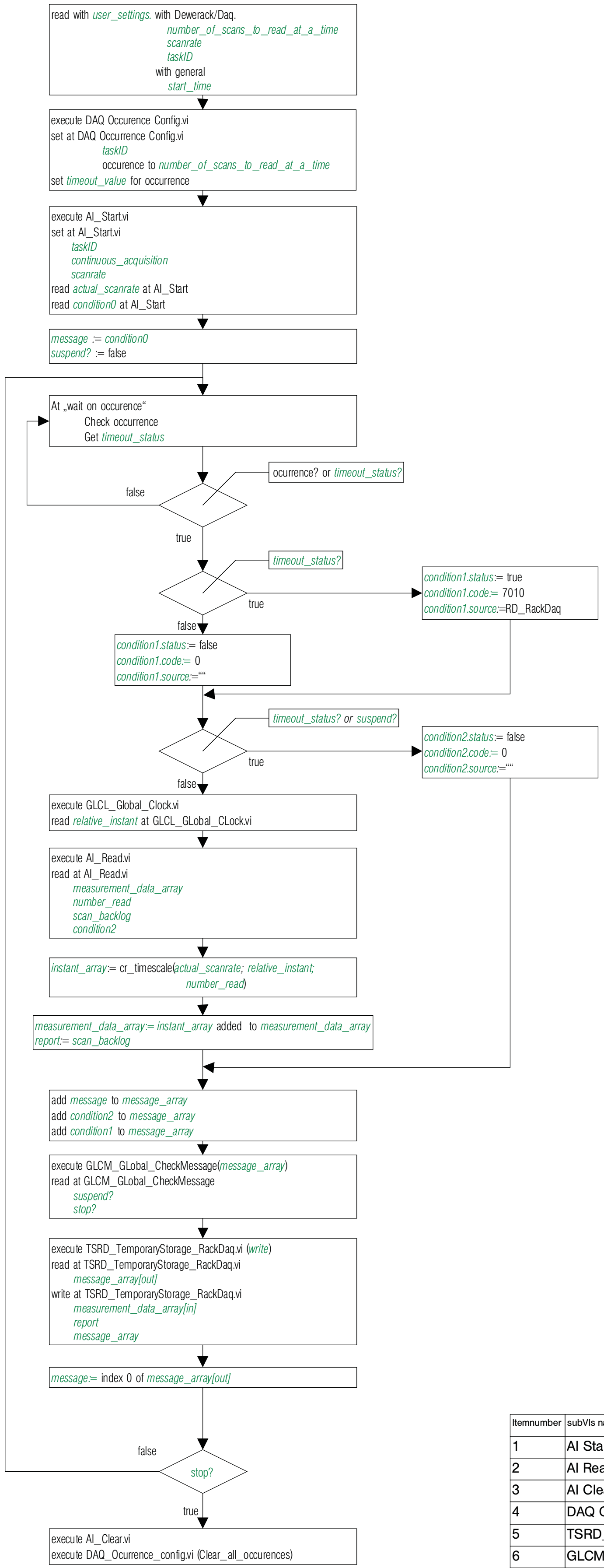
Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	4/19/01	Le	8/30/01	/home/kurt/g2415m/Fleädiag/fl-PARD_flowchart-PowerAnalyzerReadData.sda

LOLA
Institut für Elektrotechnik 2001

PARD_PowerAnalyzer_ReadData.vi

LOLA: Version 1.9

Page 1 of 1



Variables:

from *user_settings* (Input):
with Dewerack/Daq.
number_of_scans_to_read_at_a_time
scanrate
taskID
with general
start_time
to AI_Start.vi
const= continuous acquisition
from AI_Start.vi:
actual_scanrate
condition0
from AI_Read.VI:
measurement_data
scan_backlog
number_read
to TSRD_TemporaryStorage_RackDaq.vi
const= write
message_array
report
from TSRD_TemporaryStorage_RackDaq.vi
message_array[ou]
from GLCL_GLobal_CLock.vi
relative_instant
from GLCM_GLobal_CheckMessages.vi
suspend?
stop?
local:
message
timeout_value
timeout_status
instant_array
condition(1..).status
condition(1..).code
condition(1..).source

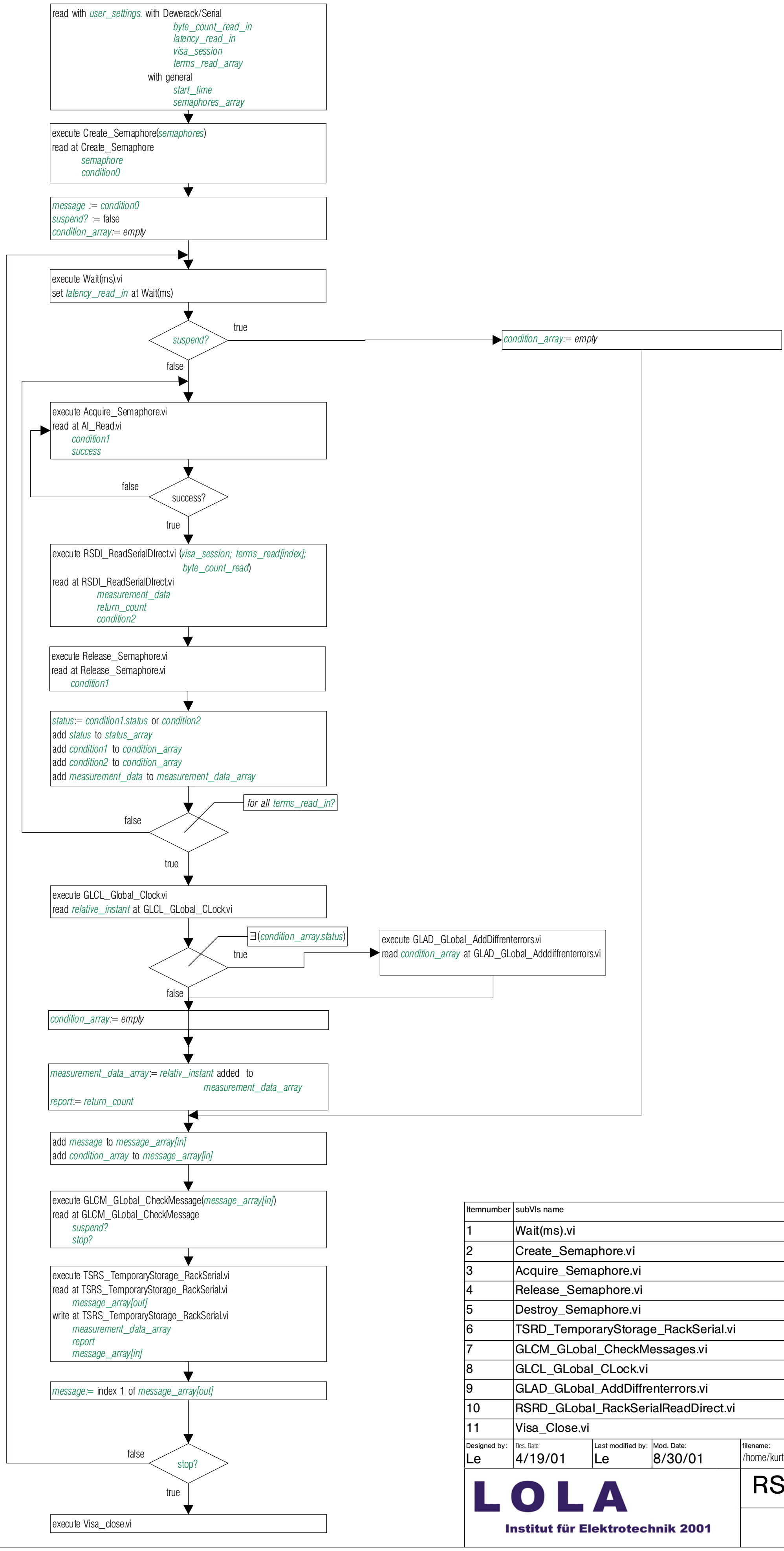
Itemnumber	subVIs name
1	AI Start.vi
2	AI Read.vi
3	AI Clear.vi
4	DAQ Occurrence Config.vi
5	TSRD_TemporaryStorage_RackDaq.vi
6	GLCM_GLobal_CheckMessages.vi
7	GLCL_GLobal_CLock.vi

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	4/19/01	Le	8/30/01	/home/kurt/g2415m/Flieádiag/fl-RDRD_flowchart-RackDaqReadData.sda

LOLA
Institut für Elektrotechnik 2001

RDRD_RackDaq_ReadData.vi

LOLA: Version 1.9



Variables:

from user_settings (Input):
with Dewerack/Serial
byte_count_read_in
latency_read_in
visa_session
terms_read_array
with general
start_time
semaphores_array
from Create_Semaphore
semaphore
condition0
from Acquire_Semaphore
condition1
from RSRD_RackSerial_ReadDirect.vi
return_count
measurement_data
condition2
from GLCL_GLobal_CLock.vi
relative_instant
from GLAD_Global_AddDiffrenterrors.vi
condition_array
to TSRD_TemporaryStorage_RackSerial.vi
message_array[in]
report
from TSRD_TemporaryStorage_RackSerial.vi
message_arra[out]
from GLCM_GLobal_CheckMessages.vi
suspend?
stop?
local:
message
condition_array
status
status_array

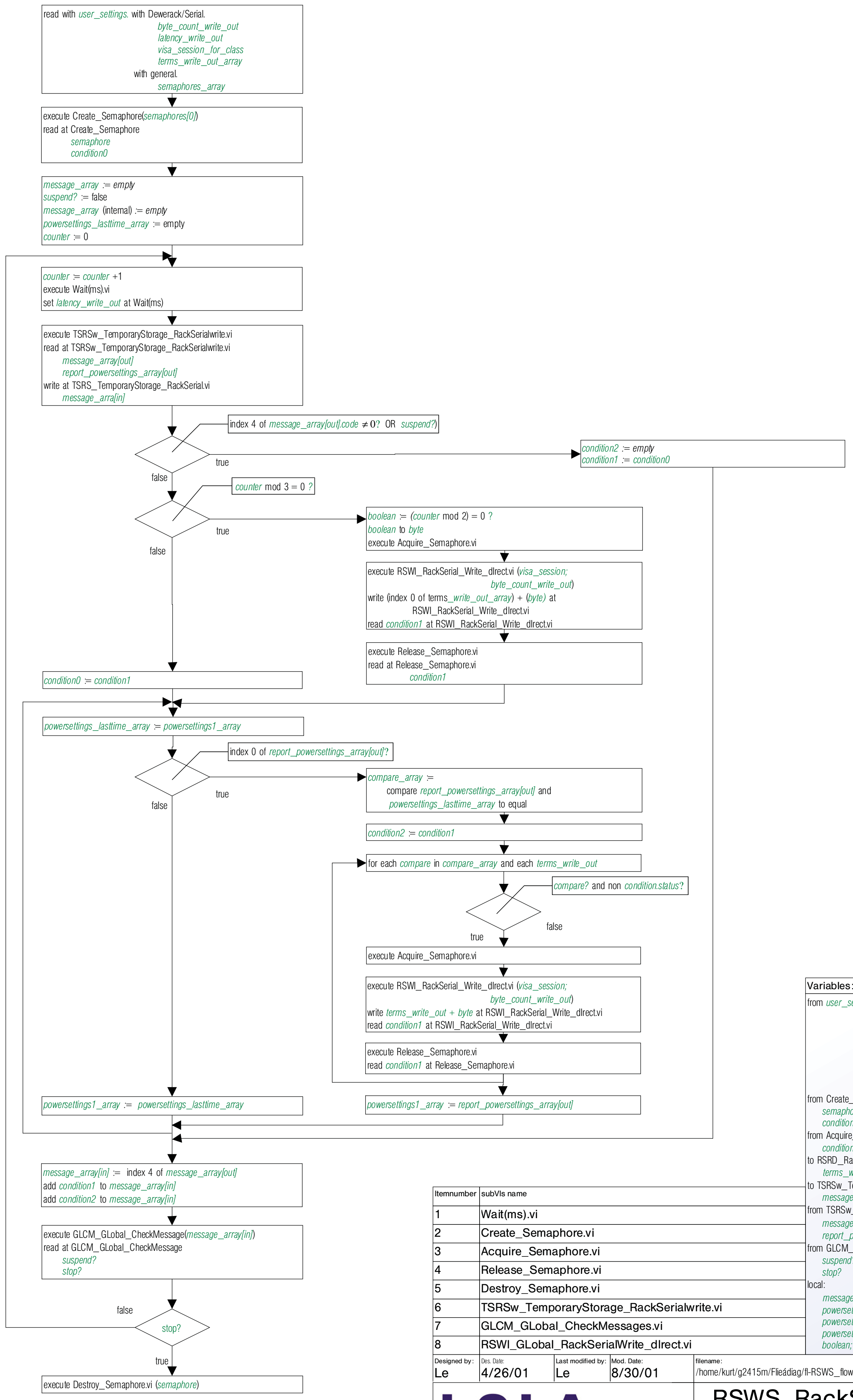
Itemnumber	subVIs name
1	Wait(ms).vi
2	Create_Semaphore.vi
3	Acquire_Semaphore.vi
4	Release_Semaphore.vi
5	Destroy_Semaphore.vi
6	TSRD_TemporaryStorage_RackSerial.vi
7	GLCM_GLobal_CheckMessages.vi
8	GLCL_GLobal_CLock.vi
9	GLAD_GLobal_AddDiffrenterrors.vi
10	RSRD_GLobal_RackSerialReadDirect.vi
11	Visa_Close.vi

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	4/19/01	Le	8/30/01	/home/kurt/g2415m/Fleádiag/fl-RSRD_flowchart-RackSerialReadData.sda

LOLA
Institut für Elektrotechnik 2001

RSRD_RackSerial_ReadData.vi

LOLA: Version 1.9



Variables:

from user_settings (Input):
with Dewerack/Serial
byte_count_write_out
latency_write_out
visa_session
terms_write_out
with general
semaphores_array
from Create_Semaphore
semaphore
condition0
from Acquire_Semaphore
condition1
to RSRD_RackSerial_WriteS7.vi
terms_write_out_array
to TSRSw_TemporaryStorage_RackSerialwrite.vi
message_array[in]
from TSRSw_TemporaryStorage_RackSeriawritel.vi
message_arra[out]
report_powersettings_array
from GLCM_GLobal_CheckMessages.vi
suspend?
stop?
local:
message_array
powersettings_array
powersettings_array1
powersettings_lasttime_array
boolean; byte

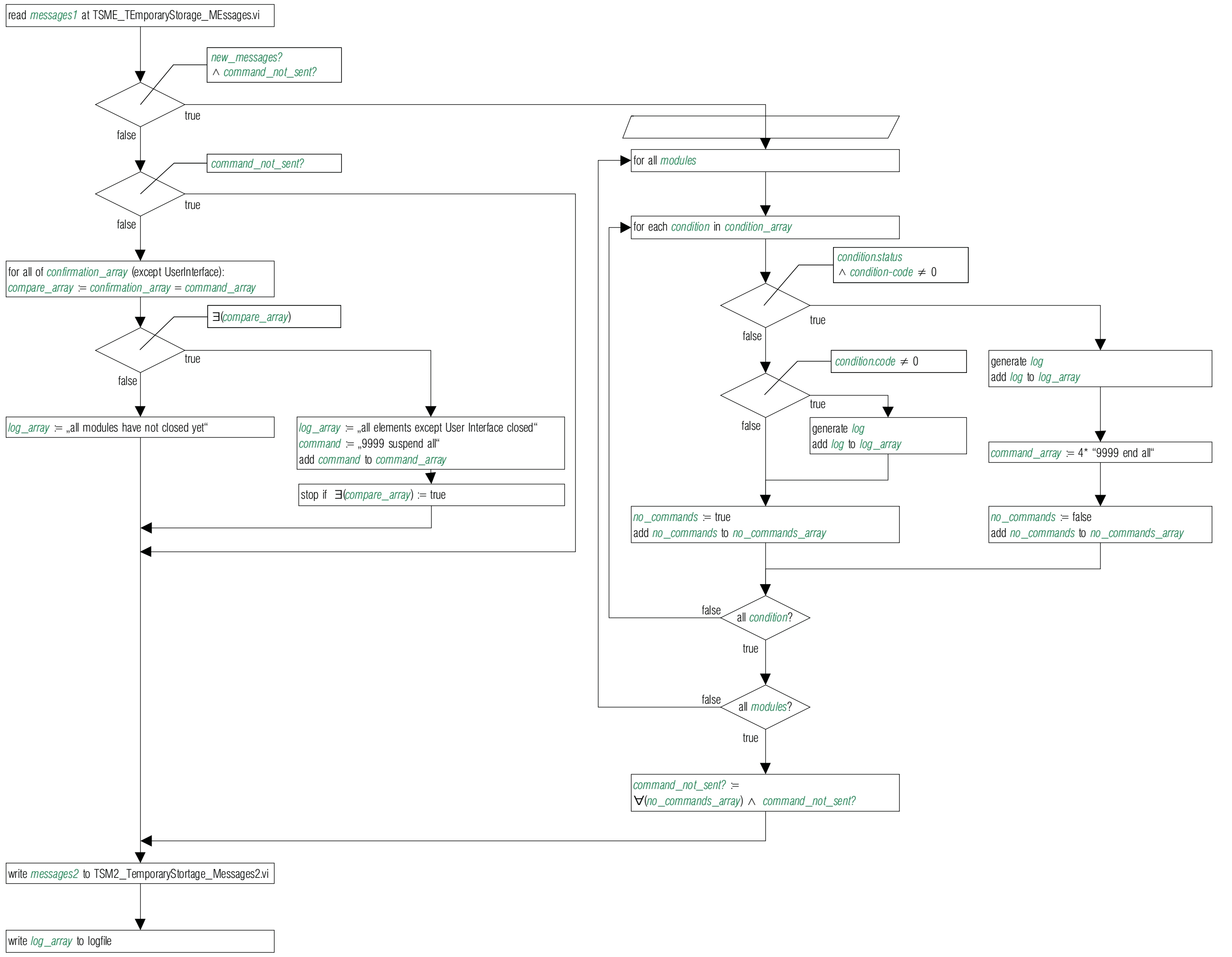
Itemnumber	subVIs name
1	Wait(ms).vi
2	Create_Semaphore.vi
3	Acquire_Semaphore.vi
4	Release_Semaphore.vi
5	Destroy_Semaphore.vi
6	TSRSw_TemporaryStorage_RackSerialwrite.vi
7	GLCM_GLobal_CheckMessages.vi
8	RSWI_GLobal_RackSerialWrite_dIrect.vi

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	4/26/01	Le	8/30/01	/home/kurt/g2415m/Flieádiag/fl-RSWS_flowchart-RackSerialWriteS7.sda

LOLA
Institut für Elektrotechnik 2001

RSWS_RackSerial_WriteS7.vi

LOLA: Version 1.9



Variables:

from TSM1_TEmporaryStorage_MEssages1.vi:
messages1 (condition_array, confirmation_array, report_array)
- *confirmation*
- *confirmation_array*
- *condition*
  - *condition.code*
  - *condition.status*
- *condition_array*
- *new_messages?*

to TSM2_TemporaryStortage_Messages2.vi:
messages2 (command_array, log_array)
- *command*
- *command_array*
- *log*
- *log_array*

local:
- *command_not_sent?*
- *no_commands*
- *no_commands_array*
- *modules*
- *codition*
- *compare_array*

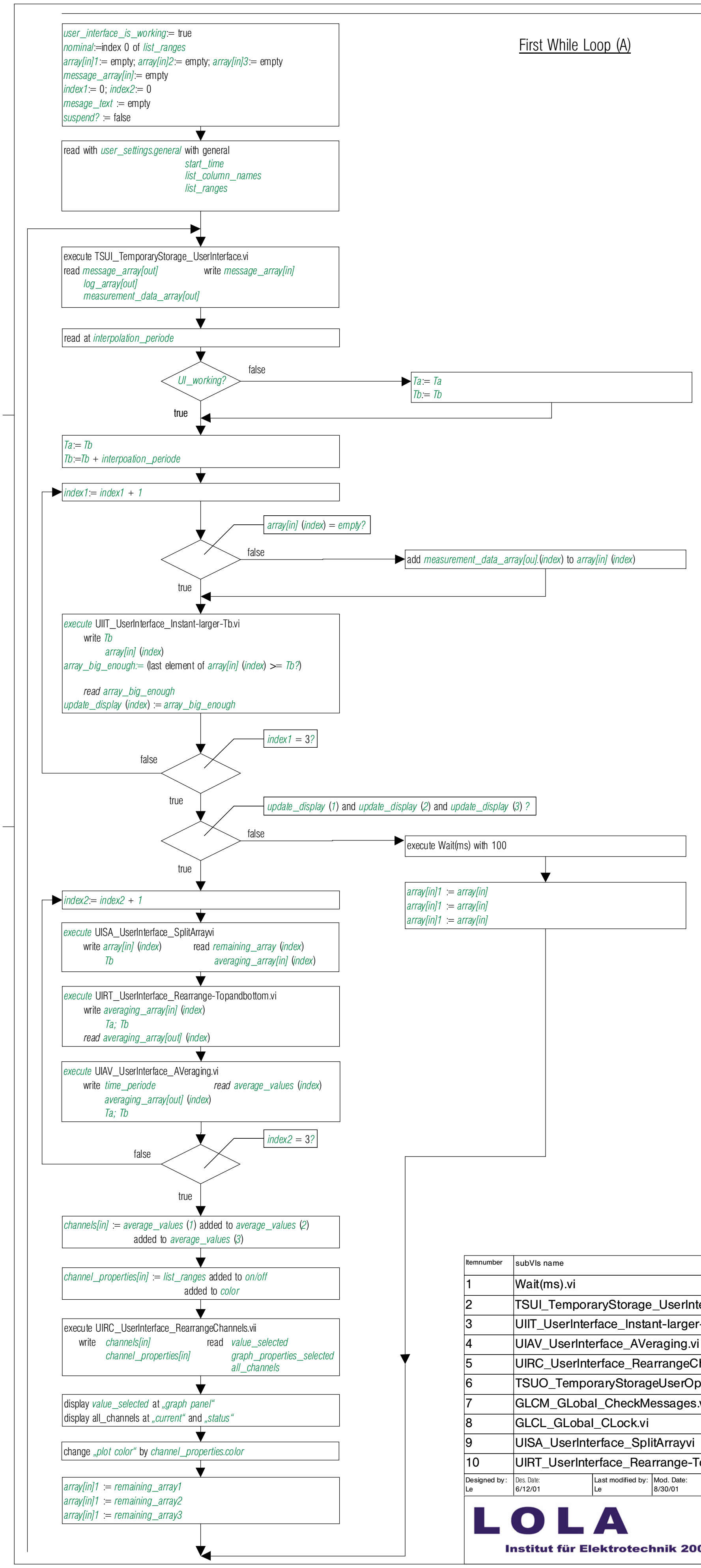
Itemnumber	subVIs name
1	TSM1_TemporaryStorage_Messages1.vi
2	TSM2_TemporaryStortage_Messages2.vi
3	SILO_SIgnageLOg.vi

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Mö	3/29/01	Mö	8/30/01	/home/kurt/g2415m/Flieádiag/fl-SICE_flowchart-SIgnageCEnter.sda

LOLA
Institut für Elektrotechnik 2001

SICE_SInageCEnter.vi

LOLA: Version 1.9



Variables:

from user_settings (Input):
with general
start_time
list_column_names
list_ranges

from UISA_UserInterface_SplitArray.vi
remaining_array
averageing_array[in]

from UIRT_UserInterface_Rearrange-Topandbottom.vi
averaging_array[out]

from GLCL_GLobal_CLock.vi
relative_instant

from UIAV_UserInterface_AVeraging.vi
average_values

to TSUI_TemporaryStorage_UserInterface.vi
message_array[in]

from TSUI_TemporaryStorage_UserInterface.vi
message_array[out] with
status, code, source

to TSUO_TemporaryStorage_UserOperation.vi
useroperation[in]
message_array[in]

from TSUI_TemporaryStorage_UserOperation.vi
message_array[out] with
status, code, source
log_array[out] with
status, code, source
measurement_data_array[out])

from GLCL_GLobal_CLock.vi
millisecond_timervalue
message_clock

to GLCL_GLobal_CLock.vi
start_time

from GLCM_GLobal_CheckMessages.vi
suspend?
stop?

to UIRC_UserInterface_RearrangeChannels.vi
channels[in]
channel_properties[in]

from UIRC_UserInterface_RearrangeChannels.vi
graph_properties
value_selected
all_channels

local:
Ta; Tb
array[in] 1; 2; 3
status
status_array
message_stop
message_ui; message_uo

Panel In/ Out:
interpolation_period
user_interface_working
graph_panel
current
status
stop
message_panel
display_mode

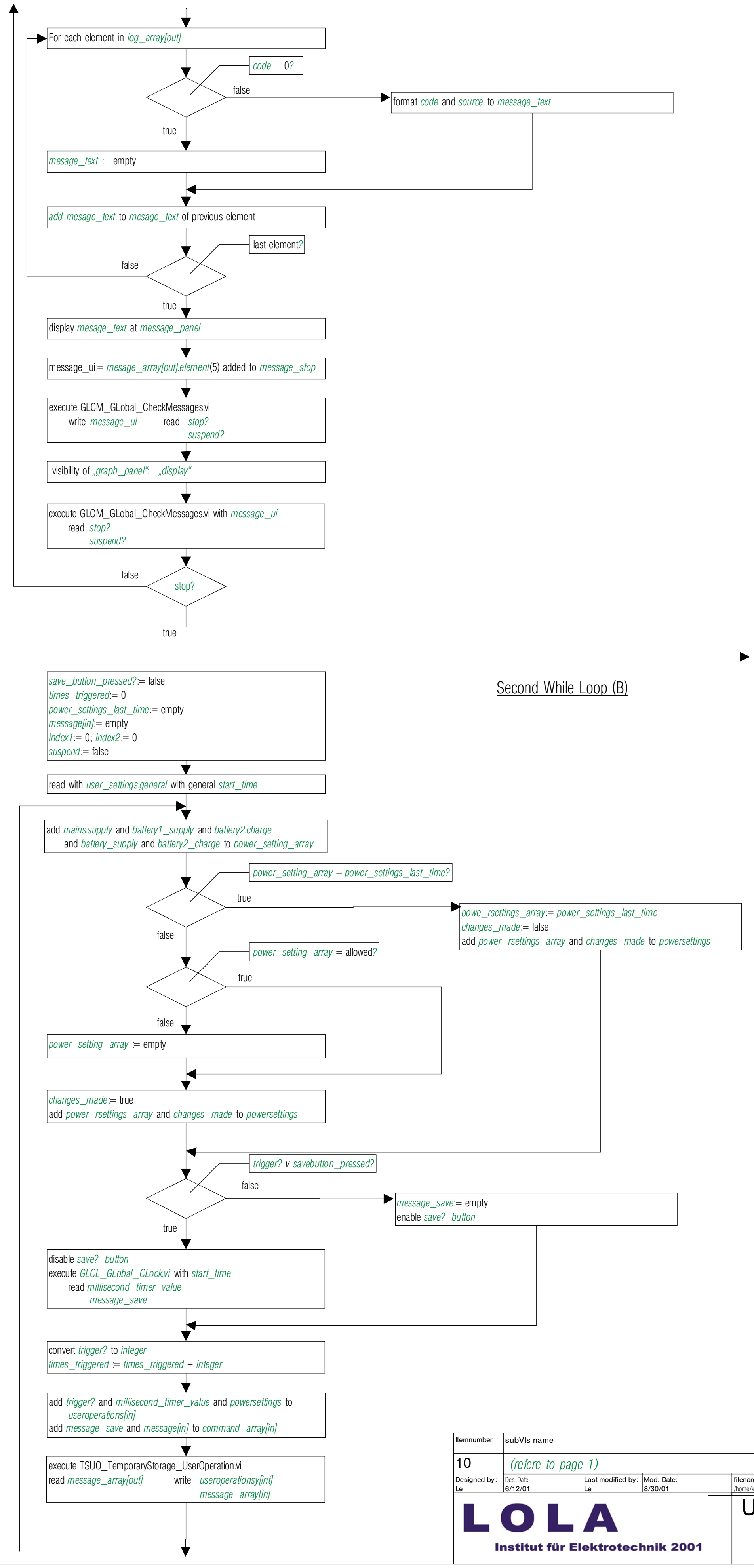
Itemnumber	subVIs name
1	Wait(ms).vi
2	TSUI_TemporaryStorage_UserInterface.vi
3	UIIT_UserInterface_Instant-larger-Tb.vi
4	UIAV_UserInterface_AVeraging.vi
5	UIRC_UserInterface_RearrangeChannels.vi
6	TSUO_TemporaryStorageUserOperation.vi
7	GLCM_GLobal_CheckMessages.vi
8	GLCL_GLobal_CLock.vi
9	UISA_UserInterface_SplitArrayvi
10	UIRT_UserInterface_Rearrange-Topandbottom.vi

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	6/12/01	Le	8/30/01	/home/kurt/g2415m/Fileâdiag/fl-UIUI_UserInterface_UserInterface.sda

LOLA
Institut für Elektrotechnik 2001

UIUI_UserInterface_UserInterface.vi

LOLA:
Version 1.9



Itemnumber	subVIs name	Variables:
10	(refere to page 1)	(refere to page 1)

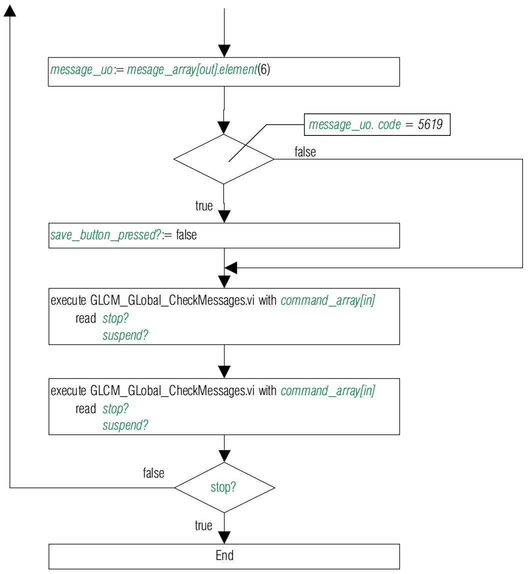
Designed by: Le | Des. Date: 6/12/01 | Last modified by: Le | Mod. Date: 8/30/01 | filename: /home/kurt/g2415m/Flleddiag/fl-UIUI_UserInterface_UserInterface.sda

LOLA

Institut für Elektrotechnik 2001

UIUI_UserInterface_UserInterface.vi

LOLA:
Version 1.9



Itemnumber	subVIs name	Variables:
10	(refere to page 1)	(refere to page 1)

Designed by:	Des. Date:	Last modified by:	Mod. Date:	filename:
Le	6/12/01	Le	8/30/01	/home/kurt/g2415m/Flieádiag/fl-UIUI_UserInterface_UserInterface.sda

LOLA
Institut für Elektrotechnik 2001

UIUI_UserInterface_UserInterface.vi

LOLA:
Version 1.9

Anhang B

PLC

B.1 Table X Inputs

B.2 Table User Input Codes

B.3 Table Y Outputs

B.4 Table PC Output Codes

B.5 State Diagram

B.6 Transition Condition

B.7 Output States

X Input							
num.	type	shortcut	name	description		connector assignment	
1	system input	LifeSign	lifesign PC	alternate signal sent by the Pc		5 inputs	E0.0-E0.
2	user input	Main_Su	main supply button	main supply on or off			
3	user input	Batt1_Su	battery1 supply button	battery1 supply on or off			
4	user input	Batt1_Ch	battery1 charge button	charge battery1 yes or no			
5	user input	Batt2_Su	battery2 supply button	battery2 supply on or off			
6	user input	Batt2_Ch	battery2 charge button	charge battery2 yes or no			
7	user input	Start	start button	enables system			
8	user input	Stopp	stopp button	pauses system			
9	system input	Close	close session	shuts down the system			
10	user input	Emergency	emergency button	emergency stopp		1 input	E0.5
11	system input	MotorProt_L	motor protection load	overcurrent relais between inverter and motor		uncertain	
12	system input	ErrorInv_L	erorr inverter load	error status of inverter (load)		1 input	E0.6
13	system input			programmable output of the load inverter		not planned	
14	system input			programmable output of the load inverter		not planned	
15	system input			programmable output of the load inverter		not planned	
16	system input	MotorProt_T	motor protection traction	overcurrent relais between inverter and motor		uncertain	
17	system input	ErrorInv_T	erorr inverter traction	error status of inverter (traction)		1 input	E0.7
18	system input			programmable output of the traction inverter		not planned	
19	system input			programmable output of the traction inverter		not planned	
20	system input			programmable output of the traction inverter		not planned	
21	system input	ResProt_L	resistor protection load	temperature control in load resistor		1 input	E1.0
22	system input	CheckSh1	check shunt1	to control system status		1 input	E1.1
23	system input	CheckSh2	check shunt2	to control system status		1 input	E1.2
24	system input	CheckSh3	check shunt3	to control system status		uncertain	
25	system input	CheckSh4	check shunt4	to control system status		uncertain	
						used 11 of	16

Tabelle B.1: X Inputs

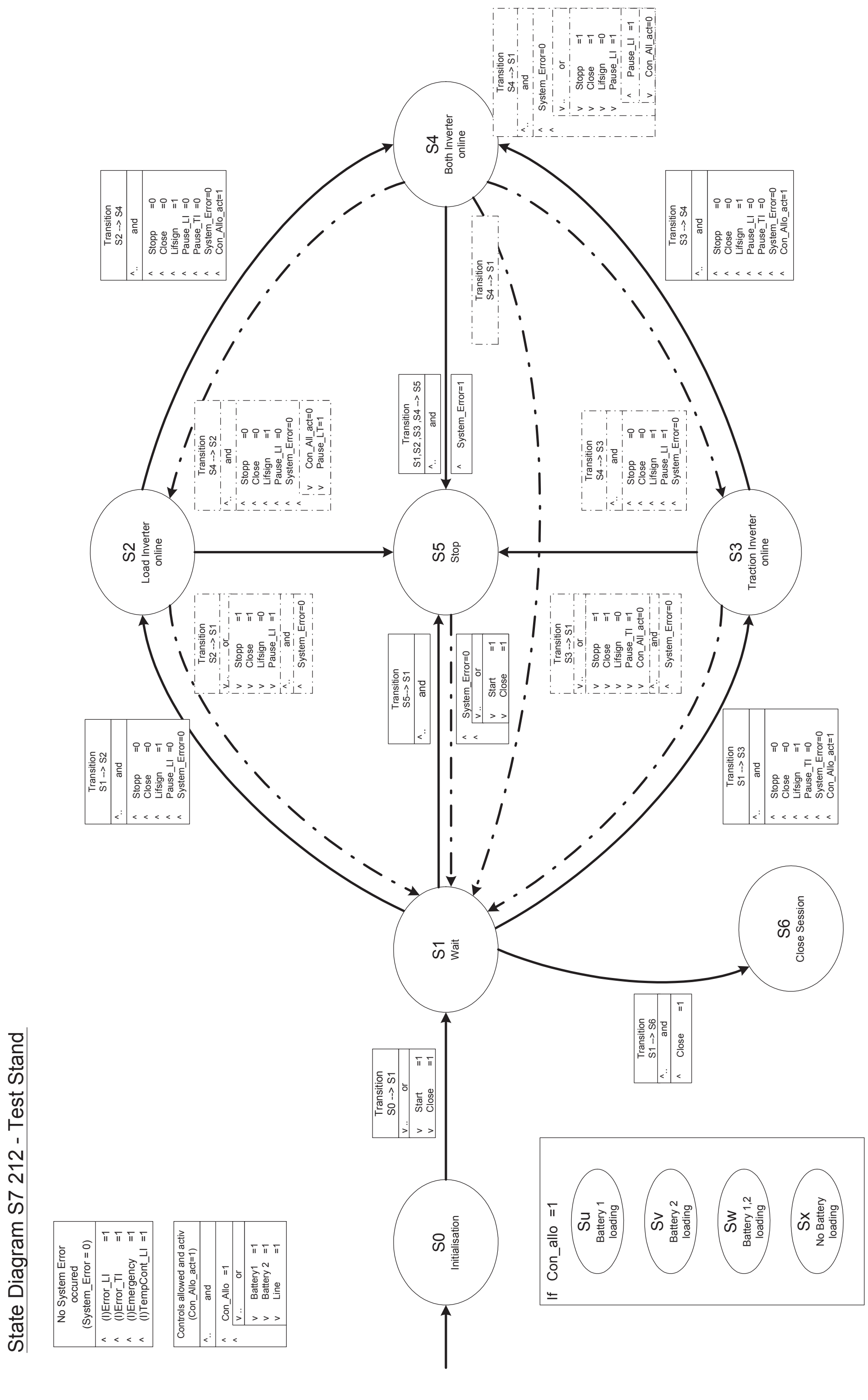


Abbildung B.1: State Diagram

Nr.	*5 Bits-Code*					*Explanation*
0	0	0	0	0	0	No supply for the Traction Drive (T. D.) chosen
1	0	0	0	0	1	Line supply for T. D.
2	0	0	0	1	0	Battery 1 supply for T. D.
3	0	0	0	1	1	Start button
5	0	0	1	0	1	Line supply for T. D. and charge battery 1
6	0	0	1	1	0	Close session
7	0	0	1	1	1	Lifesign
8	0	1	0	0	0	Battery 2 supply for T. D.
9	0	1	0	0	1	Pause load inverter
10	0	1	0	1	0	Pause traction inverter
11	0	1	0	1	1	Reset all commands
12	0	1	1	0	0	Battery 2 supply for T. D. and charge battery 1
13	0	1	1	0	1	Reset Pause load inverter
14	0	1	1	1	0	Reset Pause T. D.
15	0	1	1	1	1	Stop button
17	0	1	0	0	1	Line supply for T. D. and charge battery 2
18	0	1	0	1	0	Battery 1 supply for T. D. and charge battery 2
21	1	0	1	0	1	Line supply for T. D. and charge battery 1/2
22	1	0	1	1	0	Reverse Lifesign

Tabelle B.2: Used 5-bit Input Codes

Y Output						
num.	type	shortcut	name	description	connector assignment	
1	machine output	InvShunt_L	inverter shunt traction	shunt to en/disable powersupply for traction inverter	1 output	A0.0
2	machine output	InvShunt_T	inverter shunt load	shunt to en/disable powersupply for load inverter	1 output	A0.1
3	machine output	Main_Su_Sh	main supply shunt	shunt to en/disable main supply for traction inverter	with permission	
4	machine output	Batt1_Su_Sh	battery1 supply shunt	shunt to en/disable battery1 supply for traction inverter		
5	machine output	Batt1_Ch_Sh	battery1 charge shunt	shunt to en/disable charging of battery1		
6	machine output	Batt2_Su_Sh	battery2 supply shunt	shunt to en/disable battery2 supply for traction inverter		
7	machine output	Batt2_Ch_Sh	battery2 charge shunt	shunt to en/disable charging of battery2		
8	machine output	MeachBreak	mechanical break	...	not planned	
9	machine output	EnCont_L	enable controller (load)	enable controller in load inverter	1 output	A0.2
10	machine output	Set_TRIP_L	set TRIP (load)	sets TRIP true (load inverter)	1 output	A0.3
11	machine output	Reset_TRIP_L	reset TRIP (load)	sets TRIP false --> true (load inverter)	1 output	A0.4
12	machine output				uncertain	
13	machine output				uncertain	
14	machine output				uncertain	
15	machine output	EnCont_T	enable controller (traction)	enable controller in traction inverter	1 output	A0.5
16	machine output	Set_TRIP_T	set TRIP (traction)	sets TRIP true (traction inverter)	1 output	A1.0
17	machine output	Reset_TRIP_T	reset TRIP(traction)	sets TRIP false --> true (traction inverter)		A0.4
18	machine output				uncertain	
19	machine output				uncertain	
20	machine output				uncertain	
21	machine output	PC_Out	pc output	information output to Pc	3 outputs	A1.1- A1
22	machine output	Supply24V_Sh	supply 24V shunt	shunt to en/disable 24 V external supply	1 input	A1.4
					used 11 of 14	

Tabelle B.3: Y Outputs

Number	*3 Bits-Code*	*Explanation*
6	0 0 0	State Close (like no current)
0	0 0 1	State Initialize
1	0 1 0	State Wait
2	0 1 1	State Load Inverter Online
3	1 0 0	State Traction Inverter Online
4	1 0 1	State Both Inverter Online
5	1 1 0	State Stop

Tabelle B.4: Used 3-bit Output Codes

From State	*Jump to State*	*Transition Condition*
S0 Initialization	S1 Wait	$\overline{\text{System\_Error}} \cap (\text{Start} \cup \text{Close})$
S1 Wait	S2 Load Inverter Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_LI})} \cap \text{Lifesign}$
	S3 Traction Inv. Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_TI})} \cap (\overline{\text{No\_Supply}} \cap \text{Lifesign})$
	S5 Stop	System_Error
	S6 Close Session	Close
S2 Load Inverter Online	S1 Wait	$\overline{\text{System\_Error}} \cap (\text{Close} \cup \text{Stop} \cup \text{Pause\_LI} \cup \overline{\text{Lifsign}})$
	S4 Both Inv. Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_TI} \cup \text{Pause\_LI})} \cap (\overline{\text{No\_Supply}} \cap \text{Lifesign})$
	S5 Stop	System_Error
S3 Traction Inv. Online	S1 Wait	$\overline{\text{System\_Error}} \cap (\text{Close} \cup \text{Stop} \cup \text{Pause\_TI} \cup \overline{\text{Lifsign}} \cup \overline{\text{No\_Supply}})$
	S4 Both Inv. Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_TI} \cup \text{Pause\_LI})} \cap (\overline{\text{No\_Supply}} \cap \text{Lifesign})$
	S5 Stop	System_Error
S4 Both Inverter Online	S1 Wait	$\overline{\text{System\_Error}} \cap (\text{Close} \cup \text{Stop} \cup (\text{Pause\_LI} \cap \text{Pause\_TI}) \cup \overline{\text{Lifsign}} \cup \overline{\text{No\_Supply}})$
	S2 Load Inv. Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_LI})} \cap \text{Lifesign} \cap (Pause\_TI \cup No\_Supply)$
	S3 Traction Inv. Online	$\overline{(\text{System\_Error} \cup \text{Close} \cup \text{Stop} \cup \text{Pause\_LI})} \cap \text{Lifesign} \cap Pause\_LI)$
	S5 Stop	System_Error

Tabelle B.5: Transition Conditions

S0						S1					
InvShunt_	LOW	0		V400.(	A0.(	InvShunt_	LOW	0		V400.(	A0.(
InvShunt_	LOW	0		V400.1	A0.1	InvShunt_		X		V400.1	A0.1
EnCont_L	LOW	0		V400.2	A0.2	EnCont_L	LOW	0		V400.2	A0.2
Set_TRIP_	LOW	0		V400.3	A0.3	Set_TRIP_	HIGH	1		V400.3	A0.3
Reset_TRIP_	LOW	0		V400.4	A0.4	Reset_TRIP_	HIGH	1		V400.4	A0.4
Reset_TRIP_	LOW	0		V401.(	A0.4	Reset_TRIP_	HIGH	1		V401.(	A0.4
Set_TRIP_'	LOW	0		V400.4	A1.(	EnCont_T	LOW	0		V400.5	A0.5
EnCont_T	LOW	0		V400.5	A0.5	Set_TRIP_'	HIGH	1		V400.4	A1.(
PC_Out	XXX	0		V401.1	A1.1	PC_Out	XXX	0		V401.1	A1.1
		0		V401.2	A1.2			1		V401.2	A1.2
		1		V401.3	A1.3			0		V401.3	A1.3
Supply24V_Sh	HIGH	1		V401.4	A1.4	Supply24V_Sh	HIGH	1		V401.4	A1.4
S2						S3					
InvShunt_	HIGH	1		V400.(	A0.(	InvShunt_	LOW	0		V400.(	A0.(
InvShunt_		X		V400.1	A0.1	InvShunt_		X		V400.1	A0.1
EnCont_L	HIGH	1		V400.2	A0.2	EnCont_L	HIGH	1		V400.2	A0.2
Set_TRIP_	HIGH	1		V400.3	A0.3	Set_TRIP_	LOW	1		V400.3	A0.3
Reset_TRIP_	LOW	0		V400.4	A0.4	Reset_TRIP_	LOW	0		V400.4	A0.4
Reset_TRIP_	LOW	0		V401.(	A0.4	Reset_TRIP_	LOW	0		V401.(	A0.4
EnCont_T	LOW	0		V400.5	A0.5	EnCont_T	HIGH	1		V400.5	A0.5
Set_TRIP_'	HIGH	1		V400.4	A1.(	Set_TRIP_'	HIGH	1		V400.4	A1.(
PC_Out	XXX	0		V401.1	A1.1	PC_Out	XXX	1		V401.1	A1.1
		1		V401.2	A1.2			0		V401.2	A1.2
		1		V401.3	A1.3			0		V401.3	A1.3
Supply24V_Sh	HIGH	1		V401.4	A1.4	Supply24V_Sh	HIGH	1		V401.4	A1.4
S4						S5					
InvShunt_	HIGH	1		V400.(	A0.(	InvShunt_	LOW	0		V400.(	A0.(
InvShunt_		X		V400.1	A0.1	InvShunt_	LOW	0		V400.1	A0.1
EnCont_L	HIGH	1		V400.2	A0.2	EnCont_L	LOW	0		V400.2	A0.2
Set_TRIP_	HIGH	1		V400.3	A0.3	Set_TRIP_	LOW	0		V400.3	A0.3
Reset_TRIP_	LOW	0		V400.4	A0.4	Reset_TRIP_	LOW	0		V400.4	A0.4
Reset_TRIP_	LOW	0		V401.(	A0.4	Reset_TRIP_	LOW	0		V401.(	A0.4
EnCont_T	HIGH	1		V400.5	A0.5	EnCont_T	LOW	0		V400.5	A0.5
Set_TRIP_'	HIGH	1		V400.4	A1.(	Set_TRIP_'	LOW	0		V400.4	A1.(
PC_Out	XXX	1		V401.1	A1.1	PC_Out	XXX	1		V401.1	A1.1
		0		V401.2	A1.2			1		V401.2	A1.2
		1		V401.3	A1.3			0		V401.3	A1.3
Supply24V_Sh	HIGH	1		V401.4	A1.4	Supply24V_Sh	HIGH	1		V401.4	A1.4
S6											
InvShunt_	LOW	0		V400.(	A0.(						
InvShunt_	LOW	0		V400.1	A0.1						
EnCont_L	LOW	0		V400.2	A0.2						
Set_TRIP_	LOW	0		V400.3	A0.3						
Reset_TRIP_	LOW	0		V400.4	A0.4						
Reset_TRIP_	LOW	0		V401.(	A0.4						
EnCont_T	LOW	0		V400.5	A0.5						
Set_TRIP_'	LOW	0		V400.4	A1.(						
PC_Out	XXX	0		V401.1	A1.1						
		0		V401.2	A1.2						
		0		V401.3	A1.3						
Supply24V_Sh	LOW	0		V401.4	A1.4						

Tabelle B.6: Output States

Anhang C

Error Codes

C.1 Labview Error Codes

C.2 Poweranalyzer Error Codes

C.3 User Error Codes

Error codes in general:

Name	Code Range
Mathematics Error Codes	-23001 to -23054
Signal Processing Error Codes	-20001 to -20065
Data Acquisition VI Error Codes	-10001 to -10920
AppleEvent Error Codes	-1700 to -1719
Instrument Driver Error Codes	-1200 to -13xx
PPC Error Codes	-900 to -932
G Function Error Codes	0 to 85
Labview Specific PPC Error Codes	1 to 5
GPIB Error Codes	0 to 32
TCP and UDP Error Codes	53 to 66
Serial Port Error Codes	61 to 65
PowerAnalyzer	102 to 350
Labview Specific Apple Error Codes	1000 to 1004
MATLAB and HiQ Error Codes	1046 to 1050
PowerAnalyzer	2200 to 4322
User spezified Error Codes	5000 to 9999
Command codes:	5000 to 5999
Warning codes:	6000 to 6999
Error codes:	7000 to 9999
DDE Error Codes	14001 to 14020

Tabelle C.1: Error-codes used in LOLA

PowerAnalyzer spezified Error Codes

Classification		Description	Code
System		Syntax Error	102
		Command Header Error	110
		Header Seperator Error	111
		Character Data Error	140
		Data Out Of Range	222
		Measurement Underflow	2204
		Measurement Overflow	2207
		Queue Overflow	350
		Input Signal Overload Or Underload	2200
Channels		Error: 1-Phase instrument - requested operati	3200
	Current channel 1	Overload	4111
		Underload	4112
	Voltage channel 1	Oveload	4121
		Underload	4222
	Current channel 2	Overload	4211
		Underload	4212
	Voltage channel 2	Oveload	4221
		Underload	4222
	Current channel 3	Overload	4311
		Underload	4312
	Voltage channel 3	Oveload	4321
		Underload	4322

Tabelle C.2: Error-codes used in LOLA

User spezified Error Codes:

Classification		Description	Code
Command codes TO:	System	Reactivate module	5000
		User pressed STOP_button - STOP!	5099
	RDRD		5001
	RSRD		5101
	PARD		5201
	SASH		5301
		User pressed SAVE_button - SAVE!	5311
	RSWS		5401
		Send comands	5411
	UIUI		5501
	UIUO		5601
		Enable SAVE_button	5619
		Enable System_buttons	5629
Warning_from codes:	DAQ	Scan backlog above limit	6011
	RSRD	Temperature above limit	6111
	SASH	Saving in progress	6311
		Saving finished	6319
		not enough data in memory yet	6320
	RSWS	Sending comands in progress	6411
		Transmission completed	6419
	System	Not all modules have been closed yet	6990
Error codes:	RDRD	Time_Out occurred	7010
	System	End Program	9999

Tabelle C.3: Error-codes used in LOLA

Anhang D

Snapshots

D.1 Load Inverter - Staiger&Mohilo - Dewerack

D.2 First experimental setup

D.3 Test Bench with Experimental Setup

Abbildung D.1: Load Inverter - Staiger&Mohilo - Dewerack

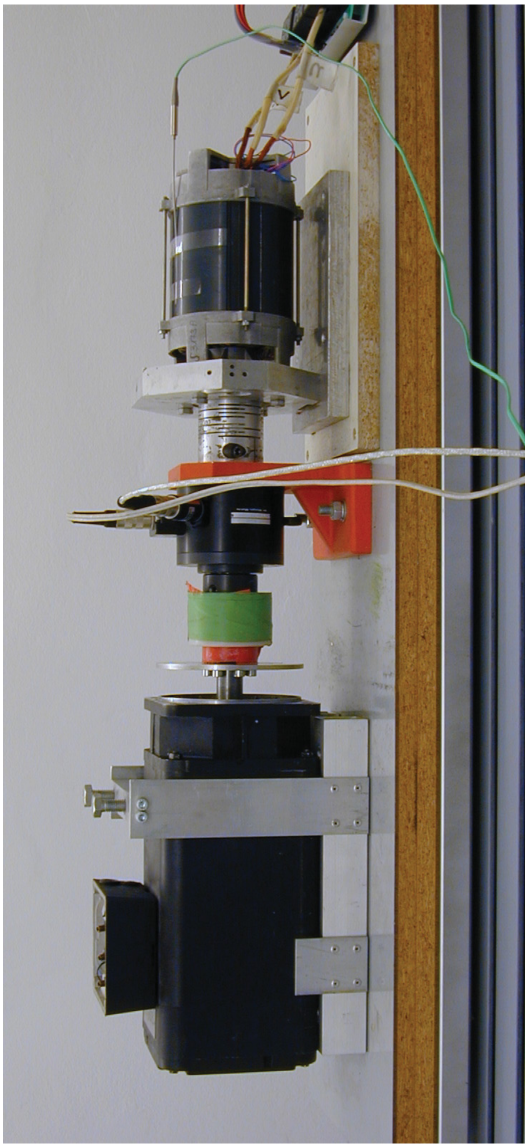


Abbildung D.2: First experimental setup

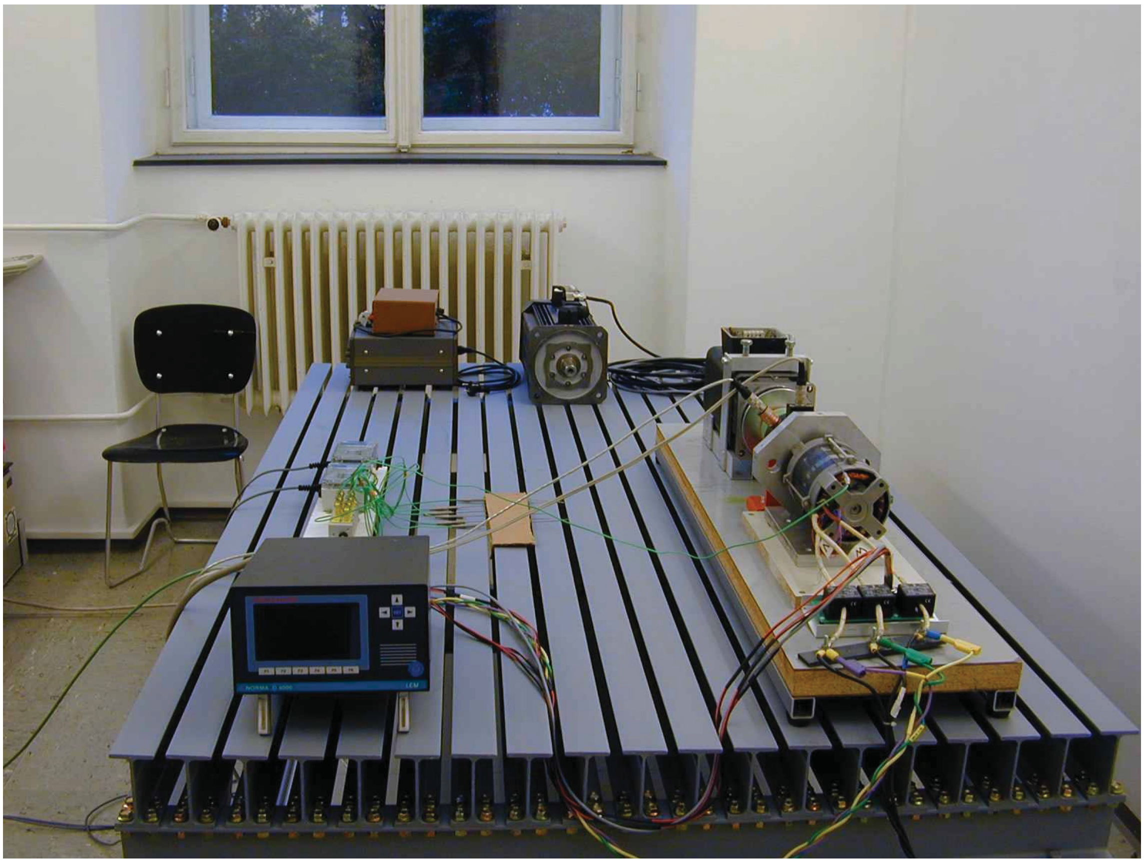
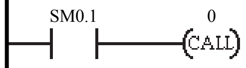


Abbildung D.3: Test Bench with Experimental Setup and Poweranalyzer

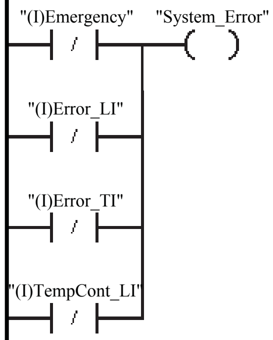
Anhang E

PLC Ladder Diagram

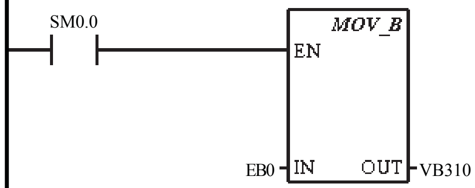
Netzwerk 1 Firstjumptoinitialization(Isexecutedonlyonceatstartoftheprogram)



Netzwerk 2 checkiferroroccuredinsystem!!

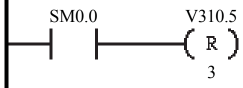


Netzwerk 3 MainprogramBEGINloadvalues

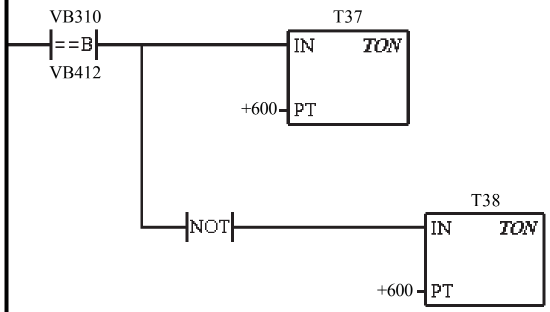


LadderDiagramforPLC-program"SecurityConcept" 28.08.01

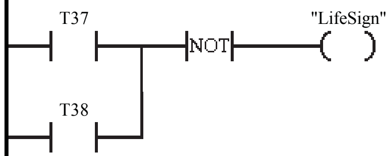
Netzwerk 4 truncatefirstthreesignificantbits



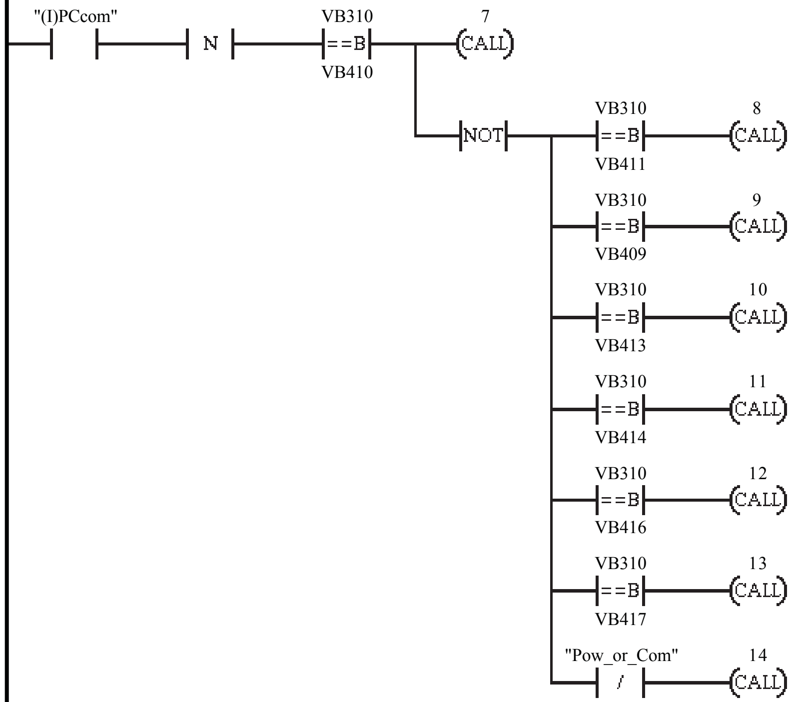
Netzwerk 5 ?PConline?(checksalternatingPCsignal)



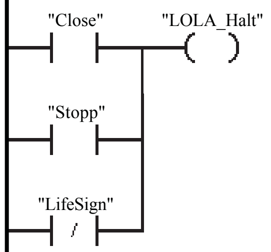
Netzwerk 6 ConditiontosetLifesigntofalse,ifPCisoff-line



Netzwerk 7 Iterpretation LOLA Messages and Check if User_Input was a Powerseting or a general Command



Netzwerk 8 Any stopp condition by LOLA?



Ladder Diagram for PLC-program "Security Concept" 28.08.01

Netzwerk 9 Jump to State 0

Netzwerk 10 Jump to State 1

Netzwerk 11 Jump to State 2

Netzwerk 12 Jump to State 3

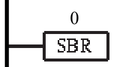
Netzwerk 13 Jump to State 4

Netzwerk 14 Jump to State 5

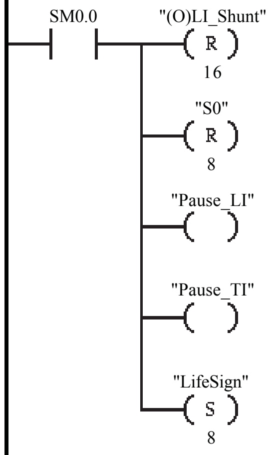
Netzwerk 15 Jump to State 6

Netzwerk 16

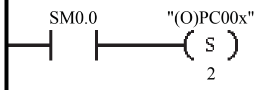
Netzwerk 17 BEGIN "state 0" %INITIALIZATION%



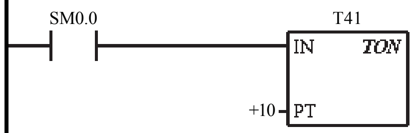
Netzwerk 18 Setalloutputsand"state"toLOW



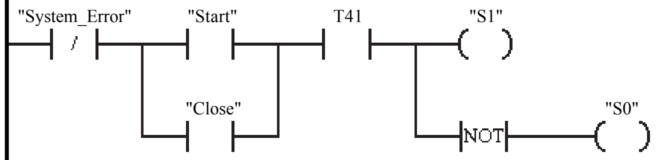
Netzwerk 19 Setoutputaccordingtostate0



Netzwerk 20 Wait for Inverter to be ready 10sec

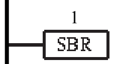


Netzwerk 21 Check if transition condition is met for S0 or S1



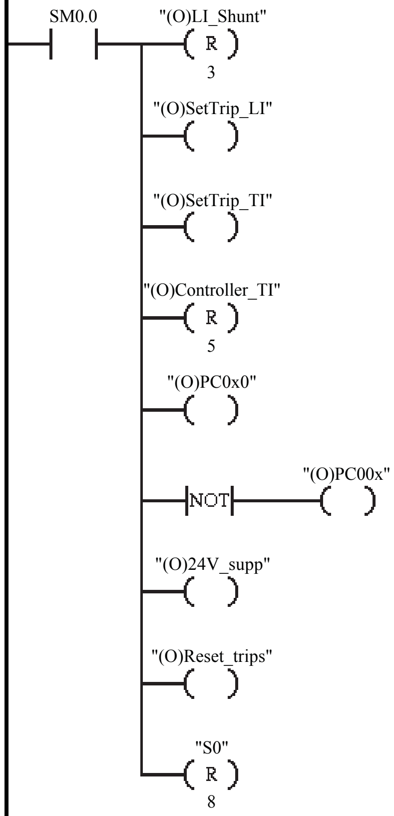
Netzwerk 22 ENDE State0

Netzwerk 23 BEGIN "state 1" %WAIT%

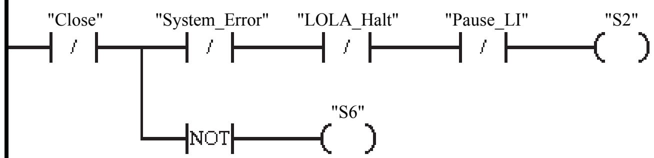


Ladder Diagram for PLC-program "Security Concept" 28.08.01

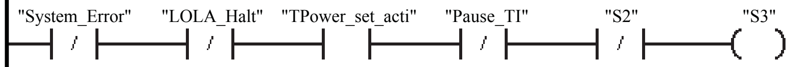
Netzwerk 24 Set output according to state 1 and all "state" variable to LOW



Netzwerk 25 Check if transition condition is met for state S6 and S2



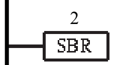
Netzwerk 26 Check if transition condition is met for S3



Netzwerk 27 Check if transition condition is met for S1

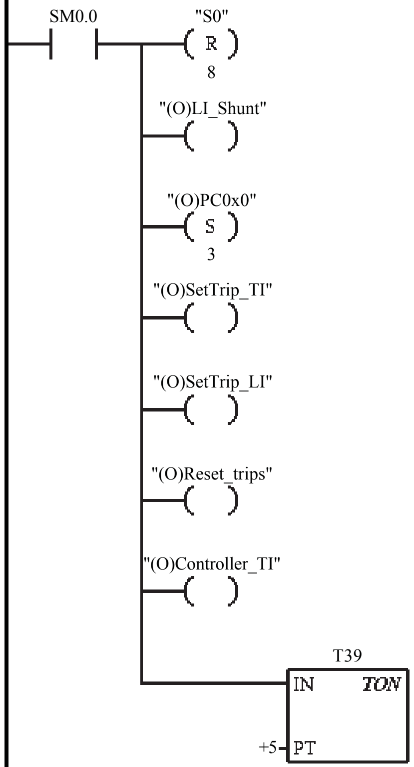
Netzwerk 28 ENDE State1

Netzwerk 29 Begin "state 2" %LOAD INVERTER ONLINE%

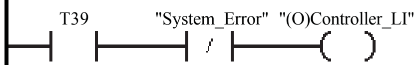


Ladder Diagram for PLC-program "Security Concept" 28.08.01

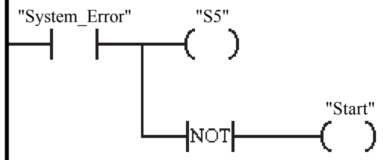
Netzwerk 30 Set all "state" variblen to LOW and Set output according to state 2 without control release



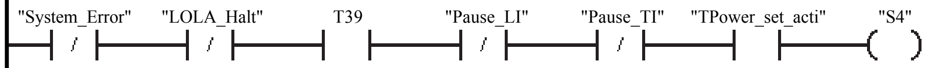
Netzwerk 31 Release controller for Load inverter after 500ms



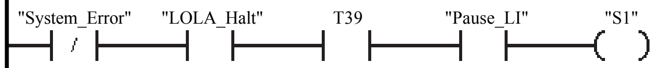
Netzwerk 32 check if transition condition is met for state S5



Netzwerk 33 check if transition condition is met for S4



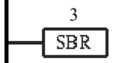
Netzwerk 34



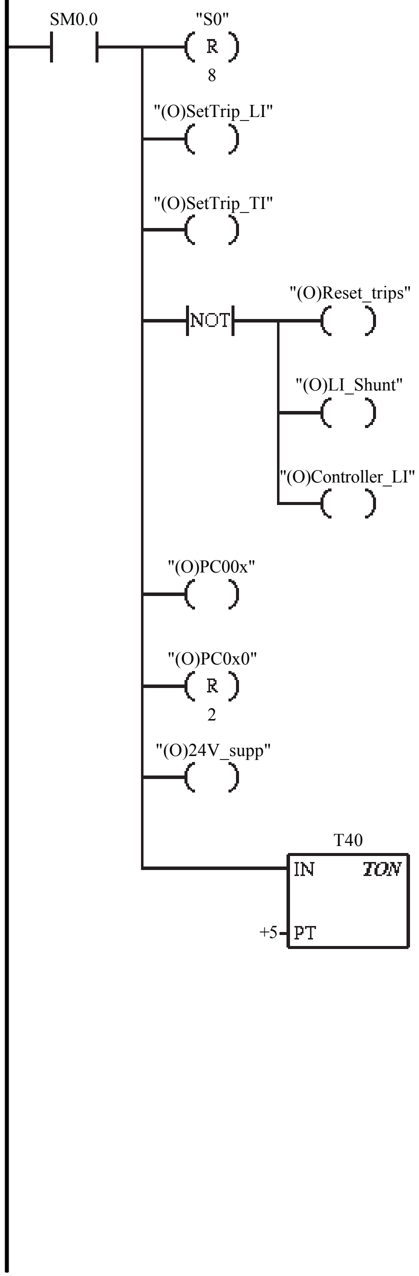
Netzwerk 35 check if transition condition is met for S2

Netzwerk 36 ENDE State2

Netzwerk 37 Begin "state 3" %TRACTION INVERTER ONLINE%

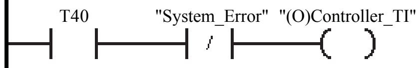


Netzwerk 38 Set all "state"-variables to LOW and Set output according to state 3 without control release

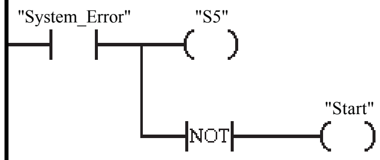


Ladder Diagram for PLC-program "Security Concept" 28.08.01

Netzwerk 39 Release traction inverter controller after 500ms



Netzwerk 40 Check if transition condition is met for state S5



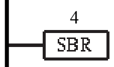
Netzwerk 41 Check if transition condition is met for S4

Netzwerk 42 Check if transition condition is met for S1

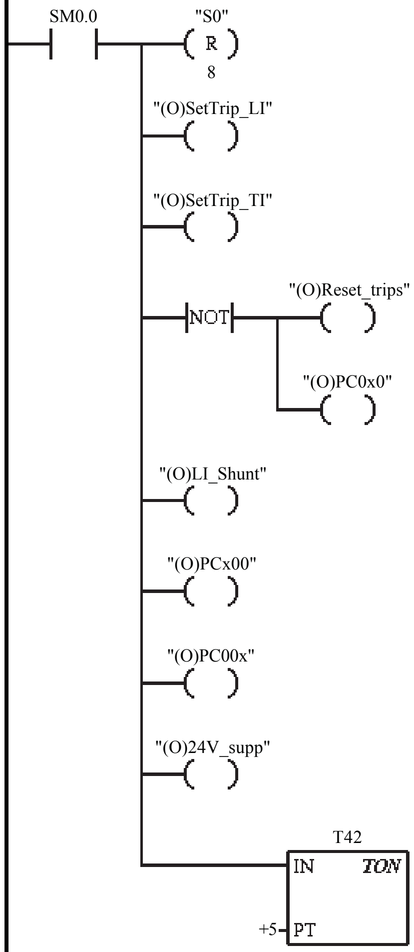
Netzwerk 43 Check if transition condition is met for S2

Netzwerk 44 ENDE State3

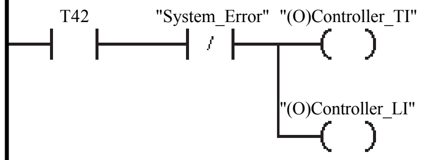
Netzwerk 45 BEGIN "state 4" %BOTH INVERTER ONLINE%



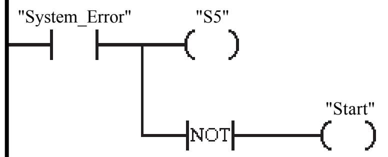
Netzwerk 46 Set all "state"-variables to LOW and Set output according to state 4 without control release



Netzwerk 47 Releaseinvertercontrollersafter500ms



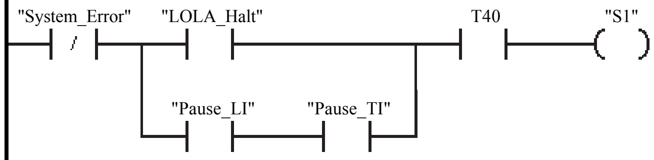
Netzwerk 48 Checkiftransitionconditionismetforstate S5



Netzwerk 49 CheckiftransitionconditionismetforS2

Netzwerk 50 CheckiftransitionconditionismetforS3

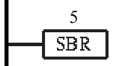
Netzwerk 51 CheckiftransitionconditionismetforS1



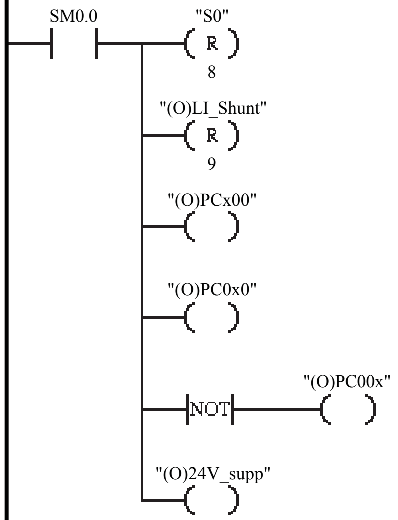
Netzwerk 52 CheckiftransitionconditionismetforS2

Netzwerk 53 ENDEState4

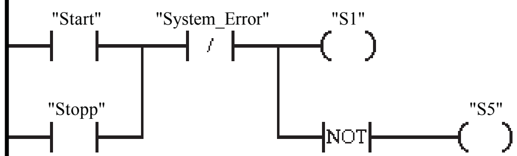
Netzwerk 54 BEGIN"state5"%StoppState!!%



Netzwerk 55 Set all "state"-variables to LOW and Set output according to state 5



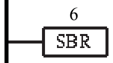
Netzwerk 56 Check if transition condition is met for state S5



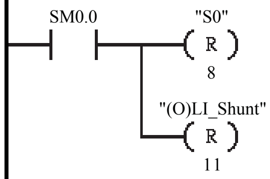
Netzwerk 57 ENDE State5

Ladder Diagram for PLC-program "Security Concept" **28.08.01**

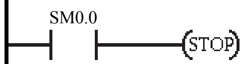
Netzwerk 58 Begin "state 6" %CLOSE ALL!!%



Netzwerk 59 Set all "state"-variables to LOW and Set output according to state 6

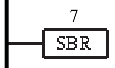


Netzwerk 60 check if transition condition is met for state S5

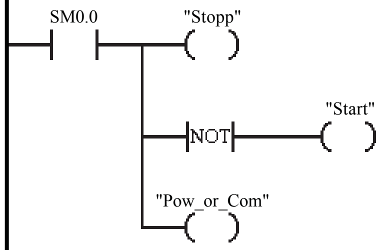


Netzwerk 61 ENDE State6

Netzwerk 62 STOP!! Command

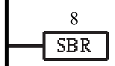


Netzwerk 63

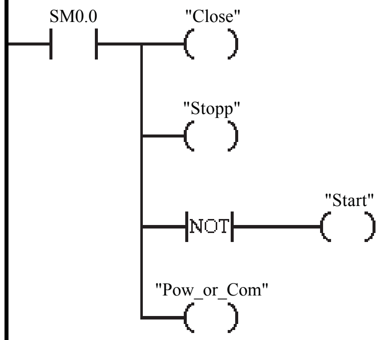


Netzwerk 64

Netzwerk 65 CLOSE!!Command

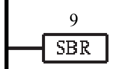


Netzwerk 66 Close

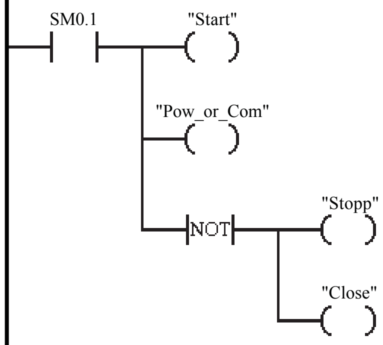


Netzwerk 67 Close

Netzwerk 68 START!! Command

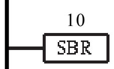


Netzwerk 69 START

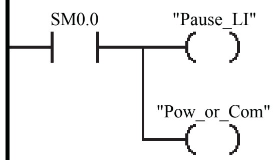


Netzwerk 70 Start

Netzwerk 71 PAUSE_LI!!Command

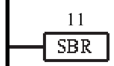


Netzwerk 72 PauseLI

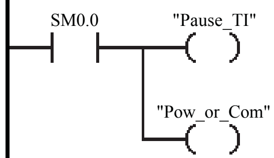


Netzwerk 73 PauseLI

Netzwerk 74 PAUSE_TI!!Command

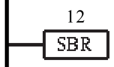


Netzwerk 75 PauseTI

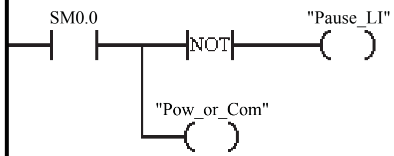


Netzwerk 76 PauseTI

Netzwerk 77 RESETPAUSELI!!command



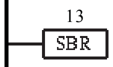
Netzwerk 78 ResetPauseLI



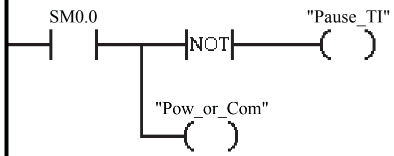
Netzwerk 79 ResetPauseLI

LadderDiagramforPLC-program"SecurityConcept" 28.08.01

Netzwerk 80 RESETPAUSETI!! command

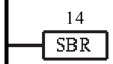


Netzwerk 81 ResetPauseTI

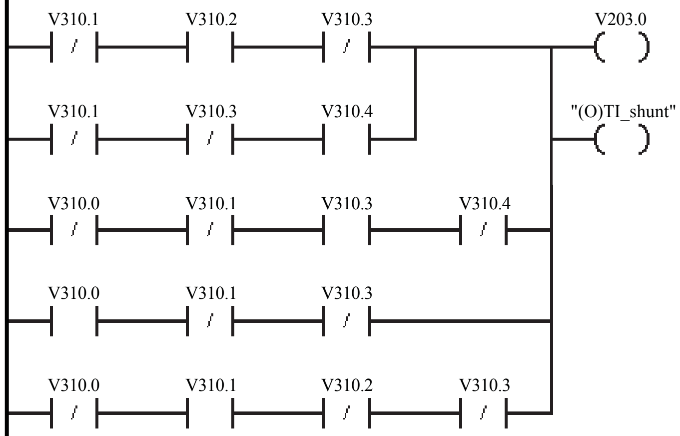


Netzwerk 82 ResetPauseTI

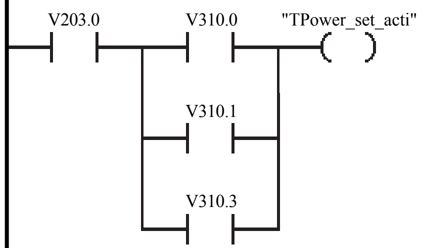
Netzwerk 83 BEGINcheckinputsfromLOLAfortractionsupplymessage



Netzwerk 84 Message valid



Netzwerk 85 MessagevalidandTPowersupplyison



Netzwerk 86 END message from LOLA check


```
//Variable definition for interpreting User-Inputs
//
VB400 0     //no supply for traction chosen
VB401 1     //line supply (tr.)
VB402 5     //line supply (tr.) and charge batt1
VB403 17    //line supply (tr.) and charge batt2
VB404 21    //line supply (tr.) and charge batt1/ batt2
VB405 2     //batt1 supply (tr.)
VB406 18    //batt1 suplly (tr.) and charge batt2
VB407 8     //batt2 supply (tr.)
VB408 12    //batt2 supply (tr.) and charge batt1
VB409 3     //start button
VB410 15    //stop button
VB411 6     //close session
VB412 7     //lifesign
VB413 9     //pause load inverter
VB414 10    //pause traction inverter
VB415 11    //Reset all commands
VB416 13    //Reset Pause load inverter
VB417 14    //Reset Pause traction inverter
VB418 22    //Revers lifsign
```

Symbolischer Name	Adresse	Kommentar
Sys_OUT	AD0	outputs of the PLC S7 212
TPower_set_Allo	V203.0	If powersetting is allowed
TPower_set_acti	V203.1	If powersetting is allowed and supply on
Pause_LI	V203.2	Should load inverter be paused
Pause_TI	V203.3	Should traction inverter be paused
Pow_or_Com	V203.4	determins if bits sent by LOLA is a command
System_Error	V203.5	System error?
LOLA_Halt	V203.6	Has LOLA sent a halt command?
Reset_All	V203.7	Reset all commands
Main	V200.0	main supply button {LV (LabView)}
Batt1_Su	V200.1	battery1 supply button {LV}
Batt1_Ch	V200.2	battery1 charge button {LV}
Batt2_Su	V200.3	battery2 supply button {LV}
Batt2_Ch	V200.4	battery2 charge button {LV}
Start	V200.5	start button {LV}
Stopp	V200.6	stopp button {LV}
Close	V200.7	close session {LV}
LifeSign	V201.0	lifsign PC {LV}
Time_switch37	V201.1	To enable wait before controller release
Time_switch38	V201.2	
Time_switch39	V201.3	
Time_switch40	V201.4	
Time_switch41	V201.5	
Time_switch42	V201.6	
state	VB202	state variable detrmines the state the system is in
S0	V202.0	state variable if TRUE main programm jumps to sub programm
S1	V202.1	state variable
S2	V202.2	state variable
S3	V202.3	state variable
S4	V202.4	state variable
S5	V202.5	state variable
S6	V202.6	state variable
S7	V202.7	state variable
(I)PCx0000	E0.0	
(I)PC0x000	E0.1	

LadderDiagramforPLC-program"SecurityConcept" 28.08.01

Symbolischer Name	Adresse	Kommentar
(I)PC00x00	E0.2	
(I)PC000x0	E0.3	
(I)PC0000x	E0.4	
(I)PCcom	E0.5	
(I)Emergency	E0.6	Emergency Button Input 1=no error
(I)Error_LI	E0.7	Error at load inverter 1= no error
(I)Error_TI	E1.0	Error at traction inverter 1=no error
(I)TempCont_LI	E1.1	Temperature control at
(I)ChechShuntL	E1.2	
(I)CheckshuntT	E1.3	
(O)LI_Shunt	A0.0	Load inverter shun
(O)TI_shunt	A0.1	Traction inverter shunt
(O)Controller_LI	A0.2	Enable controller for load inverter
(O)SetTrip_LI	A0.3	
(O)Reset_trips	A0.4	Reset both Trips
(O)Controller_TI	A0.5	Enable controller for traction inverter
(O)SetTrip_TI	A1.0	
(O)PCx00	A1.1	
(O)PC0x0	A1.2	
(O)PC00x	A1.3	
(O)24V_supp	A1.4	

Element **Netzwerk / Operation**

Element	Netzwerk	Operation		Netzwerk	Operation	
"(I)PCcom"	7	—| |—				
"(I)Emergency"	2	—|/|—				
"(I)Error_LI"	2	—|/|—				
"(I)Error_TI"	2	—|/|—				
"(I)TempCont_LI"	2	—|/|—				
EB0	3	MOV_B				
"(O)LI_Shunt"	18	—(R)				
	24	—(R)				
	30	—( )				
	38	—( )				
	46	—( )				
	55	—(R)				
	59	—(R)				
"(O)TI_shunt"	18	—(R)	(Bereich)			
	24	—(R)	(Bereich)			
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
	84	—( )				
"(O)Controller_LI"	18	—(R)	(Bereich)			
	24	—(R)	(Bereich)			
	31	—( )				
	38	—( )				
	47	—( )				
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
"(O)SetTrip_LI"	18	—(R)	(Bereich)			
	24	—( )				
	30	—( )				
	38	—( )				
	46	—( )				
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
"(O)Reset_trips"	18	—(R)	(Bereich)			
	24	—( )				

Element **Netzwerk / Operation**

Element	Netzwerk	Operation		Netzwerk	Operation	
	30	—( )				
	38	—( )				
	46	—( )				
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
"(O)Controller_TI"	18	—(R)	(Bereich)			
	24	—(R)				
	30	—( )				
	39	—( )				
	47	—( )				
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
A0.6	18	—(R)	(Bereich)	24	—(R)	(Bereich)
	55	—(R)	(Bereich)	59	—(R)	(Bereich)
A0.7	18	—(R)	(Bereich)	24	—(R)	(Bereich)
	55	—(R)	(Bereich)	59	—(R)	(Bereich)
"(O)SetTrip_TI"	18	—(R)	(Bereich)			
	24	—( )				
	24	—(R)	(Bereich)			
	30	—( )				
	38	—( )				
	46	—( )				
	55	—(R)	(Bereich)			
	59	—(R)	(Bereich)			
"(O)PCx00"	18	—(R)	(Bereich)	24	—(R)	(Bereich)
	46	—( )		55	—( )	
	59	—(R)	(Bereich)			
"(O)PC0x0"	18	—(R)	(Bereich)	24	—( )	
	30	—(S)		38	—(R)	
	46	—( )		55	—( )	
	59	—(R)	(Bereich)			
"(O)PC00x"	18	—(R)	(Bereich)	19	—(S)	
	24	—( )		30	—(S)	(Bereich)
	38	—( )		38	—(R)	(Bereich)
	46	—( )		55	—( )	
"(O)24V_supp"	18	—(R)	(Bereich)			
	19	—(S)	(Bereich)			
	24	—( )				

Element Netzwerk / Operation

```
                         30   —(S)  (Bereich)
                         38   —( )
                         46   —( )
                         55   —( )

A1.5             18   —(R)  (Bereich)

A1.6             18   —(R)  (Bereich)

A1.7             18   —(R)  (Bereich)

"Start"          21   —| |—              32   —( )
                 40   —( )               48   —( )
                 56   —| |—              63   —( )
                 66   —( )               69   —( )

"Stopp"          8    —| |—              56   —| |—
                 63   —( )               66   —( )
                 69   —( )

"Close"          8    —| |—              21   —| |—
                 25   —|/|—              66   —( )
                 69   —( )

"LifeSign"       6    —( )               8    —|/|—
                 18   —(S)

"Time_switch37"          18   —(S)  (Bereich)

"Time_switch38"          18   —(S)  (Bereich)

"Time_switch39"          18   —(S)  (Bereich)

"Time_switch40"          18   —(S)  (Bereich)

"Time_switch41"          18   —(S)  (Bereich)

"Time_switch42"          18   —(S)  (Bereich)

V201.7           18   —(S)  (Bereich)

"S0"             9    —| |—              18   —(R)
                 21   —( )               24   —(R)
                 30   —(R)               38   —(R)
                 46   —(R)               55   —(R)
                 59   —(R)
```


Element Netzwerk / Operation

```
"S1"      10   —| |—                 18   —(R)  (Bereich)
          21   —( )                  24   —(R)  (Bereich)
          27   —( )                  30   —(R)  (Bereich)
          34   —( )                  35   —|/|—
          38   —(R)  (Bereich)       42   —( )
          43   —|/|—                 46   —(R)  (Bereich)
          51   —( )                  52   —|/|—
          55   —(R)  (Bereich)       56   —( )
          59   —(R)  (Bereich)

"S2"      11   —| |—                 18   —(R)  (Bereich)
          24   —(R)  (Bereich)       25   —( )
          26   —|/|—                 27   —|/|—
          30   —(R)  (Bereich)       35   —( )
          38   —(R)  (Bereich)       46   —(R)  (Bereich)
          49   —( )                  55   —(R)  (Bereich)
          59   —(R)  (Bereich)

"S3"      12   —| |—                 18   —(R)  (Bereich)
          24   —(R)  (Bereich)       26   —( )
          27   —|/|—                 30   —(R)  (Bereich)
          38   —(R)  (Bereich)       43   —( )
          46   —(R)  (Bereich)       50   —( )
          52   —( )                  55   —(R)  (Bereich)
          59   —(R)  (Bereich)

"S4"      13   —| |—                 18   —(R)  (Bereich)
          24   —(R)  (Bereich)       30   —(R)  (Bereich)
          33   —( )                  35   —|/|—
          38   —(R)  (Bereich)       41   —( )
          43   —|/|—                 46   —(R)  (Bereich)
          52   —|/|—                 55   —(R)  (Bereich)
          59   —(R)  (Bereich)

"S5"      14   —| |—                 18   —(R)  (Bereich)
          24   —(R)  (Bereich)       30   —(R)  (Bereich)
          32   —( )                  35   —|/|—
          38   —(R)  (Bereich)       40   —( )
          43   —|/|—                 46   —(R)  (Bereich)
          48   —( )                  52   —|/|—
          55   —(R)  (Bereich)       56   —( )
          59   —(R)  (Bereich)

"S6"      15   —| |—                 18   —(R)  (Bereich)
          24   —(R)  (Bereich)       25   —( )
          27   —|/|—                 30   —(R)  (Bereich)
```

Element	Netzwerk / Operation			
	38	—(R) (Bereich)	46	—(R) (Bereich)
	55	—(R) (Bereich)	59	—(R) (Bereich)
"S7"	18	—(R) (Bereich)	24	—(R) (Bereich)
	30	—(R) (Bereich)	38	—(R) (Bereich)
	46	—(R) (Bereich)	55	—(R) (Bereich)
	59	—(R) (Bereich)		
"TPower_set_Allo"	84	—( )		
	85	—| |—		
"TPower_set_acti"	26	—| |—		
	33	—| |—		
	85	—( )		
"Pause_LI"	18	—( )	25	—|/|—
	33	—|/|—	34	—| |—
	41	—|/|—	49	—| |—
	50	—|/|—	51	—| |—
	72	—( )	78	—( )
"Pause_TI"	18	—( )	26	—|/|—
	33	—|/|—	41	—|/|—
	49	—|/|—	50	—| |—
	51	—| |—	75	—( )
	81	—( )		
"Pow_or_Com"	7	—|/|—	63	—( )
	66	—( )	69	—( )
	72	—( )	75	—( )
	78	—( )	81	—( )
"System_Error"	2	—( )		
	21	—|/|—		
	25	—|/|—		
	26	—|/|—		
	31	—|/|—		
	32	—| |—		
	33	—|/|—		
	34	—|/|—		
	39	—|/|—		
	40	—| |—		
	41	—|/|—		
	42	—|/|—		
	47	—|/|—		
	48	—| |—		

Element	Netzwerk / Operation			
	49	—|/|—		
	50	—|/|—		
	51	—|/|—		
	56	—|/|—		
"LOLA_Halt"	8	—( )	25	—|/|—
	26	—|/|—	33	—|/|—
	34	—| |—	41	—|/|—
	42	—| |—	49	—|/|—
	50	—|/|—	51	—| |—
V310.0	84	—|/|—	84	—| |—
	84	—|/|—	85	—| |—
V310.1	84	—|/|—	84	—|/|—
	84	—| |—	84	—|/|—
	84	—|/|—	85	—| |—
V310.2	84	—| |—	84	—|/|—
V310.3	84	—|/|—	84	—|/|—
	84	—| |—	84	—|/|—
	84	—|/|—	85	—| |—
V310.4	84	—| |—	84	—|/|—
V310.5	4	—(R)		
V310.6	4	—(R) (Bereich)		
V310.7	4	—(R) (Bereich)		
VB310	3	MOV B	5	—|==B|—
	7	—|==B|—	7	—|==B|—
	7	—|==B|—	7	—|==B|—
	7	—|==B|—	7	—|==B|—
	7	—|==B|—		
VB409	7	—|==B|—		
VB410	7	—|==B|—		
VB411	7	—|==B|—		
VB412	5	—|==B|—		

Ladder Diagram for PLC-program "Security Concept" 28.08.01

Element Netzwerk / Operation

Element	Netzwerk	Operation	Netzwerk	Operation
VB413	7	—|==B|—		
VB414	7	—|==B|—		
VB416	7	—|==B|—		
VB417	7	—|==B|—		
T37	5	TON	6	—| |—
T38	5	TON	6	—| |—
T39	30	TON	31	—| |—
	33	—| |—	34	—| |—
T40	38	TON	39	—| |—
	41	—| |—	42	—| |—
	49	—| |—	50	—| |—
	51	—| |—		
T41	20	TON	21	—| |—
T42	46	TON	47	—| |—
SM0.0	3	—| |—	4	—| |—
	18	—| |—	19	—| |—
	20	—| |—	24	—| |—
	30	—| |—	38	—| |—
	46	—| |—	55	—| |—
	59	—| |—	60	—| |—
	63	—| |—	66	—| |—
	72	—| |—	75	—| |—
	78	—| |—	81	—| |—
SM0.1	1	—| |—	69	—| |—

Anhang F

Programming Diagram of LOLA

Contents

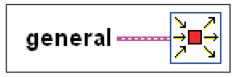


Figure 1: Connector Pane of DICE_DIstribution_CEnter.vi

1 DICE_DIstribution_CEnter.vi

The Distribution Center handles data. This data consists of measurement data and messages. Measurement data are the digitalized quantities from the measurement hardware. Messages consist of:

condition : Is information about the condition of each module (subVI's running at the same time).

report : All information that is necessary to control the Test Stand.

command : Information sent to modules to control their behaviour.

confirmation : Is an answer of each module to a given *command* .

log : Is information consisting of concentrated and reformatted *conditions* . This information is saved into a log-file and displayed at the userinterface.

The Distribution Centers duty is to collect the data from Temporary Storages, sort it and distribute it to modules. In this way each module gets the data and only the data it needs.

1.1 Connector Pane

See figure 1 on page 10.

1.2 Front Panel

1.3 Controls and Indicators

general

- **standard_path** path where the program is located
- **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
- **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
  - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
- **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
  - **value**
- **semaphores** is an array of names of all existing semaphores
  - **name**

1.4 Block Diagram

See figure 2 on page 11.

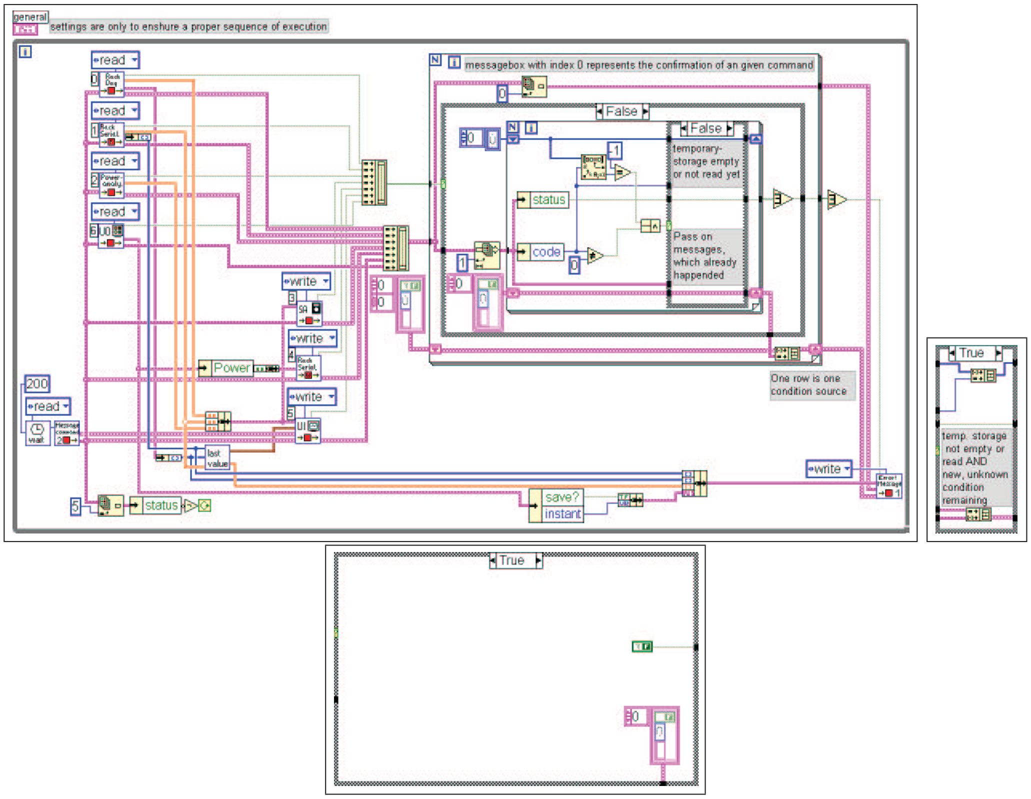
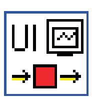
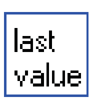


Figure 2: Block Diagram of DICE_DIstribution_CEnter.vi

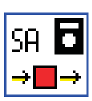
1.5 List of SubVIs



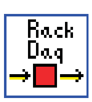
TSUI_TemporaryStorage_UserInterface.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSUI_TemporaryStorage_UserInterface.vi



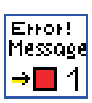
DILV_DIstribution_LastValues.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Distribution.llb\DILV_DIstribution_LastValues.vi



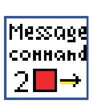
TSSA_TemporaryStorage_SAve.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSSA_TemporaryStorage_SAve.vi



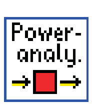
TSRD_TemporaryStorage_Rack_Daq.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSRD_TemporaryStorage_Rack_Daq.vi



TSM1_TemporaryStorage_Messages1.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSM1_TemporaryStorage_Messages1.vi



TSM2_TemporaryStorage_Messages2.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSM2_TemporaryStorage_Messages2.vi



TSPA_TemporaryStorage_PowerAnalyzer.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSPA_TemporaryStorage_PowerAnalyzer.vi

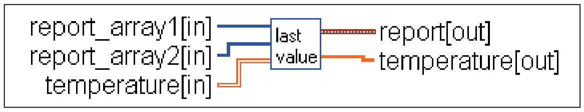
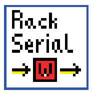
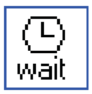


Figure 3: Connector Pane of DILV_DIstribution_LastValues.vi

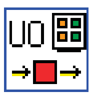
TSRSr_TemporaryStorage_Rack_Serialread.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSRSr_TemporaryStorage_Rack_Serialread.vi



TSRSw_TemporaryStorage_Rack_Serialwrite.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSRSw_TemporaryStorage_Rack_Serialwrite.vi



GLWA_GLobalWAit.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLWA_GLobalWAit.vi



TSUO_TemporaryStorage_UserOperation.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSUO_TemporaryStorage_UserOperation.vi

1.6 History

DICE_DIstribution_CEnter.vi History Current Revision: 115

2 DILV_DIstribution_LastValues.vi

For displaying the report data at the Userinterface only the actual values are of interest, because the evaluation of all these values is done in SICE_SInage_CEnter.vi. Therefore except the last values the rest of the incoming array will be truncated.

2.1 Connector Pane

See figure 3 on page 12.

2.2 Front Panel

See figure 4 on page 13.

2.3 Controls and Indicators

[U32] **report_array1[in]**

- [U32] **Return count**

[U32] **report_array2[in]**

- [U32]

[DBL] **temperature[in]**

- [DBL] **Measured Value**

Figure 4: Front Panel of DILV_DIstribution_LastValues.vi

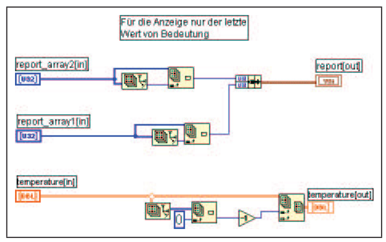


Figure 5: Block Diagram of DILV_DIstribution_LastValues.vi

report[out]

- **scan_backlog**
- **return_count**

temperature[out]

- **Measured Value**

2.4 Block Diagram

See figure 5 on page 13.

2.5 List of SubVIs

2.6 History

DILV_DIstribution_LastValues.vi History Current Revision: 11

3 GCEH_Global_Changed_Error_Handler.vi

Changes: extension of known error codes

This error handler is used primarily to inform the user if an input error exists, to describe the error, and to identify where it occurred. The information for this is derived from the inputs error in, error code, and error source, and from an internal error description table. The table describes all errors that can be created by LabView or its associated I/O operations. The handler has provisions to take alternative actions, such as to cancel or set an error status, and to test for and describe user-defined errors. See instruction on the front panel.

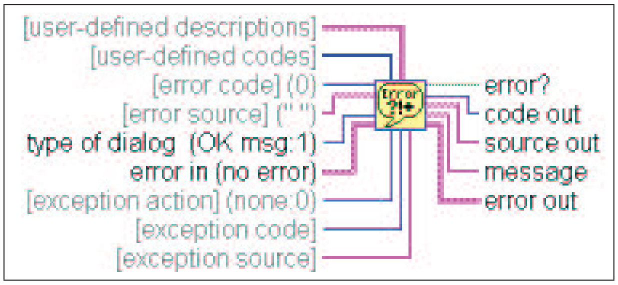


Figure 6: Connector Pane of GCEH_Global_Changed_Error_Handler.vi

3.1 Connector Pane

See figure 6 on page 14.

3.2 Front Panel

See figure 7 on page 15.

3.3 Controls and Indicators

type of dialog (OK msg:1) type of dialog determines what type dialog box will be displayed, if any. Regardless of its value, the VI outputs error information and message describing the error. 0: displays no dialog box. This is useful if you want to have programmatic control over how the error is handled. 1: (the default value) displays a dialog box with a single OK button. After the user responds, the VI returns control to the main VI. 2: displays a dialog box with buttons allowing the user to either continue or stop. If the user cancels, the VI calls the Stop function to halt execution.

[**exception code**] exception code is the error code that you want to treat as an exception. By default, it is 0.

[**exception source**] exception source is the error message that you want to use to test for an exception. By default, it is an empty string.

[**error source**] **( )** error source is an optional string you can use to describe the source of error code. The VI uses the string in message if there is an error.

[**error code**] **(0)** error code is a numeric error code. If error in indicates an error, the VI ignores error code. If not, the VI tests it. A non-zero value signifies an error.

[**exception action**] **(none:0)** exception action is a way for you to create exceptions to error handling. You can treat what is normally an error as no error, or treat a no error condition as an error using this parameter. 0: (the default value) performs no error exception handling. 1: cancels error under the following conditions: If the VI detects an error, as described in the status and error code parameters, and if that error code value matches exception code and the error source value matches exception source, the VI sets status out to FALSE, code out to 0, and source out to an empty string. An empty exception source string matches any error source string. 2: sets error under the following conditions: If the VI detects no error, as described in the status and error code parameters, but the code value of error in matches exception code and the error source value matches exception source, the VI sets status out to TRUE, code out to the code value from error in, and source out to the source value from error in.

error in (no error) error in describes an error that you want to check. If not wired, this VI checks error code for errors.

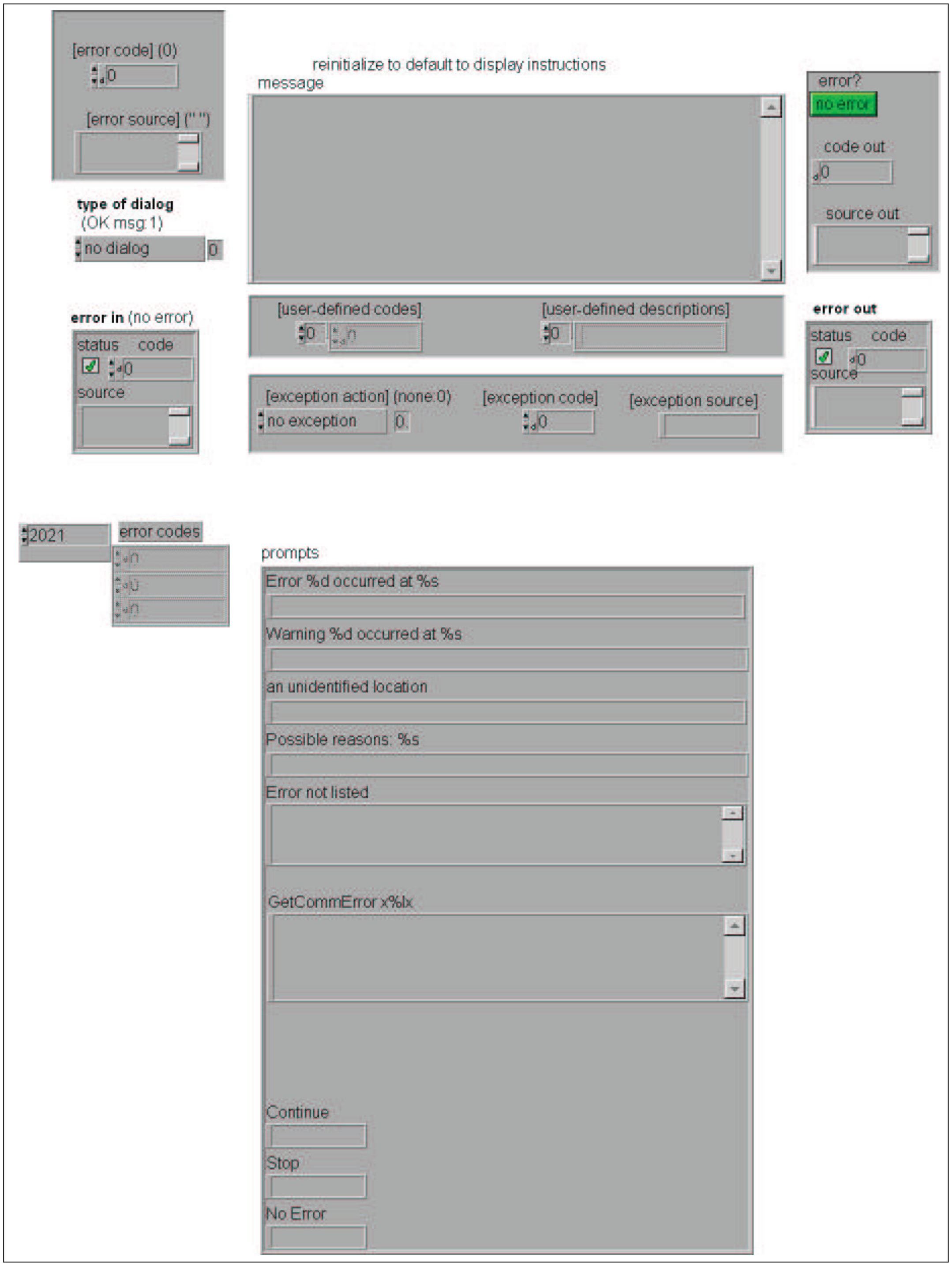


Figure 7: Front Panel of GCEH_Global_Changed_Error_Handler.vi

status The status boolean is either TRUE (cross) for an error, or FALSE (checkmark) for no error or a warning.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

code The code input identifies the error or warning reported.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

source The source string describes the origin of the error or warning reported.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

[**user-defined codes**] user-defined codes is an array of the numeric error codes you define in your VIs. The VI searches this array after searching the internal list of error codes. Codes in the range of 5000 to 9999 are reserved for users.

[**user-defined descriptions**] user-defined descriptions is an array of descriptions of user-defined codes. If an incoming error matches one in user-defined codes, the VI uses the corresponding description from user-defined descriptions in message.

error descriptions

error codes

prompts

- **Error %d occurred at %s**
- **Warning %d occurred at %s**
- **Possible reasons: %s**
- **Error not listed**
- **GetCommError x%lx**
- **Continue**
- **Stop**
- **an unidentified location**
- **No Error**

code out code out is the error code indicated by the error in or error code.

source out source out indicates the source of the error.

error?

error out error out contains the same information as status out, code out, and source out. It has the same structure as error in.

status The status boolean is either TRUE (cross) for an error, or FALSE (checkmark) for no error or a warning.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

code The code input identifies the error or warning reported.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

source The source string describes the origin of the error or warning reported.

The pop-up option Explain Error (or Explain Warning) gives more information about the error displayed.

message message describes the error code that occurred, the source of the error, and a description of the error.

3.4 Block Diagram

See figure 8 on page 18.

3.5 List of SubVIs

3.6 History

GCEH_Global_Changed_Error_Handler.vi History Current Revision: 42

4 GLAD_GLobal_AddDiffrent_errors.vi

This Vi rearranges *message_array[in]* in the way that multiple messages with the same *code* appear only once in *message_array[out]* .

4.1 Connector Pane

See figure 9 on page 18.

4.2 Front Panel

See figure 10 on page 19.

4.3 Controls and Indicators

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be condition,commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be condition,commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

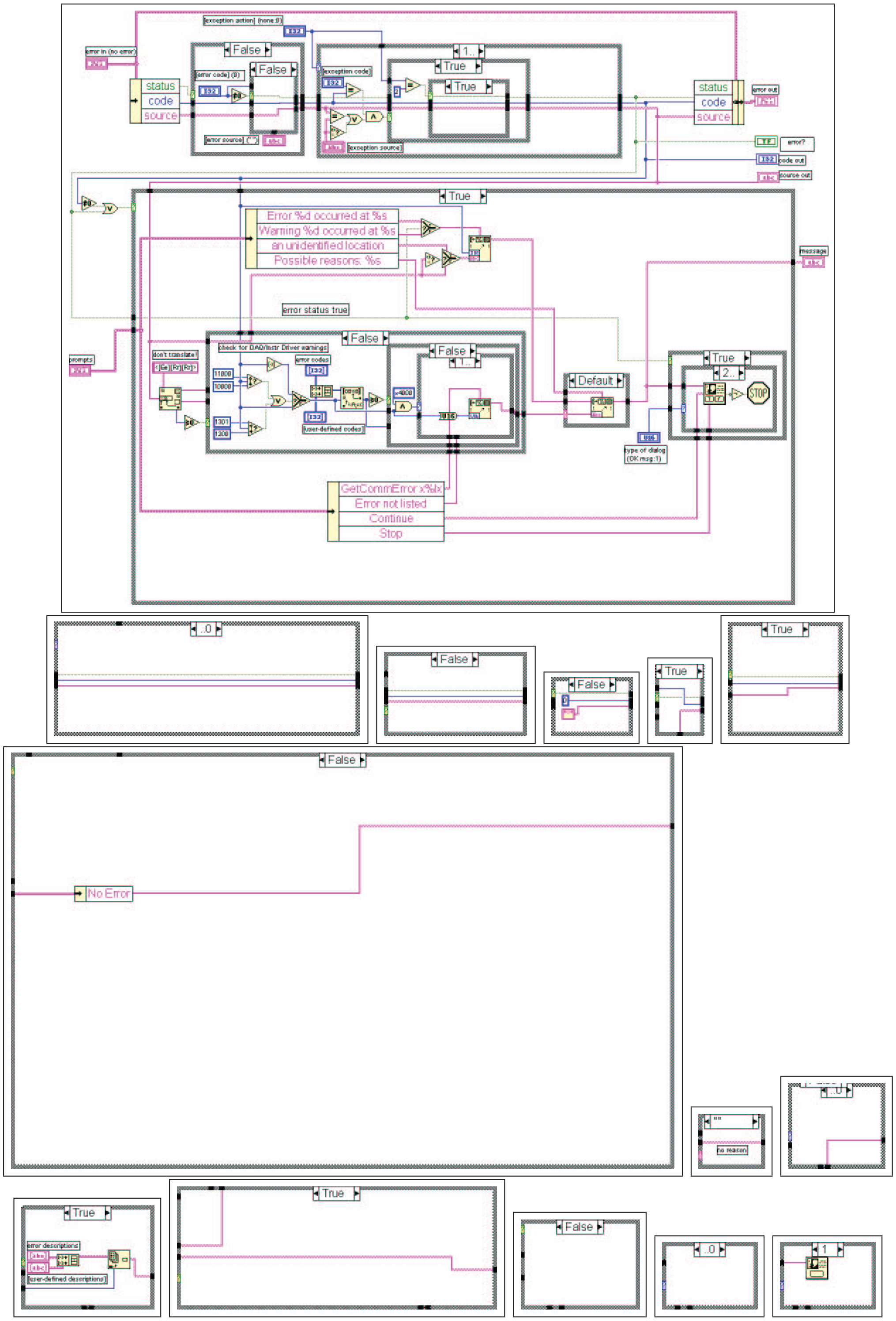


Figure 8: Block Diagram of GCEH_Global_Changed_Error_Handler.vi

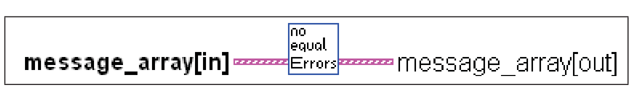


Figure 9: Connector Pane of GLAD_GLobal_AddDiffrent_errors.vi

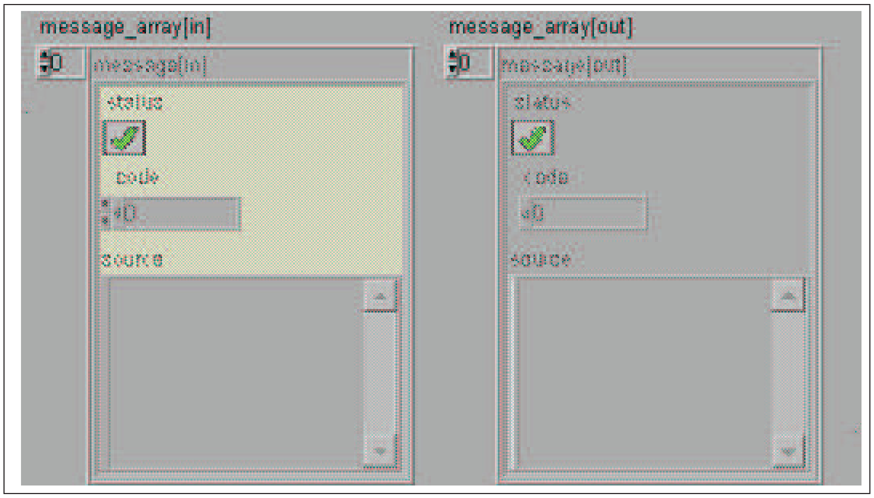


Figure 10: Front Panel of GLAD_GLobal_AddDiffrent_errors.vi

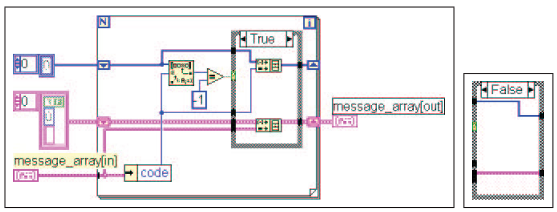


Figure 11: Block Diagram of GLAD_GLobal_AddDiffrent_errors.vi

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

4.4 Block Diagram

See figure 11 on page 19.

4.5 List of SubVIs

4.6 History

GLAD_GLobal_AddDiffrent_errors.vi History Current Revision: 18

5 GLAE_GLobal_AddtwoErrors.vi

Passes on the message cluster with *status* = true. If both clusters contain *status* = true, *error[in]_high* will be passed on.

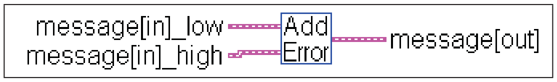


Figure 12: Connector Pane of GLAE_GLobal_AddtwoErrors.vi

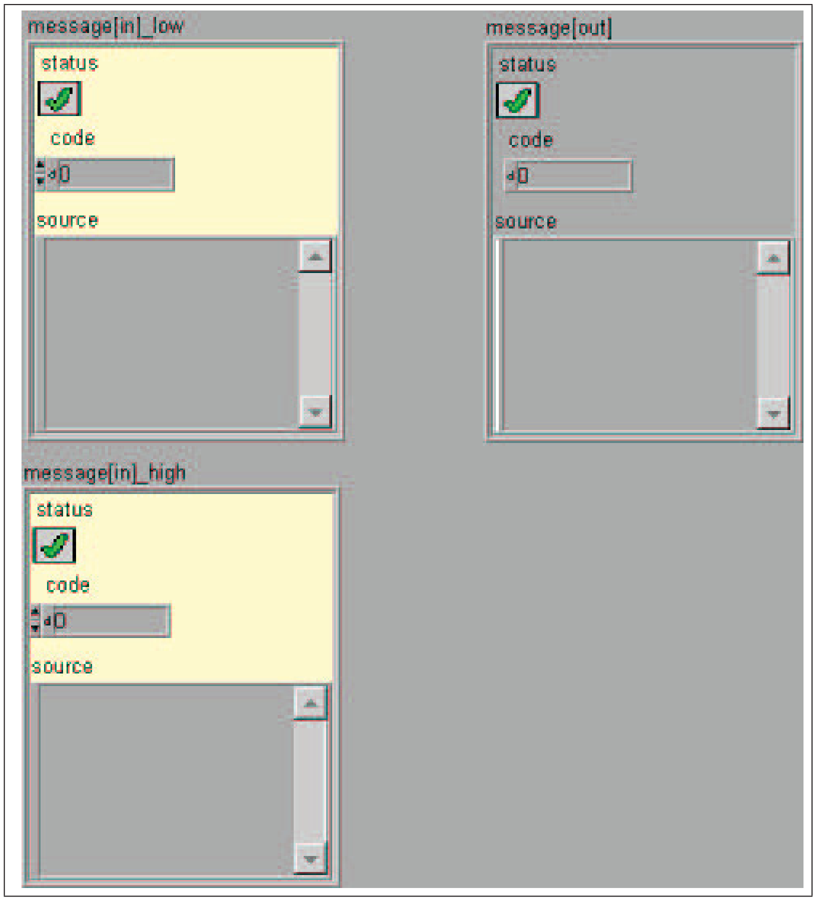


Figure 13: Front Panel of GLAE_GLobal_AddtwoErrors.vi

5.1 Connector Pane

See figure 12 on page 20.

5.2 Front Panel

See figure 13 on page 20.

5.3 Controls and Indicators

message[in]_low is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

This cluster got a lower priority compared to the cluster *message[in]_high* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

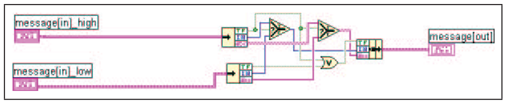


Figure 14: Block Diagram of GLAE_GLobal_AddtwoErrors.vi

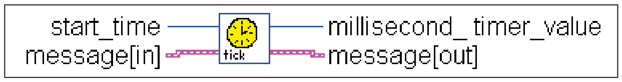


Figure 15: Connector Pane of GLCL_GLobal_CLock.vi

message[in]_high is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

This cluster got a higher priority compared to the cluster *message[in]_low* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

5.4 Block Diagram

See figure 14 on page 21.

5.5 List of SubVIs

5.6 History

GLAE_GLobal_AddtwoErrors.vi History Current Revision: 9

6 GLCL_GLobal_CLock.vi

This Vi computes the actual time with reference to *start_time* .

millisecond_timer_value := current time - *start_time*

6.1 Connector Pane

See figure 15 on page 21.

6.2 Front Panel

See figure 16 on page 22.

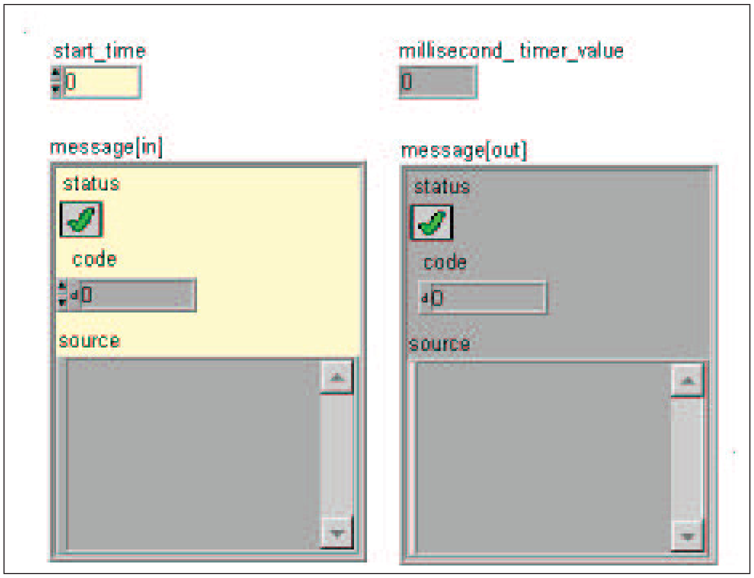


Figure 16: Front Panel of GLCL_GLobal_CLock.vi

6.3 Controls and Indicators

start_time [ms] is the time reference. In *LOLA*it is set at INCH_INitialisation_Config-Hardware.vi at the start of the program.

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

millisecond_ timer_value is the time period in [ms] between the actual time and *start_time* .

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

6.4 Block Diagram

See figure 17 on page 23.

6.5 List of SubVIs

6.6 History

GLCL_GLobal_CLock.vi History Current Revision: 23

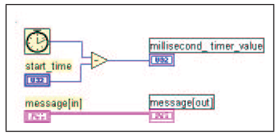


Figure 17: Block Diagram of GLCL_GLobal_CLock.vi

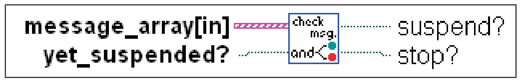


Figure 18: Connector Pane of GLCM_GLobal_CheckMessages.vi

7 GLCM_GLobal_CheckMessages.vi

Looks up *message_array[in]* if any code of the *message_array[in]* is equal to 9999 (global stop condition) or 5000 (reactivation condition, not implemented yet). In case one *code* = 9000, *suspend?* := TRUE and *stop?* := TRUE. In case one *code* is 5000, *suspend?* := FALSE.

7.1 Connector Pane

See figure 18 on page 23.

7.2 Front Panel

See figure 19 on page 24.

7.3 Controls and Indicators

yet_suspended? is true if higher vi is suspended, false if it is active.

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be condition,commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

suspend?

stop?

7.4 Block Diagram

See figure 20 on page 24.

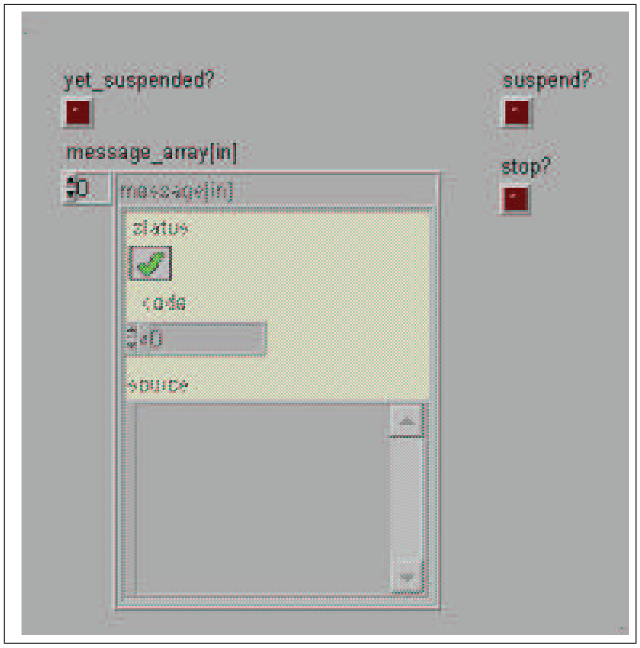


Figure 19: Front Panel of GLCM_GLobal_CheckMessages.vi

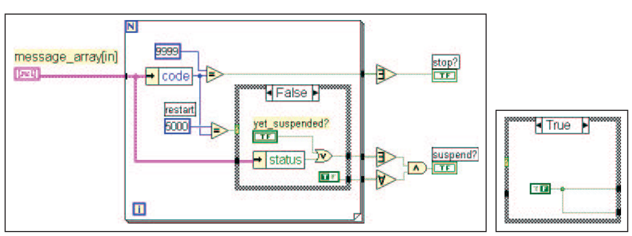


Figure 20: Block Diagram of GLCM_GLobal_CheckMessages.vi

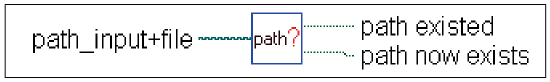


Figure 21: Connector Pane of GLCP_GLobal_Check_Path.vi

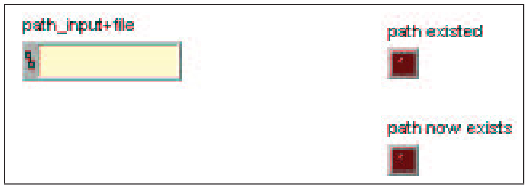


Figure 22: Front Panel of GLCP_GLobal_Check_Path.vi

7.5 List of SubVIs

7.6 History

GLCM_GLobal_CheckMessages.vi History Current Revision: 29

8 GLCP_GLobal_Check_Path.vi

checks if the given path is valid and creates it if it does not exist

8.1 Connector Pane

See figure 21 on page 25.

8.2 Front Panel

See figure 22 on page 25.

8.3 Controls and Indicators

path_input+file path that should be checked

path existed true if *input_path* existed

path now exists true if *input_path* exists after execution

8.4 Block Diagram

See figure 23 on page 26.

8.5 List of SubVIs

8.6 History

GLCP_GLobal_Check_Path.vi History Current Revision: 7

9 GLSA_Stop_to_All.vi

This VI stops VIs running at the same time.

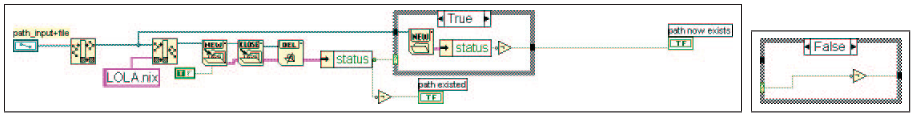


Figure 23: Block Diagram of GLCP_GLobal_Check_Path.vi

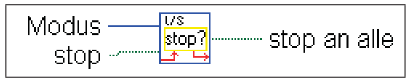


Figure 24: Connector Pane of GLSA_Stop_to_All.vi

9.1 Connector Pane

See figure 24 on page 26.

9.2 Front Panel

See figure 25 on page 27.

9.3 Controls and Indicators

stop

mode

stop to all

9.4 Block Diagram

See figure 26 on page 27.

9.5 List of SubVIs

9.6 History

GLSA_Stop_to_All.vi History Current Revision: 16

10 GLSN_GLobal_SearchforNumber.vi

searches the index of *number_to_search_for* in *array_in*

10.1 Connector Pane

See figure 27 on page 27.

10.2 Front Panel

See figure 28 on page 27.

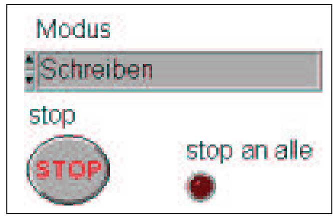


Figure 25: Front Panel of GLSA_Stop_to_All.vi

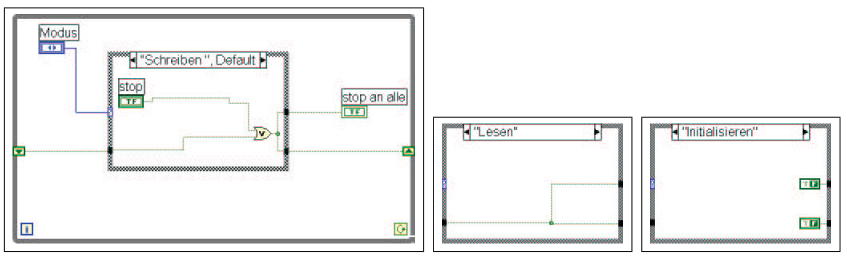


Figure 26: Block Diagram of GLSA_Stop_to_All.vi

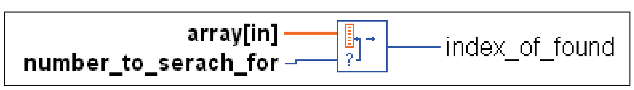


Figure 27: Connector Pane of GLSN_GLobal_SearchforNumber.vi

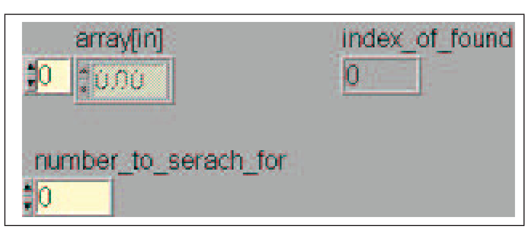


Figure 28: Front Panel of GLSN_GLobal_SearchforNumber.vi

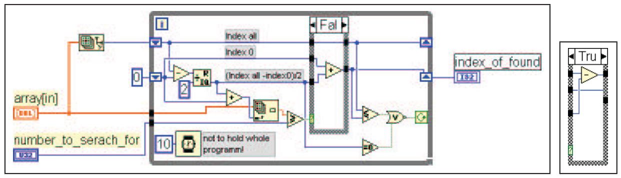


Figure 29: Block Diagram of GLSN_GLobal_SearchforNumber.vi

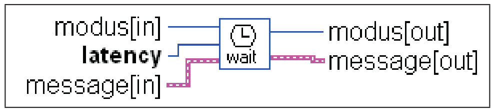


Figure 30: Connector Pane of GLWA_GLobalWAit.vi

10.3 Controls and Indicators

array[in] array in which *number_to_search_for* should be found

Measured Value

number_to_serach_for number to look for

index_of_found index of found element in *array[in]*

10.4 Block Diagram

See figure 29 on page 28.

10.5 List of SubVIs

10.6 History

GLSN_GLobal_SearchforNumber.vi History Current Revision: 8

11 GLWA_GLobalWAit.vi

This VI waits *latency* milliseconds and is usually used to release processor resources for certain time period.

11.1 Connector Pane

See figure 30 on page 28.

11.2 Front Panel

See figure 31 on page 29.

11.3 Controls and Indicators

latency time period to halt the calling VI

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

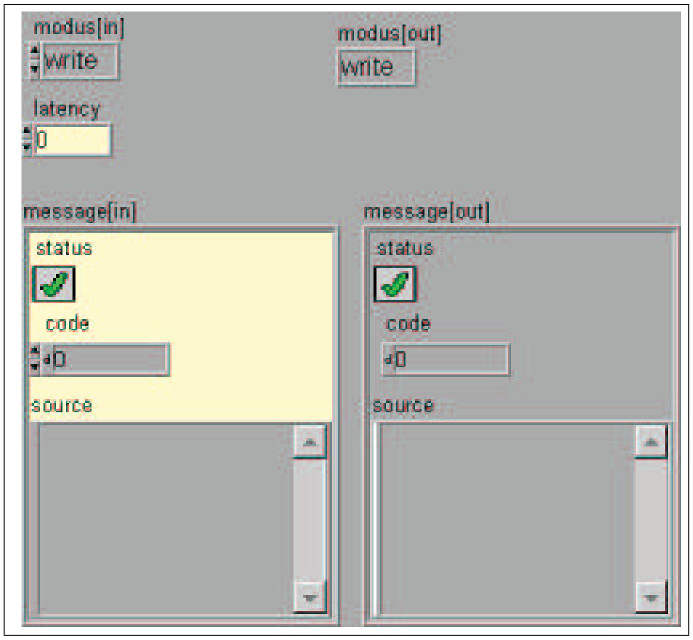


Figure 31: Front Panel of GLWA_GLobalWAit.vi

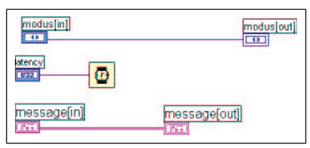


Figure 32: Block Diagram of GLWA_GLobalWAit.vi

- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

modus[in]

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

modus[out]

11.4 Block Diagram

See figure 32 on page 29.

11.5 List of SubVIs

11.6 History

GLWA_GLobalWAit.vi History Current Revision: 13

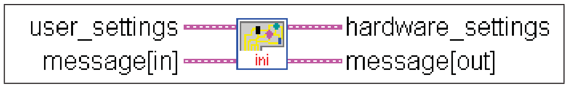


Figure 33: Connector Pane of HCCE_HardwareConfig_CEnter.vi

12 HCCE_HardwareConfig_CEnter.vi

Start of all measurement.

12.1 Connector Pane

See figure 33 on page 30.

12.2 Front Panel

12.3 Controls and Indicators

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.
- **VISA session (for class)** is a unique logical identifier used to communicate with a resource.
- **user settings[in]**
  - **general**
    - **Numeric_for_nothing** this numeric is for nothing
    - **standard_path** is the path where the program is located
    - **list_column_names** is an array containing the titles of the measurement categories
      The names' order must match to the commands' order.
      - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
    - **list_ranges** is an array of all rated-, warning- and alarm-level ordered according to *list_column_names*
      - **value**
    - **semaphores** is an array of names of all existing semaphores
      - **name**
  - **Dewerack/serial**
    - **baud_rate**
    - **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
    - **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
    - **timeout_value_read[in]** (not in use at the moment)
    - **timeout_value_write[out]** (not in use at the moment)

- **byte_ count_ read[in]** is the number of bytes to be read
- **byte_count_write[out]** is the number of bytes to be written
- **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
- **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
- **terms_init** is an array of commands for initialization of the Dewerack
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)

Dewerack/DAQ

- **device** is the device number of the DAQ board
- **scan_rate** is the number of scans performed per second
- **buffer_size** buffer size is the number of scans each buffer should hold
- **scans_to_read_at_a_time** number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
  Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.
- **channels** specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Poweranalyzer

- **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
- **instrument_descriptor** is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver
- **terms_init** is an array of commands (called terms here) for the initialization of the Poweranalyzer
  - **term** for the initialization request at the Poweranalyzer (called term here)
- **terms_service** is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases
  - **term** command for the read request at the Poweranalyzer (called term here)

saving

- **save_interval** determines the amount of measurement data saved
- **main_register_faktor** determines the amount of measurement data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*

datafile_paths name and location of files where the measured data is saved

path consists of path (location) and filename

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

hardware_settings

general

standard_path path where the program is located

start_time [ms] is the time reference. In *LOLAstart_time* is set at INKH_-INitialisation_KonfigHardware.vi at the start of the program.

list_column_names is an array containing the titles of the measurement categories The names' order must match to the commands' order.

name_of_channel channel name is displayed in the user interface, it consists of identifier and unit

list_ranges is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*

value

semaphores is an array of names of all existing semaphores

name

Dewerrack/serial

latency_read[in] is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop

latency_write[out] is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop

timeout_value_read[in] (not in use at the moment)

timeout_value_write[out] (not in use at the moment)

byte_ count_ read[in] is the number of bytes to be read

byte_count_write[out] is the number of bytes to be written

latency_sign_of_life is the time period between two alternating signals sent to the PLC S7 200

terms_read_in is an array of addresses respectively commands for the read request at a certain port

term address respectively command for the read request at a certain port (called term here)

terms_write_out is an array of addresses respectively commands for the write request at a certain port

term address respectively command for the read request at a certain port (called term here)

VISA_session_(for_class) is a unique logical identifier used to communicate with a resource.

Dewerack/DAQ

- **scans_to_read_at_a_time** number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
  Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.
- **taskID** is a unique logical identifier used to communicate with a resource.
- **scan_rate** is the number of scans performed per second

Poweranalyzer

- **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
- **VISA session** is a unique logical identifier used to communicate with a resource.
- **terms_service** is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases
  - **term** command for the read request at the Poweranalyzer (called term here)

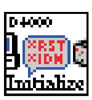
Saving

- **save_interval** determines the amount of Measurement-Data saved
- **main_register_faktor** determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*
- **datafile_paths** name and location of files where the measured data is saved
  - **path** consists of path (location) and filename

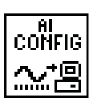
12.4 Block Diagram

See figure 34 on page 34.

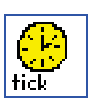
12.5 List of SubVIs



LND4000_Initialize.vi
C:\Programme\National Instruments\Labview\instr.lib\-D4000\LND4000.llb\LND4000_Initialize.vi



AIConfig.vi
C:\Programme\National Instruments\Labview\vi.lib\DAQ\AI.LLB\AIConfig.vi



GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCL_GLobal_CLock.vi

GCEH_Global_Changed_Error_Handler.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GCEH_Global_Changed_Error_Handler.vi

12.6 History

HCCE_HardwareConfig_CEnter.vi History Current Revision: 77

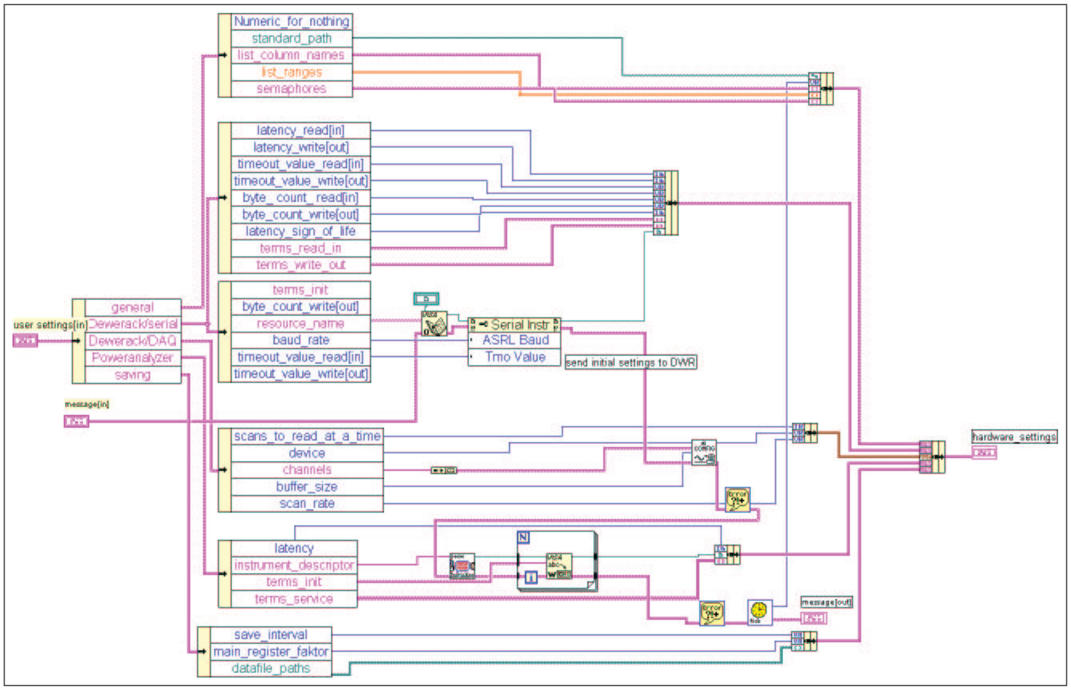


Figure 34: Block Diagram of HCCE_HardwareConfig_CEnter.vi

Figure 35: Connector Pane of IN1N_INitialization_1dNatospradsh.vi

13 IN1N_INitialization_1dNatospradsh.vi

converts one dimensional array of numbers (u32) to spreadsheet string

13.1 Connector Pane

See figure 35 on page 34.

13.2 Front Panel

See figure 36 on page 35.

13.3 Controls and Indicators

[DBL] **array** array of numbers (DBL)

[DBL] **Numeric**

[abc] **spreadsheet string**

13.4 Block Diagram

See figure 37 on page 35.

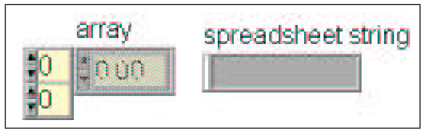


Figure 36: Front Panel of IN1N_INitialization_1dNatospradsh.vi

Figure 37: Block Diagram of IN1N_INitialization_1dNatospradsh.vi

13.5 List of SubVIs

13.6 History

IN1N_INitialization_1dNatospradsh.vi History Current Revision: 13

14 IN1S_INitialization_1dstring_array_to_Spradsheet.vi

converts one dimensional array of strings to spreadsheet string

14.1 Connector Pane

See figure 38 on page 36.

14.2 Front Panel

See figure 39 on page 36.

14.3 Controls and Indicators

[abc] **array** array of strings

[abc] **String**

[abc] **spreadsheet string**

14.4 Block Diagram

See figure 40 on page 37.

14.5 List of SubVIs

14.6 History

IN1S_INitialization_1dstring_array_to_Spradsheet.vi History Current Revision: 12

15 INCE_INitialization_CEnter.vi

This VI enables the user to edit the settings in a comfortable way. The graphic shows a map of the system. Clicking on the appropriate field will popup a menu where the settings can be changed.

Figure 38: Connector Pane of IN1S_INitialization_1dstring_array_to_Spradsheet.vi

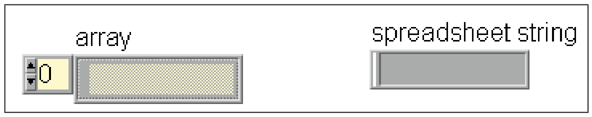


Figure 39: Front Panel of IN1S_INitialization_1dstring_array_to_Spradsheet.vi

15.1 Connector Pane

See figure 41 on page 37.

15.2 Front Panel

See figure 42 on page 37.

15.3 Controls and Indicators

- **user settings[in]**
  - **general**
    - **Numeric_for_nothing** this numeric is for nothing
    - **standard_path** is the path where the program is located
    - **list_column_names** is an array containing the titles of the measurement categories
      The names' order must match to the commands' order.
      - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
    - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
      - **value**
    - **semaphores** is an array of names of all existing semaphores
      - **name**
  - **Dewerack/serial**
    - **baud_rate**
    - **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
    - **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
    - **timeout_value_read[in]** (not in use at the moment)
    - **timeout_value_write[out]** (not in use at the moment)
    - **byte_ count_ read[in]** is the number of bytes to be read
    - **byte_count_write[out]** is the number of bytes to be written
    - **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
    - **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
    - **terms_init** is an array of commands for initialization of the Dewerack

Figure 40: Block Diagram of IN1S_INitialization_1dstring_array_to_Spradsheet.vi

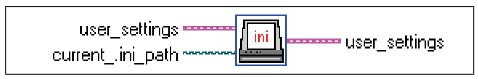


Figure 41: Connector Pane of INCE_INitialization_CEnter.vi

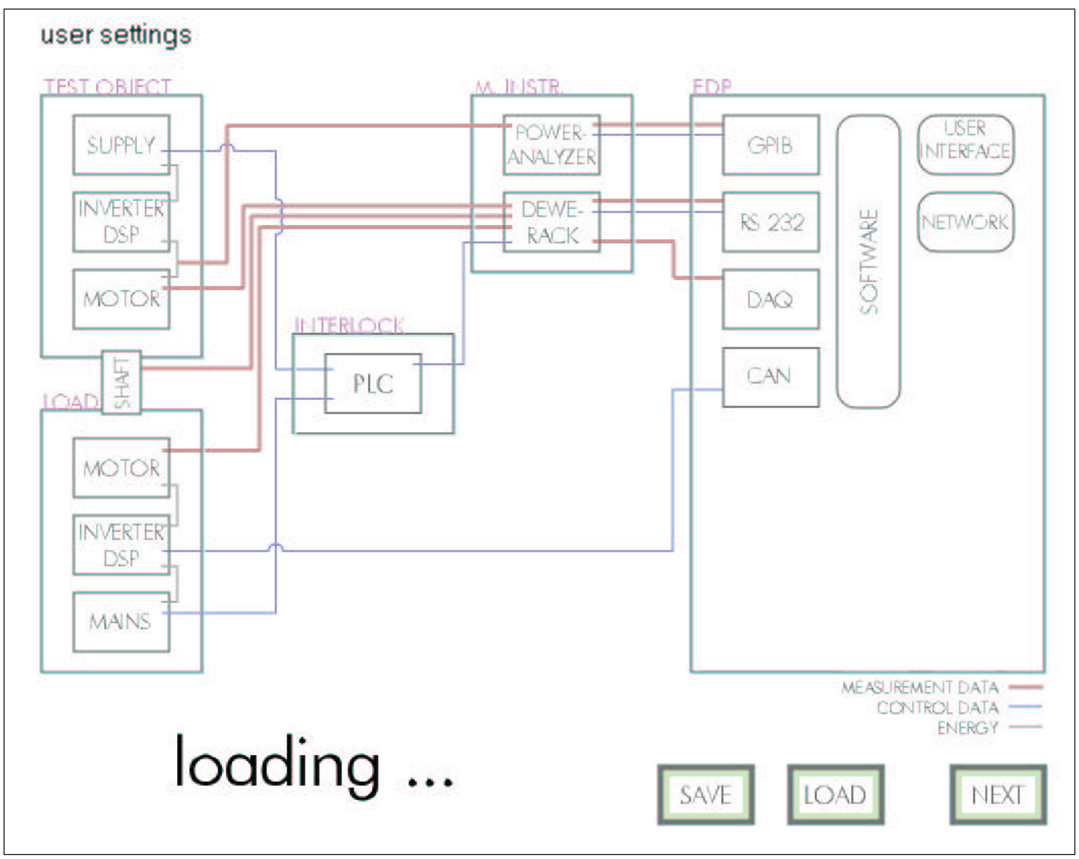


Figure 42: Front Panel of INCE_INitialization_CEnter.vi

term address respectively command for the read request at a certain port (called term here)

terms_read_in is an array of addresses respectively commands for the read request at a certain port

term address respectively command for the read request at a certain port (called term here)

terms_write_out is an array of addresses respectively commands for the write request at a certain port

term address respectively command for the read request at a certain port (called term here)

Dewerack/DAQ

device is the device number of the DAQ board

scan_rate is the number of scans performed per second

buffer_size buffer size is the number of scans each buffer should hold

scans_to_read_at_a_time number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.

channels specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Poweranalyzer

latency is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop

instrument_descriptor is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver

terms_init is an array of commands (called terms here) for the initialization of the Poweranalyzer

term for the initialization request at the Poweranalyzer (called term here)

terms_service is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases

term command for the read request at the Poweranalyzer (called term here)

saving

save_interval determines the amount of Measurement-Data saved

main_register_faktor determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*

datafile_paths name and location of files where the measured data is saved

path consists of path (location) and filename

current_.ini_path

Picture click on the appropriate field on the system map to edit the according settings

user settings[out]

- **general**
  - **Numeric_for_nothing** this numeric is for nothing
  - **standard_path** is the path where the program is located
  - **list_column_names** is an array containing the titles of the measurement categories
    The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**
  - **semaphores** is an array of names of all existing semaphores
    - **name**
- **Dewerack/serial**
  - **baud_rate**
  - **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
  - **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
  - **timeout_value_read[in]** (not in use at the moment)
  - **timeout_value_write[out]** (not in use at the moment)
  - **byte_ count_ read[in]** is the number of bytes to be read
  - **byte_count_write[out]** is the number of bytes to be written
  - **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
  - **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
  - **terms_init** is an array of commands for initialization of the Dewerack
    - **term** address respectively command for the read request at a certain port (called term here)
  - **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
    - **term** address respectively command for the read request at a certain port (called term here)
  - **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
    - **term** address respectively command for the read request at a certain port (called term here)
- **Dewerack/DAQ**
  - **device** is the device number of the DAQ board
  - **scan_rate** is the number of scans performed per second
  - **buffer_size** buffer size is the number of scans each buffer should hold
  - **scans_to_read_at_a_time** number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
    Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.

channels specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Poweranalyzer

- **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
- **instrument_descriptor** is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver
- **terms_init** is an array of commands (called terms here) for the initialization of the Poweranalyzer
  - **term** for the initialization request at the Poweranalyzer (called term here)
- **terms_service** is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases
  - **term** command for the read request at the Poweranalyzer (called term here)

saving

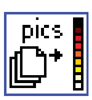
- **save_interval** determines the amount of Measurement-Data saved
- **main_register_faktor** determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*
- **datafile_paths** name and location of files where the measured data is saved
  - **path** consists of path (location) and filename

display current settings are displayed here

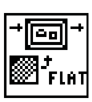
15.4 Block Diagram

See figure 43 on page 41, figure 44 on page 42, figure 45 on page 43 and figure 46 on page 44.

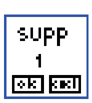
15.5 List of SubVIs



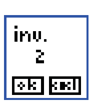
INLP_INitialisation_LoadPictures.vi
D:\Version 2.0\INitialization.llb\INLP_INitialisation_LoadPictures.vi



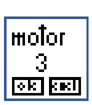
Draw Flattened Pixmap.vi
C:\programme\national instruments\Labview\vi.lib\picture\picture.llb\Draw Flattened Pixmap.vi



INM01_INitialisation_Menue01(supply).vi
D:\Version 2.0\INitialization.llb\INM01_INitialisation_Menue01(supply).vi



INM02_INitialisation_Menue02(inverter).vi
D:\Version 2.0\INitialization.llb\INM02_INitialisation_Menue02(inverter).vi



INM03_INitialisation_Menue03(motor).vi
D:\Version 2.0\INitialization.llb\INM03_INitialisation_Menue03(motor).vi

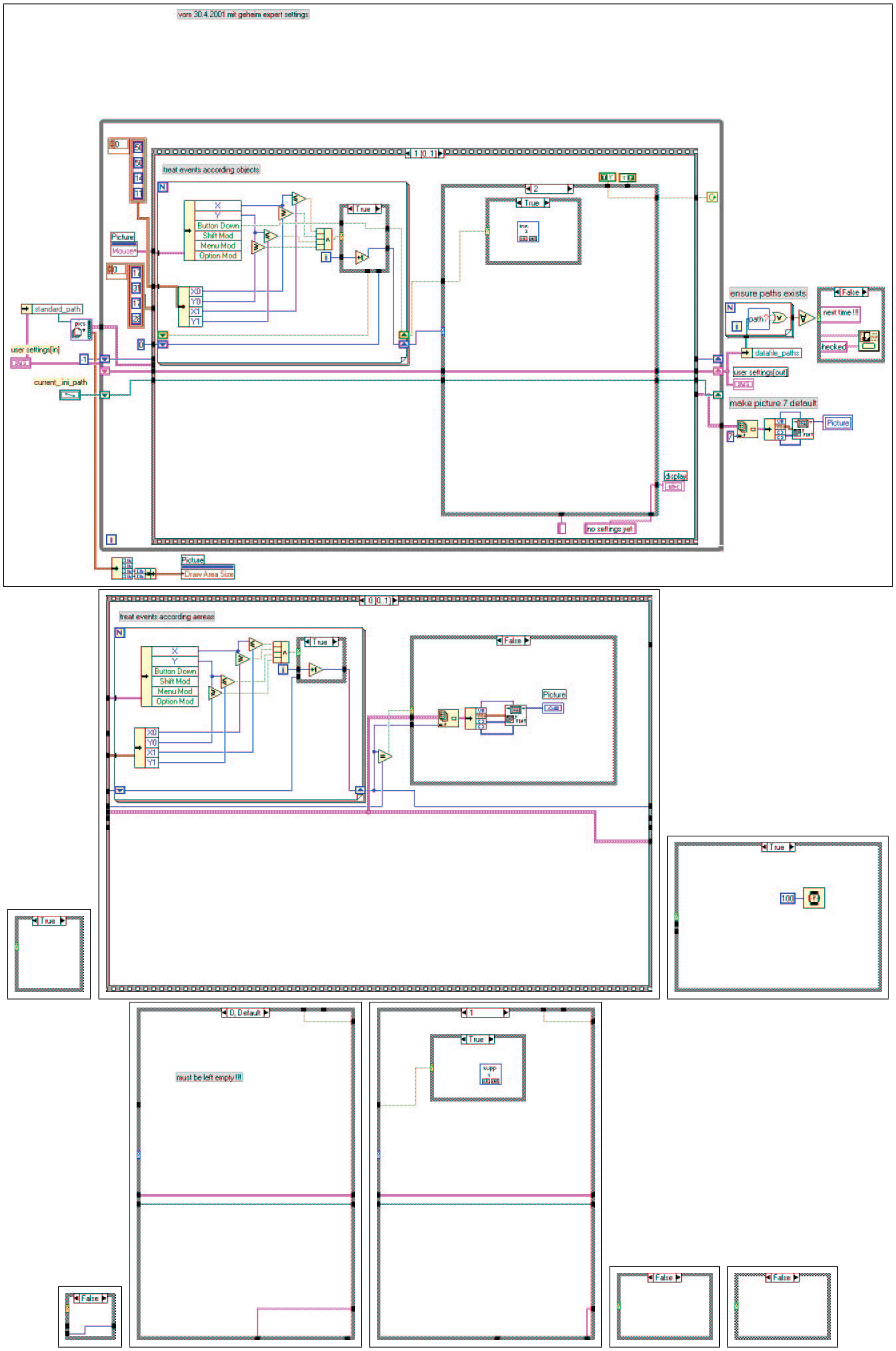


Figure 43: Block Diagram (a) of INCE_INitialization_CEnter.vi

Figure 44: Block Diagram (b) of INCE_INitialization_CEnter.vi

Figure 45: Block Diagram (c) of INCE_INitialization_CEnter.vi

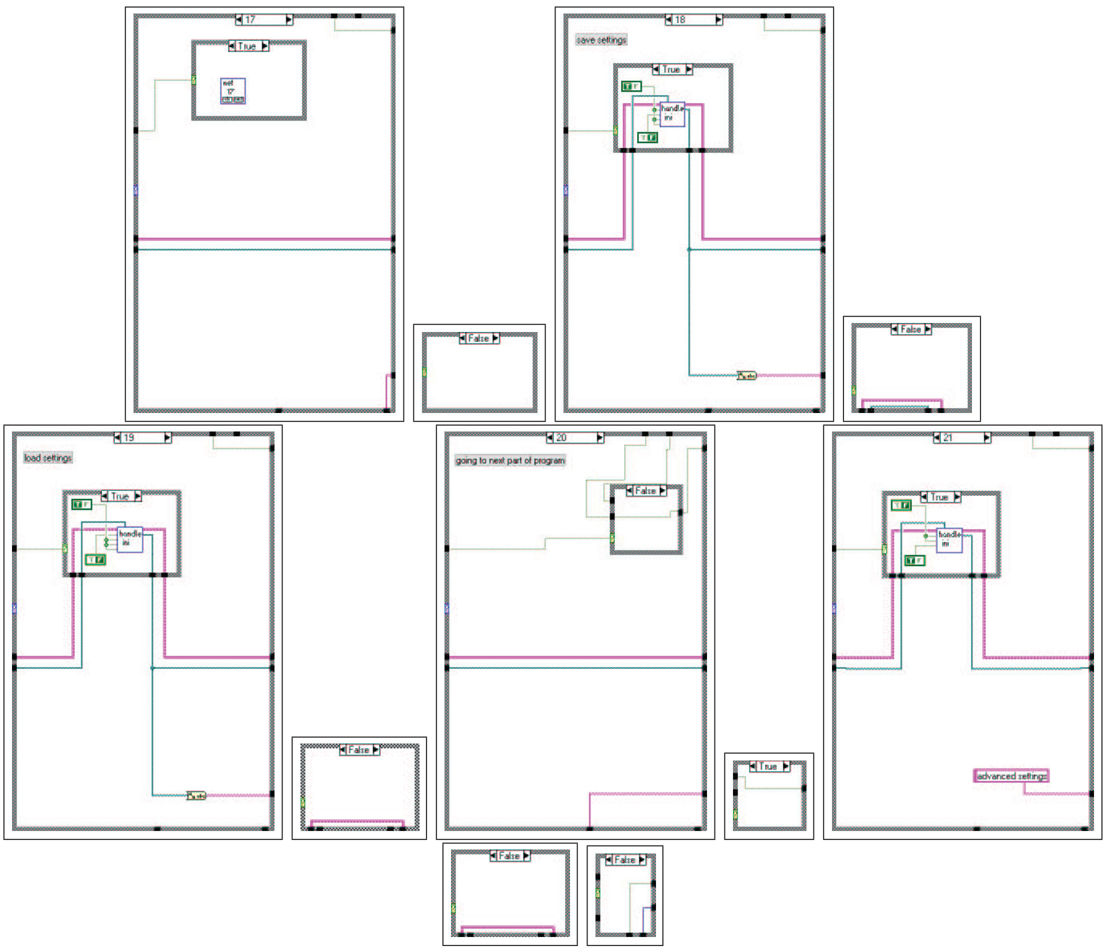
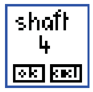
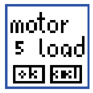


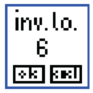
Figure 46: Block Diagram (d) of INCE_INitialization_CEnter.vi



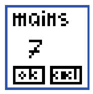
INM04_INitialisation_Menue04(shaft).vi
D:\Version 2.0\INitialization.llb\INM04_INitialisation_Menue04(shaft).vi



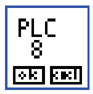
INM05_INitialisation_Menue05(motor-load).vi
D:\Version 2.0\INitialization.llb\INM05_INitialisation_Menue05(motor-load).vi



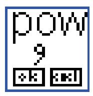
INM06_INitialisation_Menue06(inverter-load).vi
D:\Version 2.0\INitialization.llb\INM06_INitialisation_Menue06(inverter-load).vi



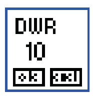
INM07_INitialisation_Menue07(mains).vi
D:\Version 2.0\INitialization.llb\INM07_INitialisation_Menue07(mains).vi



INM08_INitialisation_Menue08(PLC).vi
D:\Version 2.0\INitialization.llb\INM08_INitialisation_Menue08(PLC).vi

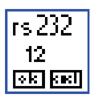


INM09_INitialisation_Menue09(Poweranalyzer).vi
D:\Version 2.0\INitialization.llb\INM09_INitialisation_Menue09(Poweranalyzer).vi

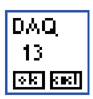


INM10_INitialisation_Menue10(dewerack).vi
D:\Version 2.0\INitialization.llb\INM10_INitialisation_Menue10(dewerack).vi

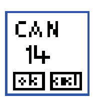
INM11_INitialisation_Menue11(GPIB).vi
D:\Version 2.0\INitialization.llb\INM11_INitialisation_Menue11(GPIB).vi



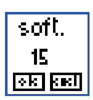
INM12_INitialisation_Menue12(rs232).vi
D:\Version 2.0\INitialization.llb\INM12_INitialisation_Menue12(rs232).vi



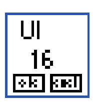
INM13_INitialisation_Menue13(DAQ).vi
D:\Version 2.0\INitialization.llb\INM13_INitialisation_Menue13(DAQ).vi



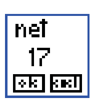
INM14_INitialisation_Menue14(CAN).vi
D:\Version 2.0\INitialization.llb\INM14_INitialisation_Menue14(CAN).vi



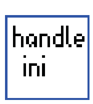
INM15_INitialisation_Menue15(software).vi
D:\Version 2.0\INitialization.llb\INM15_INitialisation_Menue15(software).vi



INM16_INitialisation_Menue16(UI).vi
D:\Version 2.0\INitialization.llb\INM16_INitialisation_Menue16(UI).vi



INM17_INitialisation_Menue17(network).vi
D:\Version 2.0\INitialization.llb\INM17_INitialisation_Menue17(network).vi



INIF_INitialization_Ini_File.vi
D:\Version 2.0\INitialization.llb\INIF_INitialization_Ini_File.vi

GLCP_GLobal_Check_Path.vi
D:\Version 2.0\global.llb\GLCP_GLobal_Check_Path.vi

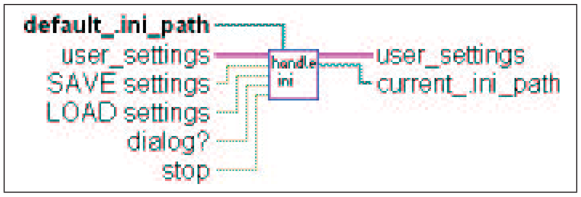


Figure 47: Connector Pane of INIF_INitialization_Ini_File.vi

15.6 History

INCE_INitialization_CEnter.vi History Current Revision: 76

16 INIF_INitialization_Ini_File.vi

This VI saves and loads Settings into and from a .ini file.

*LOLA*Settings are saved in files called LOLA.ini. The default .ini file is \settings\LOLA.ini.

16.1 Connector Pane

See figure 47 on page 46.

16.2 Front Panel

See figure 48 on page 47.

16.3 Controls and Indicators

user settings

- **general**
  - **Numeric_for_nothing** this numeric is for nothing
  - **standard_path** is the path where the program is located
  - **list_column_names** is an array containing the titles of the measurement categories
    The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**
  - **semaphores** is an array of names of all existing semaphores
    - **name**
- **Dewerack_serial**
  - **baud_rate**
  - **latency_readin** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
  - **latency_writeout** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
  - **timeout_value_readin** (not in use at the moment)
  - **timeout_value_writeout** (not in use at the moment)

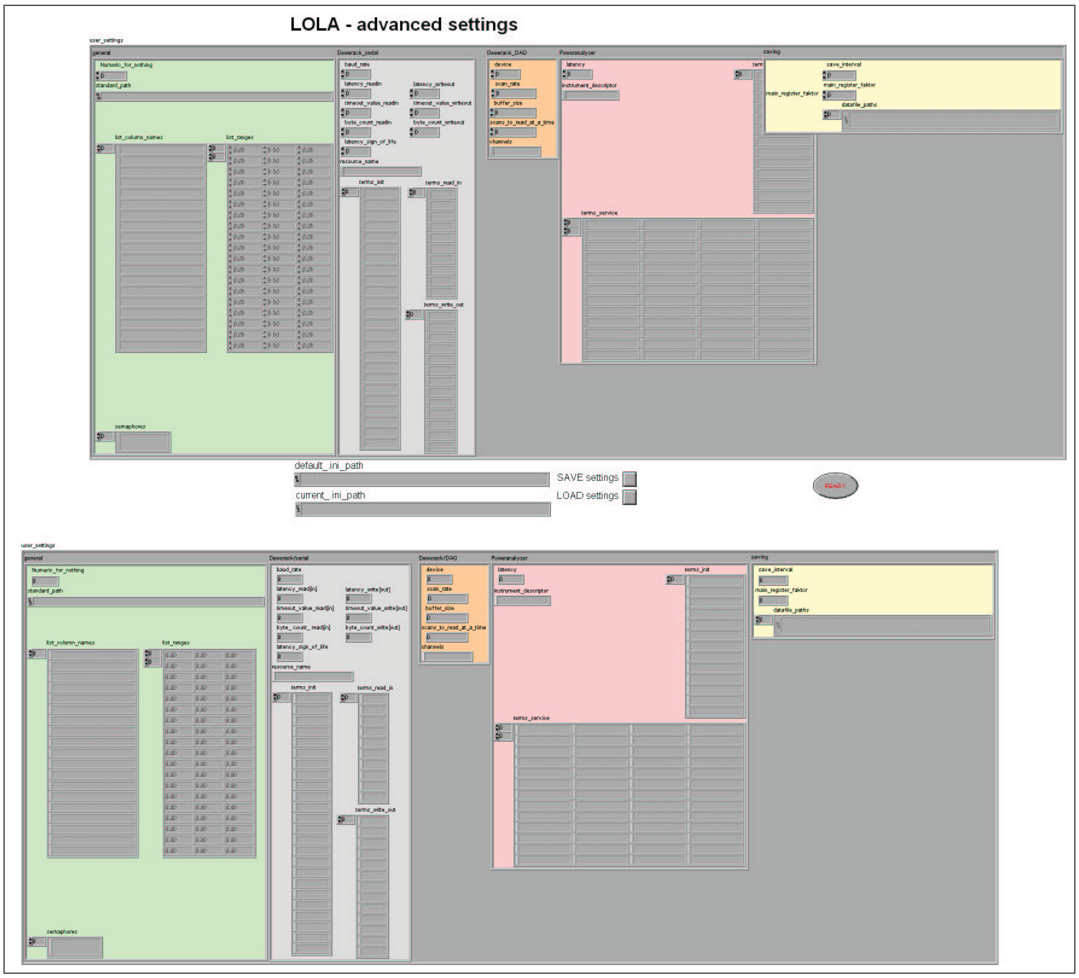


Figure 48: Front Panel of INIF_INitialization_Ini_File.vi

- **byte_count_readin** is the number of bytes to be read
- **byte_count_writeout** is the number of bytes to be written
- **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
- **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
- **terms_init** is an array of commands for initialization of the Dewerack
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)

Dewerack_DAQ

- **device** is the device number of the DAQ board
- **scan_rate** is the number of scans performed per second
- **buffer_size** buffer size is the number of scans each buffer should hold
- **scans_to_read_at_a_time** number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
  Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.
- **channels** specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Poweranalyzer

- **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
- **instrument_descriptor** is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver
- **terms_init** is an array of commands (called terms here) for the initialization of the Poweranalyzer
  - **term** for the initialization request at the Poweranalyzer (called term here)
- **terms_service** is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases
  - **term** command for the read request at the Poweranalyzer (called term here)

saving

- **save_interval** determines the amount of Measurement-Data saved
- **main_register_faktor** determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*

- **datafile_paths** name and location of files where the measured data is saved
  - **path** consists of path (location) and filename

SAVE settings

LOAD settings

stop

default_.ini_path

dialog?

current_.ini_path

user settings[out]

- **general**
  - **Numeric_for_nothing** this numeric is for nothing
  - **standard_path** is the path where the program is located
  - **list_column_names** is an array containing the titles of the measurement categories
    The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**
  - **semaphores** is an array of names of all existing semaphores
    - **name**
- **Dewerack/serial**
  - **baud_rate**
  - **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
  - **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
  - **timeout_value_read[in]** (not in use at the moment)
  - **timeout_value_write[out]** (not in use at the moment)
  - **byte_ count_ read[in]** is the number of bytes to be read
  - **byte_count_write[out]** is the number of bytes to be written
  - **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
  - **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
  - **terms_init** is an array of commands for initialization of the Dewerack
    - **term** address respectively command for the read request at a certain port (called term here)
  - **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
    - **term** address respectively command for the read request at a certain port (called term here)
  - **terms_write_out** is an array of addresses respectively commands for the write request at a certain port

term address respectively command for the read request at a certain port (called term here)

Dewerack/DAQ

device is the device number of the DAQ board

scan_rate is the number of scans performed per second

buffer_size buffer size is the number of scans each buffer should hold

scans_to_read_at_a_time number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.

channels specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.

Poweranalyzer

latency is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop

instrument_descriptor is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver

terms_init is an array of commands (called terms here) for the initialization of the Poweranalyzer

term for the initialization request at the Poweranalyzer (called term here)

terms_service is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases

term command for the read request at the Poweranalyzer (called term here)

saving

save_interval determines the amount of Measurement-Data saved

main_register_faktor determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*

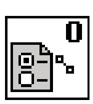
datafile_paths name and location of files where the measured data is saved

path consists of path (location) and filename

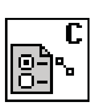
16.4 Block Diagram

See figure 49 on page 51, figure 50 on page 52, and See figure 51 on page 53.

16.5 List of SubVIs



Open Config Data.vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Open Config Data.vi



Close Config Data.vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Close Config Data.vi

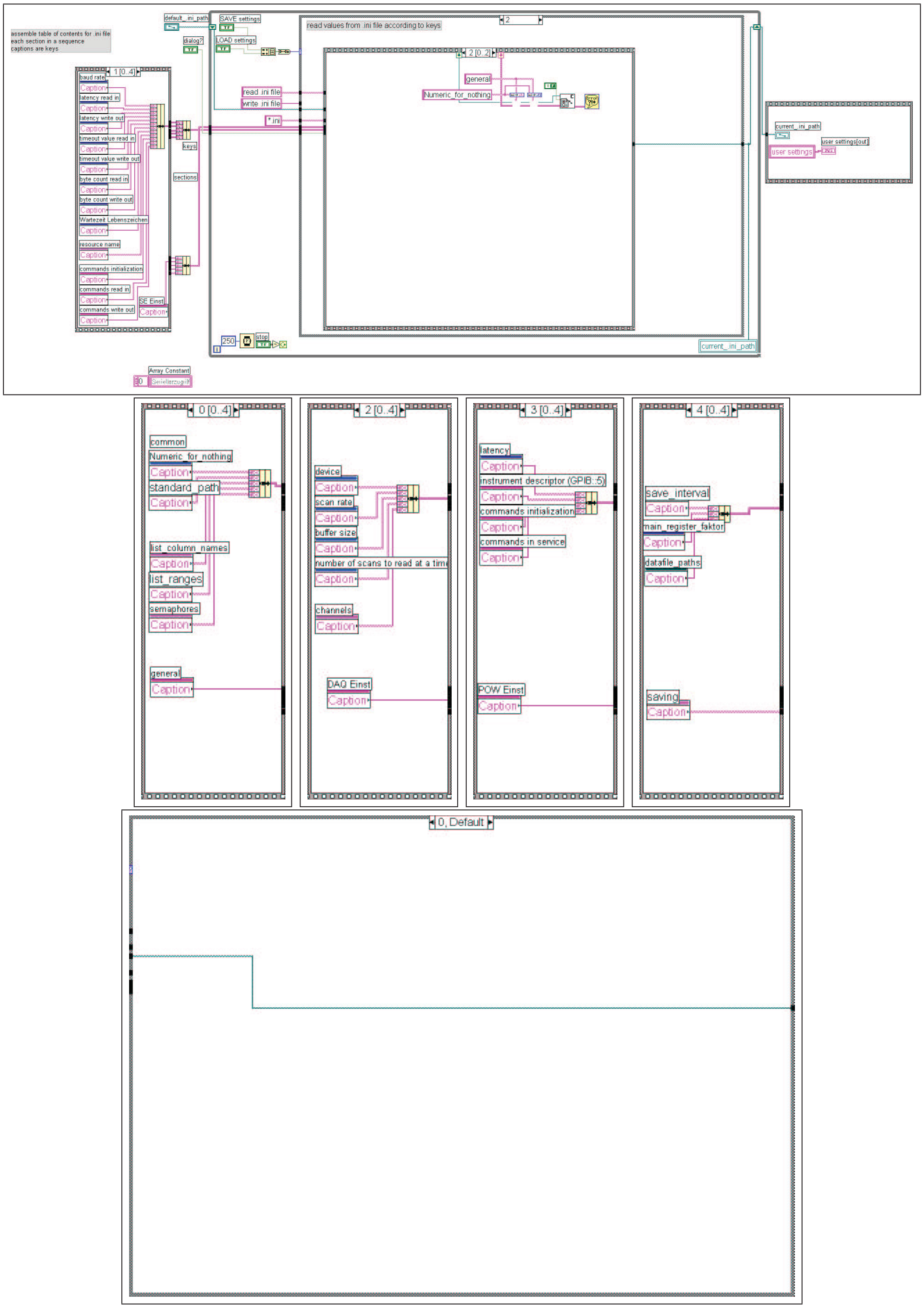


Figure 49: Block Diagram (a) of INIF_INitialization_Ini_File.vi

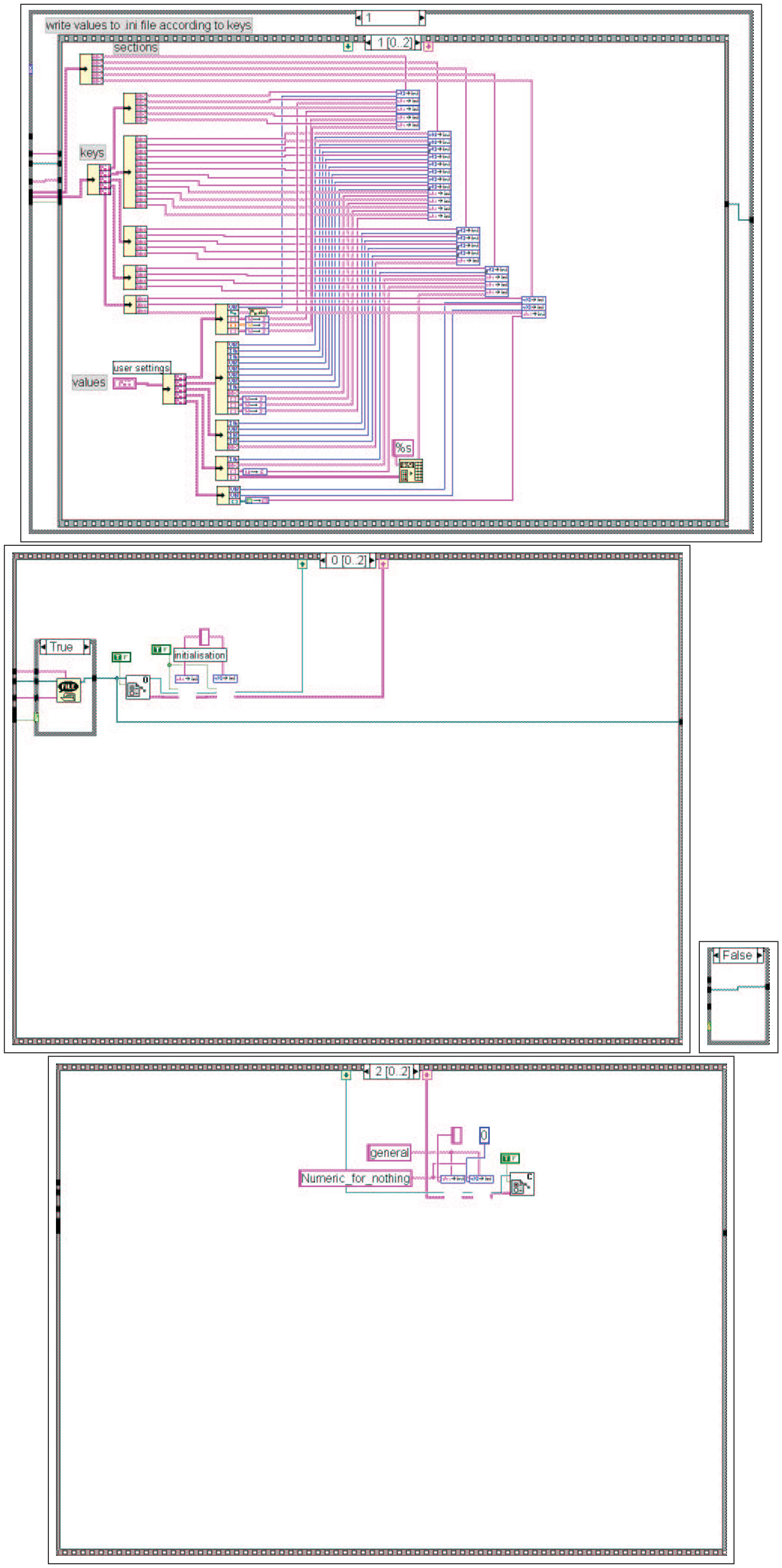


Figure 50: Block Diagram (b) of INIF_INitialization_Ini_File.vi

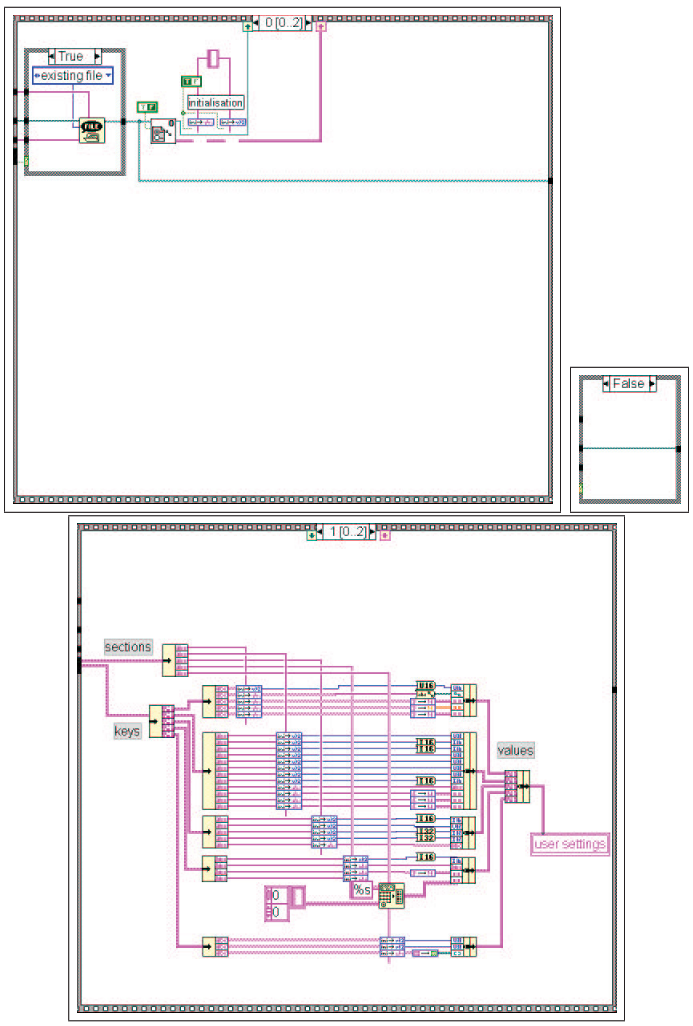
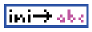


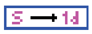
Figure 51: Block Diagram (c) of INIF_INitialization_Ini_File.vi

IN1S_INitialization_1dstring_array_to_Spradsheet.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\-IN1S_INitialization_1dstring_array_to_Spradsheet.vi

INRN_INitialization_Read_Number.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INRN_INitialization_Read_Number.vi



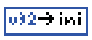
INRS_INitialization_Read_String.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INRS_INitialization_Read_String.vi



INS1_Spradsheet_to_1dstring_array.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INS1_Spradsheet_to_1dstring_array.vi

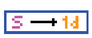
INPS_INitialization_Path_array_to_Spradsheet.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\-INPS_INitialization_Path_array_to_Spradsheet.vi

INSP_Initialization_Spradsheet_to_Path_array.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\-INSP_Initialization_Spradsheet_to_Path_array.vi

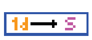


INWN_INitialization_Write_Number.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INWN_INitialization_Write_Number.vi

INWS_INitialization_Write_String.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INWS_INitialization_Write_String.vi



INSN_spradsheet_to_1dNumber_array.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\INSN_spradsheet_to_1dNumber_array.vi



IN1N_INitialization_1dNatospradsh.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\INitialization.llb\IN1N_INitialization_1dNatospradsh.vi

GCEH_Global_Changed_Error_Handler.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GCEH_Global_Changed_Error_Handler.vi

16.6 History

INIF_INitialization_Ini_File.vi History Current Revision: 137

17 ININ_INitialistation_INitialisation.vi

Initialization of all Temporary-Storages.

17.1 Connector Pane

See figure 52 on page 55.

17.2 Front Panel

See figure 53 on page 55.

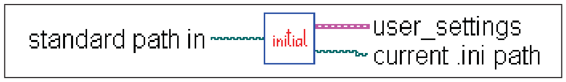


Figure 52: Connector Pane of ININ_INitialistation_INitialisation.vi

Figure 53: Front Panel of ININ_INitialistation_INitialisation.vi

17.3 Controls and Indicators

standard path in

user settings

- **general**
  - **Numeric_for_nothing** this numeric is for nothing
  - **standard_path** is the path where the program is located
  - **list_column_names** is an array containing the titles of the measurement categories
    The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**
  - **semaphores** is an array of names of all existing semaphores
    - **name**
- **Deweracks/serial**
  - **baud_rate**
  - **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
  - **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
  - **timeout_value_read[in]** (not in use at the moment)
  - **timeout_value_write[out]** (not in use at the moment)
  - **byte_ count_ read[in]** is the number of bytes to be read
  - **byte_count_write[out]** is the number of bytes to be written
  - **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
  - **resource_name** is the instrument descriptor of the interface addressing the Dewerack for VISA driver
  - **terms_init** is an array of commands for initialization of the Dewerack
    - **command** address respectively command for the initialization request at a certain port
  - **terms_read_in** is an array of addresses respectively commands for the read request at a certain port

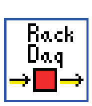
    - **command** address respectively command for the read request at a certain port
  - **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
    - **command** address respectively command for the write request at a certain port
- **Dewerack/DAQ**
  - **device** is the device number of the DAQ board
  - **scan_rate** is the number of scans performed per second
  - **buffer_size** buffer size is the number of scans each buffer should hold
  - **scans_to_read_at_a_time** number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
    Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.
  - **channels** specifies the set of analog input channels for a group and task. You cannot assign a channel to more than one group. The default input is channel 0. See the description of the Analog Input Group Config VI for a detailed description of this parameter and the valid syntax for the channel strings.
- **Poweranalyzer**
  - **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
  - **instrument_descriptor** is the instrument descriptor of the interface addressing the Poweranalyzer for the VISA driver
  - **terms_init** is an array of commands for the initialization of the Poweranalyzer
    - **command** for the initialization request at the Poweranalyzer
  - **terms_service** is an array of commands for the read request
    the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases
    - **term** command for the read request at the Poweranalyzer
- **saving**
  - **save_interval** determines the amount of Measurement-Data saved
  - **main_register_faktor** determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*
  - **datafile_paths** name and location of files where the measured data is saved
    - **path** consists of path (location) and filename

current .ini path

17.4 Block Diagram

See figure 54 on page 57.

17.5 List of SubVIs



TSRD_TemporaryStorage_Rack_Daq.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSRD_TemporaryStorage_Rack_Daq.vi

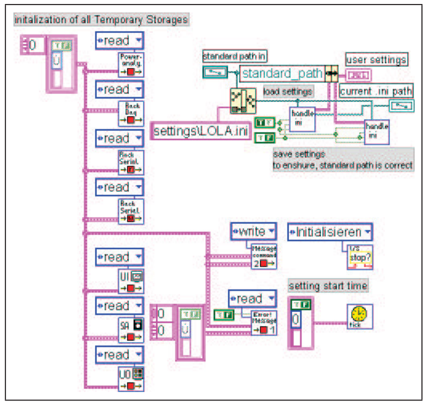
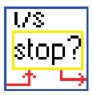
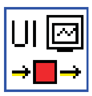


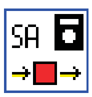
Figure 54: Block Diagram of ININ_INitialistation_INitialisation.vi



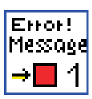
GLSA_Stop_to_All.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Global.llb\GLSA_Stop_to_All.vi



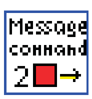
TSUI_TemporaryStorage_UserInterface.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSUI_TemporaryStorage_UserInterface.vi



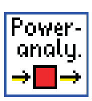
TSSA_TemporaryStorage_SAve.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-
Temporary_Storages.llb\TSSA_TemporaryStorage_SAve.vi



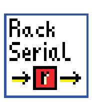
TSM1_TemporaryStorage_Messages1.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-
Temporary_Storages.llb\TSM1_TemporaryStorage_Messages1.vi



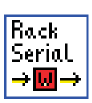
TSM2_TemporaryStorage_Messages2.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-
Temporary_Storages.llb\TSM2_TemporaryStorage_Messages2.vi



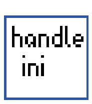
TSPA_TemporaryStorage_PowerAnalyzer.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSPA_TemporaryStorage_PowerAnalyzer.vi



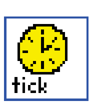
TSRSr_TemporaryStorage_Rack_Serialread.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSRSr_TemporaryStorage_Rack_Serialread.vi



TSRSw_TemporaryStorage_Rack_Serialwrite.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSRSw_TemporaryStorage_Rack_Serialwrite.vi



INIF_INitialization_Ini_File.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\INitialization.llb\INIF_INitialization_Ini_File.vi



GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Global.llb\GLCL_GLobal_CLock.vi

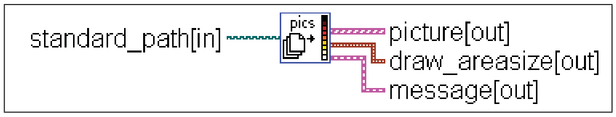


Figure 55: Connector Pane of INLP_INitialisation_LoadPictures.vi

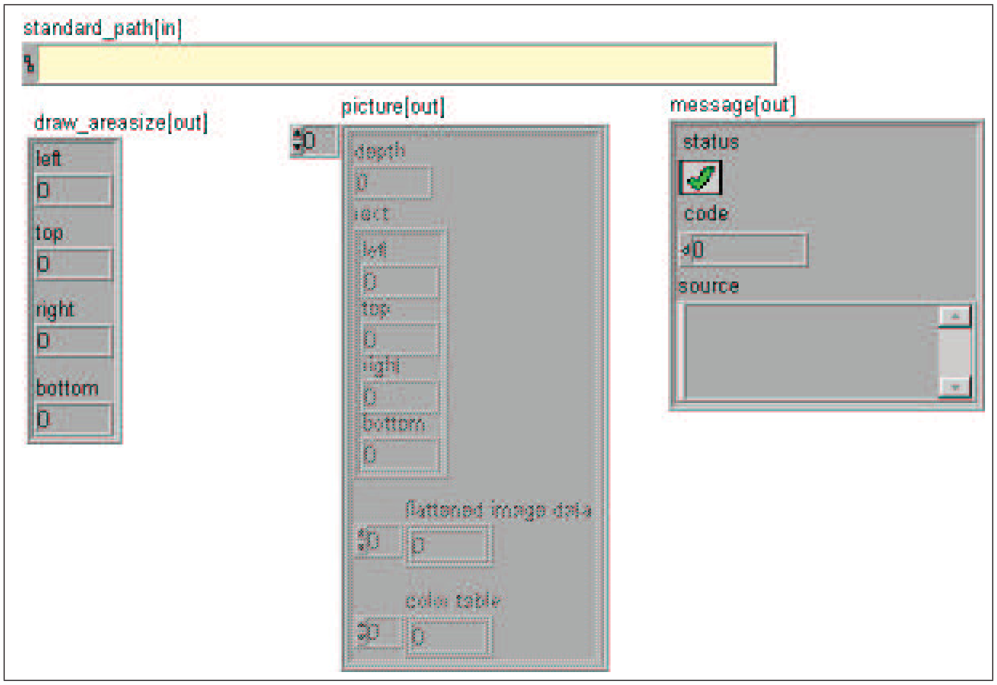
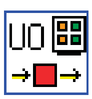


Figure 56: Front Panel of INLP_INitialisation_LoadPictures.vi



TSUO_TemporaryStorage_UserOperation.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSUO_TemporaryStorage_UserOperation.vi

17.6 History

ININ_INitialistation_INitialisation.vi History Current Revision: 106

18 INLP_INitialisation_LoadPictures.vi

This VI reads picture files that are necessary for the front panel of INCE_INitialization_CEnter.vi.

18.1 Connector Pane

See figure 55 on page 58.

18.2 Front Panel

See figure 56 on page 58.

18.3 Controls and Indicators

 **standard_path[in]** path where the programm is located

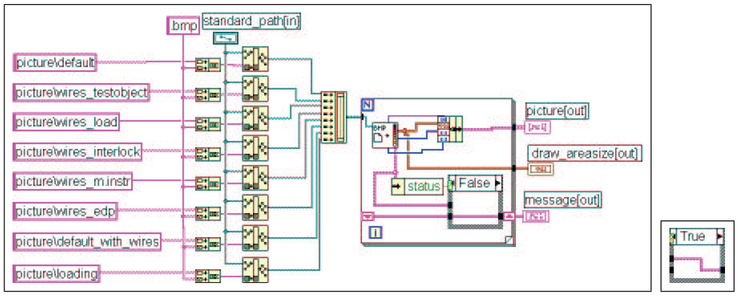


Figure 57: Block Diagram of INLP_INitialisation_LoadPictures.vi

picture[out]

- **picture_cluster**
  - **depth**
  - **rect**
    - **left**
    - **top**
    - **right**
    - **bottom**
  - **flattened image data**
  - **color table**

draw_areasize[out]

- **left**
- **top**
- **right**
- **bottom**

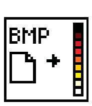
message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

18.4 Block Diagram

See figure 57 on page 59.

18.5 List of SubVIs



Read BMP File.vi
C:\Programme\National Instruments\Labview\vi.lib\picture\bmp.llb\Read BMP File.vi

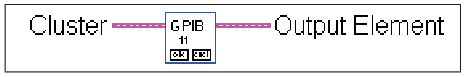


Figure 58: Connector Pane of INM11_INitialisation_Menue11(GPIB).vi

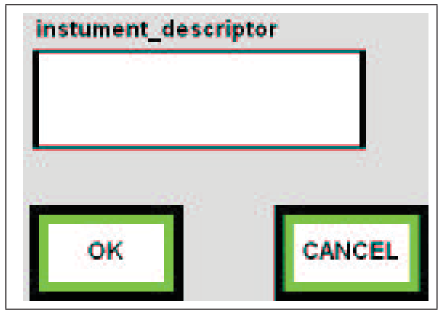


Figure 59: Front Panel of INM11_INitialisation_Menue11(GPIB).vi

18.6 History

INLP_INitialisation_LoadPictures.vi History Current Revision: 15

19 INM11_INitialisation_Menue11(GPIB).vi

prompts the settings for GPIB interface

19.1 Connector Pane

See figure 58 on page 60.

19.2 Front Panel

See figure 59 on page 60.

19.3 Controls and Indicators

cancel

ok

instument_descriptor

Cluster

- **Output Element**

Output Element

- **instument_descriptor**

19.4 Block Diagram

See figure 60 on page 61.

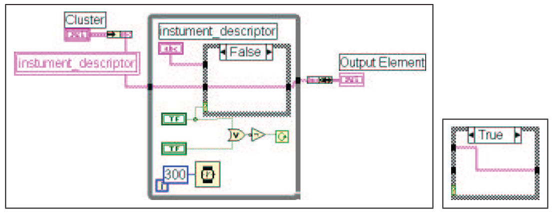


Figure 60: Block Diagram of INM11_INitialisation_Menue11(GPIB).vi

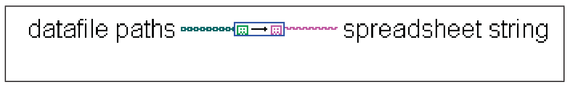


Figure 61: Connector Pane of INPS_INitialization_Path_array_to_Spradsheet.vi

19.5 List of SubVIs

19.6 History

INM11_INitialisation_Menue11(GPIB).vi History Current Revision: 35

20 INPS_INitialization_Path_array_to_Spradsheet.vi

converts path to spreadsheet string

20.1 Connector Pane

See figure 61 on page 61.

20.2 Front Panel

See figure 62 on page 62.

20.3 Controls and Indicators

data-file paths array of paths

- **path**

spreadsheet string

20.4 Block Diagram

See figure 63 on page 62.

20.5 List of SubVIs

20.6 History

INPS_INitialization_Path_array_to_Spradsheet.vi History Current Revision: 7

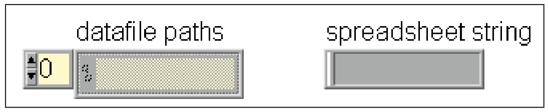


Figure 62: Front Panel of INPS_INitialization_Path_array_to_Spradsheet.vi

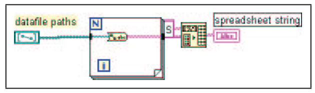


Figure 63: Block Diagram of INPS_INitialization_Path_array_to_Spradsheet.vi

21 INRN_INitialization_Read_Number.vi

reads out U32 of the .ini file

initialization necessary, so that the file ID has not to be handed over for further readings

21.1 Connector Pane

See figure 64 on page 63.

21.2 Front Panel

See figure 65 on page 63.

21.3 Controls and Indicators

error[in]

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

section The name of the section from which to read the specified key.

key The name of the key to read.

init? true, if VI is initialized

refnum in The reference number of the configuration data.

error[out]

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

value the wanted *value* of *key* in *section*

refnum out The reference number of the configuration data.

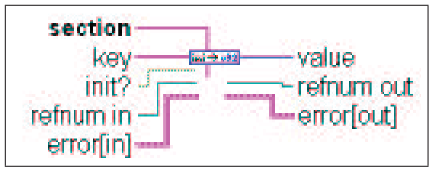


Figure 64: Connector Pane of INRN_INitialization_Read_Number.vi

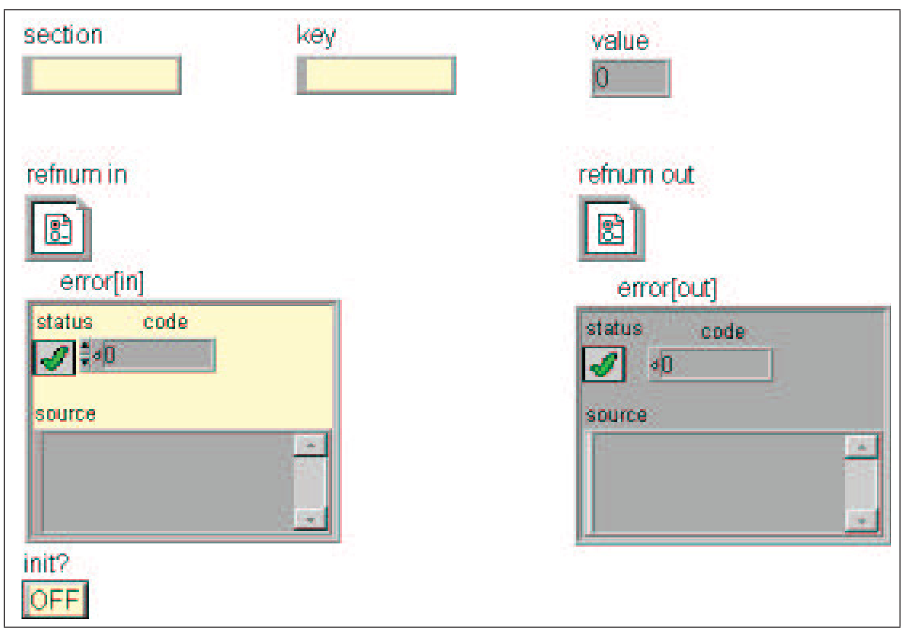


Figure 65: Front Panel of INRN_INitialization_Read_Number.vi

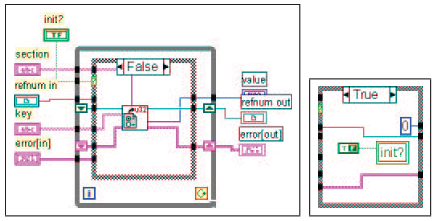


Figure 66: Block Diagram of INRN_INitialization_Read_Number.vi

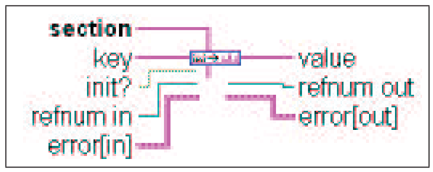
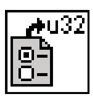


Figure 67: Connector Pane of INRS_INitialization_Read_String.vi

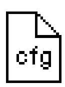
21.4 Block Diagram

See figure 66 on page 64.

21.5 List of SubVIs



Read Key (U32).vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Read Key (U32).vi



Config Data RefNum
C:\Programme\National Instruments\Labview\vi.lib\Utility\config.llb\Config Data RefNum

21.6 History

INRN_INitialization_Read_Number.vi History Current Revision: 23

22 INRS_INitialization_Read_String.vi

reads out string of the .ini file

initialization necessary, so that the file ID has not to be handed over for further readings

22.1 Connector Pane

See figure 67 on page 64.

22.2 Front Panel

See figure 68 on page 65.

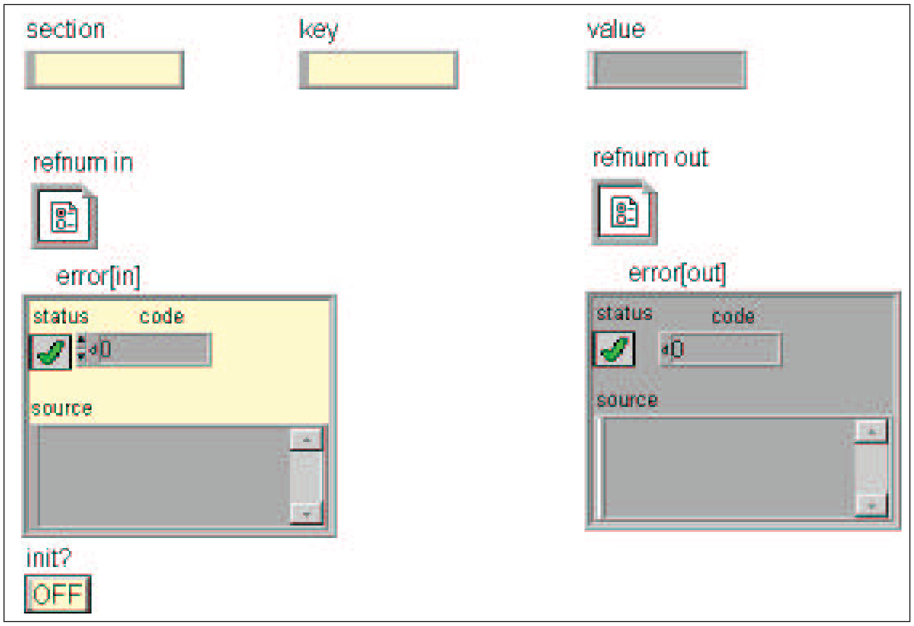


Figure 68: Front Panel of INRS_INitialization_Read_String.vi

22.3 Controls and Indicators

error[in]

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

section The name of the section from which to read the specified key.

key The name of the key to read.

init? true, if VI is initialized

refnum in The reference number of the configuration data.

error[out]

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

value the wanted *value* of *key* in *section*

refnum out The reference number of the configuration data.

22.4 Block Diagram

See figure 69 on page 66.

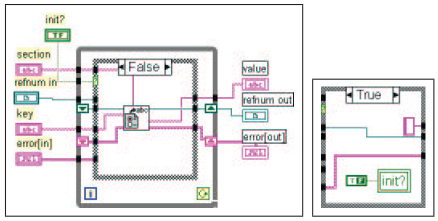
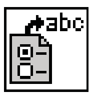


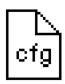
Figure 69: Block Diagram of INRS_INitialization_Read_String.vi

Figure 70: Connector Pane of INS1_Spradsheet_to_1dstring_array.vi

22.5 List of SubVIs



Read Key (String).vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Read Key (String).vi



Config Data RefNum
C:\Programme\National Instruments\Labview\vi.lib\Utility\config.llb\Config Data RefNum

22.6 History

INRS_INitialization_Read_String.vi History Current Revision: 22

23 INS1_Spradsheet_to_1dstring_array.vi

converts spreadsheet string to one dimension array of string

23.1 Connector Pane

See figure 70 on page 66.

23.2 Front Panel

See figure 71 on page 67.

23.3 Controls and Indicators

 **spreadsheet string**

 **array**

23.4 Block Diagram

See figure 72 on page 67.

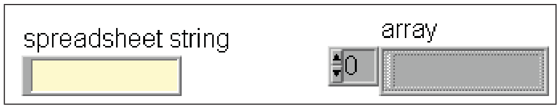


Figure 71: Front Panel of INS1_Spradsheet_to_1dstring_array.vi

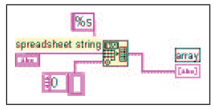


Figure 72: Block Diagram of INS1_Spradsheet_to_1dstring_array.vi

23.5 List of SubVIs

23.6 History

INS1_Spradsheet_to_1dstring_array.vi History Current Revision: 10

24 INSN_spradsheet_to_1dNumber_array.vi

converts spreadsheet string to one dimension array of number (DBL)

24.1 Connector Pane

See figure 73 on page 68.

24.2 Front Panel

See figure 74 on page 68.

24.3 Controls and Indicators

spreadsheet string

array

Numeric

24.4 Block Diagram

See figure 75 on page 69.

24.5 List of SubVIs

24.6 History

INSN_spradsheet_to_1dNumber_array.vi History Current Revision: 12

25 INSP_Initialization_Spradsheet_to_Path_array.vi

converts string to path

Figure 73: Connector Pane of INSN_spradsheet_to_1dNumber_array.vi

Figure 74: Front Panel of INSN_spradsheet_to_1dNumber_array.vi

25.1 Connector Pane

See figure 76 on page 69.

25.2 Front Panel

See figure 77 on page 69.

25.3 Controls and Indicators

spreadsheet string

data-file paths

path

25.4 Block Diagram

See figure 78 on page 69.

25.5 List of SubVIs

25.6 History

INSP_Initialization_Spradsheet_to_Path_array.vi History Current Revision: 8

26 INWN_INitialization_Write_Number.vi

reads out U32 of the .ini file

initialization necessary, so that the file ID has not to be handed over for further readings

26.1 Connector Pane

See figure 79 on page 69.

26.2 Front Panel

See figure 80 on page 69.

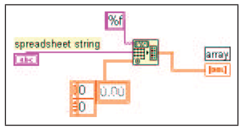


Figure 75: Block Diagram of INSN_spradsheet_to_1dNumber_array.vi

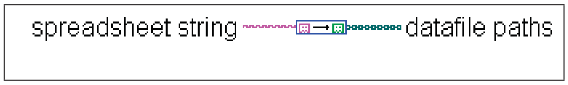


Figure 76: Connector Pane of INSP_Initialization_Spradsheet_to_Path_array.vi

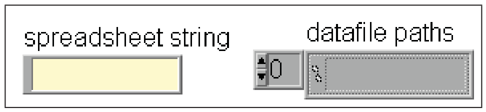


Figure 77: Front Panel of INSP_Initialization_Spradsheet_to_Path_array.vi

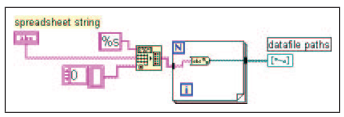


Figure 78: Block Diagram of INSP_Initialization_Spradsheet_to_Path_array.vi

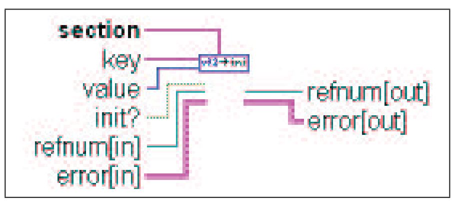


Figure 79: Connector Pane of INWN_INitialization_Write_Number.vi

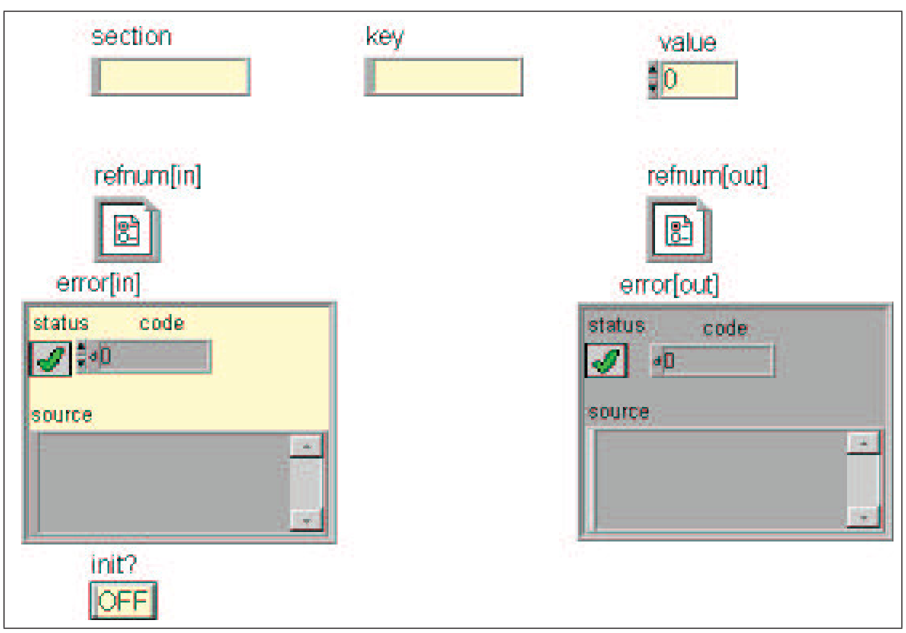


Figure 80: Front Panel of INWN_INitialization_Write_Number.vi

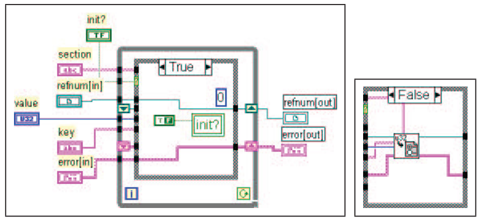


Figure 81: Block Diagram of INWN_INitialization_Write_Number.vi

26.3 Controls and Indicators

error[in]

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

section The name of the section from which to read the specified key.

key The name of the key to read.

init? true, if VI is initialized

value the wanted *value* of *key* in *section*

refnum[in] The reference number of the configuration data.

error[out]

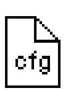
- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

refnum[out] The reference number of the configuration data.

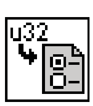
26.4 Block Diagram

See figure 81 on page 70.

26.5 List of SubVIs



Config Data RefNum
C:\Programme\National Instruments\Labview\vi.lib\Utility\config.llb\Config Data RefNum



Write Key (U32).vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Write Key (U32).vi

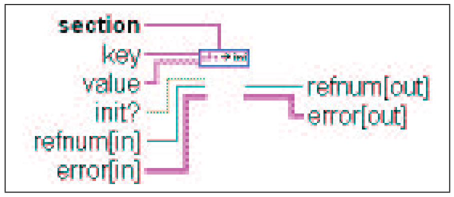


Figure 82: Connector Pane of INWS_INitialization_Write_String.vi

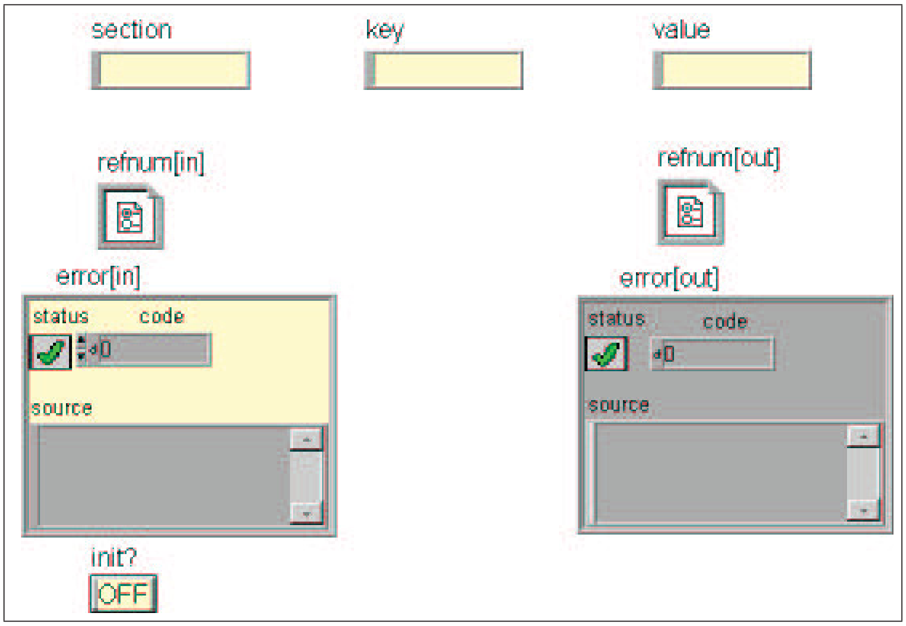


Figure 83: Front Panel of INWS_INitialization_Write_String.vi

26.6 History

INWN_INitialization_Write_Number.vi History Current Revision: 26

27 INWS_INitialization_Write_String.vi

reads out string of the .ini file

initialization necessary, so that the file ID has not to be handed over for further readings

27.1 Connector Pane

See figure 82 on page 71.

27.2 Front Panel

See figure 83 on page 71.

27.3 Controls and Indicators

 **error[in]**

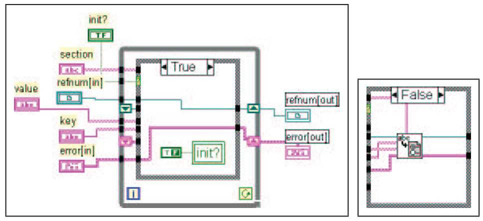


Figure 84: Block Diagram of INWS_INitialization_Write_String.vi

- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

section The name of the section from which to read the specified key.

key The name of the key to read.

value the wanted *value* of *key* in *section*

init? true, if VI is initialized

refnum[in] The reference number of the configuration data.

error[out]

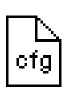
- **status** is TRUE if an error occurred, or FALSE if not. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **code** is the number identifying an error or warning. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero or a warning code.
- **source** indicates the origin of the error, if any. Usually *source* is the name of the VI in which the error occurred.

refnum[out] The reference number of the configuration data.

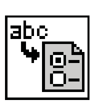
27.4 Block Diagram

See figure 84 on page 72.

27.5 List of SubVIs



Config Data RefNum
C:\Programme\National Instruments\Labview\vi.lib\Utility\config.llb\Config Data RefNum



Write Key (String).vi
C:\Programme\National Instruments\Labview\vi.lib\UTILITY\config.llb\Write Key (String).vi

27.6 History

INWS_INitialization_Write_String.vi History Current Revision: 24

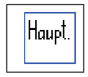


Figure 85: Connector Pane of LOLA_Low_Observing_Laboratory_Application.vi

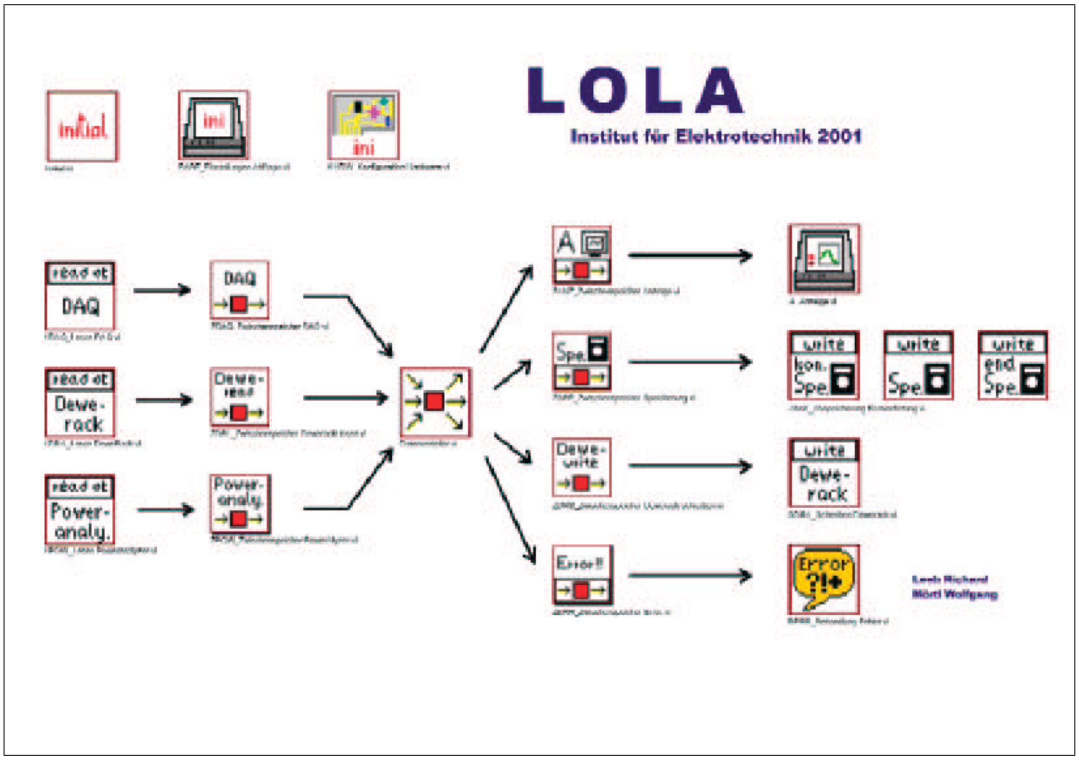


Figure 86: Front Panel of LOLA_Low_Observing_Laboratory_Application.vi

28 LOLA_Low_Observing_Laboratory_Application.vi

This is the main program.

28.1 Connector Pane

See figure 85 on page 73.

28.2 Front Panel

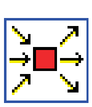
See figure 86 on page 73.

28.3 Controls and Indicators

28.4 Block Diagram

See figure 87 on page 74.

28.5 List of SubVIs



DICE_DIstribution_CEnter.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Distribution.llb\DICE_DIstribution_CEnter.vi

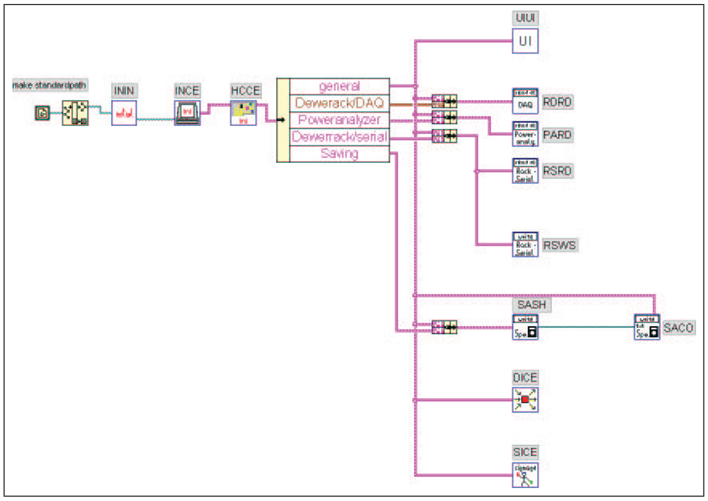
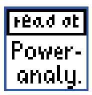


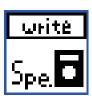
Figure 87: Block Diagram of LOLA_Low_Observing_Laboratory_Application.vi



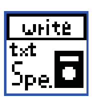
PARD_PowerAnalyzer_ReadData.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Poweranalyzer.llb\PARD_PowerAnalyzer_ReadData.vi

RDRD_RackDAQ_ReadData.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\RackDaq.llb\RDRD_RackDAQ_ReadData.vi

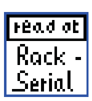
SICE_SIgnage_CEnter.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Signage.llb\SICE_SIgnage_CEnter.vi



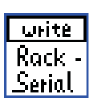
SASH_SAveSHot.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\SAve.llb\SASH_SAveSHot.vi



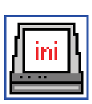
SACO_SAve_COnversion.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\SAve.llb\SACO_SAve_COnversion.vi



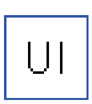
RSRD_RackSerial_ReadData.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\RackSerial.llb\RSRD_RackSerial_ReadData.vi



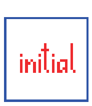
RSWS_RackSerial_WriteS7.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\RackSerial.llb\RSWS_RackSerial_WriteS7.vi



INCE_INitialization_CEnter.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\INitialization.llb\INCE_INitialization_CEnter.vi



UIUI_UserInterface_UserInterface.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\UserInterface.llb\UIUI_UserInterface_UserInterface.vi



ININ_INitialistation_INitialisation.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\INitialization.llb\ININ_INitialistation_INitialisation.vi

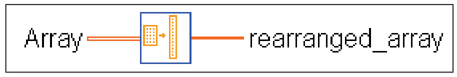


Figure 88: Connector Pane of PARC_PowerAnalyzer_RearrangeChannels.vi

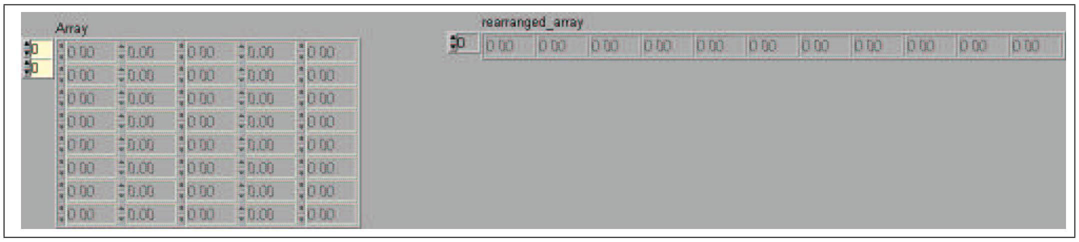
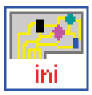


Figure 89: Front Panel of PARC_PowerAnalyzer_RearrangeChannels.vi



HCCE_HardwareConfig_CEnter.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\HardwareConfig.llb\HCCE_HardwareConfig_CEnter.vi

28.6 History

LOLA_Low_Observing_Laboratory_Application.vi History Current Revision: 98

29 PARC_PowerAnalyzer_RearrangeChannels.vi

This array changes the order of elements in the first three columns to allow an easier handling in UIUI_UserInterface_UserInterface.vi.

29.1 Connector Pane

See figure 88 on page 75.

29.2 Front Panel

See figure 89 on page 75.

29.3 Controls and Indicators

[DBL] **Array** Array to be rearranged.

  DBL **Measured Value**

[DBL] **rearranged_array** Rearranged array.

  DBL **Measured Value**

29.4 Block Diagram

See figure 90 on page 76.

29.5 List of SubVIs

29.6 History

PARC_PowerAnalyzer_RearrangeChannels.vi History Current Revision: 3

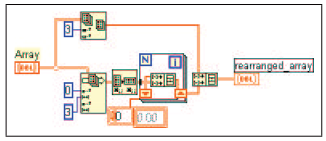


Figure 90: Block Diagram of PARC_PowerAnalyzer_RearrangeChannels.vi

Figure 91: Connector Pane of PARD_PowerAnalyzer_ReadData.vi

30 PARD_PowerAnalyzer_ReadData.vi

This VI is designed to read data (measurement data) from instruments connected to the PC by a GPIB Interface. At the moment one instrument, Poweranalyzer4000 is connected. All actual values of the three phases are acquired with this VI and are passed on to the VI TSPA_TemporayStorage_PowerAnalyzer.vi.

30.1 Connector Pane

See figure 91 on page 76.

30.2 Front Panel

See figure 92 on page 77.

30.3 Controls and Indicators

- **hardware_settings**
  - **general**
    - **standard_path** path where the program is located
    - **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
    - **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
      - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
    - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
      - **value**
    - **semaphores** is an array of names of all existing semaphores
      - **name**
  - **Poweranalyzer**
    - **latency** is the time in milliseconds for PARD_PowerAnalyzer_ReasData.vi to wait between two executions of the while loop
    - **VISA session** is a unique logical identifier used to communicate with a resource.
    - **terms_service** is an array of commands (called terms here) for the read request the structure is: first, second and third collum are for the first, second and third phase; the fourth collum is for all phases

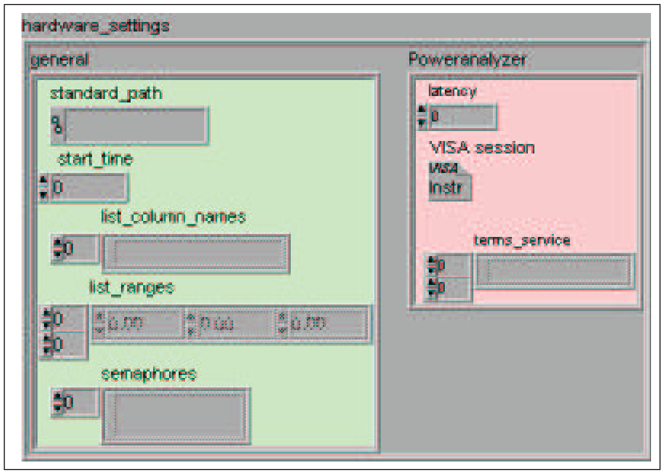


Figure 92: Front Panel of PARD_PowerAnalyzer_ReadData.vi

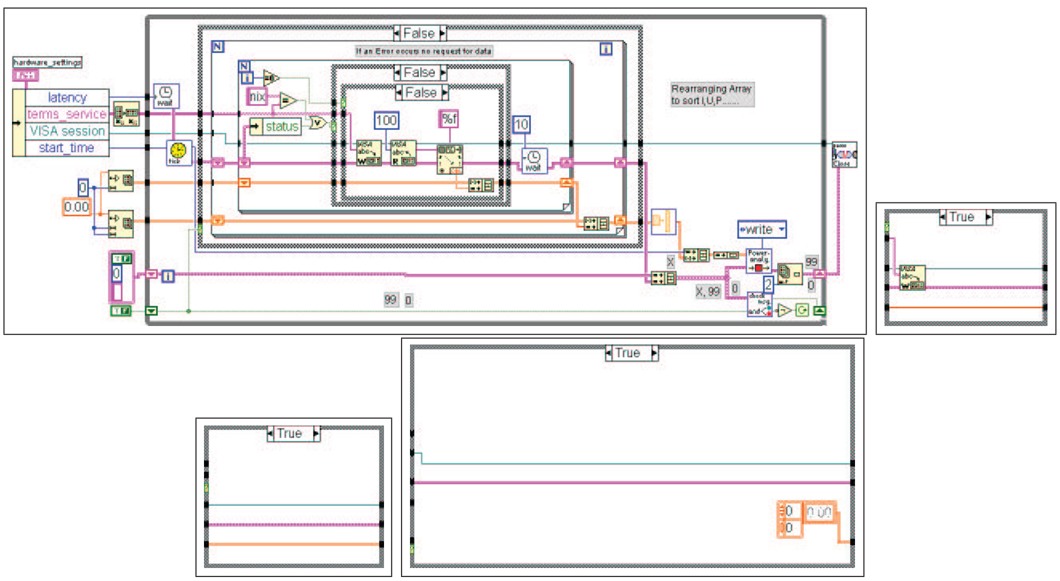


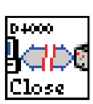
Figure 93: Block Diagram of PARD_PowerAnalyzer_ReadData.vi

term command for the read request at the Poweranalyzer (called term here)

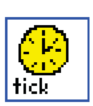
30.4 Block Diagram

See figure 93 on page 77.

30.5 List of SubVIs

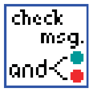


LND4000 Close.vi
C:\Programme\National Instruments\Labview\instr.lib\D4000\LND4000.llb\LND4000 Close.vi

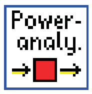


GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCL_GLobal_CLock.vi

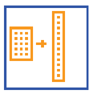
Figure 94: Connector Pane of RDRD_RackDAQ_ReadData.vi



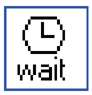
GLCM_GLobal_CheckMessages.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Global.llb\GLCM_GLobal_CheckMessages.vi



TSPA_TemporaryStorage_PowerAnalyzer.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSPA_TemporaryStorage_PowerAnalyzer.vi



PARC_PowerAnalyzer_RearrangeChannels.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Poweranalyzer.llb\-
PARC_PowerAnalyzer_RearrangeChannels.vi



GLWA_GLobalWAit.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\global.llb\GLWA_GLobalWAit.vi

30.6 History

PARD_PowerAnalyzer_ReadData.vi History Current Revision: 113

31 RDRD_RackDAQ_ReadData.vi

Taken from analogin.llb: Acquire N Scans- Async Occurrence.vi

DESCRIPTION:
This is a timed acquisition, meaning that a hardware clock is used to control the acquisition for fast and accurate timing. It is also a continuous circular buffered acquisition. This means that a software buffer is used between your DAQ board and LabView. While data are being transferred from your board into one part of the input buffer, LabView is reading data from another part of that buffer. The occurrence makes the acquisition asynchronous (allowing processor time for other things to run) by causing the loop to wait until the number of scans to read at a time is available before it is read into LabView. For this reason it is important that you set your parameters such that LabView reads data out of the buffer fast enough to keep up with the rate at which your board is writing new data into the buffer. If not, unread data will be overwritten and an error will occur.

DAQ VIs USED:
AI Config.vi, DAQ Occurrence Config.vi, AI Start.vi, AI Read.vi, AI Clear.vi.

31.1 Connector Pane

See figure 94 on page 78.

31.2 Front Panel

See figure 95 on page 79.

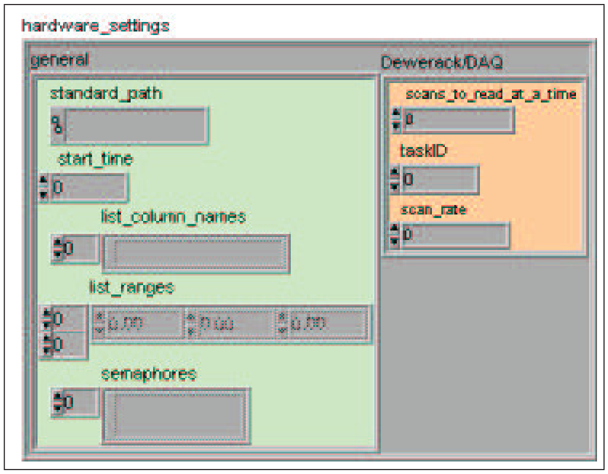


Figure 95: Front Panel of RDRD_RackDAQ_ReadData.vi

31.3 Controls and Indicators

hardware_settings

general

standard_path path where the program is located

start_time [ms] is the time reference. In *LOLAstart_time* is set at INKH_-INitialisation_KonfigHardware.vi at the start of the program.

list_column_names is an array containing the titles of the measurement categories The names' order must match to the commands' order.

name_of_channel channel name is displayed in the user interface, it consists of identifier and unit

list_ranges is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*

value

semaphores is an array of names of all existing semaphores

name

Dewerack/DAQ

scans_to_read_at_a_time number of scans to read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabView to set number of scans to read equal to the value of the number of scans to acquire control when AI Start was called. If number of scans to read is -1 and number of scans to acquire was 0, LabView sets number of scans to read to 100.
Execution of the loop is halted until the *number_of_scans_to_read* is available in the internal memory of the DAQ board.

taskID is a unique logical identifier used to communicate with a resource.

scan_rate is the number of scans performed per second

31.4 Block Diagram

See figure 96 on page 80.

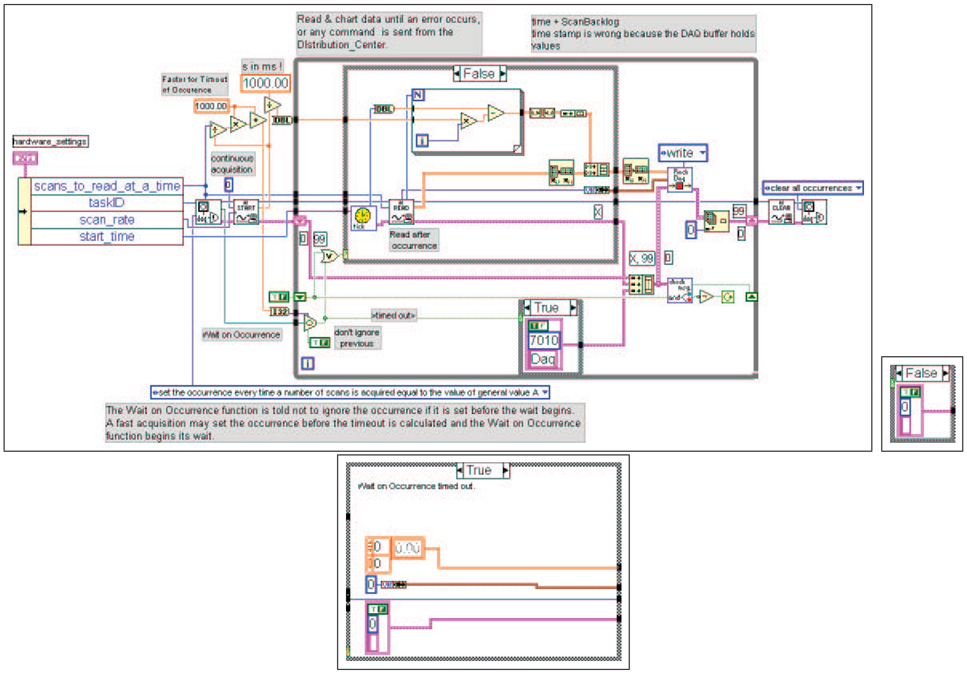
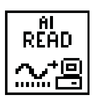
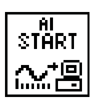


Figure 96: Block Diagram of RDRD_RackDAQ_ReadData.vi

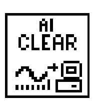
31.5 List of SubVIs



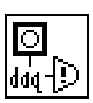
AI Read.vi
C:\Programme\National Instruments\Labview\vi.lib\DAQ\AI.LLB\AI Read.vi



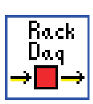
AI Start.vi
C:\Programme\National Instruments\Labview\vi.lib\DAQ\AI.LLB\AI Start.vi



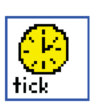
AI Clear.vi
C:\Programme\National Instruments\Labview\vi.lib\DAQ\AI.LLB\AI Clear.vi



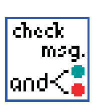
DAQ Occurrence Config.vi
C:\Programme\National Instruments\Labview\vi.lib\DAQ\MISC.LLB\DAQ Occurrence Config.vi



TSRD_TemporaryStorage_Rack_Daq.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSRD_TemporaryStorage_Rack_Daq.vi



GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCL_GLobal_CLock.vi



GLCM_GLobal_CheckMessages.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCM_GLobal_CheckMessages.vi

31.6 History

RDRD_RackDAQ_ReadData.vi History Current Revision: 137

Figure 97: Connector Pane of RSRD_RackSerial_ReadData.vi

Figure 98: Front Panel of RSRD_RackSerial_ReadData.vi

32 RSRD_RackSerial_ReadData.vi

This VI is designed to read control and measurement data from any serial source. At the moment one serial source is implemented. It is the RS232 interface of the Dewerack. All actual values of the temperature sensors (seven K_modules, two infrared sensors) are acquired with this VI and are passed on to the VI TSRSr_TemporayStorage_RackSerialread.

32.1 Connector Pane

See figure 97 on page 81.

32.2 Front Panel

See figure 98 on page 81.

32.3 Controls and Indicators

hardware_settings

- **general**
  - **standard_path** path where the program is located
  - **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
  - **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**

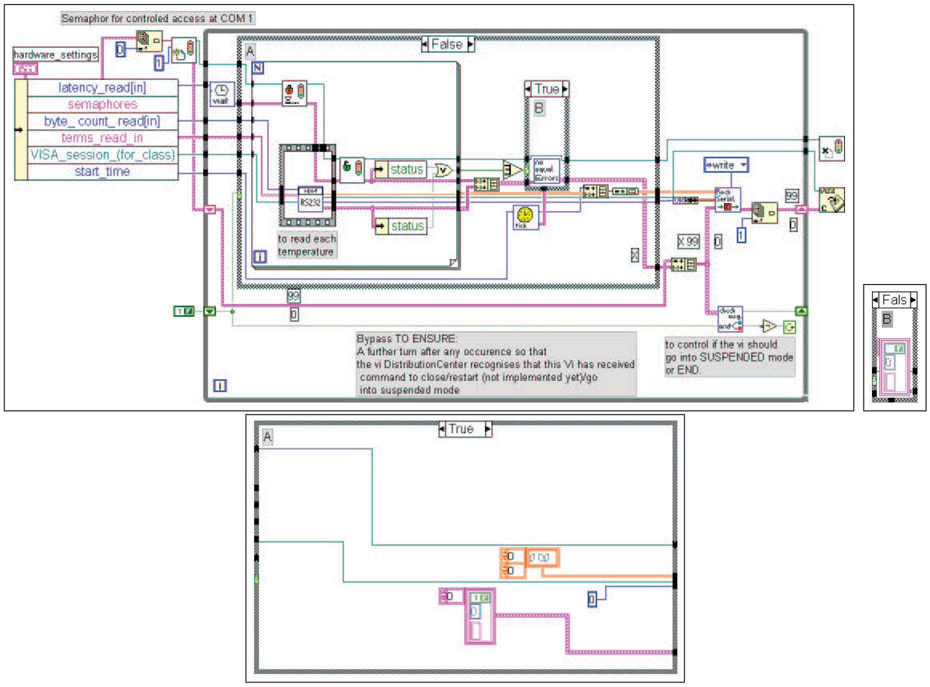


Figure 99: Block Diagram of RSRD_RackSerial_ReadData.vi

- **semaphores** is an array of names of all existing semaphores
  - **name**

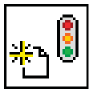
Dewerrack/serial

- **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
- **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
- **timeout_value_read[in]** (not in use at the moment)
- **timeout_value_write[out]** (not in use at the moment)
- **byte_ count_ read[in]** is the number of bytes to be read
- **byte_count_write[out]** is the number of bytes to be written
- **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
- **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **VISA_session_(for_class)** is a unique logical identifier used to communicate with a resource.

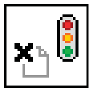
32.4 Block Diagram

See figure 99 on page 82.

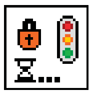
32.5 List of SubVIs



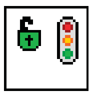
Create Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Create Semaphore.vi



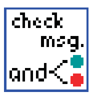
Destroy Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Destroy Semaphore.vi



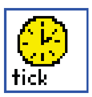
Acquire Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Acquire Semaphore.vi



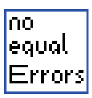
Release Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Release Semaphore.vi



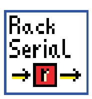
GLCM_GLobal_CheckMessages.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCM_GLobal_CheckMessages.vi



GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCL_GLobal_CLock.vi

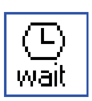


GLAD_GLobal_AddDiffrent_errors.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLAD_GLobal_AddDiffrent_errors.vi



TSRSr_TemporaryStorage_Rack_Serialread.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Temporary_Storages.llb\-TSRSr_TemporaryStorage_Rack_Serialread.vi

RSRI_RackSerial_Read_dIrect.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\RackSerial.llb\RSRI_RackSerial_Read_dIrect.vi



GLWA_GLobalWAit.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLWA_GLobalWAit.vi

32.6 History

RSRD_RackSerial_ReadData.vi History Current Revision: 144

33 RSRI_RackSerial_Read_dIrect.vi

This Vi returns one value which refers to the address at the called pad.

FURTHER DETAIL
Deweracк and the communication need at an average 14 [ms] to generate an answer for a request. As VISAread.vi doesn't release cpu resources while waiting for the answer, the subVI GLWA_GLobal_WAit.vi is put after VISAwrite.vi and before VISAwrite.vi. The subVI waits

Figure 100: Connector Pane of RSRI_RackSerial_Read_dIrect.vi

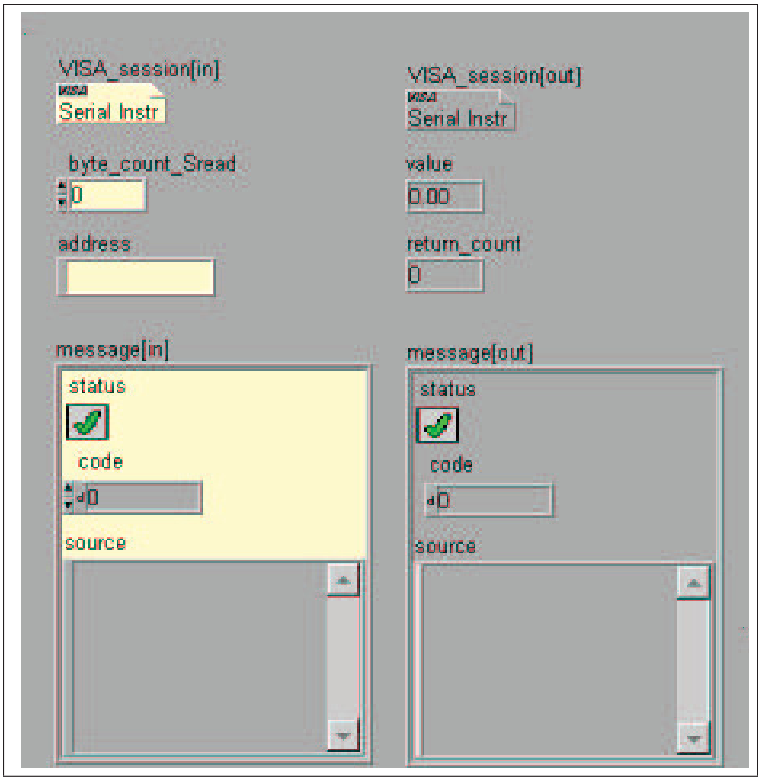


Figure 101: Front Panel of RSRI_RackSerial_Read_dIrect.vi

these 14 [ms] and additionally releases cpu resources.
In case the answer arrives before VISAread.vi is executed, data is buffered at the RS232 board.

33.1 Connector Pane

See figure 100 on page 84.

33.2 Front Panel

See figure 101 on page 84.

33.3 Controls and Indicators

address for the read request

byte_count_Sread is the number of bytes to be read.

VISA_session[in] is a unique logical identifier used to communicate with a resource.

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

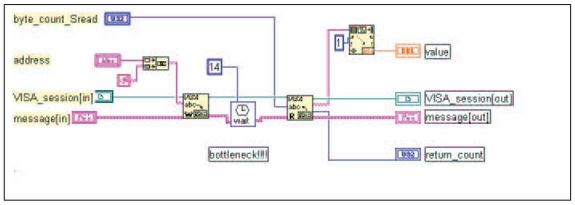


Figure 102: Block Diagram of RSRI_RackSerial_Read_dIrect.vi

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

return_count contains the number of bytes actually read.

value value at the port read. In this case a temperature.

VISA_session[out] is a unique logical identifier used to communicate with a resource.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

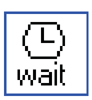
code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

33.4 Block Diagram

See figure 102 on page 85.

33.5 List of SubVIs



GLWA_GLobalWAit.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLWA_GLobalWAit.vi

33.6 History

RSRI_RackSerial_Read_dIrect.vi History Current Revision: 32

34 RSWI_RackSerial_Write_dIrect.vi

This VI is used to write out a bit to a port with the address in *terms_write* .

34.1 Connector Pane

See figure 103 on page 86.

34.2 Front Panel

See figure 104 on page 86.

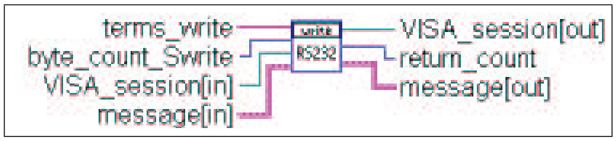


Figure 103: Connector Pane of RSWI_RackSerial_Write_dIrect.vi

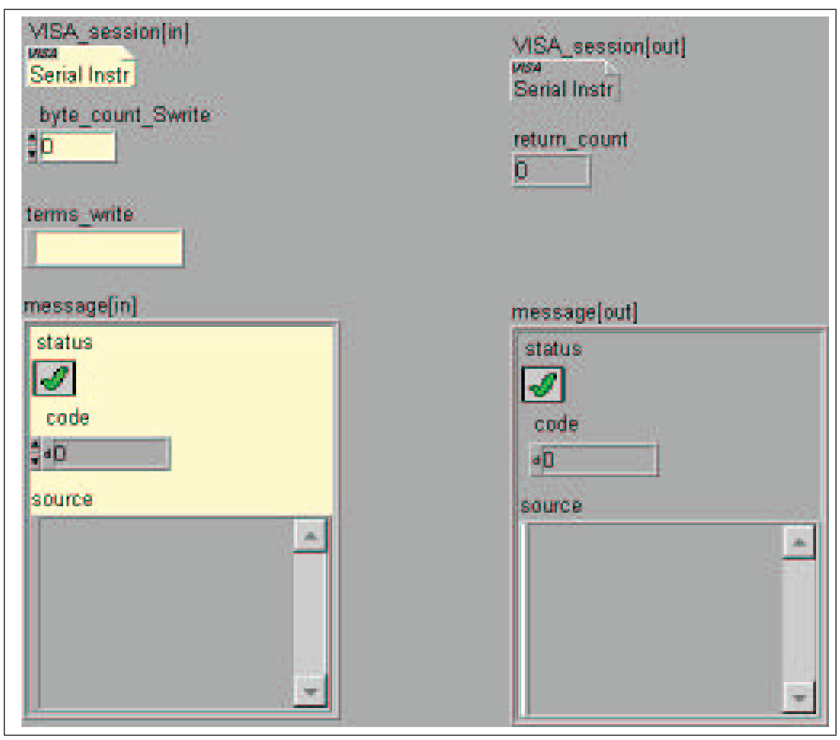


Figure 104: Front Panel of RSWI_RackSerial_Write_dIrect.vi

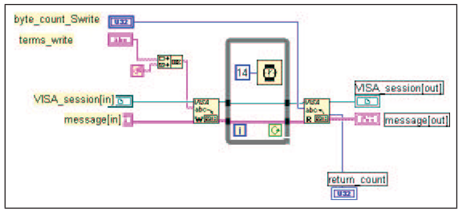


Figure 105: Block Diagram of RSWI_RackSerial_Write_dIrect.vi

34.3 Controls and Indicators

terms_write is a write command including an address for the port and a value (high / low).

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

byte_count_Swrite is the number of bytes to be written.

VISA_session[in] is a unique logical identifier used to communicate with a resource.

return_count passes on the number of bytes read after writing.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

VISA_session[out] is a unique logical identifier used to communicate with a resource.

34.4 Block Diagram

See figure 105 on page 87.

34.5 List of SubVIs

34.6 History

RSWI_RackSerial_Write_dIrect.vi History Current Revision: 19

35 RSWS_RackSerial_WriteS7.vi

This VI sets commands for PLC S 212. The commands are 5 bit coded. The bits are set by addressing a digital output via RS232.The addresses are given in the array *terms_write* the values in the array *power_settings[out]* .

Figure 106: Connector Pane of RSWS_RackSerial_WriteS7.vi

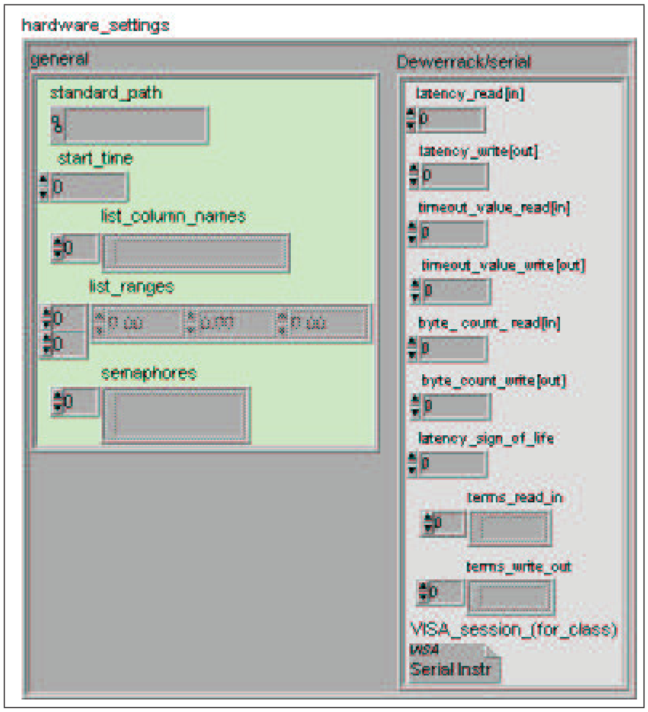


Figure 107: Front Panel of RSWS_RackSerial_WriteS7.vi

35.1 Connector Pane

See figure 106 on page 88.

35.2 Front Panel

See figure 107 on page 88.

35.3 Controls and Indicators

hardware_settings

general

standard_path path where the program is located

start_time [ms] is the time reference. In *LOLAstart_time* is set at INKH_-INitialisation_KonfigHardware.vi at the start of the program.

list_column_names is an array containing the titles of the measurement categories The names' order must match to the commands' order.

name_of_channel channel name is displayed in the user interface, it consists of identifier and unit

list_ranges is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*

value

semaphores is an array of names of all existing semaphores

- **name**

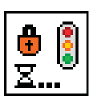
Dewerrack/serial

- **latency_read[in]** is the time in milliseconds for RSRD_RSReadData.vi to wait between two executions of the while loop
- **latency_write[out]** is the time in milliseconds for RSWD_RSWriteData.vi to wait between two executions of the while loop
- **timeout_value_read[in]** (not in use at the moment)
- **timeout_value_write[out]** (not in use at the moment)
- **byte_ count_ read[in]** is the number of bytes to be read
- **byte_count_write[out]** is the number of bytes to be written
- **latency_sign_of_life** is the time period between two alternating signals sent to the PLC S7 200
- **terms_read_in** is an array of addresses respectively commands for the read request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **terms_write_out** is an array of addresses respectively commands for the write request at a certain port
  - **term** address respectively command for the read request at a certain port (called term here)
- **VISA_session_(for_class)** is a unique logical identifier used to communicate with a resource.

35.4 Block Diagram

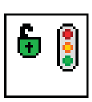
See figure 108 on page 90.

35.5 List of SubVIs

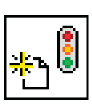


Acquire Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Acquire Semaphore.vi

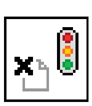
RSWI_RackSerial_Write_dIrect.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\RackSerial.llb\RSWI_RackSerial_Write_dIrect.vi



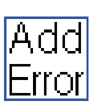
Release Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Release Semaphore.vi



Create Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Create Semaphore.vi



Destroy Semaphore.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\semaphor.llb\Destroy Semaphore.vi



GLAE_ GLobal_AddtwoErrors.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLAE_ GLobal_AddtwoErrors.vi

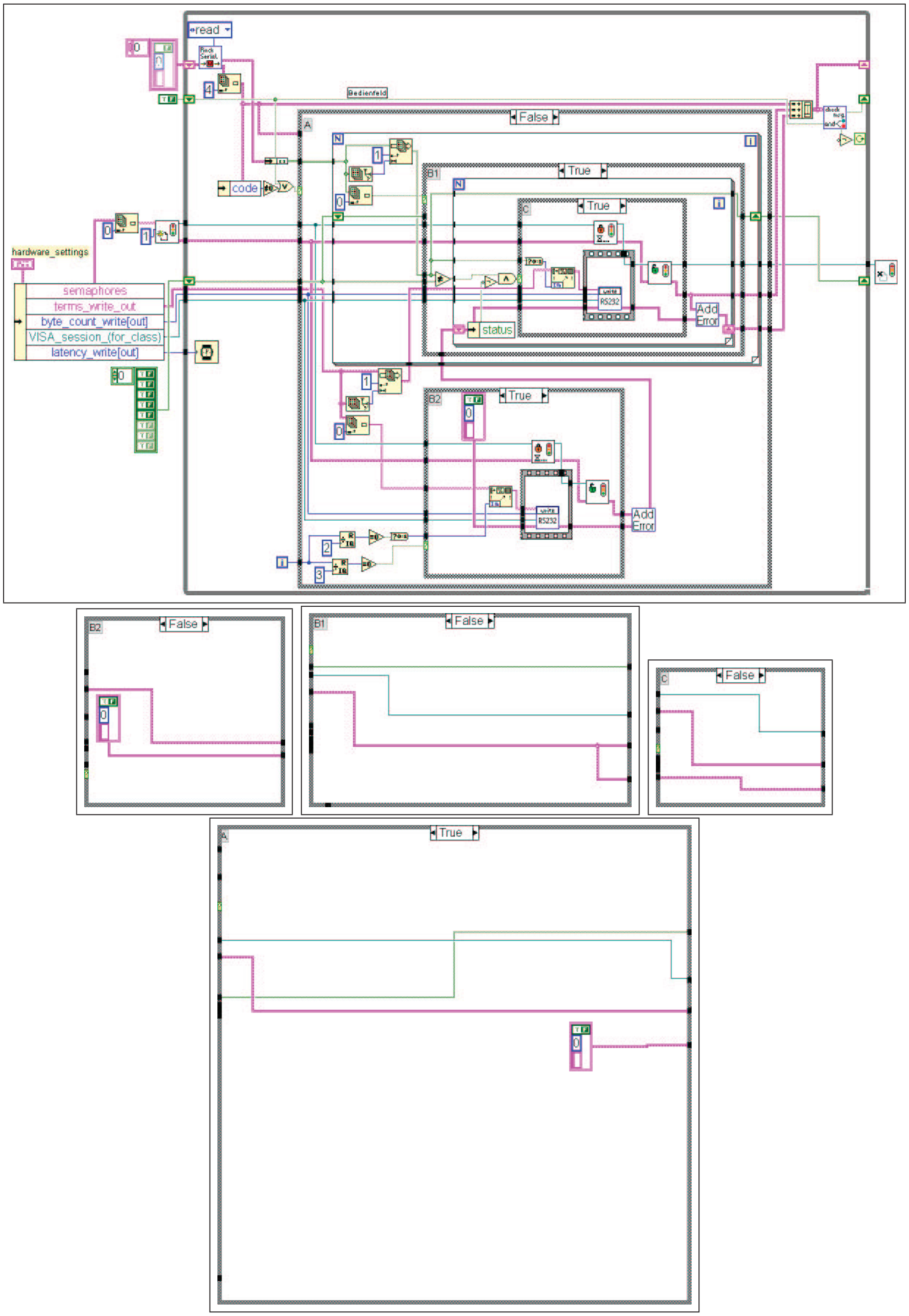


Figure 108: Block Diagram of RSWS_RackSerial_WriteS7.vi

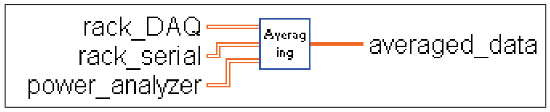


Figure 109: Connector Pane of SAAV_SAve_AVeraging.vi

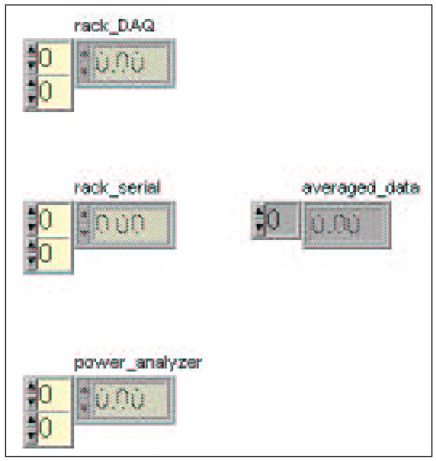
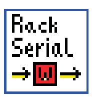
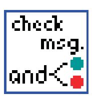


Figure 110: Front Panel of SAAV_SAve_AVeraging.vi



TSRSw_TemporaryStorage_Rack_Serialwrite.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Temporary_Storages.llb\-
TSRSw_TemporaryStorage_Rack_Serialwrite.vi



GLCM_GLobal_CheckMessages.vi
D:\Aktuelle_Arbeit\Version 1.8 in
Arbeit\Global.llb\GLCM_GLobal_CheckMessages.vi

35.6 History

RSWS_RackSerial_WriteS7.vi History Current Revision: 28

36 SAAV_SAve_AVeraging.vi

This VI takes the average of all data taken during *saving_intervall* and aggregates them to one array with the trigger instant as time stamp.

36.1 Connector Pane

See figure 109 on page 91.

36.2 Front Panel

See figure 110 on page 91.

36.3 Controls and Indicators

[DBL] **rack_serial**

[DBL] **Measured Value**

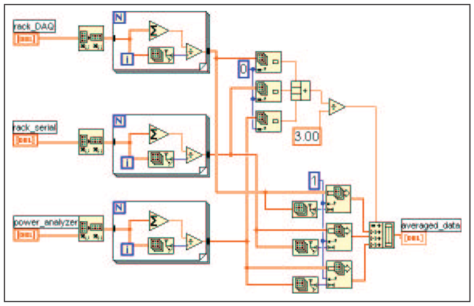


Figure 111: Block Diagram of SAAV_SAve_AVeraging.vi

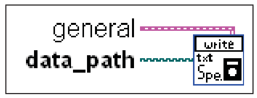


Figure 112: Connector Pane of SACO_SAve_COnversion.vi

power_analyzer

Measured Value

rack_DAQ

Measured Value

averaged_data one- dimensional array representing the average of all acquired data within the given time-range of the saving period.

36.4 Block Diagram

See figure 111 on page 92.

36.5 List of SubVIs

36.6 History

SAAV_SAve_AVeraging.vi History Current Revision: 9

37 SACO_SAve_COnversion.vi

converts all acquired measurement data from datalog format to text format

37.1 Connector Pane

See figure 112 on page 92.

37.2 Front Panel

See figure 113 on page 93.

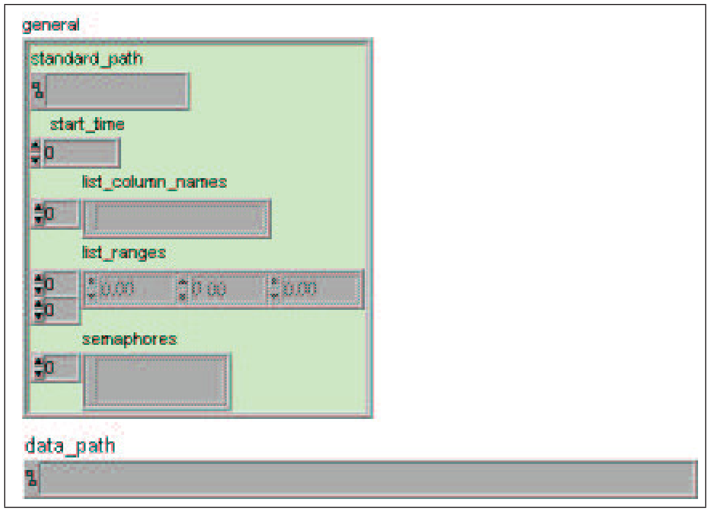


Figure 113: Front Panel of SACO_SAve_COnversion.vi

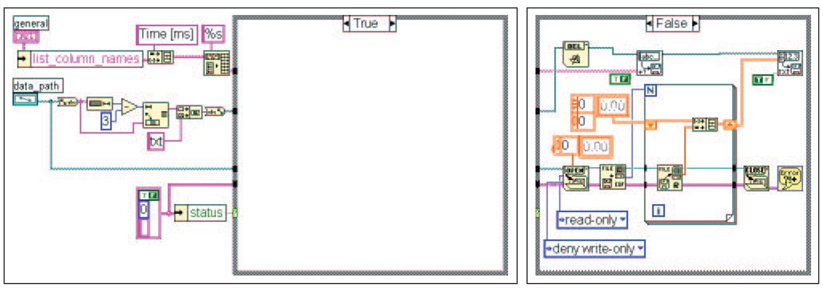


Figure 114: Block Diagram of SACO_SAve_COnversion.vi

37.3 Controls and Indicators

general

- **standard_path** path where the program is located
- **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
- **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
  - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
- **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
  - **value**
- **semaphores** is an array of names of all existing semaphores
  - **name**

data_path path where measurement data is stored

37.4 Block Diagram

See figure 114 on page 93.

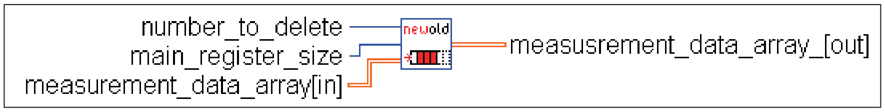


Figure 115: Connector Pane of SADO_SAveDeleteOldest.vi

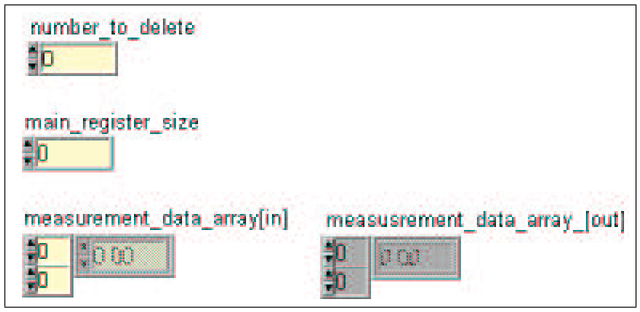
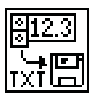
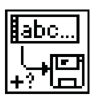


Figure 116: Front Panel of SADO_SAveDeleteOldest.vi

37.5 List of SubVIs



Write To Spreadsheet File.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\file.llb\Write To Spreadsheet File.vi



Write Characters To File.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\file.llb\Write Characters To File.vi

Changed_Error_Handler.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\Changed_Error_Handler.vi

37.6 History

SACO_SAve_COnversion.vi History Current Revision: 46

38 SADO_SAveDeleteOldest.vi

controls which data should be passed on to the shift register. If the time interval of *measurement_data_array[in]* is greater than *time_interval* , an amount of elements equal to *number_to_delete* is deleted in each dimension.

38.1 Connector Pane

See figure 115 on page 94.

38.2 Front Panel

See figure 116 on page 94.

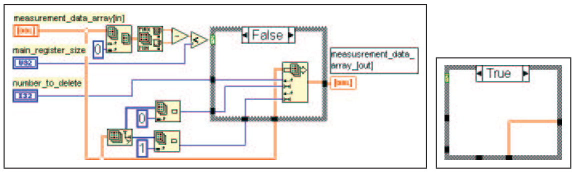


Figure 117: Block Diagram of SADO_SAveDeleteOldest.vi

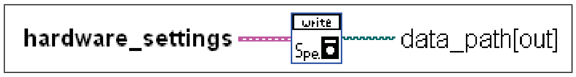


Figure 118: Connector Pane of SASH_SAveSHot.vi

38.3 Controls and Indicators

U32 **main_register_size** determines the size *measurement_data_array[in]* should have.

DBL **measurement_data_array[in]**

  DBL **measured_value**

I32 **number_to_delete** is the number of elements (starting with the first) that should be deleted if the size of *measurement_data_array[in]* is greater than *main_register_size*

DBL **measusrement_data_array_[out]**

  DBL **measured_value**

38.4 Block Diagram

See figure 117 on page 95.

38.5 List of SubVIs

38.6 History

SADO_SAveDeleteOldest.vi History Current Revision: 6

39 SASH_SAveSHot.vi

This VI manages the single shot saving. Whenever the trigger button on the panel of the User-interface is pressed measurement data is saved. The data is averaged and saved to hard-disk.

39.1 Connector Pane

See figure 118 on page 95.

39.2 Front Panel

See figure 119 on page 96.

Figure 119: Front Panel of SASH_SAveSHot.vi

39.3 Controls and Indicators

hardware_settings

- **general**
  - **standard_path** path where the program is located
  - **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_-INitialisation_KonfigHardware.vi at the start of the program.
  - **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
    - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit
  - **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
    - **value**
  - **semaphores** is an array of names of all existing semaphores
    - **name**
- **Saving**
  - **save_interval** determines the amount of Measurement-Data saved
  - **main_register_faktor** determines the amount of Measurement-Data buffered in the main register. The size of the main register is *main_register_factor* multiplied by *save_interval*
  - **datafile_paths** name and location of files where the measured data is saved
    - **path** consists of path (location) and filename

data_path[out] path where measurement data is stored

39.4 Block Diagram

See figure 120 on page 97.

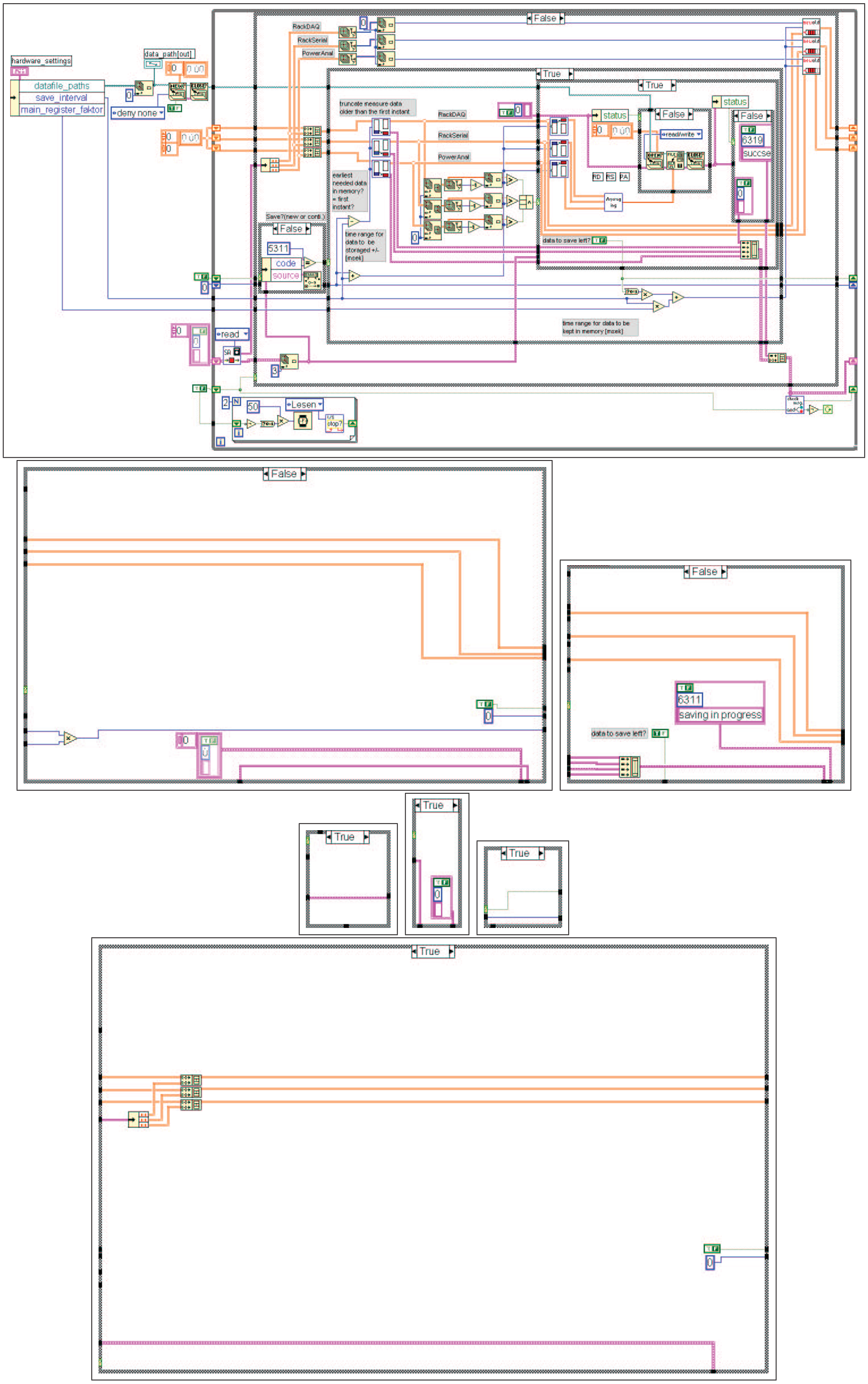


Figure 120: Block Diagram of SASH_SAveSHot.vi

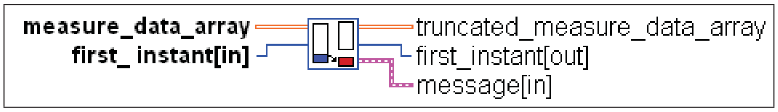
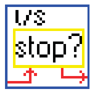
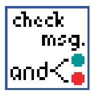


Figure 121: Connector Pane of SATB_SAveTruncateButtom.vi

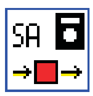
39.5 List of SubVIs



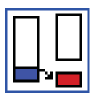
GLSA_Stop_to_All.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLSA_Stop_to_All.vi



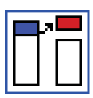
GLCM_GLobal_CheckMessages.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCM_GLobal_CheckMessages.vi



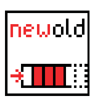
TSSA_TemporaryStorage_SAve.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSSA_TemporaryStorage_SAve.vi



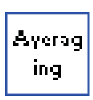
SATB_SAveTruncateButtom.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\SAve.llb\SATB_SAveTruncateButtom.vi



SATT_SAveTruncateTop.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\SAve.llb\SATT_SAveTruncateTop.vi



SADO_SAveDeleteOldest.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\SAve.llb\SADO_SAveDeleteOldest.vi



SAAV_SAve_AVeraging.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\SAve.llb\SAAV_SAve_AVeraging.vi

39.6 History

SASH_SAveSHot.vi History Current Revision: 95

40 SATB_SAveTruncateButtom.vi

This VI truncates the first part of the array up to the index of *first_instant*

40.1 Connector Pane

See figure 121 on page 98.

40.2 Front Panel

See figure 122 on page 99.

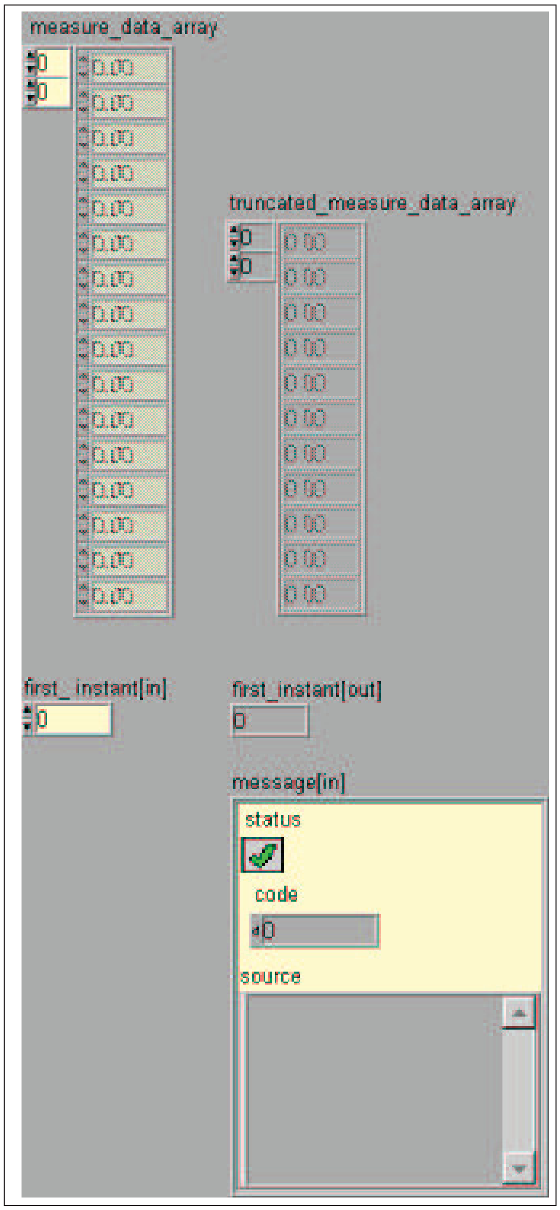


Figure 122: Front Panel of SATB_SAveTruncateButtom.vi

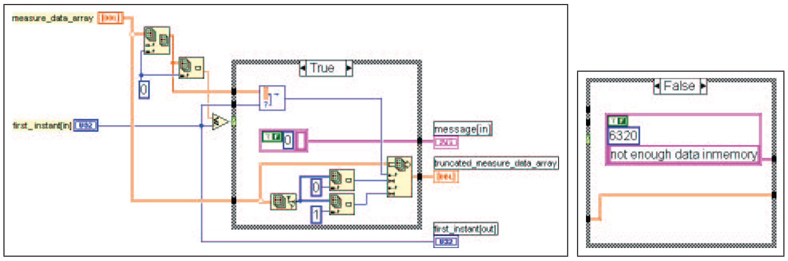


Figure 123: Block Diagram of SATB_SAveTruncateButtom.vi

40.3 Controls and Indicators

first_ instant[in] refers to a certain index at *measurement_data_array* , this index becomes the index 0 in *truncated_measurement_data_array*

measure_data_array measurement data array to be truncated

Measured Value

truncated_measure_data_array equals *measurement_data_array* minus all elements with index smaller than *first_instant*

Measured Value

first_instant[out] refers to a certain index at *measurement_data_array* , this index becomes the index 0 in *truncated_measurement_data_array*

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

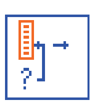
code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

40.4 Block Diagram

See figure 123 on page 100.

40.5 List of SubVIs



GLSN_GLobal_SearchforNumber.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLSN_GLobal_SearchforNumber.vi

40.6 History

SATB_SAveTruncateButtom.vi History Current Revision: 16

41 SATT_SAveTruncateTop.vi

truncates the last part of the array beginning with the index of *last_instant*

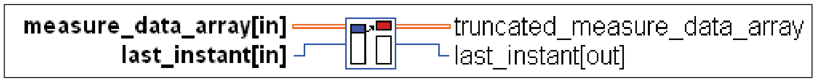


Figure 124: Connector Pane of SATT_SAveTruncateTop.vi

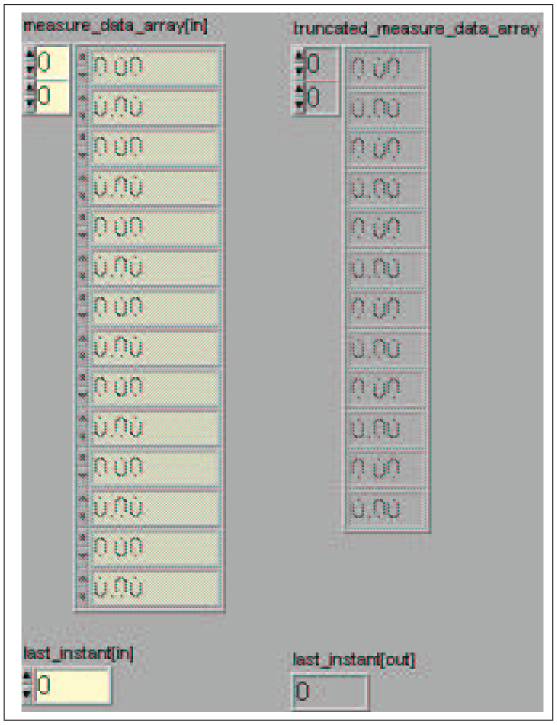


Figure 125: Front Panel of SATT_SAveTruncateTop.vi

41.1 Connector Pane

See figure 124 on page 101.

41.2 Front Panel

See figure 125 on page 101.

41.3 Controls and Indicators

I32 **last_instant[in]** refers to a certain index at *measurement_data_array* , this index becomes the last index in *truncated_measurement_data_array*

DBL **measure_data_array[in]** measurement data array to be truncated

DBL **Measured Value**

DBL **truncated_measure_data_array** equals *measurement_data_array* minus all elements with index greater than *first_instant*

DBL **Measured Value**

I32 **last_instant[out]** refers to a certain index at *measurement_data_array* , this index becomes the last index in *truncated_measurement_data_array*

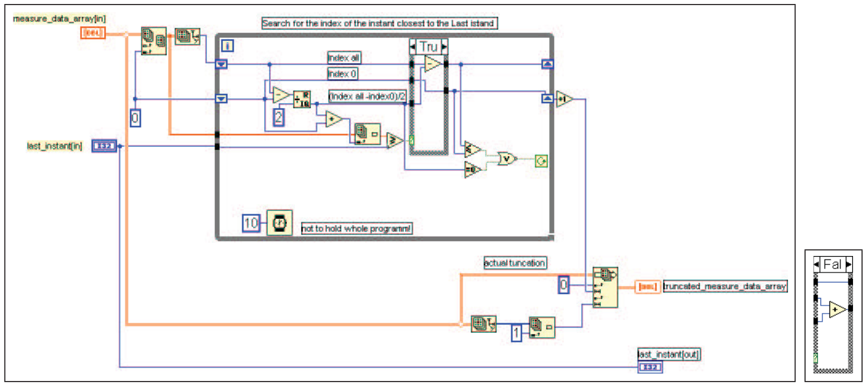


Figure 126: Block Diagram of SATT_SAveTruncateTop.vi

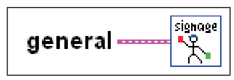


Figure 127: Connector Pane of SICE_SIgnage_CEnter.vi

41.4 Block Diagram

See figure 126 on page 102.

41.5 List of SubVIs

41.6 History

SATT_SAveTruncateTop.vi History Current Revision: 11

42 SICE_SIgnage_CEnter.vi

This VI provides communication and a condition evaluation for all modules.

42.1 Connector Pane

See figure 127 on page 102.

42.2 Front Panel

See figure 128 on page 103.

42.3 Controls and Indicators

general

- **standard_path** path where the program is located
- **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
- **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
  - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit

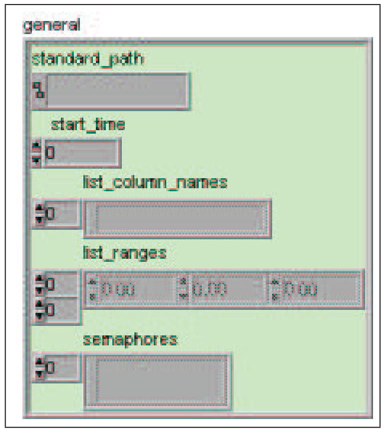


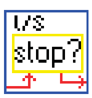
Figure 128: Front Panel of SICE_SIgnage_CEnter.vi

- **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
  - **value**
- **semaphores** is an array of names of all existing semaphores
  - **name**

42.4 Block Diagram

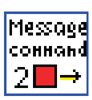
See figure 129 on page 104 and figure 130 on page 105.

42.5 List of SubVIs

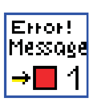


GLSA_Stop_to_All.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLSA_Stop_to_All.vi

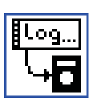
GCEH_Global_Changed_Error_Handler.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GCEH_Global_Changed_Error_Handler.vi



TSM2_TemporaryStorage_Messages2.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSM2_TemporaryStorage_Messages2.vi



TSM1_TemporaryStorage_Messages1.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\-Temporary_Storages.llb\TSM1_TemporaryStorage_Messages1.vi



SILO_Signage_LOg.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Signage.llb\SILO_Signage_LOg.vi

42.6 History

SICE_SIgnage_CEnter.vi History Current Revision: 86

43 SILO_Signage_LOg.vi

This VI saves *log_array* with date and time to hard-disk.

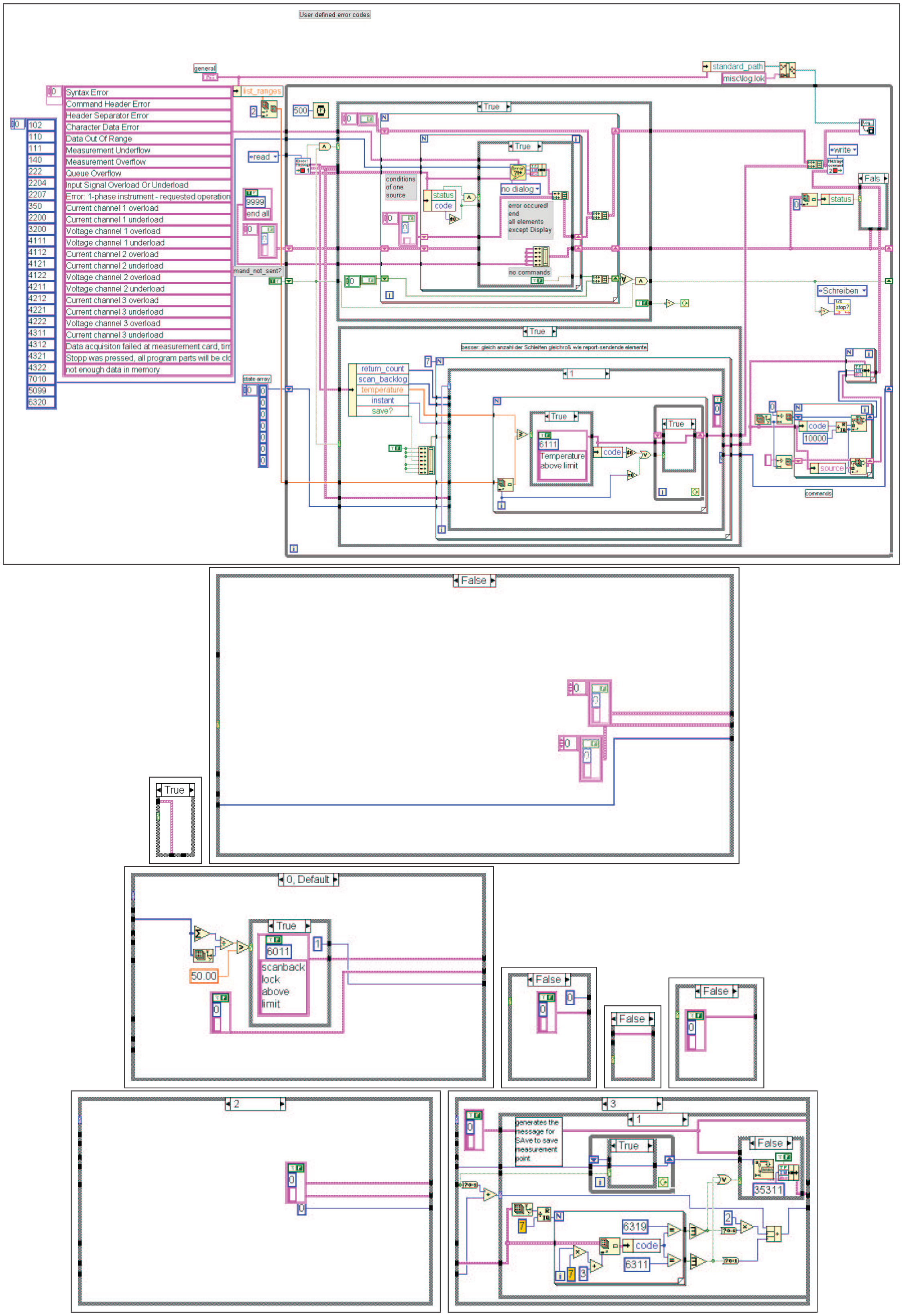


Figure 129: Block Diagram of SICE_SIgnage_CEnter.vi

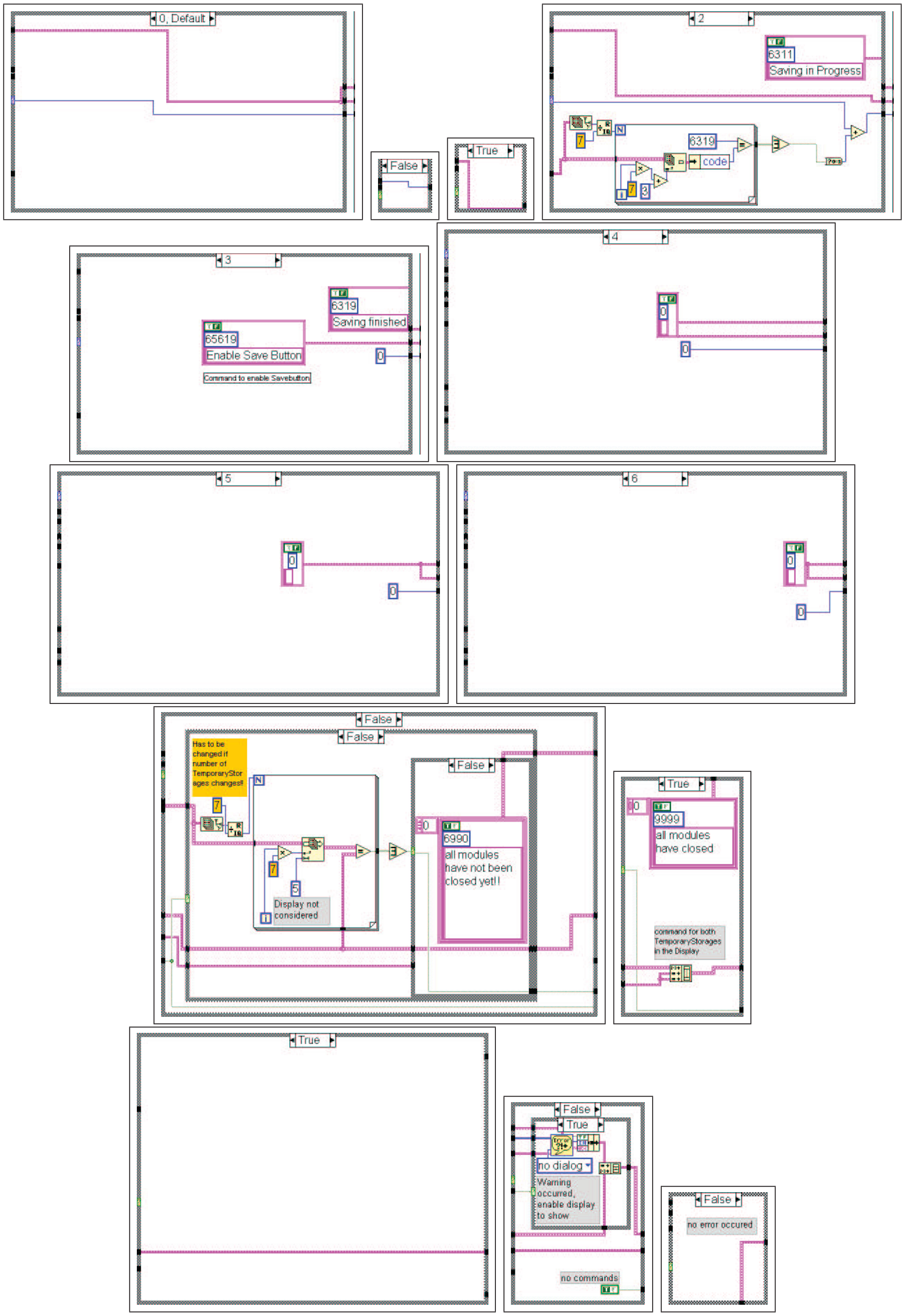


Figure 130: Block Diagram of SICE_SIgnage_CEnter.vi

Figure 131: Connector Pane of SILO_Signage_LOg.vi

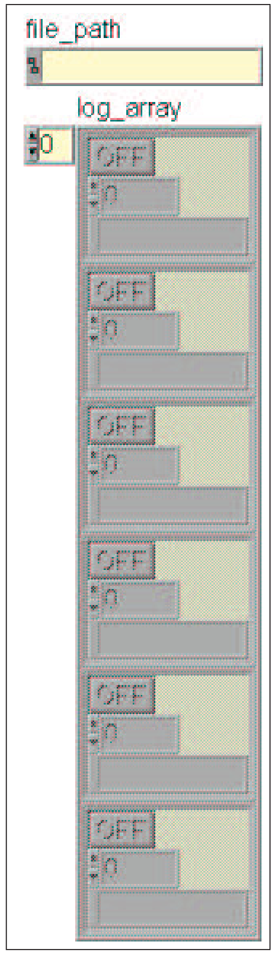


Figure 132: Front Panel of SILO_Signage_LOg.vi

43.1 Connector Pane

See figure 131 on page 106.

43.2 Front Panel

See figure 132 on page 106.

43.3 Controls and Indicators

log_array are messages consisting of concentrated and reformatted conditions

- **error out**
  - TF
  - I32
  - abc

file_path where *log_array* is saved

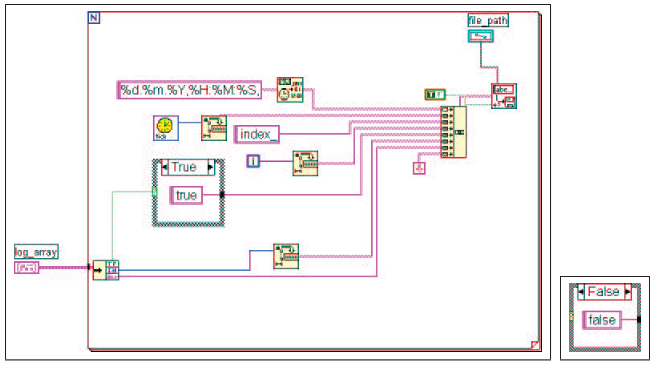
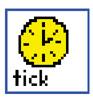


Figure 133: Block Diagram of SILO_Signage_LOg.vi

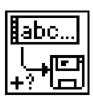
43.4 Block Diagram

See figure 133 on page 107.

43.5 List of SubVIs



GLCL_GLobal_CLock.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\Global.llb\GLCL_GLobal_CLock.vi



Write Characters To File.vi
C:\Programme\National Instruments\Labview\vi.lib\Utility\file.llb\Write Characters To File.vi

43.6 History

SILO_Signage_LOg.vi History Current Revision: 13

44 TSM1_TemporaryStorage_Messages1.vi

This VI is used as temporary storage of messages and reports. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *reports[out]* is read and the referring storage emptied
2. *confirmation_array[out]* is read and the referring storage is emptied
3. *condition_array[out]* is read and the referring storage is emptied
4. *new_messages?* := FALSE.

WRITE:
1. *reports[in]* is added to *reports[out]*
2. *confirmation_array[in]* is added to *confirmation_array[out]*
3. if last *reports[in].useroperation.save* = FALSE then *reports[in].useroperation.save* is added to *reports[out*]
4. *condition_array[in]* is added to *condition_array[out]*
5. if *new_messages?[out]* = FALSE *new_messages?[out]* is replaced by *new_message?[in]*

44.1 Connector Pane

See figure 134 on page 108.

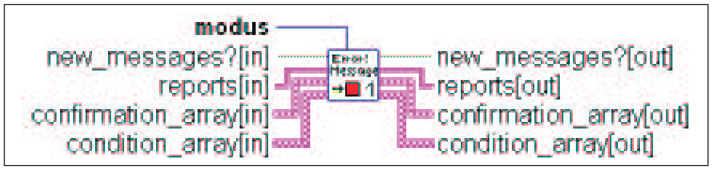


Figure 134: Connector Pane of TSM1_TemporaryStorage_Messages1.vi

44.2 Front Panel

See figure 135 on page 109.

44.3 Controls and Indicators

modus determines the access mode of the caller.

new_messages?[in] if new messages are available or not

condition_array[in] is an array of *condition[in]* containing information about the condition of each module

- **condition[in]** is a cluster that describes the condition. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

confirmation_array[in] is an array of *confirmation[in]*

- **confirmation[in]** is a cluster that describes the confirmation. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

reports[in] information necessary to control the test stand

- **return_count**
  - **Return count** passes on the number of bytes actually read in RSRI_Rack-Serial_Read_dIrect.vi.
- **scan_backlog**
  - scan backlog is the amount of data acquired minus the amount of data read in RDRD_RacDaq_ReadData.vi
- **temperature**
  - **Measured Value**
- **useroperation** contains the condition of the trigger button on the panel of userinterface and the instant when it was pressed
  - **save?**
  - **instant**

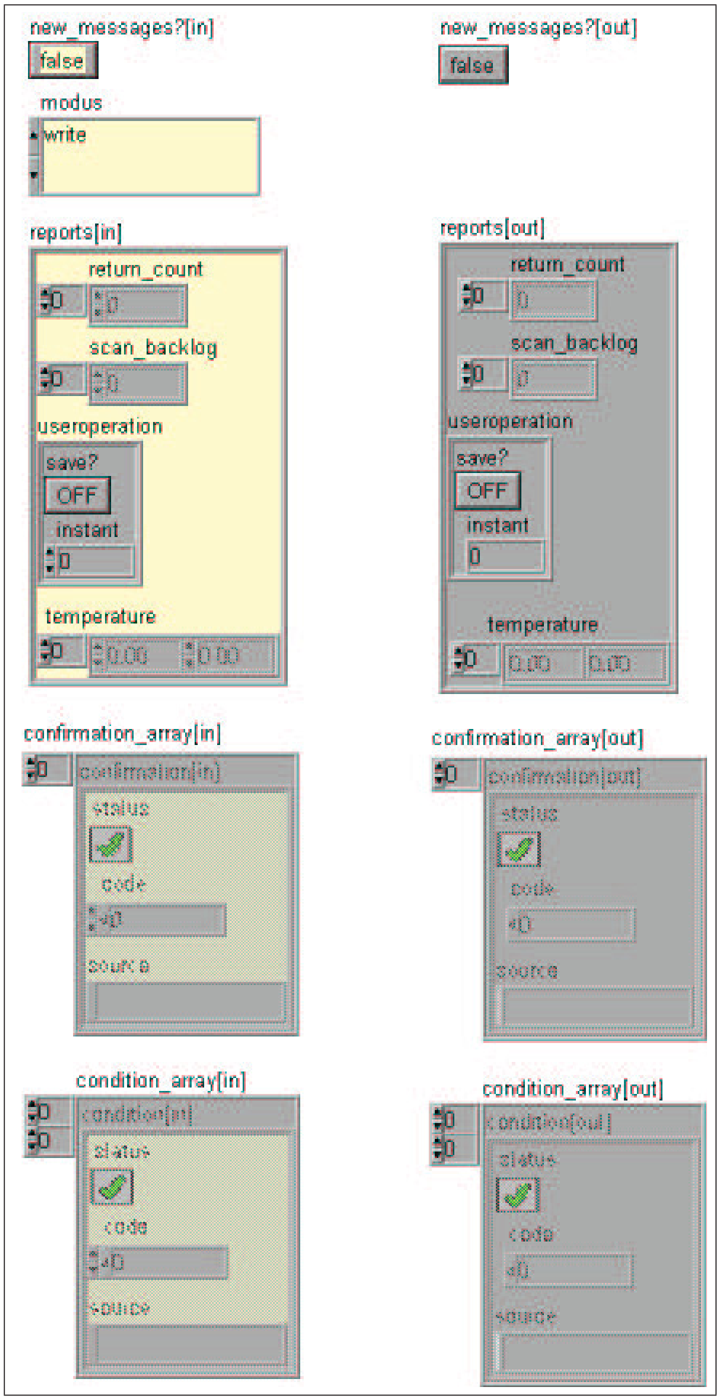


Figure 135: Front Panel of TSM1_TemporaryStorage_Messages1.vi

condition_array[out] is an array of *condition[out]* containing information about the condition of each module

condition[out] is a cluster that describes the condition. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

confirmation_array[out] is an array of *confirmation[out]*

confirmation[out] is a cluster that describes the confirmation. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message . If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

new_messages?[out] if new messages are available or not

reports[out] information necessary to control the test stand

return_count

Return count passes on the number of bytes actually read in RSRI_Rack-Serial_Read_dIrect.vi.

scan_backlog

scan backlog is the amount of data acquired minus the amount of data read in RDRD_RacDaq_ReadData.vi

temperature

Measured Value

useroperation contains the condition of the trigger button on the panel of userinterface and the instant when it was pressed

save?

instant

44.4 Block Diagram

See figure 136 on page 111.

44.5 List of SubVIs

44.6 History

TSM1_TemporaryStorage_Messages1.vi History Current Revision: 63

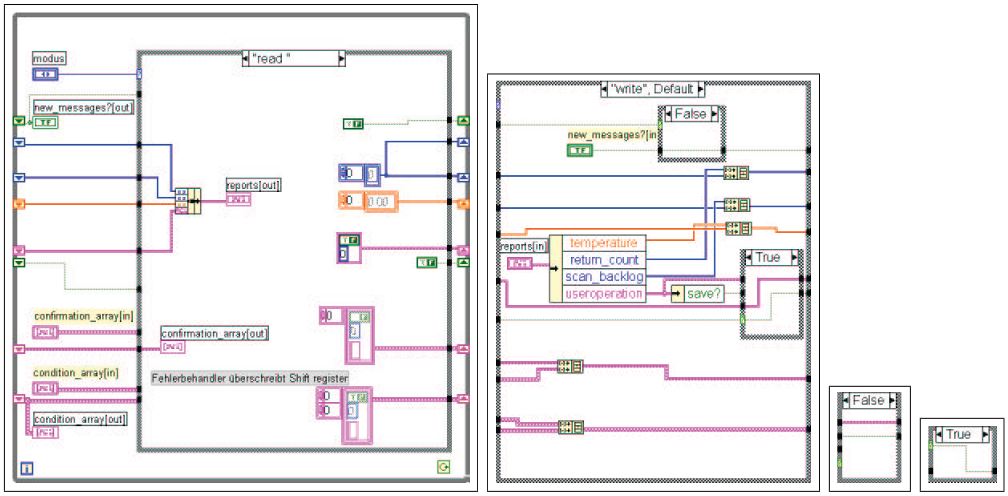


Figure 136: Block Diagram of TSM1_TemporaryStorage_Messages1.vi

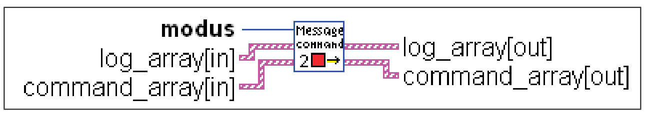


Figure 137: Connector Pane of TSM2_TemporaryStorage_Messages2.vi

45 TSM2_TemporaryStorage_Messages2.vi

This VI is used as temporary storage of messages. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *command_array[out]* is read
2. *log_array[out]* is read

WRITE commands:
1. *command_array[out]* is replaced by *command_array[in]*
2. *log_array[out]* is replaced by *log_array[in]*

45.1 Connector Pane

See figure 137 on page 111.

45.2 Front Panel

See figure 138 on page 112.

45.3 Controls and Indicators

modus determines the access mode of the caller.

log_array[in] are messages consisting of concentrated and reformatted conditions

- **error out**
  - **status**
  - **code**

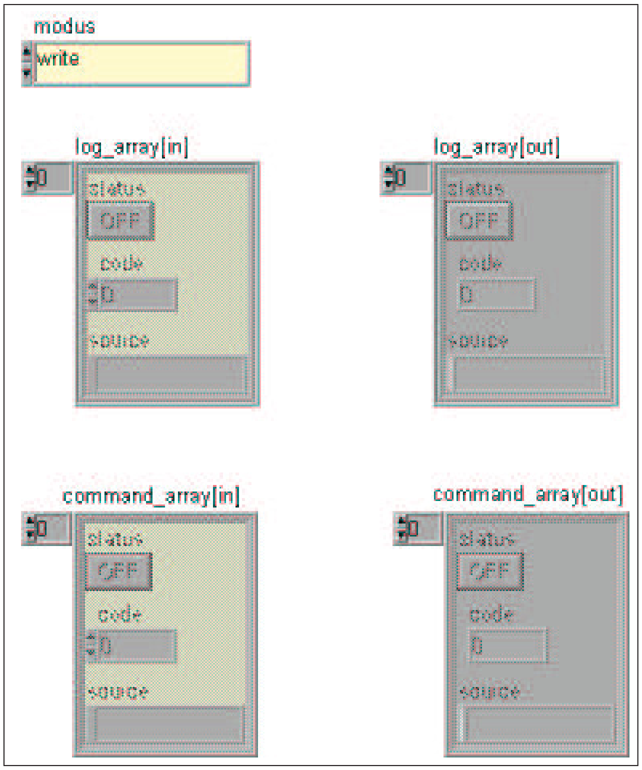


Figure 138: Front Panel of TSM2_TemporaryStorage_Messages2.vi

source

command_array[in] are messages to control the behavior of modules

error out

status

code

source

log_array[out] are messages consisting of concentrated and reformatted conditions

error out

status

code

source

command_array[out] are messages to control the behavior of modules

error out

status

code

source

45.4 Block Diagram

See figure 139 on page 113.

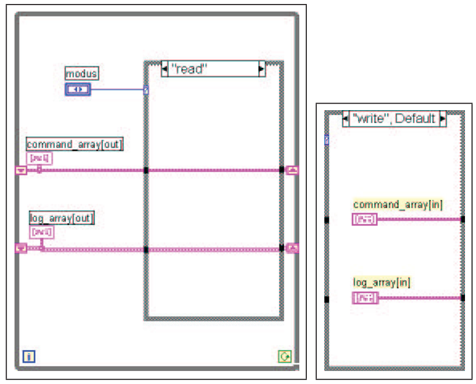


Figure 139: Block Diagram of TSM2_TemporaryStorage_Messages2.vi

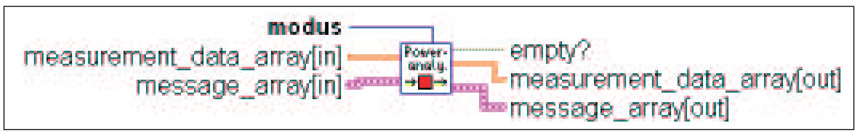


Figure 140: Connector Pane of TSPA_TemporaryStorage_PowerAnalyzer.vi

45.5 List of SubVIs

45.6 History

TSM2_TemporaryStorage_Messages2.vi History Current Revision: 52

46 TSPA_TemporaryStorage_PowerAnalyzer.vi

This VI is used as temporary storage of data and messages. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *measurement_data[out]* , *report_array[out]* is read and the referring storage emptied
2. *message_array[out]* is read and afterwards replaced by *message_array[in]*
3. *empty* := TRUE.

WRITE:
1. *measurement_data_array[in]* is added to *measurment_data_array[out]*
2. if *empty?* = FALSE and *message_array[out]* contains no error *message_array[out]* is replaced by *message_array[in]* .
4. *empty?* := FALSE

46.1 Connector Pane

See figure 140 on page 113.

46.2 Front Panel

See figure 141 on page 114.

Figure 141: Front Panel of TSPA_TemporaryStorage_PowerAnalyzer.vi

46.3 Controls and Indicators

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

measurement_data_array[in] passes on measured data from the writing VI to the storage.

Measured Value represents in this case a temperature value.

modus determines the access mode of the caller.

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

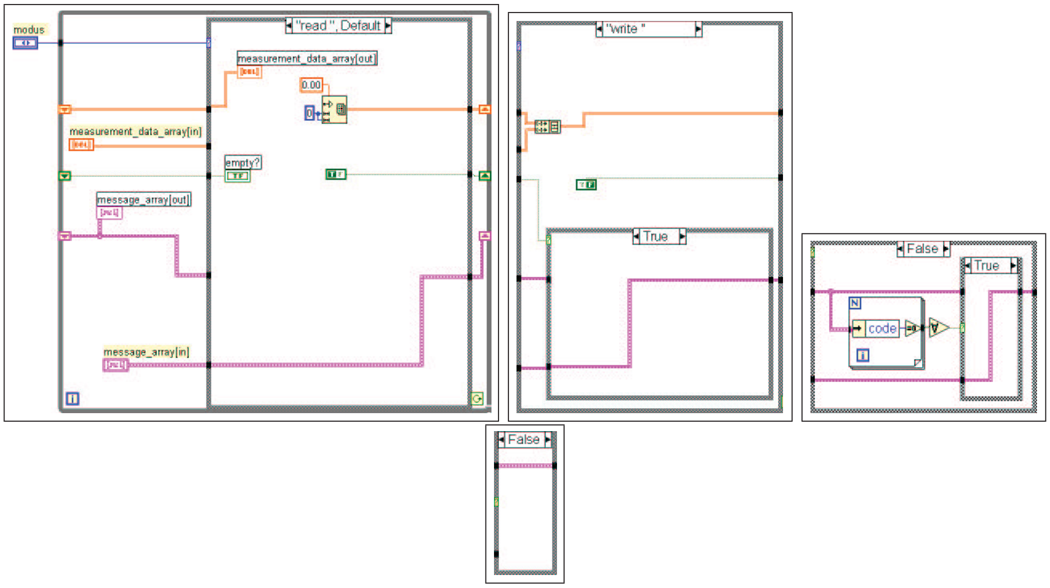


Figure 142: Block Diagram of TSPA_TemporaryStorage_PowerAnalyzer.vi

- **status** is TRUE if an error occurred, or FALSE if not.
- **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
- **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

measurement_data_array[out] The stored measurement data is passed on to the reading VI.

- **Measured Value** represents in this case a temperature value.

empty? indicates if the temporary storage contains any data.

46.4 Block Diagram

See figure 142 on page 115.

46.5 List of SubVIs

46.6 History

TSPA_TemporaryStorage_PowerAnalyzer.vi History Current Revision: 45

47 TSRD_TemporaryStorage_Rack_Daq.vi

This VI is used as temporary storage of data, messages and reports. There are two ways to access this VI, mode: READ and mode WRITE.

READ:
1. *measurement_data_array[out]* , *report_array[out]* is read and the referring storage emptied
2. *message_array[out]* is read and afterwards replaced by *message_array[in]*
3. *empty* := TRUE

WRITE:

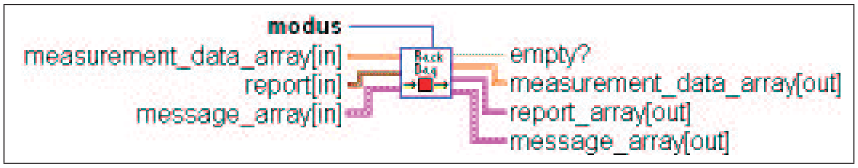


Figure 143: Connector Pane of TSRD_TemporaryStorage_Rack_Daq.vi

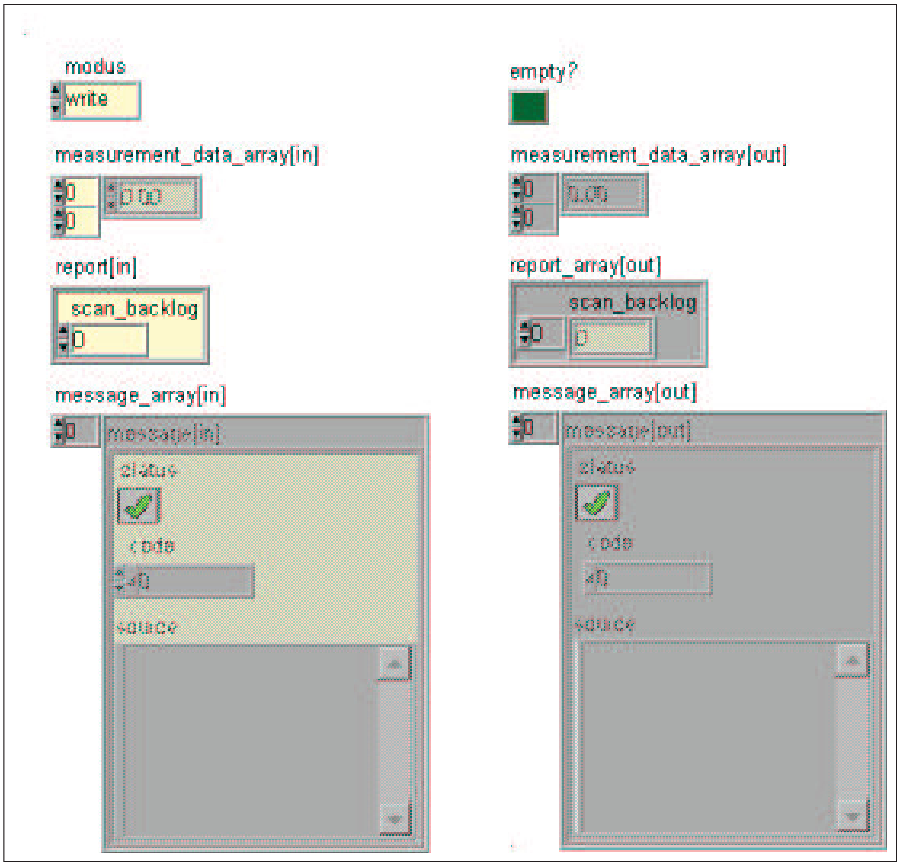


Figure 144: Front Panel of TSRD_TemporaryStorage_Rack_Daq.vi

1. *measurement_data_array[in]* is added to *measurment_data_array[out]*
2. *report[in]* is added to *report_array[out]*
3. if *empty?* = FALSE and *message_array[out]* contains no error then *message_array[out]* is replaced by *message_array[in]*
4. *empty?* := FALSE

47.1 Connector Pane

See figure 143 on page 116.

47.2 Front Panel

See figure 144 on page 116.

47.3 Controls and Indicators

report[in] passes on *scan_backlog* from the writing vi to the storage.

- **scan_backlog** scan backlog is the amount of data acquired minus the amount of data read.

modus determines the access mode of the caller.

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

measurement_data_array[in] passes on measured data from the writing vi to the storage.

- **Measured Value** represents in this case a temperature value.

measurement_data_array[out] The stored measurement data is passed on to the reading vi.

- **Measured Value**

report_array[out] passes on an array on to reading VI. This array contains all occurred scan- backlogs of RDRD_RackDAQ_Read_Data.vi.

- **scan_backlog** scan backlog is the amount of data acquired minus the amount of data read.
  - **scan_backlog** scan backlog is the amount of data acquired minus the amount of data read.

empty? indicates if the temporary storage contains any data.

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[out]** is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

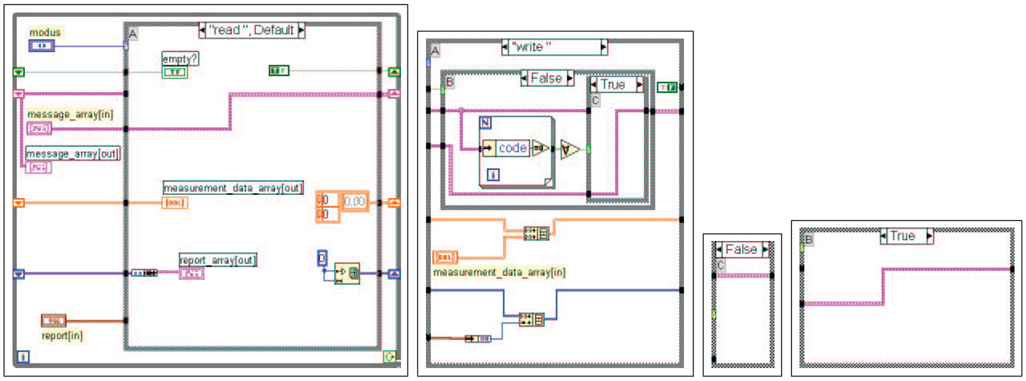


Figure 145: Block Diagram of TSRD_TemporaryStorage_Rack_Daq.vi

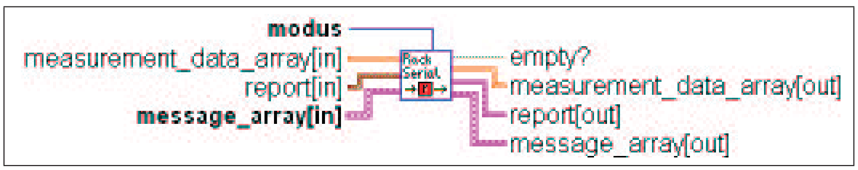


Figure 146: Connector Pane of TSRSr_TemporaryStorage_Rack_Serialread.vi

47.4 Block Diagram

See figure 145 on page 118.

47.5 List of SubVIs

47.6 History

TSRD_TemporaryStorage_Rack_Daq.vi History Current Revision: 62

48 TSRSr_TemporaryStorage_Rack_Serialread.vi

This VI is used as temporary storage of data, messages and reports. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *measurement_data[out]* , *report_array[out]* is read and the referring storage emptied
2. *message_array[out]* is read and afterwards replaced by *message_array[in]*
3. *empty* := TRUE.

WRITE:
1. *measurement_data_array[in]* is added to *measurment_data_array[out]*
2. *report[in]_return_count* is added to *report_array[out]_return_count*
3. if *empty?* = FALSE and *message_array[out]* contains no error then *message_array[out]* is replaced by *message_array[in]*
4. *empty?* := FALSE

48.1 Connector Pane

See figure 146 on page 118.

Figure 147: Front Panel of TSRSr_TemporaryStorage_Rack_Serialread.vi

48.2 Front Panel

See figure 147 on page 119.

48.3 Controls and Indicators

measurement_data_array[in] passes on measured data from the writing VI to the storage.

Measured Value represents in this case a temperature value.

modus determines the access mode of the caller.

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[in] is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

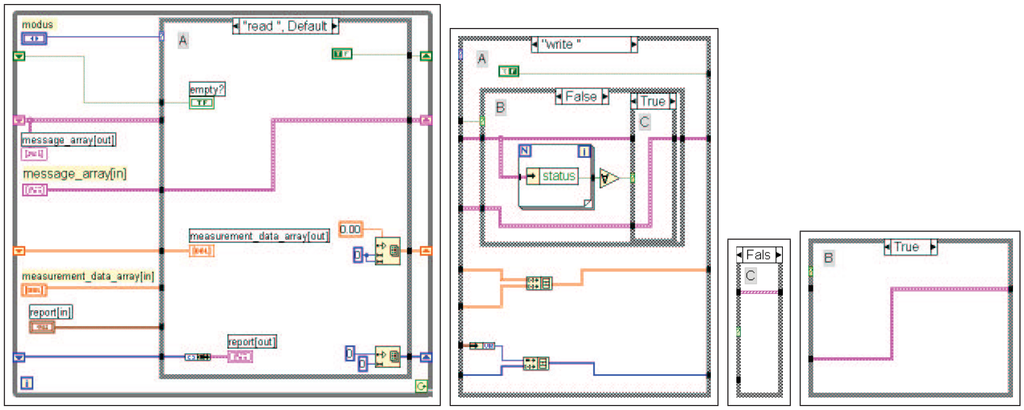


Figure 148: Block Diagram of TSRSr_TemporaryStorage_Rack_Serialread.vi

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

report[in] passes on the number of bytes actually read in RSRI_RackSerial_Read_dIrect.vi.

return_count passes on the number of bytes actually read in RSRI_RackSerial_-Read_dIrect.vi.

measurement_data_array[out] The stored measurement data is passed on to the reading VI.

Measured Value represents in this case a temperature value.

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

empty? indicates if the temporary storage contains any data.

report[out] passes on the number of bytes actually read in RSRI_RackSerial_Read_-dIrect.vi.

return_count passes on an array of numbers referring to the amount of bytes read in RSRI_RackSerial_Read_dIrect.vi.

return_count

48.4 Block Diagram

See figure 148 on page 120.

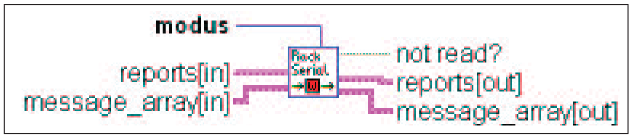


Figure 149: Connector Pane of TSRSw_TemporaryStorage_Rack_Serialwrite.vi

48.5 List of SubVIs

48.6 History

TSRSr_TemporaryStorage_Rack_Serialread.vi History Current Revision: 64

49 TSRSw_TemporaryStorage_Rack_Serialwrite.vi

This VI is used as temporary storage of data and messages. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *reports[out].power_settings* is read and the referring storage emptied
2. only if *not_read?* = FALSE and *message_array[out]* contains no error, *message_array[out]* is replaced by *message_array[in]*
3. *not_read?* :=FALSE.

WRITE:
1. *reports[in].power_settings* is added to *reports[out].power_settings*
2. *message_array[out]* is read and afterwards replaced by *message_array[in]*
3. *not_read?* := TRUE.

49.1 Connector Pane

See figure 149 on page 121.

49.2 Front Panel

See figure 150 on page 122.

49.3 Controls and Indicators

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

modus determines the access mode of the caller.

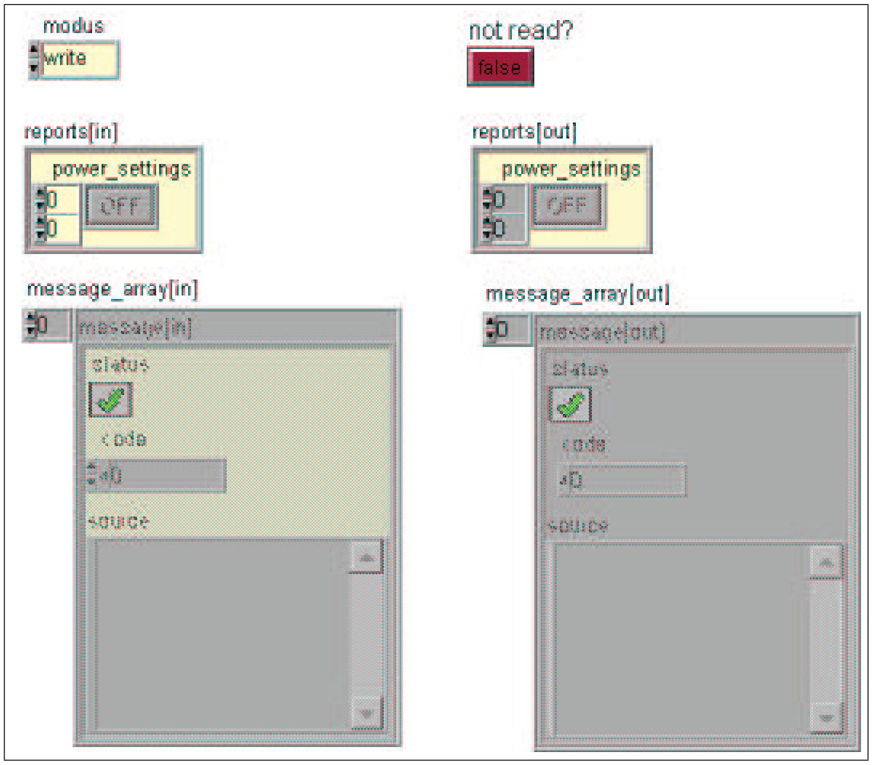


Figure 150: Front Panel of TSRSw_TemporaryStorage_Rack_Serialwrite.vi

reports[in] information necessary to control the test stand

- **power_settings** is an array of boolean representing different supply and charging states for the line and the two batteries.
  - **Charge Battery 2**

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[out]** is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

not read? indicates if the temporary storage contains any data.

reports[out] information necessary to control the test stand

- **power_settings** is an array of boolean representing different supply and charging states for the line and the two batteries.
  - **Charge Battery 2**

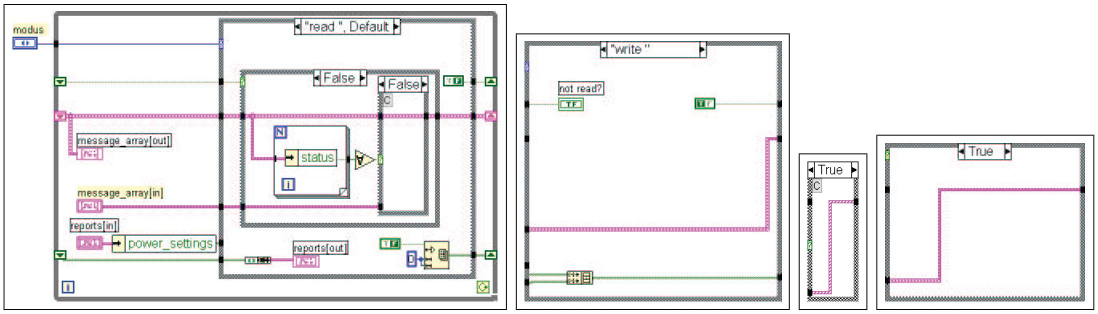


Figure 151: Block Diagram of TSRSw_TemporaryStorage_Rack_Serialwrite.vi

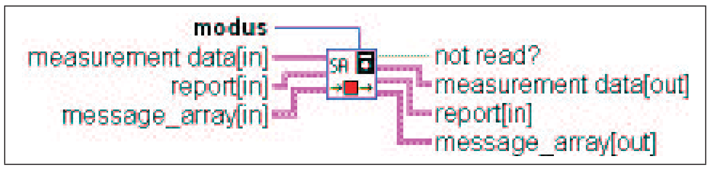


Figure 152: Connector Pane of TSSA_TemporaryStorage_SAve.vi

49.4 Block Diagram

See figure 151 on page 123.

49.5 List of SubVIs

49.6 History

TSRSw_TemporaryStorage_Rack_Serialwrite.vi History Current Revision: 59

50 TSSA_TemporaryStorage_SAve.vi

This VI is used as temporary storage of data, messages and reports. There are two ways to access this VI, mode: READ and mode: WRITE.

READ:
1. *measurement_data[out]* , *report_array[out]* is read and the referring storage emptied
2. only if *not read?* = FALSE and *message_array[out]* contains no error, *message_array[out]* is replaced by *message_array[in]*
3. *not read?* :=FALSE.

WRITE:
1. *measurement_data_array[in]* is added to *measurment_data_array[out]*
2. *report[in]* is added to *report_array[out]*
3. *message_array[out]* is read and afterwards replaced by *message_array[in]*
4. *not read?* is set to TRUE.

50.1 Connector Pane

See figure 152 on page 123.

50.2 Front Panel

See figure 153 on page 124.

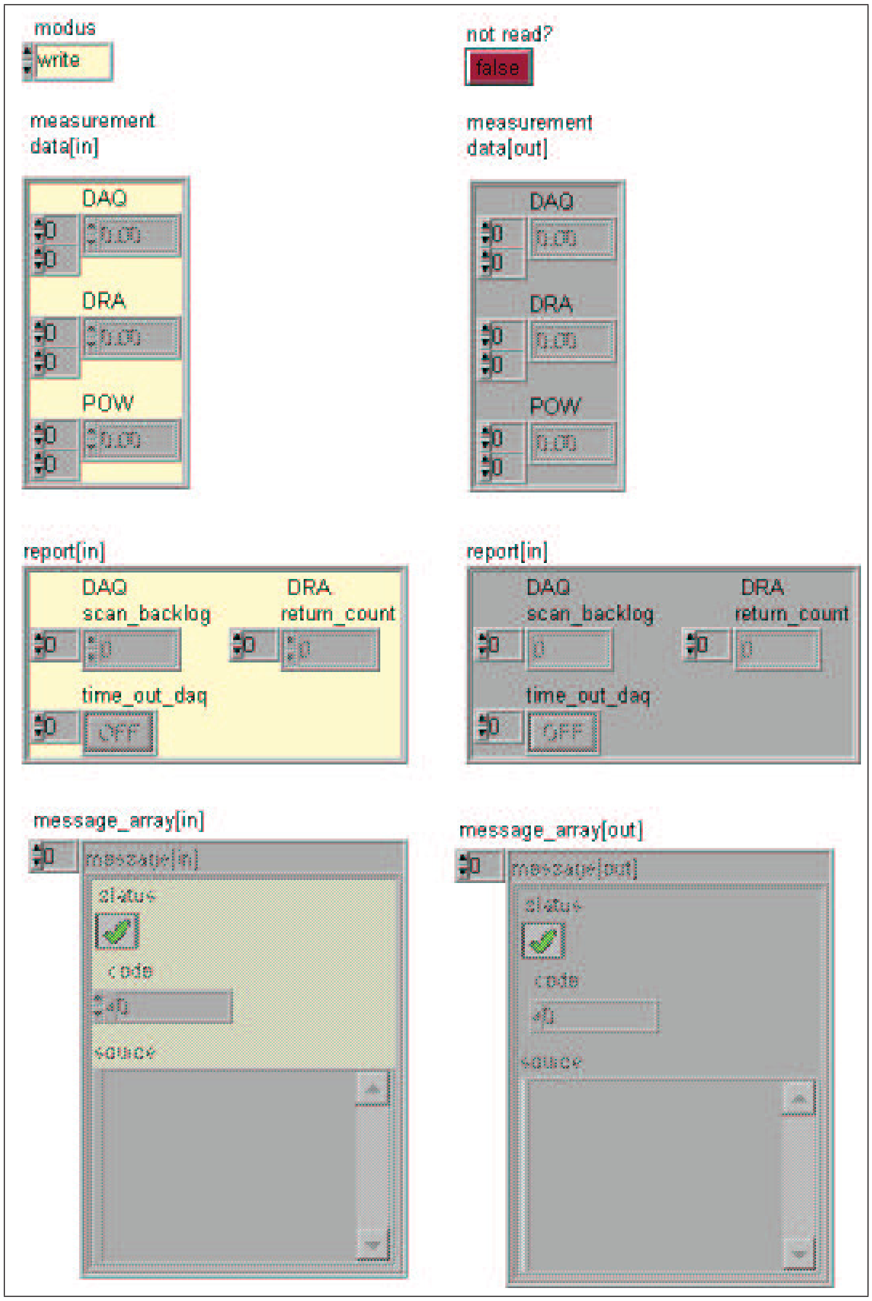


Figure 153: Front Panel of TSSA_TemporaryStorage_SAve.vi

50.3 Controls and Indicators

measurement data[in]

- **DAQ** passes on measurement data acquired at RDRD_RackDAQ_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **DRA** passes on measurement data acquired at RSRD_RackSerial_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **POW** passes on measurement data acquired at PARD_PowerAnalyzer_ReadData.vi from the writing vi to the storage.
  - **Measured Value**

report[in]

- **DAQ scan_backlog** passes on an array on to reading VI. This array contains all occurred scan- backlogs of RDRD_RackDAQ_Read_Data.vi.
- **time_out_daq** contains boolean expressions. The value true implies that a time out condition occurred at RDRD_RackDAQ_ReadData.vi.
- **DRA return_count** passes on an array of numbers referring to the amount of bytes read in RSRI_RackSerial_Read_dIrect.vi.
  - **Return count**

modus determines the access mode of the caller.

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

not read? indicates if the temporary storage contains any data.

measurement data[out]

- **DAQ** passes on measurement data acquired at RDRD_RackDAQ_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **DRA** passes on measurement data acquired at RSRD_RackSerial_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **POW** passes on measurement data acquired at PARD_PowerAnalyzer_ReadData.vi from the writing vi to the storage.

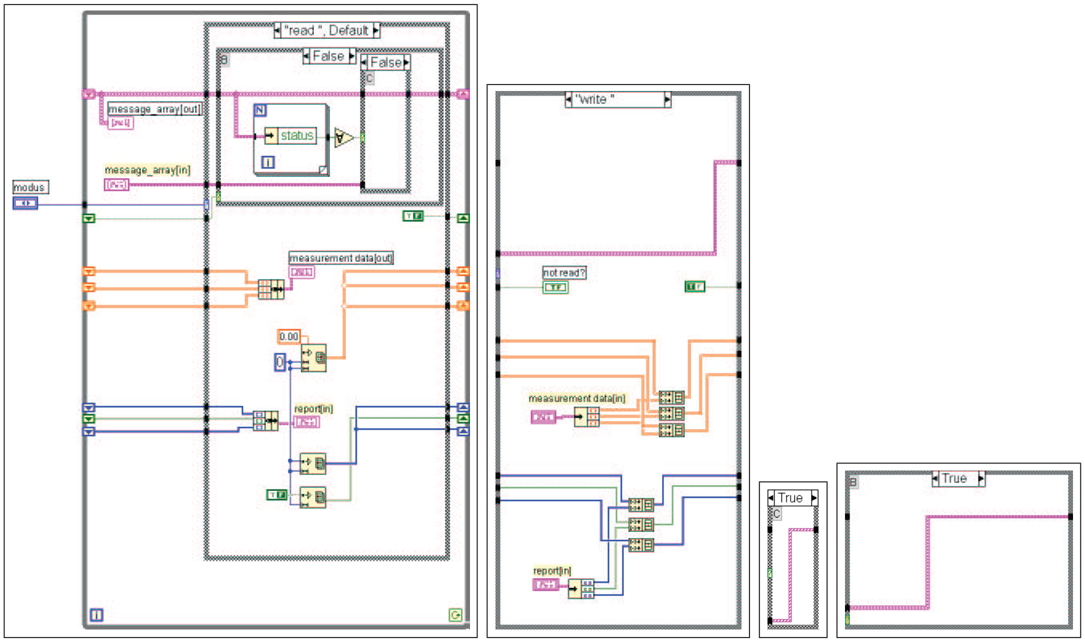


Figure 154: Block Diagram of TSSA_TemporaryStorage_SAve.vi

Measured Value

report[in]

DAQ scan_backlog passes on an array on to reading VI. This array contains all occurred scan- backlogs of RDRD_RackDAQ_Read_Data.vi.

time_out_daq contains boolean expressions. The value true implies that a time out condition occurred at RDRD_RackDAQ_ReadData.vi.

DRA return_count passes on an array of numbers referring to the amount of bytes read in RSRI_RackSerial_Read_dIrect.vi.

Return count

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

50.4 Block Diagram

See figure 154 on page 126.

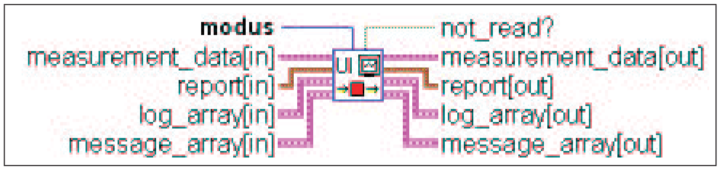


Figure 155: Connector Pane of TSUI_TemporaryStorage_UserInterface.vi

50.5 List of SubVIs

50.6 History

TSSA_TemporaryStorage_SAve.vi History Current Revision: 46

51 TSUI_TemporaryStorage_UserInterface.vi

This VI is used as temporary storage of data, messages and reports. There are two ways to access this VI, mode: READ and mode:WRITE.

READ:
1. *measurement_data[out]* and *log_array[out]* is read and the referring storage emptied
2. if *not_read?* = TRUE then *message_array[out]* is read and replaced by *command_array[in]*
else
3.*not_read* := FALSE

WRITE:
1. *measurement_data[in]* is added to *mesurement_data[out]*
2. *log_array[in]* is added to *log_array[out]*
3. *message_array[out]* is read and replace by *command_array[in]*
4. *not_read?* := TRUE

51.1 Connector Pane

See figure 155 on page 127.

51.2 Front Panel

See figure 156 on page 128.

51.3 Controls and Indicators

measurement_data[in] all measurment_data to visualize at the screen

- **DAQ** passes on measurement data acquired at RDRD_RackDAQ_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **DRA** passes on measurement data acquired at RSRD_RackSerial_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **POW** passes on measurement data acquired at PARD_PowerAnalyzer_ReadData.vi from the writing vi to the storage.
  - **Measured Value**

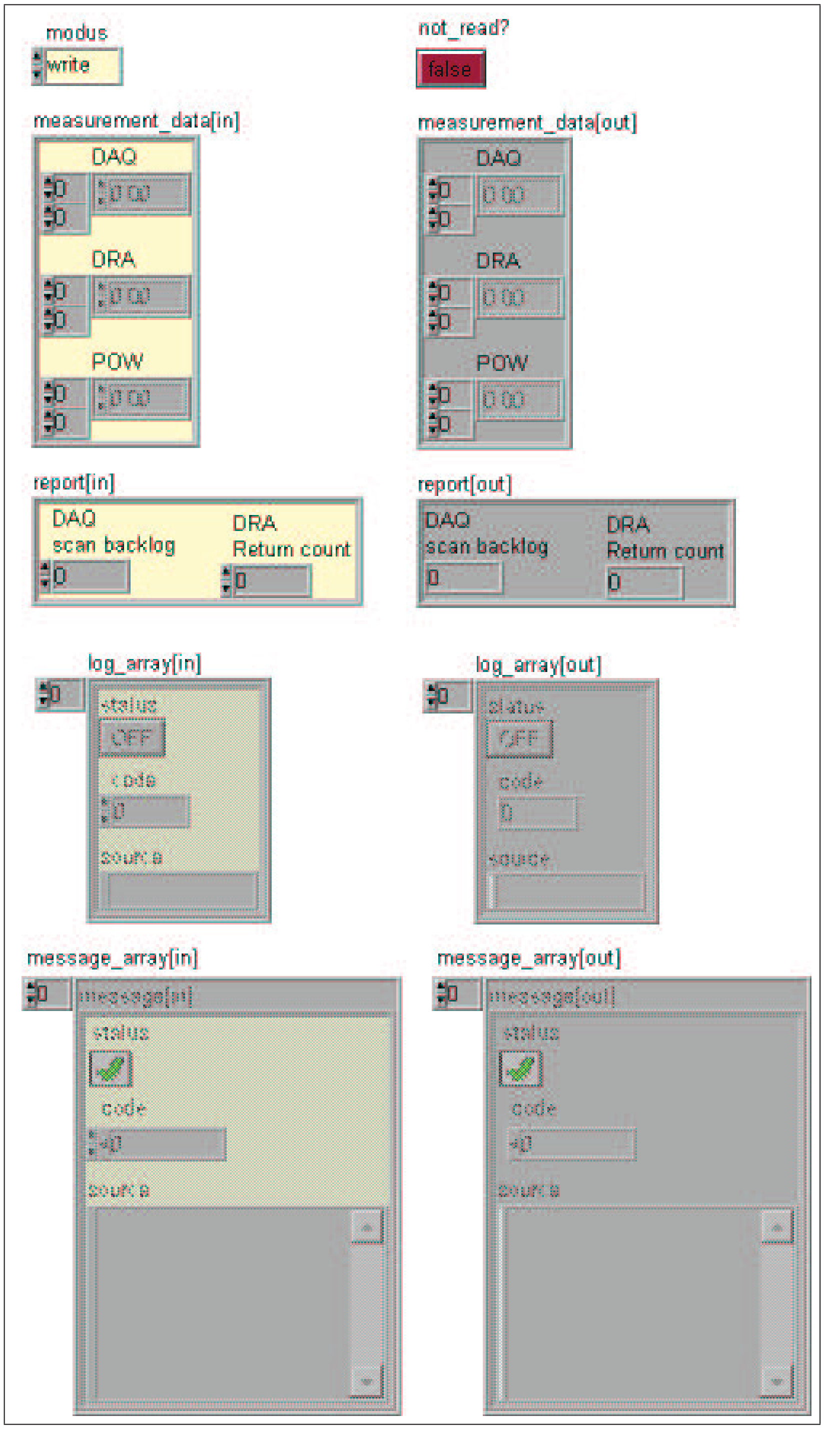


Figure 156: Front Panel of TSUI_TemporaryStorage_UserInterface.vi

report[in]

- **DAQ scan backlog** scan backlog is the amount of data acquired minus the amount of data read
- **DRA Return count** passes on the number of bytes actually read in RSRI_Rack-Serial_Read_dIrect.vi.

modus determines the access mode of the caller.

log_array[in] are messages consisting of concentrated and reformatted conditions

-
  - **status**
  - **code**
  - **source**

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

not_read? indicates if the temporary storage contains any data.

measurement_data[out] all measurment_data to visualize at the screen

- **DAQ** passes on measurement data acquired at RDRD_RackDAQ_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **DRA** passes on measurement data acquired at RSRD_RackSerial_ReadData.vi from the writing vi to the storage.
  - **Measured Value**
- **POW** passes on measurement data acquired at PARD_PowerAnalyzer_ReadData.vi from the writing vi to the storage.
  - **Measured Value**

report[out]

- **DAQ scan backlog** scan backlog is the amount of data acquired minus the amount of data read
- **DRA Return count** passes on the number of bytes actually read in RSRI_RackSerial_Read_dIrect.vi.

log_array[out] are messages consisting of concentrated and reformatted conditions

-
  - **status**
  - **code**

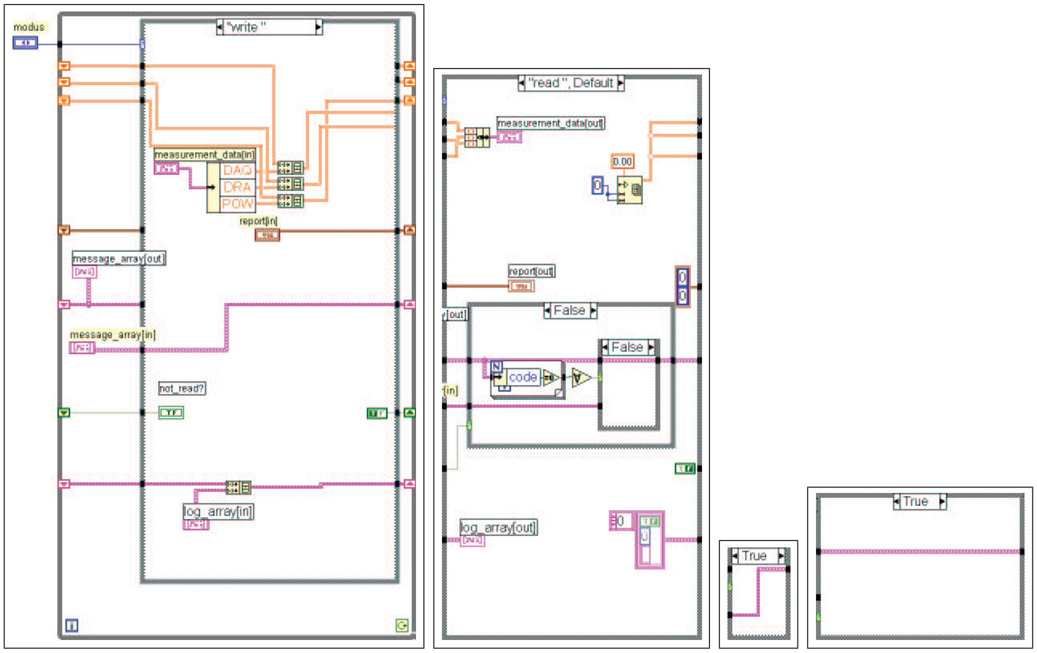


Figure 157: Block Diagram of TSUI_TemporaryStorage_UserInterface.vi

source

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

message[out] is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .

status is TRUE if an error occurred, or FALSE if not.

code is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.

source indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

51.4 Block Diagram

See figure 157 on page 130.

51.5 List of SubVIs

51.6 History

TSUI_TemporaryStorage_UserInterface.vi History Current Revision: 62

52 TSUO_TemporaryStorage_UserOperation.vi

This VI is used as temporary storage of data, messages and reports. There are two ways to access this VI, mode: READ and mode: WRITE.

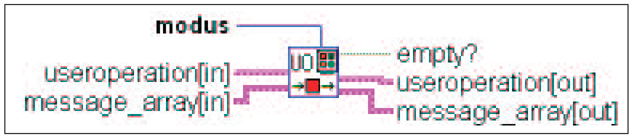


Figure 158: Connector Pane of TSUO_TemporaryStorage_UserOperation.vi

READ:
1. *useroperation[out]* is read and the referring storage emptied
2. *message_array[out]* is read and afterwards replaced by *message_array[in]*
3. *empty* := TRUE

WRITE:
1. if *save?* = TRUE then *useroperation[out]* is replaced by *useroperation[in]* else *useroperation[out]* is emptied.
3. if *empty?* = FALSE and *message_array[out]* contains no error then *message_array[out]* is replaced by *message_array[in]*
4. *empty?* := FALSE

52.1 Connector Pane

See figure 158 on page 131.

52.2 Front Panel

See figure 159 on page 132.

52.3 Controls and Indicators

modus determines the access mode of the caller.

useroperation[in]

- **save?** indicates if the save_button is pressed or not
- **instant** instant when save_button was pressed
- **Power Settings** array of boolean, referring to the powersettings which were set at the InterAction Panel of the User Interface.
  - **Charge Battery 2**

message_array[in] is an array of *message[in]* which describe the error status before the actual VI executes. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[in]** is a cluster that describes the message status before the actual VI's execution. This message is checked or passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

empty? indicates if the temporary storage contains any data.

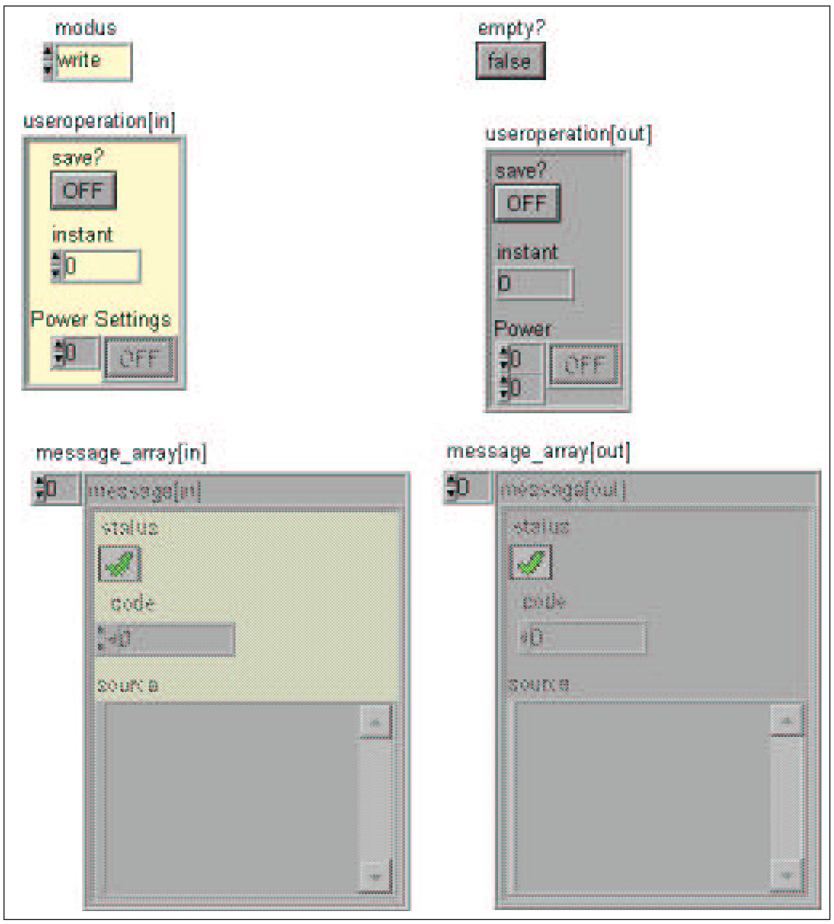


Figure 159: Front Panel of TSUO_TemporaryStorage_UserOperation.vi

useroperation[out]

- **save?** indicates if the save_button is pressed or not
- **instant** instant when save_button was pressed
- **Power** array of boolean, referring to the powersettings which were set at the InterAction Panel of the User Interface.
  - **Charge Battery 2**

message_array[out] is an array of *message_[out]* which describe the message status after the actual VI's execution. Usually the messages are checked or passed on. Messages can be conditions, commands, confirmations, logs and reports. The collectivity of all messages ensures the communication between all parts of *LOLA*.

- **message[out]** is a cluster that describes the message status before the actual VI executes. This message is checked or only passed on. It contains *status* , *code* and *source* .
  - **status** is TRUE if an error occurred, or FALSE if not.
  - **code** is the number identifying a message. If *status* is TRUE, *code* is an error code. If *status* is FALSE, *code* can be zero, a warning or a command code.
  - **source** indicates the origin of the message, if any. Usually *source* is the name of the VI in which the message occurred. *Source* can also give further detail.

52.4 Block Diagram

See figure 160 on page 134.

52.5 List of SubVIs

52.6 History

TSUO_TemporaryStorage_UserOperation.vi History Current Revision: 70

53 UIAV_UserInterface_AVeraging.vi

array_averaging :
First column: time axis
Further column: according measurement values

This VI calculates the area below the graphs represented by the measurement values and divides it by *time_period* (Tb-Ta). *average_values* is the result.

53.1 Connector Pane

See figure 161 on page 134.

53.2 Front Panel

See figure 162 on page 134.

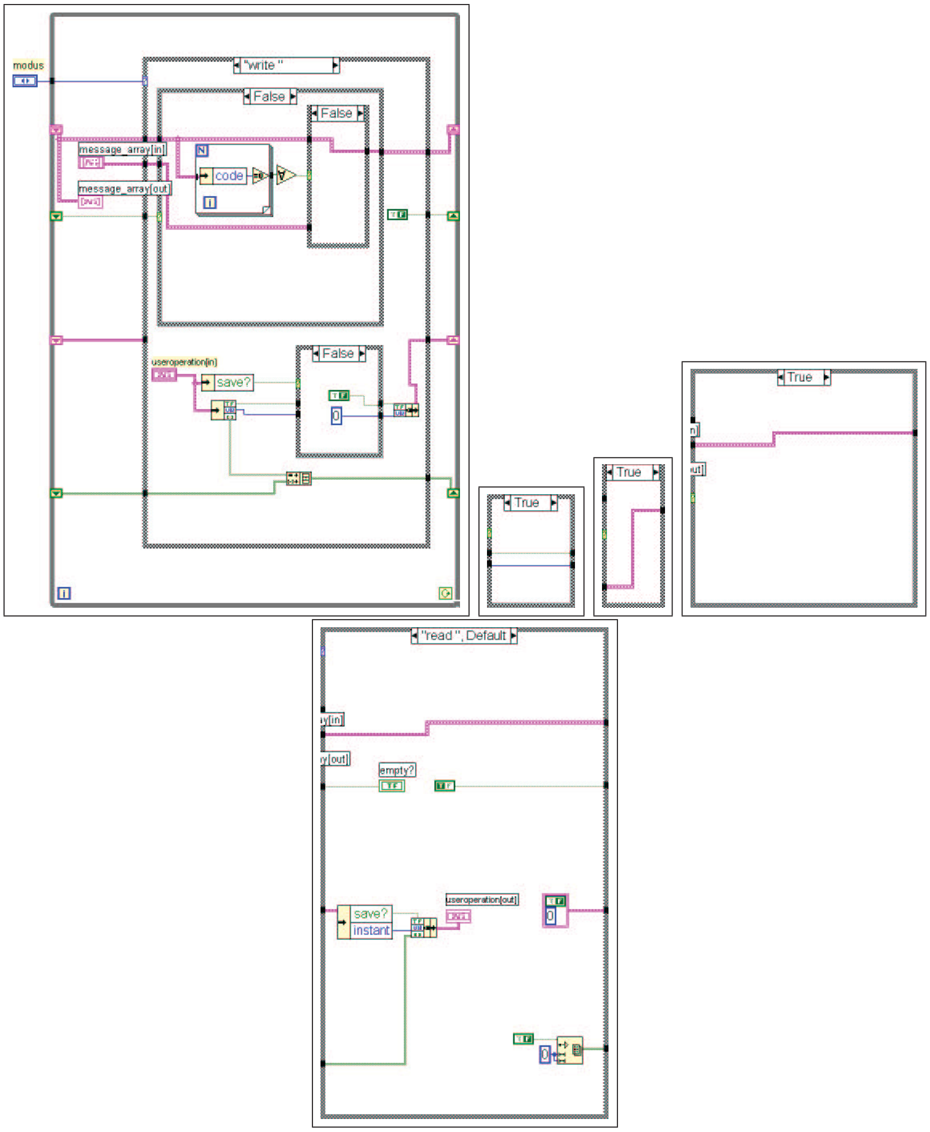


Figure 160: Block Diagram of TSUO_TemporaryStorage_UserOperation.vi

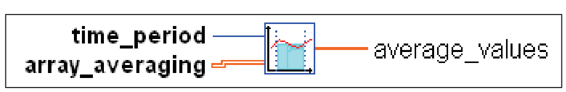


Figure 161: Connector Pane of UIAV_UserInterface_AVeraging.vi

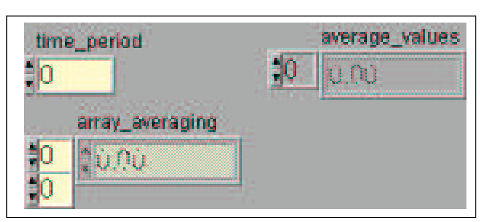


Figure 162: Front Panel of UIAV_UserInterface_AVeraging.vi

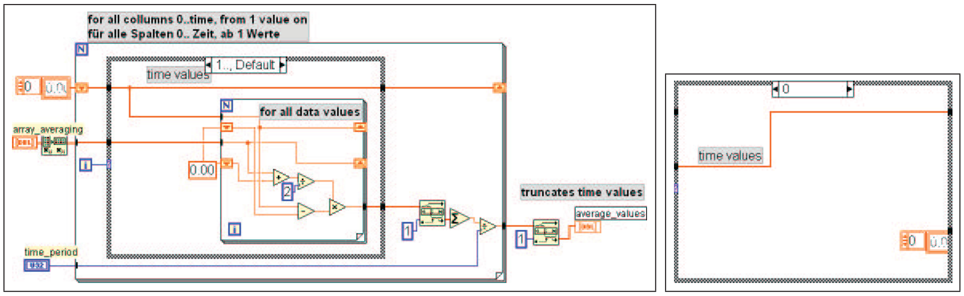


Figure 163: Block Diagram of UIAV_UserInterface_AVeraging.vi

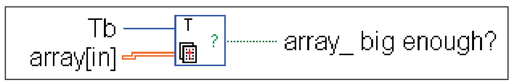


Figure 164: Connector Pane of UIIT_UserInterface_Instant-larger-Tb.vi

53.3 Controls and Indicators

time_period the time period defines the interval for averaging it is Tb-Ta

array_averaging array that should be averaged

first row is the time, the other are data

values

average_values

value

53.4 Block Diagram

See figure 163 on page 135.

53.5 List of SubVIs

53.6 History

UIAV_UserInterface_AVeraging.vi History Current Revision: 25

54 UIIT_UserInterface_Instant-larger-Tb.vi

If the latest instant is larger than *Tb array[in]* is big enough to be averaged

54.1 Connector Pane

See figure 164 on page 135.

54.2 Front Panel

See figure 165 on page 136.

Figure 165: Front Panel of UIIT_UserInterface_Instant-larger-Tb.vi

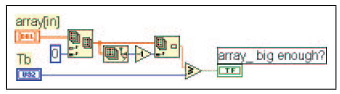


Figure 166: Block Diagram of UIIT_UserInterface_Instant-larger-Tb.vi

54.3 Controls and Indicators

Tb

array[in]

  Measured Value

array_ big enough? If the latest instant is larger than Tb the array is big enough

54.4 Block Diagram

See figure 166 on page 136.

54.5 List of SubVIs

54.6 History

UIIT_UserInterface_Instant-larger-Tb.vi History Current Revision: 30

55 UIRC_UserInterface_RearrangeChannels.vi

This VI rearranges channels according to wether they are turned on or off and calculates the value relative to the rated value for all channels.

55.1 Connector Pane

See figure 167 on page 137.

55.2 Front Panel

See figure 168 on page 137.

55.3 Controls and Indicators

channel_properties[in] properties listed for each channel

  on/off_array

    Boolean

  color_array

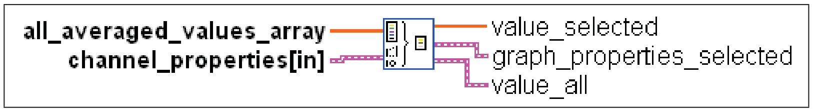


Figure 167: Connector Pane of UIRC_UserInterface_RearrangeChannels.vi

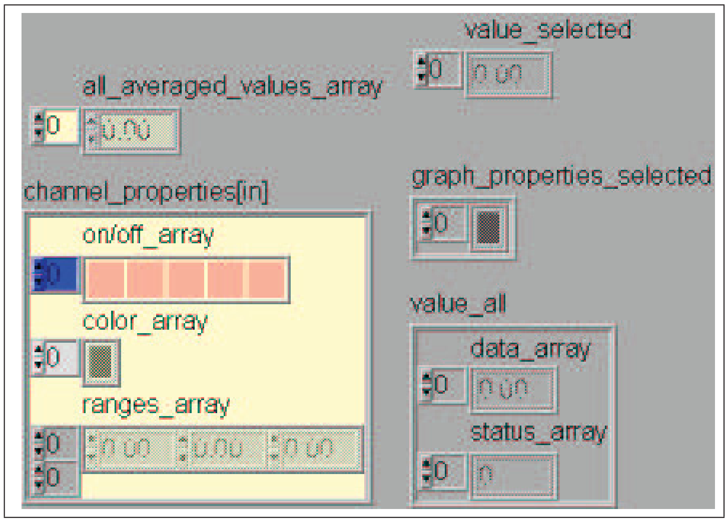


Figure 168: Front Panel of UIRC_UserInterface_RearrangeChannels.vi

color

ranges_array

range

all_averaged_values_array raw input data for the Display Graph

value

value_selected values of channels that are selected for being displayed in the Display Graph.

value

graph_properties_selected graph properties for channels that are selected for display graph

colors_array

color

value_all changed values for all channels

data_array

value

status_array

value

55.4 Block Diagram

See figure 169 on page 138.

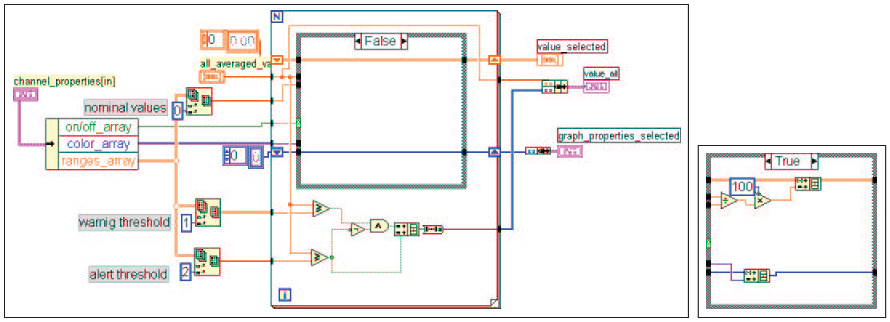


Figure 169: Block Diagram of UIRC_UserInterface_RearrangeChannels.vi

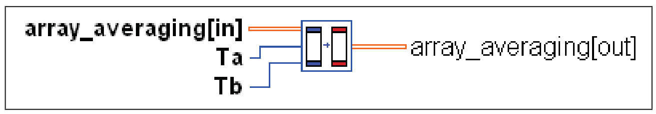


Figure 170: Connector Pane of UIRT_UserInterface_Rearrange-Topandbottom.vi

55.5 List of SubVIs

55.6 History

UIRC_UserInterface_RearrangeChannels.vi History Current Revision: 32

56 UIRT_UserInterface_Rearrange-Topandbottom.vi

First and last record are changed in the way that their time value is equal to *Ta* and *Tb* . The according measurement-values are computed via linear interpolation.

The result is that the area under the graph lasts only from Ta to Tb.

56.1 Connector Pane

See figure 170 on page 138.

56.2 Front Panel

See figure 171 on page 139.

56.3 Controls and Indicators

Tb end of interval for interpolation

array_averaging[in] array in

the span of the time-axis overlaps the span from Ta to Tb.

Measured Value

Ta start of interval for interpolation

array_averaging[out] array out

the time axis has Ta as first and Tb as last value.

s

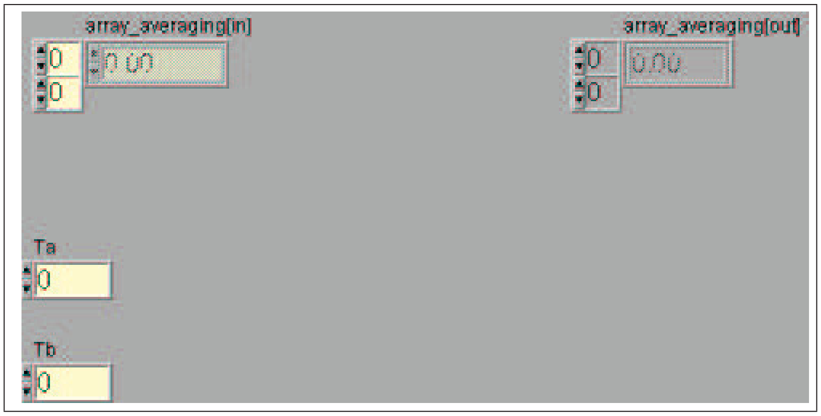


Figure 171: Front Panel of UIRT_UserInterface_Rearrange-Topandbottom.vi

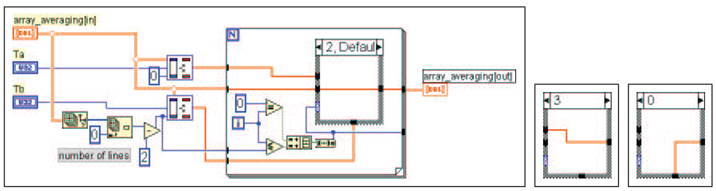
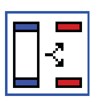


Figure 172: Block Diagram of UIRT_UserInterface_Rearrange-Topandbottom.vi

56.4 Block Diagram

See figure 172 on page 139.

56.5 List of SubVIs



UITC_UserInterface_Topandbottom-Create.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\UserInterface.llb\-
UITC_UserInterface_Topandbottom-Create.vi

56.6 History

UIRT_UserInterface_Rearrange-Topandbottom.vi History Current Revision: 28

57 UISA_UserInterface_SplitArray.vi

This VI splits array[in] into array_averaging and array_remaining at instant Tb.

Array_averaging lasts from index=0 to index(Tb)+1.

Array_remaining lasts from index(Tb)-1 to the end.

57.1 Connector Pane

See figure 173 on page 140.

57.2 Front Panel

See figure 174 on page 140.

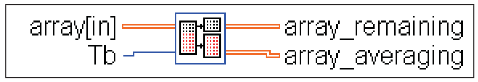


Figure 173: Connector Pane of UISA_UserInterface_SplitArray.vi

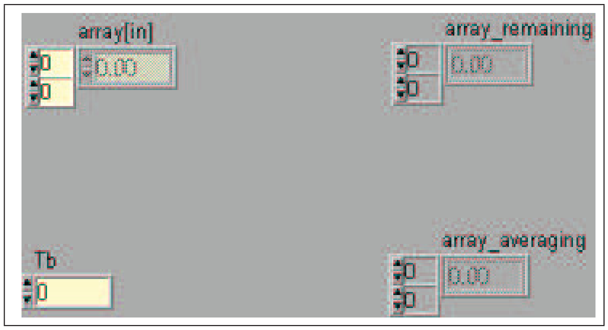


Figure 174: Front Panel of UISA_UserInterface_SplitArray.vi

57.3 Controls and Indicators

Tb instant where array[in] is split

array[in] input array

value

array_averaging lasts from index=0 to index(Tb)+1.

value

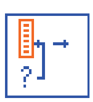
array_remaining lasts from index(Tb)-1 to the end.

Measured Value

57.4 Block Diagram

See figure 175 on page 141.

57.5 List of SubVIs



GLSN_GLobal_SearchforNumber.vi
D:\Aktuelle_Arbeit\Version 1.8 in Arbeit\global.llb\GLSN_GLobal_SearchforNumber.vi

57.6 History

UISA_UserInterface_SplitArray.vi History Current Revision: 12

58 UITC_UserInterface_Topandbottom-Create.vi

interpolates the values of *array [in]* after *position_index* at time position *time_value* .

58.1 Connector Pane

See figure 176 on page 141.

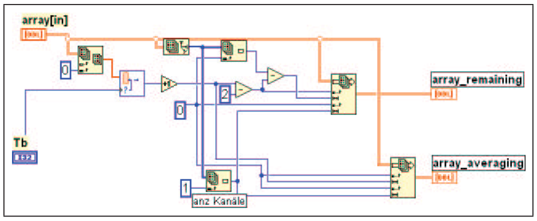


Figure 175: Block Diagram of UISA_UserInterface_SplitArray.vi

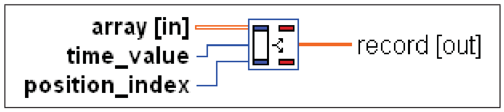


Figure 176: Connector Pane of UITC_UserInterface_Topandbottom-Create.vi

58.2 Front Panel

See figure 177 on page 142.

58.3 Controls and Indicators

array [in]

Measured Value

position_index

time_value

record [out] interpolated record

58.4 Block Diagram

See figure 178 on page 142.

58.5 List of SubVIs

58.6 History

UITC_UserInterface_Topandbottom-Create.vi History Current Revision: 17

59 UIUI_UserInterface_UserInterface.vi

The User Interface displays data and offers the ability to interact with the system.

The User Interface is divided into five panels. The Channel Panel, Graph Panel, Status Panel and the Message Panel display information. The Interaction Panel contains buttons and scrollbars to interact with the system.

CHANNEL PANEL
The Channel Panel is divided into rows and lines. Each line contains information of a channel.
The *on/off* row consists of buttons to turn the graph of the channel on and off.
The *color* row is to select the color of the according graph.
The *names* row contains the name and the unit of the channel.

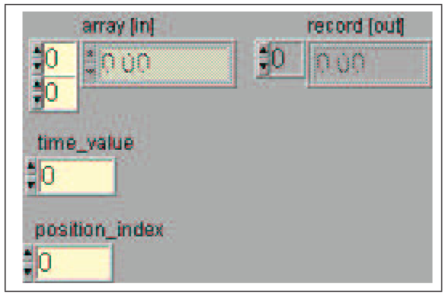


Figure 177: Front Panel of UITC_UserInterface_Topandbottom-Create.vi

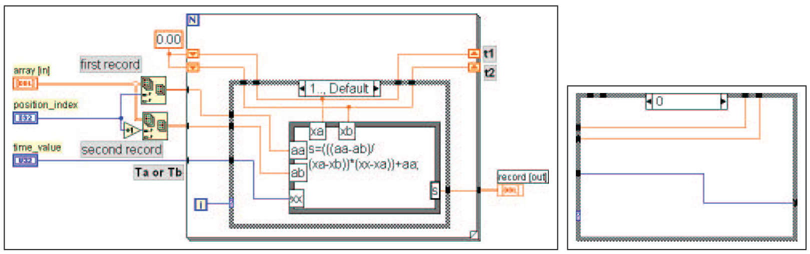


Figure 178: Block Diagram of UITC_UserInterface_Topandbottom-Create.vi

The *nominal* row contains the nominal value of the channel.
The *current* row contains the actual value of the channel.
The *status* row displays the status of the channel that can be green (means that the system is in *operating condition*), yellow (means that the channel has reached the warning level), red (means that the channel has reached the alarm level)

GRAPH PANEL
The Graph Panel displays trends of turned on channels. Graphs are linear interpolations of original data. This is to bring them all into one display.
The abscissa (time-axis) indicates interpolation periods.
The ordinate (value-axis) indicates the relative value in percent (%) of the nominal value.

STATUS PANEL
The Status Panel is a sketch of the test assembly where status signs indicate the state of the system.

MESSAGE PANEL
The Messages Panel displays user-relevant messages.

INTERACTION PANEL
Most buttons for user-interaction are located in the Interaction Panel.
The user can:
1. Stop the program
2. Trigger a measurement point
3. Turn on and off parts of the test assembly

59.1 Connector Pane

See figure 179 on page 143.

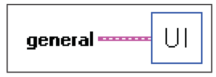


Figure 179: Connector Pane of UIUI_UserInterface_UserInterface.vi

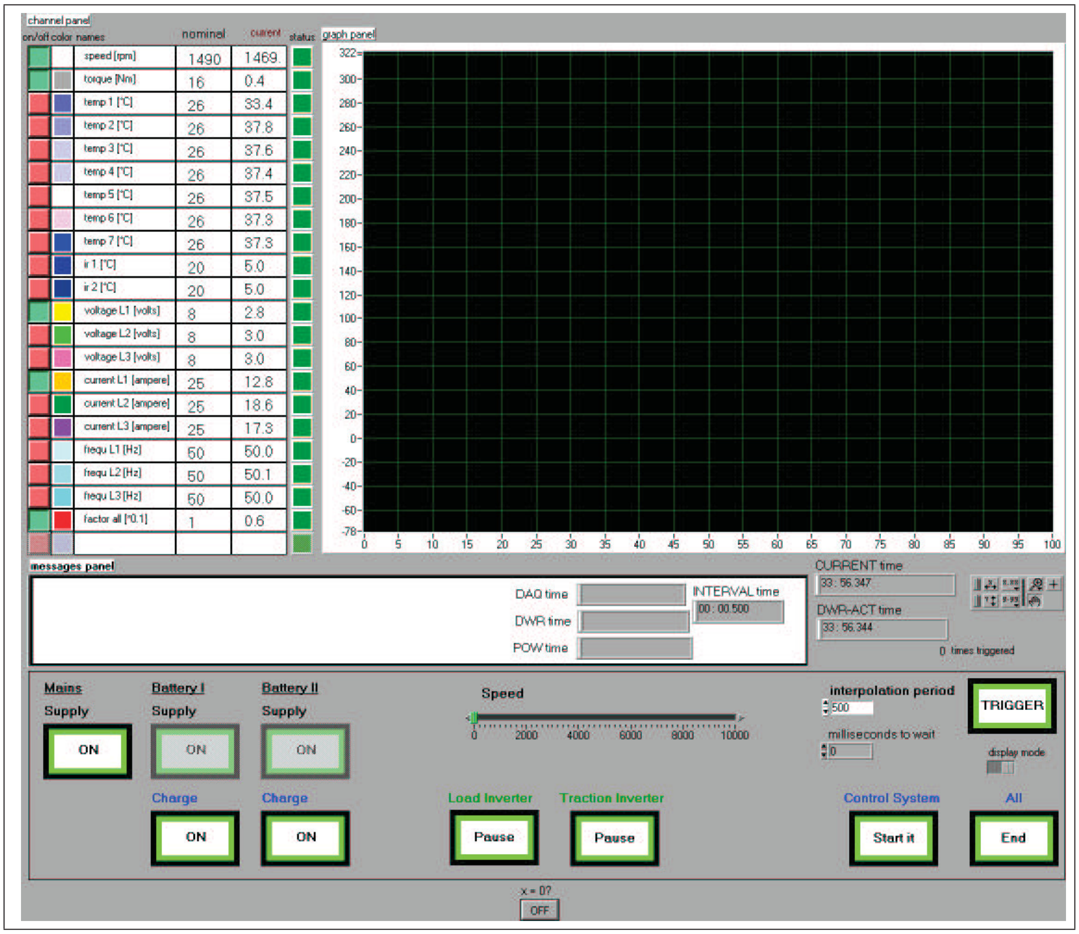


Figure 180: Front Panel of UIUI_UserInterface_UserInterface.vi

59.2 Front Panel

See figure 180 on page 143.

59.3 Controls and Indicators

interpolation period period of time which is relevant for averaging each channel

trigger when the trigger button is pressed, one measuring is taken

general

- **standard_path** path where the program is located
- **start_time** [ms] is the time reference. In *LOLAstart_time* is set at INKH_INitialisation_KonfigHardware.vi at the start of the program.
- **list_column_names** is an array containing the titles of the measurement categories The names' order must match to the commands' order.
  - **name_of_channel** channel name is displayed in the user interface, it consists of identifier and unit

- **list_ranges** is an array of all rated-, warning- and alarm-levels ordered according to *list_column_names*
  - **value**
- **semaphores** is an array of names of all existing semaphores
  - **name**

battery2_supply

battery2_charge

main_supply

battery1_supply

battery1_charge

color array of buttons to change color of the according channel in the graph.

- **color** color of channel

stop when the stop button is pressed, the program is stopped

on/off array of buttons to turn displaying the channels in the graph on and off

- **Boolean** channel on/off

display mode switch to change between display modes either graph or status information

Speed

milliseconds to wait

Start when the trigger button is pressed, one measuring is taken

Pause Traction Inverter when the trigger button is pressed, one measuring is taken

Pause load inverter when the trigger button is pressed, one measuring is taken

motor test

inverter test

current current value of channel

- **value** current value of channel

user interface is working if this indicator is blinking, the userinterface is working
will become obsolete as user interface is always working

names of channels including unit

- **String** name of channel including unit

nominal nominal value of channel is displayed here

- **Numeric** nominal value of channel is displayed here

motor test

motor test

messages panel all relevant messages are displayed here

graph panel each channel is displayed relatively to its nominal value in percent (%)

status green .... in operating condition
yellow .... warning
red .... alarm

Ring green .... in operating condition
yellow .... warning
red .... alarm

times triggered displays number of times triggered since start of measurement

INTERVAL time

CURRENT time

DWR-ACT time

POW time

DWR time

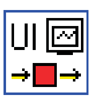
DAQ time

x = 0?

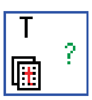
59.4 Block Diagram

See figure 181 on page 146 and figure 182 on page 147.

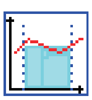
59.5 List of SubVIs



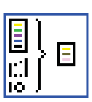
TSUI_TemporaryStorage_UserInterface.vi
D:\Version 2.0\Temporary_Storages.llb\-
TSUI_TemporaryStorage_UserInterface.vi



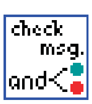
UIIT_UserInterface_Instant-larger-Tb.vi
D:\Version
2.0\UserInterface.llb\UIIT_UserInterface_Instant-larger-Tb.vi



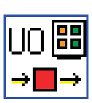
UIAV_UserInterface_AVeraging.vi
D:\Version
2.0\UserInterface.llb\UIAV_UserInterface_AVeraging.vi



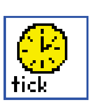
UIRC_UserInterface_RearrangeChannels.vi
D:\Version
2.0\UserInterface.llb\UIRC_UserInterface_RearrangeChannels.vi



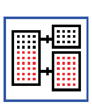
GLCM_GLobal_CheckMessages.vi
D:\Version 2.0\Global.llb\GLCM_GLobal_CheckMessages.vi



TSUO_TemporaryStorage_UserOperation.vi
D:\Version 2.0\Temporary_Storages.llb\-
TSUO_TemporaryStorage_UserOperation.vi



GLCL_GLobal_CLock.vi
D:\Version 2.0\Global.llb\GLCL_GLobal_CLock.vi



UISA_UserInterface_SplitArray.vi
D:\Version
2.0\UserInterface.llb\UISA_UserInterface_SplitArray.vi

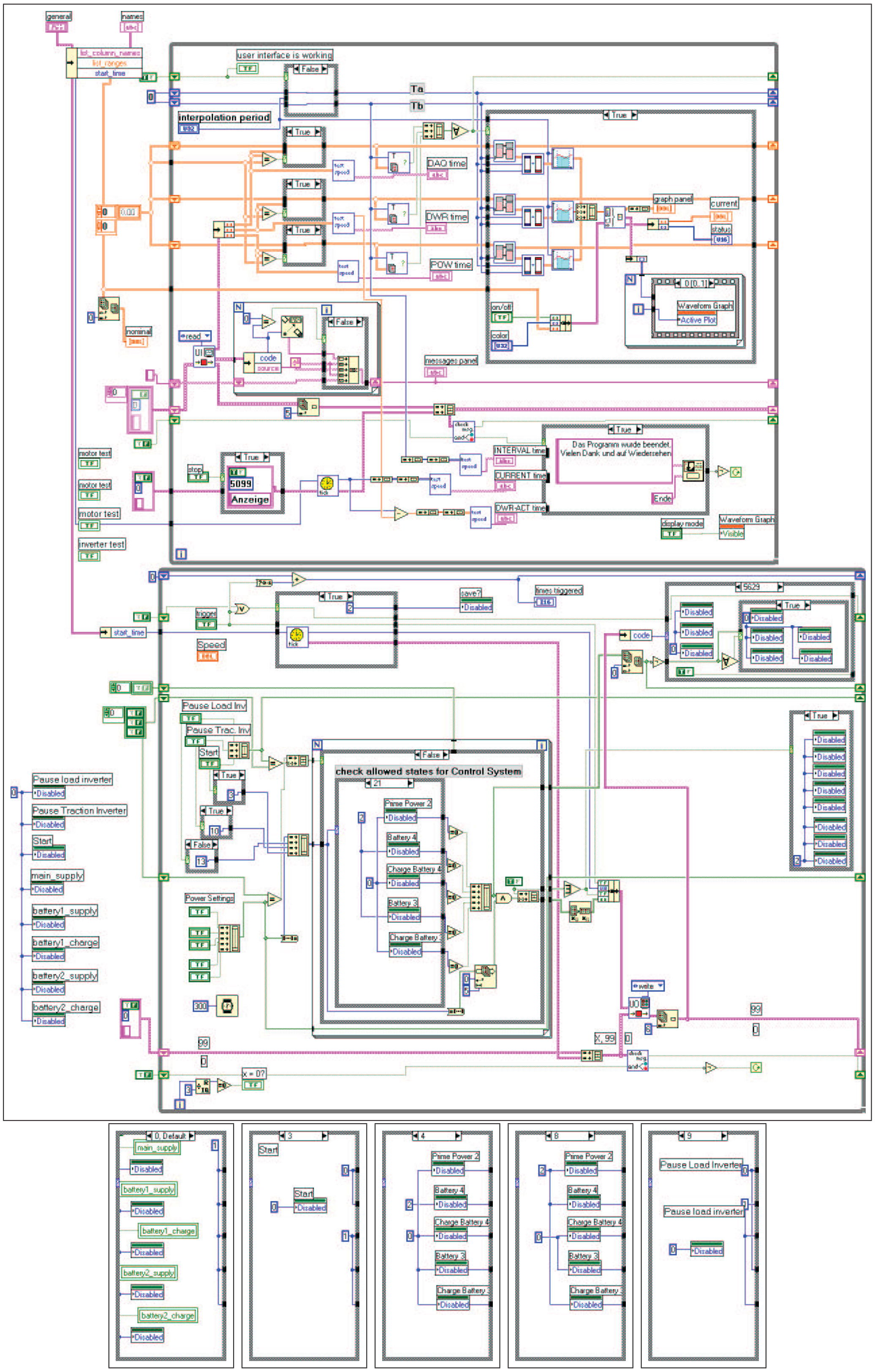


Figure 181: Block Diagram (a) of UIUI_UserInterface_UserInterface.vi

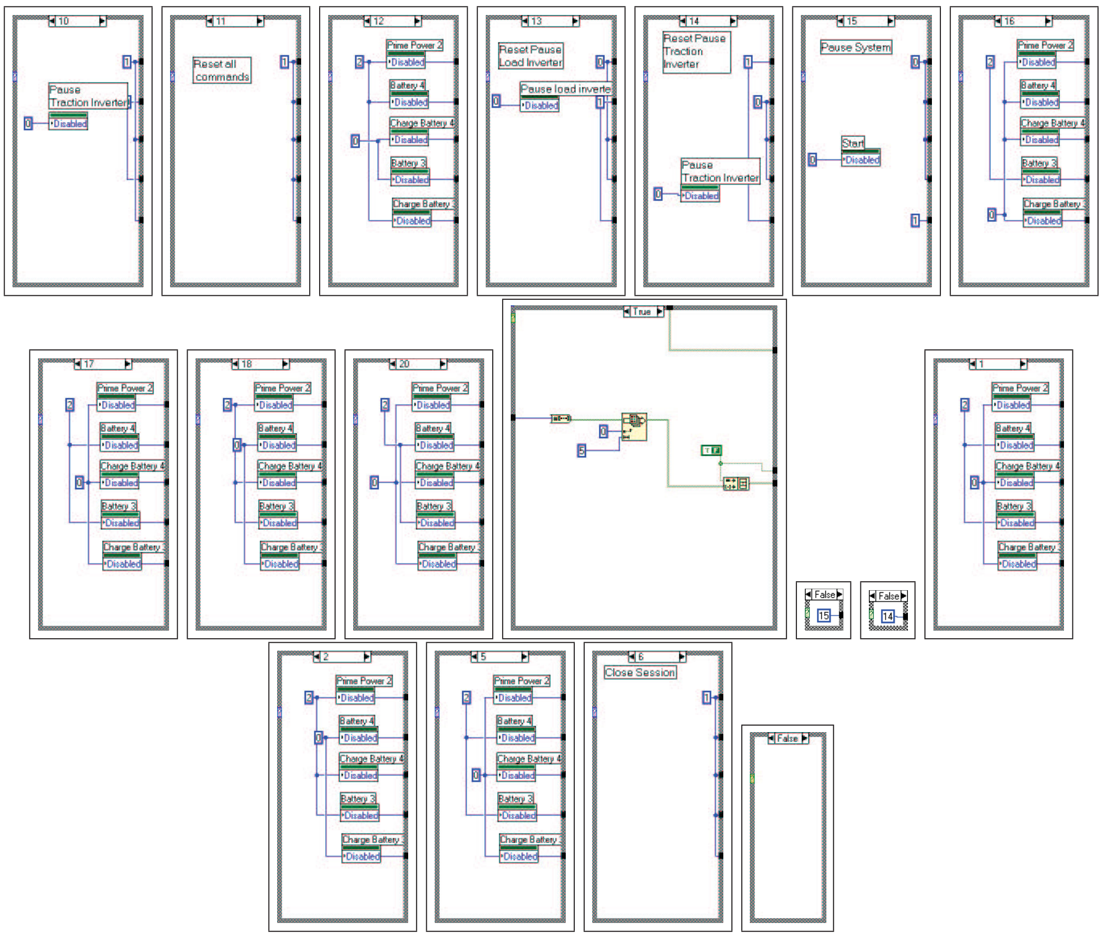
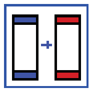
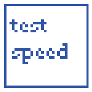


Figure 182: Block Diagram (b) of UIUI_UserInterface_UserInterface.vi



UIRT_UserInterface_Rearrange-Topandbottom.vi
D:\Version 2.0\UserInterface.llb\UIRT_UserInterface_Rearrange-Topandbottom.vi



GLTS_GLobal_TestSpeed.vi
D:\Version 2.0\misc\TestVis\GLTS_GLobal_TestSpeed.vi

59.6 History

UIUI_UserInterface_UserInterface.vi History Current Revision: 222

Abbreviations

A/D Analogue/ Digital

API Application Programming Interface

DAQ Data AcQuisition

DICE DICE_DIstritbution_CEnter.vi

DDS Data Delivery System

GPIB General Purpose Interface Bus

HCCE HCCE_HardwareConfiguration_CEnter.vi

INCE INCE_INinizialisation_CEnter.vi

ININ ININ_INinizialisation_INizialisation.vi

I/O Input/ Output

LOLA Low Observing Laboratory Application

MGS Message Gathering System

PARD PARD_PowerAnalyzer_ReadData.vi

PC Personal Computer

PLC Programmable Logical Controller

RDRD RDRD_RackDaq_ReadData.vi

RPM Revolution Per Minute

RS232 Recommended Standard 232

RSRD RSRD_RackSerial_ReadData.vi

RSWS RSWS_RackSerial_WriteS7.vi

SACO SACO_SAve_COnversion.vi

SASH SASH_SAve_SHot.vi

SICE SICE_SInage_CEnter.vi

TRMS True Root Means Square

VI Visual Instrument

VISA Virtual Instrumentation Software Architecture

VME Versa Module Eurocard

VXI VME eXtension for Instrumentation

Literaturverzeichnis

[1] Wolfgang Kurt Mörtl, *Teststand of Electrical Drives, Acquisition of Stationary and Transient Quantities using a Personal Computer*. Institute of Electrical Engineering Montan University Leoben (2001)

[2] Lenze, *Global Drive, Servo Umrichter 9300*. 1998 Lenze GmbH & Co KG, Stand 06.99

[3] Lenze, *Betriebsanleitung Global Drive, Servo Umrichter 9300*. 1998 Lenze GmbH & Co KG, Stand 2.0 1298

[4] Lenze, *Betriebsanleitung, Bremseinheit 9350*. 1997 Lenze GmbH & Co KG, Stand 03.04.2000

[5] Lenze, *Betriebsanleitung , Servo Motoren*. 1997 Lenze GmbH & Co KG, Stand 03.04.2000

[6] Staiger & Mohilo, *Drehmoment, Drehzahl, MeSSeinrichtung I40014*. 1990 Staiger & Mohilo, Stand 27.04.90BE GS2511

[7] Staiger & Mohilo, *Bedienungsanleitung Nr.1232,*. 1990 Staiger & Mohilo, Stand 06.96 Nr.1232

[8] Dewetron, *Dewe-Rack, Technical Reference Manual*. 1998 - 2000 Dewetron GesmbH, Release July 2000

[9] Dewetron, *Dewe-Rack, Software User Manual*. 1998 - 2000 Dewetron GesmbH, Release July 1999

[10] Lem, *Poweranalyzer D4000, User Manual*. 1999 Lem Norma GesmbH, Release 1999

[11] Lem, *Software, LabVIew Driver for D4000*. 1999 Lem Norma GesmbH, Release 1999

[12] Ωœmega, *OS5550/ OS550-BB Series, Industrial Infrared Thermometer/ Transmitter, Users Guide*. 2001 Ω œmega

[13] Siemens, *SIMATIC, Automatisierungssystem S/-200, Systemhandbuch*. Siemens AG 1997

[14] National Instruments, *LabView^{TM}, User Manual 5.1*. Januar 1998 Edition, Part number 320999B-01

[15] National Instruments, *LabView^{TM} and BridgeView^{TM}, G Programmin Reference Manual*. Januar 1998 Edition, Part number 321296B-01

[16] National Instruments, *LabView, Function and VI Reference Manual*. Januar 1998 Edition, Part number 321526B01

[17] National Instruments, *LabView, Data Acquisition Basics Manual*. Januar 2000 Edition, Part number 320997E-01

[18] National Instruments, *DAQ, 6023E/6024E/6025E User Manual, Multifunction I/O Boards for PCI, PXI and CompactPCI Bus Computers*. Januar 1999 Edition, Part number 322072B-01

[19] National Instruments, *NI-DAQ^{TM} User Manual for PC Compatibles, Version 6.7*. January 2000 Edition, Part number 321644F-01

[20] Ben Shneiderman *Designing the User Interface, Strategies for Effective Human-Computer Interaction*. 3rd edition, Addison Wesley Longman, Inc. 1998

[21] John G. Proakis, Dimitris G. Manolaeikis, *Digital Signal Processing; Principles, Algorithms, and Applications* 3rd. edition, New Jersey: Prentice-Hall, Inc. 1996.

[22] Ned Mohan, Tore M. Undeland, William P. Robbins, *Power Electronics: Converters, Applications, and Design*. John Wiley & Sons, INC., second edition 1989, 1995

[23] National Instruments, *Main www.ni.com*. July 2001

[24] LEM Norma, *www.lem.com*. July 2001

[25] Ω œmega, *www.omeg.com* July 2001

[26] Dewetron, sl www.dewetron.com July 2001