MOST

Media **O**riented **S**ystems **T**ransport

Multimedia and Control Networking Technology

Electrical Control Line Specification
Rev 1.1.1
11/2011

MOSTCO CONFIDENTIAL

See page 3 for the terms of disclosure

MOST® COOPERATION

Legal Notice

COPYRIGHT

LICENSE DISCLAIMER

Nothing on any MOST Cooperation Web Site, or in any MOST Cooperation document, shall be construed as conferring any license under any of the MOST Cooperation or its members or any third party's intellectual property rights, whether by estoppel, implication, or otherwise.

CONTENT AND LIABILITY DISCLAIMER

MOST Cooperation or its members shall not be responsible for any errors or omissions contained at any MOST Cooperation Web Site, or in any MOST Cooperation document, and reserves the right to make changes without notice. Accordingly, all MOST Cooperation and third party information is provided "AS IS". In addition, MOST Cooperation or its members are not responsible for the content of any other Web Site linked to any MOST Cooperation Web Site. Links are provided as Internet navigation tools only.

MOST COOPERATION AND ITS MEMBERS DISCLAIM ALL WARRANTIES WITH REGARD TO THE INFORMATION (INCLUDING ANY SOFTWARE) PROVIDED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall MOST Cooperation or its members be liable for any damages whatsoever, and in particular MOST Cooperation or its members shall not be liable for special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue, or loss of use, arising out of or related to any MOST Cooperation Web Site, any MOST Cooperation document, or the information contained in it, whether such damages arise in contract, negligence, tort, under statute, in equity, at law or otherwise.

FEEDBACK INFORMATION

Any information provided to MOST Cooperation in connection with any MOST Cooperation Web Site, or any MOST Cooperation document, shall be provided by the submitter and received by MOST Cooperation on a non-confidential basis. MOST Cooperation shall be free to use such information on an unrestricted basis.

TRADEMARKS

MOST Cooperation and its members prohibit the unauthorized use of any of their trademarks. MOST Cooperation specifically prohibits the use of the MOST Cooperation LOGO unless the use is approved by the Steering Committee of MOST Cooperation.

SUPPORT AND FURTHER INFORMATION

For more information on the MOST technology, please contact:

MOST Cooperation
Administration
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 966 50 00

E-mail: contact@mostcooperation.com
Web: www.mostcooperation.com

This Specification is Confidential Information of the MOST Cooperation. It may only be disclosed to member companies. Member companies wishing to discuss these Specifications with suppliers or other third parties must ensure that a commercially standard form of non-disclosure agreement has been previously executed by the party receiving such Specifications. Use of these Specifications may only be for purposes for which they are intended by the MOST Cooperation. Unauthorized use or disclosure is a violation of law.

Contents

BIBLIOGRAPHY **5**

DOCUMENT HISTORY **6**

1 INTRODUCTION **8**

1.1 Scope 8
1.2 Definitions 8

2 OVERVIEW **9**

3 ECL PROTOCOL **10**

3.1 System Test 10
3.1.1 Start Sequence 11
3.1.2 Parameter Sequence 11
3.1.3 Result Sequence 12
3.1.4 Reaction on Malfunction 14
3.2 Electrical Wake-up 15
3.2.1 Single EWU and Multiple EWUs 15
3.2.2 Multiple Initiators 15
3.2.3 Start of Network Service and RBD Activation 15

4 PHYSICAL DEFINITIONS **16**

4.1 ECL Circuit 16
4.2 Voltage Drops 18
4.3 Impulses 19
4.3.1 General 19
4.3.2 Wake-up Impulses 21
4.4 Triggers 22

5 SOFTWARE DESIGN [INFORMATIVE] **24**

5.1 Architecture 24
5.2 ECL Software Module 25
5.2.1 System Test Initiator Component 25
5.2.2 System Test Participant Component 25
5.2.3 Wakeup Initiator Component 26
5.2.4 Wakeup Participant Component 26
5.3 Interfaces 27
5.3.1 Timer Interface 27
5.3.2 Test Execution Application Interface 27
5.3.3 ECL Low Level Interface 28
5.3.4 System Test Initiator Interface 28
5.3.5 System Test Participant Interface 29
5.3.6 Wakeup Initiator Interface 29
5.3.7 Wakeup Participant Interface 29
5.4 State Machines 29
5.4.1 System Test Initiator State Machine 30
5.4.2 System Test Participant State Machine 33

6 APPENDIX A: LIST OF FIGURES **35**

7 APPENDIX B: LIST OF TABLES **36**

Bibliography

All documents, which are referenced by this MOST document, are listed here along with their versions.

Document		Revision
[1]	**MOST Specification**	3.0

Document History

Changes Revision 1.1 to Revision 1.1.1

Change Ref.	Section	Changes
1V11_001	General	– Correction of typos and clerical errors.
1V11_002	3.1	– Modified Figure 3-1 so that the different sequences always end on high level. – Introduced P_{Sync} and reduced the lower limit of $t_{StartUp}$ from 4 s to 0.1s.
1V11_003	3.1.1	– Reduced the lower limit of $t_{StartUp}$ from 4 s to 0.1s.
1V11_004	3.1.2	– Improved description of system tests identified by P1 – P5. – For the “MOST signal result, RBD result” test, all devices start performing RBD on the rising edge of P_{Sync}. – Renamed description of “MOST signal result, threshold” to “MOST signal result, coding error result”. – Introduced P_{Sync}.
1V11_005	3.1.3	– Replaced reference to “optical test” with “MOST signal result test”. – During the ECL system test result sequence the participant shall not enter sleep mode. – Distinguished between mandatory and optional MOST signal result tests. – For compliance testing, the initiator must have the ability to detect *On* = 0 and *On* = 1. – Completely revised content of Table 3-1 and Table 3-2. – For unsupported ECL tests, the participant answers with *En* = *On* = 1.
1V11_006	3.1.4	– Completely revised.
1V11_007	3.2	– Completely revised and restructured.

1V11_008	4.1	– Modified Table 4-1:
		  – Distinguished between General and Optional Termination Resistor.
		  – Removed sleep mode Termination Resistor entry.
		– Improved description of pulses that are ignored.

1V11_009	4.3.1	– Modified description of logical levels so that the range between $V_{inactive\text{-}min}$ and $V_{BATTERY}$ corresponds to “1” and the range between $V_{GND\_BATTERY}$ and $V_{active\text{-}max}$ to “0”. – In Table 4-5 and Table 4-6, t_{Pause} is now defined as pause after TSI or EWU.
1V11_010	4.3.2	– No longer referring to “wake-up with or without retries” but single EWU and multiple EWUs. – Placed description of single EWU in front of the description of multiple EWUs.
1V11_011	5.4.1	– Transition from state “StartEclContinued” to state “Idle” added to reflect requirements from Table 3-3: Reaction on malfunction, based on timing range. – P_{Sync} replaces the “6^{th} $t_{TestParam}$”.
1V11_012	5.4.2	– Fixed typo in figure of state machine.

Changes Revision 1.0.1 to Revision 1.1

Change Ref.	Section	Changes
1V1_001	General	– Corrected minor clerical errors. – Renamed t_{TWU} to t_{TSI}.
1V1_002	1.2	Added TSI and WI to definitions.
1V1_003	2	Shortened description of electrical wake-up.
1V1_004	3	Added description of initiator and participant roles.
1V1_005	3.1	Modified *Figure 3-1: Sequence of the system test*.
1V1_006	3.1.1	– Turned *Start Sequence* paragraph into new section. – Description completely revised.
1V1_007	3.1.2	– Turned *Parameter Sequence* paragraph into new section. – Modified description of voltage levels and their corresponding logical values.
1V1_008	3.1.3	– Turned *Result Sequence* into new section. – Added *Table 3-1: Electrical and optical result values for standard tests*, which defines the En and On results.
1V1_009	3.1.4	New section *Reaction on Malfunction*.
1V1_010	4	Chapter completely reworked.
1V1_011	5	New chapter *Software Design [Informative]*.

Changes Revision 1.0 to Revision 1.0.1

Change Ref.	Section	Changes
1V01_001	General	Correction of minor clerical errors.
1V01_002	Bibliography	Promoted “References” section to “Bibliography” to match other MOST Cooperation documents.
1V01_003	3.1	Modified Figure 3-1 – introduced a pause of 2 * $t_{TestSep}$ after $t_{TestSync}$ – removed number of retries because this is System Integrator specific Modified description according to change in Figure 3-1.
1V01_004	4.4	Modified Figure 4-4 – introduced a pause of 2 * $t_{TestSep}$ after $t_{TestSync}$ – removed number of retries because this is System Integrator specific Modified description according to change in Figure 4-4.

Revision 1.0

Change Ref.	Section	Changes
1V0_001	General	Initial Version.

1 Introduction

1.1 Scope

This document defines the requirements for the optional Electrical Control Line (ECL) of a MOST system. The ECL allows triggering diagnostic functionalities like starting the ring break diagnosis or performing an electrical wake-up of the system.

1.2 Definitions

Term	Description
CU	Critical Unlock
ECL	Electrical Control Line
ECU	Electronic Control Unit
ISO	International Organization for Standardization
MOST	Media Oriented Systems Transport
RBD	Ring Break Diagnosis
SSO	Sudden Signal Off
TSI	System Test Start Impulse
WI	Wake-up Impulse

Table 1-1: Definitions

2 Overview

This specification defines a protocol for the optional Electrical Control Line (ECL) of a MOST system.

The ECL connects multiple ECUs. These ECUs can exchange information in a serial and digital way. Every ECU can use the ECL as sender and receiver.

The ECL may be used to trigger specific diagnostic mechanisms, for example, the ring break diagnosis (RBD) specified in the MOST Specification [1], if these can not be triggered by the method “Switch to Power” because of the system design. By the protocol defined here for the ECL, the system test initiator can trigger different tests with certain parameters and collect the results from all participating devices in the network. The participating devices are those that are connected to the ECL.

If required, the ECL can also be used to generate an electrical wake-up.

3 ECL Protocol

The protocol on the ECL provides two separate functions: first, a function for triggering a system test, which can be parameterized to provide different results, and second, a method for waking up the system by an electrical wake-up signal.

A device connected to the ECL can act in the role of an initiator and/or in the role of a participant.

There are two kinds of initiators: system test initiator and electrical wake-up initiator. The system test initiator triggers system tests and collects the results from the participants. The electrical wake-up initiator creates the electrical wake-up signal.

There are two kinds of participants: system test participant and electrical wake-up participant. A system test participant reports system test results to the initiator. An electrical wake-up participant is woken up upon detection of the wake-up signal.

In general, the protocol defines two states. The default value is a voltage level of V_{SUPPLY} (or close to V_{SUPPLY}). A sender can pull the voltage level of the ECL down to ground (or close to ground).

The ECL can be used for a MOST system that contains a maximum of 20 devices.

The System Integrator may define node classes. Every device is associated with exactly one node class. Within one system, a node class must not be assigned to more than one device.

Every node class occupies a result slot in the ECL test result sequence. Thus, the more node classes are defined, the longer the ECL test will last. The maximum number of node classes (m_C) is determined by the System Integrator. Since the number of devices is limited to 20, there may be more node classes than devices, leaving the test result slots of those classes that are not associated with a device unoccupied.

3.1 System Test

The system test is separated into a start sequence, then a sequence, which defines the parameters of the test, and finally a sequence, which collects the results of the test. The following diagram shows the whole sequence of the system test, the different phases of which are described in this section.

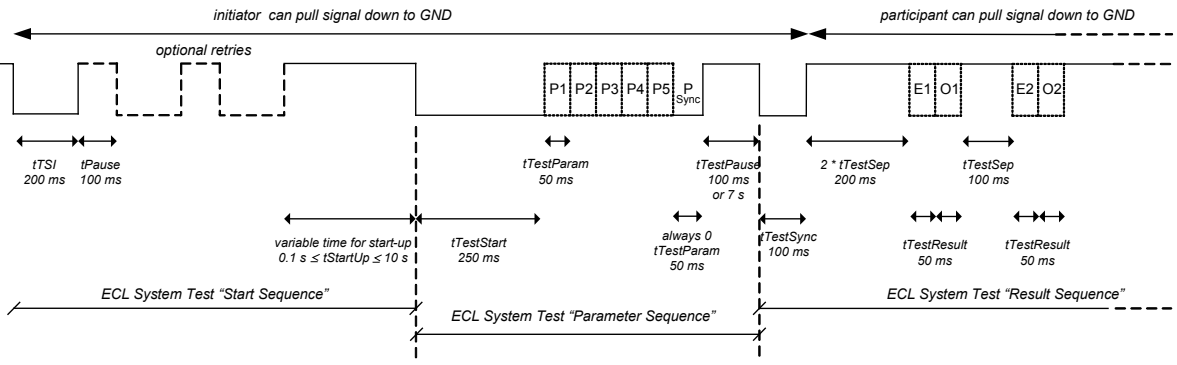

Figure 3-1: Sequence of the system test

3.1.1 Start Sequence

There is exactly one system test initiator device in the system; this device has to generate the start sequence. Therefore, this initiator device is the sender during the start sequence. The participants are the receivers during the start sequence.

After a system test start impulse (TSI) with a duration of t_{TSI} and optional retries (the number of retries is System Integrator specific), there is a variable time between 0.1 s and 10 s (System Integrator specific), to allow the system to start-up.
Optional retries can be used because depending on the ECL circuit, it is possible that only the rising edge of the first TSI can be detected.

Following the start-up phase, a signal on ground level of duration $t_{TestStart}$ marks the beginning of the system test.

Note: *After the test initiator device has sent its start sequence, it has to wait $t_{TestRestart\_Min}$ before triggering the system test again.*

$$t_{TestRestart\_Min} = t_{TestStart\_Max} + 6 * t_{TestParam\_Max} + t_{TestPause\_Max} + t_{TestSync\_Max} + t_{TestSep\_Max} + m_c * (2 * t_{TestResult\_Max} + t_{TestSep\_Max})$$

3.1.2 Parameter Sequence

After the start sequence, the initiator device generates several parameters that define the behavior of the test. Each parameter transmission lasts $t_{TestParam}$. A voltage level in the V_{active} range equals a logical parameter value of 0; a voltage level in the $V_{inactive}$ range equals a logical parameter value of 1. See chapter 4 for a more detailed definition.

- P1 to P5 define the kind of the system test:
    - P1 – P5: 00000 = MOST signal result, RBD result: All devices start performing RBD on the rising edge of P_{Sync}. The MOST signal result is returned with the corresponding bit (*On*) set to 0 by a participating device if a Stable Lock has been achieved on the link before the device at the end of the RBD phase. The test must be completed within 7 s.
    - P1 – P5: 01000 = MOST signal result, coding error result: The MOST signal result is returned with the corresponding bit (*On*) set to 0 by a participating device if a Stable Lock has been achieved and the signal quality (i.e., the number of coding errors detected) is better than a predefined threshold. This threshold (i.e., the number of acceptable coding errors), will be defined by the System Integrator. The test must be completed within 7 s.
    - P1 – P5: 10000 = only *alive* result: The participating device sets the corresponding result bit (*En*) to 0, if it has received this test signal via the ECL. This is used to test if a participant is alive (e.g., whether it is connected to power) and is connected to the ECL correctly. The test must be completed within 100 ms.
    - P1 – P5: 11000 = MOST signal result, SSO/CU result: The MOST signal result is returned with the corresponding bit (*On*) set to 0 by a participating device if it has not stored the detection of an SSO (Sudden Signal Off) or CU (Critical Unlock). The test must be completed within 100 ms.

Note: *The MOST signal results are only valid if the corresponding alive bit equals 0.*

After the parameters, P_{Sync} is set to 0 for the duration of $t_{TestParam}$ to synchronize the devices.

3.1.3 Result Sequence

The participants generate the sequence of results. Therefore these devices are the senders during the result sequence.

After the parameters, a test pause of duration $t_{TestPause}$ is inserted as separation from the results of the test. This test pause has a short value if an electrical test of the ECL is performed. In case of a MOST signal result test, it has a longer value to allow for the execution of the RBD or the evaluation of the transmission quality.

After the test pause, the initiator introduces the result phase of the ECL protocol. The falling edge marks the common trigger for the time base used during the result phase.

A pause of two times the duration of $t_{TestSep}$ marks the switch of control over the ECL from initiator to participants.

Afterwards, there are m_C slots wherein a dedicated device (participant) responds with first its *alive* result (*En*) and subsequently with its MOST signal result (*On*) by pulling the ECL down to ground for duration of $t_{TestResult}$ each. There are as many slots as there are node classes defined by the System Integrator. Each of the class slots with two results is separated by a pause of $t_{TestSep}$.
During the ECL system test result sequence, the participant shall not enter sleep mode in order to avoid unnecessary wake-ups caused by detection of other participant results.

The *alive* result will be interpreted as follows:

- *En* = 0 (V_{active}): the device corresponding to class n is *alive*,
- *En* = 1 ($V_{inactive}$): the device corresponding to class n is either not *alive* or not connected to the ECL.

The value of *On* depends on the kind of system test.

- *E1*: *alive* result of class 1
- *O1*: MOST signal result of class 1.
- *E2*: *alive* result of class 2
- *O2*: MOST signal result of class 2
- …
- Em_C : *alive* result of class m_C
- Om_C : MOST signal result of class m_C

Table 3-1 defines the *En* and *On* results for the mandatory test:

Test	P1 – P5	*En*	*On*	*En* Result	Measurement *On* Result	*On* Result
Only *alive* result	10000	0	1	device *alive*	No measurement action	N/A
		1		device not *alive*		

Table 3-1: Alive result and MOST signal result values for mandatory test

For compliance testing, the initiator must have the ability to detect *On* = 0 and *On* = 1.

Table 3-2 defines the *En* and *On* results for the optional MOST signal result tests:

Test	P1 – P5	*En*	*On*	*En* Result	Measurement *On* Result	*On* Result
MOST signal result, RBD result	00000	0	0	device *alive*	Lock detection within time $t_{TestPause}$	Stable Lock
			1			no Stable Lock
		1	1	device not *alive* or test not supported	No measurement action	N/A
MOST signal result, coding error result	01000	0	0	device *alive*	Count coding errors during the entire time of $t_{TestPause}$. Finally, store coding error counter value (CounterValue).	CounterValue ≤ N signal quality okay
			1			CounterValue > N signal quality not okay
		1	1	device not *alive* or test not supported	No measurement action	N/A
MOST signal result, SSO/CU result	11000	0	0	device *alive*	During the time $t_{TestPause}$ request the SSO/CU result via the Network Service interface.	SSO/CU result = “no result available” or “no fault saved”
			1			SSO/CU result = “Sudden Signal Off” or “Critical Unlock”
		1	1	device not *alive* or test not supported	No measurement action	N/A

Table 3-2: Alive result and MOST signal result values for optional tests

In the case that an optional ECL test is not supported, the participant answers with the result value *En* = *On* = 1 (no impulses will be generated).

The threshold N for the coding error counter value (CounterValue) has to be specified by the System Integrator.

3.1.4 Reaction on Malfunction

In Table 3-2, short circuits of the ECL line to ground and positive voltages (see section 4) or detected interfering impulses, are combined under the term “implausible signal level”.

The initiator checks its own signal levels during the test start sequence. The participant checks the signal level and timing of the test start sequence and the parameter sequence.

In the case of implausible signal levels, the system test initiator / participant shall react as follows:

<table>
<tr><th>Detected malfunction within timing range (Figure 3-1)</th><th>Initiator Reaction</th><th>Participant Reaction</th></tr>
<tr><td>t_{TSI}</td><td rowspan="3">Abort generation of system test start sequence</td><td rowspan="5">Abort reception and wait for start sequence</td></tr>
<tr><td>t_{Pause}</td></tr>
<tr><td>$t_{StartUp}$</td></tr>
<tr><td>$t_{TestStart}$</td><td rowspan="5">Ignore implausible signal level</td></tr>
<tr><td>$t_{TestParam}$</td></tr>
<tr><td>$t_{TestPause}$</td></tr>
<tr><td>$t_{TestSync}$</td><td rowspan="3">Ignore implausible signal level</td></tr>
<tr><td>$t_{TestSep}$</td></tr>
<tr><td>$t_{TestResult}$</td><td>Read parameter, depending on the used read algorithm (not specified here)</td></tr>
</table>

Table 3-3: Reaction on malfunction, based on timing range

3.2 Electrical Wake-up

An electrical wake-up impulse (EWU) of duration t_{EWU} defines the start of an electrical wake-up without a system test.

3.2.1 Single EWU and Multiple EWUs

A System Integrator may specify a certain number of EWU repetitions, for example, if the detection of a rising edge is required before wake-up evaluation can start. A single electrical wake-up impulse is referred to as “single EWU” while electrical wake-up repetitions are referred to as “multiple EWUs”.

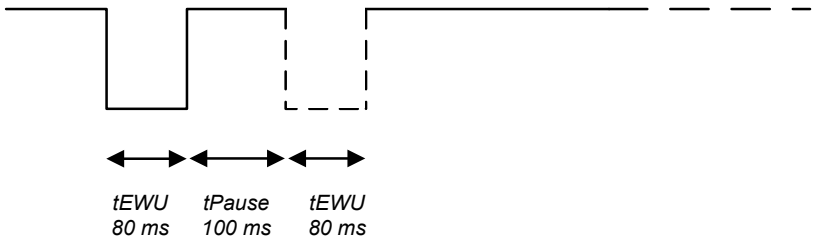

Figure 3-2: Sequence of the electrical wake-up

3.2.2 Multiple Initiators

It is possible that more than one device can create an EWU. In this case, there is more than one wake-up initiator in the system. Wake-up impulses from different initiators may overlap.
The reason for such a wake-up collision is that a later initiator has not detected the EWU of the first initiator and therefore performs its own EWU. Therefore, participants have wider timing tolerances than initiators (see Table 4-6).

When the system test initiator performs a system test, the other nodes that are able to initiate a wake-up must not perform an EWU.

3.2.3 Start of Network Service and RBD Activation

In every device, in state NetInterface Off, the Network Service and a timer $t_{RBD\_Condition}$ are started after detecting a valid EWU (see Table 4-6). Invalid EWUs are ignored. The timer $t_{RBD\_Condition}$ is stopped when the NetInterface Normal Operation state is reached. Ring Break Diagnosis (RBD) is not started.

If $t_{RBD\_Condition}$ times out, depending on the System Integrator, one of the following options applies:

- this is interpreted as RBD Phase 1 “Activation” in [1] and RBD Phase 2 “Diagnosis” of the Network Service is started
- the system test initiator performs the “MOST signal result, RBD result” system test (see 3.1.2)

Name	Min Value	Typ Value	Max Value	Description
$t_{RBD\_Condition}$	3000 ms	System Integrator specific	System Integrator specific	Timer for activation of Ring Break Diagnosis.

Table 3-4: Timing definitions for Ring Break Diagnosis activation

4 Physical Definitions

This chapter defines the impulses in detail, especially their voltage transitions and corresponding time tolerances. Furthermore, an example of the ECL circuit that is needed to send and receive these impulses is provided.

4.1 ECL Circuit

The ECL circuit utilizes a translation between the bidirectional ECL and microcontroller RX and TX pins. Furthermore, it interfaces the different voltage levels of the ECL and the peripherals of the MOST node.

The ECL circuit has to work as a bidirectional bus line driver. It has to be fully protected against short circuits of the ECL line to both ground and positive voltages ($V_{BATTERY}$, V_{BAT_ECU} or V_{SUPPLY}), and should be optimized for low power operation during MOST node sleep mode. It has to be ensured that the sender as well as all receivers uniformly detect the voltage transitions as defined in section 4.3.

Figure 4-1 shows a sample application of an ECL circuit.

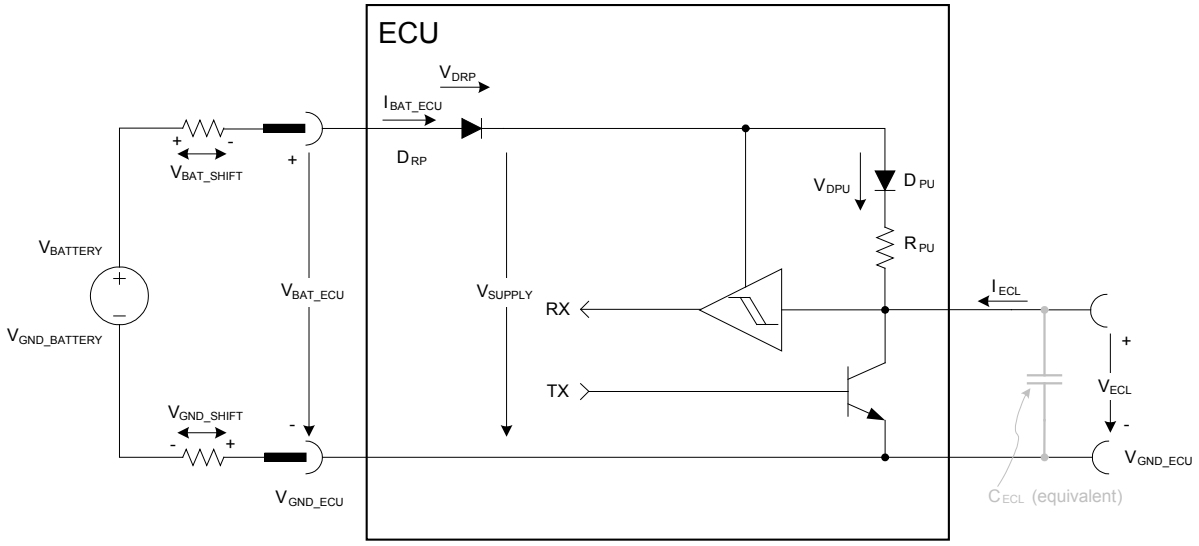

Figure 4-1: Sample application of the ECL circuit

The parameter values for any ECL circuit independent of its actual realization are given by Table 4-1. The actual values depend on System Integrator specific optimizations for low power operation, the kind of application of low-cost circuit products, and the system tolerance for high impedance.

The diode D_{RP} is used for a generic reverse voltage protection. The diode D_{PU} is used to limit excessive I_{ECL} currents in the case where $V_{ECL} > V_{SUPPLY}$

There has to be a limitation of I_{ECL} even in case of a short circuit of the ECL line to ground as well as to positive voltages. Note that Table 4-1 defines the maximum value of I_{ECL} when a short to power exists to avoid the damage of the ECL circuit. However, the System Integrator can require a lower value of I_{ECL} in order to avoid discharging the vehicle battery.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Bus Capacitance	C_{ECL}		—	—	20	nF
General Termination Resistor	R_{PU}	Pull-up resistor	20	30	60	kΩ
Optional Termination Resistor	R_{PU}	Pull-up resistor if node acts as system test initiator	0.9	1.0	1.1	kΩ
ECL Current	I_{ECL}	ECL Current limitation for driver active state	—	—	40	mA
ECL Current	I_{ECL}	ECL Current limitation in case of short circuit	—	—	200	mA

Table 4-1: Parameters of ECL

There are further requirements that are effective for any ECL circuit realization:

- Only a quiescent current I_{ECL} defined by the System Integrator is drawn when MOST node is disconnected from V_{BAT_ECU} ($V_{BAT\_ECU} = V_{GND\_ECU}$).
- I_{BAT_ECU} is minimized during sleep mode.

A pulse with length of less than 50 μs in a 50 ms window is invalid and is ignored by the ECL circuit or the software driver.

4.2 Voltage Drops

V_{SUPPLY} is defined as the supply voltage of the ECL transceiver of the appropriate ECU. V_{BAT_ECU} is defined as the voltage across the battery connector of the ECU. $V_{BATTERY}$ is defined as the voltage across the battery itself. Because of voltage drops over, e.g., diodes inside the ECU, and a general voltage drop across cable resistance, it is

$$V_{BATTERY} > V_{BAT\_ECU} > V_{SUPPLY} > V_{ECL}.$$

V_{GND_ECU} is defined as the ground voltage at the ECL transceiver of the appropriate ECU. $V_{GND\_BATTERY}$ is defined as the ground voltage at the battery itself. Because of general voltage shift, it is

$$V_{GND\_BATTERY} = 0V < V_{GND\_ECU}.$$

The limits for voltage shifts V_{BAT_SHIFT} and V_{GND_SHIFT} are defined in Table 4.1.

Name	Min	Typ	Max	Unit	Description
V_{BAT_SHIFT}	0 V	—	11.5%	V_{BAT_ECU}	$V_{BAT\_SHIFT} = V_{BATTERY} - V_{GND\_SHIFT} - V_{BAT\_ECU}$
V_{GND_SHIFT}	0 V	—	11.5%	V_{BAT_ECU}	$V_{GND\_SHIFT} = V_{GND\_ECU} - V_{GND\_BATTERY}$
V_{DRP}	—	—	1	V	Voltage drop across reverse protection diode D_{RP}
V_{DPU}	—	—	1	V	Voltage drop across pull-up diode D_{DPU}

Table 4-2: Voltage drop tolerances

4.3 Impulses

4.3.1 General

The voltage transitions of the impulses are defined by:

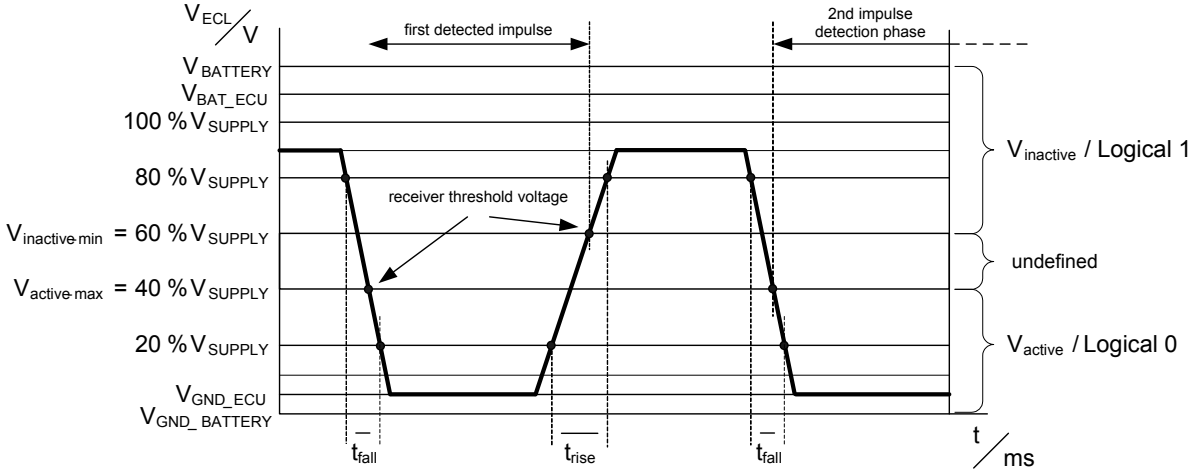

Figure 4-2: Physical sequence of a signal

The duration of the voltage transition from 80% V_{SUPPLY} to 20% V_{SUPPLY} is called t_{fall}. The duration of the voltage transition from 20% V_{SUPPLY} to 80% V_{SUPPLY} is called t_{rise}.
Note, that Figure 4-2 shows the physical signal sequence of a sending device. The signal sequence of a receiving device differs a little bit. Especially the low value of V_{ECL} does not equal V_{GND_ECU} because of different V_{GND_SHIFT}. The high values of V_{ECL} of the devices also differ because of different V_{BAT_SHIFT}.
The fall and rise times are allowed to be in the intervals as defined in Table 4.2.

Name	Min Value	Typ Value	Max Value	Description
t_{fall}	—	—	1 ms	Duration of the voltage transition from 80% V_{SUPPLY} to 20% V_{SUPPLY}
t_{rise}	—	—	2 ms	Duration of the voltage transition from 20% V_{SUPPLY} to 80% V_{SUPPLY}

Table 4-3: Timing definitions of fall time and rise time

If the voltage is in the range between $V_{inactive\text{-}min}$ and $V_{BATTERY}$ then on logical level a “1” is defined. In this case the voltage is also called $V_{inactive}$. If the voltage lies in an interval between $V_{GND\_BATTERY}$ and $V_{active\text{-}max}$ then on logical level a “0” is defined. In this case the voltage is also called V_{active}. If the voltage lies in an interval between $V_{active\text{-}max}$ and $V_{inactive\text{-}min}$, then the logical level is unpredictable. Table 4-4 shows the parameter values.

Name	Min Value	Typ Value	Max Value	Description
$V_{inactive}$ (Logical "1")	60% V_{SUPPLY} ($V_{inactive-min}$)	—	—	The logical state is also called "passive" or "inactive" because there is no sender that pulls the voltage to ground
V_{active} (Logical "0")	—	—	40% V_{SUPPLY} ($V_{active-max}$)	The logical state is also called "active" because there is a sender that pulls the voltage to ground

Table 4-4: Voltage definitions

Additionally, this section defines the durations of the logical impulses of Figure 3-1 together with their corresponding tolerances as follows.

Table 4-5 defines the durations and corresponding tolerances of time intervals for the sender side:

Name	Min Value	Typ Value	Max Value	Description
t_{TSI}	195 ms	200 ms	205 ms	System test start impulse (TSI)
t_{Pause}	95 ms	100 ms	105 ms	Pause after TSI or EWU
$t_{TestPause}$	95 ms	100 ms	105 ms	Waiting while testing
$t_{TestPause}$	6995 ms	7000 ms	7005 ms	Alternative while testing
t_{EWU}	75 ms	80 ms	85 ms	Wake-up signal for system wake-up

Table 4-5: Timing definitions on sender side

Table 4-6 defines the durations and corresponding tolerances of time intervals for the receiver side:

Name	Min Value	Typ Value	Max Value	Description
t_{TSI}	170ms	200 ms	230 ms	Impulse for a system test
t_{Pause}	70 ms	100 ms	130 ms	Pause after TSI or EWU
$t_{TestPause}$	70 ms	100 ms	130 ms	Waiting while testing
$t_{TestPause}$	6970 ms	7000 ms	7030 ms	Alternative waiting
t_{EWU}	50 ms	80 ms	110 ms	Impulse for system wake-up with one wake-up initiator in the system
t_{EWU}	50 ms	80 ms	135 ms	Impulse for system wake-up with more than one wake-up initiator in the system

Table 4-6: Timing definitions on receiver side

Note: *The value of $t_{TestPause}$ in Table 4-5 and Table 4-6 depends on the kind of test as defined in section 3.1.*

Depending on the system definitions, there are additionally the following time values (that are required on sender side as well as on receiver side) as defined in Table 4-7:

Name	Min Value	Typ Value	Max Value	Description
$t_{StartUp}$	100 ms	—	10000 ms	Startup impulse

Table 4-7: System definition dependent time values

4.3.2 Wake-up Impulses

Two types of wake-up mechanisms are considered:

1. Wake-up with single EWU (electrical wake-up)

With a single EWU, the wake-up has to be triggered by the falling edge of the first impulse.

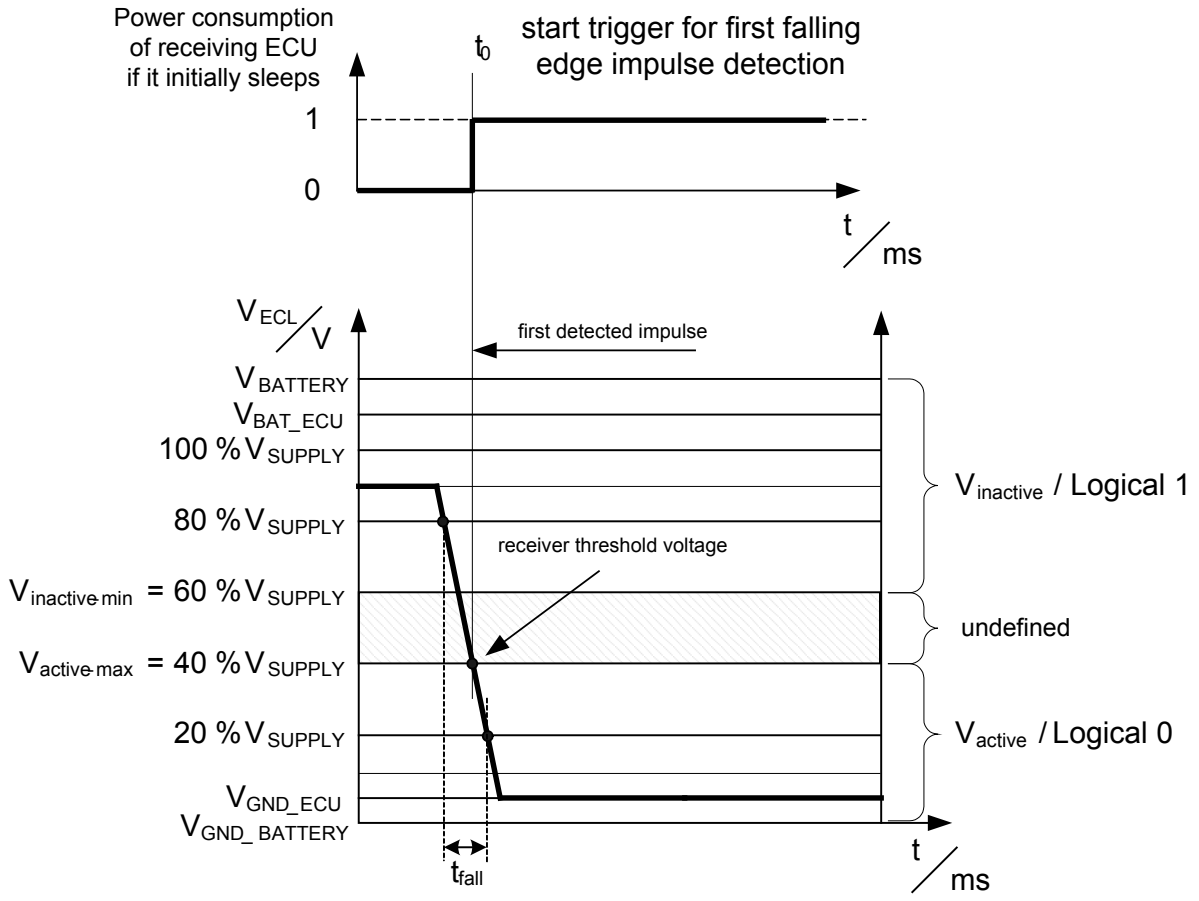

Figure 4-3: Wake-up with single EWU

2. Wake-up with multiple EWUs

The wake-up with multiple EWUs relies on the repetition of the wake-up impulse; it does not matter which impulse the receiving ECU detects: the first or any of the subsequent impulses.

4.4 Triggers

Participants have to trigger several impulses and therefore have to regard some trigger intervals. Figure 4-4 shows the required triggers.

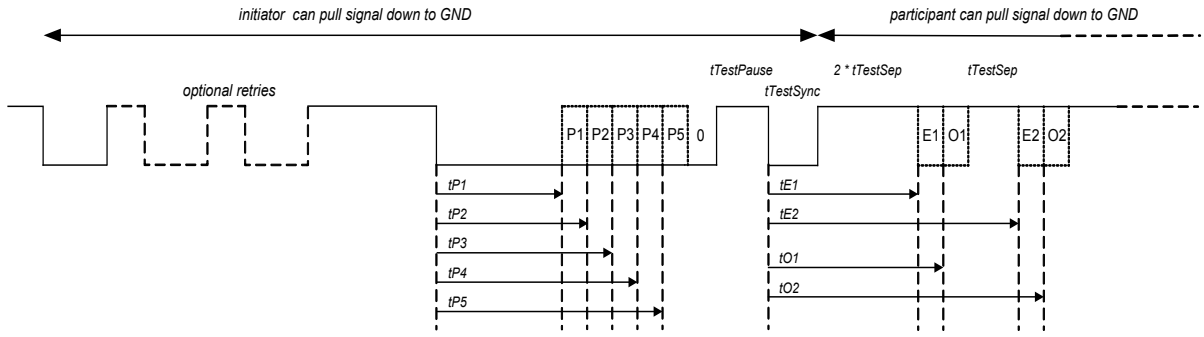

Figure 4-4: Triggers

A participant must trigger the parameters P1 to P5. This means that the initiator begins to send the parameter Pn at the time

$t_{Pn} = t_{TestStart} + (n - 1) * 50\ ms \pm 5\ ms$,

and the participants begin to receive this parameter at the time

$t_{Pn} = t_{TestStart} + (n - 1) * 50\ ms \pm 15\ ms$.

Note that both formulas above define additionally the allowed tolerances.

An initiator must trigger the results E1 to Em_C and O1 to Om_C. This means that participant *n* begins to send the parameter *En* at the time

$t_{En} = t_{TestSync} + n * 200\ ms \pm 5\ ms$,

and the initiator begins to receive this parameter at the time

$t_{En} = t_{TestSync} + n * 200\ ms \pm 15\ ms$.

Furthermore, that participant *n* begins to send parameter *On* at the time

$t_{On} = t_{TestSync} + n * 200\ ms + 50\ ms \pm 5\ ms$,

and the initiator begins to receive this parameter at the time

$t_{On} = t_{TestSync} + n * 200\ ms + 50\ ms \pm 15\ ms$.

Again, the formulas above define additionally the allowed tolerances.

The separation times $t_{TestSep}$, respectively exist to avoid undesirable interferences of two impulses sent by different participants because of tolerance effects. This separation time is already considered in the formulas above by the 200 ms value.

The formulas lead to impulse durations on sender and receiver side. They result in values as follows:

Name	Min Value	Typ Value	Max Value	Description
$t_{TestStart}$	—	250 ms	—	Duration of starting the test
$t_{TestParam}$	—	50 ms	—	Duration of a test parameter impulse
$t_{TestSync}$	—	100 ms	—	Duration of the sync phase for triggering
$t_{TestSep}$	—	100 ms	—	Duration of separation time to avoid undesirable interferences of two impulses generated by different instances: Initiator / participant n or participant n / participant n+1
$t_{TestResult}$	—	50 ms	—	Duration of a test result impulse (*En* or *On*)

Table 4-8: Times for trigger-dependent impulses

5 Software Design [Informative]

The following chapter defines a software design for the ECL realization.

5.1 Architecture

The software architecture is shown in Figure 5-1.

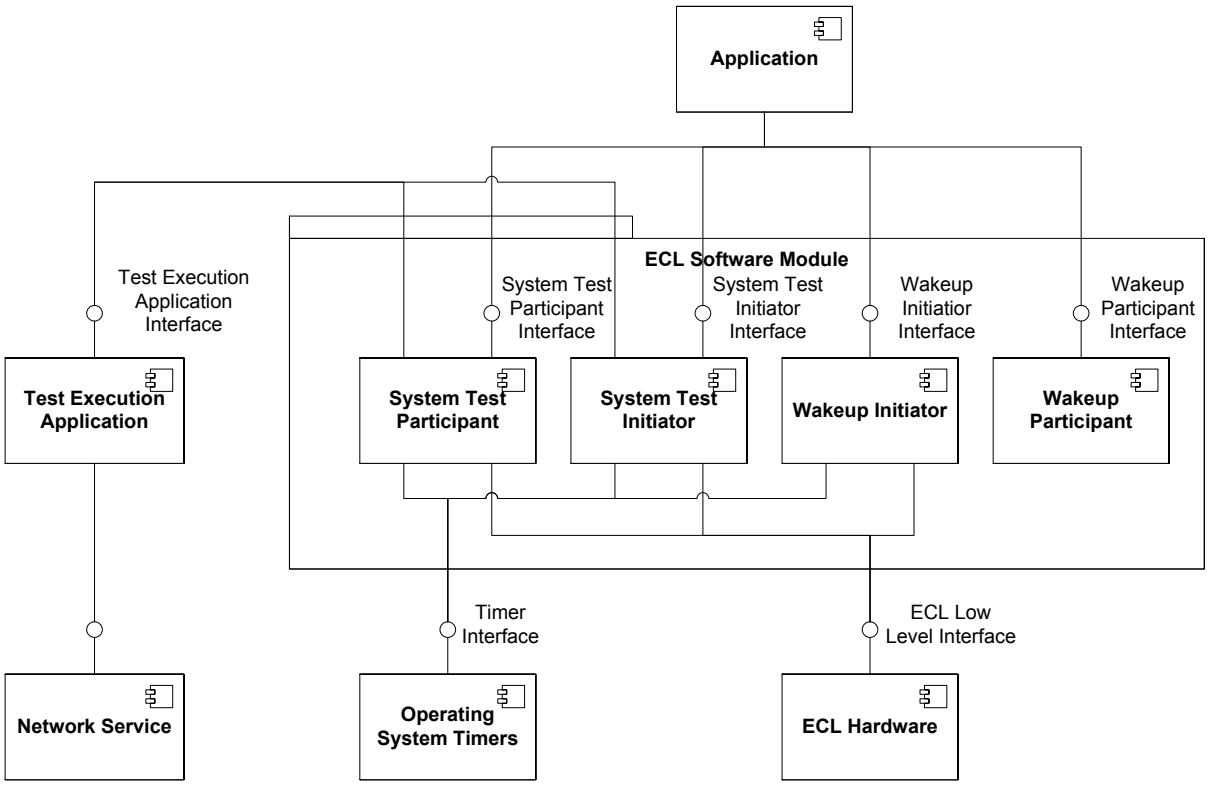

Figure 5-1: General software architecture

The ECL Software Module consists of the following components:

- System Test Participant
- System Test Initiator
- Wakeup Initiator
- Wakeup Participant

The ECL Software Module has the following interfaces to other components:

- System Test Participant Interface
- System Test Initiator Interface
- Wakeup Initiator Interface
- Wakeup Participant Interface
- Test Execution Application Interface
- Timer Interface
- ECL Low Level Interface

There are several surrounding components which are using and being used by the ECL Software Module. The Application component stands for all components which are actively “using” the ECL Software Module, for example, diagnostic services, FBlock Enhanced Testability and power management. The Test Execution Application component has the task to execute the system test, for example, RBD via MOST Network Service according to the parameters P1 – P5 and deliver the results back to the ECL Software Module. The Operating System Timers component is used by the ECL Software Module to have an exact time base for sending and receiving ECL sequences. The ECL Hardware component is used by the ECL Software Module to send and receive the signals of the ECL.

5.2 ECL Software Module

The System Test components are the core components which are implementing the system test according to section 3.1. The Wakeup components are responsible for the processing of wake-up requests according to section 3.2.

Every system implementing the ECL Specification consists of one device that serves as system test initiator - all other devices are participants.

5.2.1 System Test Initiator Component

The System Test Initiator component implements the System Test Initiator state machine (refer to section 5.4.1).

The **System Test Initiator Interface** of the System Test Initiator component is used from

- the Application module to start / stop the System Test Initiator module
- the Application module to start system tests and receive results
- the Application module to be notified about the current state of the ECL state machine, for example, started, finished with error, or finished ok

The System Test Initiator component uses the **Timer Interface** of the Operating System Timers component to:

- get a time base for sending / receiving ECL signals

The System Test Initiator component uses the **ECL Low Level Interface** of the ECL Hardware component to

- send / receive ECL signals

The System Test Initiator component uses the **Test Execution Application Interface** to

- trigger system tests
- receive the results of the system tests

5.2.2 System Test Participant Component

The System Test Participant component implements the System Test Participant state machine (refer to section 5.4.2).

The **System Test Participant Interface** of the System Test Participant component is used from

- the Application module to start / stop the System Test Participant module
- the Application module to be notified about TSIs
- the Application module to be notified about the current state of the ECL state machine, for example, started, finished with error, or finished ok

The System Test Participant component uses the **Timer Interface** of the Operating System Timers component to

- get a time base for sending / receiving ECL signals

The System Test Participant component uses the **ECL Low Level Interface** of the ECL Hardware component to

- send / receive ECL signals
- activate / deactivate interrupts

The System Test Participant component uses the **Test Execution Application Interface** of the Test Execution Application to

- trigger system tests
- receive the results of the system tests

5.2.3 Wakeup Initiator Component

The Wakeup Initiator component implements the sending of the electrical wake-up impulse on the ECL (refer section 3.2).

The **Wakeup Initiator Interface** of the Wakeup Initiator module is used from

- the Application component to start / stop the Wakeup Initiator component
- the Application component to trigger an ECL wake-up impulse (electrical wake-up)

The Wakeup Initiator component uses the **Timer Interface** of the Operation System Timers component to

- get a time base for sending / receiving ECL signals

The Wakeup Initiator component uses the **ECL Low Level Interface** of the ECL Hardware to

- send ECL signals

5.2.4 Wakeup Participant Component

The Wakeup Participant component implements the reception of the electrical wake-up impulse.

Note: *Due to the fact that the electrical wake-up impulse must also be detected and validated out of ECU sleep mode, it will most likely be implemented in hardware. Because of that, the Software Design chapter will not deal with it.*

5.3 Interfaces

The following section describes interfaces between the involved software components.

Note: *The interface definitions show a possible but not mandatory way of software implementation. They may be not comprehensive but give a good idea of the necessary functions.*

5.3.1 Timer Interface

A timer module has to provide an interface to a timer that must not deviate from the timing tolerances defined in Table 4-5, Table 4-6, Table 4-7 and section 4.3.

The timer command interface should support the following functions:

`<t> = T_start(<duration>)`	Starts timer <t> with <duration> duration.
`T_stop(<t>)`	Stops timer <t>.

The timer callback interface should support the following function:

`Tcb_timeout(<t>)`	Is called when a timer expires. <t> indicates which timer timed out.

5.3.2 Test Execution Application Interface

This interface provides the API to the Test Execution Application. The Test Execution Application is responsible for executing the actual test requested by ECL parameter P1 – P5.

Note: *The Software Integrator is responsible for ensuring that function calls of the test execution application are not executed in the same task context as the ECL module.*

The Test Execution Application command interface should support the following function:

`TEA_result(<result>)`	In case the test execution application chooses to run a test when called by TEAcb_start() this function can be used to report the <result> back to the system test module. The <result> is sent later on as bit *On* in the ECUs time slot.

The Test Execution Application callback interface should support the following function:

`TEAcb_start(<test>)`	Is called at the end of the parameter sequence to indicate the Test Execution Application which type of test <test> was requested by the initiator, for example, if <test> == RBD, the MOST Network Service must be triggered to run the RBD.

5.3.3 ECL Low Level Interface

The ECL Low Level Interface provides an interface to handle the Electrical Control Line voltage shift and the configuration of the interrupt.

The ECL Low Level command interface should support the following functions:

`ECL_irqConfig`	Configures the interrupt for the ECL.
`ECL_irqEnable`	Enables the interrupt for the ECL.
`ECL_irqDisable`	Disables the interrupt for the ECL.
`ECL_setSignal(<level>)`	Sets the signal level <level> of the ECL.
`<level> = ECL_readSignal`	Reads the signal level <level> of the ECL.

The ECL Low Level Drivercallback interface should support the following function:

`ECLcb_irq`	Is called when an ECL level change is detected.

5.3.4 System Test Initiator Interface

The System Test Initiator Interface provides ways to start the ECL system test, forward test result to the diagnosis application, report to the diagnosis application when the ECL system test was started or stopped.

The System Test Initiator command interface should support the following functions:

`STI_start`	Starts the ECL Software Module and moves the internal state machine to state idle.
`STI_end`	Stops the ECL Software Module.
`STI_startSystemTest (<retries>, <startupTime>, <test>, <testPauseTime>, <mc>)`	Starts the selected system test <test>. <retries> defines the number of TSI retries; for example, when <retries> = 0, the following start sequence is sent: t_{TSI} - $t_{StartUp}$. When <retries> = 1, the start sequence is: t_{TSI} - t_{Pause} - t_{TSI} - $t_{StartUp}$. <startupTime> defines the time $t_{StartUp}$ before the parameter sequence starts. Timer <testPauseTime> defines the time the initiator has to wait for participants for their test execution. Counter <mc> defines the numbers of slots used by participants during the result sequence. <mc> is used by the System Test Initiator component to calculate $t_{TestRestart}$.
`<result> = STI_getSystemTestResult`	Requests the result <result> of the system test. The result is only valid if STIcb_status delivered status <status> == FinishedOk. ***Note:*** *<result> should be initialized with NoResultAvailable and also set to NoResultAvailable after leaving state "Idle".*

The System Test Initiator callback interface should support the following function:

`STIcb_status(<status>)`	Is called when the ECL Software Module has a new status <status> for the application.

5.3.5 System Test Participant Interface

The System Test Participant Interface provides ways to start and stop the System Test Participant component.

The System Test Participant command interface should support the following functions:

`STP_start`	Starts the ECL Software Module and moves the internal state machine to state Idle.
`STP_end`	Stops the ECL Software Module.

The System Test Initiator callback interface should support the following function:

`STPcb_status(<status>)`	Is called when the ECL Software Module has a new status <status> for the application.

5.3.6 Wakeup Initiator Interface

The Wakeup Initiator Interface provides ways to wake up the ECL participants.

The Wakeup Initiator command interface should support the following functions:

`WI_start`	Starts the ECL Software Module.
`WI_end`	Stops the ECL Software Module.
`WI_wakeup(<retries>)`	Starts the electrical wake-up. <retries> defines the number of wake-up impulse retries; for example, when <retries> = 0, the wake-up only consists of t_{EWU}. When <retries> = 1, the wake-up sequence is: t_{EWU} - t_{Pause} - t_{EWU}.

5.3.7 Wakeup Participant Interface

The Wakeup Participant Interface provides ways to inform the power management application about a valid electrical wake-up event.

Note: *Due to the fact that the electrical wake-up impulse must also be detected and validated out of ECU sleep mode it will most likely be implemented in hardware. Because of that the whole Software Design chapter will not deal with it.*

5.4 State Machines

This section shows state machines for system test initiator module and system test participant module. The state machines for the electrical wake-up module are not modelled because they are trivial.

5.4.1 System Test Initiator State Machine

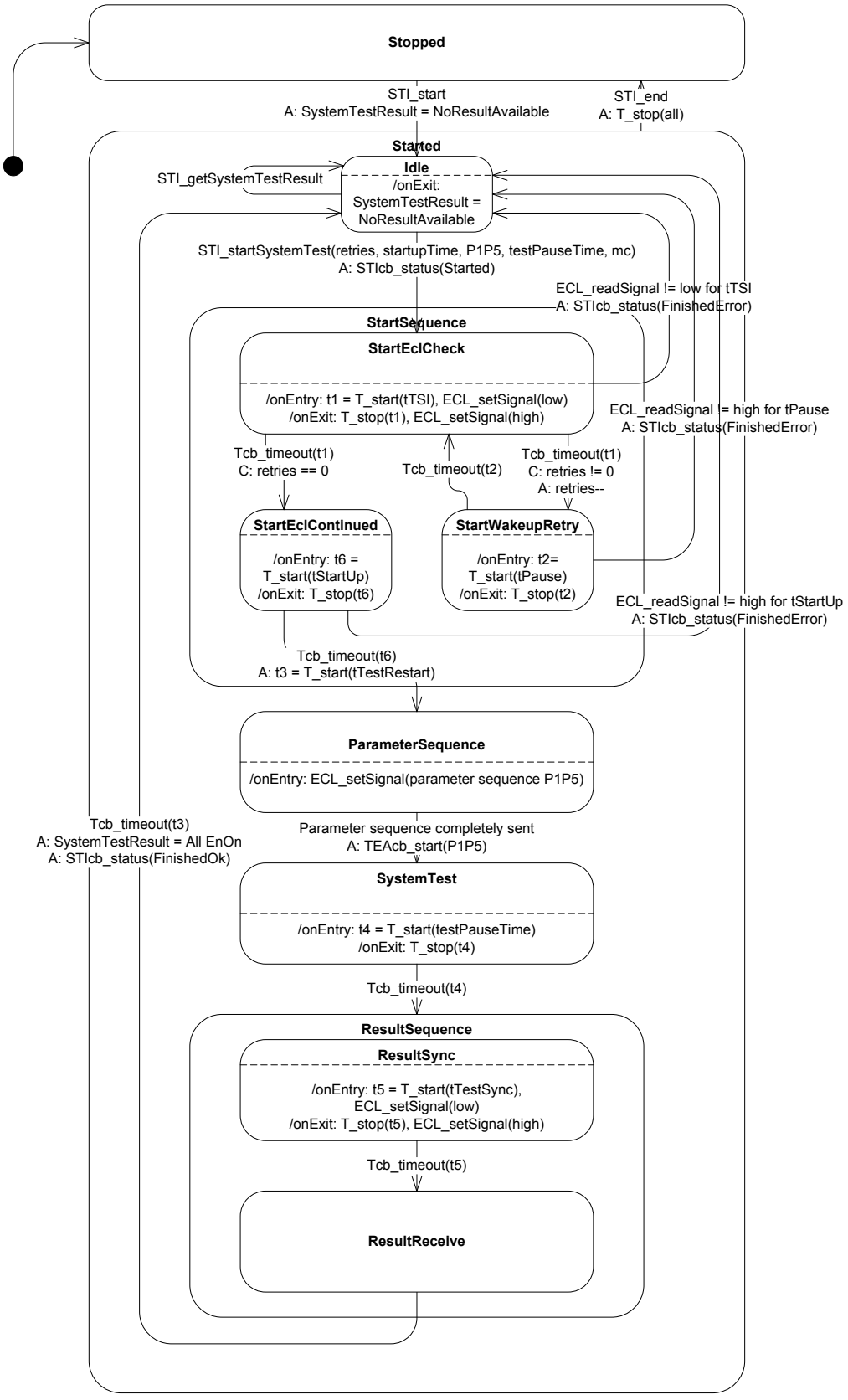

Figure 5-2: System test initiator state machine

Figure 5-2 illustrates a state machine that a system test initiator implements. Table 5-1 describes the states used in Figure 5-2.

State	Description
Stopped	This is the initial state after startup. The state changes to “Idle” when function STI_start is called. STI_start initializes SystemTestResult with NoResultAvailable and reserves system resources.
Started	This is the parent state for all states except “Stopped”. The state changes to “Stopped” when function STI_end is called. STI_end stops all timers and frees system resources.
Idle	In this state, the initiator waits for a trigger to start the system test. On exit, SystemTestResult is set to NoResultAvailable. The state changes to “StartEclCheck” if function STI_startSystemTest is called. The number of TSI retries, the startup time $t_{StartUp}$, the parameters P1 – P5 and the duration of $t_{TestPause}$ is communicated via parameters. The application is notified about the start. The state changes to itself if function STI_getSystemTestResult is called. “Idle” is the only state where calling STI_getSystemTestResult delivers a valid result.
StartSequence	This is the parent state for “StartEclCheck”, “StartEclContinued”, and “WakeupRetry”. ***Note:*** *This state is for optical clustering only.*
StartEclCheck	In this state, the TSI is sent by the initiator. The initiator also verifies the correct signal level during t_{TSI}. On entry, timer t1 (t_{TSI}) is started and a low signal is sent on the ECL. On exit, timer t1 is stopped and a high signal is sent on the ECL. The state changes to “StartEclContinued” on timeout of timer t1 when the number of retries is 0. The state changes to “WakeupRetry” on timeout of timer t1 when the number of retries is greater than 0. On transition the retry counter is reduced by one. The state changes to “Idle” if the initiator reads a different ECL signal than it is sending. On transition, the ECL Software Module calls STIcb_status with an error value to inform the upper layer about the detected problem.
StartWakeupRetry	In this state, the initiator waits for timeout of timer t2 (t_{Pause}). On entry, timer t2 is started. On exit, timer t2 is stopped. The state changes to “StartEclCheck” on timeout of timer t2. The state changes to “Idle” if the initiator reads a different ECL signal than it is sending. On transition, the ECL Software Module calls STIcb_status with an error value to inform the upper layer about the detected problem.
StartEclContinued	In this state, the rest of the start sequence is sent by the initiator, that is, $t_{StartUp}$. On entry, timer t6 is started. On exit, timer t6 is stopped. The state changes to “ParameterSequence” if timer t6 has expired. At this point, timer t3 ($t_{TestRestart}$) is started.

State	Description
	The state changes to “Idle” if the initiator reads a different ECL signal than it is sending. On transition, the ECL Software Module calls STIcb_status with an error value to inform the upper layer about the detected problem.
ParameterSequence	In this state, the parameter sequence is sent by the initiator, that is, $t_{TestStart}$, 6 x $t_{TestParam}$. The state changes to “SystemTest” if the parameter sequence has been sent completely (at the end of P_{Sync}). On this transition, the ECL Software Module notifies the Test Execution Application to run a system test. Which system test is requested is handed over via the parameters P1 to P5.
SystemTest	In this state, the initiator waits for the participants finishing their requested system test. On entry, timer t4 ($t_{TestPause}$) is started. The duration of $t_{TestPause}$ is set by testPauseTime. On exit, timer t4 is stopped. The state changes to “ResultSync” on timeout of timer t4.
ResultSequence	This is the parent state for “ResultSync” and “ResultReceive”. ***Note:*** *This state is for optical clustering only.*
ResultSync	In this state, the initiator sends the synchronization signal on the ECL. On entry, timer t5 ($t_{TestSync}$) is started and the ECL level is set to low. On exit, timer t5 is stopped and the ECL is released. The state changes to “ResultReceive” when timer t5 expires. ***Note:*** *This is last state where the initiator actively drives the ECL.*
ResultReceive	In this state, the initiator waits for the results of the participants. The state changes to “Idle” on timeout of timer t3 ($t_{TestRestart}$). The received *En*/*On* results of all participants are stored in SystemTestResult and the application gets notified about the successful completion.

Table 5-1: ECL initiator state description

5.4.2 System Test Participant State Machine

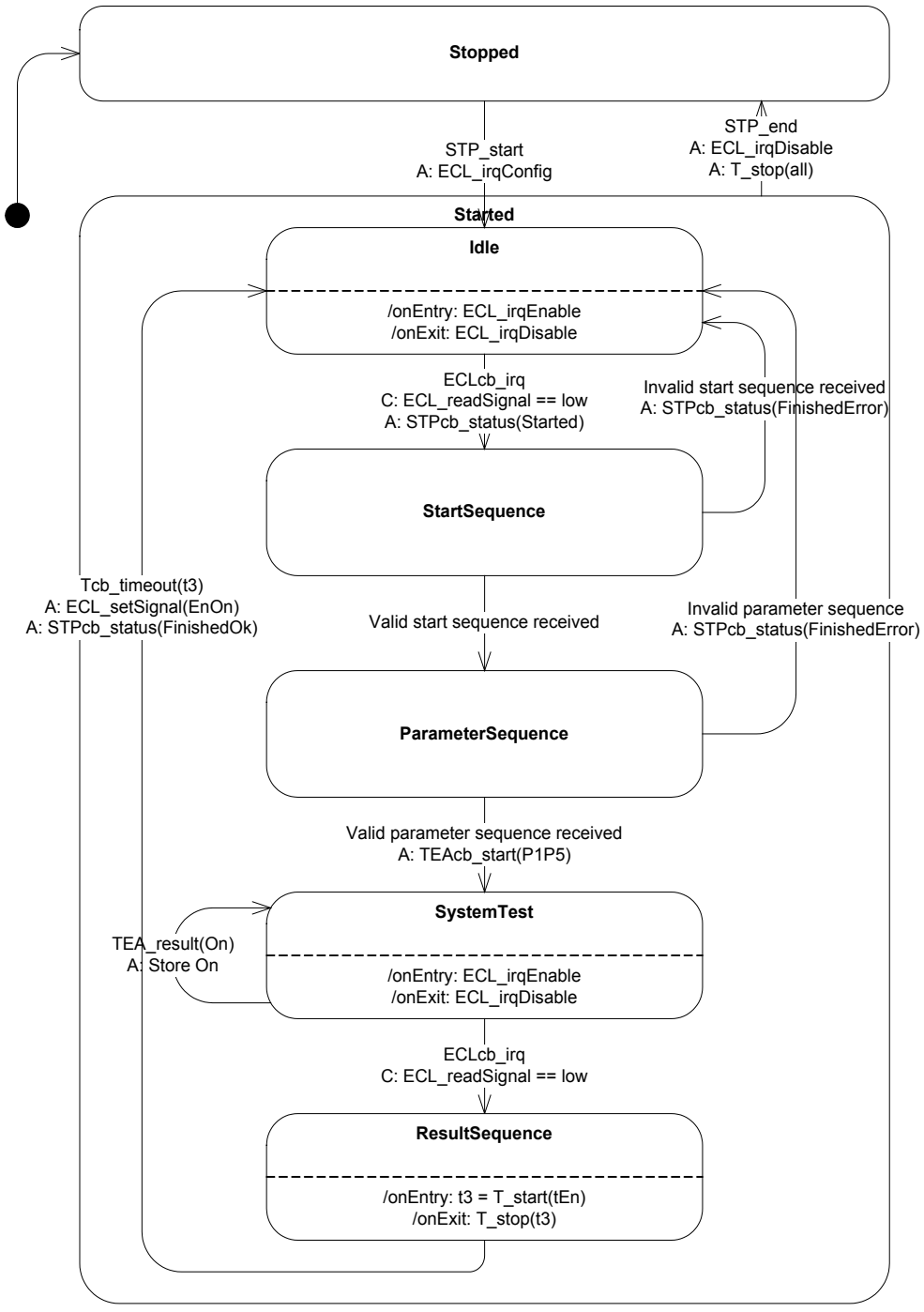

Figure 5-3: System Test participant state machine

Figure 5-3 illustrates a state machine that a system test participant implements. Table 5-2 describes the state used in Figure 5-3.

State	Description
Stopped	This is the initial state after startup. The state changes to “Idle” when function STP_start is called. STP_start configures the interrupt and reserves system resources.
Started	This is the parent state for all states except “Stopped”. The state changes to “Stopped” when function STP_end is called. STP_end disables the interrupt, stops all timers, and frees system resources.
Idle	In this state, the participant waits for a TSI. On entry, the ECL interrupt is enabled in order to get notified on ECL level changes. On exit, the interrupt is disabled because up to the parameter sequence all ECL changes are polled. The state changes to “StartSequence” if function ECLcb_irq is called and the ECL level is low. The application is notified about the start.
StartSequence	In this state, the participant waits for the start sequence to be sent by the initiator. The state changes to “ParameterSequence” when a valid start sequence is detected. The start sequence consists of t_{TSI} and its optional retries and also includes $t_{StartUp}$. The state changes to “Idle” in then case of an invalid start sequence, for example, if the timing of t_{TSI} is not correct. The application is notified about the ECL error.
ParameterSequence	In this state, the participant waits for the parameter sequence to be sent by the initiator. The state changes to “SystemTest” when a valid start sequence is detected. In this case, the information about a requested system test is communicated to the test execution application by calling TEAcb_start. The parameters of TEAcb_start (P1 – P5) signal the type of system test which must be executed. The state changes to “Idle” in the case of an invalid parameter sequence, for example, if the timing of $t_{TestStart}$ is not correct. The application is notified about the ECL error.
SystemTest	In this state, the participant is executing the system test. On entry, the interrupt is enabled to get notified about the beginning of the result sequence. On exit, the interrupt is disabled. The state changes to “ResultSequence” if ECLcb_irq is called and the read in ECL signal is low. The state transitions to itself if function TEA_result is called. The result On is stored for later sending.
ResultSequence	In this state, the participant waits for its result time slot and then reports the *En* and *On* values to the initiator. On entry, timer t3 (t_{En}) is started. On exit, timer t3 is stopped. The state changes to “Idle” when timer t3 expires. The participant transmits *En*/*On* to the initiator and notifies the application about successful execution of result transmission.

Table 5-2: ECL participant state description

6 Appendix A: List of Figures

Figure 3-1: Sequence of the system test 10
Figure 3-2: Sequence of the electrical wake up 15
Figure 4-1: Sample application of the ECL circuit 16
Figure 4-2: Physical sequence of a signal 19
Figure 4-3: Wake-up without retries 21
Figure 4-4: Triggers 22
Figure 5-1: General software architecture 24
Figure 5-2: System test initiator state machine 30
Figure 5-3: System Test participant state machine 33

7 Appendix B: List of Tables

Table 1-1: Definitions 8
Table 3-1: Electrical and optical result values for standard tests 12
Table 3-2: Reaction on malfunction, based on timing range 14
Table 4-1: Parameters of ECL 17
Table 4-2: Voltage drop tolerances 18
Table 4-3: Timing definitions of fall time and rise time 19
Table 4-4: Voltage definitions 20
Table 4-5: Timing definitions on sender side 20
Table 4-6: Timing definitions on receiver side 20
Table 4-7: System definition dependent time values 20
Table 4-8: Times for trigger-dependent impulses 23
Table 5-1: ECL initiator state description 32
Table 5-2: ECL participant state description 34

Notes: