Individual duty cycles for driving high beam and low beam with TLD5542-1QV

Firmware description

About this document

Scope and purpose

This document describes the firmware SW_HBLBIndividualDutyCycle_V1.0 used to drive 2 LED light functions, low beam (LB) and high beam (HB) with individual duty cycles by means of a single TLD5542-1QV. The proposed example firmware is developed for the TLD5542-1_MULTI_EVAL evalboard which includes a TLE9842-2QX MCU platform. The code is based on LITIX™ Power Flex SPI LLD [2] and it can be easily ported to other 32-bit MCUs. System architecture and firmware can be adapted to drive several numbers of light functions (e.g. 3 or 4). This implementation has HB duty cycle ≤ LB duty cycle to enable further BOM saving. In addition, the application is easily portable to fully independent duty cycle solutions.

Intended audience

Software Engineer, Hardware Engineer

Table of contents

About this document **1**
Table of contents **1**
1 System architecture **2**
1.1 Drive bypass switches and TLD5542-1QV during load transitions 4
1.2 Schedule and prepare repetitive load transitions 6
1.3 Safety mechanism 10
2 Software architecture **13**
2.1 FSM states 14
2.2 FSM events 14
2.3 FSM actions 15
2.4 Software modules 17
2.5 Interface APIs 18
3 Requirement specification **19**
3.1 Hardware requirements 19
3.2 Software requirements 19
3.3 Porting firmware 21
4 List of abbreviations **24**
5 List of references **25**
Revision history **26**

1 System architecture

The system is composed of a buck-boost power stage (TLD5542-1QV) which drives two light functions (LB and HB) connected in series. HB can be individually and quickly bypassed to produce efficient and fast load transitions (steps). Repetitive transitions at PWM dimming frequency such as 200 Hz) are used to simultaneously drive LB and HB with individual duty cycles where HB duty cycle is always smaller or equal to LB duty cycle. In this architecture an MCU is needed to:

- Drive bypass switches and TLD5542-1QV during load transitions
- Schedule and prepare repetitive load transitions
- Implement safety mechanism (e.g. battery protection, load diagnosis)

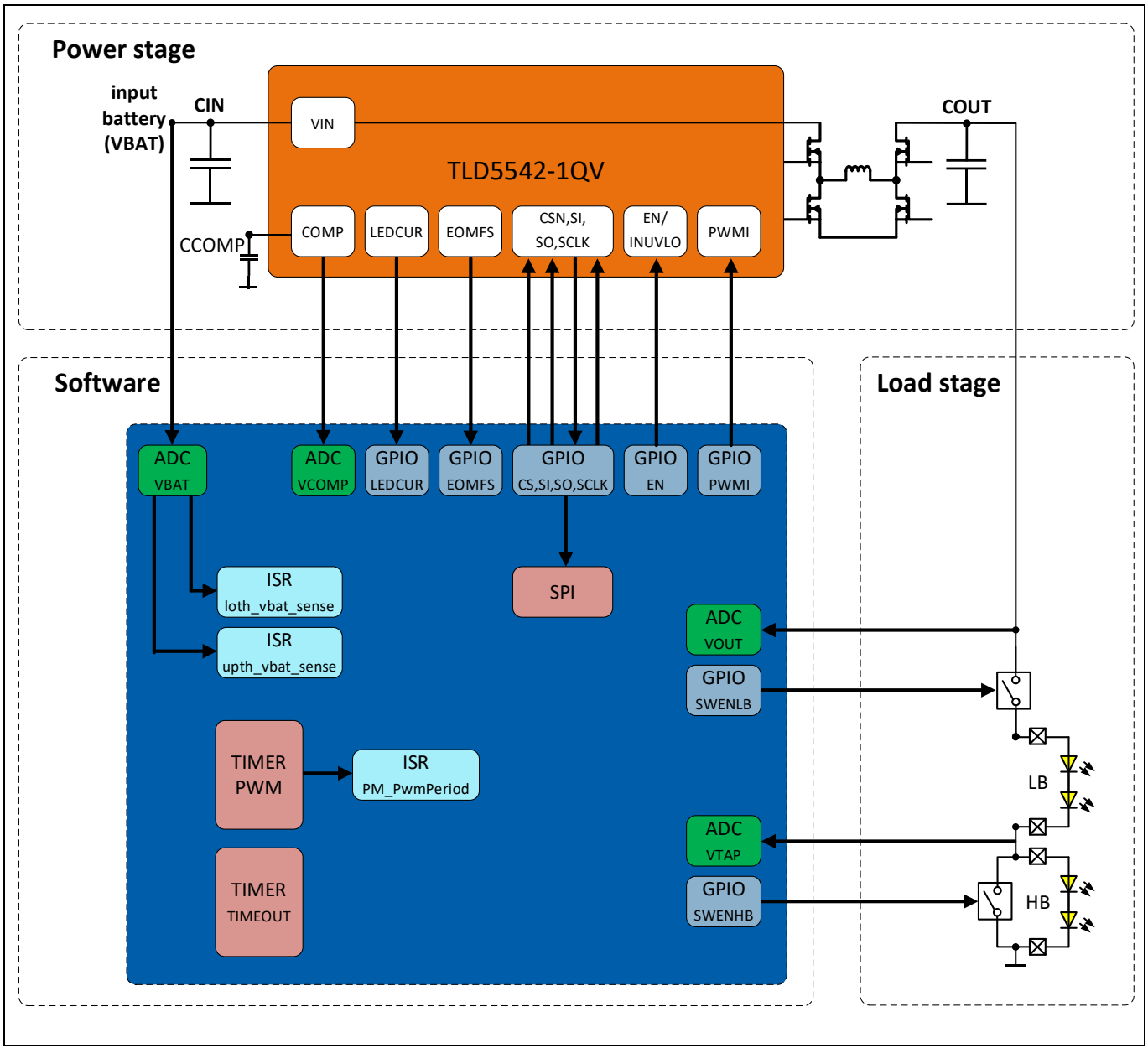


Figure 1 System architecture

System architecture

In this system architecture each PWM period can be described by an ordered sequence of PWM phases. Each phase is defined by the list of all functions activated light function combination “LFC” and their duration (duty cycle). Periodically repeating this sequence allows LB and HB to be powered by individual duty cycles. Load transitions (step-up/step-down) when changing phase have to be carefully scheduled and synchronized by the MCU to produce the desired light duty cycle.

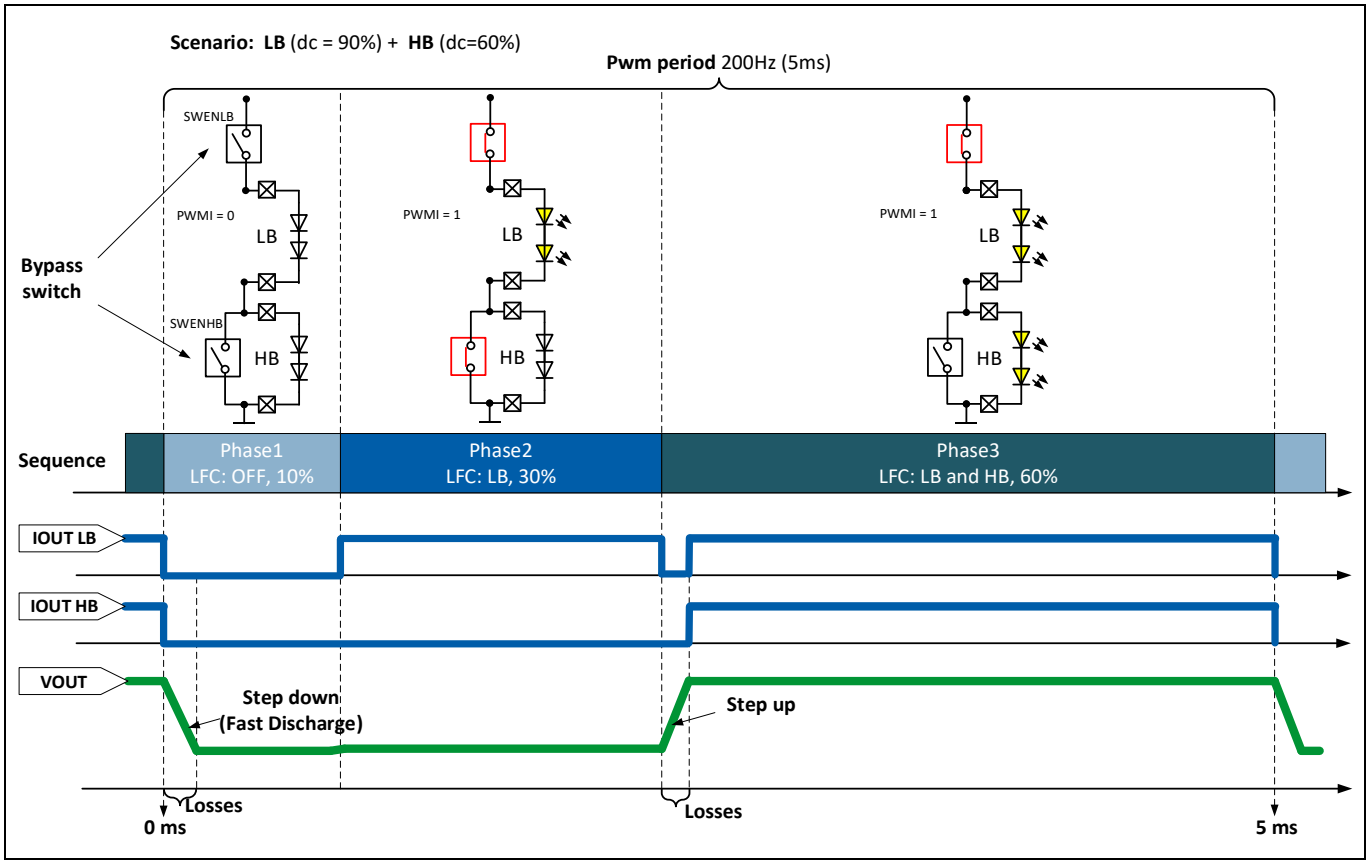


Figure 2 PWM period description

Table 1 Key concepts

Definition	Description
Light function combination (LFC)	The sum of multiple light functions (series connection) which are active at the same time
PWM period	period of PWM activities (i.e. 200 Hz → 5 ms)
Scenario	List of all light functions present in a PWM period and requested duty cycle
Phase	PWM period portion specified by the LFC and phase duty cycle
Sequence	Ordered list of phases which would be activated during the PWM period
Bypass switch	Switch (usually MOSFET) used to short circuit a light function
Fast discharge (FD-MFS)	TLD5542-1QV feature to control a fast discharge of the output capacitor to a target fraction of the initial output voltage
Losses	A short interruption of the output current observed during the LFC transitions, until output voltage is stable and the device control loop is able to provide the target output current to the new LFC

1.1 Drive bypass switches and TLD5542-1QV during load transitions

MCU drives TLD5542-1QV and bypass switches during load steps. There are two types of load steps:

- Total LED count increases (step-up), only bypass switch reconfiguration. No action needed from TLD5542-1QV
- Total LED count decreases (step-down), TLD5542-1QV fast discharge needed to avoid current spike in the LEDs that have to remain on

1.1.1 Step-up load transitions

When the number of connected LEDs increases the final output voltage V_{OUTF} and compensation voltage V_{COMPF} are higher than initial V_{OUTL} and V_{COMPL}. The following steps describe the step-up procedure:

- V_{OUTL} and V_{COMPL} are sampled (ADCs) and stored just before adjusting bypass switches. These values are used in the upcoming step-down load transitions as target for TLD5542-1QV fast discharge procedure.
- Bypass switch is reconfigured so as to enable the new LFC
- Output current flow is temporary interrupted (losses), until TLD5542-1QV reaches the new operation point (V_{OUTF} , V_{COMPF}) and current flows through the new load again. This time measured simply monitoring TLD5542-1QV LEDCUR pin status which is low when there is no current flowing into the load (See LEDCUR behavior in datasheet [1]). MCU can measure losses by polling LEDCUR pin status until becomes high (LEDCUR = 1). Polling must be delayed by a certain BLANKTIME to avoid wrong LEDCUR = 1 detection due to bypass switch closure delay time

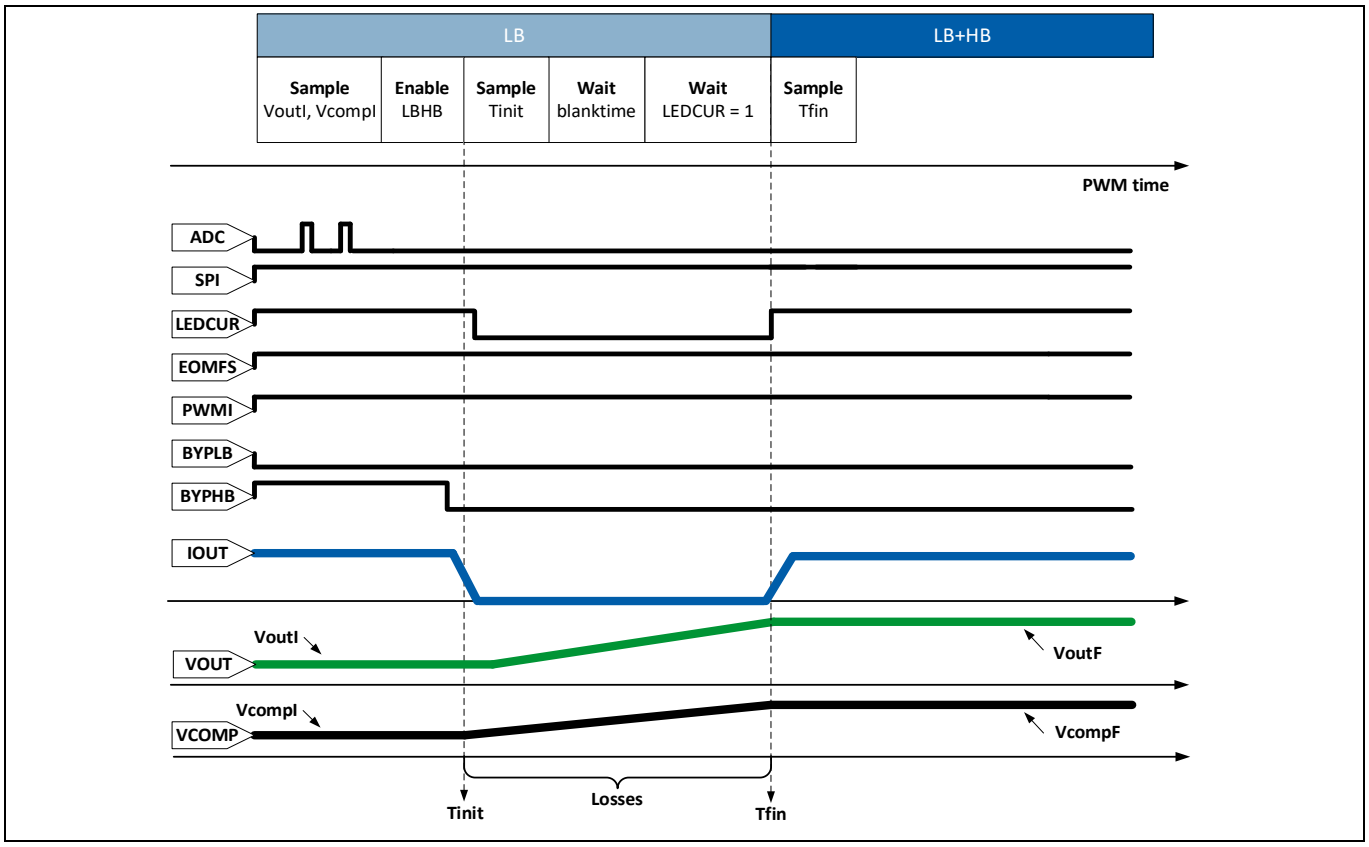


Figure 3 LED step up (LB to LB+HB) timing diagram

1.1.2 Step-down load transitions

When the number of connected LEDs decreases, V_{OUTF} and V_{COMPF} are smaller than V_{OUTL} and V_{COMPL}. Both C_{OUT} and C_{COMP} have to be carefully discharged before adjusting bypass switches. TLD5542-1QV offers fast discharge procedure (See Chapter 6.5 of TLD5542-1QV [1]). C_{OUT} is discharged to a fraction of (V_{OUTL}*NUM/DEN) of its value at the moment when the routine is initiated while C_{COMP} is discharged by a constant current for a programmable time “MFSDLY”. The following steps describe the step-down procedure:

- The MCU calculates and communicates (SPI) the best combination of “NUM”, “DEN” and “MFSDLY” to discharge C_{OUT} and C_{COMP} to the target operation point (V_{OUTF}, V_{COMPF}). This target is known because measured in the previous PWM cycles. Slightly over-discharging C_{COMP} is useful to guarantee smooth current waveforms and compensate unexpected operation point variations (e.g. battery variations)
- Fast discharge operation starts with the SPI frame which contains “NUM” and “SOMFS” command. MCU implements strategies “wait trigger” to synchronize and schedule this SPI transmission in order to obtain accurate light functions duty cycle
- TLD5542-1QV EOMFS is low during fast discharge execution. MCU polls “EOMFS” until it becomes high (end of fast discharge) and thereafter, reconfigures bypass switches. The output current flow is temporary interrupted (losses) during fast discharge. This time can be measured simply by monitoring TLD5542-1QV EOMFS pin status (see EOMFS behavior [1])

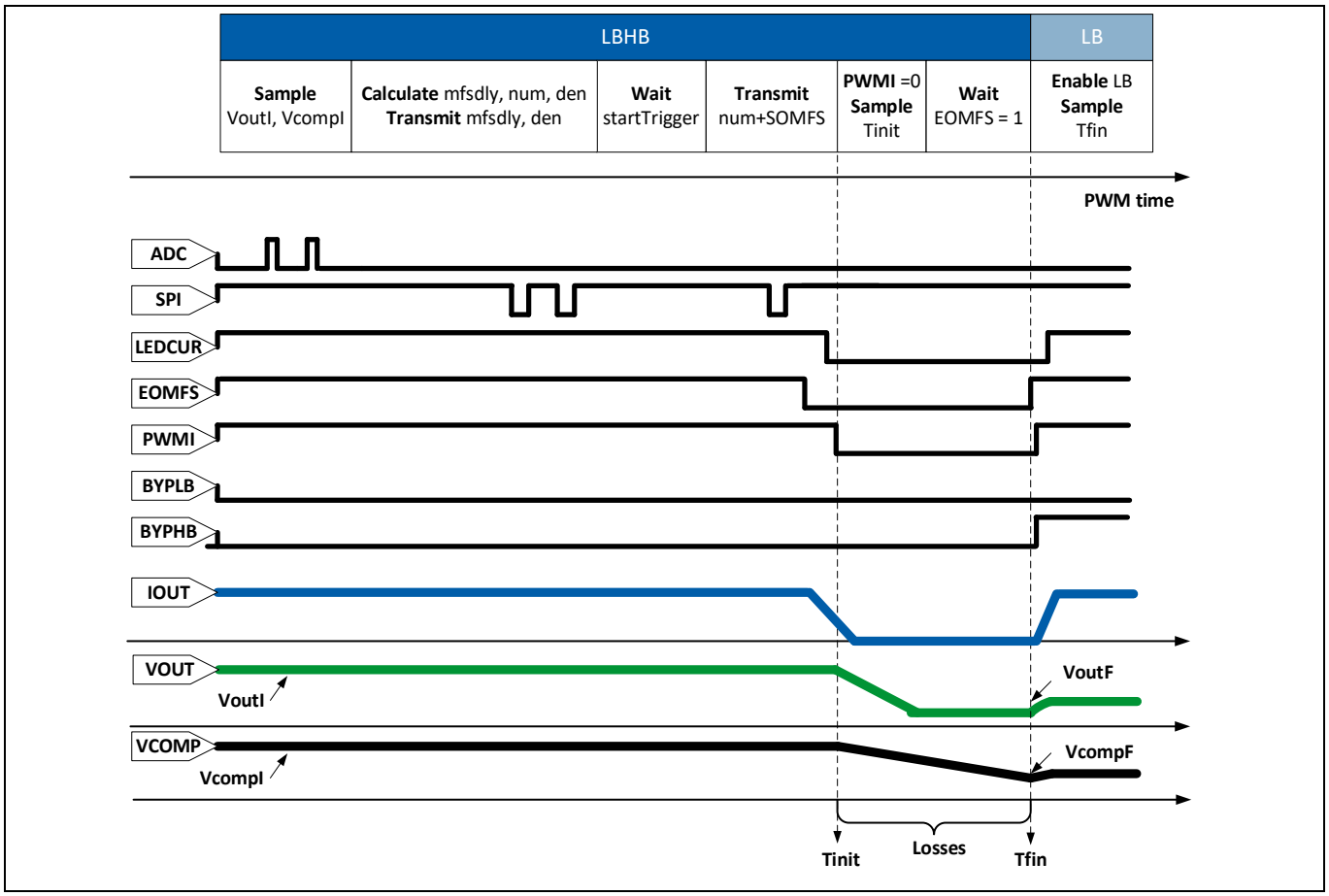


Figure 4 LED step down (LB+HB to LB) timing diagram

1.2 Schedule and prepare repetitive load transitions

The MCU must ensure the requested duty cycle of each light function despite the output current losses and execution time of MCU activities (e.g. fast discharge). To do so, duty cycle losses have to be compensated and MCU execution time has to be managed so as not to impact the effective produced duty cycle.

1.2.1 Duty cycle losses compensation

A short interruption of the output current “losses” is observed during both step-up and step-down load transition in all the active light functions, until V_{OUT} is stable and the device control loop is able to provide the target output current. These losses impact on the final LED light accuracy because the effective light duty cycle produced by the module is smaller than the target one. To maintain the requested average current, the losses measured in a certain PWM cycle can be used to adjust the next PWM cycle. The simplest solution is to increase both LB and HB duty cycle according to losses. This approach works only if the sequence presents an OFF phase big enough to accommodate the compensated LB and HB duty cycle.

For example, in a requested scenario (LB 80%, HB 60%, and I_{OUT_nom} 1000 mA) before compensation:

Table 2 Duty cycle target versus effective before loss2 compensation

	Target			Effective		
	Duty cycle [%]	**Nominal Current [mA]**	**Average current [mA]**	**Duty cycle [%]**	**Nominal Current [mA]**	**Average current [mA]**
LB	80	1000	800	80 - loss2 = 77	1000	770
HB	60	1000	600	60 - loss2 = 57	1000	570

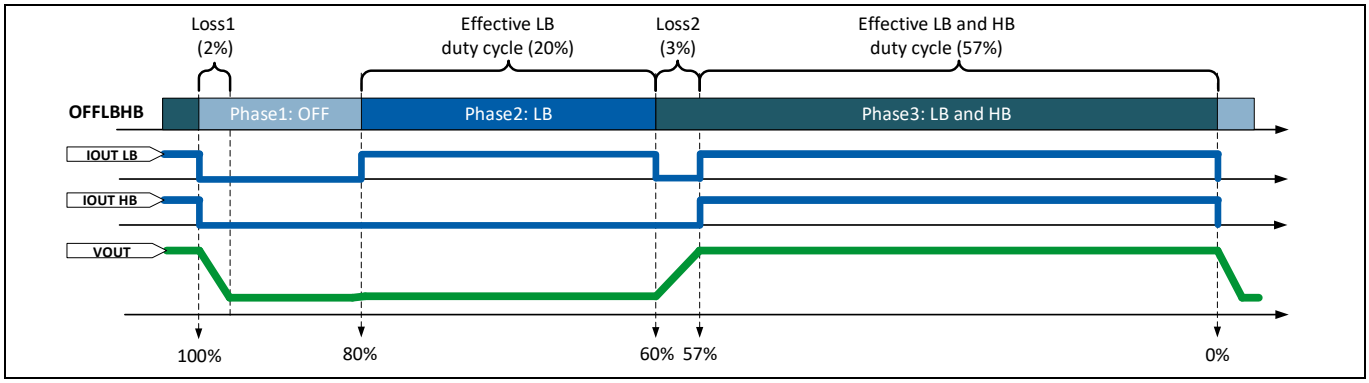


Figure 5 LB 80% and HB 60% before loss compensation

The next PWM cycle both LB and HB duty cycle can be increased by loss2 to compensate the losses.

Table 3 Duty cycle target versus effective after loss2 compensation

	Target			Effective		
	Duty cycle [%]	**Nominal Current [mA]**	**Average current [mA]**	**Duty cycle [%]**	**Nominal Current [mA]**	**Average current [mA]**
LB	80	1000	800	83 - loss2 = 80	1000	800
HB	60	1000	600	63 - loss2 = 60	1000	600

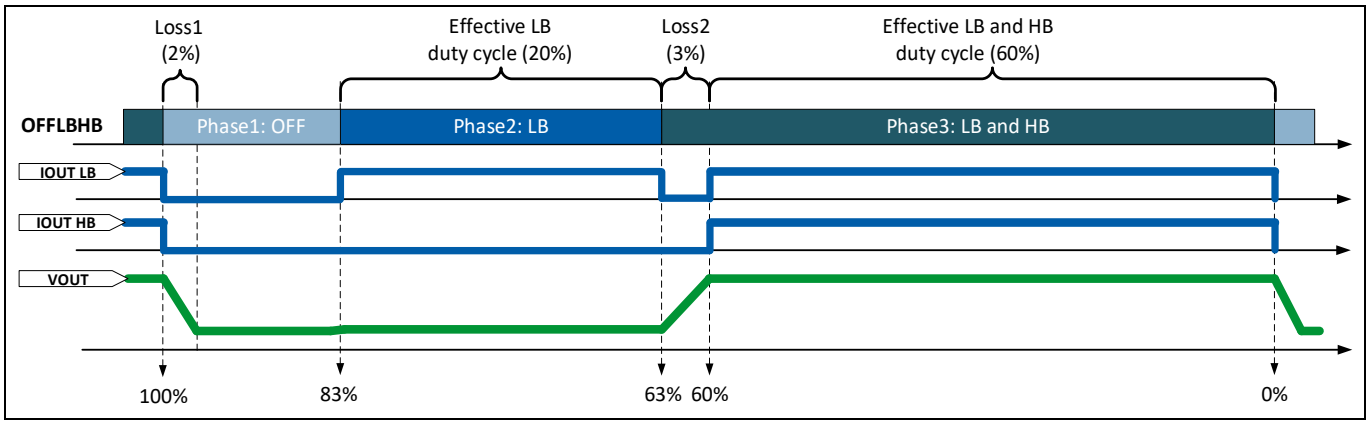


Figure 6 LB 80% and HB 60% after loss compensation

When increasing duty cycle is not possible (e.g. LB 100%, no OFF-phase present), the losses can only be compensated by boosting the nominal current in order to maintain the requested average current.
For example, in a requested scenario (LB 100%, HB 80%, I_{OUT_nom} 1000 mA).

Table 4 Duty cycle target versus effective before loss1 and loss2 compensation

	Target			Effective		
	Duty cycle [%]	Nominal Current [mA]	Average current [mA]	Duty cycle [%]	Nominal Current [mA]	Average current [mA]
LB	100	1000	1000	100 - loss1 - loss2 = 95	1000	950
HB	80	1000	800	80 - loss2 = 77	1000	770

To fulfil the target output current average, the nominal current has to be boosted "I_{OUT_comp}" and HB duty cycle has to be modulated according to the new nominal current.

- $I_{OUT_comp} = I_{OUT_nom}$*100%/95% = 1053 mA.
- HB duty cycle = 80%*1000 mA/1053 mA + loss2 = 79%

Table 5 Duty cycle target versus effective after loss1 and loss2 compensation

	Target			Effective		
	Duty cycle [%]	Nominal Current [mA]	Average current [mA]	Duty cycle [%]	Nominal Current [mA]	Average current [mA]
LB	100	1000	1000	100 - loss1 - loss2 = 95	1053	1000
HB	80	1000	800	79 - loss2 = 76	1053	800

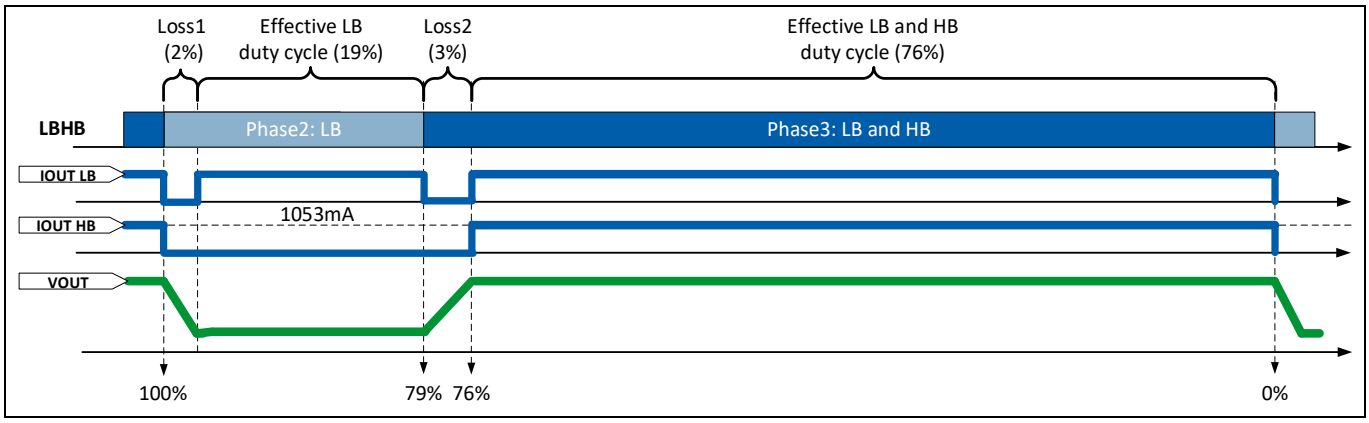


Figure 7 LB 100%, HB 60% after loss compensation

1.2.2 Prepare new PWM cycle

Load step-up transitions require simple and fast real time MCU activities with negligible execution time. Step-down load transitions, losses compensation, new sequence calculation, and load fault detection mechanism require more complex and time-consuming activities (calculations, ADC reading, SPI transmission etc.). To simplify the SW structure, all these procedures have been aggregated into a single activity “prepare new cycle” which is executed only once before a new PWM cycle is initiated. To do so, the PWM sequences must be designed so as to guarantee the need for a single step-down per cycle at most (see Table 6).

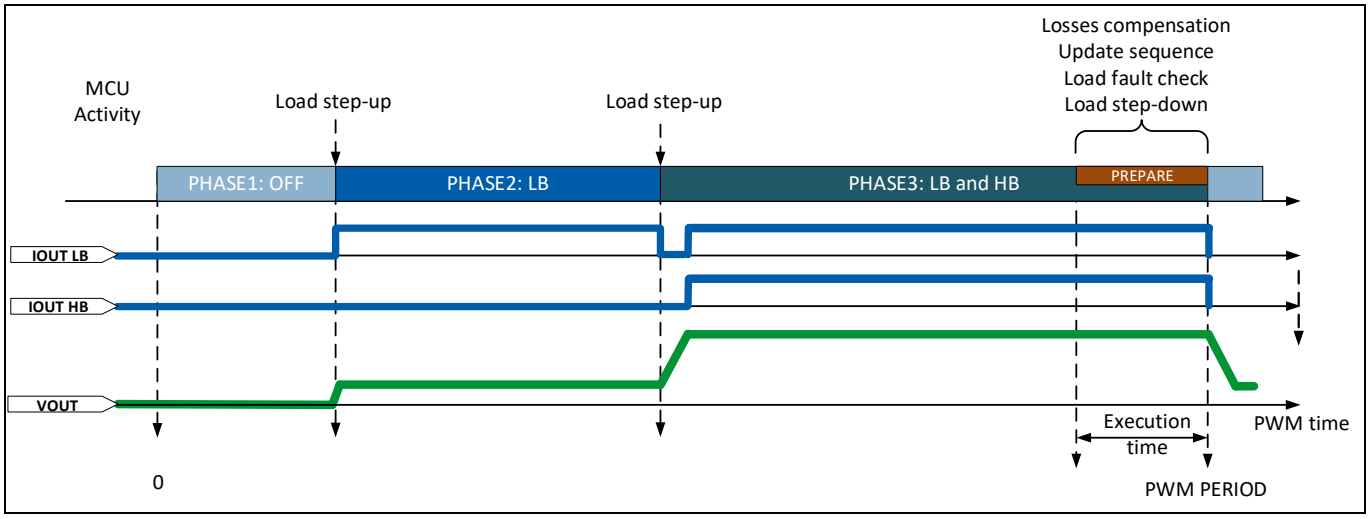


Figure 8 Prepare new PWM cycle

Execution time of “prepare new cycle” is not constant and to ensure the correct duty cycle/PWM period, the final load transition must start exactly at the end of the ongoing PWM cycle. To do so, the beginning of MCU activities must be anticipated by a certain execution time margin and a flexible delay must be added to synchronize load transition with the end of the PWM cycle. When MCU is busy with these activities the LED load cannot be changed. Therefore the “prepare new cycle” execution time limits the minimum duty cycle which can be obtained with this SW solution.

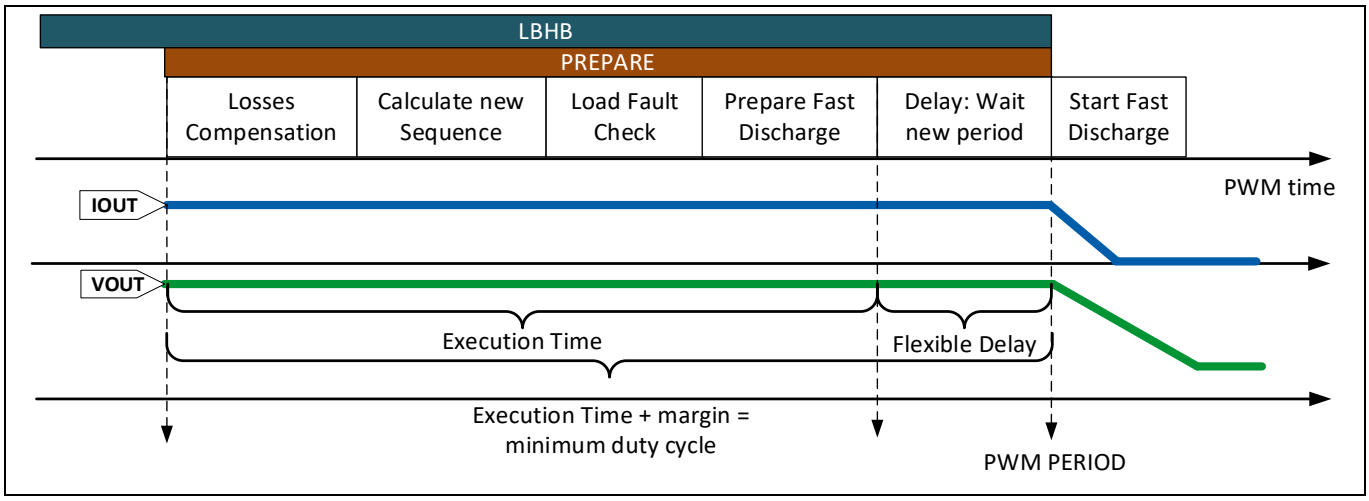


Figure 9 Prepare new PWM cycle

Table 6 Sequences produced by LB and HB conditions

Name	Inputs		Sequence					
			phase1		phase2		phase3	
	LB duty cycle [%]	HB duty cycle [%]	LFC	Duration [%]	LFC	Duration [%]	LFC	Duration [%]
LB	100	0	LB	100	-	-	-	-
LBHB	100	min < HBdc < 100	LB	100-HBdc	LBHB	Hbdc	-	-
LBHBt	100	100	LBHB	100	-	-	-	-
OFFLB	min < LBdc < 100	0	OFF	100-LBdc	LB	LBdc	-	-
OFFLBHB	min < LBdc < 100	min < HBdc < LBdc	OFF	100-LBdc	LB	LBdc	LBHB	HBdc
OFFLBHBt	min < LBdc < 100	LBdc	OFF	100-LBdc	LBHB	LBdc	-	-

The following diagrams show typical sequences specified in Table 6.

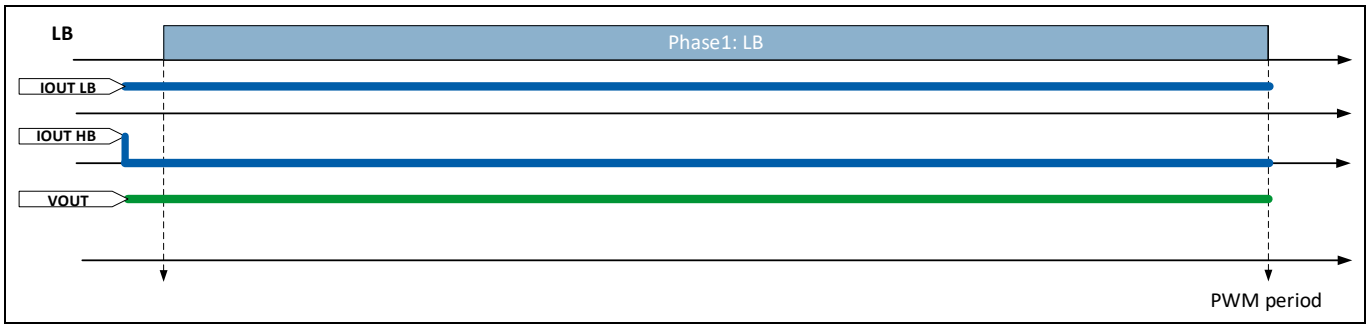


Figure 10 Scenario LB for LBdc = 100%

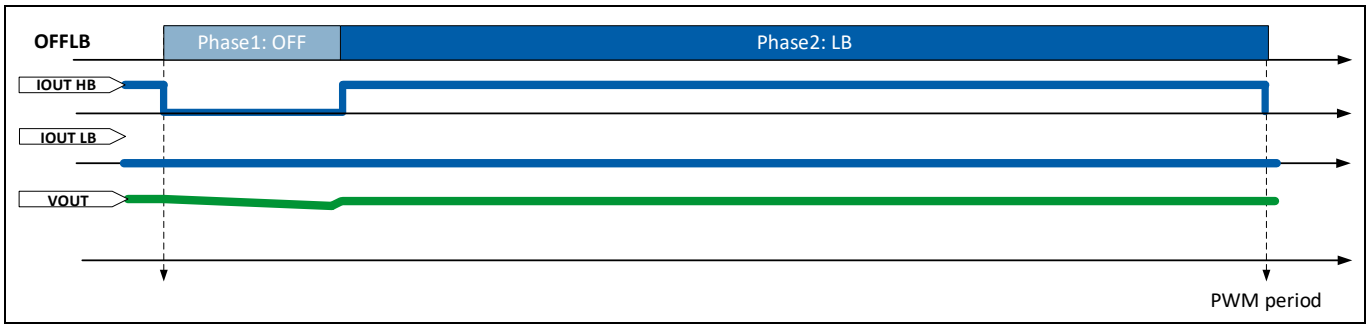


Figure 11 Scenario OFFLB for LBdc = 80%

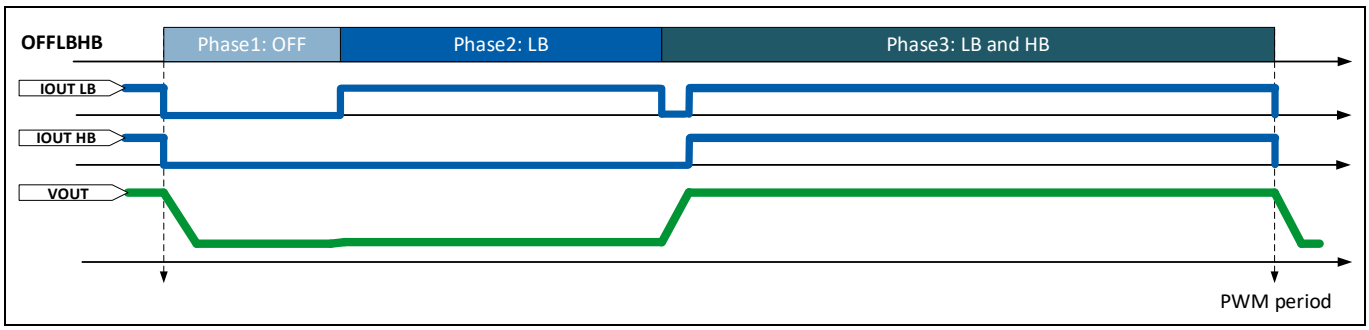


Figure 12 Scenario OFFLBHB for LBdc = 80% and HBdc = 60%

1.3 Safety mechanism

Input battery protection against undervoltage/overvoltage can be implemented by MCU, for example using upper/lower threshold Interrupt service routines. TLD5542-1QV status can be monitored through SPI interface by reading the standard diagnosis register (see Chapter 10.2 datasheet [1]). In particular TLD5542-1QV has output overvoltage (OUTOV) and short circuit (SHRTLED) protections, useful to spot LED load faults.
The status of the module is reported on the “diagnosisLog” register.

Table 7 diagnosisLog 16-bit

7	6	5	4	3	2	1	0
TLD5542	VBAT_UV	VBAT_OV	TAP OPEN	HB SHORT	HB_OL	LB_SHORT	LB_OL

Name	Mask [Hex]	Description
LB_OL	0x01	1b when VOUT wire is disconnected or LB LEDs are open
LB_SHORT	0x02	1b when VOUT wire is shorted to VTAP or GND or LB LEDs are shorted
HB_OL	0x04	1b when VOUT wires GND wire are disconnected or HB LEDs are open
HB SHORT	0x08	1b when TAP wire is shorted to GND or HB LEDs are shorted
TAP OPEN	0x10	1b when TAP wire is disconnected
VBAT_OV	0x20	1b when input battery VOUT exceed the extended battery range
VBAT_UV	0x40	1b when input battery VOUT drops below the extended battery range
TLD5542	0x80	1b when any fault/warning between “SWRST_BSTUV”, “UVLORST”, “TER”, “TSD”, “TW”, “STATE = IDLE”, “STATE = LIMPHOME” has been reported by TLD5542-1QV SPI standard diagnosis

1.3.1 Input battery protection

The proposed battery V_{BAT} protection strategy implements independent hysteresis for enter/exit both overvoltage and undervoltage condition:

- LED module activities are enabled only when vbat is in the valid battery range: $V_{BAT} > V_{BAT_UV_EXIT}$ and $V_{BAT} < V_{BAT_OV_EXIT}$ (input battery over/undervoltage exit thresholds)
- LED module activities are disabled as soon as input battery exceed or drops below the extended battery range: $V_{BAT} \leq V_{BAT_UV_ENTRY}$ or $V_{BAT} \geq V_{BAT_OV_ENTRY}$ (input battery over/undervoltage entry thresholds)

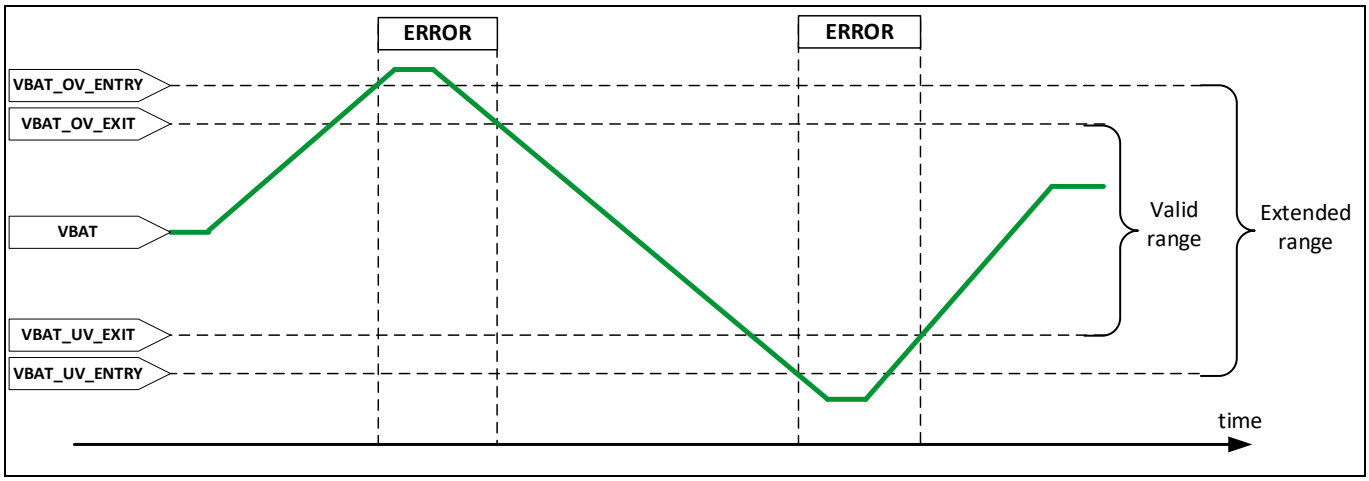


Figure 13 Battery management strategy

1.3.2 Load fault detection

In systems with LB and HB driven by a single DC-DC, the controller is assumed to be connected to the load with 3 wires: VOUT, VTAP, and GND (see Figure 14A). Periodically reading TLD5542-1QV standard diagnosis may not be enough to spot any possible load fault in this architecture. Additional diagnosis procedures based on ADCs “ADCVOUT” and “ADCVTAP” readings have been introduced to periodically analyze load status and identify faults which cause effects on light output but do not trigger OUTOV or SHRTLED (i.e. VTAP OPEN, VOUT SHORT to TAP and VTAP SHORT to GND on LB+HB), see Table 8.

Table 8 Load faults (either LED or WIRE failure) impact on system behavior

Active scenario		LB		LB+HB	
		TLD5542-1QV SPI diagnosis	**Effect on light**	**TLD5542-1QV SPI diagnosis**	**Effect on light**
Fault	**LB OPEN** (VOUT wire or LB LED open)	OUTOV	ALLOFF	OUTOV	ALLOFF
	VTAP OPEN (VTAP wire open)	-	HB ON	-	HBdc = LBdc
	HB OPEN (GND wire or HB LED open)	-	-	OUTOV	ALLOFF
	VOUT SHORT TO TAP (VOUT wire short to TAP or LB LED shorted)	SHRTELD	ALLOFF	-	LB OFF
	VOUT SHORT-TO-GND (VOUT wire short-to-GND or LB and HB LED shorted)	SHRTLED	ALLOFF	SHRTLED	ALLOFF
	VTAP SHORT TO GND (VTAP wire short-to-GND or HB LED shorted)		-	-	HB OFF

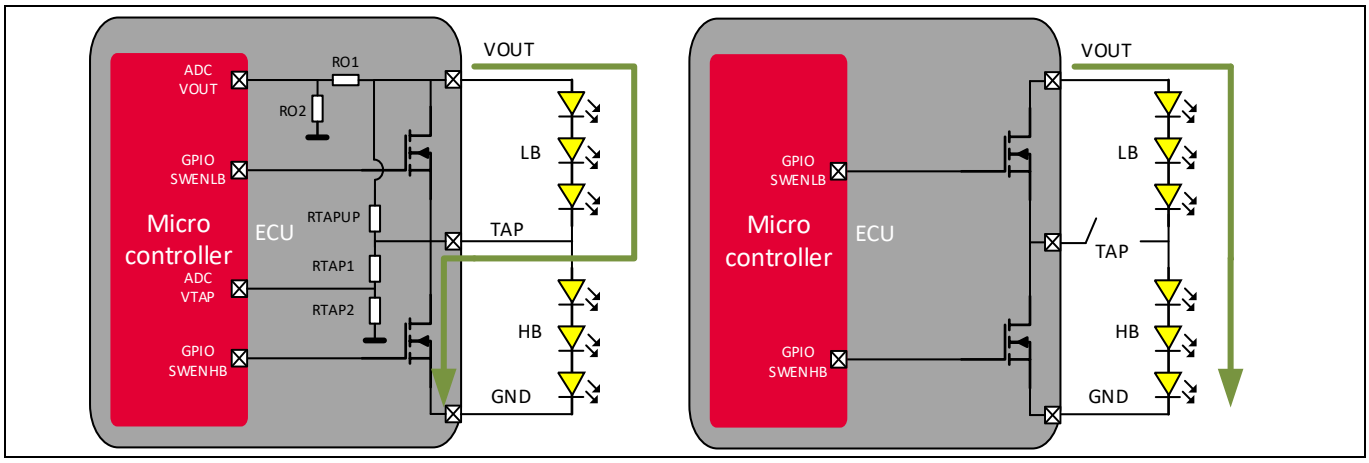


Figure 14 A: 3 wire load connection, B: broken TAP wire

For example, when HB is off and LB on, current only flows through LB (Figure 15 A) however, if TAP is open (Figure 15 B), output current flows through LB and HB together and neither TLD5542-1QV SHRTLED nor TLD5542-1QV OUTOV are triggered. At the first LB activation the “check TAP” procedure must be used to spot TAP OPEN fault. The following steps describe the procedure:

- Enable HB
- MCU starts polling GPIO_LEDCUR, waiting for a high-to-low transition caused by HB activation
- If TAP is properly connected GPIO_LEDCUR will change to low as soon as HB bypass is completely closed
- If TAP is open or shorted to GND, the GPIO_LEDCUR will never change to low. After a certain timeout the procedure stops and report a fault. If TAP is open ADCTAP is pulled up by the ADC resistor divider net (RTAPUP, RTAP1 and RTAP2). On the contrary ADCTAP is forced to GND when TAP is shorted
- Disable HB. Fast Discharge could be necessary because TLD5542-1QV keep switching during the procedure leading V_{OUT} to increase but if HB bypass is fast enough it is sufficient to reduce output current (i.e. 1/3 of nominal) during the procedure

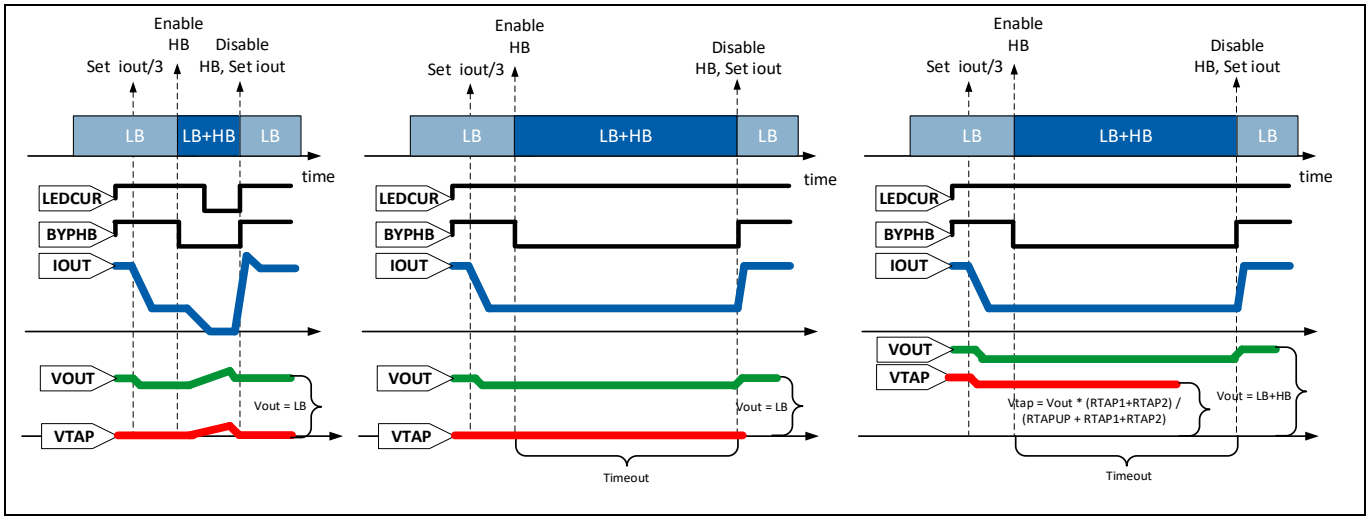


Figure 15 Figure A: VTAP properly connected. Figure B: VTAP shorted to GND, Figure C: VTAP open

Note: RTAP1, RTAP2, RO1 and RO2 used to scale VOUT and VTAP to MCU ADCs operating range (i.e. 0 to 5 V)

2 Software architecture

To implement all the activities controlled by MCU presented in System architecture, Chapter 1 we propose the following software SW_HBLBIndividualDutyCycle_V1.0 example composed of a finite state machine where state transitions are scheduled by the status of a timer “TIMER_PWM”, the asynchronous “ERROR_S” signal and the synchronous signal “SCENARIO”.

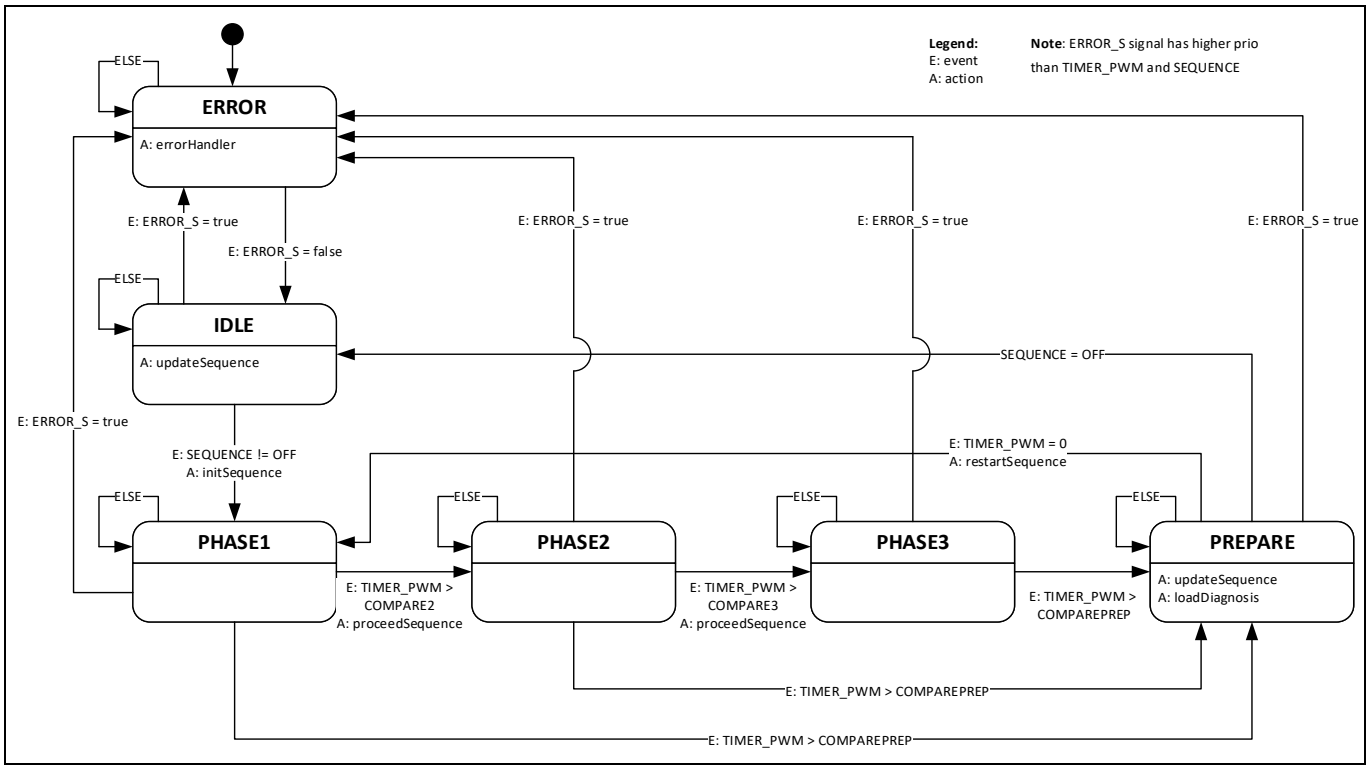


Figure 16 Finite state machine

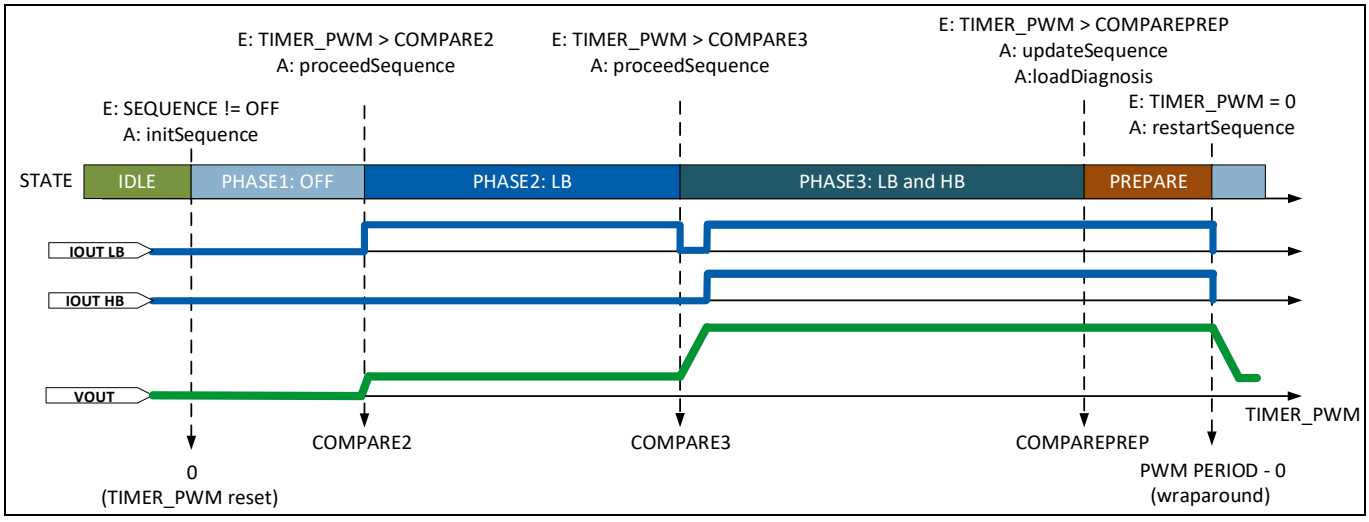


Figure 17 Finite state machine timing diagram

2.1 FSM states

The following states represent the status of the system during normal PWM activity (PHASE1, PHASE2, PHASE3, and PREPARE), IDLE or ERROR conditions:

- **ERROR:** Input battery is out of range or there are pending fault/warning detected by TLD5542-1QV SPI standard diagnosis or load fault spotted by dedicated load diagnosis procedures waiting to be solved
- **IDLE:** TLD5542-1QV is stopped and the system is waiting for new scenarios. No current flows through the load
- **PHASE1:** TLD5542-1QV is active. Current flows through the LFC specified by phase 1 (except when off)
- **PHASE2:** TLD5542-1QV is active. Current flows through the LFC specified by phase 2
- **PHASE3:** TLD5542-1QV is active. Current flows through the LFC specified by phase 3
- **PREPARE:** TLD5542-1QV is active. Current flows through the LFC specified by the last activated phase before entering this state. MCU prepare all data and Fast Discharge (if needed) for the up-coming new PWM cycle is executed (see Prepare new PWM cycle, Chapter 1.2.2)

2.2 FSM events

- The "**TIMER_PWM"** (generic timer with programmable wrap-around) provides a time reference used to allocate the correct PWM duty cycle at each phase. TIMER_PWM wrap-around (TIMER_PWM = 0) indicates the beginning of a new PWM cycle and TIMER_PWM runtime value is compared with values "COMPARE2", "COMPARE3" and "COMPAREPREP" to trigger state transitions.
    - **COMPARE2** specifies the time instant to enter PHASE2. This value depends on the requested scenario and is calculated before starting every PWM cycle, see Update sequence Chapter 2.3.1
    - **COMPARE3** specifies the time instant to enter PHASE3. This value depends on the requested scenario and is calculated before starting every PWM cycle, see Update sequence Chapter 2.3.1
    - **COMPAREPREP** specifies the time instant to enter PREPARE. This value is constant and depends on MCU performance, see Prepare new PWM cycle, Chapter 1.2.2
    - **TIMER_PWM = 0** (TIMER_PWM wraparound) indicates the time instant to restart the PWM phase sequence. FSM must enter PHASE1
- The signal "**ERROR_S"** is set by the safety mechanisms under the following conditions:
    - When asynchronous services to detect input battery overvoltage/undervoltage are triggered (e.g. upper/lower ADC threshold interrupts)
    - When any synchronous TLD5542-1QV standard diagnosis SPI reports any fault/warning
    - When any synchronous LED load diagnosis mechanism (e.g. VTAP readings) detects LED fault
    - The ERROR_S signal is reset only by the procedure which evaluates and repairs errors
- The synchronous signal "**SCENARIO"** indicates whether the PWM cycle is not empty (off), meaning that at least one light function has to be enabled and PWM activities must start. Scenario is updated by updateSequence action (Chapter 2.3.1) using input data (LB duty cycle, HB duty cycle, and nominal current) provided by the user through dedicated APIs.

2.3 FSM actions

At any state transition the MCU executes at least one of the following actions.

2.3.1 Update sequence

This activity is executed before starting or restarting a sequence and it is intended to prepare all necessary data to schedule and execute phase transitions. According to input scenario (HB duty, LB duty, and nominal current) received from the user via dedicated APIs it:

- Prepares the ordered sequence of phases (see Chapter 1.2.2, Prepare new PWM cycle). For each phase define the LFC and the time to enable it (compare2, compare3)
- Executes losses compensation procedure (see Chapter 1.2.1) which (if needed) adjusts duty cycles and nominal current according to losses measured in the previous cycle
- Modifies the requested scenario according to load faults reported in diagnosis log. Light functions with reported faults are exclude from the sequence.

2.3.2 Initiate sequence

This activity is used to start a new sequence of PWM phases.

- Initialize TLD5542-1QV by setting the EN pin to high and sending SPI configuration commands.
- Fast discharge is used to discharge both C_{OUT} and C_{COMP} to minimum values (V_{OUTMIN}, $V_{COMPCLAMP}$)
- SPI analog dimming command is transmitted to set the nominal current
- LFC of the first PWM phase is enabled
- TIMER_PWM is reset

2.3.3 Progress sequence

This activity is used to proceed in the sequence of PWM phases. According to all structures of the allowed sequences (see Figure 8 and Table 6) this activity involves only LED load step-up (See Chapter 1.1.1).

2.3.4 Restart sequence

This activity is used to restart the sequence of PWM phases. Different actions are performed according to the structure of the up-coming sequence and the LFC active when restart sequence is triggered:

- If the active LFC and the LFC of the first phase in the new sequence are equal, no action is needed (e.g. LB 100%). Wait until the TIMER_PWM = 0 which is the wrap-around event (e.g. LB 100%, Figure 7).
- If the active LFC and the LFC of the second phase in the new sequence are equal and the LFC of the first phase is OFF, PWMI has to be set LOW just after the TIMER_PWM = 0 wraparound event (e.g. LB 80%, Figure 5 and Figure 6)
- If the remaining cases LFC changes after restarting sequence. LED step-down transition is required (see Chapter 1.1.1). The fast discharge procedure must start just after the TIMER_PWM = 0 wraparound event, see an example in Figure 16

To simplify SW structure the flag “eopwms” has been introduced to indicate that TIMER = 0 wrap-around has occurred (eopwms = false). The flag is the set true before leaving restart sequence activity.

2.3.5 Update current and load diagnosis

The regulated output current is refreshed or updated (e.g. losses compensation) just before restart of the sequence, by writing the analog dimming register. The SPI response to this write register command is a standard diagnosis, useful to detect any possible TLD5542-1QV warning/fault. The SHRTLED and OUTOV protection report short-to-ground or overvoltage occurred during the entire PWM cycle. When driving multiple functions in series, protection does not occur, therefore additional checks are performed using ADCs and others techniques as described in Chapter 1.3.2. If any fault/warning is detected “ERROR_S” is set and FSM is forced to enter ERROR state.

2.3.6 Stop sequence

This activity stops TLD5542-1QV by setting EN and PWMI pin low.

2.3.7 Error handler

This activity is repeatedly performed when FSM is in ERROR state. TLD5542-1QV is stopped and route cause of the error conditions are checked. Any battery error, TLD5542-1QV Faults/warning and load faults are reported on the dedicated “diagnosisLog. As soon as error is solved it resets ERROR_S and FSM enters IDLE state:

- Check if battery voltage is out of the range (see Chapter 1.3.1)
- Check the last received SPI standard diagnosis response and reacts if any TLD5542-1QV fault/warning between SWRST_BSTUV, UVLORST, STATE, TER, IVCCUVLO, TSD and TW is reported. (Note: no reactions have been developed in this example code. The user shall modify the code with his/her own procedures)
- Check if load faults have been detected by dedicated services (see Update current and load diagnosis, Chapter 2.3.5)

2.4 Software modules

Figure 18 shows the layered SW_HBLBIndividualDutyCycle_V1.0 structure which implements the FSM presented in previous chapters.

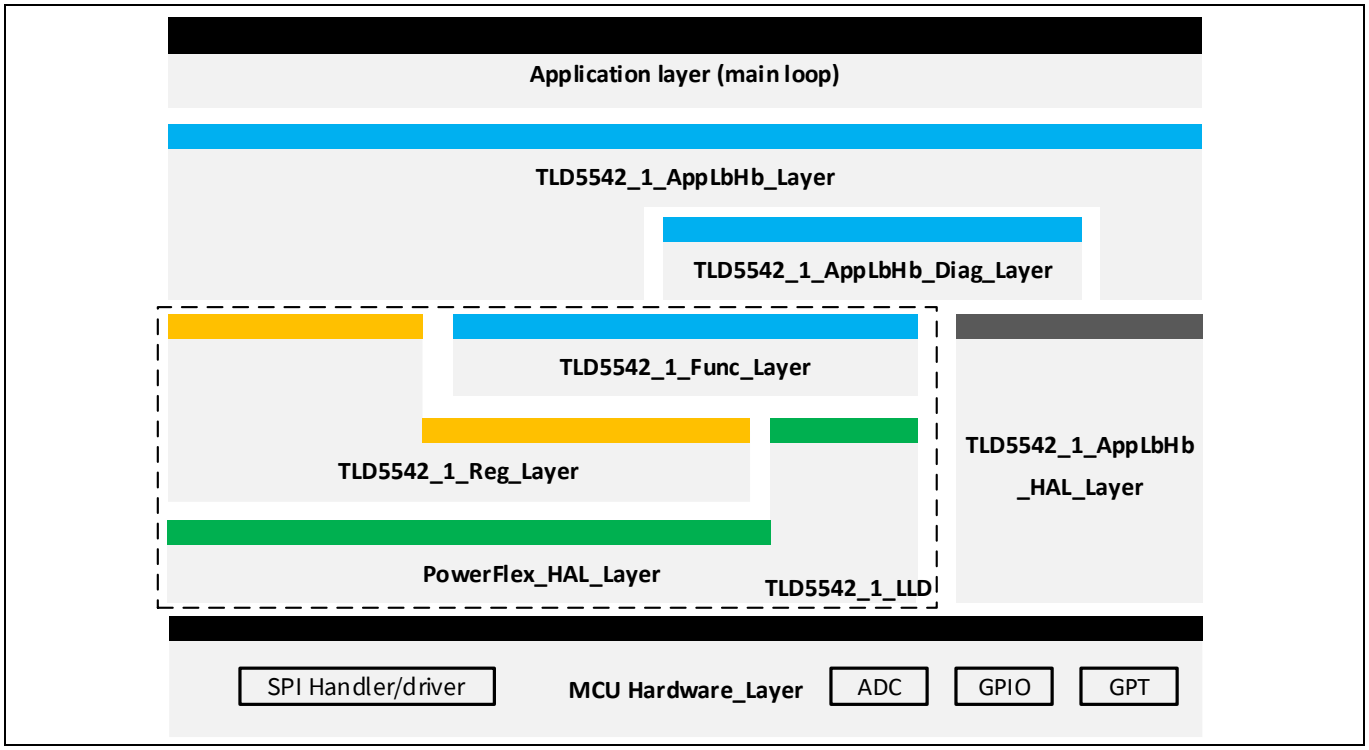


Figure 18 Software layered structure

- The finite state machine, its activities and events are implemented in the “TLD5542_1_AppLbHb_layer”, except for safety mechanisms and diagnosis services which are implemented in the module TLD5542_1_AppLbHb_Diag_Layer.
- Fast discharge, analog dimming and other functionalities intended to configure and monitor TLD5542-1QV are implemented using APIs offered by TLD5542-1_LLD low level drivers: either “TLD5542_1_Func_Layer” or “TLD5542_1_Reg_Layer”.
- The module “TLD5542_1_AppLbHb_HAL_layer” creates a hardware abstraction layer between the finite state machine and the MCU hardware components (GPIOs, GPTs and ADCs).

2.5 Interface APIs

The user (Application Layer) can communicate with lower layers through the following asynchronous APIs offered by TLD5542_1_AppLbHb_Layer.

Table 9 TL5542_1_AppLbHb_set

Service Name	**TL5542_1_AppLbHb_set**	
Syntax	Bool TL5542_1_AppLbHb_set(Uint32 iout, uint32 dutyHB, uint32 dutyLB)	
Parameters (in)	I_{OUT}	LED load (LB and HB) nominal current [mA]
Parameters (in)	dcHBp	HB PWM duty cycle [‰]
Parameters (in)	dcLBp	LB PWM duty cycle [‰]
Return value	Bool	true when inputs are valid false when inputs are not valid
Description	Allows the user to change scenario by updating nominal output current and duty cycles of both LB and HB. Input data are validated before being used to generate new PWM phase sequences. $I_{OUTmin} < I_{OUT} < I_{OUTmax}$ dcMin < dcHBp < 10000 ‰ dcMin < dcLBp < 10000 ‰ dcHBp <= dcLBp	

Table 10 TLD5542_1_AppLbHb_getDiagnosisLog

Service Name	**TLD5542_1_AppLbHb_getDiagnosisLog**	
Syntax	Uint16 TL5542_1_AppLbHb_getDiagnosisLog(void)	
Return value	Uint16	Diagnosis log register
Description	Return to the user diagnosis Log register which report the status of the module (see Table 7)	

Table 11 TLD5542_1_AppLbHb_clearDiagnosisLog

Service Name	**TLD5542_1_AppLbHb_clearDiagnosisLog**	
Syntax	void TL5542_1_AppLbHb_clearDiagnosisLog(uint16 mask)	
Parameters (in)	mask	Load fault mask to be clear. Possible pre-defined choice: LB_OL_MASK LB_SHORT_MASK HB_OL_MASK HB_SHORT_MASK TAP_BROKEN_MASK TLD5542_ERROR_MASK
Description	Allows the user to rest load faults reported on the diagnosis Log	

3 Requirement specification

3.1 Hardware requirements

The following list presents the microcontroller hardware specification.

HWRQ.01: The firmware requires #4 Output GPIOs *"PWMI", "EN", "SWENLB"* and *"SWENHB"*

HWRQ.02: The firmware requires #2 Input GPIOs *"EOMFS"* and *"LEDCUR"*

HWRQ.03: The firmware requires #4 ADCs *"VOUT", "VCOMP", "VBAT"* and *"VTAP"* with sampling time smaller than 20 µs and at least 10-bit accuracy

HWRQ.04: The firmware requires #1 SPI 4-wire interface "*CSN", "SCLK", "MOSI"* and *"MISO". "SCLK"* frequency up to 5 MHz, CPOL=0, CPHA = 1

HWRQ.05: The firmware requires #1 programmable 16-bit timer *"TIMER_PWM"* with at least 1 MHz clock and programmable wrap-around value *"PWMPERIOD"*

HWRQ.06: The firmware requires #1 interrupt service routine *"ISR_PWM_PwmPeriod"* generated by *"TIMER_PWM"* wraparound events

HWRQ.07: The firmware requires #1 programmable 16-bit timer "*TIMER_TIMEOUT*" with at least 1 MHz clock

HWRQ.08: The firmware requires #1 programmable ADC upper threshold interrupt service routine *"ISR_OV_Vbat_Sense"*. Not mandatory

HWRQ.09: The firmware requires #1 programmable ADC lower threshold interrupt service routine *"ISR_UV_Vbat_Sense".* Not mandatory

3.2 Software requirements

The following list presents all the services (functions/procedures) intended to create a hardware abstraction layer between the MCU hardware components (SPI driver/handler, GPTs, GPIOs, ADCs) and the application software. Those services involve MCU hardware peripheral and must be developed according to the selected MCU. The user shall link the services to the application software through dedicated function pointers placed into TLD5542_1_AppLbHb_HAL.*c* file, see Porting firmware, Chapter 3.3

SWRQ.01: Software shall implement the "*MCU_GPIO_setENlow"* service to set TLD5542-1QV EN pin low

Syntax	void MCU_GPIO_setENlow (void)

SWRQ.02: Software shall implement the "*MCU_GPIO_setENhigh"* service to set TLD5542-1QV EN pin high

Syntax	void MCU_GPIO_setENhigh (void)

SWRQ.03: Software shall implement the "*MCU_GPIO_setPWMIlow"* service to set TLD5542-1QV PWMI pin low

Syntax	void MCU_GPIO_setPWMIlow (void)

SWRQ.04: Software shall implement the "*MCU_GPIO_setPWMIhigh"* service to set TLD5542-1QV PWMI pin high

Syntax	void MCU_GPIO_setPWMIhigh (void)

SWRQ.05: Software shall implement the "*MCU_GPIO_getEOMFS"* service to get TLD5542-1QV EOMFS pin status

Syntax	Uint8 MCU_GPIO_getEOMFS (void)

Return	status	0 if EOMFS is low, 1 if EOMFS is high

SWRQ.06: Software shall implement the "*MCU_GPIO_getLEDCUR"* service to get TLD5542-1QV LEDCUR pin status

Syntax	Uint8 MCU_GPIO_getLEDCUR (void)	
Return	status	0 if EOMFS pin is low, 1 if EOMFS pin is high

SWRQ.07: Software shall implement the "*MCU_GPIO_setLfcOFF"* service to configure load bypass switches and PWMI in order to disable both LB and HB ("PWMI = 0, LB bypass switch off, HB bypass switch off)

Syntax	void MCU_GPIO_setLfcOFF (void)

SWRQ.08: Software shall implement the "*MCU_GPIO_setLfcLB"* service to configure load bypass switches and PWMI in order to enable only LB (PWMI = 1, LB bypass switch on, HB bypass switch on)

Syntax	void MCU_GPIO_setLfcLB (void)

SWRQ.09: Software shall implement the "*MCU_GPIO_setLfcLBHB"* service to configure load bypass switches and PWMI in order to enable only HB (PWMI = 1, LB bypass switch on, HB bypass switch off)

Syntax	void MCU_GPIO_setLfcLBHB (void)

SWRQ.10: Software shall implement the "*MCU_TIMER_resetTimeout"* to reset TIMER_TIMEOUT

Syntax	void *MCU_TIMER_resetTimeout* (void)

SWRQ.11: Software shall implement the "*MCU_TIMER_getTimeout"* to get TIMER_TIMEOUT value on runtime

Syntax	Uint16 MCU_TIMER_getTimeout (void)	
Return	value	TIMER_TIMEOUT value

SWRQ.12: Software shall implement the "*MCU_TIMER_resetPWM"* to reset TIMER_PWM

Syntax	void *MCU_TIMER_resetPWM* (void)

SWRQ.13: Software shall implement the "*MCU_TIMER_getPWM"* to get *"TIMER_PWM"* value on runtime

Syntax	Uint16 MCU_TIMER_getPWM (void)	
Return	value	TIMER_PWM value

SWRQ.14: Software shall implement the "*MCU_TIMER_getPWMPeriod"* to get *"TIMER_PWM"* period match (wrap-around) value

Syntax	Uint16 MCU_TIMER_getPWMPeriod (void)	
Return	value	TIMER_PWM value

SWRQ.15: Software shall set "*eopwms*" external global variable true at every "*ISR_OV_Vbat_Sense"* occurrence

SWRQ.16: Software shall implement the "*MCU_ADC_readVcomp"* to measure TLD5542-1QV compensation capacitor voltage *"VCOMP"* [mV]

Syntax	Uint32 MCU_ADC_readVcomp (void)	
Return	value	"VCOMP" voltage [mV]

SWRQ.17: Software shall implement the "*MCU_ADC_readVout"* to measure TLD5542-1QV output capacitor voltage *"VOUT"* [mV]

Syntax	Uint32 MCU_ADC_readVout (void)	
Return	value	"VOUT" voltage [mV]

SWRQ.18: Software shall implement the "*MCU_ADC_readVcomp"* to measure TLD5542-1QV input battery voltage *"VBAT"* [mV]

Syntax	Uint32 MCU_ADC_readVbat (void)	
Return	value	"VBAT" voltage in [mV]

SWRQ.19: Software shall implement the "*MCU_ADC_readVtap"* to measure LED load TAP voltage "VTAP" [mV]

Syntax	Uint32 MCU_ADC_readVtap (void)	
Return	value	"VTAP" voltage in [mV]

SWRQ.20: Software shall implement the "*MCU_ADC_setVbatOVUVTh"* to set both undervoltage and overvoltage threshold values of *"ISR_UV_Vbat_Sense"* and *"ISR_OV_Vbat_Sense"* respectively. Not mandatory

Syntax	Uint32 *MCU_ADC_setVbatOVUVTh* (Uint32 undervoltage_mV, Uint32 overvoltage_mV)	
ParamIN	undervoltage_mV	Input battery undervoltage threshold [mV]
ParamIN	overvoltage_mV	Input battery overvoltage threshold [mV]

SWRQ.21: Software shall set "ERROR_S" external global variable true at every "*ISR_OV_Vbat_Sense"* occurrence.

SWRQ.22: Software shall set "ERROR_S" external global variable true at every "*ISR_OV_Vbat_Sense"* occurrence.

3.3 Porting firmware

The steps allow the user to adapt the proposed firmware SW_HBLBIndividualDutyCycle_V1.0 on his/her own system hardware (e.g. different MCU):

- Ensure that the selected MCU fulfills all the HW requirements presented in Chapter 3.1 (except for not mandatory)
- Provide all SW services described in Chapter 3.2 and link them to the module TLD5542_1_AppLbHb_HAL_Layer.c through dedicated function pointers (see Figure 19).
- Include TLD5542-1QV LLD low level drivers as described in the Powerflex application note [2]
- Adjust (if needed) system parameters presented in Table 12 and which are defined in TLD5542_1_AppLbHb_HAL_Layer.h module
- Include the example software in the application
- Include the finite state machine and its communication APIs into the application main loop

Table 12 System parameters

Name	Typ. value	Description
RFB_mOhm	100	TLD5542-1QV output sensing resistor value [mΩ]
CCOMP	15	TLD5542-1QV compensation capacitor value [nF]
FSW	420	TLD5542-1QV switching frequency value [kHz]
VCOMPDISCHMARGIN	70	C_{COMP} discharge margin applied on MFSDLY calculation to ensure smooth current restart after fast discharge
TIMEOUTSOFTSTART	65000	TLD5542-1QV soft-start period (see Chapter 6.2 TLD5542-1QV datasheet [1]) Used as timeout for blocking while instruction (i.e. while (LEDCUR == 0)) [100 ns] 65000 = 6.5 ms
TBLANKBYP	15	HB maximum bypass switch aperture time [µs]
VOUTMIN	3000	Minimum expected LED function forward voltage [mV]
VBAT_UV_ENTRY	6000	Input battery voltage threshold to enter undervoltage condition [mV]
VBAT_UV_EXIT	8000	Input battery voltage threshold to exit undervoltage condition [mV]
VBAT_OV_ENTRY	18000	Input battery voltage threshold to enter overvoltage condition [mV]
VBAT_OV_EXIT	20000	Input battery voltage threshold to exit overvoltage condition [mV]
PREPAREPWMDC	500	Prepare PWM cycle activity execution time. Defined as duty cycle of PWM period [‰]

```
/*********************************************************************************
                    Links to external services
*********************************************************************************/
/*link your functions to HAL interfaces*/

static ptr_GPIO_setENlow_Type ptr_GPIO_setENlow = &TLE9842_GPIO_setENlow;          /*SWRQ.01*/
static ptr_GPIO_setENhigh_Type ptr_GPIO_setENhigh = &TLE9842_GPIO_setENhigh;      /*SWRQ.02*/
static ptr_GPIO_setPWMIlow_Type ptr_GPIO_setPWMIlow = &TLE9842_GPIO_setPWMIlow;   /*SWRQ.03*/
static ptr_GPIO_setPWMIhigh_Type ptr_GPIO_setPWMIhigh = &TLE9842_GPIO_setPWMIhigh;/*SWRQ.04*/
static ptr_setLfcOFF_Type ptr_setLfcOFF = &TLE9842_setLfcOFF;                      /*SWRQ.07*/
static ptr_setLfcLB_Type ptr_setLfcLB = &TLE9842_setLfcLB;                         /*SWRQ.08*/
static ptr_setLfcLBHB_Type ptr_setLfcLBHB = &TLE9842_setLfcLBHB;                   /*SWRQ.09*/
static ptr_GPIO_getEOMFS_Type ptr_GPIO_getEOMFS = &TLE9842_GPIO_getEOMFS;         /*SWRQ.05*/
static ptr_GPIO_getLEDCUR_Type ptr_GPIO_getLEDCUR = &TLE9842_GPIO_getLEDCUR;      /*SWRQ.06*/
```

Figure 19 Link services to TLD5542_1_AppLbHb_HAL_Layer.c

```
/*********************************************************************************
                    Global Functions definition
*********************************************************************************/

void TLE9842_GPIO_setENlow(void){
  PORT->P2_PUDSEL.bit.PP5 = 0;  /*TLE9842-2QX MCU code*/
}
```

Figure 20 SWRQ.01: GPIO_setENlow developed for TLE9842-2QX MCU

```
/*******************************************************************************
**                      Includes                                              **
*******************************************************************************/
#include "tle_device.h"                 /*MCU configuration files*/
#include "TLD5542_1_AppLbHb_Layer.h"    /*TLD5542_1 Application*/


int32_t main(void){

  TLE_Init();                            /*MCU hardware initialization*/
  TLD5542_1_AppLbHb_set(1000,8000,6000); /*Set default scenario iout
                                          1000mA, LB duty 80%, HB duty 60%*/
  /*****************************************************************************
  ** main endless loop                                                        **
  *****************************************************************************/

  while (1){

    TLD5542_1_AppLbHb_FSM();

    /***************************************************************************
    ** watch dog                                                              **
    ***************************************************************************/
    if(WDT1_Service() == 1){};
  }
}
```

Figure 21 Include application modules and FSM

4 List of abbreviations

Table 13 Abbreviations

Abbreviation	Definition
ADC	analog digital converter
API	application programming interface
dc	duty cycle
FD	fast discharge
FSM	finite state machine
GPIO	general purpose input and output
GPT	general purpose timer
HB	high beam
LB	low beam
LED	light emitting diode
LFC	light function combination
MCU	microcontroller unit
PWM	pulse width modulation
SPI	serial peripheral interface
SW	software

5 List of references

[1] Infineon TLD5542-1 Data Sheet Rev.1.10, 2019

[2] Infineon Z8F70273244 LITIX™ PowerFlex SPI low level driver AN Rev.1.00

Revision history

Document version	Date of release	Description of changes
Rev.1.00	2021-05-25	Initital release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-05-25

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2021 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Z8F80148615

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.