XMC1302 based eDisconnect Board Firmware

XMC1302 Firmware

Contact

* **Nyan Paing Soe (IFAP DC PSS SI PS INNO)**
* **Seet Wei Ren Ivan (IFAP DC PSS SI PS TM)**

About this document

# Scope and purpose

This document describes the firmware implementation on XMC1302 to control eDisconnect Board.

# Intended Audience

Anyone working with 2ED4820 gate driver and XMC1302.

Table of contents

[Contact 1](#_Toc148562287)

[About this document 1](#_Toc148562288)

[Table of contents 2](#_Toc148562289)

[1 Overview 4](#_Toc148562290)

[1.1 Block Diagram of eDisconnect switch 4](#_Toc148562291)

[1.2 Development tools 4](#_Toc148562292)

[1.3 External library 4](#_Toc148562293)

[2 Requirement and Specifications 5](#_Toc148562294)

[3 XMC1302 Interaface 6](#_Toc148562295)

[4 Dave Apps parameter settings 7](#_Toc148562296)

[4.1 List of Dave App 7](#_Toc148562297)

[4.2 APP Configuration and parameter settings 8](#_Toc148562298)

[4.2.1 Digital IO 8](#_Toc148562299)

[4.2.2 PWM CCU 4 8](#_Toc148562300)

[4.2.3 PWM\_0 10](#_Toc148562301)

[4.2.4 INTERRUPT 11](#_Toc148562302)

[4.2.5 ADC\_MEASUREMENT 11](#_Toc148562303)

[4.2.6 SYSTEM TIMER 13](#_Toc148562304)

[4.2.7 SPI COMMUNICATION 13](#_Toc148562305)

[5 Firmware Module 17](#_Toc148562306)

[5.1 SYS TIMERs and IRQ Handlers 17](#_Toc148562307)

[5.2 Operation State Diagram 18](#_Toc148562308)

[5.3 Protection 20](#_Toc148562309)

[5.4 SPI Communication and 2ED4820 Gate Driver register 21](#_Toc148562310)

[5.5 Firmware Structure in Dave IDE 22](#_Toc148562311)

[6 ADC sensing and processing 23](#_Toc148562312)

[6.1 Voltage Sensing 23](#_Toc148562313)

[6.2 Current sensing 24](#_Toc148562314)

[6.3 Temperture sensing 25](#_Toc148562315)

[7 System protection 27](#_Toc148562316)

[7.1 Precharge switch 27](#_Toc148562317)

[7.2 Freewheeling switch 29](#_Toc148562318)

[8 GUI 30](#_Toc148562319)

[9 Hardware Schematic 31](#_Toc148562320)

[References 34](#_Toc148562321)

[Revision history 35](#_Toc148562322)

[Figure 1 Block diagram of eDisconnect switch 4](#_Toc148562166)

[Figure 2 PWM CCU4 PWMA general setting 8](#_Toc148562167)

[Figure 3 Hardware connection of PWMA event to PWMB 9](#_Toc148562168)

[Figure 4 PWMB General setting 9](#_Toc148562169)

[Figure 5 PWMB External Event Setting 9](#_Toc148562170)

[Figure 6 PWM\_0 General setting 10](#_Toc148562171)

[Figure 7 PWM\_0 Event setting 10](#_Toc148562172)

[Figure 8 PWM\_0 Pin setting 10](#_Toc148562173)

[Figure 9 PWM\_0 Hardware connection to INTERRUPT\_1 irq 11](#_Toc148562174)

[Figure 10 INTERRUPT\_1 setting 11](#_Toc148562175)

[Figure 11 ADC\_MEASUREMENT\_0 General setting 12](#_Toc148562176)

[Figure 12 ADC\_MEASUREMENT\_0 Measurement setting 12](#_Toc148562177)

[Figure 13 ADC\_MEASUREMENT\_0 Interrupt setting 12](#_Toc148562178)

[Figure 14 SYSTIMER General setting 13](#_Toc148562179)

[Figure 15 SPI DRIVER General setting 13](#_Toc148562180)

[Figure 16 SPI DRIVER Advanced setting 14](#_Toc148562181)

[Figure 17 SPI DRIVER Interrupt setting 15](#_Toc148562182)

[Figure 18 SPI DRIVER Pin Setting 16](#_Toc148562183)

[Figure 19 Timing diagram for Handlers 17](#_Toc148562184)

[Figure 20 Block diagram for System timer and handler 18](#_Toc148562185)

[Figure 21 Operation state diagram 19](#_Toc148562186)

[Figure 22 Flowchart for system level protection 20](#_Toc148562187)

[Figure 23 2ED4820 Register overview 21](#_Toc148562188)

[Figure 24 Firmware structure in Dave IDE 22](#_Toc148562189)

[Figure 25 Voltage sensing for battery and load voltage 23](#_Toc148562190)

[Figure 26 Low side Shunt current sensor through 2ED4820 24](#_Toc148562191)

[Figure 27 Thermistor circuit 25](#_Toc148562192)

[Figure 28 Plot of Temperature vs ADC voltage and ADC values 25](#_Toc148562193)

[Figure 29 RT characteristics of B57861S 26](#_Toc148562194)

[Figure 30 Temperature and ADC range 26](#_Toc148562195)

[Figure 31 Precharge and Freewheel setting in GUI 27](#_Toc148562196)

[Figure 32 Flowchart for Precharge algorithm 28](#_Toc148562197)

[Figure 33 Timing diagram for Freewheel operation 29](#_Toc148562198)

[Figure 34 Captured freewheel signal (CH1: Freewheel signal, CH2: Load voltage, CH3: delay signal) 29](#_Toc148562199)

[Figure 35 GUI for eDisconnect switch in MicroInspector 30](#_Toc148562200)

# Overview

The eDisconnect board Firmware is developed to control Infineon microcontroller XMC 1302 and smart high side MOSFET gate driver 2ED4820 to operate high power switch at battery terminal. Flexible precharge method and protection are also implemented in this firmware.

## Block Diagram of eDisconnect switch

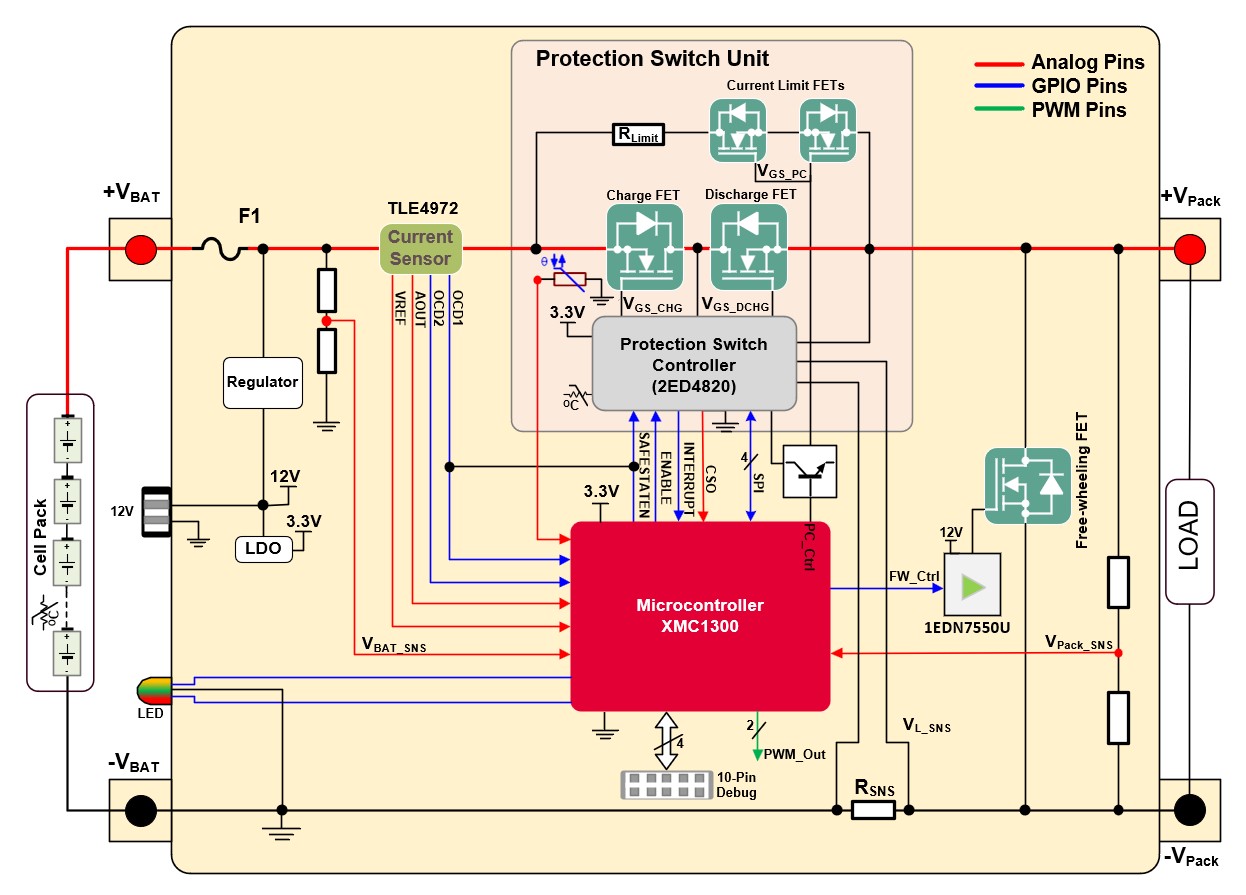


Figure Block diagram of eDisconnect switch

## Development tools

The firmware is developed with Dave IDE from Infineon developer center. The GUI is developed and operate with MicroInspector. XMC link debugger probe is required.

## External library

The firmware uses 2ED4820 Library which is mentioned in session 5.5.

# Requirement and Specifications

Table Specification of eDisconnetion switch

| Parameter | | Min. | Typ. | Max. | Peak | Unit | Note | Req. Type |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cell Voltage | | 2.6 | 3.6 | 4.2 | 4.25 | V | LG Li-ion  INR21700M50L 4800mAh |  |
| Voltage | 14S7P | 35 | 48 | 58.8 | - | V | Programmable to choose the battery Configuration | HW,SW |
| Current (cont.) | 14S7P | - | 54 | 100 | - | A | @ TJ,max = 100⁰C | HW,SW |
| Output Voltage | | 35 | 48 | 58.8 | - | V |  | HW |
| Precharge Time | | - | 10 | 1000 | - | ms |  | SW |
| Precharge Current | | - | 5 | - | 33.6 | A |  | SW, HW |
| Over Voltage (OV) | | - | - | 60 | - | V |  | SW |
| Under Voltage (UV) | | - | - | 30 | - | V |  | SW |
| Over Current (OC) | | - | - | 130 | - | A | 130% of 3C Discharge | SW |
| Transient Current (TC) | | - | - | 150 |  | A | 150% @ t=200us | SW |
| Short Circuit Current | | 150 | - | 3000 | TBD | A |  | SW |
| Temperature | | 15 | - | 80 | 100 | °C |  | SW,HW |
| Response time | | 1 | - | 50 | TBD | us |  | SW |
| VAUX | | 3.3 | 3.3 | 5 | - | V | VDD for CS, MCU | HW |
| Gate drive current | | - | - | 1A | TBD | A |  | HW |
| SPI/RS485 | |  | | | | | CS gain, OC threshold, GD control, diagnosis | SW |
| JTAG | | 1.27mm 10-pin debug connector | | | | | Microcontroller Interface | SW |
| LED | | RED, GREEN/YELLOW, AMBER | | | | | 3-pin RGB for indications | HW,SW |
| Heat Sink (YES/~~NO~~) | | Rth = 5K/W (approx) | | | | | ULP/LPD80/N80/Al | HW |
| Size | | Max: 86mm x 41mm x 33mm | | | | |  |  |

# XMC1302 Interaface

Table XMC1302 Pin out

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | Port | Signal | Description |
| 1 | P2\_6 | OCD2\_GPIO | For TLE4972 current programmer (Not use in V2.2) |
| 2 | P2\_7 | OCD1\_GPIO | For TLE4972 current programmer (Not use in V2.2) |
| 3 | P2\_8 | NTC\_ADC | Thermistor reading |
| 4 | P2\_9 | I\_SNS\_ADC | Current sense from 2ED4820 |
| 5 | P2\_10 | VBAT\_SNS\_ADC | Battery voltage |
| 6 | P2\_11 | VL\_SNS\_ADC | Load voltage |
| 7 | VSS | SGND | Ground |
| 8 | VDD | VDDP/VDD | 3.3V |
| 9 | P1\_3 | FW\_Ctrl | Freewheeling control signal |
| 10 | P1\_2 | PWM\_A | Freewheeling timer delay (output for debug only) |
| 11 | P1\_1 | PC\_Ctrl | Precharge control signal |
| 12 | P1\_0 | PWM\_B | Not used |
| 13 | P0\_0 | SAF\_GPIO | Safestate Enable pin to 2ED4820 |
| 14 | P0\_4 | EN\_GPIO | Enable pin to 2ED4820 |
| 15 | P0\_5 | PX. RESET | Reset pin |
| 16 | P0\_6 | MOSI\_GPIO | SPI communication |
| 17 | P0\_7 | MISO\_GPIO | SPI communication |
| 18 | P0\_8 | SCLK\_GPIO | SPI communication |
| 19 | P0\_9 | CSN\_GPIO | SPI communication |
| 20 | P0\_10 | PX\_LED\_R | Red status LED |
| 21 | P0\_12 | PX.LED.G | Green status LED |
| 22 | P0\_13 |  |  |
| 23 | P0\_14 | SWIO\_uC | Debugger date |
| 24 | P0\_15 | SWCLK\_uC | Debugger clock |
| 27 | P2.1 | VREF\_ADC | For TLE4972 current programmer (Not use in V2.2) |

# Dave Apps parameter settings

## List of Dave App

The following table lists the number of Dave app used in firmware. Whenever the changes are made in these app, it is required to generate the code first before compiling main firmware.

Table List of Dave app used in this firmware v 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| Instance Label | Dave App | Output Pin | Description |
| PX\_LED\_G | DIGITAL\_IO | P0.12 | Green LED Indicator |
| PX\_LED\_R | DIGITAL\_IO | P0.10 | Red LED Indicator |
| IO\_DRIVER\_ENABLE | DIGITAL\_IO | P1.0 | 2ED4820 enable |
| IO\_DRIVER\_SAFESTATEEN | DIGITAL\_IO | P0.4 | 2ED4820 Safestate enable |
| PWMA | PWM\_CCU4 | P1.2 | Short timer before freewheel signal is ON. (~50us) |
| PWMB | PWM\_CCU4 | P1.3 | Freesheel output pulse after switch is off |
| PWM\_0 | PWM | P2.0 | Precharge Signal |
| *TIMER\_0* | *TIMER* |  | *1ms timer (no longer used in v2.2) \*\** |
| *INTERRUPT\_0* | *INTERRUPT* |  | *timer 0 interrupt (no longer used in v2.2) \*\** |
| INTERRUPT\_1 | INTERRUPT |  | PWM\_0 periodic event |
| PIN\_INT\_DRIVER | PIN INTERRUPT | P2.0 | Fault Interrupt from 2ED4820 |
| ADC\_MEASUREMENT\_0\_CH0 | ADC\_MEASUREMENT | P2.9 | Current sensor |
| ADC\_MEASUREMENT\_0\_CH1 | ADC\_MEASUREMENT | P2.8 | Temperature sensor |
| ADC\_MEASUREMENT\_0\_CH2 | ADC\_MEASUREMENT | P2.10 | Battery voltage |
| ADC\_MEASUREMENT\_0\_CH3 | ADC\_MEASUREMENT | P2.11 | Load (Pack) voltage |
| ADC\_MEASUREMENT\_0\_CH4 | ADC\_MEASUREMENT | P2.1 | Vref (current sensor) |
| SYSTIMER\_0 | SYSTIMER |  | 3 system timers (5ms, 10ms and 50ms) |
| SPI\_DRIVER (MISO) | SPI\_MASTER | P0.7 | SPI MISO |
| SPI\_DRIVER (MOSI) | SPI\_MASTER | P0.6 | SPI MOSI |
| SPI\_DRIVER (SCLKOUT) | SPI\_MASTER | P0.8 | SPI CLK |
| SPI\_DRIVER (Slave Select) | SPI\_MASTER | P0.9 | SPI Slave select |

*\*\* It is possible to remove in upcoming version*

## APP Configuration and parameter settings

### Digital IO

Table Digital IO parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Digital IO | Pin Direction | Mode | Init Level |
| PX\_LED\_G | Input/Output | Push Pull | Low |
| PX\_LED\_R | Input/Output | Push Pull | Low |
| IO\_DRIVER\_ENABLE | Input/Output | Push Pull | Low |
| IO\_DRIVER\_SAFESTATEEN | Input/Output | Push Pull | High |

### PWM CCU 4

PWM CCU 4 module has two PWM signals (PWM A and PWM B) for freewheeling implementation. Both PWM A and PWM B are configured with single shot mode operation as it only requires one pulse after MOSFET switch is off. PWM A serves as delay timer 50us during MOSFET turn off. The falling edge of PWM A creates external event to trigger PWM B for freewheeling action.

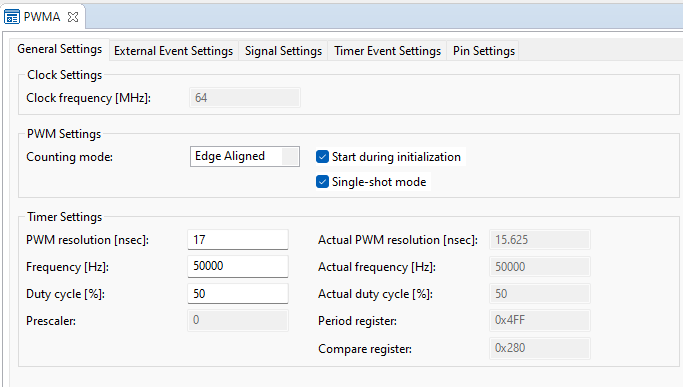


Figure PWM CCU4 PWMA general setting

A screenshot of a computer

Description automatically generated

Figure Hardware connection of PWMA event to PWMB

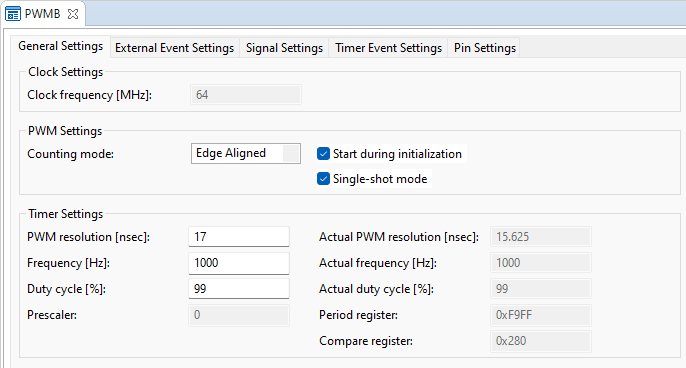


Figure PWMB General setting

A screenshot of a computer program

Description automatically generated

Figure PWMB External Event Setting

### PWM\_0

PWM\_0 signal is produced for Pre charge operation. The signal is fed to linear MOSEFT to turn on with signle pulse mode or PWM mode. The pre charge mode can be selected for different application requirement. In event settings, PWM\_0 has periodic match interrupt connected to INTERRUPT\_1.

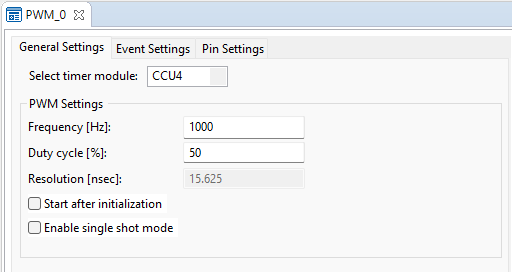


Figure PWM\_0 General setting

A screenshot of a computer

Description automatically generated

Figure PWM\_0 Event setting

A screenshot of a computer

Description automatically generated

Figure PWM\_0 Pin setting

A screenshot of a computer

Description automatically generated

Figure PWM\_0 Hardware connection to INTERRUPT\_1 irq

### INTERRUPT

This INTERRUPT\_1 app called when PWM\_0 periodic match event occurs. The function UserIRQHandler is called for PWM\_0 event clearance and pulse counting for Pre charge function.

A screenshot of a computer

Description automatically generated

Figure INTERRUPT\_1 setting

### ADC\_MEASUREMENT

ADC measurement is set up with continuous conversion. The desired sampling is set as 1000 ns. There are 5 ADC channels for measurement of the following.

Table ADC measurement signals and ports

|  |  |  |
| --- | --- | --- |
| ADC Signals | Port | Value |
| VBAT\_SNS | P2.10 | Battery Voltage |
| VL\_SNS | P2.11 | Load (Pack) voltage |
| I\_SNS\_ADC | P2.9 | Current sensor (2ED4820) |
| VREF\_ADC | P2.1 | REF voltage for TLE4972 current sensor |
| NTC\_ADC | P2.8 | Temperature sensor |

At the end of measurement, ADC\_measurement handler interrupt is called to notify the new meansurment are available.

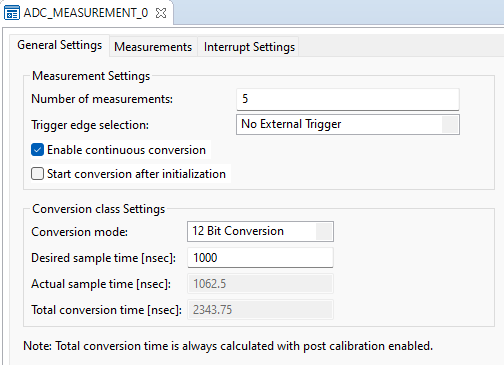


Figure ADC\_MEASUREMENT\_0 General setting

A screenshot of a computer

Description automatically generated

Figure ADC\_MEASUREMENT\_0 Measurement setting

A screenshot of a computer

Description automatically generated

Figure ADC\_MEASUREMENT\_0 Interrupt setting

### SYSTEM TIMER

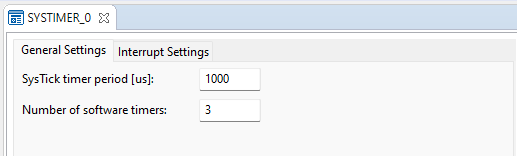


Figure SYSTIMER General setting

### SPI COMMUNICATION

A screenshot of a computer program

Description automatically generated

Figure SPI DRIVER General setting

A screenshot of a computer

Description automatically generated

Figure SPI DRIVER Advanced setting

A screenshot of a computer

Description automatically generated

Figure SPI DRIVER Interrupt setting

A screenshot of a computer

Description automatically generated

Figure SPI DRIVER Pin Setting

# Firmware Module

## SYS TIMERs and IRQ Handlers

The system timer is set up with 1ms tick. There are 3 system timers which are called periodically to impletement the entire firmware module. Each timer is assigned to call different handlers.

Table System Timer

|  |  |  |  |
| --- | --- | --- | --- |
| SYSTIMER ID | Timer event mode | Timer value | SYSTIMER Call Handler |
| TimerId\_1 | Periodic | 10ms | Operation |
| TimerId\_2 | Periodic | 50ms | GUI |
| TimerId\_3 | Periodic | 5ms | Fault |

A diagram of a diagram

Description automatically generated with medium confidence

Figure Timing diagram for Handlers

The fault handler is called at every 5ms and check the status of overall system quickly and update LED status for monitoring.

The operation handler is called at every 10 ms. It is the core of the eDisconnect switch operation. Due to continuous conversion of ADC, the new raw values are available at every handler call. The measurement values are processed to get actual reading (voltage, current, etc) in this routine. Besides, the operation state diagram (5.2) is also implemented.

The GUI handler is called at every 50ms. The handler reads the current button states and values from GUI and updated. It also provides the function to pass those button state to operation handlers for making decision in the state diagram. It also extracts data from the other handlers and prepare the values to send so that GUI can display as required.

A diagram of a system

Description automatically generated

Figure Block diagram for System timer and handler

## Operation State Diagram

There are 4 operation states depending on the enable switch, safestate, user input switch on/off and fault.

Init State, OpState = 0: Whenever the enable signal goes low, the system will be in init state. In the version 2.2, the state is bypassed (ie, enable = 1).

OFF state, OpState = 1: The eDisconnect switch is OFF in this state. The freewheel protection is included. In the event of switch off, freewheel enable and completion is checked. When the freewheel is enable and complete flag is false, the freewheel protection is executed first before sending switchoff signal to MOSFET gate driver via register. The freewheel completion flag is set to 1 if the switch is successfully turned off. When freewheel protection is disabled, it will be switch off immediately. Unless there is fault or user input to turn on, the switch will remain OFF state.

On state, Opstate = 2: The eDisconnect switch can be turned on by user switch input. During switch on event, the precharge enable and completion is checked. Precharge configuration is called and calculate the maximum pulses needed for precharge operation according to the user settings. When precharge mode is enabled and completion flag is false, the precharge operation is carried out before the turning on the MOSFET. Unless there is fault or user input to turn off, the switch will remain ON state.

Fault state, Opstate = 3: The eDisconnect switch will go to OFF state immediately after the fault trigger flag is 1 or problem detected in gate driver. During the fault state, the switch is forced to turned off. The user is not allowed to switch on until fault is cleared and returned to OFF state.

A diagram of a problem

Description automatically generated

Figure Operation state diagram

## Protection

The firmware implements several protections in the system level such as overvoltage, undervoltage and over/under temperature. The fault check function is called at every 5 ms.

To determine the fault condition, the maximum and minimum threshold values can be referred to the specification table in session 2.



Figure Flowchart for system level protection

## SPI Communication and 2ED4820 Gate Driver register

The SPI slave is provided by 2ED4820 gate driver. Therefore, in this firmware, SPI master module must be used with 16-bit protocol. The SPI read and write functions are provided by DRIVER\_2ED4820EM library. The SPI master is set up with Dave app. The parameters and configurations are described in session 4.2.7. The operation and diagnosis of the gate driver is done by accessing and updating the register content via SPI communication.

The Register STDIAG, CHDIAG and DIAG is read only diagnosis register. When interrupt pin is set to 1, microcontroller reads the status of these registers and updates in GUI. The microcontroller then cleans the failure by setting register FAILURE CLEAN and check if the bit 7 of STDIAG is cleared. Otherwise, the eDisconnect switch is set to fault state.

A close-up of a table

Description automatically generated

Figure 2ED4820 Register overview

The eDisconnect switch can be turned and off respectively by setting and clearing of MOSONCH\_A and MOSONCH\_B of MOS\_CHS\_CTRL register. All the register bits can be set by using the function call provided by gate driver library.

## Firmware Structure in Dave IDE

A screenshot of a computer program

Description automatically generated

Figure Firmware structure in Dave IDE

# ADC sensing and processing

## Voltage Sensing

The battery voltage (VBAT\_SNS\_ADC) and load voltage (VL\_SNS\_ADC) are sensed with voltage divider network.

A diagram of a computer

Description automatically generatedA diagram of a network

Description automatically generated

Figure Voltage sensing for battery and load voltage

The resistor divider value is 2.7k/(2.7k+51k) = 0.05.

To measure 48V, the voltage at VBAT\_SNS\_ADC is (48x0.05) = 2.4V which corresponds to ADC value 2978 for 12- bit ADC. Therefore, to calculate gain to convert voltage reading in microcontroller,

Voltage gain = (3.3/4096)/0.05 = **0.016**

A black and white text

Description automatically generated

## Current sensing

The lower-side current sense configuration is used in this eDisconnect switch. The shunt value is 0.05mOhm. The shunt resistor voltage is filtered and amplified by the internal amplifier provided by 2ED4820. The CSO value is connected to ADC of microcontroller through RC low pass filter with value 1kOhm and 10nF.

A diagram of a computer

Description automatically generated

Figure Low side Shunt current sensor through 2ED4820

The current value is computed by the following steps.

1. // IADC: 0 to 4096(12 bits)
2. // CSAG = Gdiff = 40V/V
3. // Rshunt= 50uOhms
4. // Moving Average Filter (Shunt Current) N=50
5. //Ki = gain adjustment

Vcso is computed with ADC gain and the shunt voltage is calculated by current sense gain CSAG = 40 configured in gate driver register. The shunt current is computed with Rshunt = 50uOhm. The computed value is furthered smoothed by using moving average filter with N = 50. The arbitrary value Ki is the gain adjustment to be made experimentally in this firmware version.

## Temperture sensing

B57861S NTC thermistor is used to sense ambient temperature around eDisconnect switch. The 10k NTC is connected to 10k pull up resistor which form voltage divider circuit at NTC\_ADC. The lookup table is developed to translate the measured ADC values into temperature reading to show in GUI.

A diagram of a computer

Description automatically generated

Figure Thermistor circuit

The charactesistics of NTC is plotted with normalized RT/R25 vs temperature.

The ADC values are then computed according to analog circuit network for the temperature value range from 0 degree to 155 degree Celsius. A good linearity is observed from 0 degree to 60 degree Celsius.

A graph with a blue line

Description automatically generatedA graph with a line

Description automatically generated

Figure Plot of Temperature vs ADC voltage and ADC values

A table of numbers with numbers

Description automatically generated A graph of a line

Description automatically generated

Figure RT characteristics of B57861S

A table of numbers and numbers

Description automatically generatedFigure Temperature and ADC range

# System protection

## Precharge switch

The precharge switch is enabled by default and it can be disabled by user via GUI. There are two precharge mode implemented in this firmware. The signle pulse mode or PWM mode can be selected. The selection table and GUI is shown.

Table Precharge mode selection

|  |  |  |  |
| --- | --- | --- | --- |
| Precharge | Mode | Frequency (Hz) | Duration (ms) |
| Enable | Single pulse | NA | 100 |
| NA | 250 |
| NA | 500 |
| PWM | 10 | 100 |
| 10 | 500 |
| 10 | 1000 |
| 50 | 100 |
| 50 | 500 |
| 50 | 1000 |
| 100 | 100 |
| 100 | 500 |
| 100 | 1000 |
| Disable | NA | NA | NA |

A screenshot of a computer

Description automatically generated

Figure Precharge and Freewheel setting in GUI

The precharge algorithm is called in the operation ON state (OpState = 2). When Precharge enable button is turned ON, the precharge completion flag is checked. If the flag is not set, the precharge algorithm is called. The condition for precharge completion is determined by the maximum pulse count generated by PWM\_0. This is applicable for both single pulse and PWM mode. Depending on the selection, the maximum pulse count is determined by precharge configuration function. The precharge failure check is not implemented in this firmware version 2.2 and before.



Figure Flowchart for Precharge algorithm

## Freewheeling switch

The freewheeling is enabled by default. It can be disabled by user via the freewheel enable button in GUI. The freewheel routine is called during switch off operation (OpState = 1).



Figure Timing diagram for Freewheel operation

When the MOSFET is turned off, after brief duration of delay (50us), the freewheel signal pulse is generated for 500us. The two PWM CCU4 is used for this operation. Firstly, PWMA is generated with predefined frequency and duty cycle to achieve delay of 50us. The PWMA falling event is captured and activate the start of PWMB for another 500 us.

A screen shot of a computer

Description automatically generated

Figure Captured freewheel signal (CH1: Freewheel signal, CH2: Load voltage, CH3: delay signal)

It should be noted that the PWMA pulse time does not represent actual delay but less than 50us. To achieve delay approximately 50us, the 50kHz frequeny with 50% duty is used.

# GUI

The user interface GUI is built with microInspector. To use GUI, XMC link isolated debugger is required. The system state shows the battery voltage, load voltage, current, and temperature value. The user can use SW ON/OFF button to turn on and off eDisconnect switch. LED RED and GREEN are displayed to align the LED status in the PCB board. Precharge and freewheel setting is also included in the GUI.

A screenshot of a device

Description automatically generated

Figure GUI for eDisconnect switch in MicroInspector

# Hardware Schematic

A computer diagram of a computer

Description automatically generated with medium confidence

A diagram of a computer

Description automatically generated

A computer screen shot of a diagram

Description automatically generated

# References

[1]

Revision history

| Document version | Date of release | Description of changes |
| --- | --- | --- |
| V1.0 | 2023-10-18 | First draft version |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |