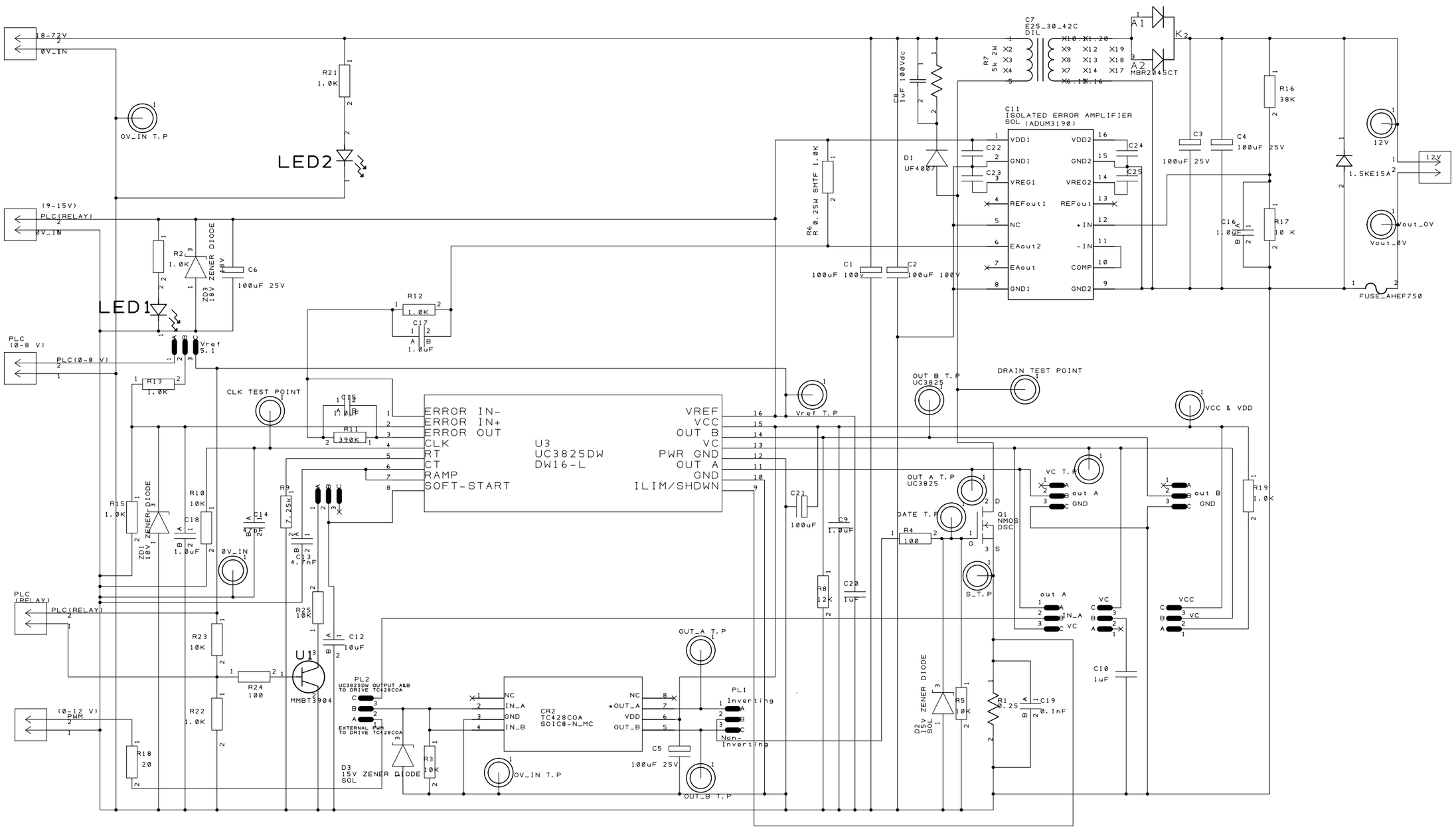
Appendix A: Flyback converter Schematic Diagram


18-72V
0V_IN
OV_IN T.P
R21
1.0K
LED2
(9-15V)
PLC(RELAY)
0V_IN
R2
1.0K
ZD3
10V ZENER DIODE
C6
100uF 25V
LED1
PLC
(0-8 V)
PLC(0-8 V)
Vref
5.1
R13
1.0K
CLK TEST POINT
R12
1.0K
C17
1.0uF
R6
0.25W SMTF 1.0K
C8
1uF 100Vdc
C7
E25_30_42C
DIL
R7
5K 2W
A1
A2
MBR2045CT
D1
UF4007
C11
ISOLATED ERROR AMPLIFIER
SOL (ADUM3190)
VDD1
GND1
VREG1
REFout1
NC
EAout2
EAout
GND1
VDD2
GND2
VREG2
REFout
+IN
-IN
COMP
GND2
C22
C23
C24
C25
C1
100uF 100V
C2
100uF 100V
C3
100uF 25V
C4
100uF 25V
R16
38K
C16
1.0uF
R17
10 K
12V
1.5KE15A
Vout_OV
Vout_0V
FUSE_AHEF750
ERROR IN-
ERROR IN+
ERROR OUT
CLK
RT
CT
RAMP
SOFT-START
U3
UC3825DW
DW16-L
VREF
VCC
OUT B
VC
PWR GND
OUT A
GND
ILIM/SHDWN
C15
1.0uF
R11
390K
Vref T.P
OUT B T.P
UC3825
DRAIN TEST POINT
VCC & VDD
OUT A T.P
UC3825
GATE T.P
Q1
NMOS
DSC
S_T.P
VC T.P
out A
GND
out B
GND
R19
1.0K
C21
100uF
C9
1.0uF
R4
100
R8
12K
C20
1uF
R15
1.0K
ZD1
10V ZENER DIODE
R10
10K
C18
1.0uF
0V_IN
R9
7.25K
C14
47pF
C13
4.7nF
R20
10K
C12
10uF
U1
MMBT3904
R23
10K
R24
100
R22
1.0K
PLC
(RELAY)
PLC(RELAY)
(0-12 V)
PWM
R18
20
PL2
UC3825DW OUTPUT A&B
TO DRIVE TC428COA
EXTERNAL PWM
TO DRIVE TC428COA
D3
15V ZENER DIODE
SOL
R3
10K
CR2
TC428COA
SOIC8-N_MC
NC
IN_A
GND
IN_B
*OUT_A
VDD
OUT_B
OUT_A T.P
OV_IN T.P
C5
100uF 25V
OUT_B T.P
PL1
Inverting
Non-
inverting
D2
15V ZENER DIODE
SOL
R5
10K
R1
0.25
C19
0.1nF
C10
1uF
VC
VCC
PROJECT:
DRAWING TITLE:
DRAWN BY:
REVISION:
SHEET: OF
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB

Appendix B:

Flyback converter Operation and Configuration

1. Introduction

An existing flyback converter was redesigned in order to improve the performance and the hardware protection of the Picogrid storage and source nodes. This report will detail the changes made as well as communicate the performance results of the improved flyback converter. A prototype flyback converter was constructed on a breadboard in order to ease the design as components could be changed with ease. This not only made testing easier but also resulted in to a more efficient flyback converter PCB.

1.1. Flyback structural overview

Figure 1 shows the converter breadboard prototype and figure 2 shows the PCB.

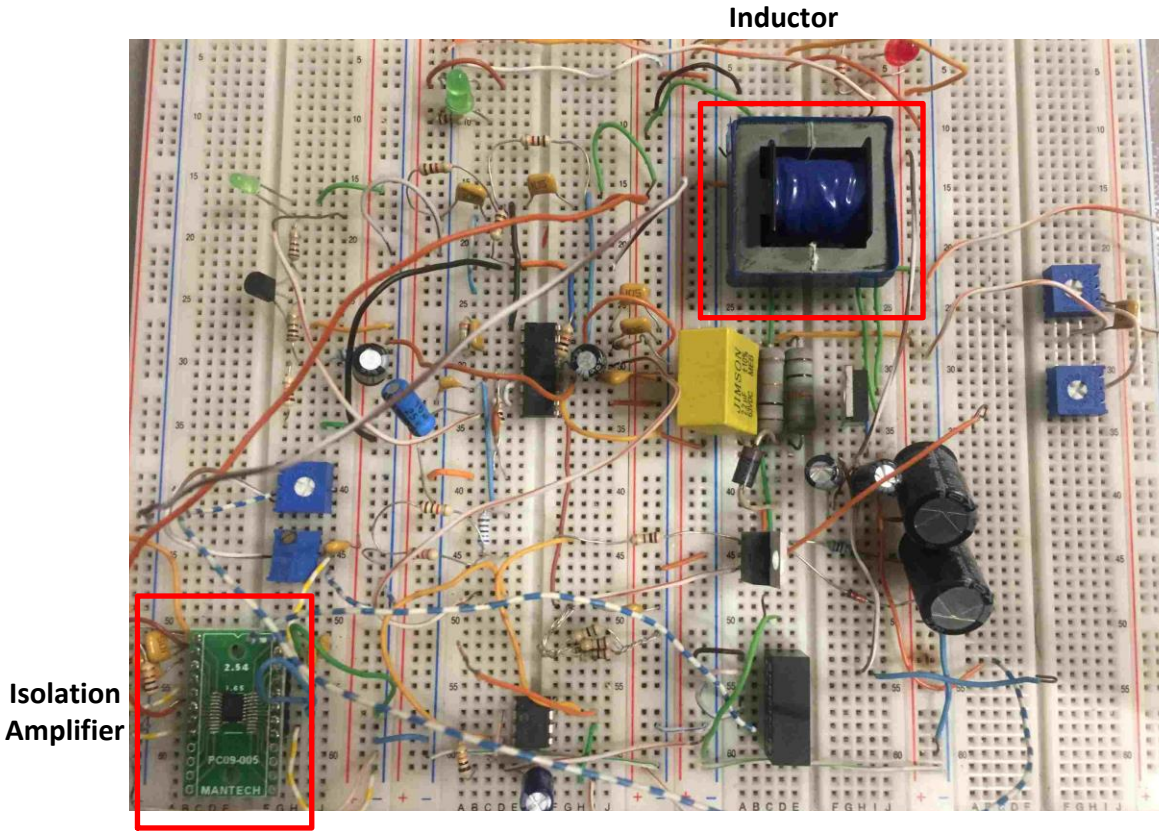


Figure 1: Flyback converter breadboard circuit

The above breadboard prototype was used to perform all tests before conversion in to a PCB layout.

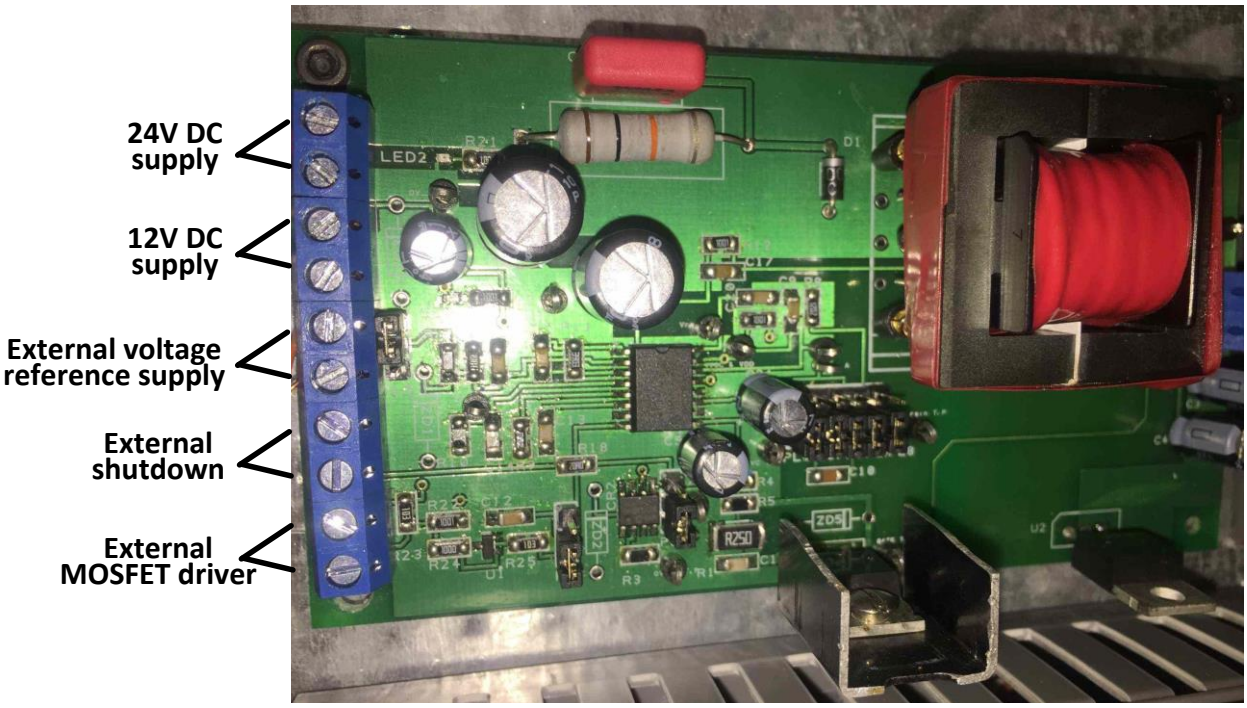


Figure 2: Flyback converter PCB circuit

1.2. Principles of Operation

A flyback converter is based on the buck boost converter topology but with an isolation transformer as opposed to the buck boost inductor [41-44]. It is the most commonly used SMPS for low power applications where the output voltage needs to be isolated from the primary input voltage. As shown in figure 3, its main components are the switching MOSFET, inductor, primary voltage supply, diode D and capacitor C.

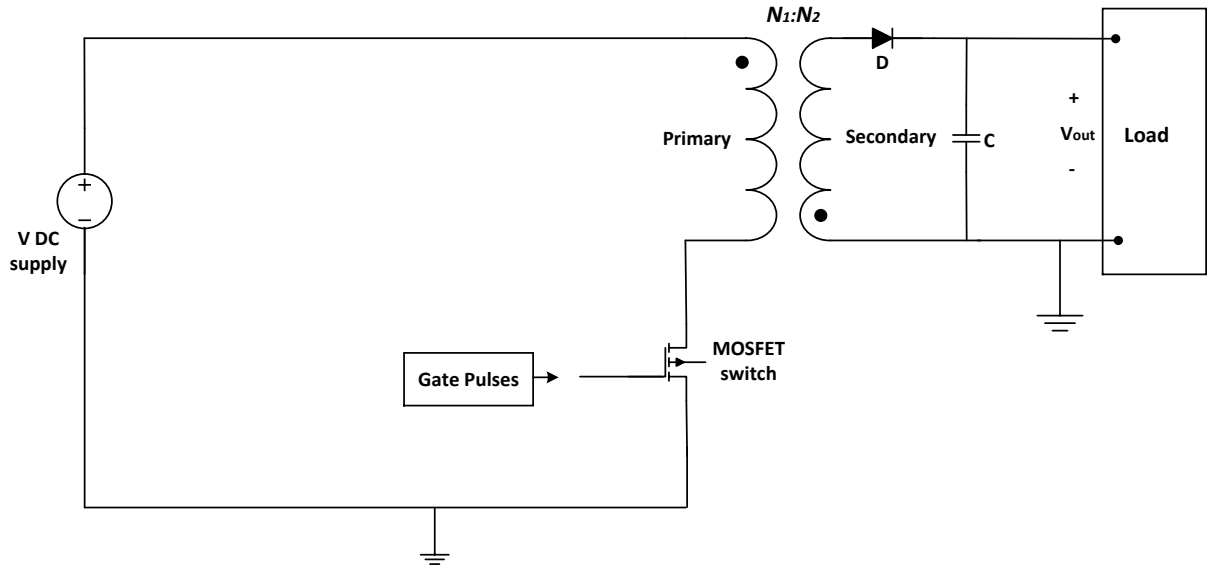


Figure 3: Flyback converter circuit

The converter has two modes of operation which are determined by the gate pulses form a PWM controller that switch the MOSFET ON and OFF. As shown in figure 4 (a) the converter stores energy in its inductive transformer during the ON time of the switching MOSFET and transfers it to the secondary output load side during the OFF time. Energy is however not transferred to the secondary output during the ON period as diode D is reverse biased. When the MOSFET is OFF as shown in figure 4 (b), current stops flowing in the primary side and according to the laws of magnetic induction, this results to a reversal of voltage polarity across the windings. This forward biases the diode and as a result allows current through to charge the output capacitor.

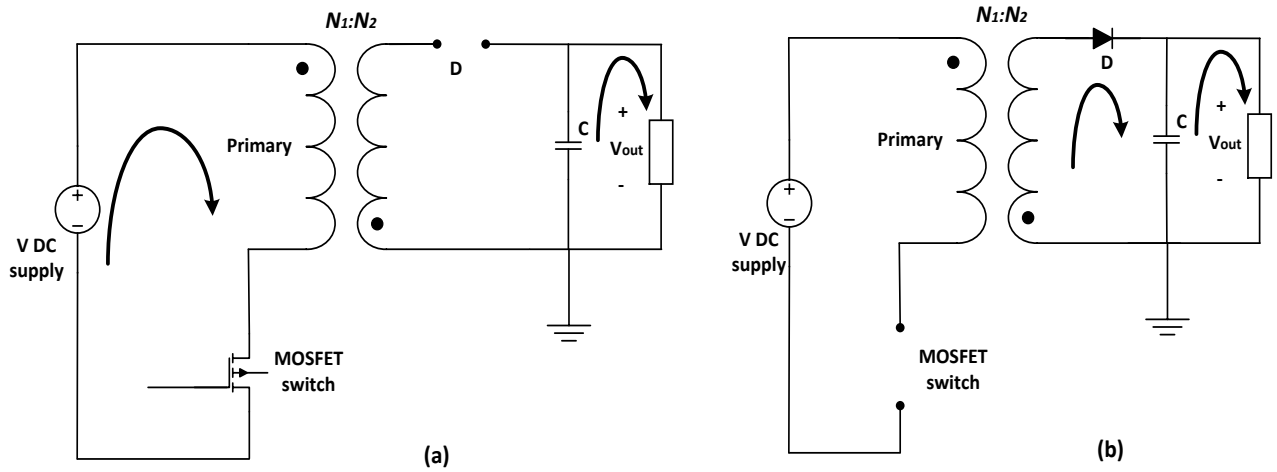


Figure 4: Flyback converter equivalent circuit for different modes; (a) MOSFET ON and (b) MOSFET OFF periods

A flyback converter's inductor-transformer must be designed so that it either maintains a continuous mode of operation or switches to discontinuous mode when under operation. Continuous mode of operation allows for a wide load variation whilst maintaining a steady voltage, whereas discontinuous mode starts degrading the output voltage with the increasing load current. However, the research was limited to using the existing flyback inductors and hence load current limiting was done using CCM resistor discussed in Appendix C.

Figure 5 shows a graph of the output load voltage (PBV) versus a varying load current of a picogrid's flyback converter when at a stable PBV of 12V. Continuous mode of operation occurs when the load current is kept below 1A at steady PBV values and above that, the flyback switches to discontinuous mode, with the current still rising but the PBV dropping steadily.

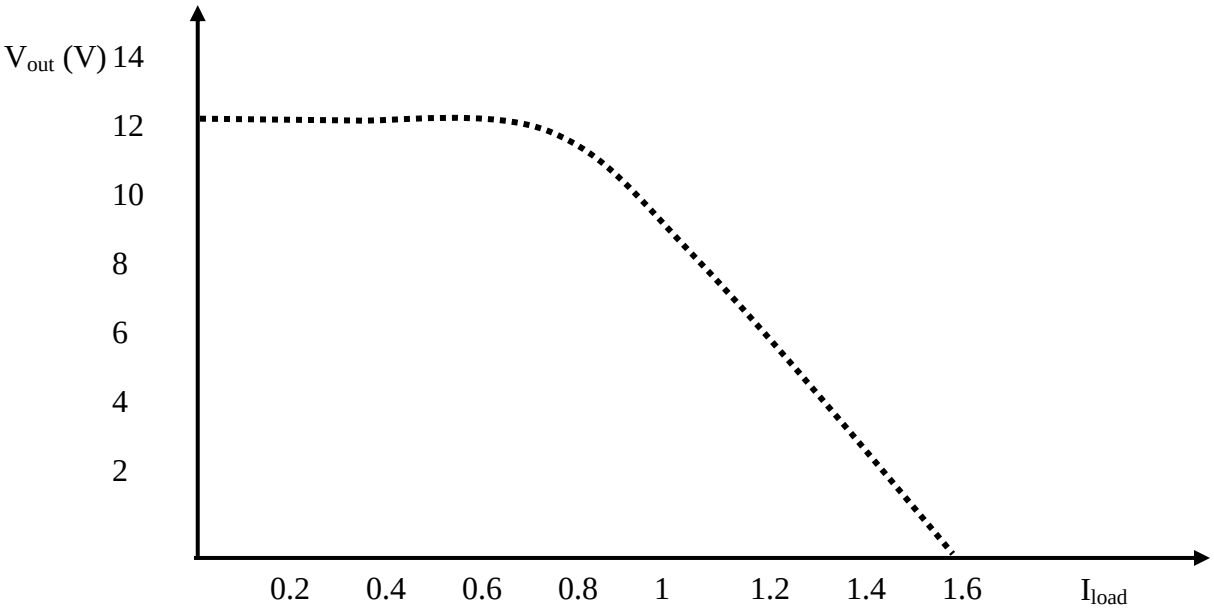


Figure 5: Flyback output voltage versus load current (with load varying between 0-50Ω)

1.3. Operation modes

There are five different modes of operation for the flyback converter circuit.

i. Internal and External PWM signal selection
ii. Internal PWM controller outputs driving the switching circuit at less than 50% duty cycle
iii. Internal PWM controller output driving the MOSFET at more than 50% duty cycle
iv. Internal and external control reference voltage selection
v. External shutdown of the PWM controller

The first mode offers an option of using an external PWM source or the internal PWM controller. Mode two and three are related to the Pulse Width Modulation (PWM) controller's output responsible for controlling the duty ratio of the switching circuit, whereas mode four allows for a selection between internal and external control signals to be used for driving the switching circuit. Mode five allows for the shutdown of the PWM control chip through an external control signal.

All the modes are achieved by using jumpers on the different pins which are labelled as PL1 to PL8 of the overall DC-DC converter circuit presented in Appendix B. This circuit shows the default location of all jumpers for the normal operation of the circuit. For clarity, the overall circuit will be zoomed in for each mode and the jumpers will be marked in red.

1.4. External and internal error reference voltage

The flyback converter was designed to have both constant and variable reference voltage to the error amplifier for flexibility by allowing other external mode of controls to operate the PWM

controller. As shown in figure 6, a jumper can be used to select between the internal constant 5V input or allow for an external input from PLCs ranging from 0-8V DC.

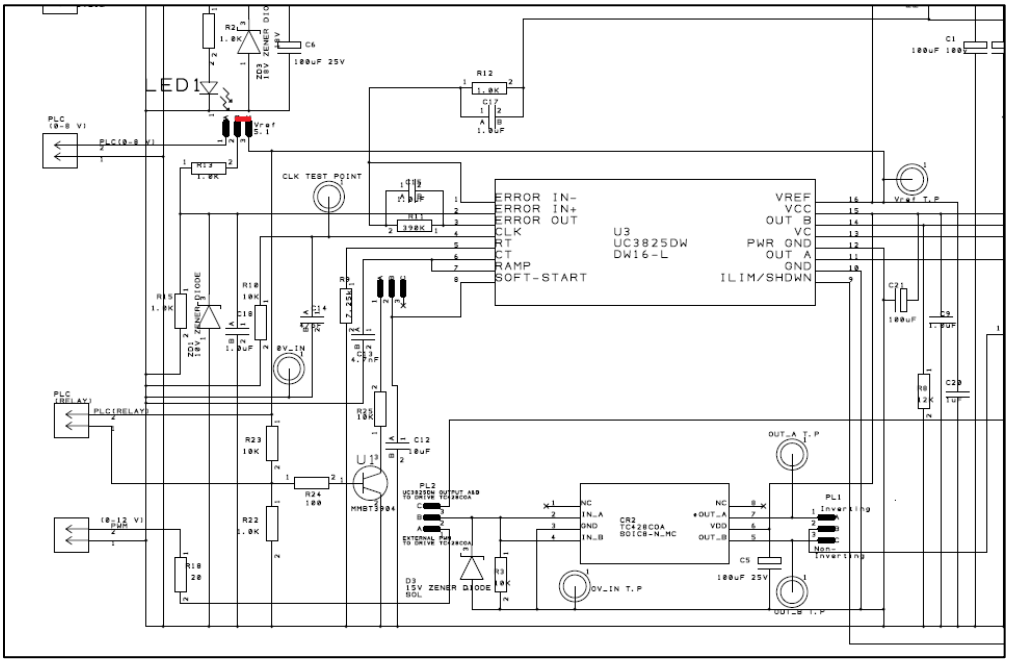


Figure 6: Internal reference voltage selection

1.5. Internal and external PWM selection

In order to use the internal PWM controller, the jumper is placed on pin PL2 of the overall circuit as shown in figure 7 below. Moving the jumper in the next location allows for an external PWM source to be used for driving the switching circuit.

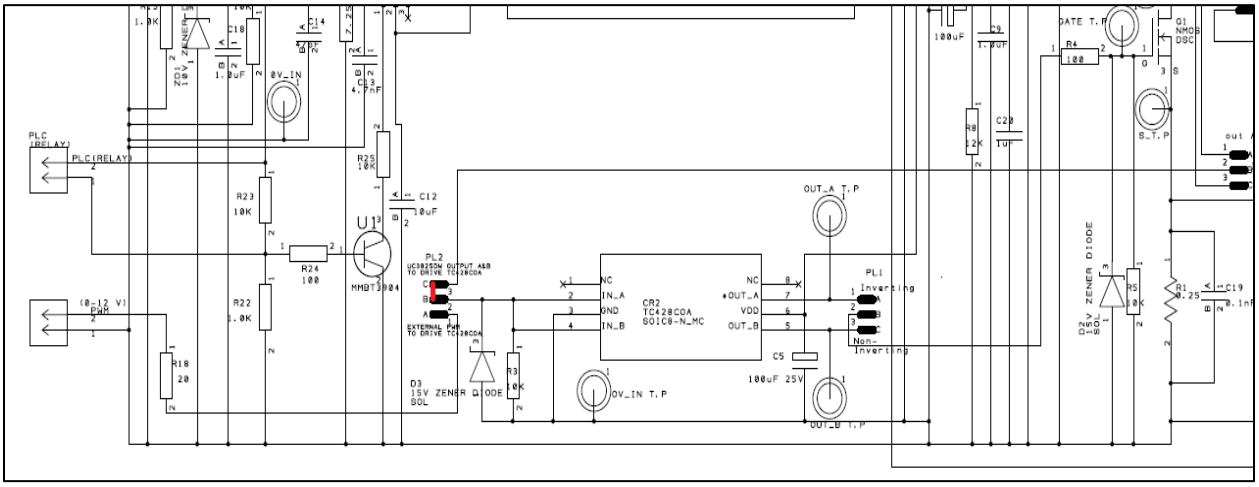


Figure 7: Internal PWM controller selection (shown in red)

1.6. External shutdown of the PWM controller

For safety purposes, the PWM controller can be shut down through an external control signal by placing a jumper on pin PL2 as shown by the red line in figure 8 below. The jumper is however

placed on the other position of pin PL2 by default so as to avoid unnecessary shutdowns from noise signals.

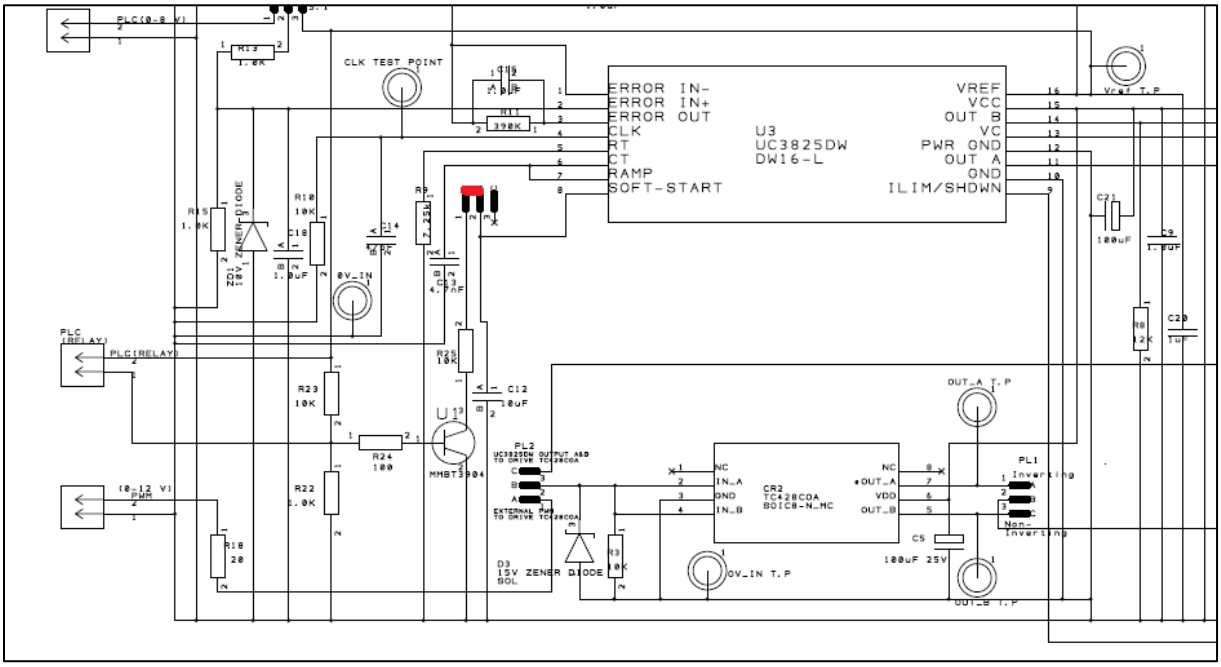


Figure 8: Jumper position for shutting down the PWM controller

2. Flyback Converter Configuration

There are two modes of configuration for the flyback converter;

i. Low current output mode which ensures a duty cycle of less than 50% from the PWM controller.
ii. A high current output guaranteeing more than 50% duty cycle from the PWM controller

2.1. Low Current output of less than 50% duty cycle

As shown in figure 9, this configuration utilises output A for driving the MOSFET at less than 50% duty cycle. Referring to the PWM controller's datasheet [41], the chip offers two high totem pole outputs for driving the switching circuit of the flyback converter. This mode is achieved on the circuit by having five jumpers placed in the configuration shown on the zoomed circuit figure 9 below.

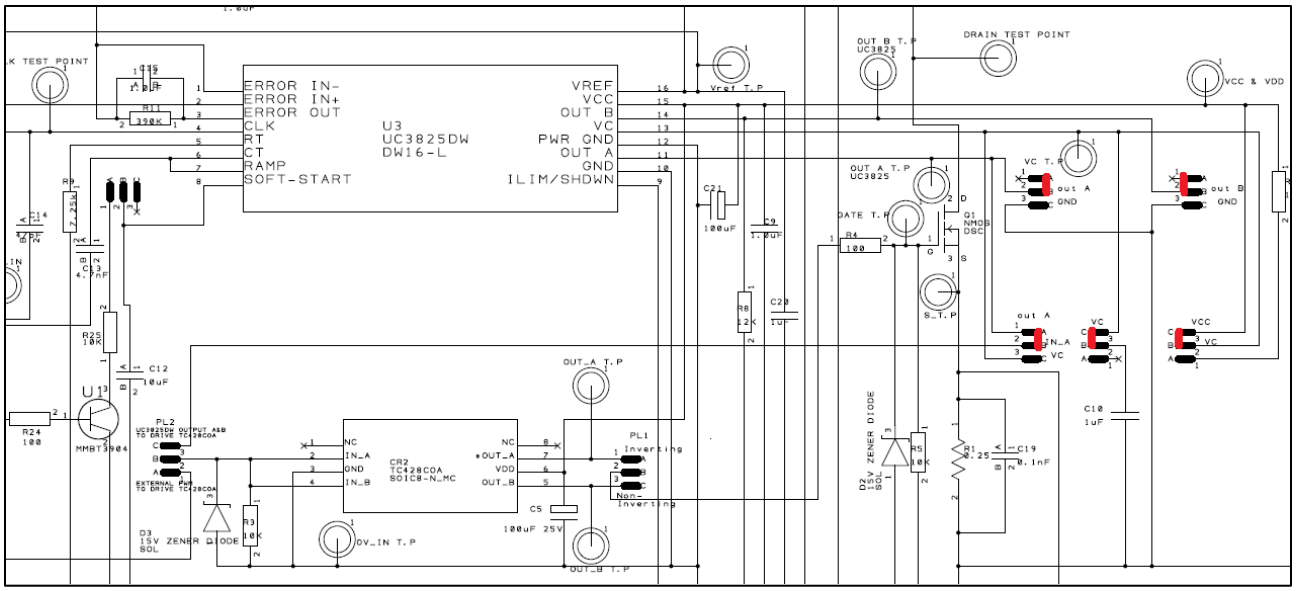


Figure 9: Zoomed circuit diagram showing jumper location (in red) for low duty cycle

2.2. PWM controller driving the switching circuit at more than 50% duty cycle

This mode gives the PWM chip a duty cycle of more than 70% by using the power supply pin (pin Vc) of the PWM controller chip [45] as the output stage for the switching driving circuit. This mode is achieved by grounding the PWM outputs A and B as well as bypassing the supply pin (Vc) by moving the five jumpers to the positions shown in figure 10 below. This is the chosen standard mode of operation.

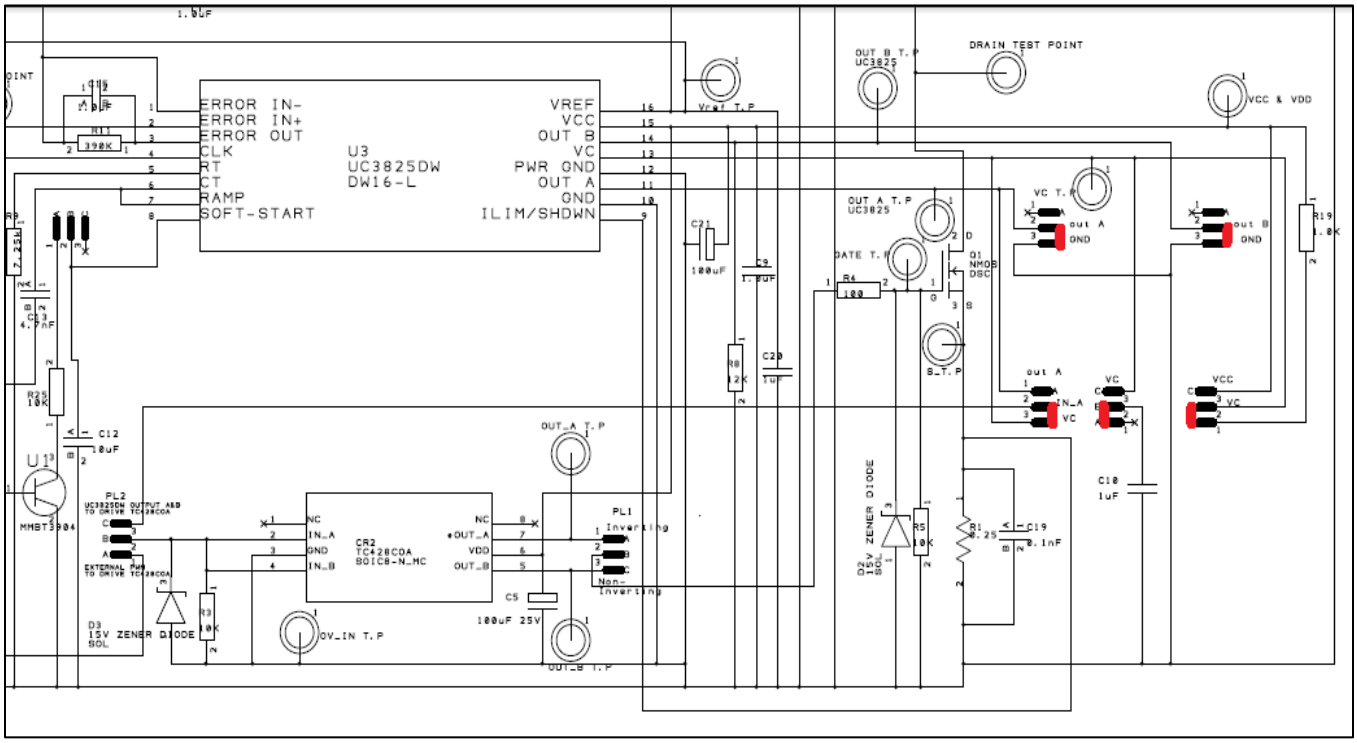


Figure 10: zoomed circuit showing jumper location for achieving more than 70% duty cycle with high current output

3. Conclusion

The operation and different configuration modes for the flyback converter have been discussed in this paper. The normal operating mode for the flyback converter is made easy by having all jumpers on the PCB moved in one direction.

Appendix C:

Flyback Converter Timing, Voltage Control and Current Control resistors

1. Introduction

The flyback converter's timing resistor, current limiting resistor and Voltage Control Mode (VCM) resistors are responsible for ensuring the correct operation of the flyback converter. This paper will discuss in detail on how the resistor values were determined and what improvement they contributed towards the performance of the flyback converter.

1. Timing Resistor (R_T)

The timing resistor connected on pin 5 of the PWM chip is responsible for setting the oscillator frequency and hence any changes made to the resistor will have great impacts on the performance of the PWM as the frequency affects the duty cycle of the pulse width modulation. Figure 1 shows a zoomed portion of the timing resistor of flyback converter from Appendix A.

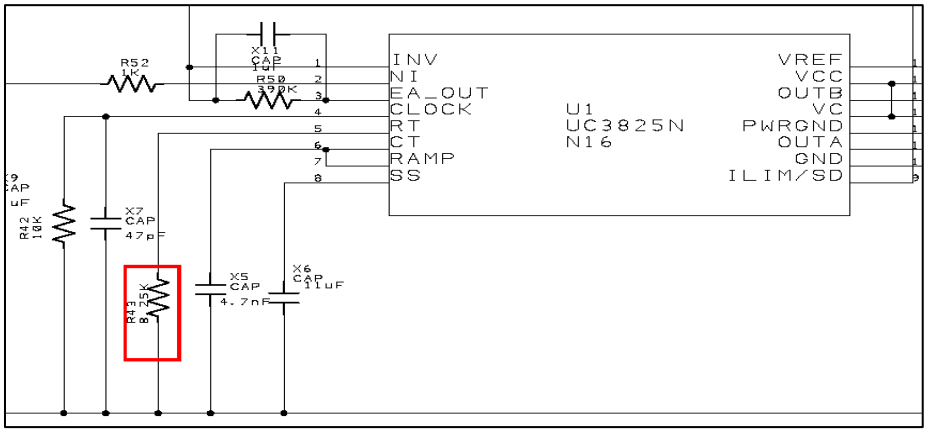


Figure 1: Timing resistor R_T of the UC3825 PWM chip

From the PWM chip's datasheet [45], the timing resistor is recommended to be in within the ranges of 1 kΩ to 100 kΩ. This ensures that the frequency and the duty cycle of the chip falls within the specified range.

The effective resistance of 8.25 kΩ was obtained using a variable resistor connected at pin 5 of the PW chip as shown in figure 2 above. The output resistance was then varied until a wide load variation was achieved. The new duty cycle of the PWM chip can then be calculated using the datasheet formula of equation (1) below;

$$R_T = \frac{3V}{10mA\ x\ (1-D_{MAX})}\ ,\quad \text{with } R_T = 8.25\ k\Omega \tag{1}$$

$$\therefore D_{MAX} = 96\%$$

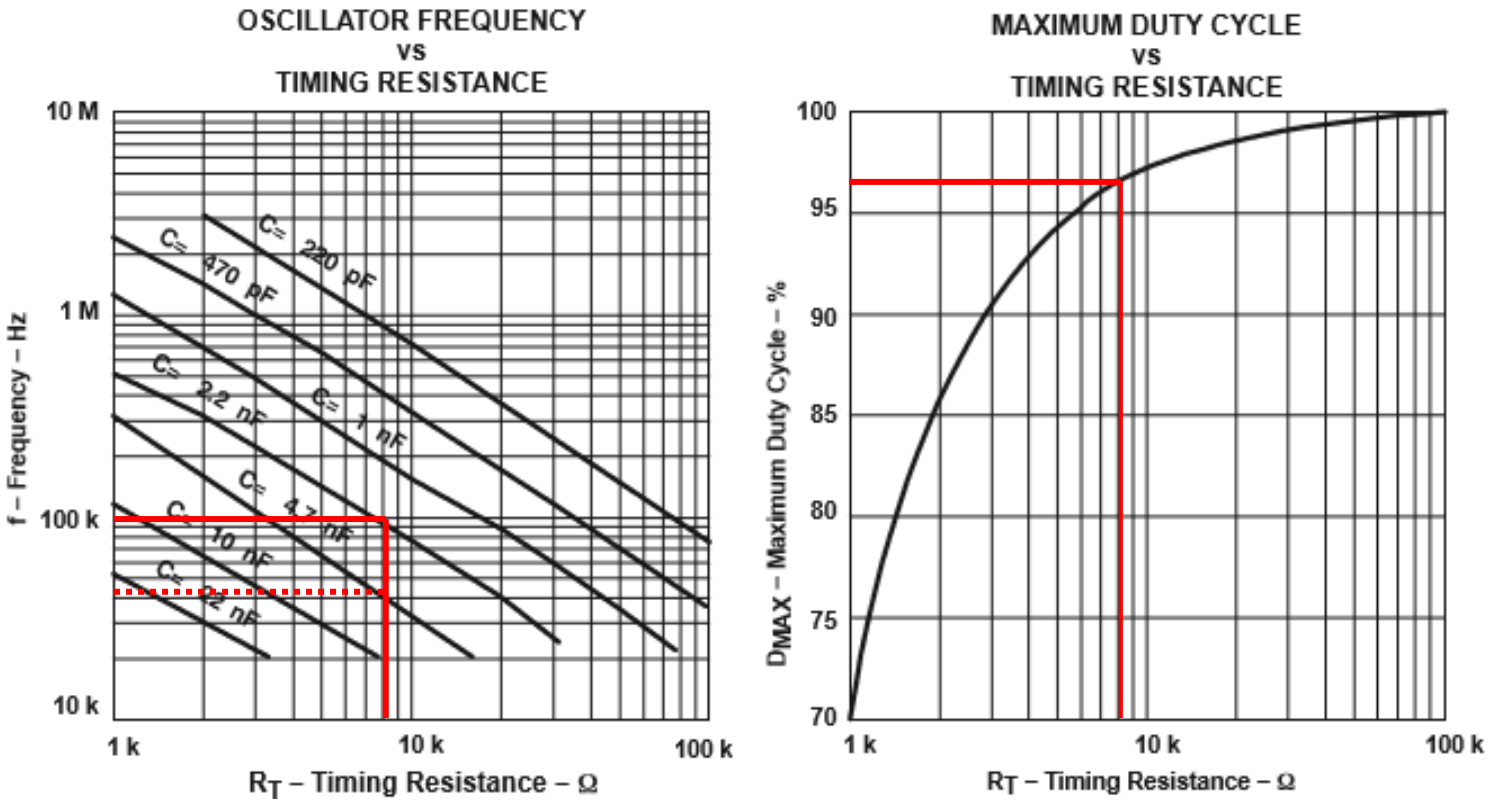
The initial flyback converter's timing resistance of 3.3 kΩ produced a duty cyle of 90% which would limit the output current to a smaller value as indicated on table 1 above.

The value of the timing capacitor (C_T) was obtained from the PWM's datasheet's equation 1 (2) and also from the oscillator frequency and timing resistor curves presented in appendix B [45].

$$C_T = \frac{1.6\ x\ D_{MAX}}{R_T\ x\ f} = 1.86\ nF,\ \ with\ f = 100kHz \tag{2}$$

$$= 4.7\ nF, \qquad with\ f = 42.7kHz$$

The oscillator frequency of the system was initially chosen at 100kHz and hence it is reasonable to change the timing capacitance to 1.86 nF so as to not only retain this frequency but also to maintain the leading edge blanking capacitance C_{LEB} [45]. However, a lower frequency reduced the overall switching loses which might cause overheating of the switching MOSFET.



(a) (b)

Figure 2: (a) Timing resistor vs oscillator frequency showing 2.2nF timing capacitor for 100kHz frequency and 4.7nF capacitor for 42.7kHz. (b) timing resistor vs the duty cycle showing 96% duty cycle obtained from the timing resistor [45]

2. Voltage Control Mode resistors (R_{16}, R_{17})

The VCM resistors shown in figure 3 form a voltage divider circuit for setting the error feedback voltage based on the picogrid's output voltage.

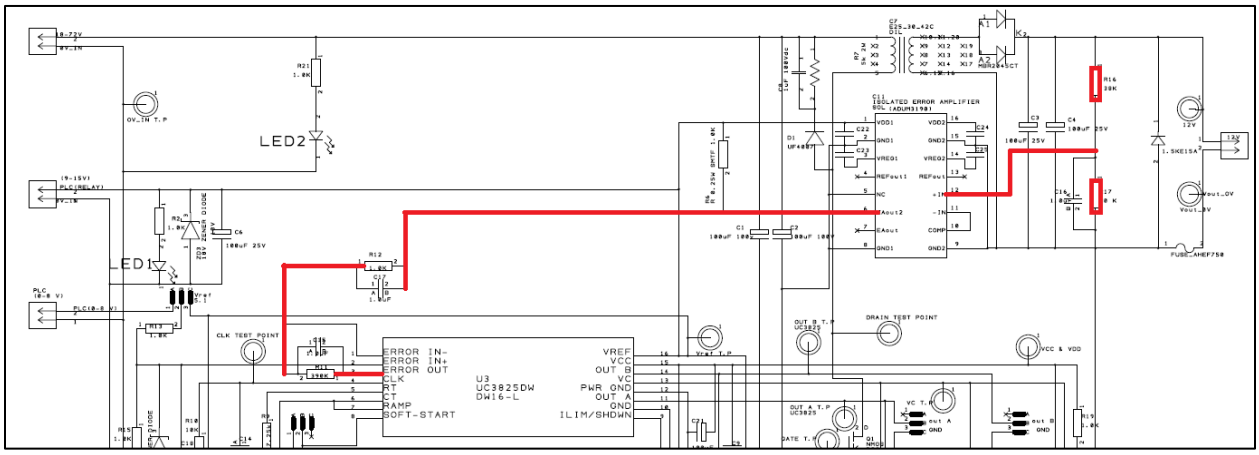


Figure 3: PWM controller VCM resistors and error voltage feedback loop

The values for the two resistors were determined in two ways;

i. Having a variable reference voltage to the PWM and a fixed load in order to determine a voltage set point as shown in figure 4 below.
ii. Having the two resistor values fixed and a variable reference voltage being fed back to the PWM's error amplifier as shown in figure 5. A variable load is then connected to the output and voltage and current measurements were recorded.

2.1. Test setup

From figure 4, the reference voltage (V_{ref}) was varied in intervals and resistor R_{17} varied with each reference voltage recorded. The feedback voltage (V_{fb}) and the output voltage to the load (V_{out}) were then monitored. Table 1 shows the resultant voltage values obtained from the test circuit.

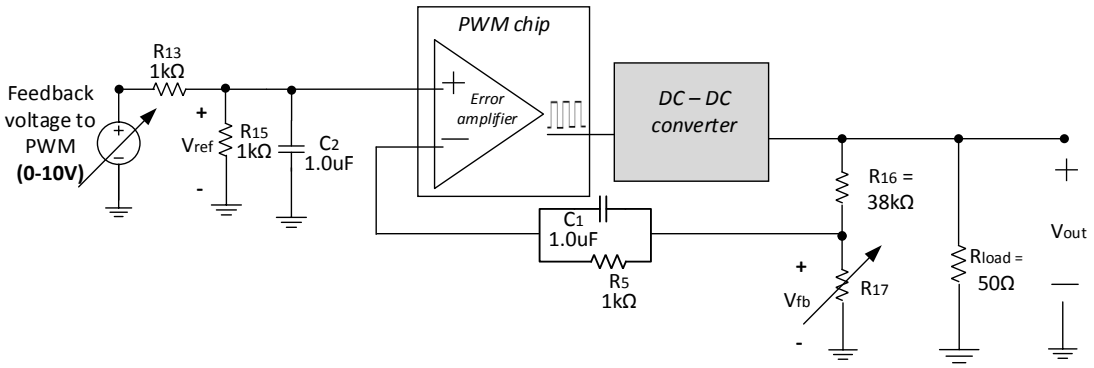


Figure 4: Test circuit for determining resistor R_{17} and the flyback's set-point

Table 1: Feedback voltage (V_{fb}) versus the output voltage to the load (V_{out}) with a varying reference voltage and resistor R_{17}

V_{ref}(V)	R_{17}(kΩ)	V_{fb}(v)	V_{out}(v)
1.52	3	1.14	19.11
2.02	6	1.9	13.89
2.51	9	2.11	11.01
2.63	10	2.53	12.1
3.03	12	2.90	11.95
3.52	16	2.86	9.73
4.53	18	2.91	9.04
5.01	21	2.88	8.13

The shaded region from table 1 shows the voltage values obtained with a 10kΩ resistor. This resistor was chosen as it not only produces the required stable voltage but also being easily available.

Figure 5 below shows the test circuit for determining the load variation using fixed resistor values obtained from figure 4 and a varying reference voltage. The results are tabulated in table 2 below.

Table 2: Load current versus voltage at different reference voltage values

V_{ref}(V)	V_{out}(v)	ΔI_{load}(A)
2	7.8	0.16 - 0.89
2.5	10	0.2 – 0.83
2.77	11	0.22 – 0.7
3	12	0.24 – 0.73
3.22	13	0.26 – 0.7
3.47	14	0.28 – 0.6

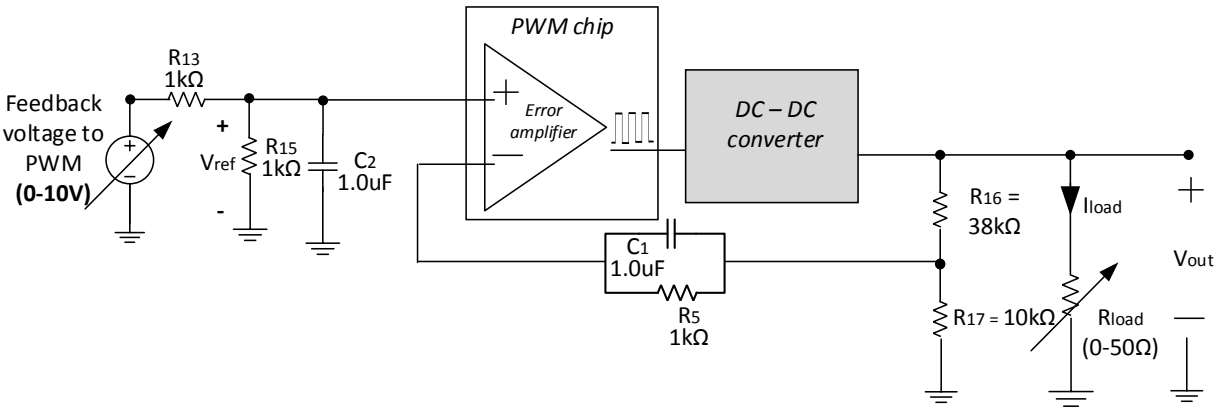


Figure 5: Flyback converter test circuit to determine VCM resistor values using a variable reference voltage and load

3. Current limiting resistor

The current limiting resistor shown by a red rectangle in figure 8 below is responsible for setting the current limit of the PWM chip through the shutdown pin 9 [45]. This is important as it not only limits the load current but also it automatically switches off the PWM chip when the maximum allowed current is reached.

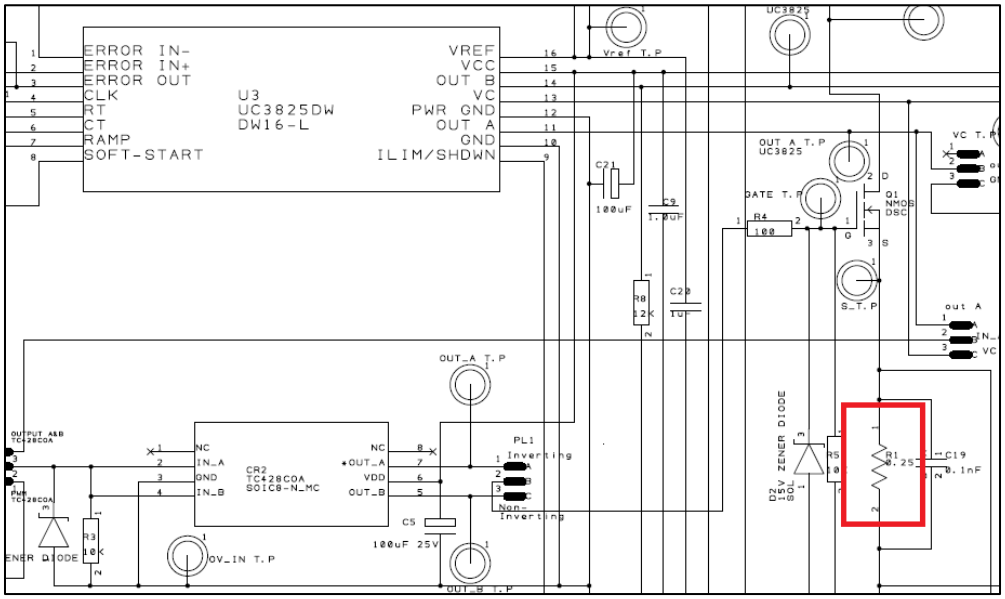


Figure 6: Current limiting resistor obtained from the flyback converter schematic in Appendix A

3.1. Test setup

The current limiting resistor was determined by varying the resistor value shown in the red rectangle of figure 6 whilst also varying the load current as shown in the test setup circuit of figure 7.

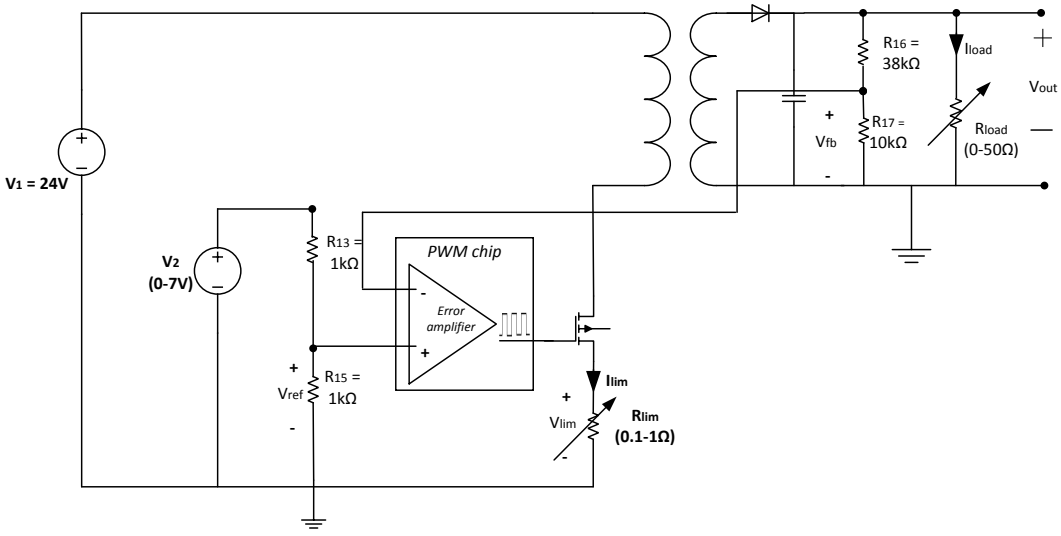


Figure 7: Test circuit to determine the relationship between the current limiting resistor and a varying load

From figure 7, the current limiting resistor is varied between 0.1Ω-1Ω and the range of the current through the load is increased at different fixed output voltages. Voltage and current measurements were recorded with every drop in voltage that was presented by the flyback converter. A value of 0.25Ω was found to give the best results of limiting the current of the flyback converter to around 0.89A as shown in table 3 below.

Table 3: Reference voltage versus output voltage and current with varying limiting resistor

V_{ref}(V)	V_{out}(v)	ΔI_{load}(A)	ΔI_{lim}(A)
2	7.8	0.16 - 0.89	0.9 – 1.1
2.5	10	0.2 – 0.83	0.9 -1.4
2.77	11	0.22 – 0.7	0.9 -1.2
3	12	0.24 – 0.73	0.9 -1.2
3.22	13	0.26 – 0.7	0.9 -1.2
3.47	14	0.28 – 0.6	0.9 -1.2

From the table, a wide load varying current and output voltage range of 11-14V can be achieved with the limiting resistor value. The voltage lies in within the grid code's PBV. A graph of the output voltage versus the load current taken from table 3 is shown in figure 8 below. Figure 9 shows the initial flyback converter's performance for comparison purposes.

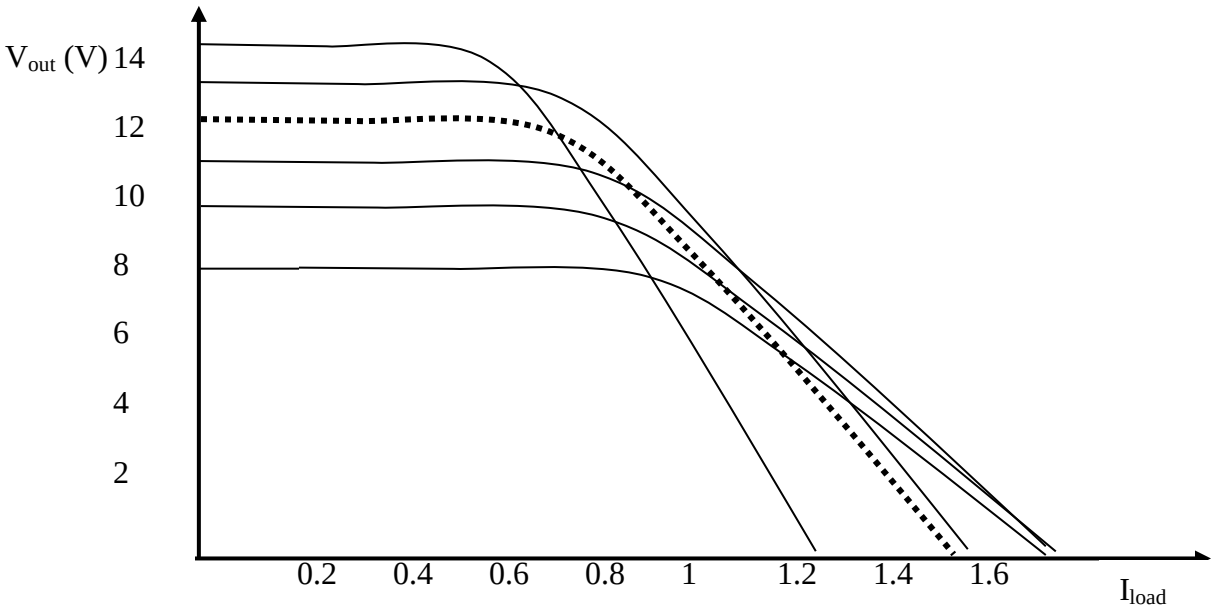


Figure 8: Output voltage versus output current of the improved flyback with improved current limiting

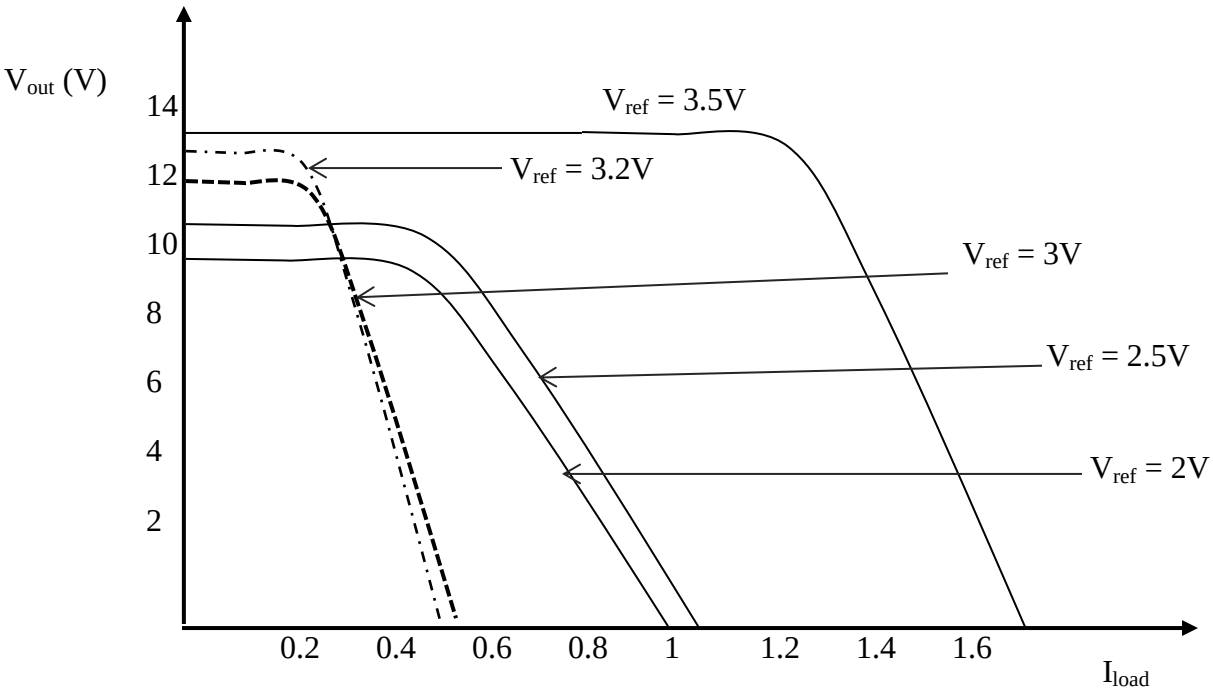


Figure 9: Output voltage versus the load current of the initial flyback converter

Comparting figure 8 to figure 9, it is clear that the load current was improved for the voltages required by the picogrid (10-14V) by determining the correct value of current limiting resistor giving efficient results.

4. Conclusion

The timing, voltage mode and current limiting resistors play a vital role in improving the performance and protection of the flyback converter that is responsible for regulating the load current and voltage. The initial flyback converter's performance was improved as a result of changing the values of these resistors.

Appendix D:

Flyback Converter Isolation Amplifier Design and Operation

1. Introduction

Opto-isolation forms part of the flyback converter protection as it ensures that the low voltage of the primary side of the flyback converter is completely isolated from the high voltage secondary side. This paper will give a detailed explanation on the design process as well as its operation through tests and results.

2. Opto-Isolation

Figure 1 shows a zoomed portion of the isolation amplifier from the complete schematic of the flyback converter in Appendix A. A high output voltage on the Picogrid bus would result in a low isolated feedback voltage being fed back to the PWM controller chip. The chosen isolation amplifier was the ADUM3190 High Stability Error Amplifier [56]

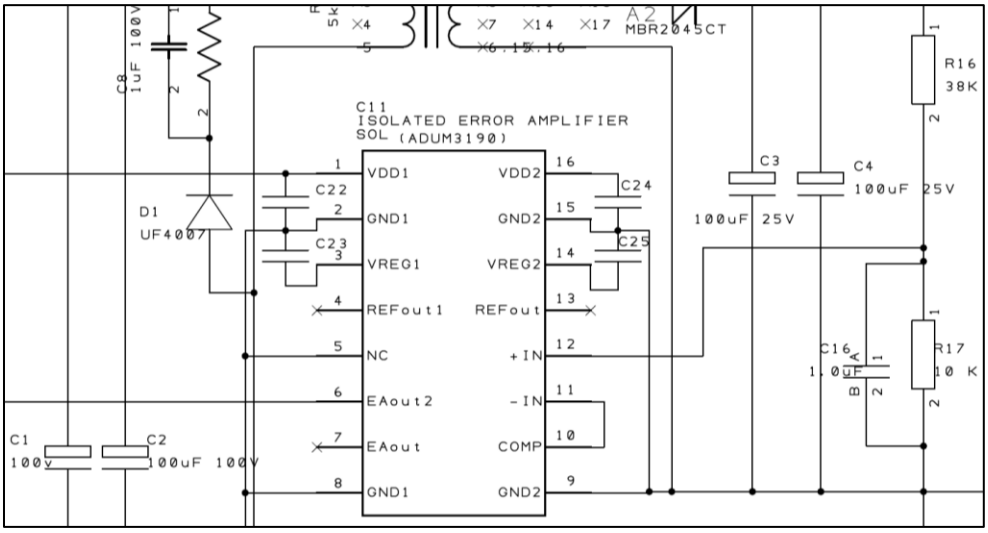


Figure 1: Isolation amplifier in the feedback loop of the flyback converter

2.1. Choice of components

The capacitor values for the configuration were obtained from the amplifier circuit diagram in the IC's datasheet [56] shown in figure below.

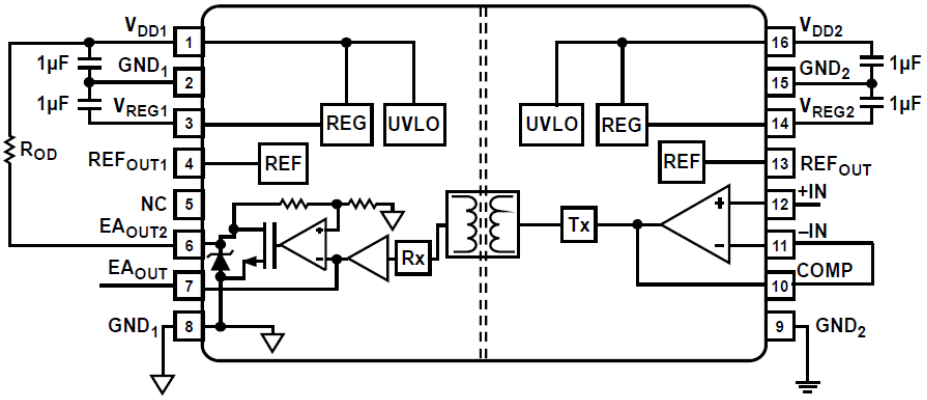


Figure 2: Flyback converter circuit with highlighted changes [56]

3. Operation and Test Setup

The aim of the test setup was to determine the relationship between Vin and Vout shown in figure 3 when the supply voltages V_2 and V_3 vary. The test setup diagram shown in figure 3 below was also used to verify the voltage mode resistors (R_{16} and R_{17}) values for the feedback voltage to the PWM controller (refer to Appendix B). The optimal voltage required for a given feedback error voltage was then obtained by varying each voltage separately.

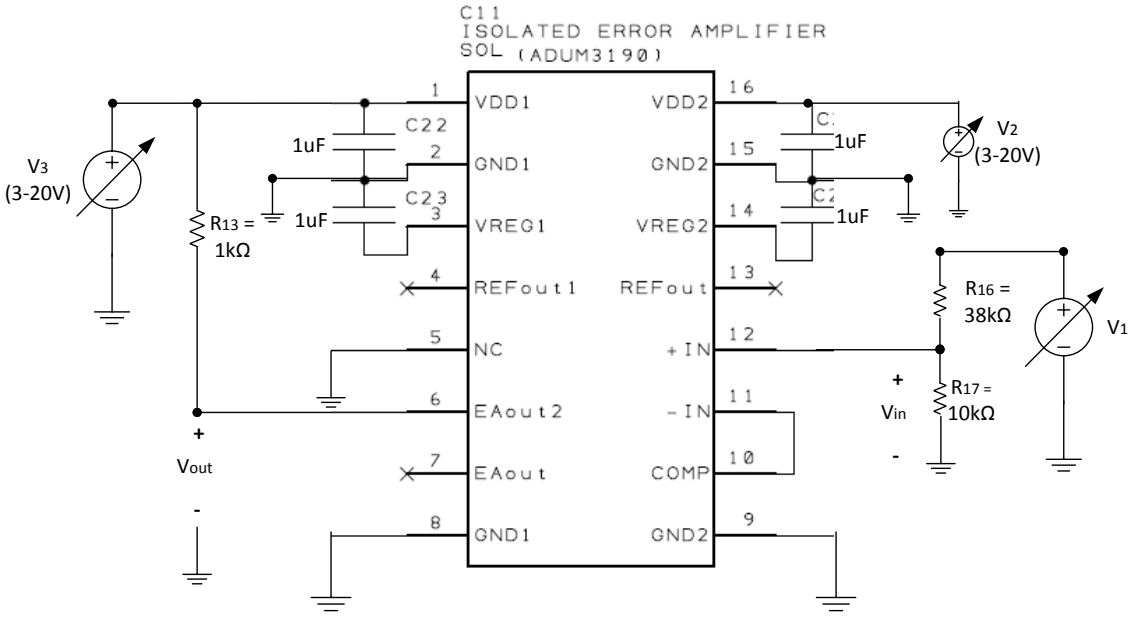


Figure 3: Isolated error amplifier test circuit

Where; V_1 = Picogrid Bus Voltage (PBV)

V_2 = Supply voltage to the isolation amplifier secondary side

V_3 = Supply voltage to the primary side

Vin = Feedback voltage from PBV

Vout = Isolated error feedback voltage to the PWM controller chip [1]

From figure 3, three tables were obtained as follows;

i. V_1 was varied whilst V_2 and V_3 remained constant as shown in table 1.
ii. V_1 and V_2 were varied whilst V_3 remained constant as shown in table 2.
iii. V_1 and V_3 were varied whilst V_2 remained constant as shown in table 3.

Table 1: V_1 varied with $V_2 = V_3 = 12V$

V_1 (V)	Vin (v)	Vout (v)
0	0	0
1	0.1	0.34
2	0.19	0.58
3	0.26	0.76
4	0.36	1.03
5	0.45	1.26
6	0.55	1.53
7	0.62	1.69
8	0.65	1.73
9	0.7	1.97
10	0.8	2.22
11	0.9	2.45
12	1	2.66
13	1.1	2.9
14	1.15	3.13
15	1.25	3.37
16	1.33	3.6

From table 1, a linear relationship between Vin and Vout can be observed when V_2 and V_3 are constant. Figure 4 shows the resultant graph obtained from this.

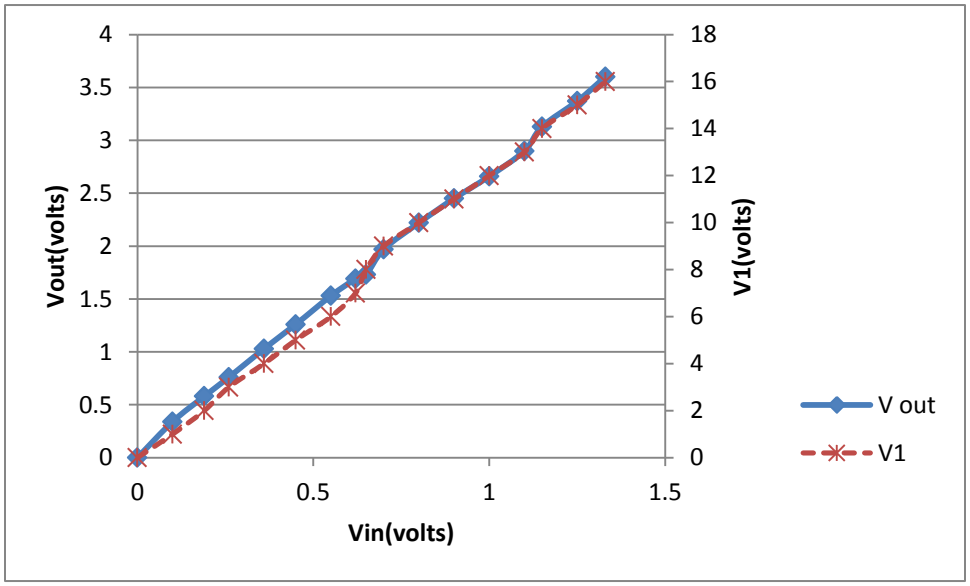


Figure 4: Relationship between Vin and Vout with constant V_2 and V_3

Table 2 below shows the relationship between Vin and Vout with V_1 and V_2 varied whilst V_3 remained constant.

Table 2: V_1 and V_2 varied with constant V_3 = 12V

V_1 (V)	V_2 (v)	Vin (v)	Vout (v)
0	0	0	5.97
1	1	0.1	5.97
2	2	0.19	5.97
3	3	0.26	5.97
4	4	0.36	1.03
5	5	0.45	1.26
6	6	0.55	1.53
7	7	0.62	1.69
8	8	0.65	1.73
9	9	0.7	1.97
10	10	0.8	2.22
11	11	0.9	2.45
12	12	1	2.66
13	13	1.1	2.9
14	14	1.15	3.13
15	15	1.25	3.37
16	16	1.33	3.6

Figure 5 below shows the graph obtained from table 2. From the graph it is clear that any voltage value of V_2 less than 3V results in a non-linear relationship between the input and output voltages Vin and Vout respectively. This therefore means that V_2 must always be greater than 3V in order to maintain the required linear relationship. A battery source of 5V or less can be used in order to avoid this.

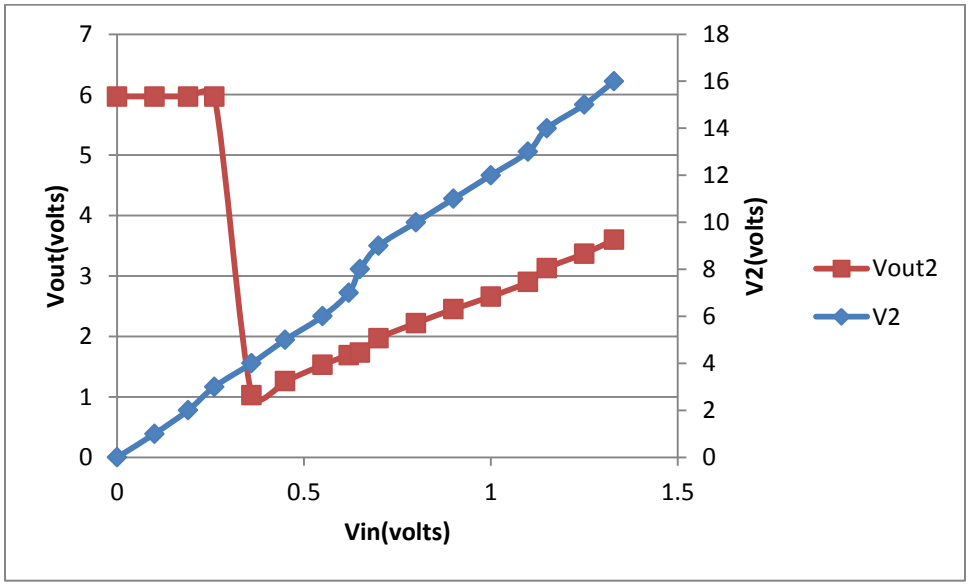


Figure 5: Relationship between Vin and Vout with a varying V_2 and constant V_3

From figure 3 above, V_1 and V_3 were then varied whilst V_3 was kept at a constant voltage and table 3 below was obtained.

Table 3: V_1 and V_3 varied with constant V_2 = 12V

V_1 (V)	V_3 (v)	Vin (v)	Vout (v)
0	0	0	0
1	1	0.1	0.12
2	2	0.19	0.31
3	3	0.26	0.76
4	4	0.36	1.03
5	5	0.45	1.26
6	6	0.55	1.53
7	7	0.62	1.69
8	8	0.65	1.73
9	9	0.7	1.97
10	10	0.8	2.22
11	11	0.9	2.45
12	12	1	2.66
13	13	1.1	2.9
14	14	1.15	3.13
15	15	1.25	3.37
16	16	1.33	3.6

Figure 6 shows the resultant graph showing the linear relationship between Vin and Vout of the test circuit of figure 3 given a varying V_3.

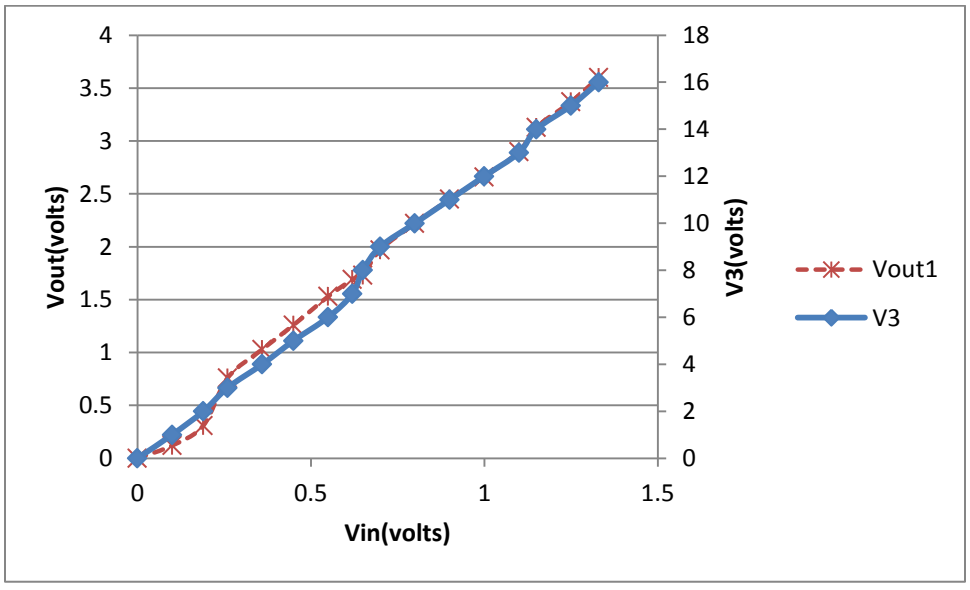


Figure 6: Relationship between Vin and Vout with a varying V_3 and constant V_2

4. Conclusion

A linear relationship between the feedback error voltage from the Picogrid Bus Voltage and the error voltage feedback to the PWM controller chip is proved to be attainable using an isolation amplifier. The amplifier non-linearity can be avoided by ensuring that the secondary voltage supply to the amplifier is kept above 3V at all times during operation.

Appendix E:

Flyback Converter Components List

Table 1: Flyback converter components

Component	Name /Part number	Quantity
PWM controller	UC3825DW	1
Isolation Amplifier	ADUM3190	1
Transistor	60V MOSFET – BUK456-60A	1
Transformer	2:1 winding ratio	1
Regulator	MBR2045CT	1
Resistors	• 1Ω • 100Ω • 270Ω • 1kΩ • 3.3kΩ • 10kΩ • 12kΩ • 39kΩ • 390kΩ	• 10 • 1 • 1 • 5 • 1 • 3 • 1 • 1 • 1
Capacitors	• 4.7nF, ceramic • 47pF, ceramic • 100pF, ceramic • 1µF, ceramic • 22µF, electrolytic, 35V • 100µF, electrolytic, 35V • 100µF, electrolytic, 100V • 2.2µF, polypropylene, WIMA	• 1 • 1 • 1 • 11 • 2 • 3 • 2 • 1
Diodes	• General Purpose Rectifier – 1N4007 • 12V Zener – 1N4742A	• 1 • 1
Fuse	• AHEF750	• 1

Appendix F:

Picogrid Node User Manual

1. Introduction

This paper provides the setup and user manual for the operation of each node on the picogrid. A brief on the network configuration for each node's PLC is provided together with steps on how to program the PLC from a host station. The setup instructions for setting up a display touch panel (HMI) used on the storage node is also discussed. It is worth noting that the information provided below is a brief guide on how to quickly setup the PLCs for programming without having to refer to the long PLC user manual [46].

2. PLC Static IP setup

In order for each node to connect to the same Picogrid network, each node's PLC has to be assigned a static IP address so as to avoid IP address conflicts. This has to be done especially if one decides to use an unmanaged network switch [49].

Each device was assigned the following **static** IP address;

- Load node – 192.168.0.1
- Solar node – 192.168.0.2
- Storage node – 192.168.0.3
- Wind node – 192.168.0.4
- HMI display touch – 192.168.0.5

These IP addresses should be used in order to access the software program of each PLC on the Picogrid.

3. Source and Load Nodes

Sources and Load nodes do not require any setup or user manual for operation. One has to simply plug the adapter in the mains. The PLCs are already setup to run automatically immediately after power is switched on.

4. Storage node

This node also automatically initialises itself once connected to the mains supply. It also comes with a Siemens HMI screen for displaying certain properties of the picogrid. Figure F1 shows the functional keys of the HMI. The display panel shows a graph of the Picogrid Bus Voltage and the state of battery in terms of its charging and discharging properties. More programs can be added to the display panel via the SIEMENS STEP 7 programming software of the TIA portal [46].

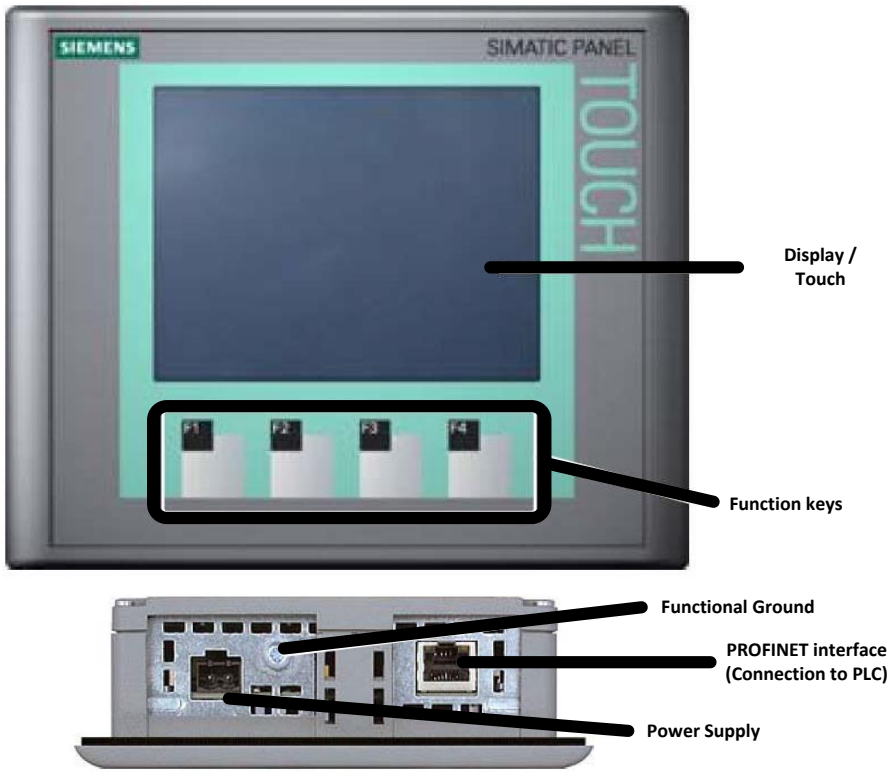


Figure F1: Simatic HMI KTP-400 structure [2]

4.1. Setup

Setting up the HMI is a simple step by step process that can be followed in the HMI manual [57]. The most important thing to note is having the static IP address setup correctly and that the host computer can detect the HMI. TIA portal allows function data to be sent to the HMI for display purposes. The screen can be customised according to once theme in the TIA portal [46].

4.2. Operation

As seen in figure F2, by touching the home button, one is able to select between the grid voltage and battery properties. The battery screen is displayed by touching the battery icon.

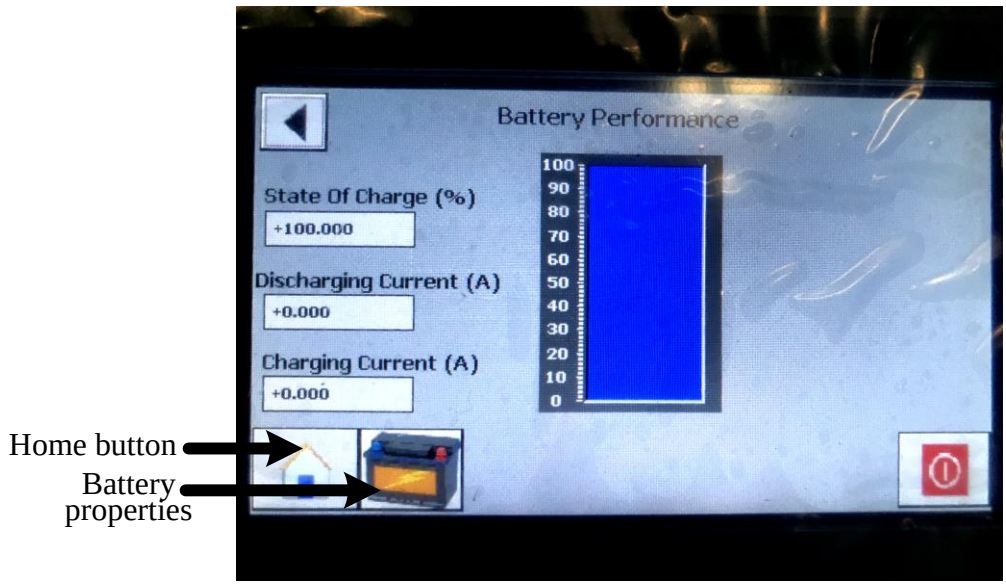


Figure 2: Selection category

4.2.1. State of Battery display

This screen shown in figure F2 displays the battery state of charge and the charging and discharging current when the node is sinking and sourcing current on the grid respectively.

4.2.2. Grid Voltage Display

This screen is selected by touching the grid voltage menu on the home button. The photo shown in figure F3 shows the grid voltage graph as a red line.

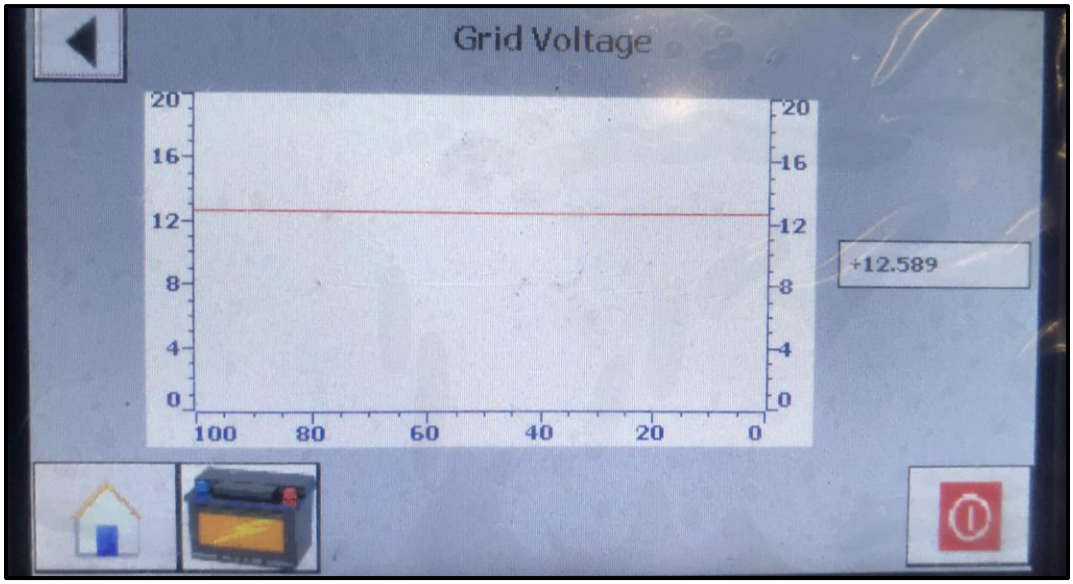


Figure F3: voltage selection

Appendix G:

Load Node Circuit Diagram

The load node demonstrator is made up of the following components shown in figure G1 below;

- 1 Siemens Simatic S7-1200 PLC.
- 4x conditional loads (R1,R2,R3,R4) as shown in figure 5 below.
- 4x non- conditional loads (R5,R6,R7,R8).
- 8x manual switches.
- 8x LEDs connected to each of the eight loads.
- R1 = 800Ω, R2 = 800Ω, R3 = 800Ω, R4 = 800Ω, R5 = 800Ω, R6 = 800Ω, R7 = 800Ω, R8 = 800Ω,

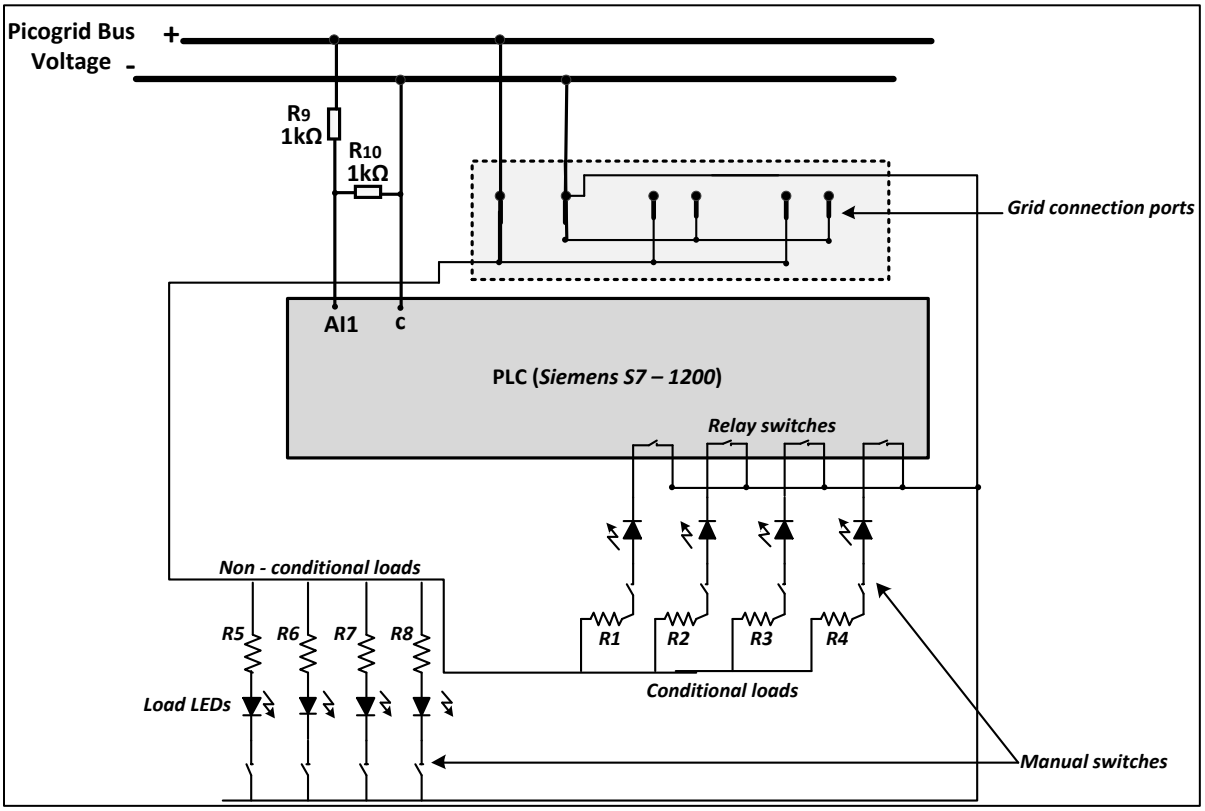


Figure G1: Load node structure

Figure G2 shows four analogue inputs that are dedicated for connecting the conditional loads and four outputs for the relay switches. Two additional inputs are used for the grid connections because the PLC analogue inputs have a maximum input voltage of 10V [46]. A voltage divider

is thus used to split the grid voltage in to two voltages of the ratio 1:2 before connecting to the PLC.

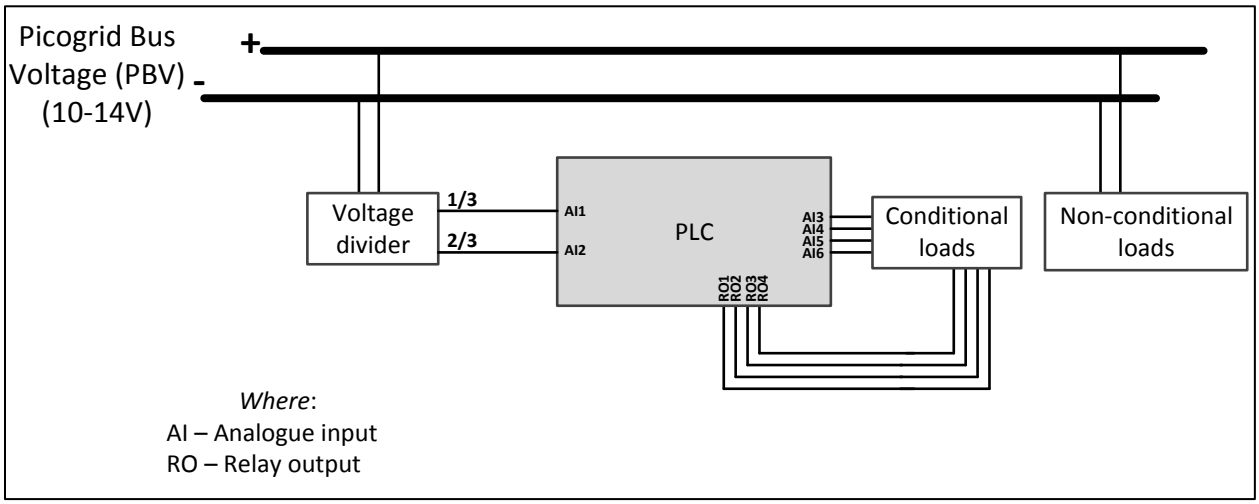
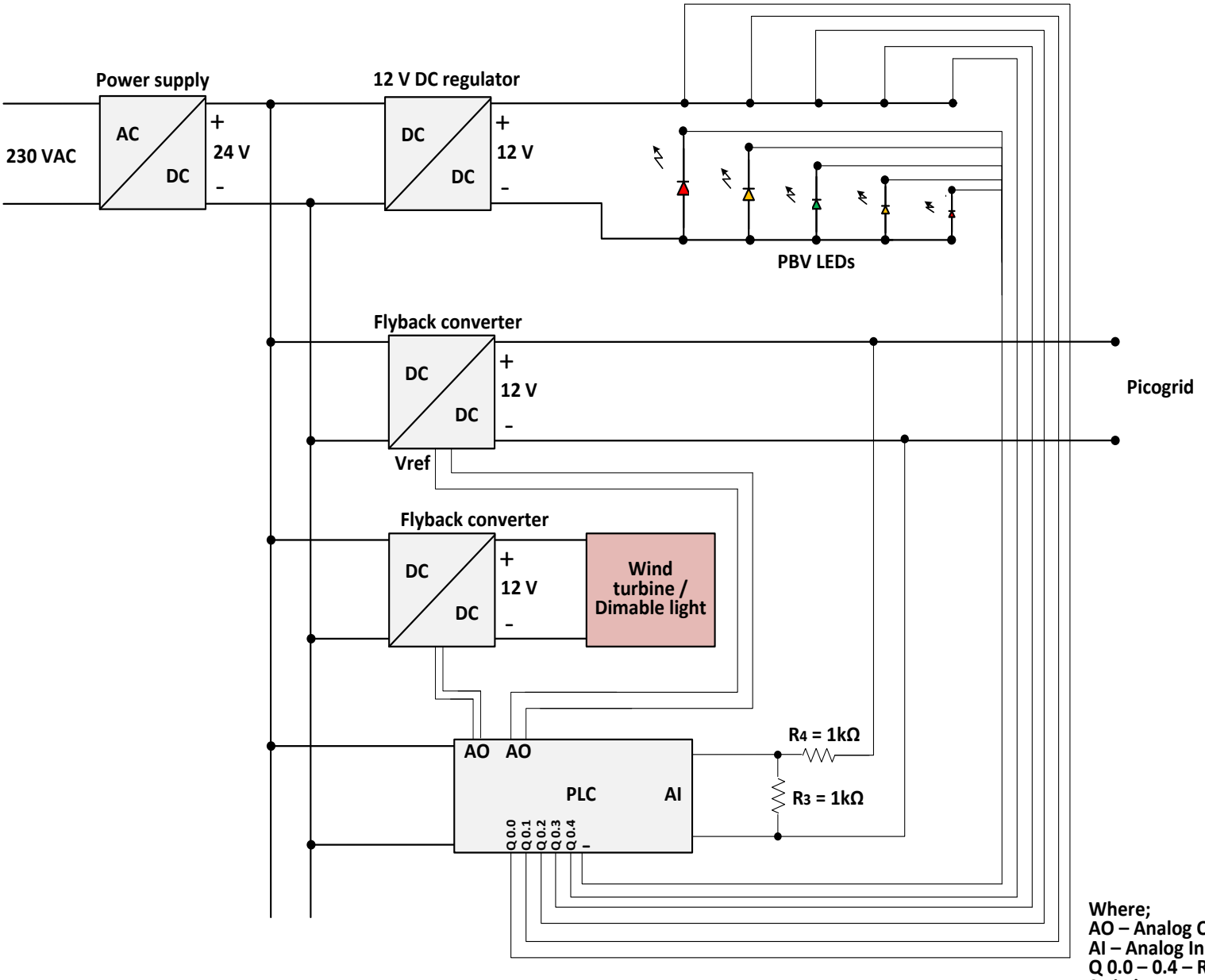


Figure G2: Detailed Load node to PLC connections

Appendix H:

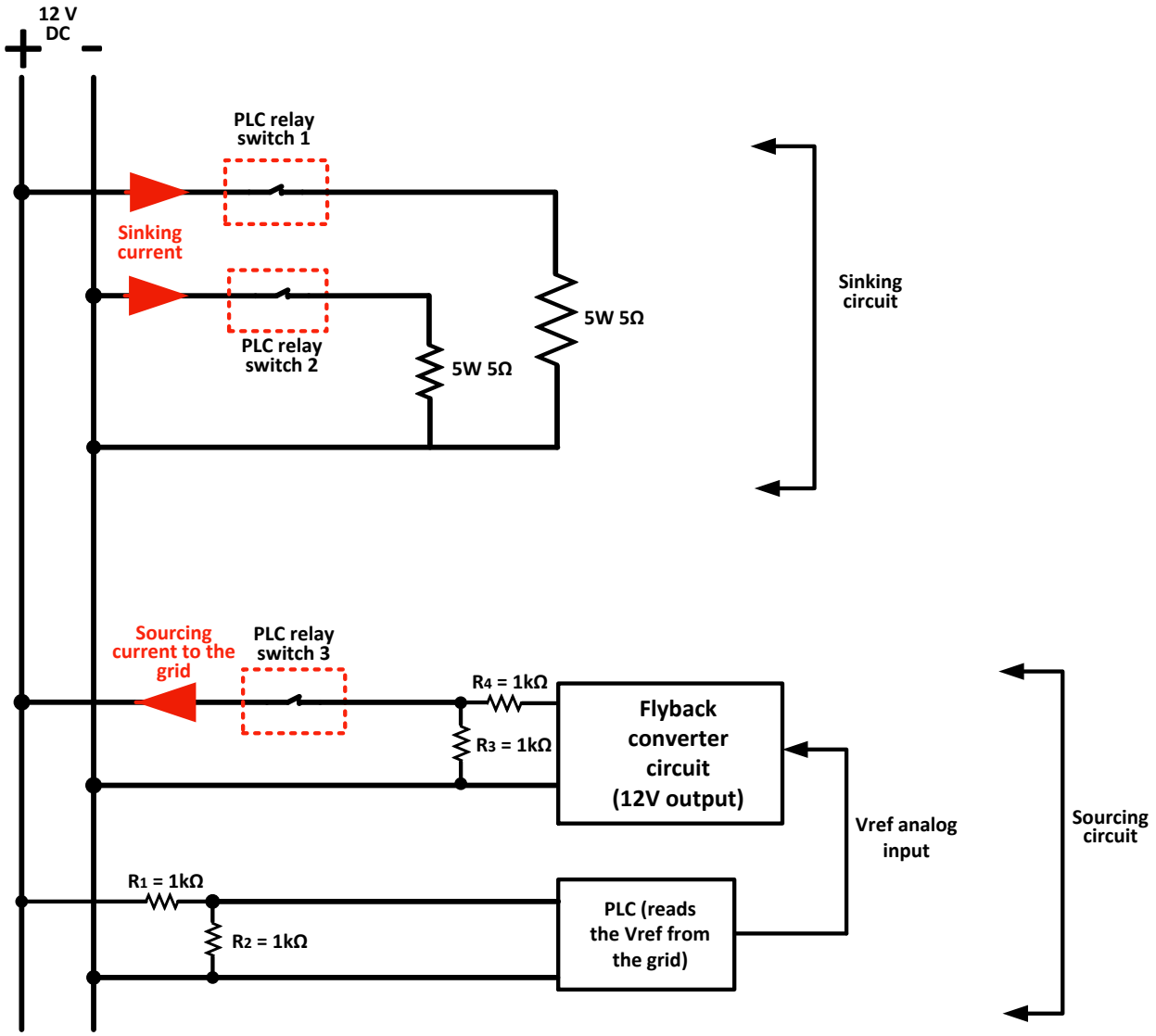
Source Node Circuit Diagram



Where;
AO – Analog Output
AI – Analog Input
Q 0.0 – 0.4 – Relay Switches

Appendix I:

Storage Node Circuit Diagram



Appendix J:

Solar Node Source code

Solar Powercontrol FC

```
// Solar power control voltage -> straight line mapping

#m := ("Variables".pwmMinDC - "Variables".pwmMaxDC) / ("Variables".pwmMaxDC -
"Variables".pwmMinDC);

#c := "Variables".pwmMaxDC - #m * "Variables".pwmMinDC;

 #solarControlVoltage:= "Variables".gridvoltagelimit - (#m
*"Variables".solarPowerControlVoltage + #c);

"solarPowerControlAnalogOutput" := SCALE_X(MIN := 0, VALUE := NORM_X(MIN := 0.0, VALUE
:= #solarControlVoltage, MAX := 10.0), MAX := 27648);
```

Gridcontrol FC

```
// Function responsible for defining the grid code.

//switch off all LED indicators

  "LED_UV":= FALSE; //Undervoltage PBV indicator

  "LED_LV":= FALSE; //low voltage PBV indicator

  "LED_stable":= FALSE; //stable PBV indicator

  "LED_HV":= FALSE; //High voltage indicator

  "LED_OV":= FALSE;  //over voltage indicator

//connect the node to the picogrid

"UVandOVrelay":= TRUE;

//alarm for tripping the node incase of heavy load or short circuit

// Case 1 :  The grid voltage is in under voltage mode ( < 9.7 V) or in over voltage mode ( >
14.2V)

//disconnect the source if below voltage (load shedding)

IF"numericalGridVoltage(ch1)" < #gridVoltageLVlowerThreshold OR
"numericalGridVoltage(ch1)" > #gridVoltageHVupperThreshold THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

  "Variables".UV_alarm :="Variables".UV_alarm + #UVcounter;
```

```
 IF "numericalGridVoltage(ch1)" < #gridVoltageLVlowerThreshold THEN

   "LED_UV":= TRUE; //indicate under voltage

 ELSE

   "LED_OV":= TRUE; //overvoltage indicator

 END_IF;

  END_IF;

//trip and restart the node in UV or OV or Short Circuit conditions

WHILE "Variables".UV_alarm > #UVtime &  "Variables".UV_alarm < #UVtime_1 DO

    "UVandOVrelay":= FALSE;

 EXIT;

END_WHILE;

// Case 2 :  The grid voltage is in low voltage mode (10 - 10.7V)

IF "numericalGridVoltage(ch1)" >= #gridVoltageLVlowerThreshold &
"numericalGridVoltage(ch1)" < #gridVoltageStableLowerThreshold & "Delay".Q THEN

 "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

 "Variables".UV_alarm := "Variables".UV_alarm - "Variables".UV_alarm;//reset

 "LED_LV":= TRUE;

END_IF;

// Case 3 : The grid voltage is between the range 11 - 13V (stable operation)

IF  "numericalGridVoltage(ch1)" >= #gridVoltageStableLowerThreshold &
"numericalGridVoltage(ch1)" <= #gridVoltageStableUpperThreshold & "Delay".Q THEN

 "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

 "LED_stable":= TRUE;

 "Variables".UV_alarm := "Variables".UV_alarm - "Variables".UV_alarm;//reset

   IF "numericalGridVoltage(ch1)" > #gridVoltageStableLowerThreshold &
"numericalGridVoltage(ch1)" < #gridVoltage_12V &"Delay".Q THEN

     "Variables".PwmVoltageControl := "Variables".PwmVoltageControl +
#PwmVoltageIncrement;

   ELSIF "numericalGridVoltage(ch1)" >= #gridVoltage_12V & "numericalGridVoltage(ch1)" <
#gridVoltageStableUpperThreshold &"Delay".Q THEN

     "Variables".PwmVoltageControl := "Variables".PwmVoltageControl +
```

```
#PwmVoltageIncrement;

    END_IF;

    END_IF;

// Case 4 :  The grid voltage is in high voltage mode (13.2 - 14V)

IF "numericalGridVoltage(ch1)" > #gridVoltageStableUpperThreshold &
"numericalGridVoltage(ch1)" < #gridVoltageHVupperThreshold & "Delay".Q THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

   "LED_HV":= TRUE;

 END_IF;

//    ------------------------------

  WHILE "Variables".PwmVoltageControl > "Variables".solarPowerControlVoltage  DO

   "Variables".PwmVoltageControl := "Variables".solarPowerControlVoltage -
#PwmVoltageIncrement;

  END_WHILE;

  WHILE "Variables".PwmVoltageControl < "Variables".pwmMinDC  DO

   "Variables".PwmVoltageControl := "Variables".solarPowerControlVoltage +
#PwmVoltageIncrement;

  END_WHILE;

  // Delay ------------------------------

IF "Delay".Q = false THEN

   #startDelay := true;

  ELSE

   #startDelay := false;

  END_IF;

"Delay".TON(IN := #startDelay,

          PT := T#1ms,

          Q => #delayQ,

         ET => #delayET);

// Ensure PWM Control voltage is within limits, then write to output

IF "Variables".PwmVoltageControl <= "Variables".pwmMaxDC AND
"Variables".PwmVoltageControl >= "Variables".pwmMinDC THEN
```

```
  "solarPowerControlAnalogOutput(ch0)" := SCALE_X(MIN:=0, VALUE:=NORM_X(MIN:=0,
VALUE:="Variables".PwmVoltageControl, MAX:=12), MAX:=33178);

END_IF;
```

SolarPowerControl FC

```
// Ensure PWM Control voltage is within limits, then write to output

IF "Variables".PwmVoltageControl <= "Variables".pwmMaxDC AND
"Variables".PwmVoltageControl >= "Variables".pwmMinDC THEN

  "solarPowerControlAnalogOutput(ch0)" := SCALE_X(MIN:=0, VALUE:=NORM_X(MIN:=0,
VALUE:="Variables".PwmVoltageControl, MAX:=10), MAX:=27648);

END_IF;

// 27648 is equivalent to 10 V

"AnalogOut_10V(ch3)" := 27648;
```

Convert Inputs FC

```
// Convert numerical value to scaled real value

"Variables".solarPowerControlVoltage := SCALE_X(MIN:= "Variables".pwmMinDC,
VALUE:=NORM_X(MIN:=0, VALUE:="solarPowerControlAnalogInput(ch0)", MAX:=27648),
MAX:="Variables".pwmMaxDC);

// Convert real value to numerical equivalent

"Variables".numericalSolarPowerControlVoltage := SCALE_X(MIN:= 0,
VALUE:=NORM_X(MIN:=0.0, VALUE:="Variables".solarPowerControlVoltage , MAX:=10.0),
MAX:=27648);
```

Appendix K:

Wind Node Source Code

Convertinputs FB

```
// 27648 is equivalent to 10 V //check manual

"AnalogOut_10V(ch3)" := 27648;

// Convert numerical value to scaled real value

"Variables".WindTurbineControlVoltage := SCALE_X(MIN:= "Variables".pwmMinDC,
VALUE:=NORM_X(MIN:=0, VALUE:="windTurbineControlAnalogInput(ch0)", MAX:=27648),
MAX:="Variables".pwmMaxDC);

// Convert real value to numerical equivalent

"Variables".NumericalWIndTurbineControlVoltage := SCALE_X(MIN:= 0,
VALUE:=NORM_X(MIN:=0.0, VALUE:="Variables".WindTurbineControlVoltage , MAX:=10.0),
MAX:=27648);
```

Windturbinemotorcontrol FB

```
// Wind Turbine control voltage -> straight line mapping

#m := ("Variables".pwmMinDC - "Variables".pwmMaxDC) / ("Variables".pwmMaxDC -
"Variables".pwmMinDC);

#c := "Variables".pwmMaxDC - #m * "Variables".pwmMinDC;

#MotorControlVoltage := "Variables".gridvoltagelimit - (#m
*"Variables".WindTurbineControlVoltage + #c);

"windTurbineControlAnalogOutput(ch0)" := SCALE_X(MIN := 0, VALUE := NORM_X(MIN := 0.0,
VALUE := #MotorControlVoltage, MAX := 10.0), MAX := 27648);
```

Gridcontrol FB

```
//switch off all LEDs

 "LED_UV":= FALSE;

 "LED_LV":= FALSE;

 "LED_Stable":= FALSE;

 "LED_HV":= FALSE;
```

```
  "LED_OV":= FALSE;

// Case 1 :  The grid voltage is in under voltage mode ( < 9.7 V)

IF "numericalGridVoltage(ch1)" < #gridVoltageLVlowerThreshold &"Delay".Q THEN

   "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

   "LED_UV":= TRUE;

   END_IF;

// Case 2 :  The grid voltage is in low voltage mode (10 - 10.7V)

IF "numericalGridVoltage(ch1)" > #gridVoltageLVlowerThreshold &
"numericalGridVoltage(ch1)" < #gridVoltageStableLowerThreshold & "Delay".Q THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

  "LED_LV":= TRUE;

END_IF;

// Case 3 : The grid voltage is between the range 11 - 13V (stable operation)

IF  "numericalGridVoltage(ch1)" > #gridVoltageStableLowerThreshold &
"numericalGridVoltage(ch1)" < #gridVoltageStableUpperThreshold & "Delay".Q THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

  "LED_Stable":= TRUE;

    IF "numericalGridVoltage(ch1)" > #gridVoltageStableLowerThreshold &
"numericalGridVoltage(ch1)" < #gridVoltage_12V &"Delay".Q THEN

    "Variables".PwmVoltageControl := "Variables".PwmVoltageControl +
#PwmVoltageIncrement;

    ELSIF "numericalGridVoltage(ch1)" >= #gridVoltage_12V & "numericalGridVoltage(ch1)" <
#gridVoltageStableUpperThreshold &"Delay".Q THEN

      "Variables".PwmVoltageControl := "Variables".PwmVoltageControl +
#PwmVoltageIncrement;

    END_IF;

    END_IF;

// Case 4 :  The grid voltage is in high voltage mode (13.2 - 14V)

IF "numericalGridVoltage(ch1)" > #gridVoltageStableUpperThreshold &
"numericalGridVoltage(ch1)" <= #gridVoltageHVupperThreshold & "Delay".Q THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

 "LED_HV":= TRUE;
```

```
END_IF;

// Case 5 :  The grid voltage is in over voltage mode ( > 14.2V)

IF "numericalGridVoltage(ch1)" > #gridVoltageHVupperThreshold &"Delay".Q THEN

  "Variables".PwmVoltageControl := "Variables".PwmVoltageControl + #PwmVoltageIncrement;

  "LED_OV":= TRUE;

  END_IF;

//     ------------------------------

  WHILE "Variables".PwmVoltageControl > "Variables".WindTurbineControlVoltage  DO

    "Variables".PwmVoltageControl := "Variables".WindTurbineControlVoltage -
#PwmVoltageIncrement;

  END_WHILE;

  WHILE "Variables".PwmVoltageControl < "Variables".pwmMinDC  DO

    "Variables".PwmVoltageControl := "Variables".WindTurbineControlVoltage +
#PwmVoltageIncrement;

  END_WHILE;

  // Delay ------------------------------

IF "Delay".Q = false THEN

    #startDelay := true;

  ELSE

    #startDelay := false;

  END_IF;

"Delay".TON(IN := #startDelay,

            PT := T#1ms,

            Q => #delayQ,

           ET => #delayET);

// Ensure PWM Control voltage is within limits, then write to output

IF "Variables".PwmVoltageControl <= "Variables".pwmMaxDC AND
"Variables".PwmVoltageControl >= "Variables".pwmMinDC THEN

  "windTurbineControlAnalogOutput(ch0)" := SCALE_X(MIN:=0, VALUE:=NORM_X(MIN:=0,
VALUE:="Variables".PwmVoltageControl, MAX:=10), MAX:=27648);

END_IF;
```

Appendix L:

Load Node PLC code

Pollgridvoltage1 (FB)

```
// The grid voltage is obtained from a voltage divider circuit.

// Thus, the digital count grid voltage obtained is half the true value of the grid voltage

// Convert the digital count grid voltage to a an analogue grid voltage

#grid_voltage:= UINT_TO_REAL("GridVoltage");

#grid_voltage_2 := UINT_TO_REAL("GridVoltage_2");

//Determining the mean for acuracy

#normalised_grid_voltage:= #grid_voltage/"GeneralParameters".full_scale_range;

#normalised_grid_voltage_2:= #grid_voltage_2 / "GeneralParameters".full_scale_range;//full scale range incorporates the error obtained from the resistor value tolerance

//convert grid voltage to analog value

#analogue_grid_voltage := ((#normalised_grid_voltage + #normalised_grid_voltage_2) / "GeneralParameters".grid_voltage_scale_factor) ;

// Return true if in undervoltage band (2nd load shedding phase)

IF #analogue_grid_voltage < "GeneralParameters".grid_voltage_under_voltage THEN

  #within_band:= TRUE;

// Return false if the grid voltage is in the acceptable band

ELSE

  #within_band:= FALSE;

END_IF;
```

Pollgridvoltage2 (FB)

```
// The grid voltage is obtained from a voltage divider circuit.
```

```
// Thus, the digital count grid voltage obtained is half the true value of the grid voltage

// Convert the digital count grid voltage to a an analogue grid voltage

#grid_voltage_2 := UINT_TO_REAL("GridVoltage_2");

#grid_voltage:= UINT_TO_REAL("GridVoltage");

#normalised_grid_voltage:= #grid_voltage/"GeneralParameters".full_scale_range;

#normalised_grid_voltage_2:= #grid_voltage_2 / "GeneralParameters".full_scale_range;

#analogue_grid_voltage := ((#normalised_grid_voltage + #normalised_grid_voltage_2) /
"GeneralParameters".grid_voltage_scale_factor) ;

// Return true if in low voltage band (1st load shedding phase)

IF #analogue_grid_voltage >= "GeneralParameters".grid_voltage_under_voltage &
#analogue_grid_voltage <= "GeneralParameters".grid_voltage_low_voltage THEN

  #within_band_2:= TRUE;

// return false if not within the low voltage band

ELSE

  #within_band_2:= false;

END_IF;
```

turnplcswitcheson (FB)

```
//Turn on the PLC Switches of the load if the corresponding manual switches are turned off

IF "ManualSwitch_Load1" = FALSE THEN //if manual switch is off

  "PLCSwitch_Load1":= TRUE;         //PLC switch is ON

END_IF;

IF "ManualSwitch_Load2" = FALSE THEN

  "PLCSwitch_Load2":= TRUE;

END_IF;

IF "ManualSwitch_Load3" = FALSE THEN
```

```
  "PLCSwitch_Load3":= TRUE;
END_IF;
IF "ManualSwitch_Load4" = FALSE THEN
  "PLCSwitch_Load4":= TRUE;
END_IF;
```

delaygridvoltagepolling (FB)

```
// Activate a timer of the delay that activates the output of the delay after 200ms of time have elapse after the delay is activated
// Activate the timer of the delay if the output of the delay is false
IF #delay_output = false THEN
    #delay_initiator := true;
// Reseting the output of the delay and the initiating variable of the delay to prepare for another activation of the delay
  ELSE
    #delay_initiator := false;
  END_IF;
// Declration of the delay
"DelayGridVoltageMeasurement".TON(IN:=#delay_initiator,
                                  PT:=#delay_period,
                                  ET=>#delay_elapsed_time,
                                  Q=> #delay_output);
// Wait till the timer has finished running before exiting this block of code
WHILE "DelayGridVoltageMeasurement".Q = false AND "DelayGridVoltageMeasurement".IN = true DO
  ;
END_WHILE;
```

Manualswitchonplcswitchon1-4 (FB)

```
//Turn on the PLC Switches of the load if and only if the corresponding manual switches are turned on, i.e. closed

IF "ManualSwitch_Load1" = TRUE THEN

  "PLCSwitch_Load1":= TRUE;

END_IF;

IF "ManualSwitch_Load2" = TRUE THEN

  "PLCSwitch_Load2":= TRUE;

END_IF;

//Turn on the PLC Switches of the load if and only if the corresponding manual switches are turned

IF "ManualSwitch_Load3" = TRUE THEN

  "PLCSwitch_Load3":= TRUE;

END_IF;

IF "ManualSwitch_Load4" = TRUE THEN

  "PLCSwitch_Load4":= TRUE;

END_IF;
```

Manualswitchonplcswitchoff1-4 (FB)

```
//Turn off the PLC Switches of the load if and only if the corresponding manual switches are turned on, i.e. closed

IF "ManualSwitch_Load1" = TRUE THEN

  "PLCSwitch_Load1":= FALSE;

END_IF;

IF "ManualSwitch_Load2" = TRUE THEN

  "PLCSwitch_Load2":= FALSE;
```

```
END_IF;
IF "ManualSwitch_Load3" = TRUE THEN
  "PLCSwitch_Load3":= FALSE;
END_IF;
IF "ManualSwitch_Load4" = TRUE THEN
  "PLCSwitch_Load4":= FALSE;
END_IF;
```

Managementofloadswitching (FB)

```
// Check if all manual switches are closed.
// manual_switches_off = true if all manual switches are closed
// else manual_switches_off = false
"CheckIfAllSwitchesOff"(all_manual_switches_off=> #manual_switches_off);
// Code corresponding to a case where all manual switches are off, i.e. open
IF #manual_switches_off THEN
  // Turn on the PLC switches of all the manual switches
  "TurnPLCSwitchesOn"();
  IF #delay_finished THEN
     // Check the grid voltage if it is within the band or not
     "PollGridVoltage_1"(within_band=> #grid_voltage_within_undervoltage);
     "PollGridVoltage_2"(within_band_2=> #grid_voltage_within_lowvoltage);
    //1st load shedding
    IF #grid_voltage_within_lowvoltage THEN
      "ManualSwitchOnPlCSwitchOff_34"();  //(10V-10.9V)
    // 2nd load shedding
```

```
    ELSIF #grid_voltage_within_undervoltage THEN
      "ManualSwitchOnPlCSwitchOff_12"();  //<9V)
    //switch on all switches
    ELSE "ManualSwitchOnPLCSwitchOn_12"();
        "ManualSwitchOnPLCSwitchON_34"();
    END_IF;
  END_IF;
  // Turn on (close) the PLC switches of all the manual switches that are open
  "TurnPLCSwitchesOn"();
END_IF;
```

Appendix M:

Storage Node Source code

input data prep

```
"Variables".GridVoltage := SCALE_X(MIN:= 0.0, VALUE:= NORM_X(MIN:= 0, VALUE:= "GridVoltageAnalogInput(ch1)", MAX:= 27648), MAX:= 10.0);

//sourcing mode - LV

IF "Variables".GridVoltage >= "Variables".gridLowVoltage AND "Variables".GridVoltage < "Variables".gridVoltage_OC_low THEN

//maximum allowed discharge current = 2.5A

"Variables".sourcingCurrent := SCALE_X(MIN:= 0.0, VALUE:= NORM_X(MIN:= 0, VALUE:= "sourcingCurrentAnalogInput(ch2)", MAX:= 27648), MAX:= 2.5);

ELSIF  "Variables".GridVoltage >= "Variables".gridHighVoltage AND "Variables".GridVoltage <= "Variables".gridVoltageUpperLimit THEN

//maximum charging current = 2A

"Variables".sinkingCurrent := SCALE_X(MIN:= 0.0, VALUE:= NORM_X(MIN:= 0, VALUE:= "sinkingCurrentAnalogInput(ch3)", MAX:= 27648), MAX:= 2);

ELSE

    "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;

    "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;

END_IF;

#curr := SCALE_X(MIN:= -10.0, VALUE:= NORM_X(MIN:= 0, VALUE:= "sourcingCurrentAnalogInput(ch2)", MAX:= 65535), MAX:= 10.0);

"Variables".batteryDischargingVoltage:= SCALE_X(MIN:= 0.0, VALUE:= NORM_X(MIN:= 0, VALUE:="BatteryVoltage(ch0)", MAX:= 27648), MAX:= 10.0);

"Variables".gridVoltageHMI := "Variables".GridVoltage * 2.0;

"Variables".DOD := "Variables".DODHMI / 100.0;

IF "Variables".DOD < 0 OR "Variables".DOD > 1 THEN

  "Variables".DOD := 0.5;

  "Variables".DODHMI := 50.0;

END_IF;

"ch2_out" := 13500; //constant reference out to DC-DC converter
```

---

output dataprep

"sourcingPWMVoltageControlOutput(ch0)" := SCALE_X(MIN:= 0, VALUE:= NORM_X(MIN:= 0.0, VALUE:= "Variables".sourcingPWMControl, MAX:= 10.0), MAX:= 27648);

"sinkingPWMVoltageControlOutput(ch1)" := SCALE_X(MIN:= 0, VALUE:= NORM_X(MIN:= 0.0, VALUE:= "Variables".sinkingPWMControl, MAX:= 10.0), MAX:= 27648);

GridControl FB

```
//Sourcing (discharging battery until V =13V)
IF "GridVoltageAnalogInput(ch1)" < #gridVoltageStableUpperThreshold AND "GridVoltageAnalogInput(ch1)" > #gridVoltage_0V THEN
  "Sourcing_Relay":= TRUE;//push power to the grid
//disconnect node when there is a short circuit
ELSIF "GridVoltageAnalogInput(ch1)" <= #gridVoltage_0V  THEN
  "Sourcing_Relay":= FALSE;
 //disconnect node when battery is depleated below (30%) and delay for 6s before attempting to restart
ELSIF "Variables".SOC < 30 THEN
  "Variables".LowBatAlarm := "Variables".LowBatAlarm + #LowBatCounter;
END_IF;
//low battery alarm timer before attempting to restart (6seconds)
WHILE "Variables".LowBatAlarm > #LowBaTimer &  "Variables".LowBatAlarm < #LowBaTimer_1 DO
  "Sourcing_Relay":= FALSE;
EXIT;
END_WHILE;
// Delay ------------------------------
IF "Delay".Q THEN
  #startDelay1 := false;
ELSE
  #startDelay1 := true;
END_IF;
"Delay".TON(IN:=#startDelay1,
            PT:=T#1ms,
            Q=>#delayQ,
```

```
        ET=>#delay1ET);
```

StorageGridControl FB

```
// "stateOfBattery = true" when the battery is a source

// The available ampere hours of the storage plant, given the DOD.

#useableAmpereHours := (1-"Variables".DOD) * "Variables".BatteryCapacity;

#ampereHoursAtDOD := "Variables".DOD * "Variables".BatteryCapacity;

// Case 1: under voltage condition

IF "Variables".GridVoltage < "Variables".gridVoltageLowerLimit AND "Variables".SOC >
#ampereHoursAtDOD

  AND "Variables".sourcingPWMControl <= "Variables".PWMmaxDC THEN

  "Variables".sourcingPWMControl := "Variables".sourcingPWMControl;

  "Variables".SOC :="Variables".SOC;

  "Variables".sourcingCurrent := "Variables".sourcingCurrent + #pwmIncrement;

   "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;

// Case 2: over voltage condition

ELSIF "Variables".GridVoltage > "Variables".gridVoltageUpperLimit AND "Variables".SOC >
#ampereHoursAtDOD

  AND "Variables".sourcingPWMControl < "Variables".PWMmaxDC THEN

    "Variables".sourcingPWMControl := "Variables".sourcingPWMControl + #pwmIncrement;

    "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;

     "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;

   END_IF;

 //Open circuit (no load)

 IF "Variables".GridVoltage > "Variables".gridVoltage_OC_low AND"Variables".GridVoltage
<= "Variables".gridVoltage_OC_upp

   AND "Variables".SOC > #ampereHoursAtDOD AND "Variables".sourcingPWMControl <
"Variables".PWMmaxDC AND #stateOfBattery = true THEN

      "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;

      "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;
```

```
END_IF;

// Case 3 :low voltage condition (battery discharging/sourcing)

IF "Variables".GridVoltage >= "Variables".gridVoltageLowerLimit AND
"Variables".GridVoltage < "Variables".gridLowVoltage

  AND "Variables".SOC > #ampereHoursAtDOD AND "Variables".sourcingPWMControl <
"Variables".PWMmaxDC AND #stateOfBattery = true THEN

  "Variables".sourcingPWMControl := "Variables".sourcingPWMControl + #pwmIncrement;

  "Variables".sourcingCurrent := "Variables".sourcingCurrent + #pwmIncrement;

   "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;

  "Variables".SOC := "Variables".SOC - 0.001;

// if grid voltage is within the band and there is available charge and the direction of current is
positive (i.e. current flow is to the grid - sourcing)

ELSIF "Variables".GridVoltage >= "Variables".gridLowVoltage AND "Variables".GridVoltage
< "Variables".gridVoltage_OC_low

  AND "Variables".SOC > #ampereHoursAtDOD AND #stateOfBattery = true THEN

  "Variables".SOC := "Variables".SOC - 0.001;

  "Variables".sourcingCurrent := "Variables".sourcingCurrent + #pwmIncrement;

  "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;

  // if available charge is depleted then switch to sinking mode

 ELSIF  "Variables".SOC <= #ampereHoursAtDOD AND #stateOfBattery = true THEN

   "Variables".sourcingPWMControl := "Variables".PWMminDC;

   "Variables".sinkingPWMControl := "Variables".PWMminDC;

   #stateOfBattery := false;

 END_IF;

// Case 2: The storage is operating as a sink (Charging) high voltage mode

IF "Variables".GridVoltage >= "Variables".gridHighVoltage AND "Variables".GridVoltage <=
"Variables".gridVoltageUpperLimit

  AND "Variables".SOC < "Variables".BatteryCapacity AND "Variables".sinkingPWMControl <
"Variables".PWMmaxDC AND #stateOfBattery = true THEN

  IF "Variables".batteryDischargingVoltage < 9.0  THEN

    "Variables".sinkingPWMControl := "Variables".sinkingPWMControl + #pwmIncrement;
```

```
    "Variables".sinkingCurrent := "Variables".sinkingCurrent + #pwmIncrement;
    "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;
    "Variables".SOC := "Variables".SOC + 0.001;
  END_IF;
  "Variables".SOC := "Variables".SOC + 0.001;
  "Variables".sinkingCurrent := "Variables".sinkingCurrent + #pwmIncrement;
  "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;
ELSIF "Variables".GridVoltage > "Variables".gridVoltage12V_UP AND
"Variables".GridVoltage < "Variables".gridVoltageUpperLimit
  AND #stateOfBattery = false AND "Variables".sinkingPWMControl >
"Variables".PWMminDC AND "Variables".SOC < "Variables".BatteryCapacity THEN
  "Variables".sinkingPWMControl:= "Variables".sinkingPWMControl + #pwmIncrement;
  "Variables".SOC := "Variables".SOC + 0.001;
   "Variables".sinkingCurrent := "Variables".sinkingCurrent + #pwmIncrement;
     "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;
  //if battery is full reset all controls
ELSIF "Variables".SOC = "Variables".BatteryCapacity AND #stateOfBattery = false THEN
  "Variables".sinkingPWMControl := "Variables".PWMminDC;
  "Variables".sourcingPWMControl := "Variables".PWMminDC;
  "Variables".SOC := "Variables".BatteryCapacity;
      "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;
     "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;
  #stateOfBattery := true;
  //reset to start value in undervoltage
ELSIF "Variables".GridVoltage < "Variables".gridVoltageLowerLimit AND
"Variables".GridVoltage > "Variables".gridVoltage0V AND #stateOfBattery = false THEN
  "Variables".sourcingPWMControl := "Variables".PWMminDC;
  "Variables".sinkingPWMControl := "Variables".PWMminDC;
  "Variables".sourcingCurrent := "Variables".sourcingCurrent - "Variables".sourcingCurrent;
  "Variables".sinkingCurrent := "Variables".sinkingCurrent - "Variables".sinkingCurrent;
```

```
  #stateOfBattery := true;
END_IF;
// Delay ---------------------------------
IF #delayQ = false THEN
  #startDelay := true;
ELSE
  #startDelay := false;
END_IF;
"Delay".TON(IN:= #startDelay,
            PT:= T#2ms,
            Q=> #delayQ,
            ET=> #delayET);
// if sourcing pwm voltage exceeds the set max, then limit it to set max pwm voltage
IF "Variables".sourcingPWMControl >= "Variables".PWMmaxDC THEN
  "Variables".sourcingPWMControl := "Variables".PWMmaxDC;
END_IF;
// if sinking pwm voltage exceeds the set max, then limit it to set max pwm voltage
IF "Variables".sinkingPWMControl >= "Variables".PWMmaxDC THEN
  "Variables".sinkingPWMControl := "Variables".PWMmaxDC;
 //  "Variables".SOC := "Variables".SOC + 0.001;
 END_IF;
IF "Variables".sinkingPWMControl <= "Variables".PWMminDC THEN
  "Variables".sinkingPWMControl := "Variables".PWMminDC;
 END_IF;
```

Appendix N:

Picogrid Bus Voltage LED indicator PLC code

```
// Responsible FOR indicating the current voltage state OF the Picogrid Bus voltage (PBV)

// Convert the digital count grid voltage to a an analogue grid voltage

#grid_voltage_2 := UINT_TO_REAL("GridVoltage_2");

#grid_voltage:= UINT_TO_REAL("GridVoltage");

#normalised_grid_voltage:= #grid_voltage/"GeneralParameters".full_scale_range;

#normalised_grid_voltage_2:= #grid_voltage_2 / "GeneralParameters".full_scale_range;

#analogue_grid_voltage := ((#normalised_grid_voltage + #normalised_grid_voltage_2) / "GeneralParameters".grid_voltage_scale_factor) ;

  // Case 1 :  The grid voltage is in under voltage  ( < 9.7 V)

 IF #analogue_grid_voltage < #gridVoltageUnderVoltage &"Delay".Q THEN

    "LED_UV":= TRUE;

    "LED_LV":= FALSE;

    "LED_stable":= FALSE;

    "LED_HV":= FALSE;

    "LED_OV":= FALSE;

  END_IF;

  // Case 2 :  The grid voltage is in over voltage  ( > 14 V)

  IF #analogue_grid_voltage > #gridVoltageOverVoltage &"Delay".Q THEN

    "LED_UV":= FALSE;

    "LED_LV":= FALSE;

    "LED_stable":= FALSE;

    "LED_HV":= FALSE;

    "LED_OV":= TRUE;
```

```
    END_IF;

    IF #analogue_grid_voltage < #gridVoltageLVlowerThreshold AND #analogue_grid_voltage >
#gridVoltageUnderVoltage &"Delay".Q THEN

      "LED_UV":= TRUE;

      "LED_LV":= FALSE;

      "LED_stable":= FALSE;

      "LED_HV":= FALSE;

      "LED_OV":= FALSE;

    // Case 3 :  The grid voltage is in low voltage mode (10 - 10.9)

    ELSIF #analogue_grid_voltage > #gridVoltageLVlowerThreshold & #analogue_grid_voltage <
#gridVoltageStableLowerThreshold & "Delay".Q THEN

      "LED_UV":= FALSE;

      "LED_stable":= FALSE;

      "LED_OV":= FALSE;

      "LED_LV":=TRUE;

      "LED_HV":=FALSE;

//  Case 4 : The grid voltage is between the range 11 - 13V (stable operation)

    ELSIF #analogue_grid_voltage > #gridVoltageStableLowerThreshold &
#analogue_grid_voltage < #gridVoltageStableUpperThreshold & "Delay".Q THEN

      "LED_UV":= FALSE;

      "LED_LV":= FALSE;

      "LED_HV":= FALSE;

      "LED_OV":= FALSE;

      "LED_stable":= TRUE;


    END_IF;

// Case 5 :  The grid voltage is in high voltage mode (13.2 - 14V)

IF #analogue_grid_voltage > #gridVoltageStableUpperThreshold & #analogue_grid_voltage <=
```

```
#gridVoltageHVupperThreshold & "Delay".Q THEN
    "LED_HV":= TRUE;
    "LED_stable":= FALSE;
    "LED_UV":= FALSE;
    "LED_LV":= FALSE;
    "LED_OV":= FALSE;
END_IF;
// Case 6 :  The grid voltage is in over voltage mode ( > 14.2V)
IF #analogue_grid_voltage > #gridVoltageHVupperThreshold & "Delay".Q THEN
    "LED_OV":= TRUE;
    "LED_stable":= FALSE;
    "LED_UV":= FALSE;
    "LED_LV":= FALSE;
    "LED_HV":= FALSE;
 END_IF;
// Delay -----------------------------
IF "Delay".Q THEN
 #startDelay1 := false;
ELSE
 #startDelay1 := true;
END_IF;
"Delay".TON(IN:=#startDelay1,
          PT:=T#1s,
          Q=>#delayQ,
          ET=>#delay1ET);
```

Appendix O:

Additional Graphical results and Analysis

1.1. Combined conditional and non-conditional load profile

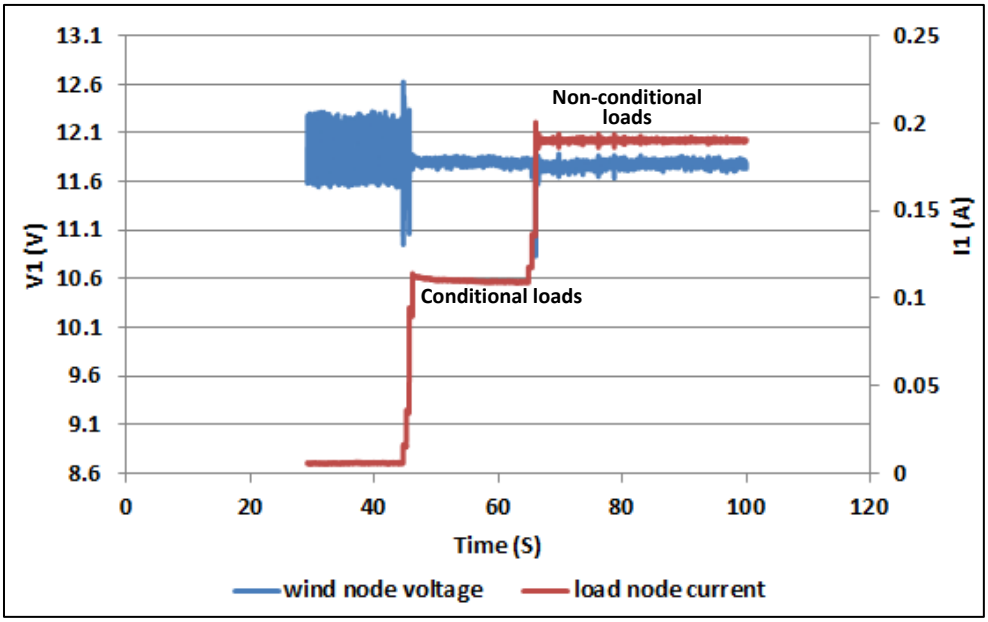


Figure O1: Conditional and non-conditional loads profile

1.2. System Tests – Load connected on Wind Node (case 1)

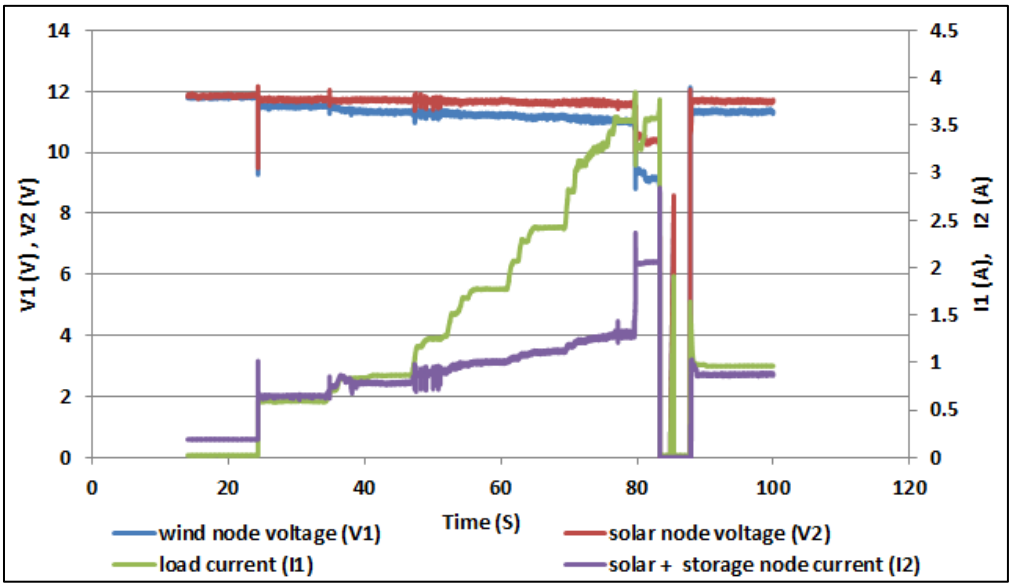


Figure O2: voltage and current versus time

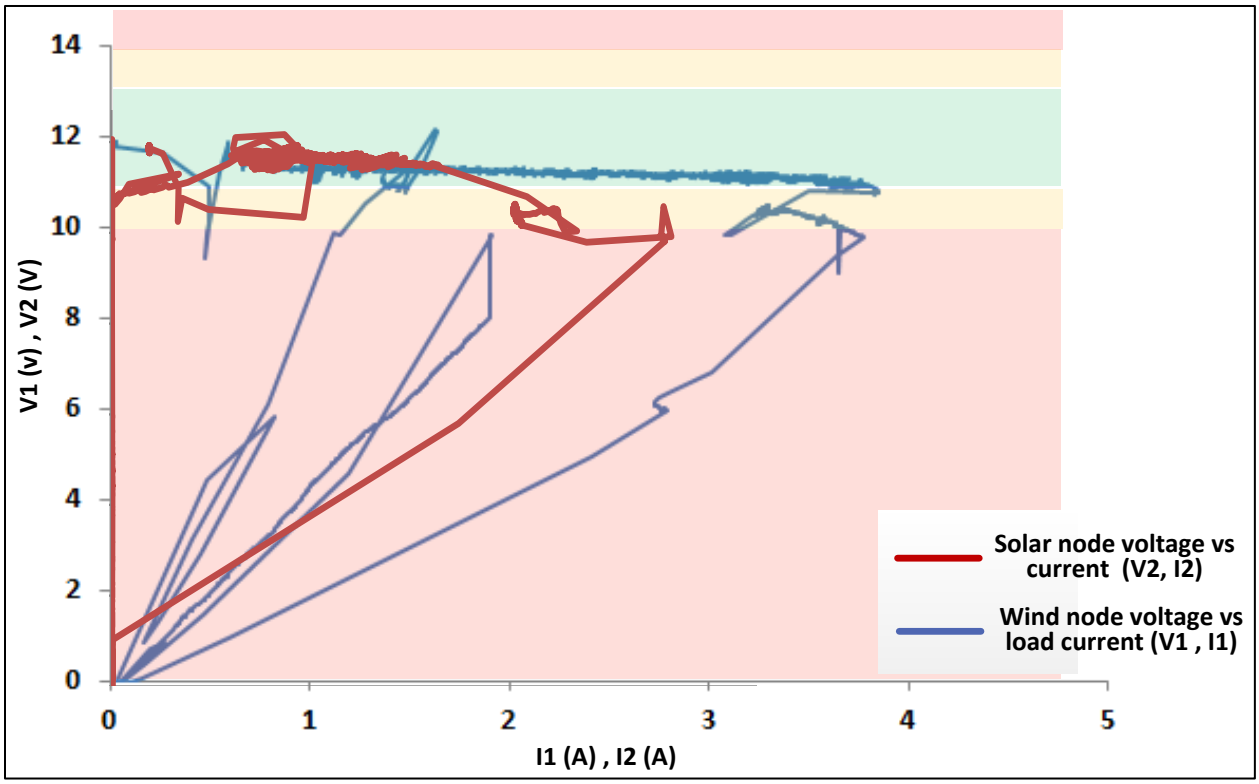


Figure O3: Voltage versus current measurements with load on wind node

1.3. System Tests – Load connected on Storage Node (case 3)

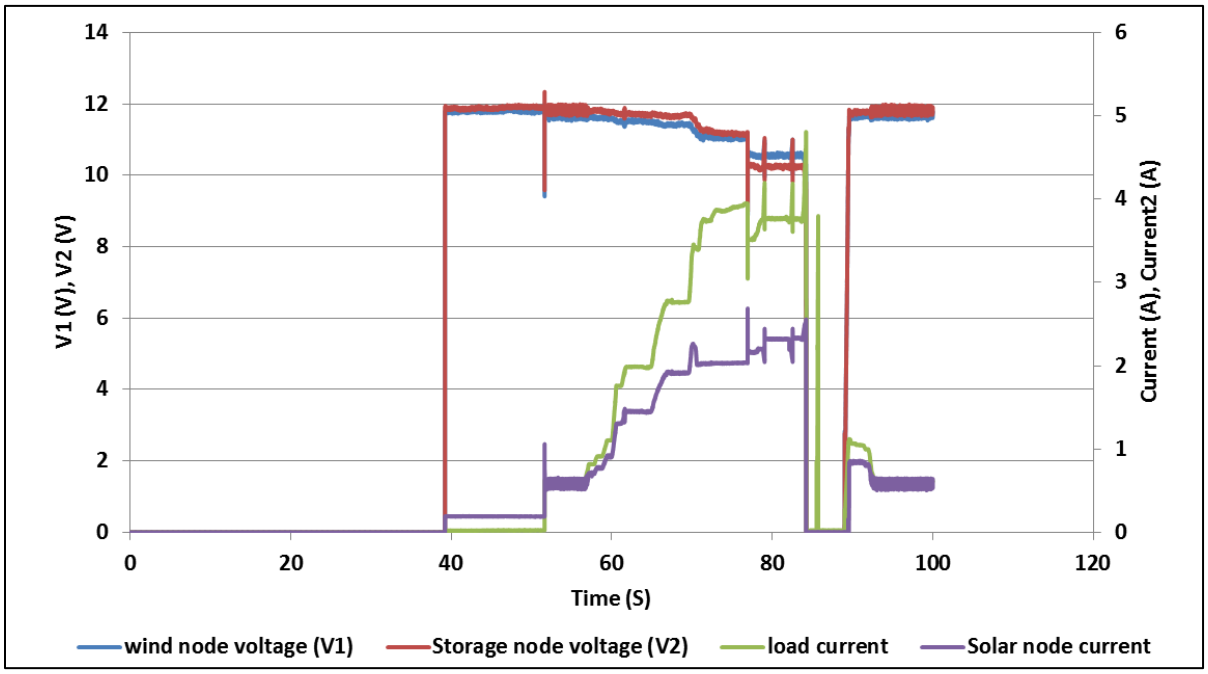


Figure 1: voltage and current versus time

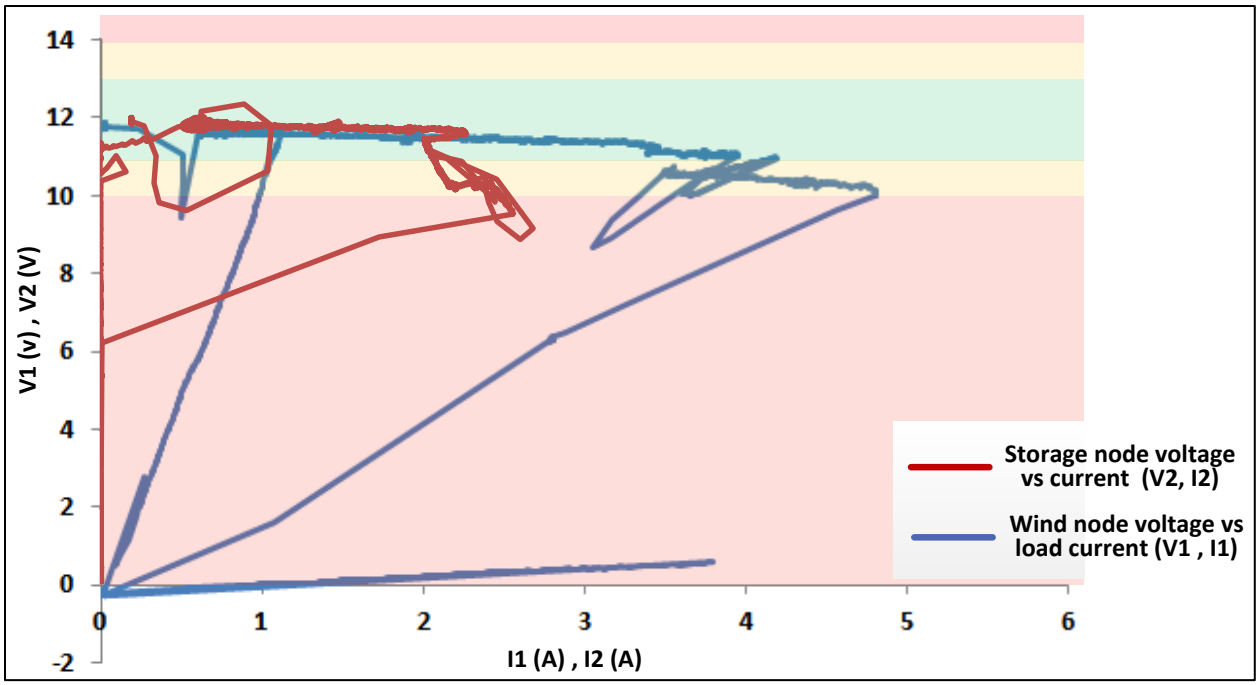


Figure O5: Voltage versus current measurements with load on storage node

1.4. System Tests – Load connected on Solar Node (case 4)

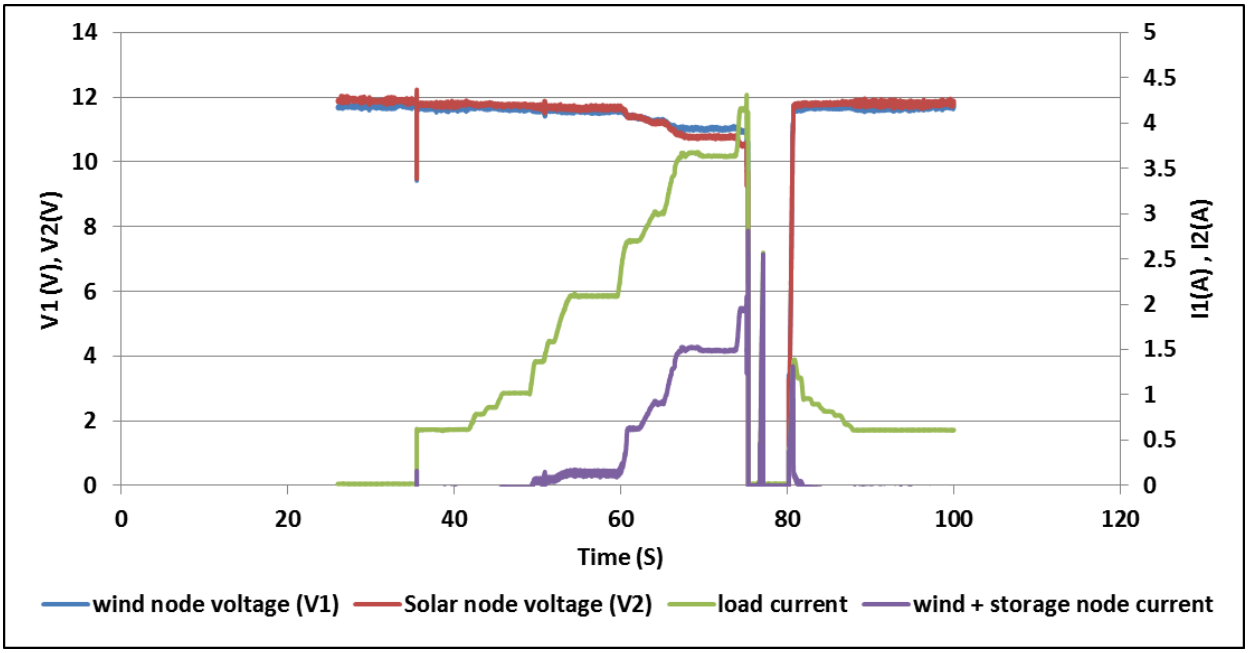


Figure O6: voltage and current versus time

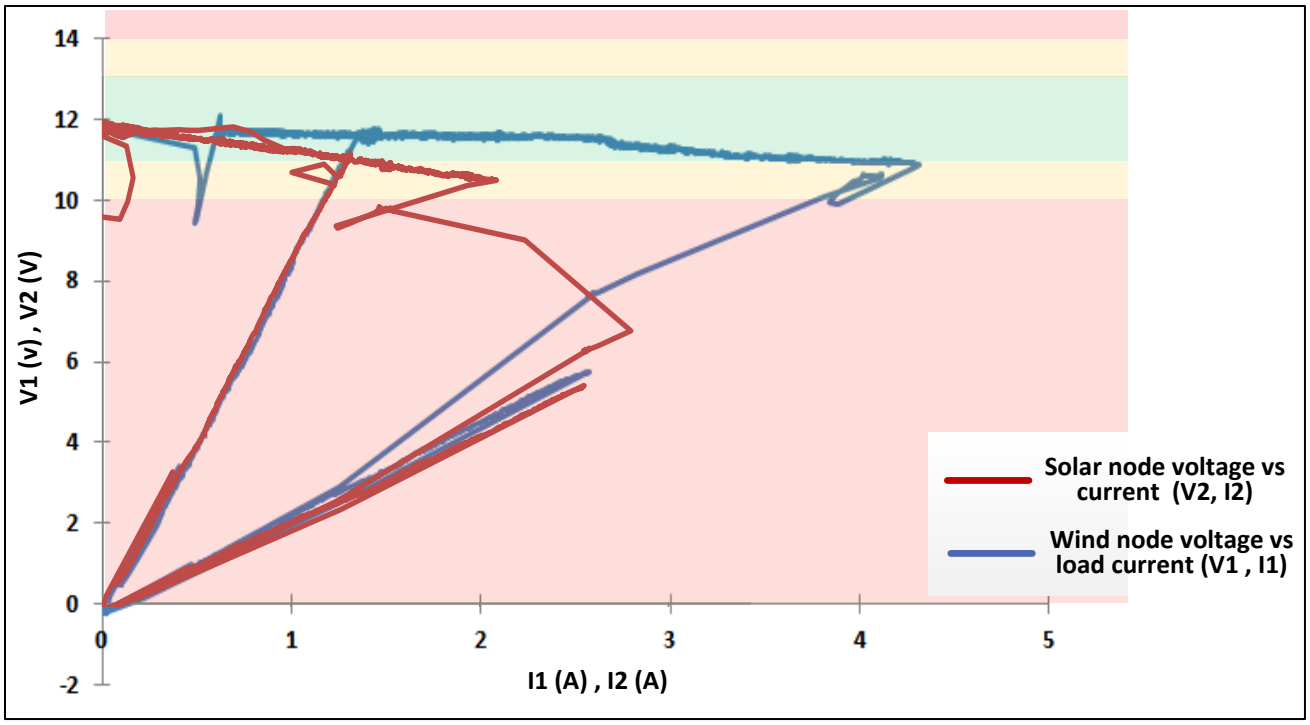


Figure O7: Voltage versus current measurements with load on solar node

Appendix P:

SIEMENS PLC Tags and Functions

1. Control System Architecture

A standardized high-level textual programming language known as Structured Control Language (SCL) [46] is utilised by the Siemens PLCs for each node. This means that the logic behind the programming can easily be transferred to different controllers at any stage of implementation. As shown in figure P1, SCL is compiled in the Totally Integrated Automation (TIA) portal development environment [46] in order to execute commands. This environment not only combines different modules of the PLC such as analog inputs and Human Machine Interface (HMI) but also offers an efficient debugging tool which enhances the efficiency of the algorithm.

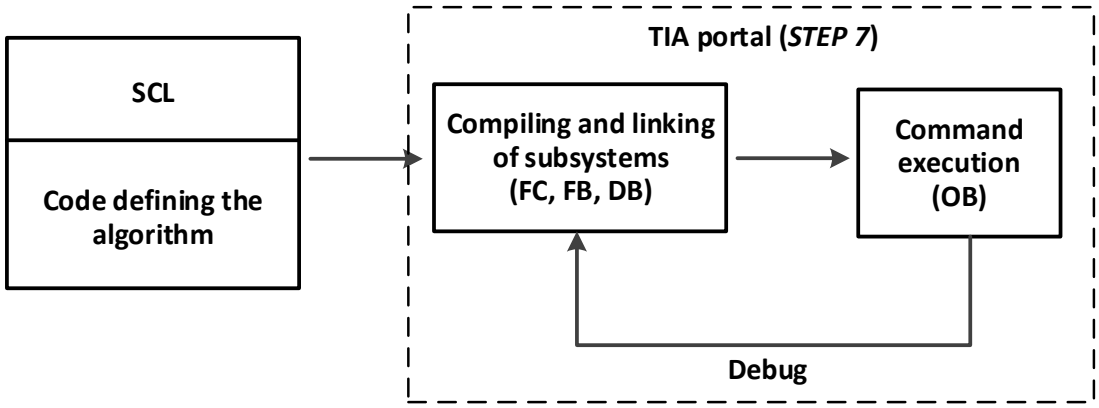


Figure P1: Siemens SCL and TIA portal [46]

STEP 7 is the programming and configuration software component of a TIA development platform. TIA platform offers an editor, a batch compiler as well as a debugging tool for analysing STEP 7 software. As shown in figure….the editor consists of Function Blocks (FBs), Data Blocks (DBs) and Organisation Blocks (OBs) for an efficient structured programming. All the blocks, with an exception of Organisation Blocks, contain the SCL codes for different functions. An Organisation Block brings all the blocks together and offers an interactive block structure of the overall control algorithm. Each node thus has one OB that shows how all functions are linked together.

2. Source Nodes (Solar and Wind)

Siemens STEP programming software has function declarations under PLC tags. These tags link the hardware to the software interface in the TIA portal by assigning unique addresses to each analog input or output as shown by the snapshots taken from both solar and wind nodes PLC tags in figure P2 and figure P3

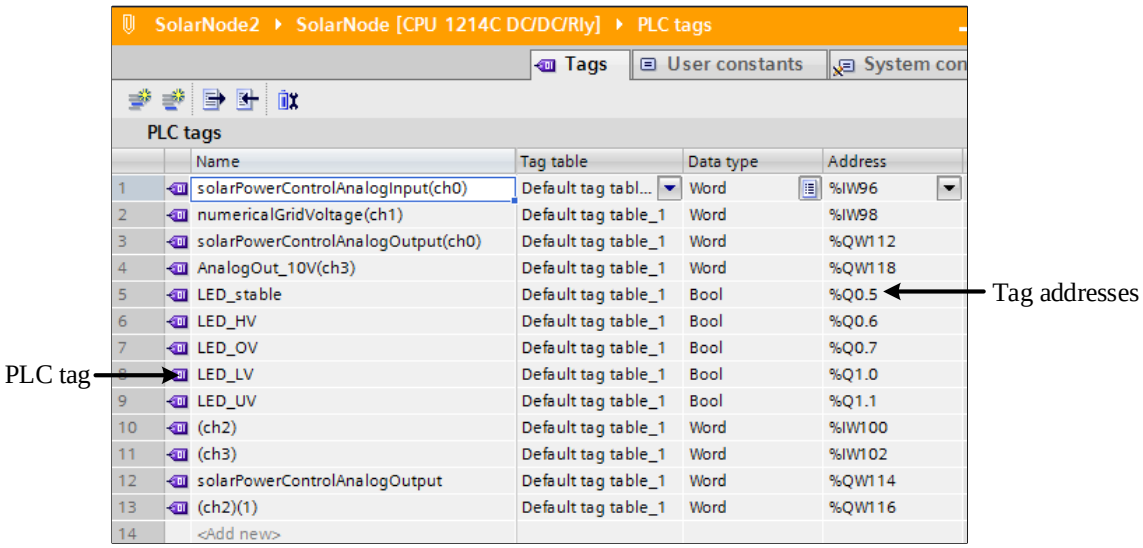


Figure P2: Solar node PLC tags

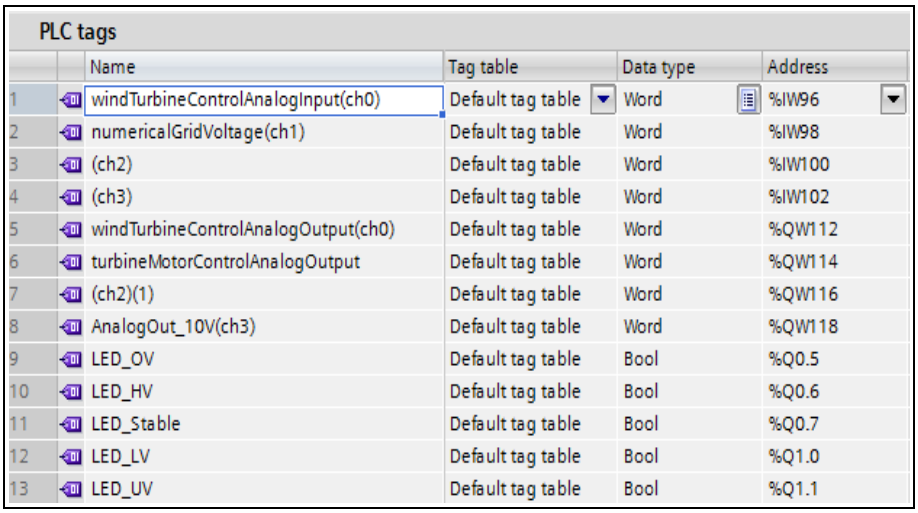


Figure P3: Wind node PLC tags

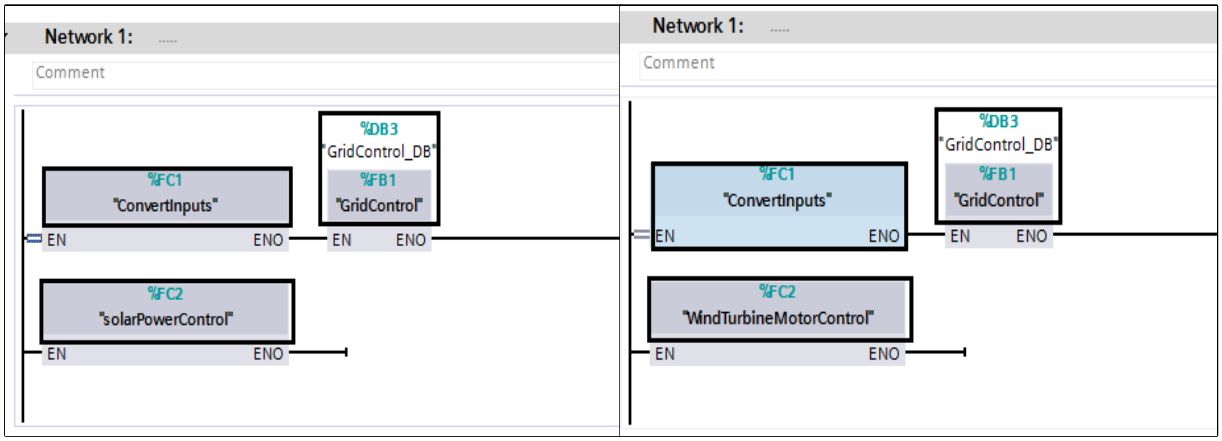
From figure…and figure….both source nodes have the following function declarations (tags);

- ***SolarPowerControlAnalogInput / windTurbineControlAnalogInput*** – Responsible for obtaining the voltage value presented by the solar and wind node regulator knobs.
- ***NumericalGridVoltage*** – Obtains the PBV from grid connection ports.
- ***SolarPowerControlAnalogOutput / WindTurbineControlAnalogOutput*** – Obtains voltage values from the regulator knobs.

- ***LED_stable, LED_HV, LED_OV, LED_LV, LED_UV*** – PBV LED indicators

Application of these tags is done via coding in the function blocks of both solar and wind node source codes in Appendix J and Appendix K respectively.

Figure P4 (a) and (b) show the main Organisation Blocks of solar and wind nodes. The function first runs two functions concurrently; "convertinputs" and "solarpowercontrol / windturbinemotorcontrol" function before calling the "gridcontrol" function.



(a) (b)

Figure P4: Snapshot of PLC organisation Blocks for (a) solar node and (b) wind node

From the figure, the following functions were implemented on both source nodes;

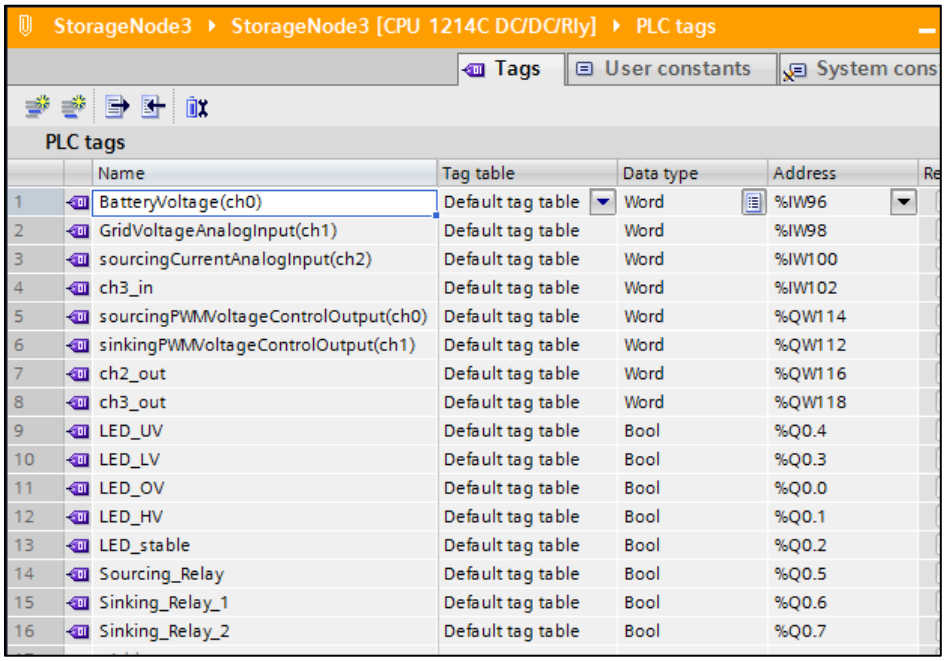
- ***SolarPowerControl / WindTurbineMotorControl*** – These functions use the same logic of providing a reference voltage for the flyback converter powering and driving the solar light and wind node motor respectively. This voltage is obtained by using a straight line mapping algorithm of the voltages obtained from the regulator knobs.
- ***ConvertInputs*** – Converts analog input values from machine's language (data word [2]) to real numerical values.
- ***GridControl*** – Obtains the converted voltage input values and applies the grid code with each PBV state. PBV LED indicator code is also incorporated inside this function.

3. Storage Node

Figure P5 shows a snapshot of a storage node's PLC tags defining the functions used for the storage node algorithm. The following tags were defined;

- ***GridVoltageAnalogInput*** – Obtains the PBV from grid connection ports.

- ***SourcingCurrentAnalogInput*** – Records the sourcing current from the storage node's flyback converter.
- ***SourcingPWMvoltageControlOutput*** – Outputs the voltage.
- ***SinkingPWMvoltageControlOutput*** – Outputs
- ***Sourcing_Relay*** **–** Opens and closes PLC Switch responsible for controlling sourcing current.
- ***Sinking_Relay_1*** **-** Opens and closes PLC Switch responsible for sinking current for high voltage PBV state.
- ***Sinking_Relay_2*** **-** Opens and closes PLC Switch responsible for sinking current during over voltage PBV state.



	Name	Tag table	Data type	Address
1	BatteryVoltage(ch0)	Default tag table	Word	%IW96
2	GridVoltageAnalogInput(ch1)	Default tag table	Word	%IW98
3	sourcingCurrentAnalogInput(ch2)	Default tag table	Word	%IW100
4	ch3_in	Default tag table	Word	%IW102
5	sourcingPWMVoltageControlOutput(ch0)	Default tag table	Word	%QW114
6	sinkingPWMVoltageControlOutput(ch1)	Default tag table	Word	%QW112
7	ch2_out	Default tag table	Word	%QW116
8	ch3_out	Default tag table	Word	%QW118
9	LED_UV	Default tag table	Bool	%Q0.4
10	LED_LV	Default tag table	Bool	%Q0.3
11	LED_OV	Default tag table	Bool	%Q0.0
12	LED_HV	Default tag table	Bool	%Q0.1
13	LED_stable	Default tag table	Bool	%Q0.2
14	Sourcing_Relay	Default tag table	Bool	%Q0.5
15	Sinking_Relay_1	Default tag table	Bool	%Q0.6
16	Sinking_Relay_2	Default tag table	Bool	%Q0.7

Figure P5: Storage node PLC tags

Figure P6 shows the main Organisation Blocks of solar and wind nodes. The function first runs two functions concurrently; "convertinputs" and "solarpowercontrol / windturbinemotorcontrol" function before calling the "gridcontrol" function.

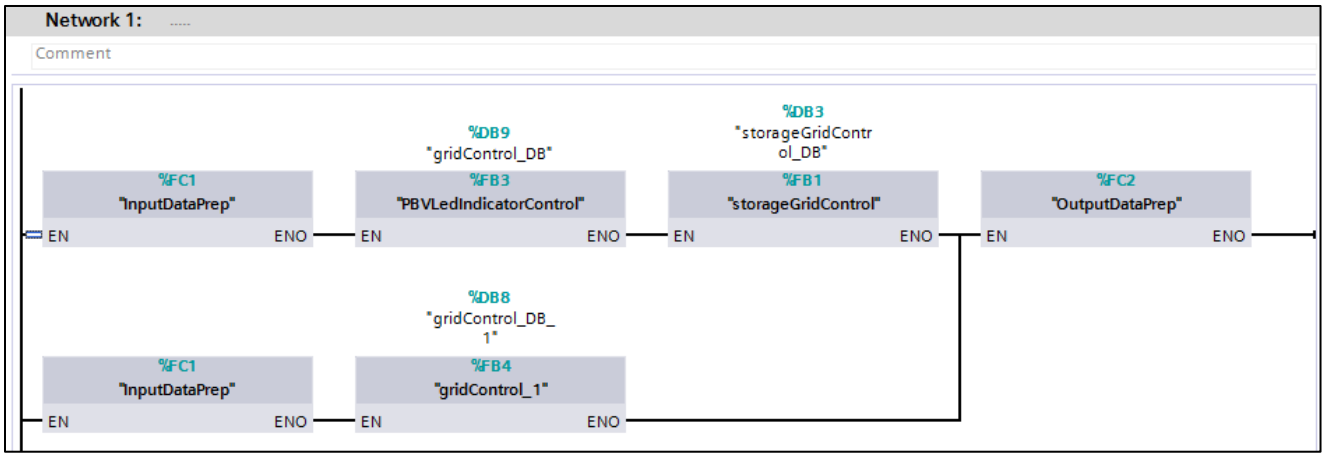


Figure P6: Snapshot of PLC organisation Blocks for storage node

From the figure, the following functions were implemented on a storage node;

- ***StorageGridControl*** – Checks what state the picogrid is in and responds by switching on the corresponding PBV LED
- ***gridControl* –** Connects the node to the grid when the bus voltage drops below 11V or the battery's State Of Charge (SOC) is above 30%.

4. Load Node

The structural control block diagram shown in figure P6 shows the interaction of different functions (FB) forming the node's control logic. A PLC first checks whether all manual switches are off then closes the PLC relay switches if this is true. If the function returns a false, the system delays a while before starting to poll the grid voltage. The grid voltage is then verified to be within the correct band as per the grid code. If the voltage is within band then PLC switches are closed only if the corresponding manual switches are closed on otherwise the PLC switches are open. The PLC switches are then closed for all the manual switches that are open before repeating the same process again.

The algorithm uses Function Blocks (FB) as shown by the SCL code in Appendix L in order to define the load node's characteristics. A load node organizational block only executes one function (*ManagementOfLoadSwitching*) which is responsible for linking all the other functions described below. This defines the SCL algorithm used by the PLC to control switching of the loads.

- ***TurnPLCSwitchesOn* –** Closes the PLC switch only if the corresponding manual switch for the load is opened.
- ***CheckIfAllSwitchesOff* –** Responsible for checking if all manual switches are open before switching on the corresponding PLC switches.

- ***Pollgridvoltage*** – This function obtains the mean PBV and combines them in to one voltage which is then converted to real values as per the grid code.
- ***ManualSwitchOnPLCSwitchOn*** – Closes the PLC switch only if the corresponding manual switch is closed.
- ***ManualSwitchOnPLCSwitchOff*** – Opens the PLC switch only if the corresponding manual switch is closed.
- ***ManagementOfLoadSwitching*** – This is the main function which manages switching of the loads by linking all the other function as shown by the OB snapshot of figure P6.

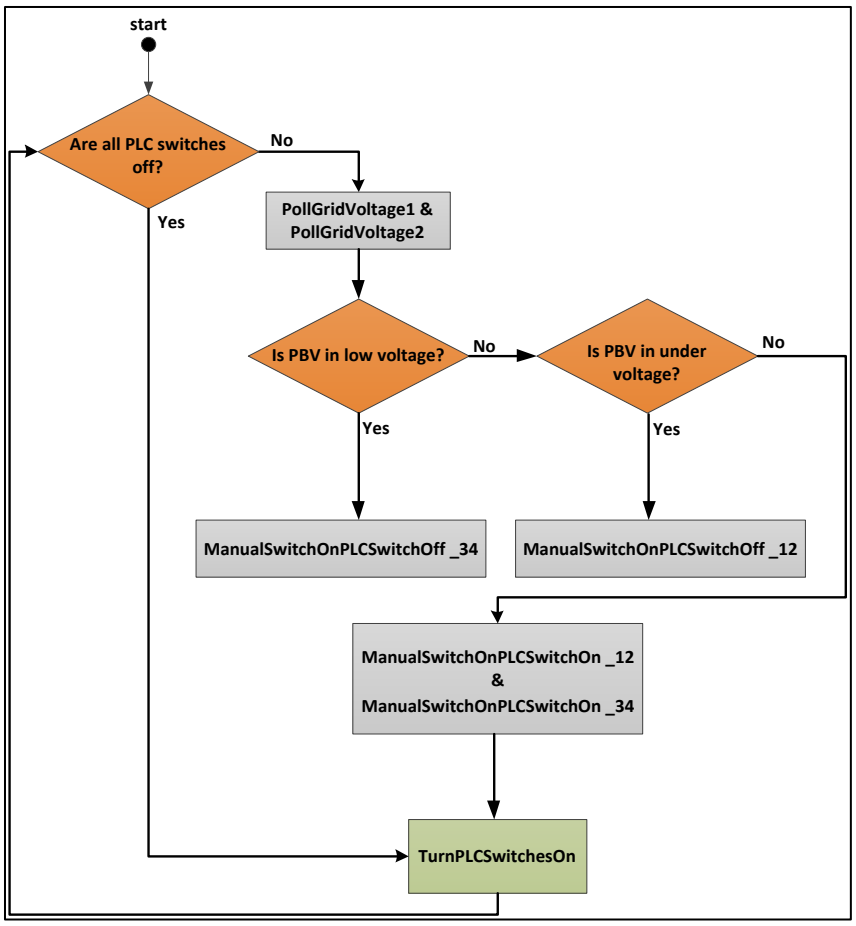


Figure P6: Load node structural control functions

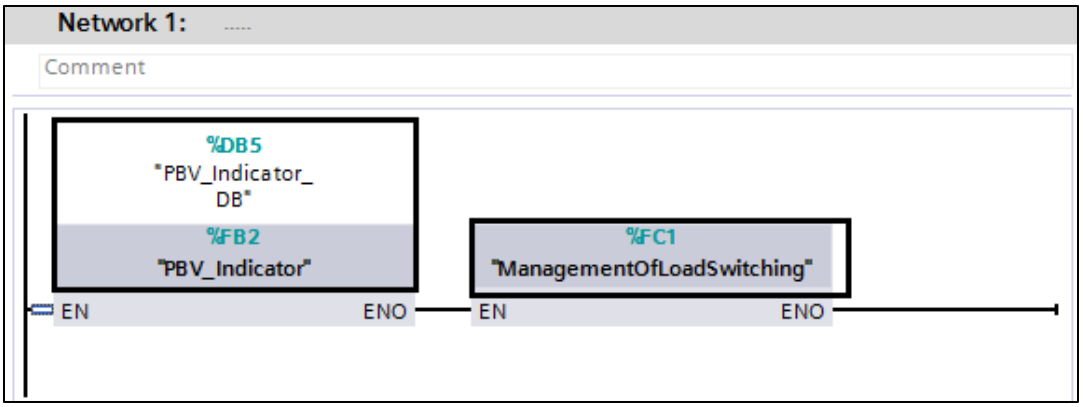


Figure P7: Load node main OB snapshot