

Implementing Photovoltaic Battery Charging System using C2000 Microcontrollers on Solar Explorer Kit

v 0.1, 1/23/2014

Manish Bhardwaj
C2000 Systems and Applications Team

ABSTRACT

Energy from renewable sources such as solar and wind, is gaining interest as the world's power demands increase and non-renewable resources deplete. Photovoltaic (PV) energy sources are considered an essential element in gaining independence from non renewable sources. Clean, ubiquitous PV is well suited except for the fact that it's only available in day time. Thus energy storage is considered a key factor in enabling increased use of PV for a variety of systems.

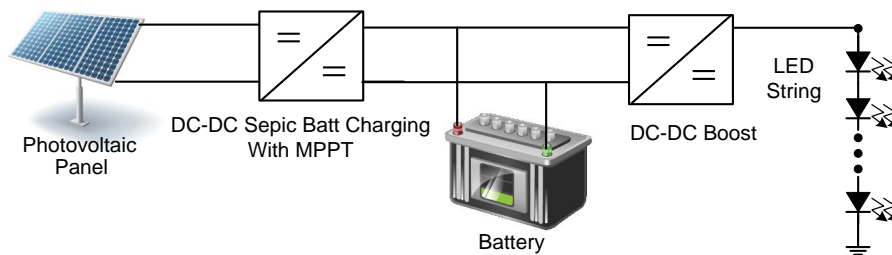


Fig 1 PV Street Lighting System

PV systems can be classified broadly into DC-DC and DC-AC systems. Adding storage capability in form of batteries enables PV systems to deliver energy at night time. This guide presents a PV Street Lighting system with battery charging system software, which implements the key features needed by a PV charger like MPPT, bulk charging, float charging and boost stage current LED control using C2000 MCU. The guide uses the Solar Explorer Kit (TMDSSOLARPEXPKIT) platform to illustrate the system with a F28035 control card.

Contents

- ABSTRACT 1**
- 1 INTRODUCTION 3**
 - 1.1 SOLAR EXPLORER KIT.....3
 - DC-DC Sepic with MPPT*..... 4
 - DC-DC Boost with MPPT*..... 5
 - 1.2 PV BATTERY CHARGING AND LED LIGHTING CONTROL DIAGRAM.....5
- 2 SOFTWARE & CONTROL DESCRIPTION..... 7**
 - 2.1 PROJECT FRAMEWORK7
 - 2.2 PROJECT DEPENDENCIES & RESOURCES9
 - 2.3 CONTROL DESCRIPTION9
 - DC-DC Sepic MPPT Control Software* 9
 - DC-DC Boost LED Current Control* 13
- 3 HARDWARE PLATFORM SETUP..... 16**
 - 3.1 HW SETUP INSTRUCTIONS16
 - 3.2 SOFTWARE SETUP17
- 4 PV BATTERY CHARGING AND LED DRIVING PROJECT 19**

1 INTRODUCTION

PV systems use battery storage to deliver power during time when PV energy is not present or the rate of PV energy available is not the same as the rate of consumption of the load in the system. DC-DC systems such as PV powered street light and standalone PV inverters (DC-AC) for rural electrification are examples of such applications. This guide provides a reference framework for such applications and discusses solutions for the key challenges in PV battery charging using an example of a PV street lighting application.

Lead acid is the predominant rechargeable battery type due to its low cost, capability to deliver large load current and wide used in automobiles which offsets its low power density. A 3 step or 2 step charger are typically used which employ current based or voltage based control depending on the charge state of the battery. The framework provides a voltage mode and current mode control example which can be applied to different battery type and charging scheme and profile. A state machine is presented for operation of the street light which can be modified by the end user easily for a particular application and battery.

1.1 Solar Explorer Kit

Solar Explorer Kit is a low voltage platform to evaluate C2000 microcontroller family of devices for renewable energy applications. The Solar Explorer kit can be used to implement a PV solar street lighting system by connecting the power stages as shown in Fig 2. Fig 3 gives a block diagram of different stages present on the Solar Explorer kit that are used for the PV street lighting system which employs battery charging and led string control. The input to the solar explorer kit is a 20V DC power supply which powers the controller and the supporting circuitry. A 50W solar panel can be connected to the board (Typical values V_{mpp} 17V, P_{max} 50W). However for quick demonstration of the power processing, a PV emulator power stage is integrated on the board along with other stages that are needed to process power. The control of the PV panel is kept separate from the control of the other stages. As PV is a light dependent source, the PV panel emulator can be used to test PV system under different lighting conditions. As the control of PV panel is executed on a separate controller a SPI link is added from the DIMM100 on the solar explorer to the PV Panel emulator controller. This simplifies the debug and demonstration. Details on the hardware and power stages present on the board can be found at:

```
controlSUITE\development_kits\  
SolarExplorer_vx.x\~Docs\SolarExplorer_HWGuide.pdf
```

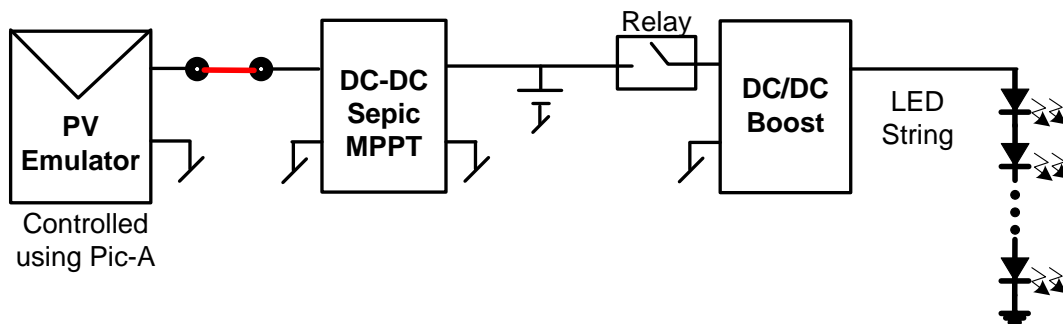


Fig 2 PV Inverter using Solar Explorer Kit

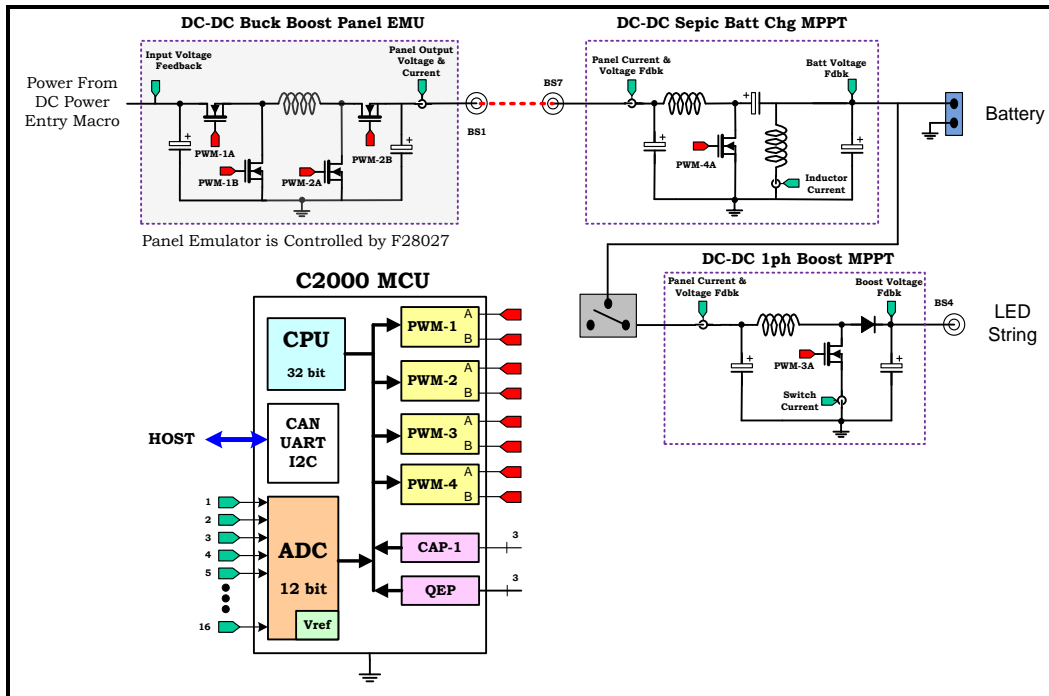


Fig 3 Solar Explorer Kit Power Stages for a PV Battery Charging and LED String Lighting System

DC-DC Sepic with MPPT

Input to this stage can come from Panel emulator block or externally connected solar panel. Fig 4 shows the power stage circuit implemented on solar explorer kit for this stage. Inductor L1, L2, MOSFET switch Q1 and diode D1, together form the sepic stage. **Error! Reference source not found.** illustrates the control scheme for the DC-DC sepic stage with MPPT for battery charging.

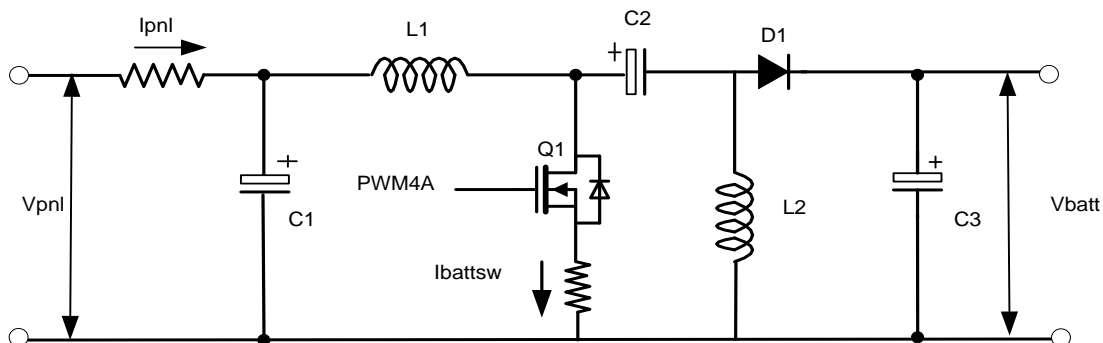


Fig 4 DC DC Sepic stage power circuit

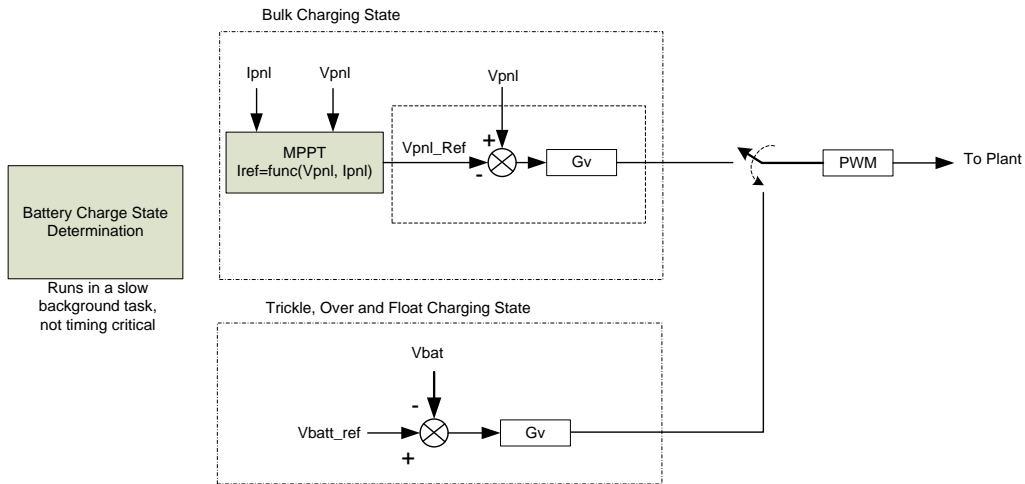


Fig 5 Control of DC-DC Sepic with MPPT

DC-DC Boost with MPPT

Input to this stage can come from Panel or the Battery output. Fig 6 shows the power stage circuit implemented on solar explorer kit for this stage. Inductor L1, MOSFET switch Q1 and diode D1, together form the boost circuit. The boost circuit operates at 100 KHz.

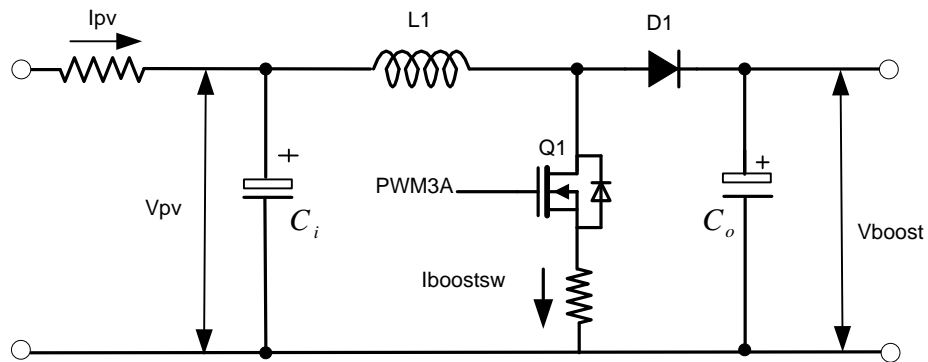


Fig 6 DC DC Boost stage power circuit

1.2 PV Battery Charging and LED lighting control diagram

Fig 7 illustrates the control scheme for a PV battery charger and LED string current control system. It is clearly noted that there are two Interrupt Service Routines (ISRs) one for closed loop control of the DC-DC sepic stage used to charge the battery (50KHz, every alternate switching period) and other for the closed loop control of DC-DC boost stage used to control the LED string current (50KHz, every alternate switching period).

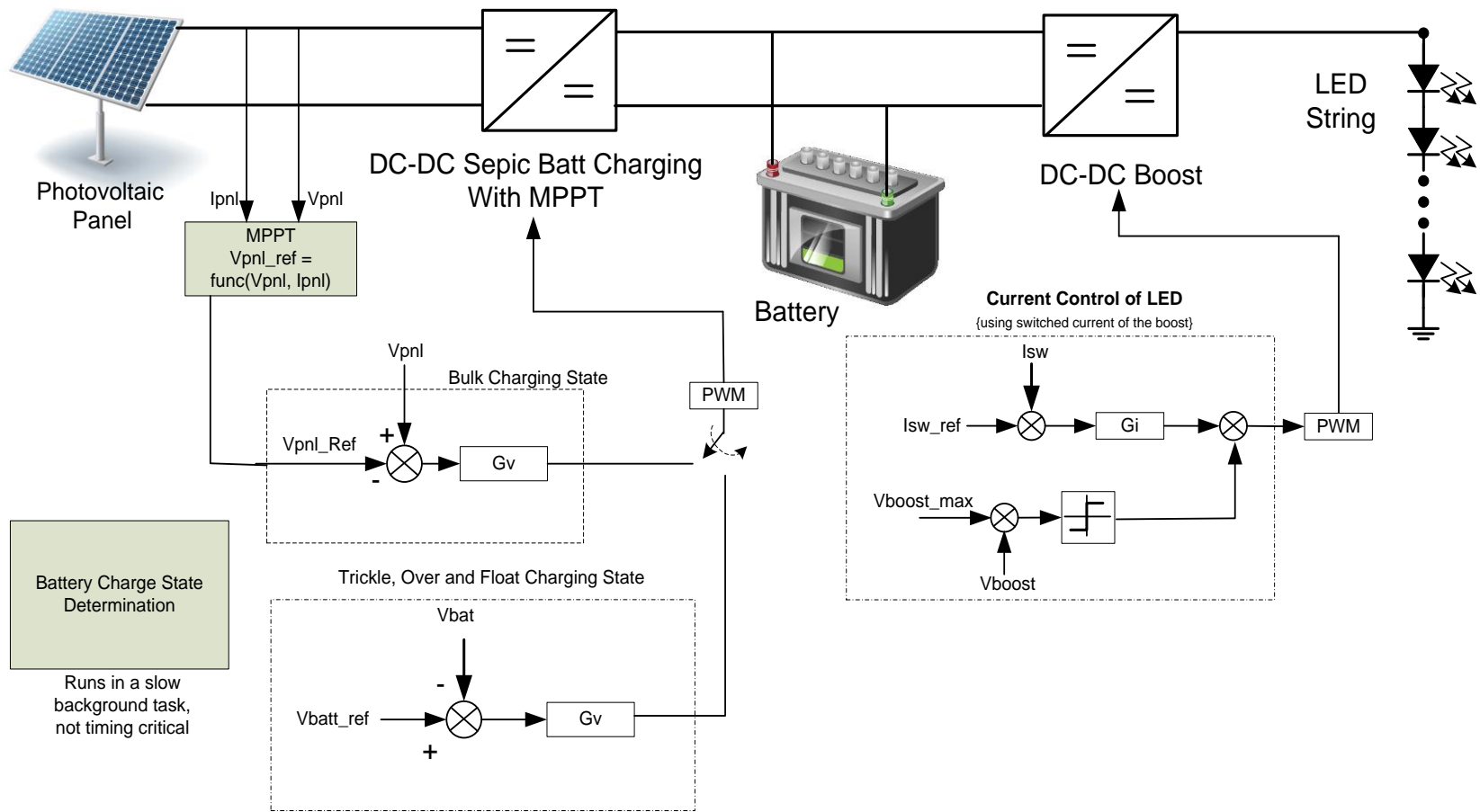


Fig 7 Control of Battery Charging with LED string control

2 Software & Control Description

This section describes the details of control and software for the PV system.

2.1 Project Framework

As shown in Fig 7 PV inverter control requires two real time ISR's one is for the closed loop control of the DC-DC sepic stage and one for boost stage. The C2000 Solar Explorer Kit project makes use of the "C-background/C-ISR/ASM-ISR" framework. The ISRs (50kHz), controlling DC-DC boost and sepic stage, run in assembly environment using the Digital Power Library. The PWM base are staggered such that the ISRs do not conflict. The project uses C-code as the main supporting program for the application, and is responsible for all system management tasks, decision making, intelligence, and host interaction.

The key framework C files used in the project are:

SolarExplorer-Main.c – this file is used to initialize, run, and manage the application. A 1 Khz isr is defined in this file for running MPPT and state machine to determine battery operation and led string drive.

SolarExplorer-DevInit_F2803x.c – This file contains all the initialization routines and configuration of IOs and peripherals for this application. This file also includes functions such as setting up the clocks, PLL, Watchdog etc. Most of functions in this file are called once during system initialization from *SolarExplorer-Main.c*.

SolarExplorer-Settings.h – This file contains of setting such as incremental build option and various defines for PWM frequency, ISR triggers, voltage values for different state machine switches that are used in the project framework.

SolarExplorer-Includes.h – This file contains of all the header files used by the project.

SolarExplorer-DPL-ISR.asm – This file contains time critical "control type" code. This file has an initialization section (one time execute) and a run-time section which executes at half the rate (50kHz) as the PWM time-base(50kHz) used to trigger it. This is used for the fast DC-DC boost closed loop control.

Fig 8 gives the structure of the PV inverter software, with the main background loop, the DC-DC ISR and the DC-AC ISR.

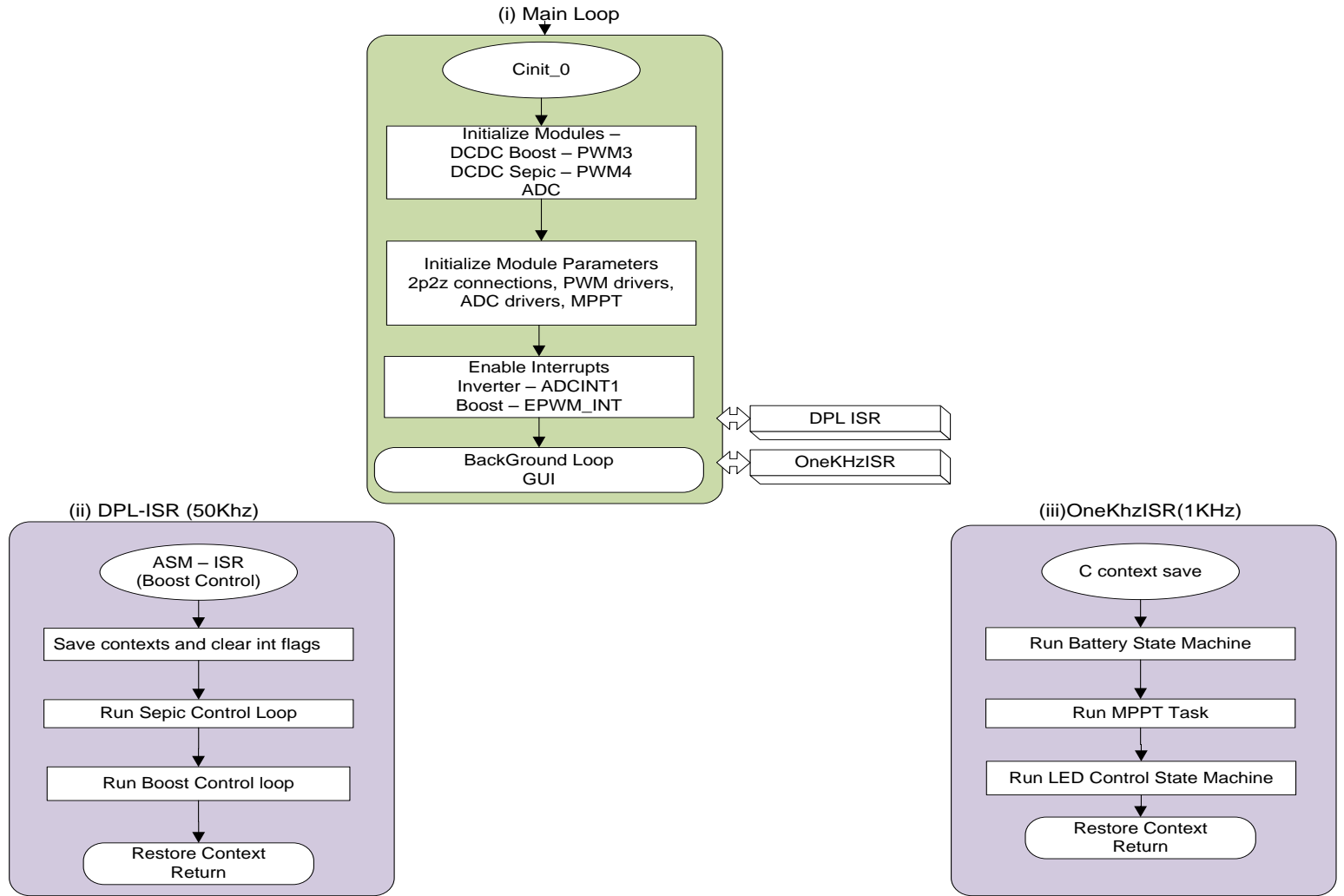


Fig 8 PV Battery Charging and LED driving Software structure (i) Main loop (ii) DPL-ISR (iii) OneKHzISR

2.2 Project Dependencies & Resources

Hardware Kit : TMDSSOLARPEXPKIT [R5]
Control Card : F28035

Software IDE : CCSv5.2.1 or later

Control Suite Dependencies

Device Support (F28035 Header Files) : controlSUITE\device_support\f2803x\v125
IQMath Library : controlSUITE\libs\math\IQmath\v160
Digital Power Library : controlSUITE\app_libs\digital_power\f2803x_v3.4
Solar Library : controlSUITE\app_libs\solar\v1.1\IQ

The guide assumes that the user has already read the following documents related to the kit:

`controlSUITE\development_kits\SolarExplorer\~Docs\SolarExplorer_HWGuide.pdf`

The above documents discuss the kit's hardware features and power stages.

2.3 Control Description

Fig 7 shows the control of PV battery charging system with LED string control. Following sections gives details of the software flow for these two modules.

DC-DC Sepic MPPT Control Software

To get the most energy out of the solar panel, panel needs to be operated at its maximum power point. Maximum power point however is not fixed due to the non linear nature of the PV cell and changes with temperature, light intensity etc. Thus different techniques are used to track maximum power point of the panel like Perturb and Observe, incremental conductance algorithms. These techniques try to track the maximum power point of the panel under given operating conditions and are thus referred to as Maximum Power Point Tracking (MPPT) techniques/algorithms. The Solar Explorer kit has a front-end sepic converter to charge a battery from the solar panel input. To charge a battery first the battery voltage is measured. It is then determined if bulk charging or float charging mode is used. For bulk charging when designing a PV battery system the panel and the battery are selected such that the battery can sink in the max current possible from the PV cell when at MPP.

The control of the stage is described in Fig 5. To track the MPP, input voltage (V_{pnl}) and Input Current (I_{pnl}) are sensed. The MPPT is realized using panel voltage control loop. The reference and feedback values for the panel voltage compensator are flipped, as reducing the panel reference means increasing the load on the panel i.e. more current being drawn i.e greater duty cycle. The MPPT controller is executed at a much slower rate ~ 10Hz.

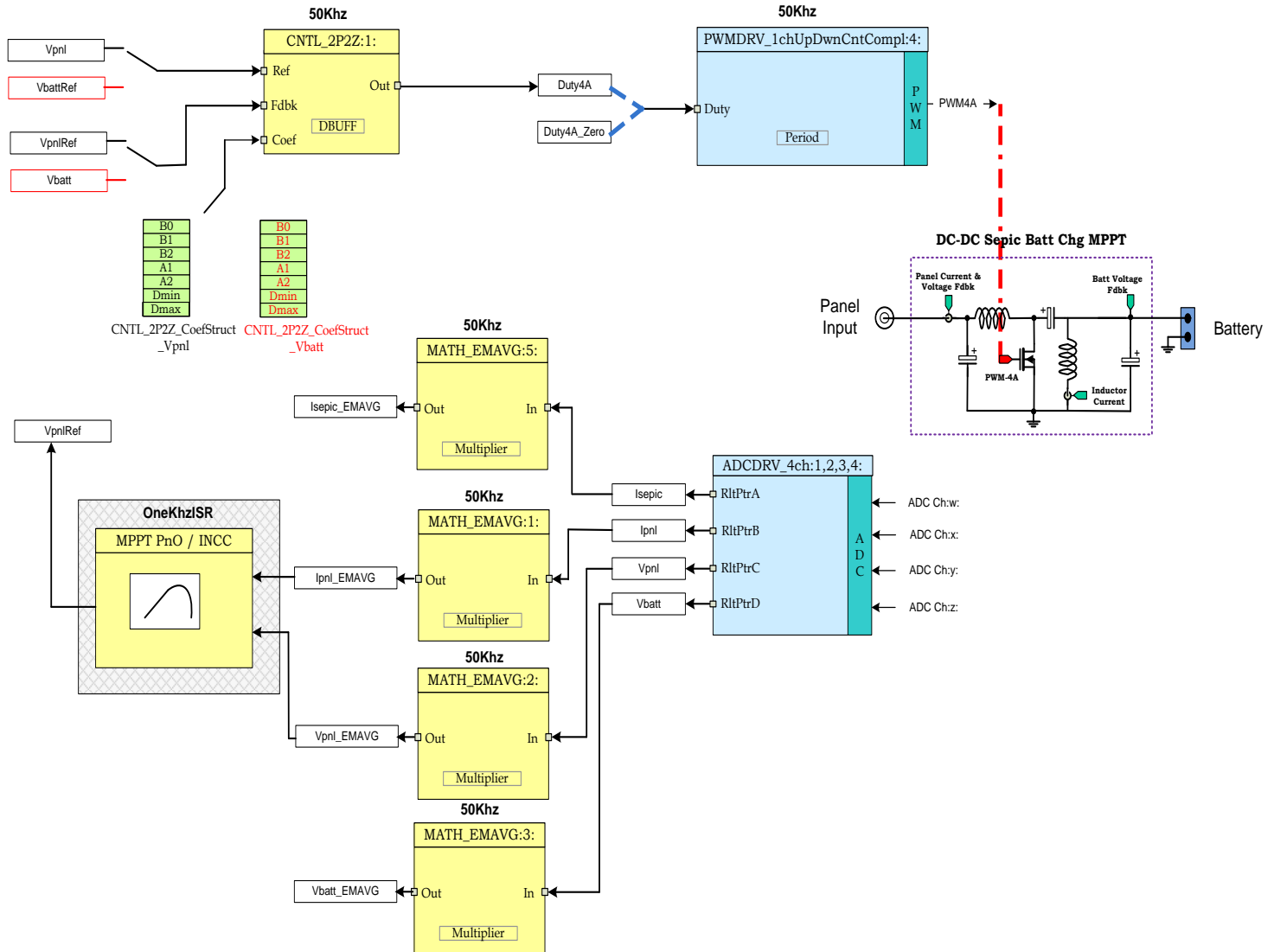


Fig 9 DC-DC Sepic with MPPT Software Diagram

Figure above illustrates the control algorithm. Notice the color coding for the software blocks. The blocks in 'dark blue' represent the hardware modules on the C2000 controller. The blocks in 'blue' are the software drivers for these modules. Blocks in 'yellow' are the controller blocks for the control loop. Although a 2-pole 2-zero controller is used here, the controller could very well be a PI/PID, a 3-pole 3-zero or any other controller that can be suitably implemented for this application. Similarly for MPP tracking, users can choose to use a different algorithm.

The software supports two modes of operations, one is bulk charging when MPPT is enabled and other is float charging when the constant voltage charging is performed. MPPT is disabled in this case. Below is the state machine for the battery state determination.

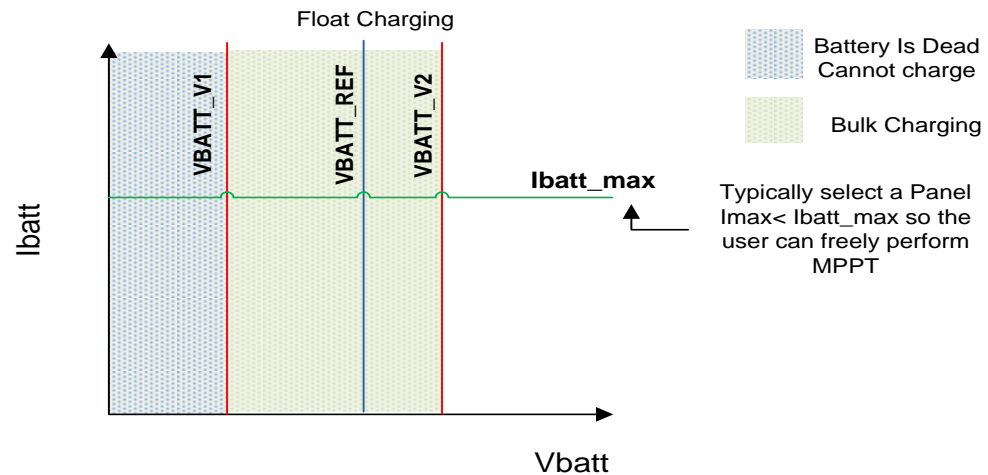
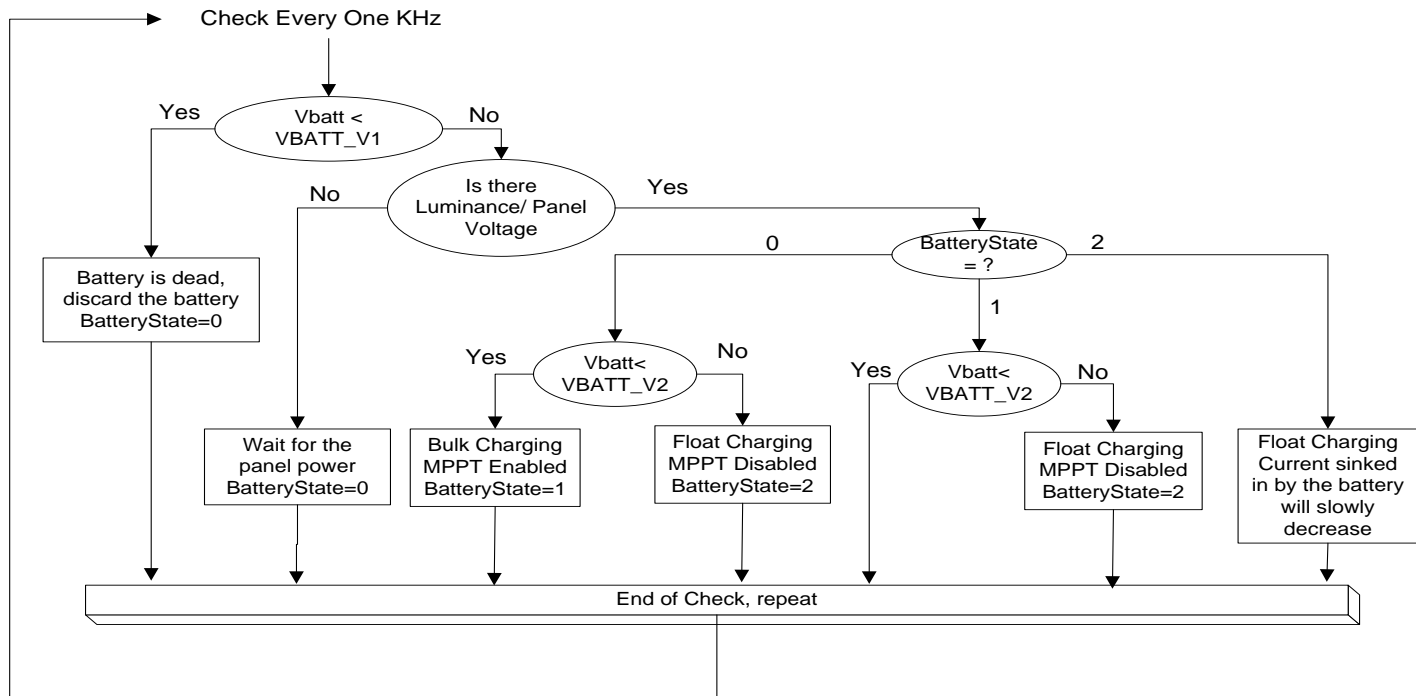


Fig 10 State Machine for Battery State Determination

Code snippet below shows the Input/Output connections between the different blocks used from the Digital Power Library to implement the DC-DC sepic with MPPT control software, this can directly be related to the control diagram above.

```

PWMDRV_1ch_UpDwnCntCompl_Duty4 = &Duty4A;

ADCDRV_4ch_RltPtrA=&Isepic;
ADCDRV_4ch_RltPtrB=&Ipnl;
ADCDRV_4ch_RltPtrC=&Vpnl;
ADCDRV_4ch_RltPtrD=&Vbatt;

MATH_EMAVG_In1=&Ipnl;
  
```

```

MATH_EMAVG_Out1=&Ipn1_EMAVG;
MATH_EMAVG_Multiplier1=_IQ30(0.001);

MATH_EMAVG_In2=&Vpn1;
MATH_EMAVG_Out2=&Vpn1_EMAVG;
MATH_EMAVG_Multiplier2=_IQ30(0.001);

MATH_EMAVG_In3=&Vbatt;
MATH_EMAVG_Out3=&Vbatt_EMAVG;
MATH_EMAVG_Multiplier3=_IQ30(0.001);

MATH_EMAVG_In5=&Isep1c;
MATH_EMAVG_Out5=&Isep1c_EMAVG;
MATH_EMAVG_Multiplier5=_IQ30(0.001);

// For Bulk Charging Mode when MPPT is on
CNTL_2P2Z_Ref1 = &Vpn1;
CNTL_2P2Z_Fdbk1= &Vpn1Ref;
CNTL_2P2Z_Out1 = &Duty4A;
CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct_Vpn1.b2;

// For float charge
CNTL_2P2Z_Ref1 = &VbattRef;
CNTL_2P2Z_Fdbk1= &Vbatt;
CNTL_2P2Z_Out1 = &Duty4A;
CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct_Vbatt.b2;

```

The late of LED2 and LED3 blinking is changed depending upon the state

LED3 Blinking very slowly and LED2 not blinking : State =0

LED3 and LED2 Blinking : State =1

LED3 blinking fast and LED2 not blinking : State=2

DC-DC Boost LED Current Control

The LED string is connected at the output of the boost stage. The average current through the LED string can be controlled as shown below.

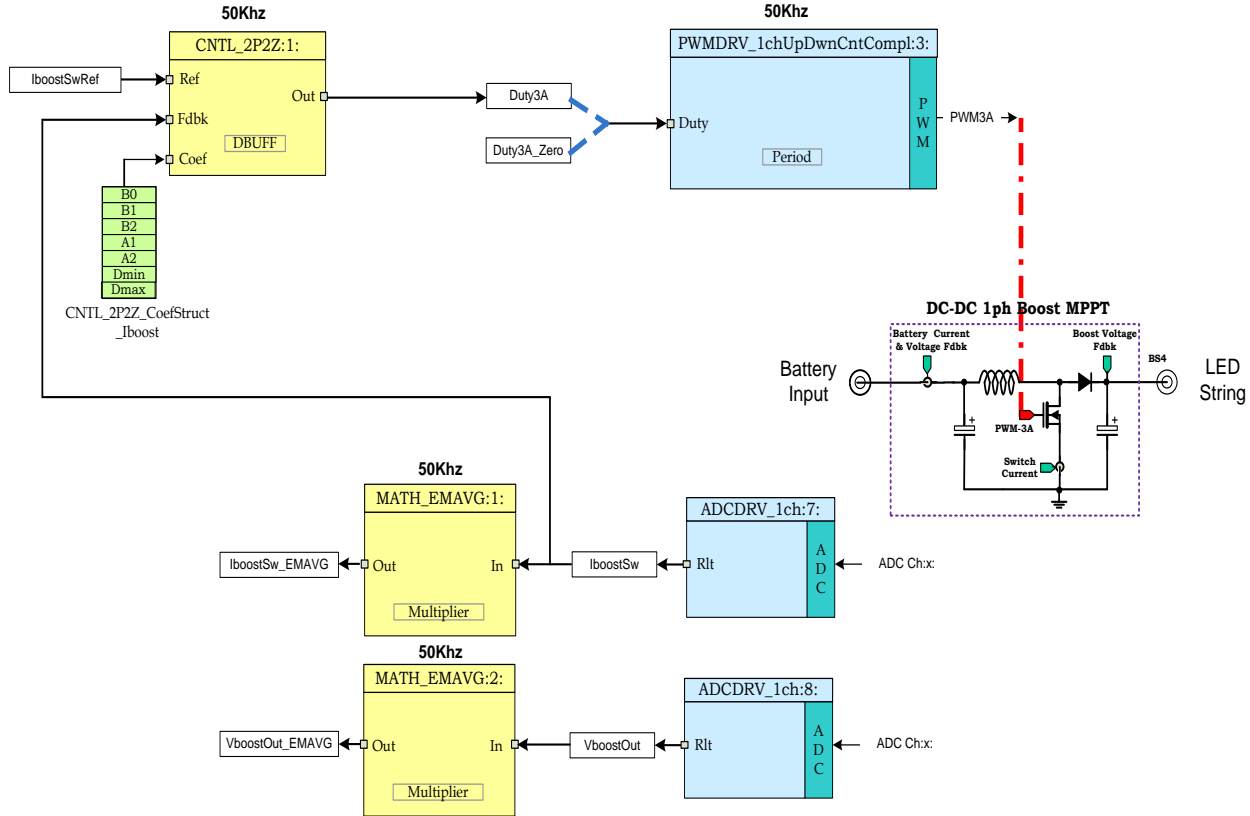


Fig 11 DC-DC Boost LED Current Control

The internal comparator is also used to trip the PWM of the boost stage in case of over voltage to avoid damage to the LED strings. The decision to turn on the LED string is based on the state machine below:

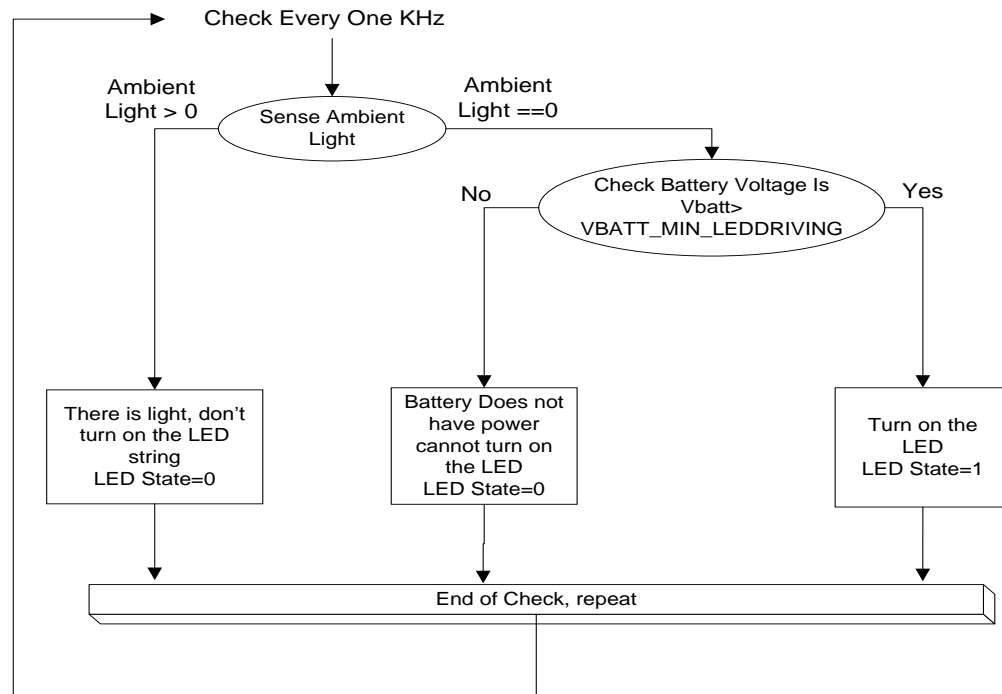


Fig 12 LED State Machine

Code snippet below shows the Input/Output connections between the different blocks used from the Digital Power Library to implement the DC-DC boost current control for LED driving, this can directly be related to the control diagram above.

```

PWMDRV_1ch_UpDwnCntCompl_Duty3 = &Duty3A;

ADCDRV_1ch_Rlt7=&IboostSw;
ADCDRV_1ch_Rlt8=&VboostOut;

MATH_EMAVG_In4=&VboostOut;
MATH_EMAVG_Out4=&VboostOut_EMAVG;
MATH_EMAVG_Multiplier4=_IQ30(0.001);

MATH_EMAVG_In6=&IboostSw;
MATH_EMAVG_Out6=&IboostSw_EMAVG;
MATH_EMAVG_Multiplier6=_IQ30(0.001);

CNTL_2P2Z_Ref2 = &IboostSwRef;
CNTL_2P2Z_Fdbk2= &IboostSw;
CNTL_2P2Z_Out2 = &Duty3A;
CNTL_2P2Z_Coef2 = &CNTL_2P2Z_CoefStruct_Iboost.b2;
  
```

Note that the Ambient Light value used for both the BatteryState and LED State determination can be taken from either the CCS watch window (while debugging) or from the Light sensor on the board. The switch between the two options is done by commenting/ un-commenting out the following section in task A1. Only two levels are used for repeatability when using the light sensor.

```
// To use the light sensor on the board uncomment the lines below
// Otherwise enter LightCommand value in the watch window

if(Gui_LightRatio>_IQ13(0.5))
    Gui_LightCommand=_IQ13(0.2);
else
    Gui_LightCommand=_IQ13(0.0);
```

3 Hardware Platform Setup

3.1 HW Setup Instructions

Note: Do not power up the board before you have verified these settings!

Before starting the labs the user must make sure the following settings are correct.

- 1) Make sure nothing is connected to the board, and no power is being supplied to the board.
- 2) Insert the controlCARD into the [Main]-J1 controlCARD connector if it is not already installed.
- 3) Do the following switch settings on the controlCARD:
 - a. Control Card SW1 is in the OFF position
 - b. Control Card SW2, Position 1 = ON, Position 2 = ON

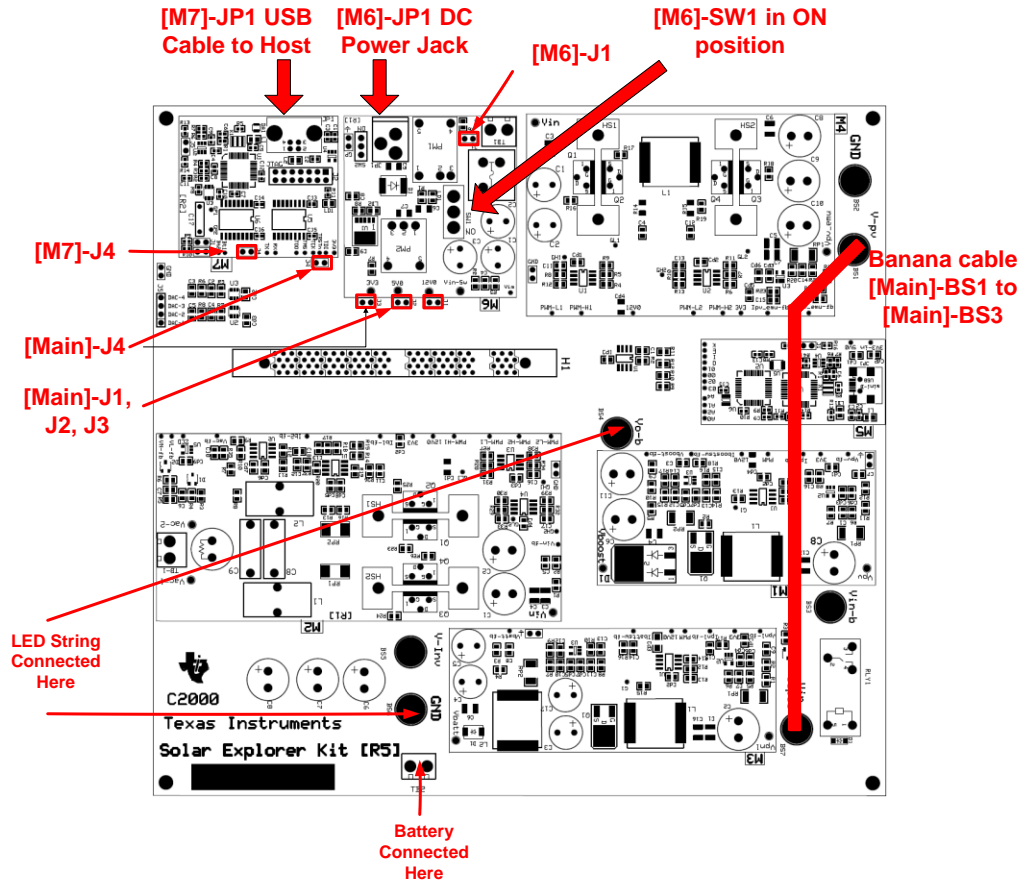


Fig 13 Hardware setting Solar Explorer Kit

- 4) The guide assumes that the TMS320F28027 microcontroller present in the M4 macro is pre-flashed with the panel emulator code with which the kit is shipped.
- 5) Connect a banana cable between [Main]-BS1 and [Main]-BS3

- 6) Verify that a LED string is connected at the output of the boost stage i.e. between [Main]-BS4 and [Main]-BS6
- 7) Make sure the [Main]-J1, [Main]-J2, [Main]-J3, [Main]-J4, [M6]-J1 and [M7]-J4 jumpers are populated. Verify [M6]-SW1 is in on position.
- 8) Connect a USB cable (B to A Cable) from [M7]-JP1 to the host computer. [M7]-LD1 will light up indicating that the USB is powered.
- 9) Now connect the DC power supply shipped with the kit to [M6]-JP1, and turn on [M6]-SW2. [M5]-LD2 will start blinking indicating the PV emulator code is running on the emulator. Turn off [M6]-SW2.

3.2 Software Setup

Installing Code Composer and controlSUITE

1. If not already installed, please install Code Composer v5.x
2. Go to <http://www.ti.com/controlsuite> and run the controlSUITE installer. Select to install the “SolarExplorer” software and allow the installer to also download all automatically checked software.

Setup Code Composer Studio to Work with SolarExplorer kit

3. Open “Code Composer Studio”.
4. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location,: (please note workspace is a location on the hard drive where all the user settings for the IDE i.e. which projects are open, what configuration is selected etc. are saved, this can be anywhere on the disk, the location mentioned below is just for reference. Also note that if this is not your first-time running Code Composer this dialog may not appear)
 - Click the “Browse...” button
 - Create the path below by making new folders as necessary.
 - “C:\Documents and Settings\My Documents \ProjectWorkspace”
 - Uncheck the box that says “Use this as the default and do not ask again”.
 - Click “OK”.
5. Add the project into your current workspace by clicking “Project->Import Existing CCS/CCE Eclipse Project”.
 - Select the root directory of the SolarExplorer. This will be:
“\controlSUITE\development_kits\SolarExplorer_vx.x\SolarExplorer_PVBattChg_F2803x”
 - Click Finish, this would copy all the projects relevant for the kit into the workspace. If you want only a particular project to be copied uncheck the box next to the other project names.

Configuring a Project

6. Expand the file structure of the project you would like to run from the C/C++ Projects tab. Right-click on this project’s name and select “Set as Active Project”, if this is not already the case.

7. Assuming this is your first time using Code Composer, the xds100-F28035 should have been set as the default target configuration. Do verify this by viewing the xds100-f28035.ccxml file in the expanded project structure and a [Active/Default] written next to it. By going to “View-> Target Configurations” you may edit existing target configurations or change the default or active configuration. You can also link a target configuration to a project in the workspace by right clicking on the Target configuration name and selecting Link to Project.
8. Fig 14 shows the project in the CCSv4 C/C++ Project tab, it shows all the key files used in the project.

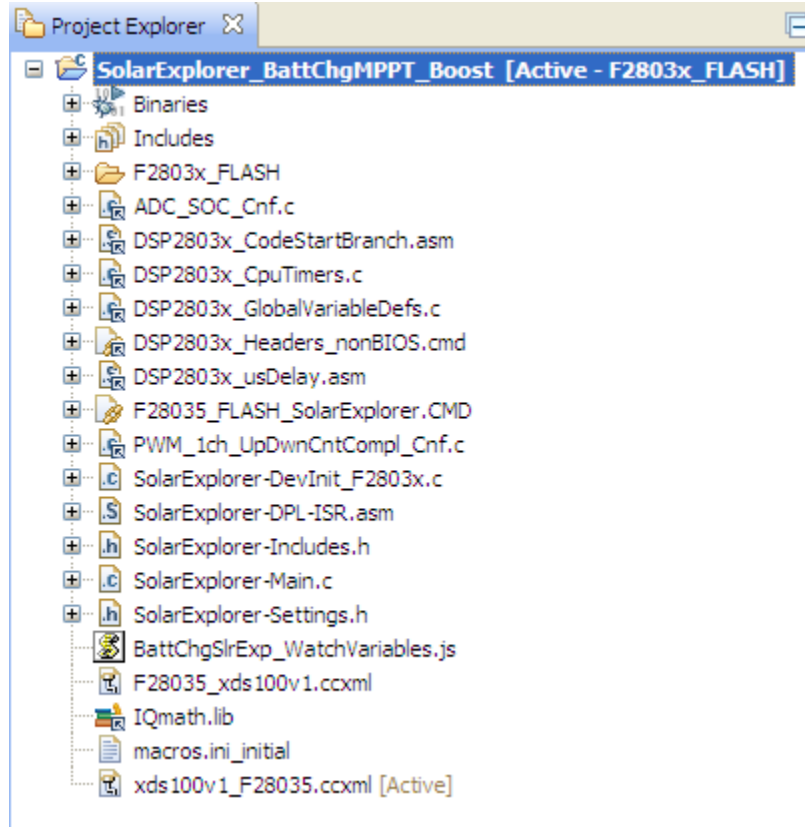


Fig 14 PV Inverter project in C/C++ tab

4 PV Battery Charging and LED Driving Project

The project simply run the control code described above. First, uncomment the following lines in the task A1 to enter ambient light value through the watch window.

```
// To use the light sensor on the board uncomment the lines below
// Otherwise enter LightCommand value in the watch window

/*if(Gui_LightRatio>_IQ13(0.5))
    Gui_LightCommand=_IQ13(0.2);
else
    Gui_LightCommand=_IQ13(0.0);
*/
```

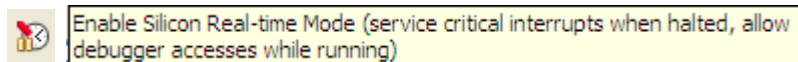
Build and Load the Project


1. Click Project → “Rebuild All” button and watch the tools run in the build window.
2. Click Target → “Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().
3. Now add variables to the expression view by first opening the scripting console through View->Scripting Console. Then on the scripting console window, select “open command file” option at the top right corner and select the “BattChrgSlrExp_WatchVariables.js” from the project folder. This will populate the Expression Window with appropriate variables needed to debug the system and the appropriate Q formats.

Using Real-time Emulation


Real-time emulation is a special emulation feature that allows the windows within Code Composer Studio to be updated at a rate up to 10 Hz *while the MCU is running*. This not only allows graphs and watch views to update, but also allows the user to change values in watch or memory windows, and see the effect of these changes in the system. This is very useful when tuning control law parameters on-the-fly.

4. Enable real-time mode by hovering your mouse on the buttons on the horizontal toolbar and clicking button






5. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
6. Click on Continuous Refresh buttons  for the watch view.

Run the Code

7. Run the project, 
8. Run using the Run button on the toolbar, or using Target-> Run on the menu bar. The expressions view will keep on refreshing with values as shown below.

Expression	Type	Value	Address
(x)= BatteryState	int	0	0x0000800E@Data
(x)= LEDState	int	0	0x0000800F@Data
(x)= BattConnectToBoost	int	0	0x00008010@Data
(x)= Gui_LightCommand	long	0.0 (Q-Value(13))	0x0000804E@Data
(x)= Gui_Vpnl	long	0.193359375 (Q-Value(9))	0x00008054@Data
(x)= Gui_Ipnl	long	0.03491210938 (Q-Value(12))	0x00008052@Data
(x)= Gui_Vbatt	long	0.115234375 (Q-Value(10))	0x0000805C@Data
(x)= Gui_Vboost	long	1.24609375 (Q-Value(9))	0x00008056@Data
(x)= Gui_Vboost_in	long	1.212890625 (Q-Value(9))	0x00008058@Data
(x)= Gui_PanelPower	long	0.0 (Q-Value(9))	0x0000804A@Data
(x)= Gui_PanelPower_Theoretical	long	0.0 (Q-Value(9))	0x00008050@Data
(x)= VpnlRef	long	0.8999999762 (Q-Value(24))	0x00008044@Data
(x)= Vpnl	long	0.00634765625 (Q-Value(24))	0x00008040@Data
(x)= Duty4A	long	0.0 (Q-Value(24))	0x0000803A@Data
(x)= IboostSwRef	long	0.0 (Q-Value(24))	0x00008026@Data
(x)= IboostSw	long	0.02172851563 (Q-Value(24))	0x00008022@Data
(x)= Duty3A	long	0.0 (Q-Value(24))	0x00008020@Data
+ Add new expression			

Fig 15 Expressions window to watch variables

9. Now, as the battery has not been connected and the LightCommand is zero, nothing will happen.
10. Connect a 12V battery that can take a max of 2 Amps current at the output of the sepic stage on the terminal block [Main]-TB2. You can see the battery voltage value now reflected in the watch view. As Light command is zero, the LED state machine will detect this and start the LED driving and you can see the LED string light up. (You must select a LED string capable of handling ~30V at the terminal.)
11. Now change the LightCommand Value to 0.2 and see that the battery state machine will kick in and start charging the battery. The LED string state machine will detect the light and switch off the LEDs.
12. Depending on what level of battery charge is there the battery can use MPPT or not. User can play with different light command value, and vary other parameters to see the effect and can modify the state machine to suit their needs.
13. Write 0 to LightCommand Value. Now, halt the processor by using the Halt button on the toolbar , or by using Target → Halt. Then take the MCU out of real-time mode by clicking on . Finally reset the MCU . You can not disconnect the battery.

End of Exercise