TDINV1000P100: 1kW Inverter Evaluation Board

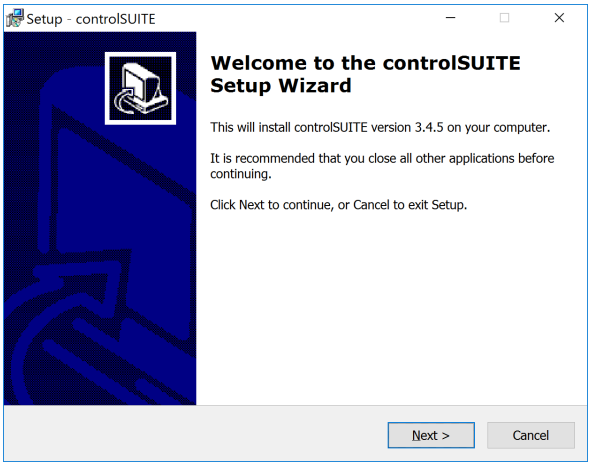
Part 1: Getting started with the code

The design files and source code for the TDINV1000P100 1kW inverter can be found at transphormusa.com/pv1kit and transphormusa.com/pv1fw, respectively, and includes the complete project. The following instructions can be used to load and build the project on your computer for inspection or modification of the code. To take maximum advantage of the existing firmware, the TDINV1000P100 firmware is based on an open-source design from Texas Instruments (TI): The HV_SOLAR_DC_AC_v1.1 Kit and included in the firmware files. Full documentation of the TI kit and firmware is available from TI as part of the controlSUITE package.

Installing Code Composer and controlSUITE

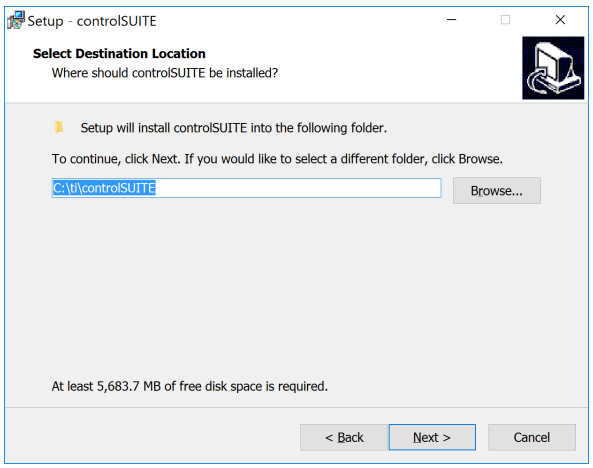
Code Composer Studio v7 is an integrated development environment based on the Eclipse open source framework and is available at processors.wiki.ti.com/index.php/Download_CCS.

1. Install Texas Instruments Code Composer v7.x
2. The controlSUITE package is not required to work with the firmware but a great deal of documentation is provided with it; to download, go to ti.com/tool/controlsuite and run the controlSUITE installer, following these steps:

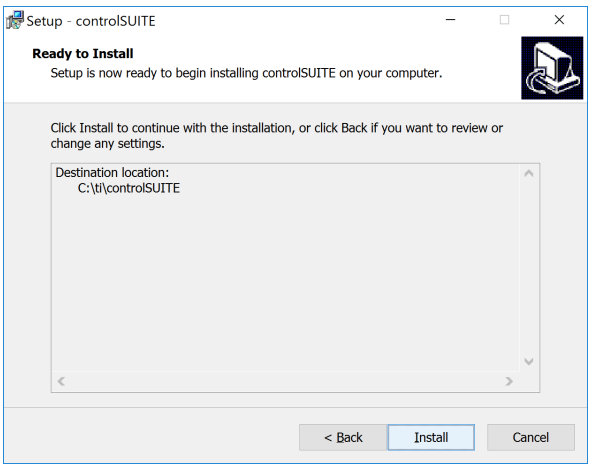


Step 1

Step 2



Step 3



Step 4

3. Once controlSUITE is installed, Code Composer can be set up to work with the TDINV1000P100 source code

Setting up Code Composer Studio to work with TDINV1000P100-KIT

4. Create a folder to use as a workspace, i.e., C:\transphorm\tdinv1000p100
5. Launch Code Composer Studio and select the desired workspace
6. Go to Project in the menu and select Import Existing CSS Eclipse Project
   a. Navigate to the firmware files, then locate SolarHV_DCAC
   b. Click Finish
7. Go to Project in the menu and select CLEAN which will perform a full build of the project
   a. After finishing the build, the files will appear under Project Explorer. The TMS320F28035 will load as the target processor—to verify, click Target Configurations under the View menu item, and then select the SolarHv_DCAC.ccxml under Projects.

Load and run the code

8. Make sure jumper JP1 is in place to allow communication over the JTAG port. Once the code is written to flash, JP1 should be removed to allow booting from flash.
   a. Connect a USB cable from the computer to the TDINV1000P100 inverter board (CN2)
   b. Connect 9V_{CC} power adaptor to the inverter (J1)
9. After completion of the build, click the Debug button, located on the top-left side of screen. The IDE will now automatically connect to the target, load the output file into the device and change to the Debug perspective
10. Run the code by pressing the RUN button in the Debug tab

Part 2: Working with the code

The Open-Loop build option is the only option currently proven and supported with this board. The alternate build options in the TI code (Closed Loop without PLL and Closed Loop with PLL) should be compatible with this board, but have not been verified.

The original code is named HV_SOLAR_DC_AC_v1.1, and can be imported from:
\controlSUITE\development_kits\HV_SOLAR_DC_AC_v1.1\SolarHv_DCAC_PiccoloB_Rev_02

The changes made to the HV_SOLAR_DC_AC_v1.1 project to support 100kHz switching are the following (line numbers given correspond to the changed code).

1. SolarHv_DCAC-Settings.h - various #defines

2. SolarHv_DCAC-main.c ln 803-805 - Used the #defines for PWM PRD

3. SolarHv_DCAC-main.c ln 963 - Used #define for sample time

4. SolarHv_DCAC-main.c ln 990 - Used #define for wSinAmp

5. SolarHv_DCAC-main.c ln 1193 – Used comment to disable eCap interrupt for zero crossing detection

6. SolarHv_DCAC-main.c ln 819 - Disabled synch of PWM3 to PWM1

7. SolarHv_DCAC-main.c ln 830 - ADC conversions start only when PWM3 counter reaches zero

8. SolarHv_DCAC-ISR.asm lines 113, 114 – use EPwm3Regs instead of EPwm1Regs

Changing the switching frequency

The #define statements in SolarHv_DCAC-Settings.h (lines 8 -48, listed below) are used to set the switching frequency. Five switching frequencies are provided, between 50kHz and 300kHz. There are four frequency-dependent parameters that need to be adjusted: the period, the decimation count, the sample time, and the amplitude control. Other frequencies can be generated following the same model. The decimation count is used to keep the loop update rate at 20kHz, regardless of switching frequency. The update rate can also be changed using the same parameters, but be careful to ensure that all computations can be completed in the time desired.

```
#define TRANSPHORM_TDINVXXXX    1
#if TRANSPHORM_TDINVXXXX

/* Uncomment only one of the following lines */

//#define TRANSPHORM_SWITCHING_FREQUENCY_50KHZ  1   // 50kHz switching frequency
#define TRANSPHORM_SWITCHING_FREQUENCY_100KHZ   1   // 100kHz switching frequency
//#define TRANSPHORM_SWITCHING_FREQUENCY_150KHZ 1   // 150kHz switching frequency
//#define TRANSPHORM_SWITCHING_FREQUENCY_200KHZ 1   // 200kHz switching frequency
//#define TRANSPHORM_SWITCHING_FREQUENCY_300KHZ 1   // 300kHz switching frequency

/* Parameters specific to switching frequency */

#if TRANSPHORM_SWITCHING_FREQUENCY_50KHZ
#define TRANSPHORM_PWM_PRD                  1200
#define TRANSPHORM_LOOP_DECIMATION_COUNT    3       // Perform ADC conversions and run control loop every 50/3 = __kHz
#define TRANSPHORM_SAMPLE_TIME              (0.00002 * TRANSPHORM_LOOP_DECIMATION_COUNT)        // 10e-6 * 5
#define TRANSPHORM_SIN_AMP                  0.156//-regular //(0.1092-> for furu...)
#elif TRANSPHORM_SWITCHING_FREQUENCY_100KHZ
#define TRANSPHORM_PWM_PRD                  600
#define TRANSPHORM_LOOP_DECIMATION_COUNT    5       // Perform ADC conversions and run control loop every 100/5 = 20kHz
#define TRANSPHORM_SAMPLE_TIME              (0.00001 * TRANSPHORM_LOOP_DECIMATION_COUNT)        // 10e-6 *
#define TRANSPHORM_SIN_AMP                  0.078 //for 340Vin - 240Vout
#elif TRANSPHORM_SWITCHING_FREQUENCY_150KHZ
#define TRANSPHORM_PWM_PRD                  400
#define TRANSPHORM_LOOP_DECIMATION_COUNT    7       // Perform ADC conversions and run control loop every 150/7 = __kHz
#define TRANSPHORM_SAMPLE_TIME              (0.0000075 * TRANSPHORM_LOOP_DECIMATION_COUNT)      // 5e-6 *
#define TRANSPHORM_SIN_AMP                  0.051   //0.051 for 340Vin - 240Vout      //0.0455 for 400Vin - 240Vout
#elif TRANSPHORM_SWITCHING_FREQUENCY_200KHZ
#define TRANSPHORM_PWM_PRD                  300
#define TRANSPHORM_LOOP_DECIMATION_COUNT    10      // Perform ADC conversions and run control loop every 200/10 = 20kHz
#define TRANSPHORM_SAMPLE_TIME              (0.000005 * TRANSPHORM_LOOP_DECIMATION_COUNT)       // 5e-6 * 10
#define TRANSPHORM_SIN_AMP                  0.039  //for 340Vin - 240Vout
#elif TRANSPHORM_SWITCHING_FREQUENCY_300KHZ
#define TRANSPHORM_PWM_PRD                  200
#define TRANSPHORM_LOOP_DECIMATION_COUNT    15      // Perform ADC conversions and run control loop every 300/15 = 20kHz
#define TRANSPHORM_SAMPLE_TIME              (0.000005 * TRANSPHORM_LOOP_DECIMATION_COUNT)       // 5e-6 * 10
#define TRANSPHORM_SIN_AMP                  0.0255 //0.0255 for 340Vin - 240Vout


#endif
```

Changing the output frequency

For open-loop operation, the output frequency is controlled by parameter dwPLL_Trace_Freq. Lines 958-961 in SolarHv_DCAC-main.c allow simple selection of 50 or 60Hz:

```
    /* Uncomment one value below for dwPll_Trace_Freq depending on desired grid frequency */
    dwPll_Trace_Freq = _IQ20(376.9911);                 //60Hz
    //dwPll_Trace_Freq = _IQ20(314.1593);                   //50Hz

```

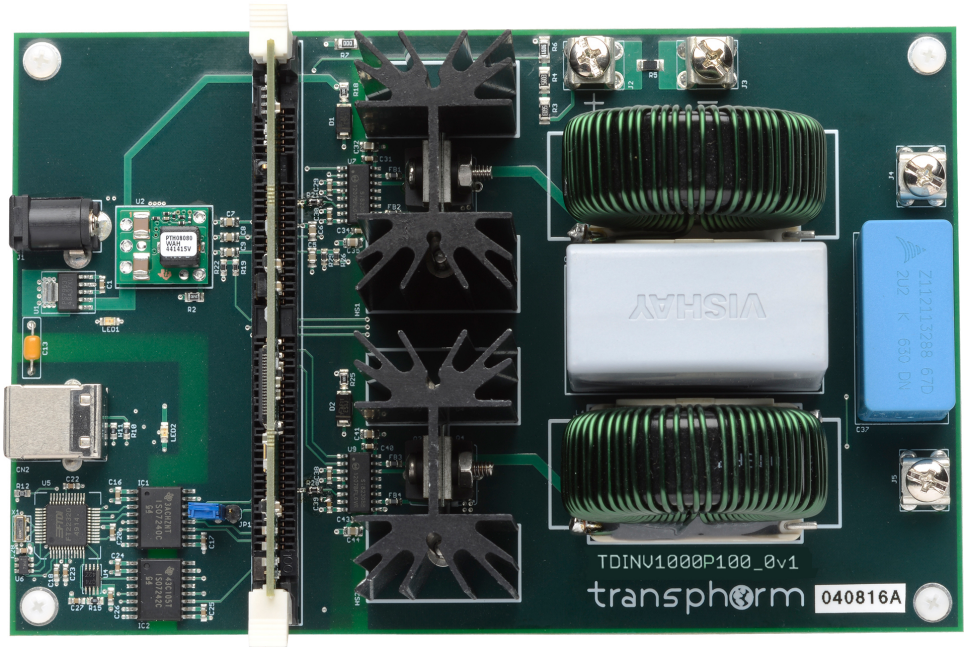


Figure 1. TDINV1000P100 1kW inverter evaluation board