PWM Demonstration System Document

Texas Instruments

Table of contents

Table of Figures

Index of Tables

1 System Overview

This system implements a set of demonstrations to demonstrate the flexibility of the PWM capabilities of the TMS320F280x DSP controllers. This demonstration offers workspaces for Code Composer Studio 2.21, which are used to simplify the process. This demonstration uses the following modules from the Digital Power Solutions Software Library:

- BUCK_DRV
- PSFB_DRV
- MPIL_DRV

This demonstration system uses an incremental build approach to build relevant parts of the system. Overall, this system consists of a 'F280x EZDSP and a scope, as is shown in Figure 1. No power hardware is needed to demonstrate PWM waveforms.

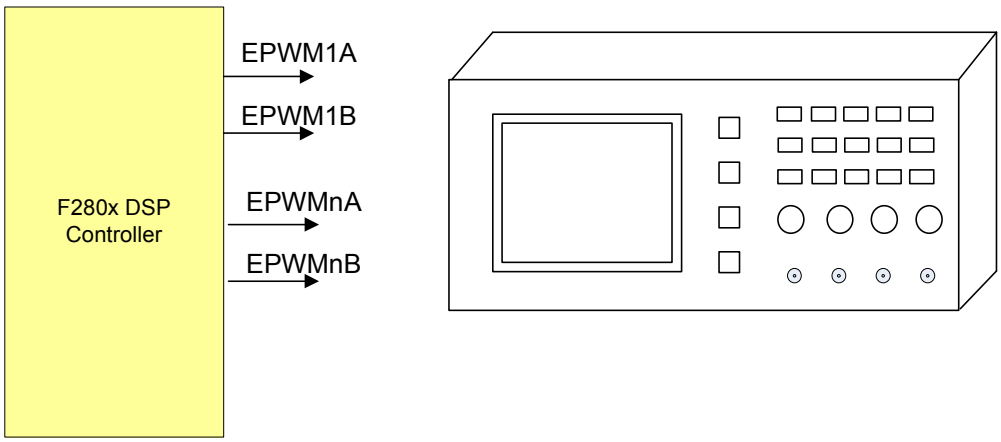

Figure 1. Single channel sine wave generator usage

Development/Emulation	Code Composer Studio V.2.21 (or above) with Real Time debugging
Target Controller	Spectrum Digital – TMS320F2808 EZDSP

2 Software structure

2.1 Directory structure

The directory structure adopted for the digital power solutions software library is shown in Figure 2. The files are separated into library and system implementations. The V100 directory is the directory named for the version of the software release. A release ‘1.00’ is released in the directory named ‘v100’. The software versioned 2.00 is released in the v200 directory. A (future) release version 2.10 would be placed in the directory ‘v210’.

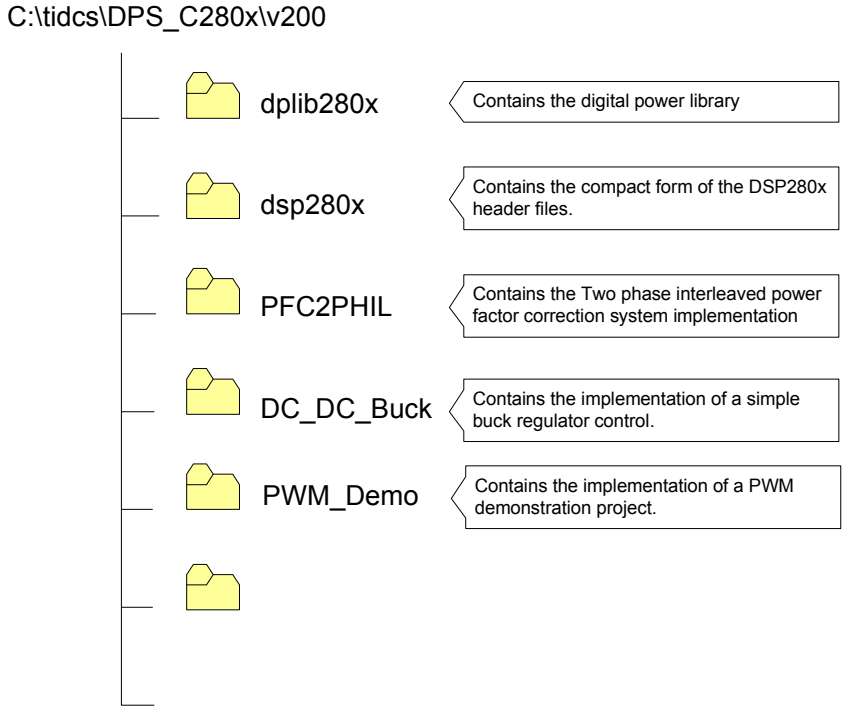

Figure 2. Directory structure used in the software release

The structure above is a significant simplification and condensation from the structure employed in previous releases. Secondly the files used from the DSP280x release have been retooled for this specific application space, mainly by compacting them in to fewer files.

2.2 Software Flowchart

The software follows the general flow depicted in the Figure 3. The software may vary a little bit from the exact flow below, depending on the configuration options specified.

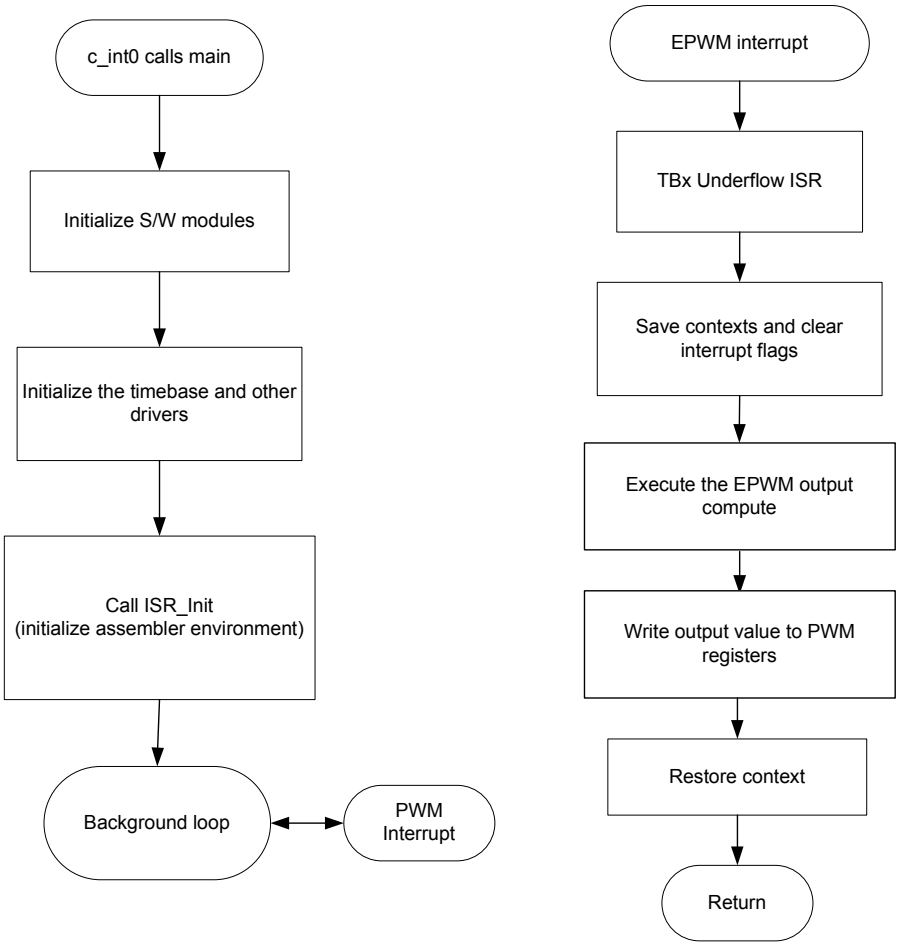

Figure 3. Software flowchart

2.3 Software configuration options

The software offers several configuration options to specify which of the EPWM time bases is used as an interrupt source, or whether the ADC interrupt is used to trigger processing.The code section below shows the code mechanism behind the interrupt framework.

```
//=====================================================================
// Interrupt Framework options
// Note: make sure same conditional assembly option is chosen in
// file PWM-Eval-ISR.asm
//---------------------------------------------------------------------
#define     EPWMn_ISR         1     // ISR triggered by EPWM
#define     ADC_ISR           0     // ISR triggered by ADC EOS
// If EPWM_ISR = 1, then choose which module
#define     EPWM1_triggers_ISR      0     // ISR triggered by EPWM1
#define     EPWM2_triggers_ISR      0     // ISR triggered by EPWM2
#define     EPWM3_triggers_ISR      0     // ISR triggered by EPWM3
#define     EPWM4_triggers_ISR      0     // ISR triggered by EPWM4
#define     EPWM5_triggers_ISR      0     // ISR triggered by EPWM5
#define     EPWM6_triggers_ISR      1     // ISR triggered by EPWM6
//---------------------------------------------------------------------
```

The mechanism depends on the code selecting only one of the choices indicated. So, for instance to use EPWM6 as an ISR trigger, the time base in EPWM unit 6 is initialized, and the framework option `EPWM6_triggers_ISR` is selected, by setting it to a 1. All other options must be set to 0. Any other options set to 1 will cause undefined operation.

Incremental build options for examples in the PWM demonstration	
IB1A	PWM driver example.
IB1B	PWM driver – two module example.
IB2A	Multiphase interleaved PWM – 2 phase example
IB2B	Multiphase interleaved PWM – 3 phase example
IB2C	Multiphase interleaved PWM – 4 phase example
IB5	Phase shifted PWM for full bridge drive.

Table 1. Incremental build options in the PWM Example

The incremental build system is used in this case to run separate examples. In the PWM_DEMO, this is used to enable six examples, as indicated in the Table 1 above.

3 Procedures for running the examples.

3.1 Building and Loading Build 1A

The workspace file (*.wks) and project file (*.pjt) for C framework to demonstrate the PWM demo are located in the ***C:\tidcs\DPS_C280x\vxyz\PWM_Demo*** directory[†]. The CCS workspace file, contains the setup information for the whole project and the debugging environment such us the graph window properties, watch window parameters, break points and probe points etc. It facilitates the user to save and restore the same environment between debugging sessions instead of reconfiguring the working environment again and again for each debugging session. [Notice that occasionally the spectrum digital driver is named differently from the default, "**sdgo2808EZDSP**". If so CCS issues a warning, but loads the workspace file successfully.]

Build 1A demonstrates the PWM capabilities of the EPWM units.

Follow the steps below to build and run the examples included in the PWM_Demo workspace.

1. To quickly execute demo using the pre-configured work environment, load the workspace file **PWM_Demo.wks** from ***C:\tidcs\DPS_C280x\vxyz\PWM_Demo*** directory.

2. Loading the workspace file will automatically open up the project file (*.pjt) for the corresponding project and show all the files relevant to the project in the FILEVIEW tab.

[†] The xyz represents the version number directory level. For instance, a 1.00 release would have v100 in its directory path, and v210 would indicate a release 2.10.

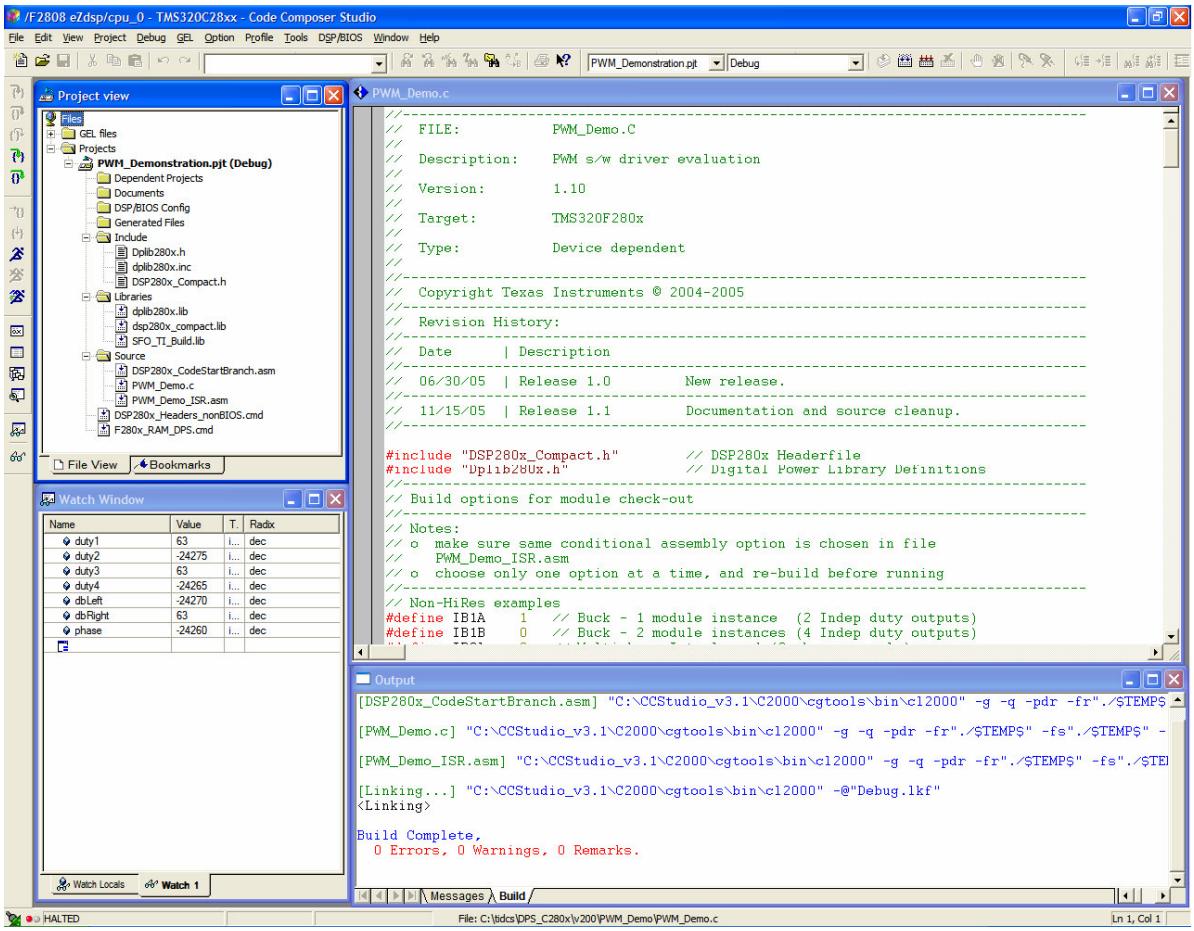

Figure 4. CCS workspace screen capture from the PWM_Demo example.

3. Open the **PWM_Demo.c** file. Set the example build to be built to Build 1A. To do this, put a '1' for the IB1A, and a '0' for all the other options in that section, near line 35. Save the file.‡

4. Open the **PWM_Demo_Isr.ASM** file. Set the example build to be built to Build 1A. To do this, put a '1' for the IB1A, and a '0' for all the other options in that section, near line 35. Save the file.

5. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

6. Once this is done, the expanded project view as part of the CCS environment will be as shown in Figure 4, **PWM_Demo.wks**.

7. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

‡ Note: After making modifications to files, it is necessary to save them, to do so hit the save icon on the CCS toolbar. Alternately, it is possible to configure CCS to automatically save files, see the Options/Editor Properties/General Tab/File Loading Group. Make sure the 'Auto-save files before build' checkbox is checked.

8. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

> **Note: This assumes that your F2808 EZDSP is set to boot from H0RAM, at address 0x000000, where this program example places the entry point. If your EZDSP setup needs changing, see the Spectrum Digital™ user guide for the EZDSP for instructions.**

9. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

10. Inspect the PWM output on EPWM1A (P8, pin 9) and EPWM1B (P8, pin 10), on the EZDSP connector P8. Vary the duty cycle by modifying the value of the variable 'duty1' and 'duty2'. A screen capture of the PWM waveforms is shown in Figure 5

11. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.

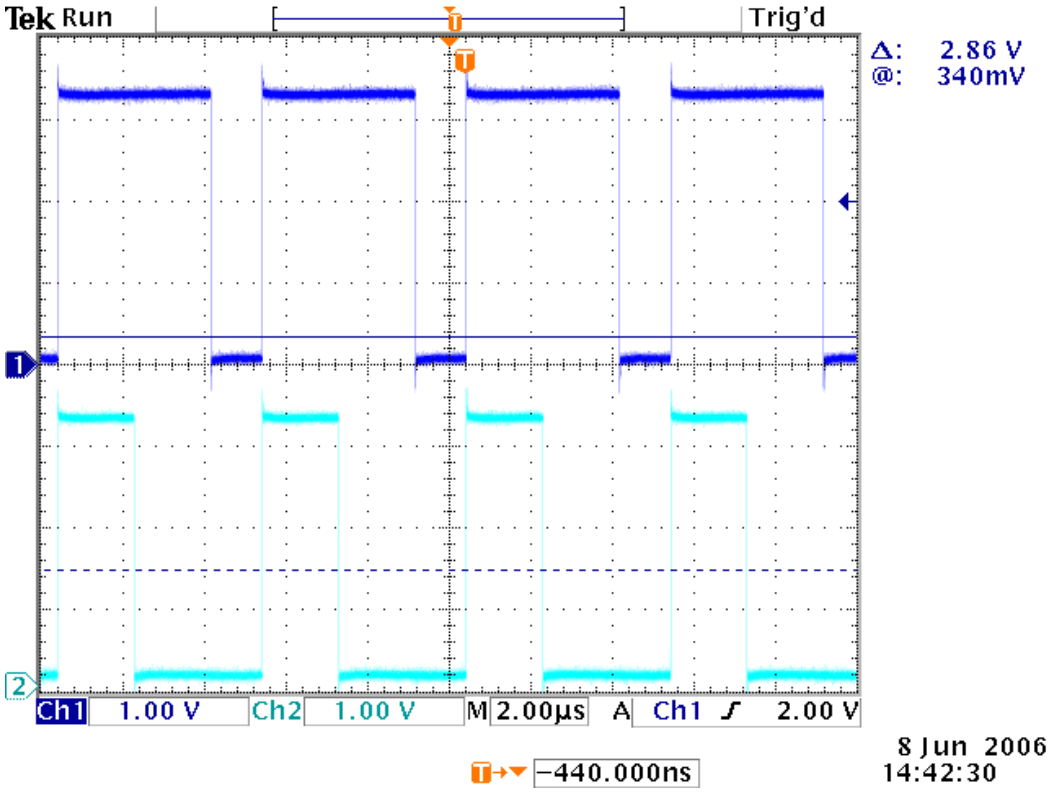

Figure 5. PWM waveform from Build1A on EPWM1A.

3.2 Building and Loading Build 1B

Build 1B demonstrates the multiple instance capabilities of the BUCK_DRV which drives the EPWM units.

(This build assumes continuing progress from the build 1A.)

1. Open the **PWM_Demo.c** file. Set the example build to be built to Build 1B. To do this, put a '1' for the IB1B, and a '0' for all the other options in that section, near line 35.

2. Open the PWM_Demo_Isr.ASM file. Set the example build to be built to Build 1B. To do this, put a '1' for the IB1B, and a '0' for all the other options in that section, near line 35.

3. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

4. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

5. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

6. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

7. Inspect the PWM output on EPWM1A (P8, pin 9) and EPWM1B (P8, pin 10) and EPWM2A (P8, pin 11) and EPWM2B (P8, pin 12) on the EZDSP connector P8. Vary the duty cycle by modifying the value of the variables duty1 – duty4. A screen capture of the PWM waveform is shown in Figure 7, with both set to a value of 0x6000. Notice that the modules are started independently – master / slave start (although available) is NOT used, thus there is some phase offset.

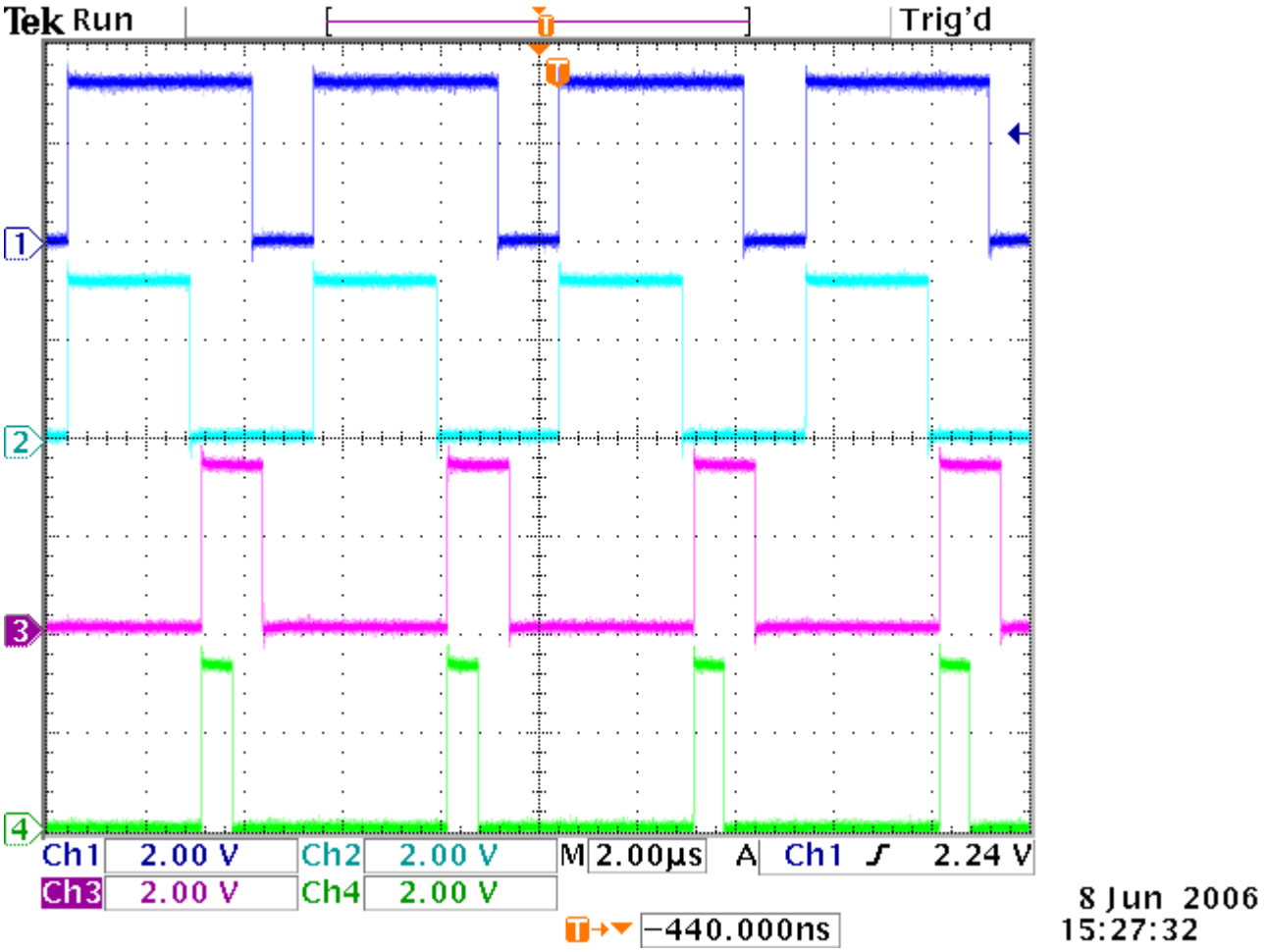

Figure 6. PWM waveform from Build1B on EPWM1A, 1B and EPWM2A, 2B.

8. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.

3.3 Building and Loading Build 2A

Build 2A demonstrates the capabilities of the EPWM units to generate multiphase waveforms with ease. Two waveforms out of phase waveforms are generated.

(This build assumes continuing progress from the build 1B.)

1. Open the **PWM_Demo.c** file. Set the example build to be built to Build 2A. To do this, put a '1' for the IB2A, and a '0' for all the other options in that section, near line 35.

2. Open the PWM_Demo_Isr.ASM file. Set the example build to be built to Build 2A. To do this, put a '1' for the IB2A, and a '0' for all the other options in that section, near line 35.

3. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

4. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

5. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

6. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

7. Inspect the PWM output on EPWM1A and EPWM2A on the EZDSP. Vary the duty cycle by modifying the value of the variable 'duty1'. (The waveforms below were captured with the duty1 variable set to 0x2000.)

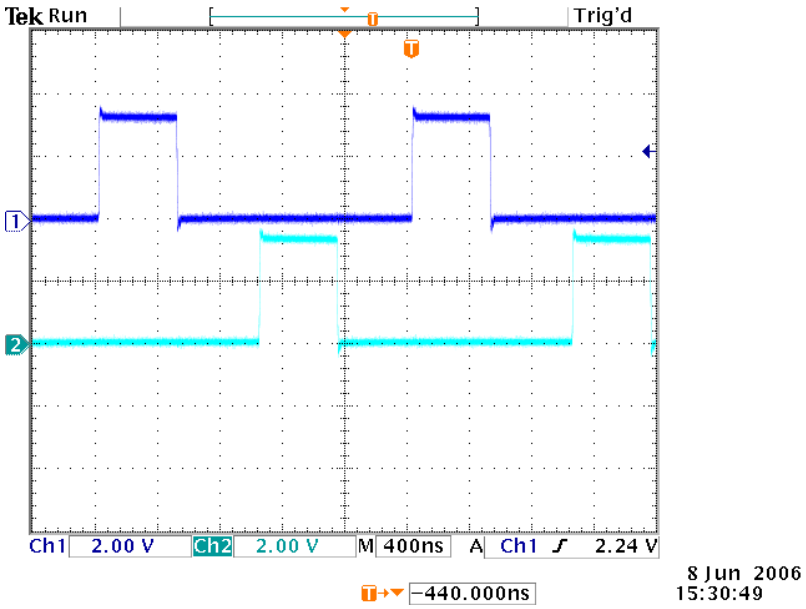

Figure 7. PWM waveform from Build2A on EPWM1A and EPWM2A.

8. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.

3.4 Building and Loading Build 2B

Build 2B demonstrates the capabilities of the EPWM units to generate multiphase waveforms with ease. Three waveforms out of phase waveforms are generated. (This build assumes continuing progress from the build 2A.)

1. Open the **PWM_Demo.c** file. Set the example build to be built to Build 2B. To do this, put a '1' for the IB2B, and a '0' for all the other options in that section, near line 35.

2. Open the PWM_Demo_Isr.ASM file. Set the example build to be built to Build 2B. To do this, put a '1' for the IB2B, and a '0' for all the other options in that section, near line 35.

3. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

4. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

5. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

6. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

7. Inspect the PWM output on EPWM1A, EPWM2A and EPWM3A on the EZDSP (pins 9,11, 13 of connector P8). Vary the duty cycle by modifying the value of the variable 'duty1'. (The waveforms below were captured with the duty1 variable set to 0x2000.)

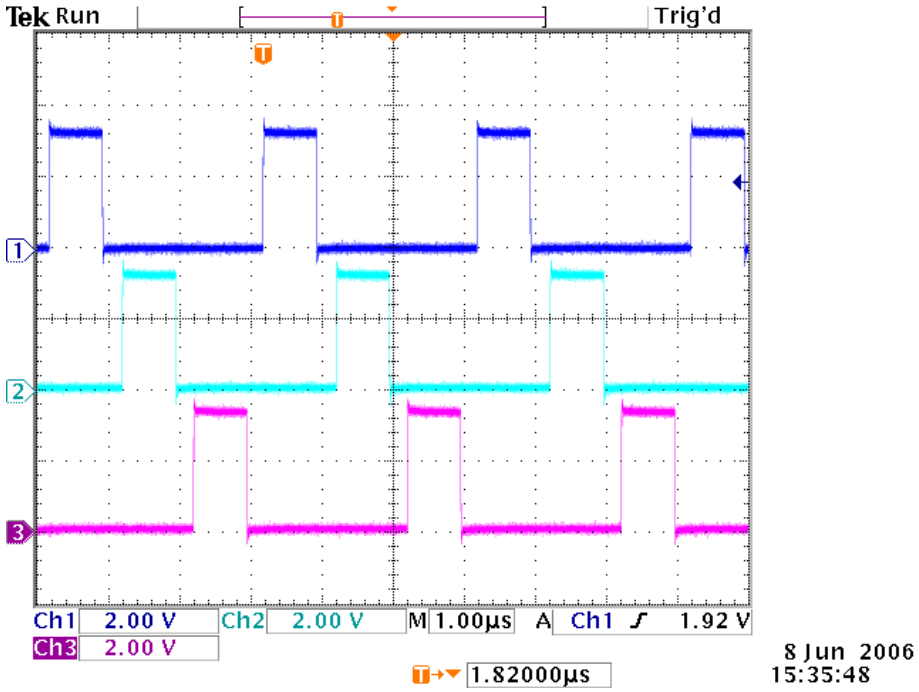

Figure 8. PWM waveform from Build2B on EPWM1A, EPWM2A and EPWM3A.

8. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.

3.5 Building and Loading Build 2C

Build 2C demonstrates the capabilities of the EPWM units to generate multiphase waveforms with ease. Four waveforms out of phase waveforms are generated.

(This build assumes continuing progress from the build 2B.)

1. Open the **PWM_Demo.c** file. Set the example build to be built to Build 2C. To do this, put a '1' for the IB2C, and a '0' for all the other options in that section, near line 35.

2. Open the PWM_Demo_Isr.ASM file. Set the example build to be built to Build 2C. To do this, put a '1' for the IB2C, and a '0' for all the other options in that section, near line 35.

3. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

4. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

5. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

6. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

7. Inspect the PWM output on EPWM1A, EPWM2A and EPWM3A, EPWM4A on the EZDSP (pins 9,11, 13, 16 of connector P8). Vary the duty cycle by modifying the value of the variable 'duty1'. (The waveforms below were captured with the duty1 variable set to 0x2000.)

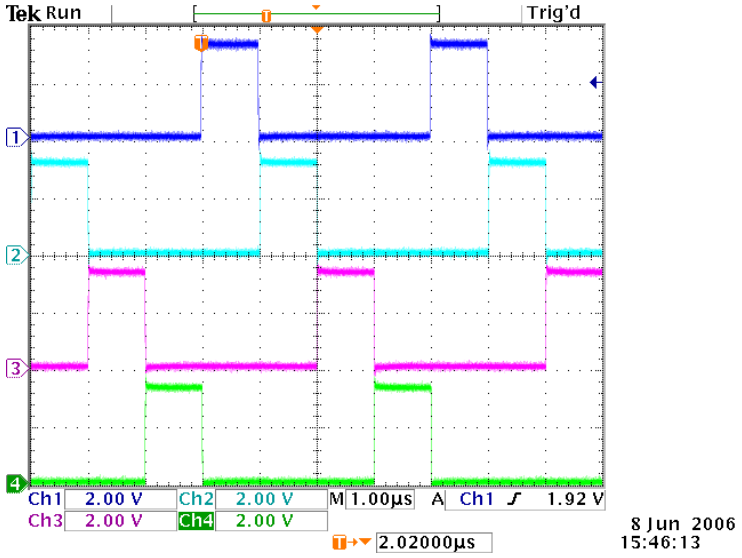

Figure 9. PWM waveform from Build 2C on EPWM1A, EPWM2A, EPWM3A and EPWM4A.

8. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.

3.6 Building and Loading Build 5

Build 5 demonstrates the capabilities of the EPWM units to generate multiphase waveforms with dead band to drive a full H bridge converter.

(This build assumes continuing progress from the build 2C.)

1. Open the **PWM_Demo.c** file. Set the example build to be built to Build 5. To do this, put a '1' for the IB5, and a '0' for all the other options in that section, near line 35.

2. Open the PWM_Demo_Isr.ASM file. Set the example build to be built to Build 5. To do this, put a '1' for the IB5, and a '0' for all the other options in that section, near line 35.

3. From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

4. To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

5. After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

6. Once things are up and running, right click on the watch window, and select continuous refresh. The 'BackTicker' variable will show activity. This is just a counter incremented in the background loop, just to provide feedback to the user of the real time mode activity.

7. Inspect the PWM output on EPWM1A, EPWM2A and EPWM3A on the EZDSP (pins 9,11, 13 of connector P8). Figure 10 shows the waveforms generated with the dead band set to 25 cycles. **Error! Reference source not found.** shows the waveforms generated with dead band of 20 cycles on EPWM1A/1B and EPWM2A/2B, with 30 cycles.

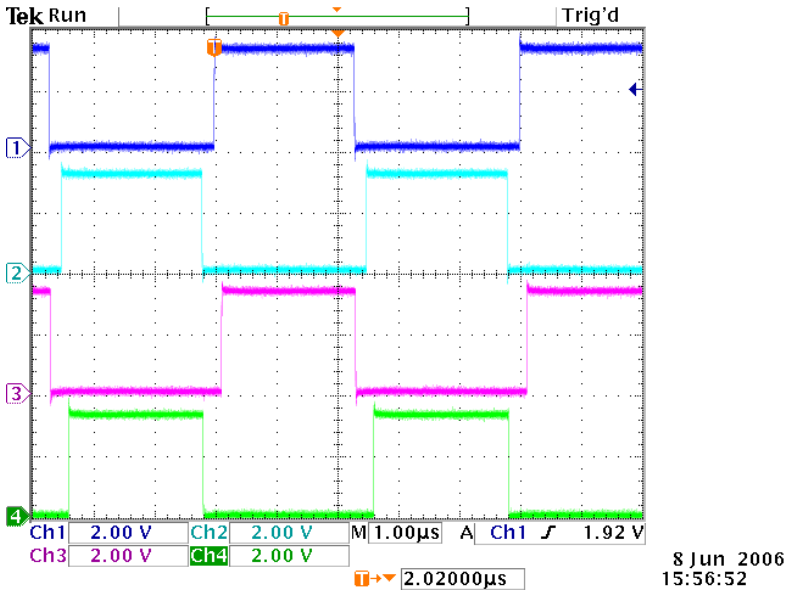

Figure 10. PWM waveform from Build 5 with a 20 cycle dead band on EPWM1A/1B and a 20 cycle dead band on EPWM2A/2B respectively.

8. To shut down, stop the DSP with the following sequence. Choose the 'Halt' option from the debug menu, then choose the 'realtime' option. This will halt the DSP and return it to stop mode. Then reset the processor.