# HR 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 high resolution 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:

- ❑ HRBUCK_DRV
- ❑ HR_PSFB_DRV
- ❑ SINGEN_1CH
- ❑ RAMPGEN_1CH
- ❑ HRPWM_DAC_DRV

In this demonstration system an incremental build approach is followed 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. If desired, an RC filter can be added to observe the modulation applied to the high resolution PWM carrier.

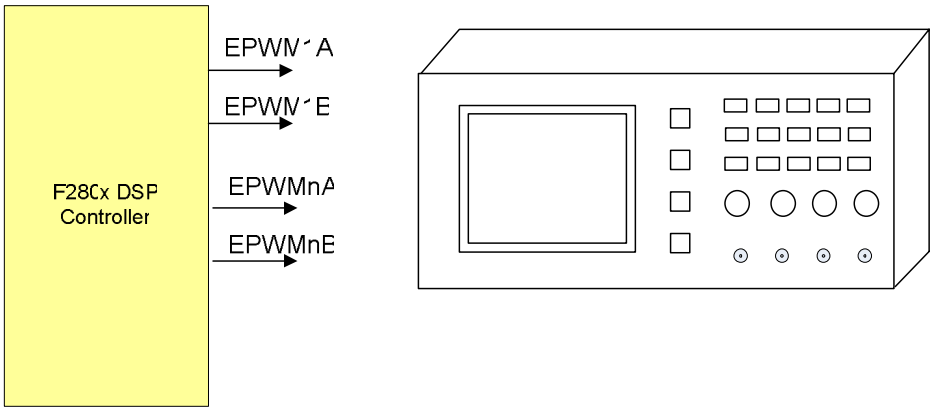


*Figure 1. Single channel sine wave generator usage*

| | |
|---|---|
| **Development/Emulation** | Code Composer Studio v2.21 (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’.

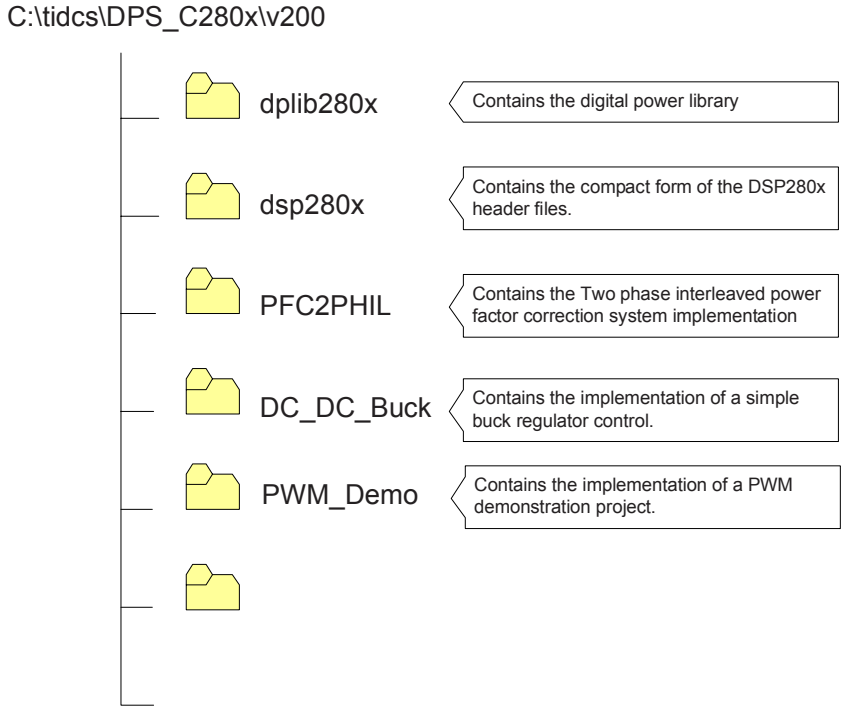


**Figure 2. Directory structure used in the software release**

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.

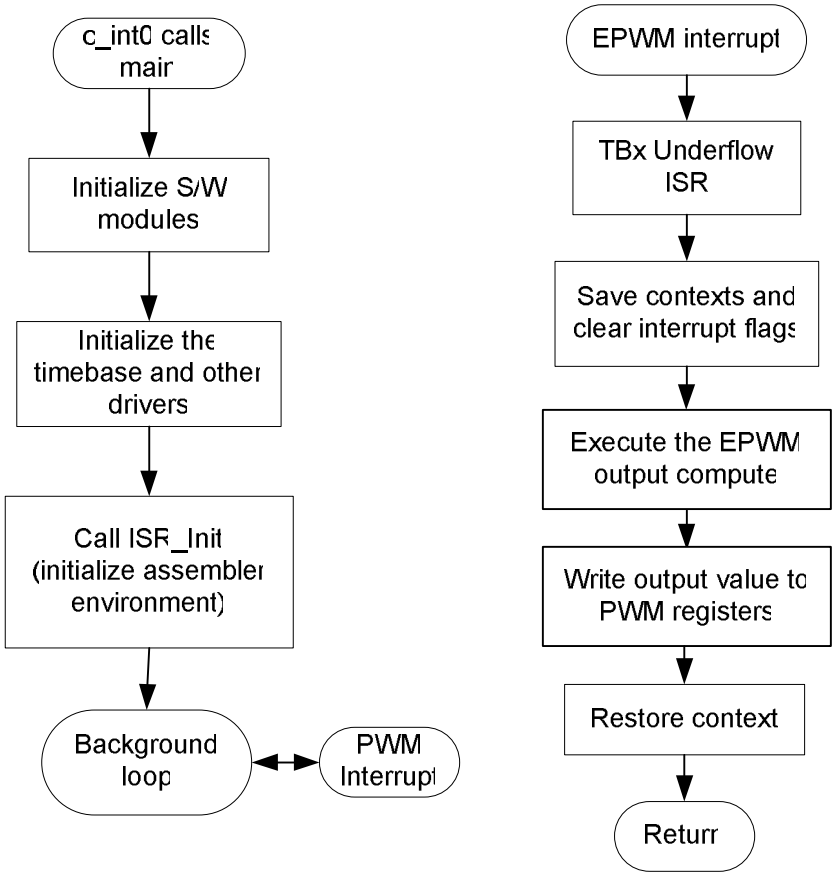


*Figure 3. Software flowchart*

## 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 high resolution PWM demonstration | |
|---|---|
| IB11 | High resolution PWM demonstration |
| IB14 | High resolution PWM with phase shift |
| IB15A | High resolution PWM with sine modulation |
| IB15B | High resolution PWM with ramp modulation |

*Table 1. Incremental build options in the HR_PWM Example*

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

# 3 Procedures for running the examples.

## 3.1 Building and Loading Build 11

The workspace file (*.wks) and project file (*.pjt) for C framework to demonstrate the HR PWM demo are located in the ***C:\tidcs\DPS_C280x\vxyz\HR_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. If so CCS issues a warning, but loads the workspace file successfully.]

Build 11 demonstrates the PWM capabilities of the HR EPWM units. Follow the steps below to build and run the examples included in the HR_PWM_Demo workspace.

1. To quickly execute demo using the pre-configured work environment, load the workspace file **HR_PWM_Demo.wks** from ***C:\tidcs\DPS_C280x\vxyz\HR_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.
3. Open the **HR_PWM_Demo.c** file. Set the example build to be built to Build 11. To do this, put a ‘1’ for the IB11, and a ‘0’ for all the other options in that section, near line 35.
4. Open the HR_PWM_Demo_Isr.ASM file. Set the example build to be built to Build 11. To do this, put a ‘1’ for the IB11, and a ‘0’ for all the other options in that section, near line 35.
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.

[†] 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.

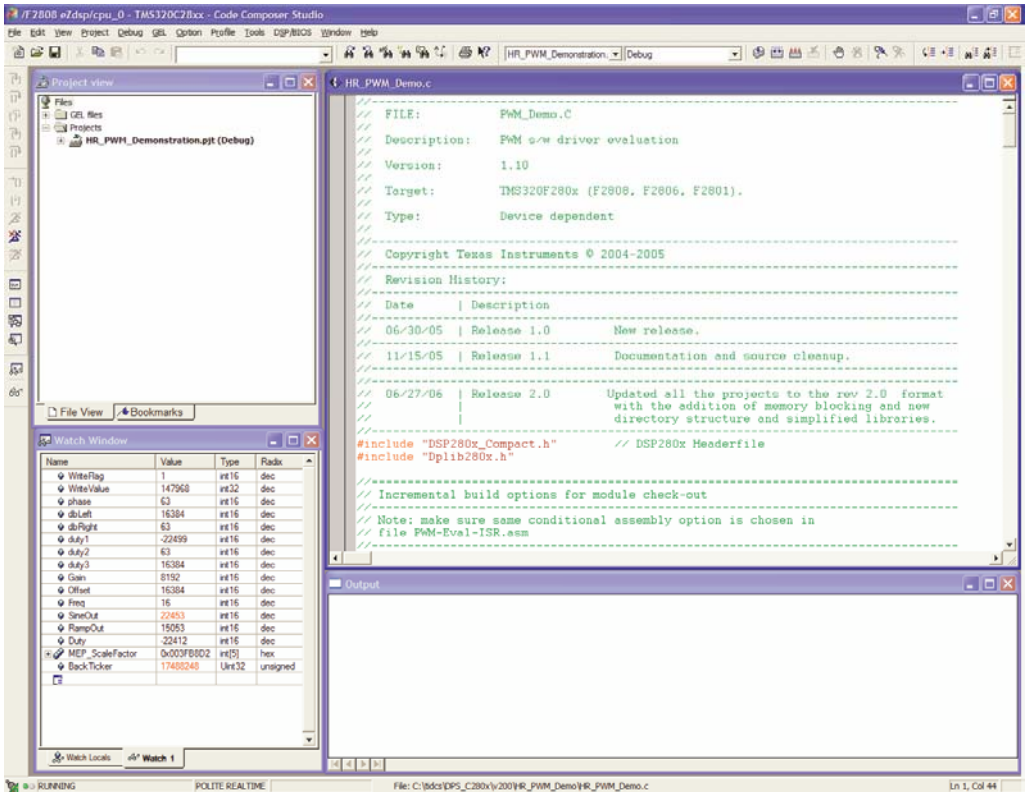


*Figure 4. CCS workspace screen capture for HR_PWM_Demo.*

6. 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: This document assumes that your F2808 EZDSP is set to boot from H0RAM, 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.**

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

8. 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.

9. Inspect the PWM output on EPWM1A (P8, pin 9), on the EZDSP. Vary the duty cycle by modifying the value of the variable 'duty1'. A screen capture of the PWM waveform is shown in Figure 5. (The value of duty1 used to capture the waveform was 0x1000.)

10. 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.

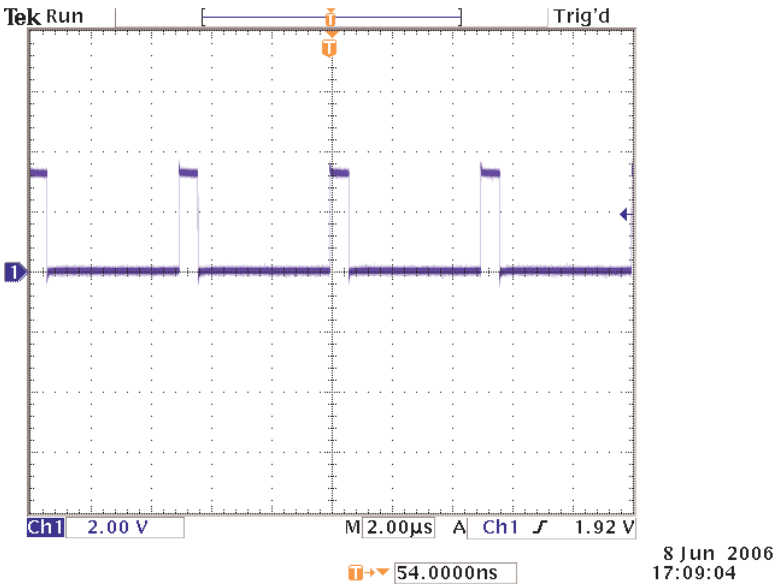


*Figure 5. PWM waveform from Build11 on EPWM1A.*

## 3.2 Building and Loading Build 14

Build 14 demonstrates the phase control capabilities of the HR EPWM units.

(This build assumes continuing progress from the build 11.)

1. Open the **HR_PWM_Demo.c** file. Set the example build to be built to Build 14. To do this, put a '1' for the IB14, and a '0' for all the other options in that section, near line 35.
2. Open the HR_PWM_Demo_Isr.ASM file. Set the example build to be built to Build 14. To do this, put a '1' for the IB14, 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.

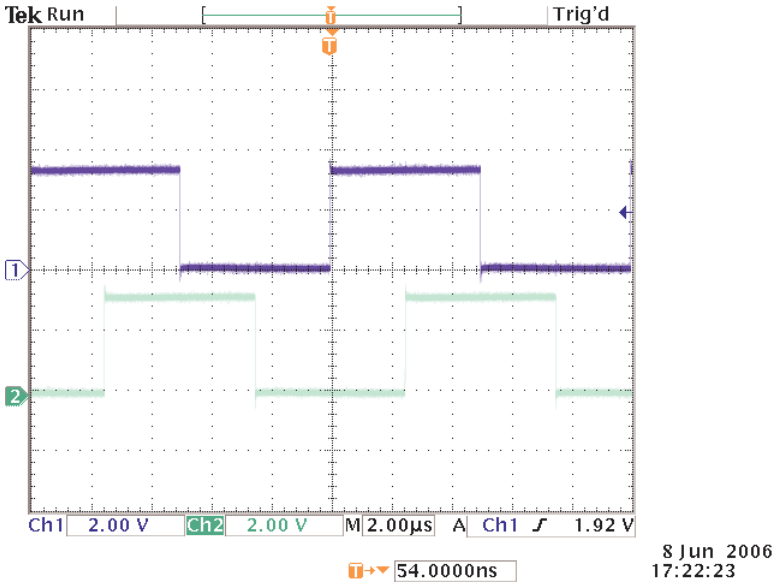


*Figure 6. PWM waveform from Build14 on EPWM1A and EPWM2A.*

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 EPWM2A (P8, pin 11) on the EZDSP. Vary the duty cycle by modifying the value of the variable 'phase'. A screen capture of the PWM waveform is shown in Figure 6, with phase set to a value of 0x4000.
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 15a

Build 15a uses the HR PWM unit(s) to output a sine signal.(This build assumes continuing progress from the build 14.)

1. Open the **HR_PWM_Demo.c** file. Set the example build to be built to Build 15a. To do this, put a '1' for the IB15a, and a '0' for all the other options in that section, near line 35.
2. Open the HR_PWM_Demo_Isr.ASM file. Set the example build to be built to Build 15a. To do this, put a '1' for the IB15a, 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 output by connecting EPWM1A (P8, pin 9) and EPWM2A to RC filters (10kohms/100nF), which filters the PWM carrier out and reconstructs the modulating waveform.
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.

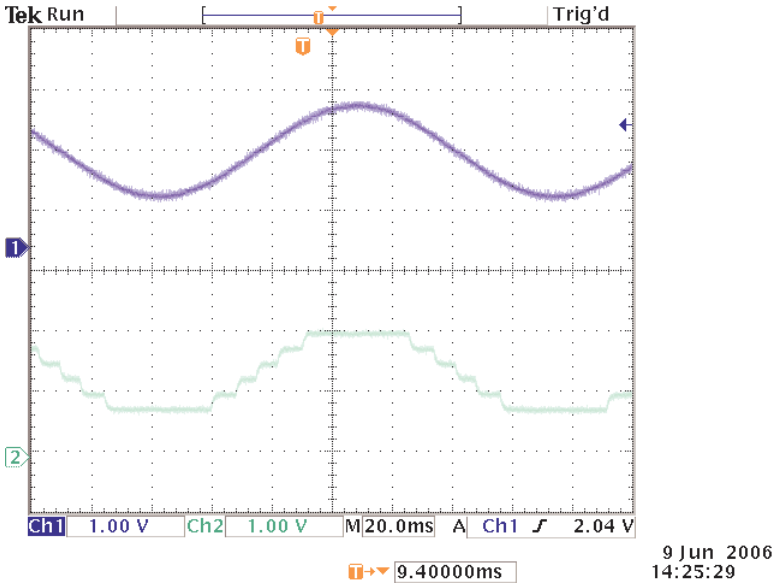


*Figure 7. Screen capture of the (filtered) EPWM1A and EPWM2A output from Build 15a. TheHigh resolution PWM can reproduce the sine wave faithfully, while the standard resolution PWM struggles, due to the high PWM frequency (8.33 MHz in this example).*

## 3.4 Building and Loading Build 15b

Build 15b implements an example where the high resolution PWM is modulated with a ramp signal generated internally in the DSP. (This build assumes continuing progress from the build 15a.)

1. Open the **HR_PWM_Demo.c** file. Set the example build to be built to Build 15b. To do this, put a '1' for the IB15b, and a '0' for all the other options in that section, near line 35.

2. Open the HR_PWM_Demo_Isr.ASM file. Set the example build to be built to Build 15b. To do this, put a '1' for the IB15b, 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 output by connecting EPWM1A (P8, pin 9) and EPWM2A to RC filters (10kohms/100nF), which filters the PWM carrier out and reconstructs the modulating waveform.

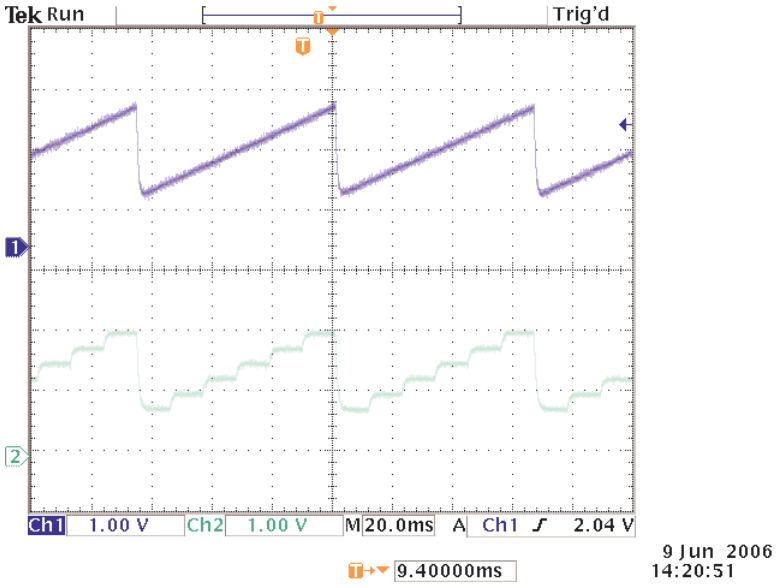


*Figure 8. Screen capture of the (filtered) EPWM1A and EPWM2A output from Build 15a. The High resolution PWM can reproduce the ramp faithfully, while the standard resolution PWM struggles, due to the high PWM frequency (8.33 MHz in this example).*

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.