Using the LP55231 Evaluation Kit

User's Guide

Literature Number: SNVU214B
April 2013–Revised December 2015

The LP55231 Evaluation Kit

This document describes how to get the LP55231 evaluation board up and running, how to use evaluation and programming software, and how to get started with programming. The application note is divided into four separate sections: General Description provides general information for getting started with the LP55231 evaluation hardware and software. LP55231 Programming shows LP55231 programming flow. Instruction Set Details gives a detailed description of the device’s instruction set. Finally, Programming Examples gives practical programming examples.

General Description

The LP55231 provides flexibility and programmability for dimming and sequencing control. Each LED can be controlled directly and independently through the serial bus (in other words, without programming the engines), or LED drivers can be controlled by programming the execution engines. The LP55231 has three independent program execution engines, so it is possible to form three independently programmable LED banks. LED drivers can be grouped based on their function so that, for example, the first bank of drivers can be assigned to the keypad illumination, the second bank to the “funlights” and the third group to the indicator LED(s). Each bank can contain 1 to 9 LED driver outputs. Instructions for program execution engines are stored in the internal program memory. The total amount of the program memory is 96 instructons and the user can allocate the memory as required by the engines. The LP55231 is programmed using an I^{2}C-compatible serial bus. Of course, it is possible to write programs for the LP55231 in the form of binary data, but the programming tools described in this document offers a more convenient way to write (and read) the registers and the SRAM memory and to program the device.

REQUIREMENTS

The following are needed to get started:

- A text editor
- LP55231 evaluation kit hardware
- LP55231 evaluation software (LP55231.exe)
- LP55231 compiler (LASM.EXE)
- FTDI virtual com port drivers (VCP)

A text editor is used to create source code for the assembler. Here, we use PSPAd as a text editor, but you should feel free to use whatever editor you're most comfortable with. PSPad is a freeware editor, © 1991 - 2007 Jan Fiala. Please see the PSPad copyright notice. PSPad text editor can be downloaded from http://www.pspad.com/

INSTALLING FTDI DRIVERS

The evaluation board has FTDI's FT232R USB to UART chip on it. FT232R allows the PC to see the evaluation board as a virtual COM port. For this FTDI VCP drivers must be installed to the host computer. Usually Windows can install the drivers automatically when the evaluation board is detected. If Windows fails to install the drivers, they can be downloaded from www.ftdichip.com. Make sure that you download and install VCP drivers and not the D2XX drivers.

Once FTDI drivers are installed, and the evaluation board is plugged into a USB port, Windows generates an USB serial port. This port is given a unique COM port number. Note that the evaluation program polls only the first 8 COM ports. If the evaluation program does not find the evaluation board you may need to change the COM port number manually. In Windows go to the Control Panel, select System, select Hardware tab, open Device Manager, select Ports and you should see the USB SerialPort number. If the port number is above 8 it will need to be changed. Right-click the USB Serial Port, select Properties, select Port Settings, click Advanced button, then select a COM port number between 1 to 8.

COPYING THE SOFTWARE

PSPad does not require installation; it can be simply unpacked into any directory. The archive contains subdirectories and must be unpacked with subdirectory preservation enabled. Also copy the Lysti.ini –file into the Syntax-folder of PSPad (for example C:\Program Files\PSPad editor\Syntax). Lysti.ini –file contains customization information for the text editor, and it must be saved into the Syntax-folder.

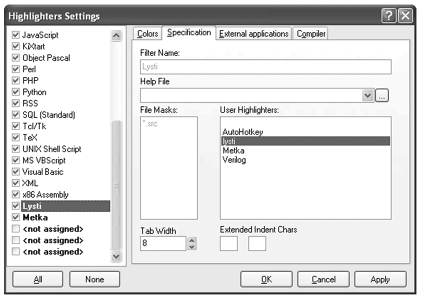
The LP55231 assembler (LASM.exe) and LP55231 evaluation software can be copied to the PC's hard disk and run without installation. The following files also need to be copied: LP55231.exe, regmap.ini, usblptio.dll, rtl60.bpl and LASM.exe. All the files must lie in the same directory. Also the source code files which are created (*.scr) should be saved/placed in the same directory as the LASM.exe before calling the assembler. Please avoid file names or directory names containing a space character, since the assembler may fail when applied to file names containing a space character. The evaluation software runs under Windows 2000/XP/Vista.

PSPAD CUSTOMIZATION

Once you run the PSPad editor, you should customize the editor for LP55231 as follows:

1. Select Settings menu > Highlighter settings.
2. Select Specification tab. See PSPad Highlighter Specification Settings for LP55231.
3. On the left (highlighter) list, click on one of bolded highlighters (marked with **<not assigned>** -tab).
4. On the right side is list of user highlighters, select the Lysti highlighter and click on it (see PSPad Highlighter Specification Settings for LP55231). Click on 'Apply' to confirm this action.

Also set the compiler search path and parameters for the LASM.exe, as shown in PSPad compiler settings for LP55231. Note: These variable names are all case-sensitive so, for example %File% will work, but %file% and %FILE% will fail to produce the expected results. . You may need to replace the shown filepath C:\LP55231\LASM\LASM.exe with your actual path. Tag the Capture Program Output Window as shown in PSPad compiler settings for LP55231. Note: These variable names are all case-sensitive so, for example %File% will work, but %file% and %FILE% will fail to produce the expected results. . Accept highlighter settings by clicking 'OK'. Finally, in the main window show the LOG window by pressing CTRL + L. All software needed should be now ready for writing and assembling the first program. Note: The maximum length of a source code line is 140 characters. Lines that are longer than 140 are not assembled correctly.



Highlighters Settings

- [✓] JavaScript
- [✓] KiXtart
- [✓] Object Pascal
- [✓] Perl
- [✓] PHP
- [✓] Python
- [✓] RSS
- [✓] SQL (Standard)
- [✓] Tcl/Tk
- [✓] TeX
- [✓] UNIX Shell Script
- [✓] MS VBScript
- [✓] Visual Basic
- [✓] XML
- [✓] x86 Assembly
- [✓] Lysti
- [✓] Metka
- [ ] <not assigned>
- [ ] <not assigned>
- [ ] <not assigned>

Specification

Filter Name: Lysti

Help File:

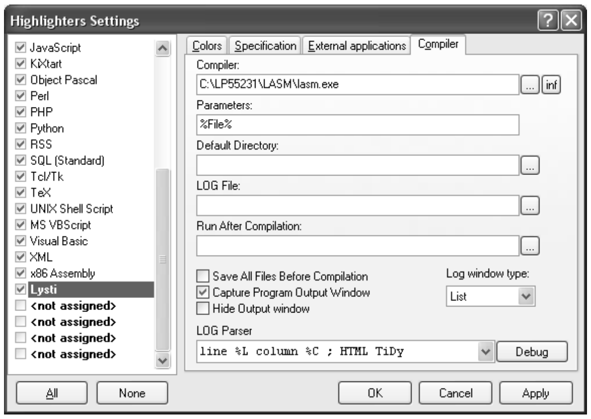
File Masks: *.src

User Highlighters: AutoHotkey, lysti, Metka, Verilog

Tab Width: 8

Extended Indent Chars:

PSPad Highlighter Specification Settings for LP55231



Highlighters Settings

- [✓] JavaScript
- [✓] KiXtart
- [✓] Object Pascal
- [✓] Perl
- [✓] PHP
- [✓] Python
- [✓] RSS
- [✓] SQL (Standard)
- [✓] Tcl/Tk
- [✓] TeX
- [✓] UNIX Shell Script
- [✓] MS VBScript
- [✓] Visual Basic
- [✓] XML
- [✓] x86 Assembly
- [✓] Lysti
- [ ] <not assigned>
- [ ] <not assigned>
- [ ] <not assigned>
- [ ] <not assigned>

Compiler

Compiler: C:\LP55231\LASM\lasm.exe

Parameters: %File%

Default Directory:

LOG File:

Run After Compilation:

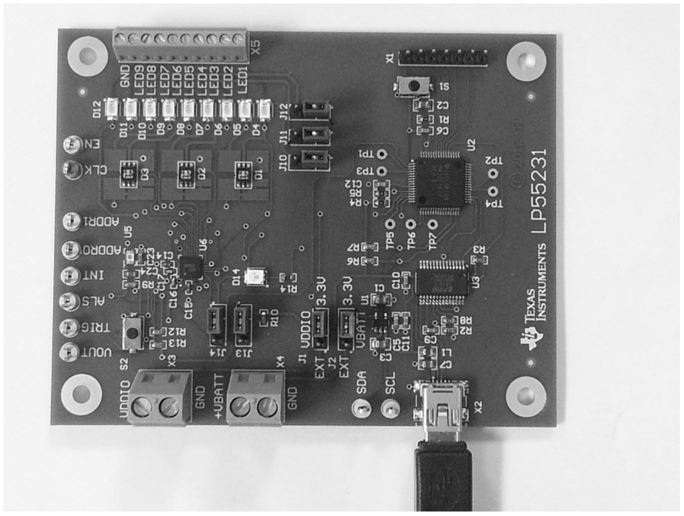
- [ ] Save All Files Before Compilation
- [✓] Capture Program Output Window
- [ ] Hide Output window

Log window type: List

LOG Parser: `line %L column %C ; HTML TiDy`

PSPad compiler settings for LP55231. Note: These variable names are all case-sensitive so, for example %File% will work, but %file% and %FILE% will fail to produce the expected results.

HARDWARE SET-UP



LP55231 Evaluation Board

The LP55231 evaluation board has USB communication and evaluation-related components assembled onto one board. (See LP55231 Evaluation Board.) The evaluation board was designed specially for evaluation and therefore is not optimized for the smallest layout size. The components are physically large to make changing of the value easy if needed. The LP55231 input voltage V_{DD} is supplied from the USB port or from an external voltage applied to the X4 connector.

LP55231 Evaluation Board (continued)

There are 7 pin connectors (jumpers) on the evaluation board for demonstrating some of the possible application options (see LP55231 Evaluation Board). The voltage supplied to the V_{DD} input of the device can be selected using J2 connector. Connecting right and center pin with jumper selects that V_{DD} is fed from USB and connecting left and center pin with jumper selects that V_{DD} is fed from connector X4. Also whether V_{DDIO} is powered from USB or from external voltage supply (X3 connector) can be selected with J1. Connecting right and center pin with jumper selects that V_{DDIO} is fed from USB and connecting left and center pin with jumper selects that V_{DDIO} is fed from connector X3. Voltage on the USB port is 5.0V and the maximum current is 500 mA. There is a voltage regulator on the evaluation board which reduces the USB bus voltage to 3.3V. Connector X4 upper connection point is for V_{DD} and lower for Ground. Connector X3 upper connection point is for V_{DDIO} and lower for Ground.

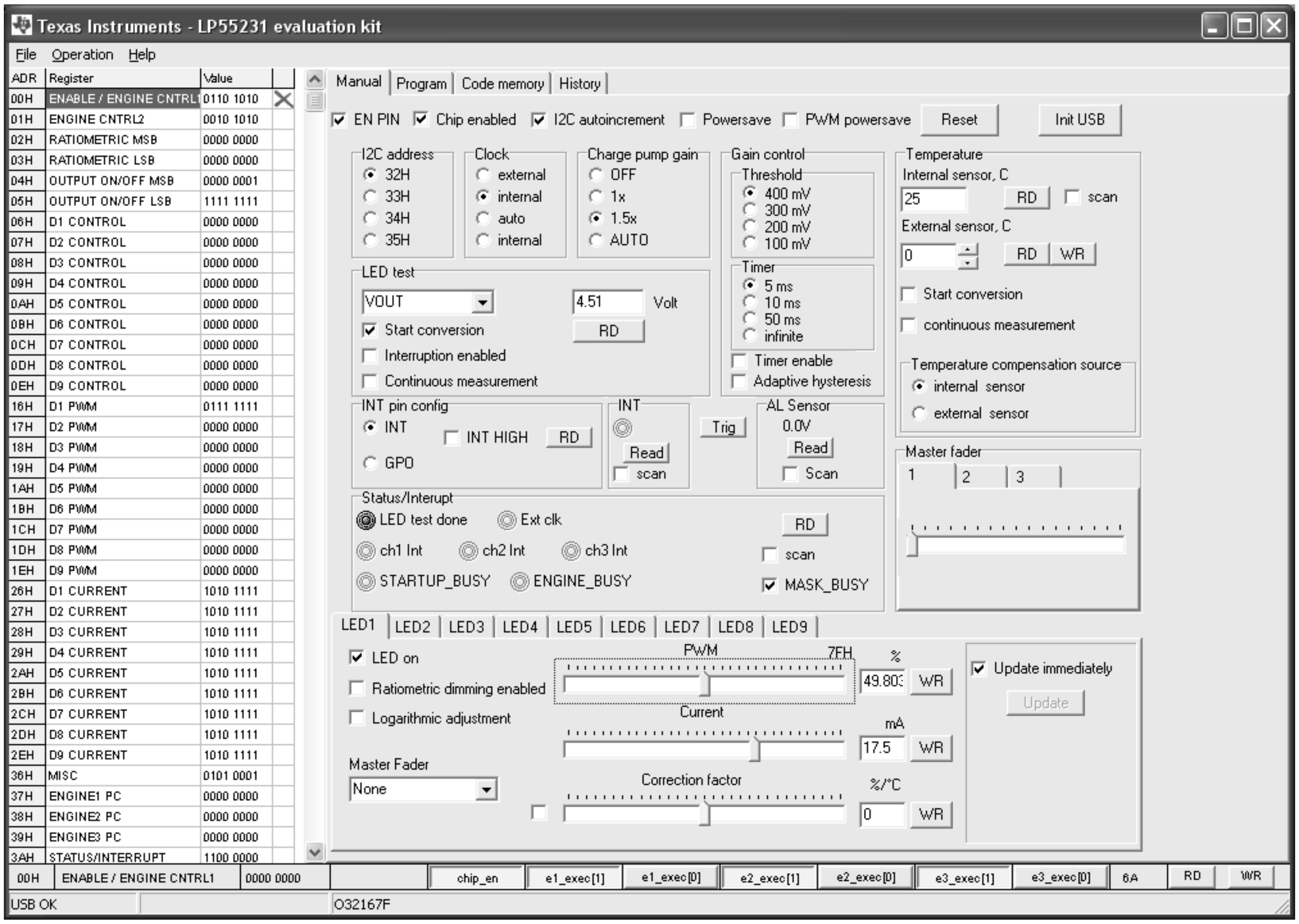
Pin connectors J10, J11, J12 are used to select whether LP55231 LED outputs are connected to WLEDs or RGBs. If lower and center pins are connected with jumper, J10 connects output 1 to D1 green, output 2 to D1 blue and output 7 to D1 red. If upper and center pins are connected with jumper output 1 is connected to D4, output 2 to D5 and output 3 to D6. Pin connector J11 connects outputs 3 to D2 green, output 4 to D2 blue, and output 8 to D2 red if lower and center pins connected with jumper. If upper and center pins are connected with jumper output 4 is connected to D7, output 5 to D8, and output 6 to D9. Pin connector J12 connects output 5 to D3 green, output 6 to D3 blue, and output 9 D3 red if lower and center pins connected with jumper. If higher and center pins are connected with jumper output 5 is connected to D10, output 8 to D11, and output 9 to D12. It is recommended that either RGB or white LEDs are used and not mixed.

Pin connectors J13 and J14 are used to connect on-board light sensor to LP55231. If J14 connectors left and center pins are connected with jumper light sensors output is connected to INT pin. If J14 connectors right and center pins are connected with jumper INT pin is connected to D14 LED. If J13 connectors left and center pins are connected with jumper light sensors GC1 pin is connected to LP55231 GPO pin. In this case pulling GPO sets the light sensor into standby mode. If this function is not desired J13 can be left open. Light sensor's GC1 pin is pulled high with R10. Jumper Options summarizes the available options.

Jumper Options

Jumper #	Left/Lower Pin	Center Pin	Right/Upper Pin
J1	VDDIO = external		
		VDDIO = 3.3V	
J2	VBATT = external		
		VBATT = 3.3V	
J10	RGB LED D1 in use (LED1 = G, LED2 = B, LED7 = R)		
		WLEDs in use (LED1 = D4, LED2 = D5, LED3 = D6)	
J11	RGB LED D2 in use (LED3 = G, LED4 = B, LED8 = R)		
		WLEDs in use (LED4 = D7, LED5 = D8, LED6 = D9)	
J12	RGB LED D3 in use (LED5 = G, LED6 = B, LED9 = R)		
		WLEDs in use (LED7 = D10, LED8 = D11, LED9 = D12)	
J13	Light sensor GC1 pin connected to LP55231 GPO pin. GPO can be used to disable light sensor.		
		Not needed. GC1 pulled up with R10 resistor.	
J14	INT pin connected to light sensor output.		
		INT pin connected to D14 LED.	

CONNECTING EVALUATION BOARD TO COMPUTER


Texas Instruments - LP55231 evaluation kit

File Operation Help

ADR	Register	Value
00H	ENABLE / ENGINE CNTRL1	0110 1010
01H	ENGINE CNTRL2	0010 1010
02H	RATIOMETRIC MSB	0000 0000
03H	RATIOMETRIC LSB	0000 0000
04H	OUTPUT ON/OFF MSB	0000 0001
05H	OUTPUT ON/OFF LSB	1111 1111
06H	D1 CONTROL	0000 0000
07H	D2 CONTROL	0000 0000
08H	D3 CONTROL	0000 0000
09H	D4 CONTROL	0000 0000
0AH	D5 CONTROL	0000 0000
0BH	D6 CONTROL	0000 0000
0CH	D7 CONTROL	0000 0000
0DH	D8 CONTROL	0000 0000
0EH	D9 CONTROL	0000 0000
16H	D1 PWM	0111 1111
17H	D2 PWM	0000 0000
18H	D3 PWM	0000 0000
19H	D4 PWM	0000 0000
1AH	D5 PWM	0000 0000
1BH	D6 PWM	0000 0000
1CH	D7 PWM	0000 0000
1DH	D8 PWM	0000 0000
1EH	D9 PWM	0000 0000
26H	D1 CURRENT	1010 1111
27H	D2 CURRENT	1010 1111
28H	D3 CURRENT	1010 1111
29H	D4 CURRENT	1010 1111
2AH	D5 CURRENT	1010 1111
2BH	D6 CURRENT	1010 1111
2CH	D7 CURRENT	1010 1111
2DH	D8 CURRENT	1010 1111
2EH	D9 CURRENT	1010 1111
36H	MISC	0101 0001
37H	ENGINE1 PC	0000 0000
38H	ENGINE2 PC	0000 0000
39H	ENGINE3 PC	0000 0000
3AH	STATUS/INTERRUPT	1100 0000

Manual | Program | Code memory | History

[✓] EN PIN [✓] Chip enabled [✓] I2C autoincrement [ ] Powersave [ ] PWM powersave Reset Init USB

I2C address: [✓ 32H] 33H 34H 35H

Clock: external [✓ internal] auto internal

Charge pump gain: OFF 1x [✓ 1.5x] AUTO

Gain control – Threshold: [✓ 400 mV] 300 mV 200 mV 100 mV

Timer: [✓ 5 ms] 10 ms 50 ms infinite

[ ] Timer enable [ ] Adaptive hysteresis

Temperature – Internal sensor, C: 25 RD [ ] scan; External sensor, C: 0 RD WR

[ ] Start conversion [ ] continuous measurement

Temperature compensation source: [✓ internal sensor] external sensor

LED test: VOUT 4.51 Volt RD; [✓] Start conversion [ ] Interruption enabled [ ] Continuous measurement

INT pin config: [✓ INT] GPO [ ] INT HIGH RD

INT: Read [ ] scan Trig

AL Sensor: 0.0V Read [ ] Scan

Status/Interupt: LED test done, Ext clk, ch1 Int, ch2 Int, ch3 Int, STARTUP_BUSY, ENGINE_BUSY RD [ ] scan [✓] MASK_BUSY

Master fader: 1 2 3

LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 | LED8 | LED9

[✓] LED on [ ] Ratiometric dimming enabled [ ] Logarithmic adjustment

Master Fader: None

PWM 7FH % 49.803 WR

Current mA 17.5 WR

Correction factor %/°C 0 WR

[✓] Update immediately Update

00H ENABLE / ENGINE CNTRL1 0000 0000 chip_en e1_exec[1] e1_exec[0] e2_exec[1] e2_exec[0] e3_exec[1] e3_exec[0] 6A RD WR

USB OK O32167F

LP55231 Evaluation Software Control Panel

1. Check that the jumpers on the evaluation board are on wanted positions.
2. Connect the evaluation board to your computer using a USB cable.
3. If this is first time using the evaluation board, install FTDI's VCP drivers.
4. Start the evaluation software *LP55231.exe*.
5. Click Init USB button.
6. Reset the LP55231 circuit by clicking the Reset button on upper right corner of the window. In the message box that appears, click OK to confirm the register reading.
7. The evaluation kit is now fully up and running, and the device can be controlled through the PC software. LP55231 Evaluation Software Control Panel shows the evaluation software user interface (Control Panel).

You should see USB OK message on the status bar (shown in the lower part of the window). If the USB communication is not working correctly, shut down the evaluation software and unplug the USB cable. Plug in the USB cable again and reset the evaluation board by pressing the on-board reset button (S1). Wait about 5 seconds and repeat steps 4 to 6 above. If evaluation software is still unable to find evaluation board, change the com port settings as described in chapter *INSTALLING FTDI DRIVERS*.

CONTROLLING EVALUATION BOARD FROM PC

The evaluation software provides read-write control over the registers within the LP55231. Bits can be set from a logical ‘1’ to a logical ‘0’ or vice versa by a mouse click; for some settings there is also a slider control provided. The evaluation software window is divided into two panels: control panel and indicator panel (see LP55231 Evaluation Software Control Panel). The leftmost panel is the indicator panel, and it shows the status of the LP55231 registers (excluding the program memory).

The rightmost panel is the control panel -- it is used to change the status of the registers, to program the device, or to debug the program depending on the selected view. The view is selected by the tabs on the top of the window. The Manual view appears when the evaluation software starts. The Manual view is used to run the device "manually"; i.e., the program execution engines are not used. It contains also some LP55231 basic settings, like serial bus address selection. In the next chapter the user is guided through the Manual view and basic operations of the LP55231. The other views are Program, Code memory, and History; these views are presented in LP55231 Programming covering programming of the LP55231.

Tip: The context-sensitive buttons in the lower part of the window can be used for direct Write/Read operations of the registers. The target register is indicated by red cross and the target can be changed by clicking on the desired register on the indicator panel.

DIRECT CONTROL

Enabling the Device and Starting the Charge Pump

The first step to start with the LP55231 is to enable the chip; tag the Chip enabled check box. At first it is best to use the following settings: Clock internal and Charge pump gain 1.5x (see LP55231 Evaluation Software Control Panel). At this point it is good to ensure the charge pump output voltage with the LP55231 built-in LED error detection. To do this, select VOUT from the LED test drop-down menu and tag Start conversion. Use the adjacent RD button to read the result: it should be around 4.5 volts. Also click on the RD button on the Status/Interrupt section to verify that the internal clock is used (Ext clk indicator light should be OFF), and that the LED test is done (LED test done indicator should be ON).

Setting the Led Current and PWM

The next step is to set the desired LED currents. This is done by Current slider. Set 5 mA current for LED1, and set PWM to 50% = 7Fhex. You should have a green LED switched on now. Note: When the Update immediately tag is set, the evaluation software will write the new settings immediately to the LP55231 registers. Otherwise, click on the Update button.

The basic idea behind LP55231 operation is to first set the LED currents to the required level; after that the current settings are not touched any more – all the fade-in and fade-out sections (ramps), and the temperature compensation is done by PWM . Each LED has its own constant current output, and all the outputs can be controlled independently. Thus LED pre-selection (matched LEDs) is not required, and this kind of architecture supports also color control. The PWM, current, and temperature compensation controls are organized under tabs labelled by LED1 to LED 9. Now, let’s set 5 mA current and 50% PWM for LED2 and 3.5 mA current and 50% PWM for LED7. As a result, D1 on evaluation board should emit white light.

Master Fader

The PWM of LED driver outputs can be controlled individually as described above; alternatively the drivers can be grouped by function to provide a quick control. The LP55231 has three master fader registers, so it is possible to form three master fader groups. To assign an LED to a group, use the Master Fader drop-down menu. Select Group 1 for LEDs 1, 2, and 7. Now you can control all the three outputs with a single master fader register. There is a Master fader slider provided on the right side of the control panel and dragging the slider under tab labelled by 1 controls now LEDs 1, 2, and 7. Note that the initial PWM and current for LEDs in a group needs to be set before using the master fader control, since master fader is simply a multiplier, which acts on the PWM registers.

Logarithmic vs Linear PWM Response

Logarithmic response is used to give the impression of a linear light intensity increase/decrease as the PWM duty cycle is raised/reduced. Logarithmic response is visually more pleasing especially when the overall brightness is low. To activate the logarithmic response, tag logarithmic adjustment. Activate logarithmic adjustment for LEDs 1, 2, and 7. Set also 5 mA current for LEDs 3 and 4, set 3.5mA for LED8. Set 50% PWM for LEDs 3, 4, and 8. Assign LEDs 3, 4, and 8 to the master fader group 1 so that you can control all the six LEDs with one slider. Now you should have logarithmic control over the LEDs 1, 2, 7 and linear control over the LEDs 3, 4, 8. To see the difference between the lin and log response, move the slider backwards and forwards to alter the intensity of all the six LEDs at once.

Temperature Compensation

The LP55231 has a temperature compensation function to correct for variations in light intensity and color caused by changes in ambient temperature.

Reset the LP55231 and start the charge pump as shown in LP55231 Evaluation Software Control Panel. Set 5 mA current, 50% PWM for LEDs 1 and 2; set 3.5 mA current, 50% PWM for LED 7.

In order to activate the compensation function tag the check box next to Correction factor slider. The slope for the temperature compensation line can be set by the slider. A simple approximation for the RGB LED temperature compensation would be +1.3%/°C for red LED and +0.2%/°C for green and blue so that the intensity of all the colors remain approximately the same over the temperature from 25°C to 60°C.

To observe the effect of the compensation on color, try the following: Under Temperature label, tag Start conversion (this enables the LP55231 internal temperature sensor) and continuous measurement (continuous temperature measurement). Enter 25°C to the External sensor box and click on WR button to write the reading to the LP55231 memory. Now toggle between internal sensor and external sensor as you warm up the LEDs and LP55231 with a hair dryer. When the “external sensor” is active, the temperature information is read from register TEMPERATURE WRITE (addr. 40H). When the temperature is 25°C all the compensation settings have no effect, so when you have the external sensor activated, you will see the “uncompensated” situation. When you activate the internal sensor, you will see the effect of the compensation as the ambient temperature is raised. Without compensation emitted light will be drifted somewhat towards a more bluish white, because the red LED element of the RGB LED shows the strongest temperature dependence.

LED Error Detection

To measure VF of an LED set PWM = 100% for the LED and set the desired LED current with the current slider. Disable temperature compensation. On LED test portion tag Start conversion and Continuous measurement. Click on RD button to read the test result from LP55231’s internal register. Try with different current settings and compare the result with the value specified by the manufacturer. If there is a short to ground in the LED circuit the result is ~0V, and if there is an open the result is ~4.5V. This feature can also be addressed to measure the voltage on VDD, VOUT, and INT pins. Typical example usage includes monitoring battery voltage or using INT pin as a light sensor interface (A/D converter value can be used as a variable in program control).

Charge Pump Gain Control

Charge pump gain can be forced to 1x or 1.5x; alternatively, the user can let the LP55231 decide the best charge pump gain based on LED forward voltage requirements by selecting AUTO mode (this is the most useful setting in real applications). Note that outputs D7, D8, and D9 are powered directly from V_{DD}, and they have no effect on gain change decision-making. LP55231 charge pump has a gain change hysteresis which prevents the mode from toggling back and forth (1x -> 1.5x -> 1x...), which would cause ripple on VIN and LED flicker.

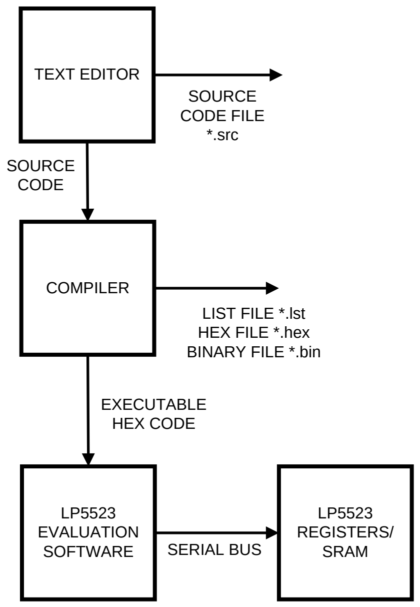
The hysteresis region width depends on the choice of threshold voltage. Threshold voltage is defined as follows VTHRESHOLD = VDD - MAX(voltage on D1 to D6). If VTHRESHOLD is larger than the set value (100 mV to 400 mV), the charge pump is in 1x mode. A good compromise solution between efficiency and performance is 200 mV. In addition, to take the charge pump total load current into account, enable the adaptive hysteresis.

Timer enable activates forced mode change. In forced mode charge pump mode change from 1.5x to 1x is attempted at the constant interval specified by the Timer control. With infinite setting the charge pump switches gain from 1x mode to 1.5x mode only. The gain reset back to 1x is enabled under certain conditions, for example in the power-save mode. Forced mode and threshold parameters are used to optimize efficiency in a real design - it is desired that 1.5x gain is used only when needed.

LP55231 Programming

PROGRAMMING FLOW CHART

Programming Flow Chart shows the typical programming flow of the LP55231. The program is first typed in with PSPad (or equivalent) text editor. Then the program is compiled into a hex and binary file. Finally the hex file is loaded into the LP55231's memory and tested.



Programming Flow Chart

RESERVED KEYWORDS

The names of registers and instructions are assembler-reserved keywords. For the LP55231, the following words are reserved and may not be used as statement labels:

Register names

- ra
- rb
- rc
- rd

Instructions

- ramp
- set_pwm
- wait
- mux_ld_start
- mux_ld_end
- mux_map_start
- mux_sel
- mux_clr

- mux_map_next
- mux_map_prev
- mux_ld_next
- mux_ld_prev
- mux_ld_addr
- mux_map_addr
- rst
- branch
- int
- end
- trigger
- jne
- jl
- jge
- je
- ld
- add
- sub

Directives

- .segment
- ds
- dw

THE STRUCTURE OF AN LP55231 PROGRAM

Example Code of LP55231 Program shows an example of an LP55231 program. When this program is run, the program will flash a red LED once per second.

Although this program is short and simple it shows all the main parts of a typical LP55231 program.

Commenting

Commenting starts with a semicolon (;). The compiler will ignore all characters after a semicolon. If you are using PSPad editor and have customized the editor according to the instructions on first pages of this document, the editor recognizes comments, directives, labels, and instructions automatically and uses different highlighter colors for different data types.

Directives

The directives are not translated directly into LP55231. Instead, directives are instructions for the LASM.exe compiler. Directives are used to adjust the location of the engine 1, 2, and 3 programs in memory and reserve memory resources in the LP55231 SRAM. For example **.segment** program1 is a directive which tells the compiler that whatever follows is the program for the program execution engine 1. An overview of the directives is given in the following table.

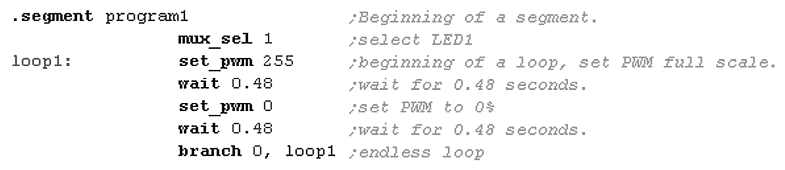
Directive	Description	Example source code
.segment	Adjust the location of the programs in SRAM. Note the leading dot	**.segment** program1 **.segment** flashing_light
ds	**D**efine **S**torage; The directive reserves memory resources in the SRAM. The ds directive takes one parameter, which is the number of words to reserve. The number of bits in a word (word length) is 16. The allocated words are initialized with zeros	**ds** 3 **ds** 17
dw	**D**efine constant **W**ord. Inserts a binary word to the SRAM.	**dw** 0000000011111111b **dw** FFABh **dw** 3

Labels

A label is a symbolic address. Labels are used to mark program line(s), like in branch instruction and labeling mapping table rows. Labels must have the colon (:) suffix. In the Example Code of LP55231 Program code *loop1:* is a label which indicates the starting address of the loop.

Instructions

Instructions are executable statements. LASM-compiler translates text-based language source instructions into hex- and binary-based executable codes. For example in the code presented below Example Code of LP55231 Program, **set_pwm** is an instruction. Almost all the instructions take operands, which may be constants (like FFh), or variables stored in the LP55231 registers (like ra) - ra stands for variable a (register a), rb, for variable b, etc.



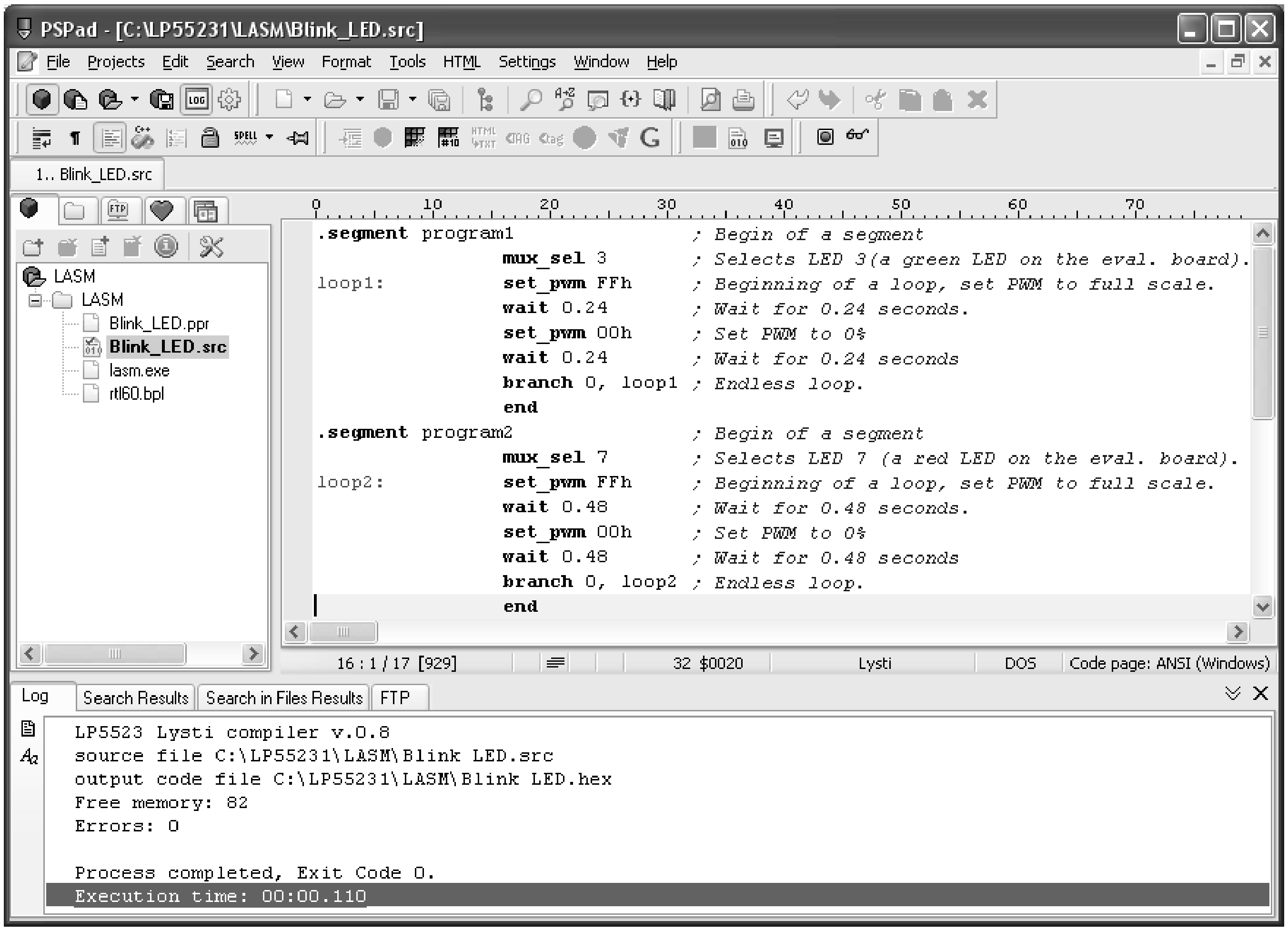
```
.segment program1               ;Beginning of a segment.
                mux_sel 1       ;select LED1
loop1:          set_pwm 255     ;beginning of a loop, set PWM full scale.
                wait 0.48       ;wait for 0.48 seconds.
                set_pwm 0       ;set PWM to 0%
                wait 0.48       ;wait for 0.48 seconds.
                branch 0, loop1 ;endless loop
```

Example Code of LP55231 Program

PRODUCING AN EXECUTABLE FILE

Once the text-based source file is typed in and saved using the text editor (PSPad), the source code window should look like the one in Example of Compiling Source Code. To call the compiler routine, select *File >> Compile*. The PSPad LOG window shows the progress of compiling. If the compiler generates error messages, LOG window is necessary for locating these errors.

A listing file, a hex file, and a binary file are produced by the LASM.exe compiler. The name of the files are the same name that you have given to the source code file, with the *.lst, *.hex or *.bin extension. *.hex and *.bin files contain the machine code.



```
.segment program1                  ; Begin of a segment
             mux_sel 3             ; Selects LED 3(a green LED on the eval. board).
loop1:       set_pwm FFh           ; Beginning of a loop, set PWM to full scale.
             wait 0.24             ; Wait for 0.24 seconds.
             set_pwm 00h           ; Set PWM to 0%
             wait 0.24             ; Wait for 0.24 seconds
             branch 0, loop1       ; Endless loop.
             end
.segment program2                  ; Begin of a segment
             mux_sel 7             ; Selects LED 7 (a red LED on the eval. board).
loop2:       set_pwm FFh           ; Beginning of a loop, set PWM to full scale.
             wait 0.48             ; Wait for 0.48 seconds.
             set_pwm 00h           ; Set PWM to 0%
             wait 0.48             ; Wait for 0.48 seconds
             branch 0, loop2       ; Endless loop.
             end
```

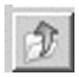
```
LP5523 Lysti compiler v.0.8
source file C:\LP55231\LASM\Blink LED.src
output code file C:\LP55231\LASM\Blink LED.hex
Free memory: 82
Errors: 0

Process completed, Exit Code 0.
Execution time: 00:00.110
```

Example of Compiling Source Code

LOADING A PROGRAM TO THE LP55231's SRAM

To upload code into the LP55231 SRAM start the LP55231.exe evaluation software. Select the Program tab Master Control of the Program Tab. The Program tab is divided into two parts: the right contains the program's source code and the compiled version of the code; the left part contains program execution engine controls. Load the generated *.hex file into the evaluation software view: Click Open Source File button (Open Source File Button In Program Tab), browse the file and click Open.



Open Source File Button In Program Tab

To download the machine code into the LP55231, click on download buttonDownload Opened Source File Into LP55231. Pressing this button sets all the engine modes to *Load*.

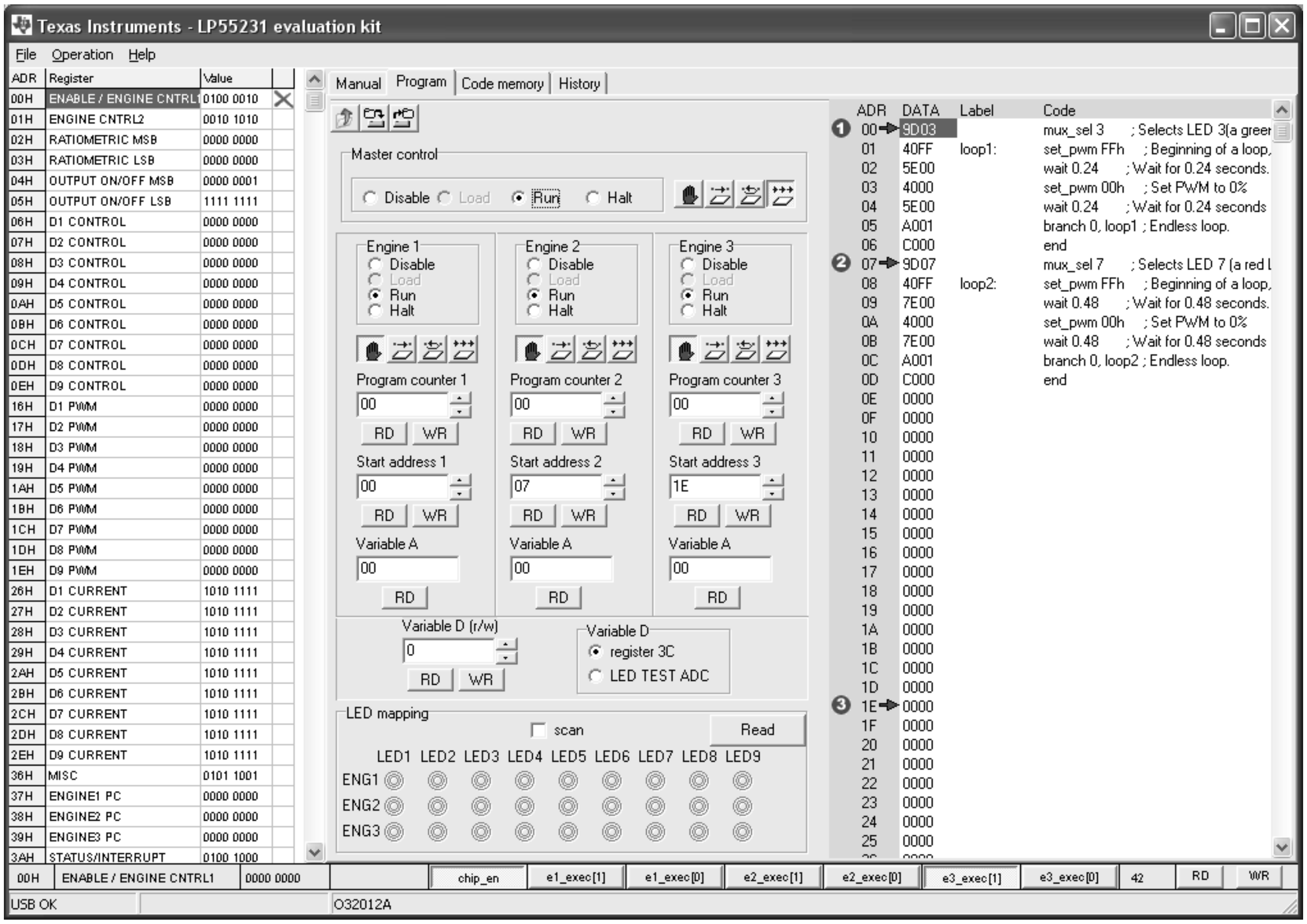
Open Source File Button In Program Tab (continued)

Download Opened Source File Into LP55231

Program can be loaded also from the *Code memory* tab (see Code Memory Tab). In this case one must set the operation mode to *Load*. In *Code memory* tab, one must first select the address where data is written. Once the address is active, hexcidecimal data can be written in the Data entry field, then push the Update button. Once all the data is updated to the wanted addresses, it can be written to the SRAM memory by pushing the Write Page, which writes two lines in code memory table (first two lines refer to page 0, next two lines to page 1, etc. Pages can be changed with the page selector) or by pushing Write All, which updates the whole memory. Once the data is written to SRAM, operation mode can be set to *Run* and Execution mode for example to *Free Run*. Note that Program Counter(s) must be set accordingly. This is not done automatically like loading program by clicking the Download-button. Note also that the program code does not show up in the program view (in *Program* tab).

RUNNING THE PROGRAM

The program is run by checking the Run from the Master control and clicking on Free run-button. This way all the engines will start at the same time. If you have fewer than three engines in use, extra engines must be disabled by checking off the box in each engine section. See Master Control of the Program Tab below: this program has only two engines in use, and the engine three is checked off. If this is not done the programs may not work correctly. Engines can also be run from individual engine control. One convenient way of debugging the programs is by running the individual engines step by step. Individual engine control can be managed also with multiple engines, but then they won't start at the same time.



ADR	Register	Value
00H	ENABLE / ENGINE CNTRL1	0100 0010
01H	ENGINE CNTRL2	0010 1010
02H	RATIOMETRIC MSB	0000 0000
03H	RATIOMETRIC LSB	0000 0000
04H	OUTPUT ON/OFF MSB	0000 0001
05H	OUTPUT ON/OFF LSB	1111 1111
06H	D1 CONTROL	0000 0000
07H	D2 CONTROL	0000 0000
08H	D3 CONTROL	0000 0000
09H	D4 CONTROL	0000 0000
0AH	D5 CONTROL	0000 0000
0BH	D6 CONTROL	0000 0000
0CH	D7 CONTROL	0000 0000
0DH	D8 CONTROL	0000 0000
0EH	D9 CONTROL	0000 0000
16H	D1 PWM	0000 0000
17H	D2 PWM	0000 0000
18H	D3 PWM	0000 0000
19H	D4 PWM	0000 0000
1AH	D5 PWM	0000 0000
1BH	D6 PWM	0000 0000
1CH	D7 PWM	0000 0000
1DH	D8 PWM	0000 0000
1EH	D9 PWM	0000 0000
26H	D1 CURRENT	1010 1111
27H	D2 CURRENT	1010 1111
28H	D3 CURRENT	1010 1111
29H	D4 CURRENT	1010 1111
2AH	D5 CURRENT	1010 1111
2BH	D6 CURRENT	1010 1111
2CH	D7 CURRENT	1010 1111
2DH	D8 CURRENT	1010 1111
2EH	D9 CURRENT	1010 1111
36H	MISC	0101 1001
37H	ENGINE1 PC	0000 0000
38H	ENGINE2 PC	0000 0000
39H	ENGINE3 PC	0000 0000
3AH	STATUS/INTERRUPT	0100 1000

Master control: Disable Load [✓ Run] Halt

	Engine 1	Engine 2	Engine 3
Mode	Disable Load [✓ Run] Halt	Disable Load [✓ Run] Halt	Disable Load [✓ Run] Halt
Program counter	00	00	00
Start address	00	07	1E
Variable A	00	00	00

Variable D (r/w): 0

Variable D: [✓ register 3C] LED TEST ADC

LED mapping

	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8	LED9
ENG1	◎	◎	◎	◎	◎	◎	◎	◎	◎
ENG2	◎	◎	◎	◎	◎	◎	◎	◎	◎
ENG3	◎	◎	◎	◎	◎	◎	◎	◎	◎

```
   ADR  DATA  Label    Code
❶  00→  9D03           mux_sel 3      ; Selects LED 3(a greer
   01   40FF  loop1:   set_pwm FFh    ; Beginning of a loop,
   02   5E00           wait 0.24      ; Wait for 0.24 seconds.
   03   4000           set_pwm 00h    ; Set PWM to 0%
   04   5E00           wait 0.24      ; Wait for 0.24 seconds
   05   A001           branch 0, loop1 ; Endless loop.
   06   C000           end
❷  07→  9D07           mux_sel 7      ; Selects LED 7 (a red l
   08   40FF  loop2:   set_pwm FFh    ; Beginning of a loop,
   09   7E00           wait 0.48      ; Wait for 0.48 seconds.
   0A   4000           set_pwm 00h    ; Set PWM to 0%
   0B   7E00           wait 0.48      ; Wait for 0.48 seconds
   0C   A001           branch 0, loop2 ; Endless loop.
   0D   C000           end
   0E   0000
   0F   0000
   10   0000
   11   0000
   12   0000
   13   0000
   14   0000
   15   0000
   16   0000
   17   0000
   18   0000
   19   0000
   1A   0000
   1B   0000
   1C   0000
   1D   0000
❸  1E→  0000
   1F   0000
   20   0000
   21   0000
   22   0000
   23   0000
   24   0000
   25   0000
```

00H ENABLE / ENGINE CNTRL1 0000 0000   42

USB OK   O32012A

Master Control of the Program Tab

Master Control of the Program Tab (continued)

There are four modes for operating the programs described below (see Master Control of the Program Tab). Operation mode is selected by clicking the desired check box.

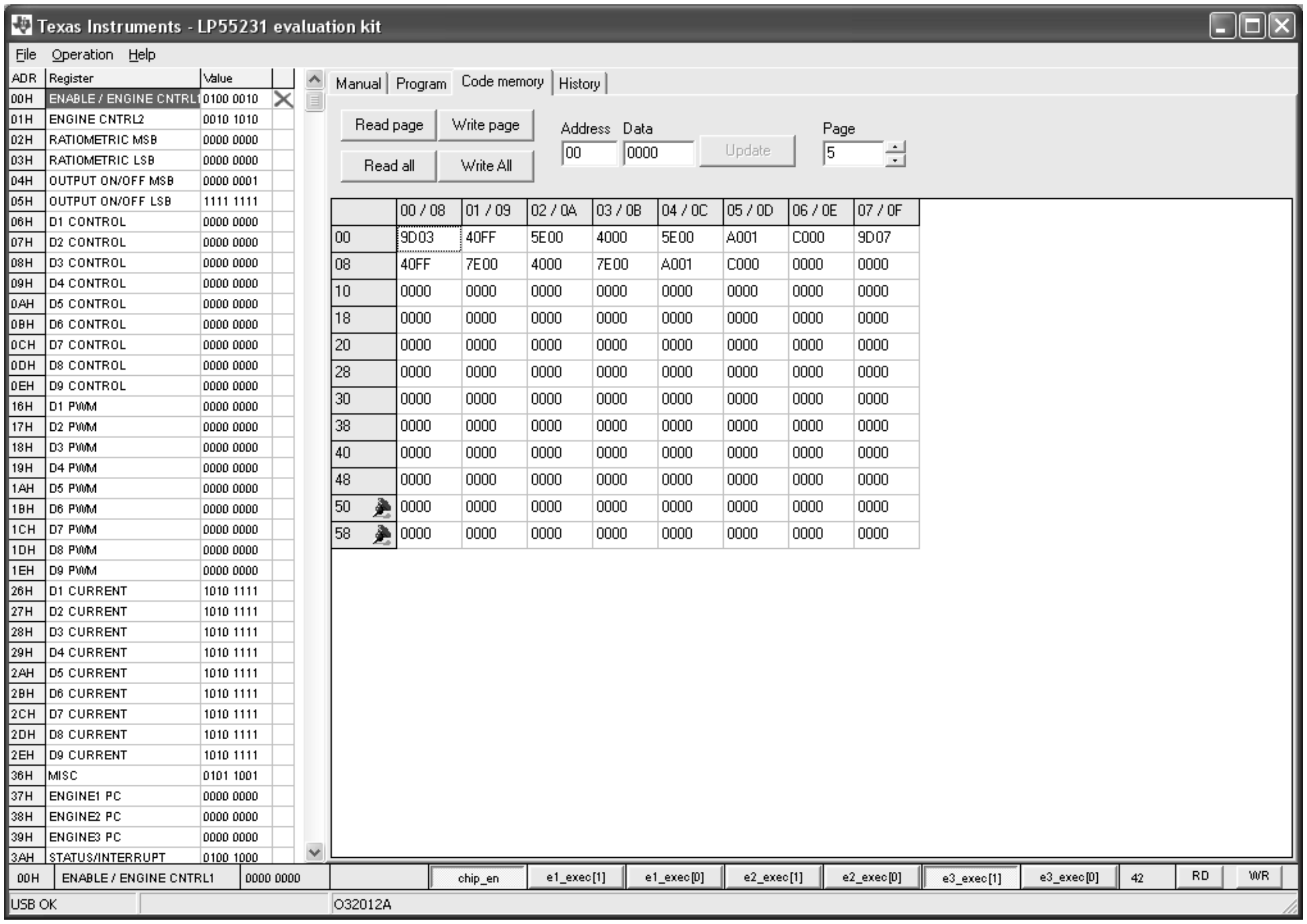
Operation modes:

- **Disable** — Engine operation is disabled, and they cannot be run.
- **Load** — In this mode writing to program memory is allowed. All the three engines are on hold while one or more engines are in load program mode. PWM values are also frozen. Program execution continues when all the engines are out of load program mode. Load program mode resets the program counter of the respective engine. Load program mode can be entered from the Disable mode only. Entering load program mode from the run program mode is not allowed. Note that load mode does not automatically load the program opened with Open Source File button (see Open Source File Button In Program Tab). When using this operation mode, one must write the program through the Code memory tab .
- **Run** — This mode executes the instructions stored in the program memory. Execution register (ENG1_EXEC etc.) bits define how the program is executed (hold, step, free run, or execute once). Program start address can be programmed to Program Counter (PC) register. The PC is reset to zero when the PC's upper limit value is reached.
- **Halt** — In this mode instruction execution aborts immediately, and engine operation halts. Execution can be continued if operation mode is set to Run again.

The following four execution modes exist. An execution mode is selected with one of the four push buttons. Functions of the buttons from left to right are:

Executions modes:

- **Stop** — Engine execution is stopped. The current instruction is executed and then execution stopped.
- **Step** — Execute the instruction at the location pointed by the PC, increment the PC by one and then reset ENG1_EXEC bits to 00 (enter Stop-mode).
- **Execute Command** — Execute the instruction pointed by the current PC value and reset ENG1_EXEC to 00 (i.e. enter Stop-mode). The difference between Step and Execute Command is that Execute Command does not increment the PC.
- **Free Run** — Start program execution from the instruction pointed by the PC.



ADR	Register	Value
00H	ENABLE / ENGINE CNTRL1	0100 0010
01H	ENGINE CNTRL2	0010 1010
02H	RATIOMETRIC MSB	0000 0000
03H	RATIOMETRIC LSB	0000 0000
04H	OUTPUT ON/OFF MSB	0000 0001
05H	OUTPUT ON/OFF LSB	1111 1111
06H	D1 CONTROL	0000 0000
07H	D2 CONTROL	0000 0000
08H	D3 CONTROL	0000 0000
09H	D4 CONTROL	0000 0000
0AH	D5 CONTROL	0000 0000
0BH	D6 CONTROL	0000 0000
0CH	D7 CONTROL	0000 0000
0DH	D8 CONTROL	0000 0000
0EH	D9 CONTROL	0000 0000
16H	D1 PWM	0000 0000
17H	D2 PWM	0000 0000
18H	D3 PWM	0000 0000
19H	D4 PWM	0000 0000
1AH	D5 PWM	0000 0000
1BH	D6 PWM	0000 0000
1CH	D7 PWM	0000 0000
1DH	D8 PWM	0000 0000
1EH	D9 PWM	0000 0000
26H	D1 CURRENT	1010 1111
27H	D2 CURRENT	1010 1111
28H	D3 CURRENT	1010 1111
29H	D4 CURRENT	1010 1111
2AH	D5 CURRENT	1010 1111
2BH	D6 CURRENT	1010 1111
2CH	D7 CURRENT	1010 1111
2DH	D8 CURRENT	1010 1111
2EH	D9 CURRENT	1010 1111
36H	MISC	0101 1001
37H	ENGINE1 PC	0000 0000
38H	ENGINE2 PC	0000 0000
39H	ENGINE3 PC	0000 0000
3AH	STATUS/INTERRUPT	0100 1000

Address: 00 Data: 0000 Page: 5

	00 / 08	01 / 09	02 / 0A	03 / 0B	04 / 0C	05 / 0D	06 / 0E	07 / 0F
00	9D03	40FF	5E00	4000	5E00	A001	C000	9D07
08	40FF	7E00	4000	7E00	A001	C000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000
18	0000	0000	0000	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000
28	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
38	0000	0000	0000	0000	0000	0000	0000	0000
40	0000	0000	0000	0000	0000	0000	0000	0000
48	0000	0000	0000	0000	0000	0000	0000	0000
50	0000	0000	0000	0000	0000	0000	0000	0000
58	0000	0000	0000	0000	0000	0000	0000	0000

00H ENABLE / ENGINE CNTRL1 0000 0000 42

USB OK O32012A

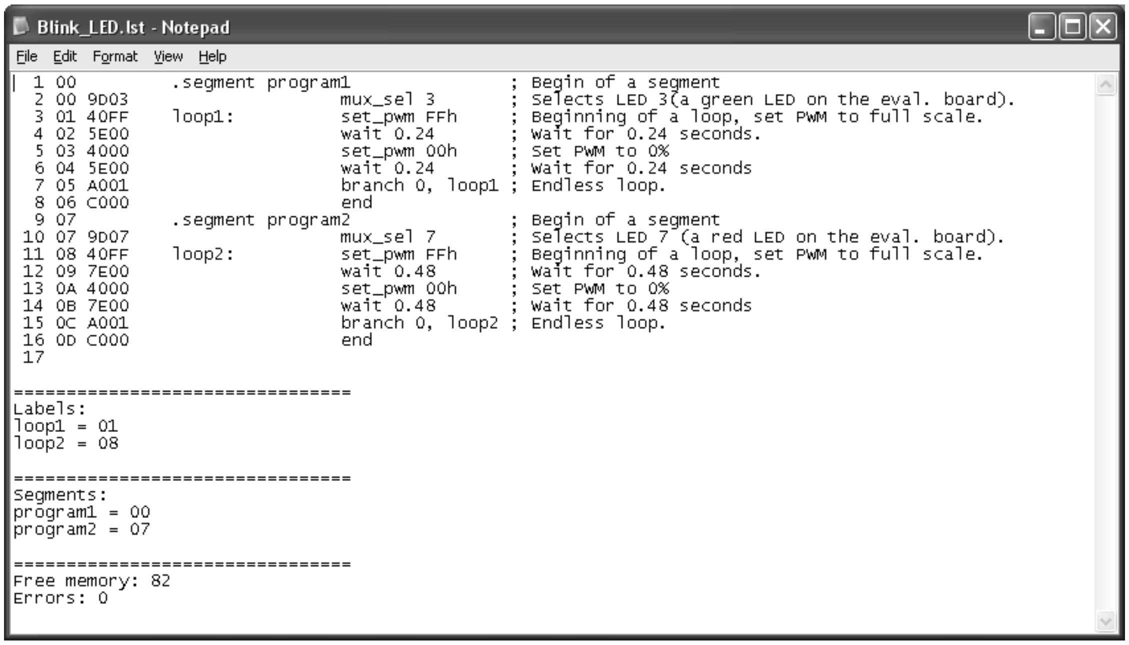
Code Memory Tab

DEBUGGING CONSIDERATIONS

There are a few ways to see how the compiler translates the instructions to machine code. Listing file (*.lst) may be used for locating assembling errors. The listing file contains the source code along with the compiled machine code. The files can be examined in any text editor. This is helpful for debugging and seeing how source code is translated into machine code. Listing File of the Example Program First shows an example of listing file. The first column is the row number, second column indicates the SRAM memory address, third shows the machine code data, and fourth column includes the source code. Note that the .segment directives show the start address of the program; i.e., where to the Program Counters should point.

From the produced two hex-files user can see the pure machine code represented in hexadecimal in two different ways. In *.he2 —file representation of data is 0xYY (YY being the changing data information). In *.hex file the data is represented YY, where YY is the changing data information. In *.he2 file the 8–bit long data elements are separated with comma whereas in *.hex file they are separated with tab. In both files data is represented like in *Code memory* tab memory table. First two columns correspond to first column in *Code memory* tab memory table, third and fourth columns correspond to second column etc. For example if the user would have mux_inc instruction (9D80), in he2 it would be 0x9D, 0x80 and in hex file it would be 9D [tab] 80. In hex-file the start addresses of the programs are at the bottom, whereas in *.he2 file they are on the first row engine 1 start address being first, engine 2 second, and engine 3 start address third. Also in the bin-file user can see the pure machine code represented in binary. First three rows represent the start addresses of the programs. After the start addresses the program code follows.

Programs can be debugged in the evaluation software *Program* tab by running the program in steps using *Step* or *Execute command* execution modes. Also one way to see what is written to the LP55231 is to look at the evaluation program *History* tab History Tab.



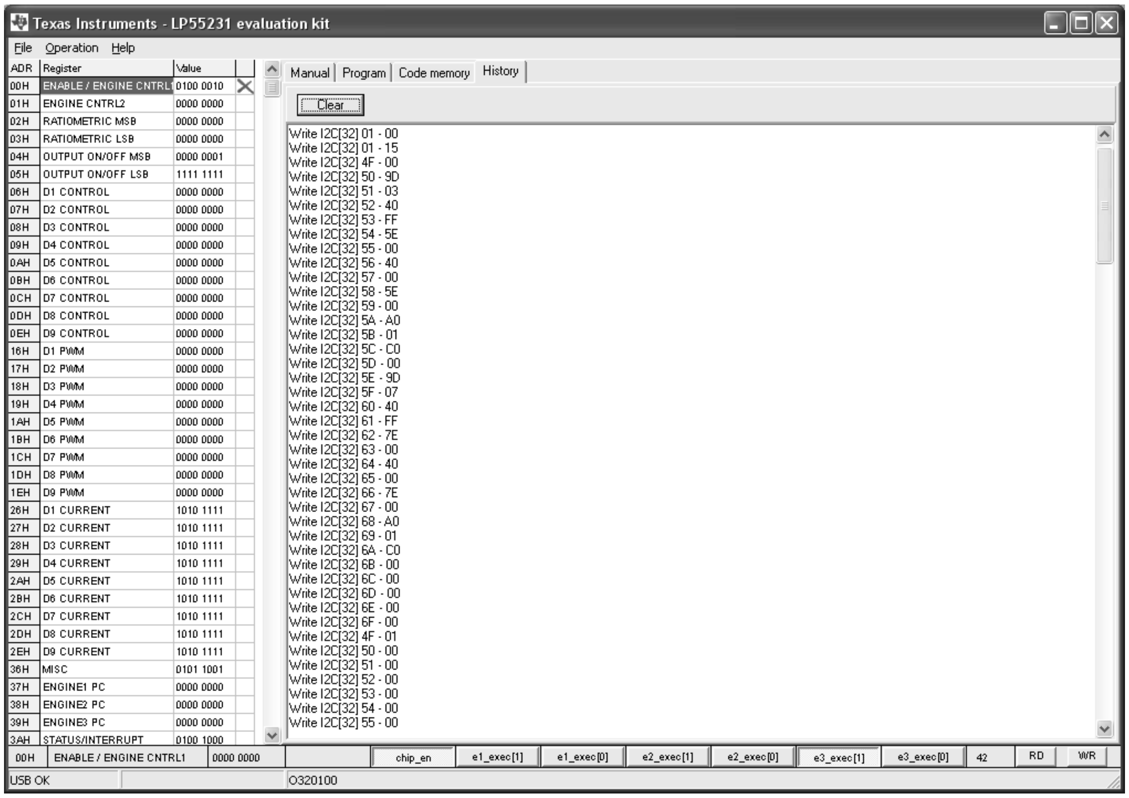
```
00         .segment program1          ; Begin of a segment
00 9D03                mux_sel 3      ; Selects LED 3(a green LED on the eval. board).
01 40FF    loop1:      set_pwm FFh    ; Beginning of a loop, set PWM to full scale.
02 5E00                wait 0.24      ; Wait for 0.24 seconds.
03 4000                set_pwm 00h    ; Set PWM to 0%
04 5E00                wait 0.24      ; Wait for 0.24 seconds
05 A001                branch 0, loop1 ; Endless loop.
06 C000                end
07         .segment program2          ; Begin of a segment
07 9D07                mux_sel 7      ; Selects LED 7 (a red LED on the eval. board).
08 40FF    loop2:      set_pwm FFh    ; Beginning of a loop, set PWM to full scale.
09 7E00                wait 0.48      ; Wait for 0.48 seconds.
0A 4000                set_pwm 00h    ; Set PWM to 0%
0B 7E00                wait 0.48      ; Wait for 0.48 seconds
0C A001                branch 0, loop2 ; Endless loop.
0D C000                end


================================
Labels:
loop1 = 01
loop2 = 08

================================
Segments:
program1 = 00
program2 = 07

================================
Free memory: 82
Errors: 0
```

Listing File of the Example Program First



ADR	Register	Value
00H	ENABLE / ENGINE CNTRL1	0100 0010
01H	ENGINE CNTRL2	0000 0000
02H	RATIOMETRIC MSB	0000 0000
03H	RATIOMETRIC LSB	0000 0000
04H	OUTPUT ON/OFF MSB	0000 0001
05H	OUTPUT ON/OFF LSB	1111 1111
06H	D1 CONTROL	0000 0000
07H	D2 CONTROL	0000 0000
08H	D3 CONTROL	0000 0000
09H	D4 CONTROL	0000 0000
0AH	D5 CONTROL	0000 0000
0BH	D6 CONTROL	0000 0000
0CH	D7 CONTROL	0000 0000
0DH	D8 CONTROL	0000 0000
0EH	D9 CONTROL	0000 0000
16H	D1 PWM	0000 0000
17H	D2 PWM	0000 0000
18H	D3 PWM	0000 0000
19H	D4 PWM	0000 0000
1AH	D5 PWM	0000 0000
1BH	D6 PWM	0000 0000
1CH	D7 PWM	0000 0000
1DH	D8 PWM	0000 0000
1EH	D9 PWM	0000 0000
26H	D1 CURRENT	1010 1111
27H	D2 CURRENT	1010 1111
28H	D3 CURRENT	1010 1111
29H	D4 CURRENT	1010 1111
2AH	D5 CURRENT	1010 1111
2BH	D6 CURRENT	1010 1111
2CH	D7 CURRENT	1010 1111
2DH	D8 CURRENT	1010 1111
2EH	D9 CURRENT	1010 1111
36H	MISC	0101 1001
37H	ENGINE1 PC	0000 0000
38H	ENGINE2 PC	0000 0000
39H	ENGINE3 PC	0000 0000
3AH	STATUS/INTERRUPT	0100 1000

```
Write I2C[32] 01 - 00
Write I2C[32] 01 - 15
Write I2C[32] 4F - 00
Write I2C[32] 50 - 9D
Write I2C[32] 51 - 03
Write I2C[32] 52 - 40
Write I2C[32] 53 - FF
Write I2C[32] 54 - 5E
Write I2C[32] 55 - 00
Write I2C[32] 56 - 40
Write I2C[32] 57 - 00
Write I2C[32] 58 - 5E
Write I2C[32] 59 - 00
Write I2C[32] 5A - A0
Write I2C[32] 5B - 01
Write I2C[32] 5C - C0
Write I2C[32] 5D - 00
Write I2C[32] 5E - 9D
Write I2C[32] 5F - 07
Write I2C[32] 60 - 40
Write I2C[32] 61 - FF
Write I2C[32] 62 - 7E
Write I2C[32] 63 - 00
Write I2C[32] 64 - 40
Write I2C[32] 65 - 00
Write I2C[32] 66 - 7E
Write I2C[32] 67 - 00
Write I2C[32] 68 - A0
Write I2C[32] 69 - 01
Write I2C[32] 6A - C0
Write I2C[32] 6B - 00
Write I2C[32] 6C - 00
Write I2C[32] 6D - 00
Write I2C[32] 6E - 00
Write I2C[32] 6F - 00
Write I2C[32] 4F - 01
Write I2C[32] 50 - 00
Write I2C[32] 51 - 00
Write I2C[32] 52 - 00
Write I2C[32] 53 - 00
Write I2C[32] 54 - 00
Write I2C[32] 55 - 00
```

00H ENABLE / ENGINE CNTRL1 0000 0000

USB OK O320100

History Tab

Instruction Set Details

This section provides the syntax with detailed examples for all the LP55231 instructions supported by the LASM assembler.

LED DRIVER INSTRUCTIONS

INSTRUCTION SYNTAX	FUNCTION	EXAMPLE	16-BIT ASSEMBLED BIT SEQUENCE	ASSEMBLED CODE HEX
ramp time, PWM	Output PWM with increasing / decreasing duty cycle.	ramp 0.6, 255	0000 1010 1111 1111	0A FF
Time is a positive constant (0.000484*PWM to 0.484*PWM);		;Ramp up to full scale over 0.6s		
PWM is a positive or negative constant (-255 to 255).		ramp 1.2,-255	0001 0101 1111 1111	15 FF
Note: time is rounded by assembler if needed.		;Ramp down to zero over 1.2s		
ramp var1, prescale, var2	Output PWM with increasing / decreasing duty cycle.	ld ra, 31	1000 0100 0000 0001	84 01
Var1 is a variable (ra, rb, rc, rd);		ld rb, 255		
Prescale is a boolean constant (pre=0 or pre=1);		ramp ra, pre=0,+rb		
Var2 is a variable (ra, rb, rc, rd).		;Ramp up to full scale over 3.9s		
		ld ra, 1	1000 0100 0001 0001	84 11
		ld rb, 255		
		ramp ra, pre=0,-rb		
		;Ramp down to zero over 0.12s		
set_pwm PWM	Generate a continuous PWM output.	set_pwm 128	0100 0000 1000 0000	4080
PWM is a constant (0-255 or 0 - FFh).		;Set PWM Duty-Cycle to 50%		
set_pwm var1	Generate a continuous PWM output.	ld rc, 128	1000 0100 0110 0010	8462
Var1 is a variable (ra, rb, rc, rd).		set_pwm rc		
		;Set PWM Duty-Cycle to 50%		
wait time	Pause for some time.	wait 0.25	0110 0000 0000 0000	6000
Time is a positive constant (0 to 0.484).		;Wait 0.25 seconds		
Note: time is rounded by assembler if needed.				

LED MAPPING INSTRUCTIONS

INSTRUCTION SYNTAX	FUNCTION	EXAMPLE	16-BIT ASSEMBLED BIT SEQUENCE	ASSEMBLED CODE HEX
mux_ld_start address	Defines the start address of the mapping data table.	mux_ld_start row1	1001 1110 0000 0000	9E00
Address is a label which specifies where to find the first row.		; The first row can be found at the address marked with row1	Assumed that row1 points to addr 00h.	
mux_map_start address	Defines the start address of the mapping data table and sets the row active.	mux_map_start row1	1001 1100 0000 0000	9C00
Address is a label which specifies where to find the first row.		; The first row can be found at the address marked with row1	Assumed that row1 points to addr 00h.	
mux_ld_end address	Defines the end address of the mapping data table.	mux_ld_end row9	1001 1100 1000 1000	9C88
Address is a label which specifies where to find the last row.		; The last row can be found at the address marked with row9	Assumed that row9 points to addr 08h.	
mux_sel output	Connects one and only one LED output to an engine.	mux_sel 1	1001 1101 0000 0001	9D01
Output is a constant (0 to 9 or 16).		; D1 output will be connected to the engine.		

mux_clr	Clears engine-to-driver mapping.	mux_clr	1001 1101 0000 0000	9D00
mux_map_next	Sets the next row active in the mapping table.	mux_map_next	1001 1101 1000 0000	9D80
mux_map_prev	Sets the previous row active in the mapping table.	mux_map_prev	1001 1101 1100 0000	9DC0
mux_ld_next	The index pointer will be set to point to the next row in the mapping table.	mux_ld_next	1001 1101 1000 0000	9D81
mux_ld_prev	The index pointer will be set to point to the previous row in the mapping table.	mux_ld_prev	1001 1101 1100 0000	9DC1
mux_ld_addr address	Sets the index pointer to point the mapping table row defined by address.	mux_ld_addr row2	1001 1111 0000 0001	9F01
Address is a label which specifies the row to which the pointer is to be moved.		; The index pointer will be set to point to the row labelled with row2.	Assumed that row2 points to addr 01h.	
mux_map_addr address	Sets the index pointer to point the mapping table row defined by address and sets the row active.	mux_map_addr row2	1001 1111 1000 0001	9F81
Address is a label which specifies the row of the table that will be set active.		; The index pointer will be set to point to the row labelled with row2 and the row will be set active.	Assumed that row2 points to addr 01h.	

BRANCH INSTRUCTIONS

INSTRUCTION SYNTAX	FUNCTION	EXAMPLE	16-BIT ASSEMBLED BIT SEQUENCE	ASSEMBLED CODE HEX
rst	Resets program counter and start the program again.	rst	0000 0000 0000 0000	0000
branch loopcount, address	Repeat a section of code.	branch 20, loop1	1010 1010 0000 0000	AA00
Loop count is a constant (0 to 63);		; define loop for 20 times	Assumed that loop1 points to addr 00h.	
Address is a label which specifies the offset.				
branch var1, address	Repeat a section of code.	ld ra, 20	1000 0110 0000 0000	8600
Var1 is a variable (ra, rb, rc, rd);		branch ra, loop1	Assumed that loop1 points to addr 00h.	
Address is a label which specifies the offset.		; define loop for 20 times		
int	Causes an interrupt.	int	1100 0100 0000 0000	C400
end interrupt, reset	End program execution.	end i	1101 0000 0000 0000	D000
Interrupt (i) is an optional flag. Reset (r) is an optional flag.		; End program execution and send an interrupt.		
trigger w{source1|source2...}	Wait a trigger.	trigger w{1}	1110 0000 1000 0000	E080
Source is the source of the trigger (1, 2, 3, e).		;Wait a		
		trigger from the engine 1.		
trigger s{target1|target2...}	Send a trigger.	trigger s{1}	1110 0000 0000 0010	E002
Target is the target of the trigger (1, 2, 3, e).		;Send a		
		trigger to the engine 1.		

jne var1, var2, address	Jump if not equal.	jne ra, rb, flash	1000 1000 0010 0001	8821
Var1 is a variable (ra, rb, rc, rd);		;Jump to flash if A != B.	Assumed that offset = 2.	
Var2 is a variable (ra, rb, rc, rd);				
Address is a label which specifies the offset.				
jl var1, var2, address	Jump if less.	jl ra, rb, flash	1000 1010 0001 0001	8A11
Var1 is a variable (ra, rb, rc, rd);		;Jump to flash if A < B.	Assumed that offset = 1	
Var2 is a variable (ra, rb, rc, rd);				
Address is a label which specifies the offset.				
jge var1, var2, address	Jump if greater or equal.	jge ra, rb, flash	1000 1100 0001 0001	8C11
Var1 is a variable (ra, rb, rc, rd);		;Jump to flash if A >= B.	Assumed that offset = 1.	
Var2 is a variable (ra, rb, rc, rd);				
Address is a label which specifies the offset.				
je var1, var2, address	Jump if equal.	je ra, rb, flash	1000 1110 0001 0001	8E11
Var1 is a variable (ra, rb, rc, rd);		;Jump to flash if A = B.	Assumed that offset = 1.	
Var2 is a variable (ra, rb, rc, rd);				
Address is a label which specifies the offset.				

DATA TRANSFER AND ARITHMETIC INSTRUCTIONS

INSTRUCTION SYNTAX	FUNCTION	EXAMPLE	16-BIT ASSEMBLED BIT SEQUENCE	ASSEMBLED CODE HEX
ld var, value	Assigns a value to a variable.	ld ra, 10	1001 0000 0000 1010	900A
Var is a variable (ra, rb, rc);		;Variable A = 10.		
Value is a constant (0 to 255 or 0 to FFh).				
add var, value	Add the 8-bit value to the variable value.	add ra, 30	1001 0001 0001 1110	911E
Var is a variable (ra, rb, rc);		;A = A + 30.		
Value is a constant (0 to 255 or 0 to FFh).				
add var1, var2, var3	Add the value of var3 to the value of var2 and store the result in var1.	add ra, rc, rd	1001 0011 0000 1010	930B
Var1 is a variable (ra, rb, rc);		;A = C + D.		
Var2 is a variable (ra, rb, rc, rd);				
Var3 is a variable (ra, rb, rc, rd);				
sub var, value	Subtract the 8-bit value from the variable value.	sub ra, 30	1001 0010 0001 1110	921E
Var is a variable (ra, rb, rc);		;A = A - 30.		
Value is a constant (0 to 255 or 0 to FFh).				

sub var1, var2, var3	Subtract the value of var3 from the value of var2 and store the result in var1.	sub ra, rc, rd	1001 0011 0001 1011	931B
Var1 is a variable (ra, rb, rc);		;A = C - D		
Var2 is a variable (ra, rb, rc, rd);				
Var3 is a variable (ra, rb, rc, rd);				

Programming Examples

This section gives practical programming examples. A series of programs introduce the main features of the LP55231 chip, and the example programs are easy to tailor to the specific needs of end application.

EXAMPLE 1: CONTROLLING MULTIPLE LEDS WITH ONE ENGINE

The example below is basically the program shown in the Example Code of LP55231 Program before (simple LP55231 program), but here the mapping table is used to establish engine1-to-LEDs connection. In the mapping table '0' means that the LED isn't connected to the engine; '1' means that the LED is connected to the engine. In the example program below bits 6 to 8 are set to ‘1’ so that outputs 7, 8, and 9 are mapped to the engine 1.

LED Mapping Chart

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Output	GPO	N/A	N/A	N/A	N/A	N/A	N/A	D9	D8	D7	D6	D5	D4	D3	D2	D1

mapping_table:	dw 0000000111000000b	;Red LEDs on the evaluation board.
		;'1' = LED is mapped; '0'= LED isn't mapped.
.segment program1		;Begin of a segment.
	mux_map_start mapping_table	;Mapping table start address in the memory.
		;The first row of the table will be activated
loop1:	set_pwm FFh	;Beginning of a loop, set PWM to full scale.
	wait 0.48	;Wait for 0.48 seconds.
	set_pwm 00h	;Set PWM to 0%.
	wait 0.48	;Wait for 0.48 seconds.
	branch 0,loop1	;Endless loop.

EXAMPLE 2: FADE-IN/FADEOUT EFFECTS AND CHANGING MAPPING OF LEDS

In this example ramp instruction is used to produce fade-in/fade-out effects. Note: The use of a logarithmic PWM profile with ramp instruction ensures the light and color changes appear smooth to the human eye. The same effect is repeated for all the LEDs by changing the engine1-to-LED mapping. After that, the intensity of the LEDs is increased gradually, and finally all the LEDs are set OFF.

row1:	dw 0000000000000001b	;Map green LED = D1 on the eval. board.
	dw 0000000000000010b	;Map blue LED = D2 on the eval. board.
	dw 0000000001000000b	;Map red LED = D7 on the eval. board.
	dw 0000000000000100b	;Map green LED = D3 on the eval. board.
	dw 0000000000001000b	;Map blue LED = D4 on the eval. board.
	dw 0000000010000000b	;Map red LED = D8 on the eval. board.
	dw 0000000000010000b	;Map green LED = D5 on the eval. board.
	dw 0000000000100000b	;Map blue LED = D6 on the eval. board.
row9:	dw 0000000100000000b	;Map red LED = D9 on the eval. board.
row10:	dw 0000000111111111b	;Map all LEDs on the eval. board.
.segment program1		;Program for engine 1.
	mux_map_start row1	;Map the first LED.
	mux_ld_end row9	;End address of the mapping data table.
loop1:	ramp 1.0, 255	;Increase PWM 0->100% in 1 second.
	ramp 1.5, -255	;Decrease PWM 100->0% in 1.5 seconds.
	wait 0.4;	;Wait for 0.4 seconds.
	mux_map_next	;Set the next row active in the mapping table.
	branch 8,loop1	;Loop 8 times

loop2:	ramp 0.5, 128	;Increase PWM by 128 steps in 0.5 second.
	mux_map_next	;Set the next row active in the mapping table.
		;Note roll over of the mapping.
	branch 17, loop2	;Loop 17 times.
	mux_map_addr row10	;Set row10 active (all LEDs mapped).
	set_pwm 0;	;Set all the LEDs OFF.
rst		;Reset program counter and start the program again.
.segment program2		
		;Program for engine 2 (empty).
rst		
.segment program3		
		;Program for engine 3(empty).
rst		

EXAMPLE 3: USING MORE THAN ONE ENGINE AND NESTED LOOPS (LOOP INSIDE A LOOP)

This program below shows how to use all the three engines simultaneously. The engines are used to create a police beacon-type light effect. Hint: To see the effect of separate engines, start/stop engines separately using Engine 1, Engine 2 and Engine 3 controls instead of the Master control.

blue1:	dw 0000000000000010b	;Map blue LED on D2.
blue2:	dw 0000000000100000b	;Map blue LED on D6.
blue3:	dw 0000000000001000b	;Map blue LED on D4.
red_led:	dw 0000000010000000b	;Map red LED on D8.
.segment program1		;Program for blue LEDs on D2 and D6.
	mux_map_start blue1	;Mapping table start address.
	mux_ld_end blue2	;Mapping table end address.
loop2:		
loop1:	set_pwm 255	
	wait 0.1	
	set_pwm 0	
	wait 0.05	
	branch 1, loop1	
	wait 0.2	
	mux_map_next	
	branch 3, loop2	;Note nested loop (loop1 is within loop2).
loop4:		
loop3:	set_pwm 255	
	wait 0.05	
	set_pwm 0	
	wait 0.05	
	branch 5, loop3	
	mux_map_next	
	branch 3, loop4	
	rst	
.segment program2		;Program for red LED on D8.
	mux_map_addr red_led	;Red LED mapping.
	wait 0.45	
	ramp 0.5, 255	;PWM 0%->100% in 0.5 second.
	ramp 1, -255	;PWM 100%->0% in 1 second.

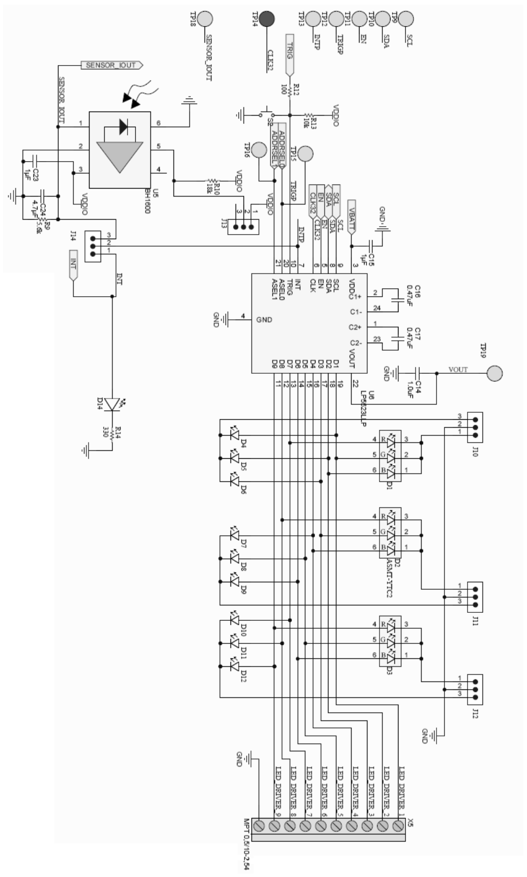
	rst	
.segment program3		;Program for blue LED on D4.
	mux_map_start blue3	;Mapping table start address.
	set_pwm 255	
	wait 0.05	
	set_pwm 0	
	wait 0.2	
	set_pwm 255	
	wait 0.05	
	set_pwm 0	
	wait 0.45	
	rst	

EXAMPLE 4: TRIGGERS AND INTERRUPT

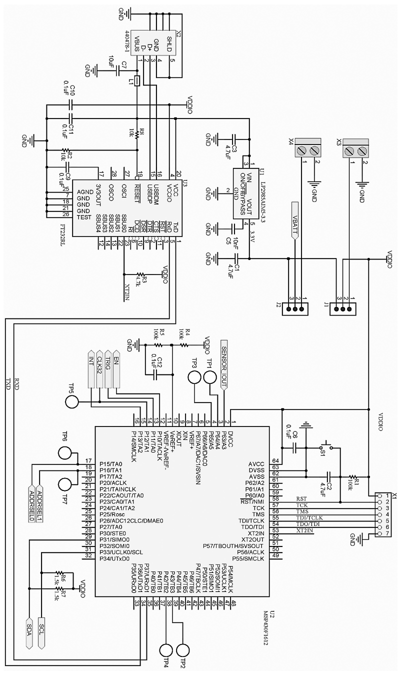
This program shows how to use triggers and the interrupt. Engine 1 sends a trigger to Engine 2 when it has set a LED to 100% PWM – Engine 2 fades the LED out to 0% PWM. Engine 1 sends an interrupt when it has finished loop1 and waits trigger from Engine 2 AND an external trigger. Pushing the button on the evaluation board causes an external trigger and the sequence starts over. Use RD button on the Status/Interrupt section (see Figure 5) to clear the interrupt.

row1:	dw 0000000000000001b	;Map green LED = D1 on the eval. board.
	dw 0000000000000010b	;Map blue LED = D2 on the eval. board.
	dw 0000000001000000b	;Map red LED = D7 on the eval. board.
	dw 0000000000000100b	;Map green LED = D3 on the eval. board.
	dw 0000000000001000b	;Map blue LED = D4 on the eval. board.
	dw 0000000010000000b	;Map red LED = D8 on the eval. board.
	dw 0000000000010000b	;Map green LED = D5 on the eval. board.
	dw 0000000000100000b	;Map blue LED = D6 on the eval. board.
row9:	dw 0000000100000000b	;Map red LED = D9 on the eval. board.
.segment program1		;Program for engine 1.
	mux_map_start row1	;Map the first LED.
	mux_ld_end row9	;End address of the mapping data table.
loop1:	ramp 1.0, 255	;Increase PWM 0->100% in 1 second.
	trigger s{2}	;Send trigger to engine2
	mux_map_next	;Set the next row active in the mapping table.
	branch 8,loop1	;Loop 8 times.
	int	;Send an interrupt.
	trigger w{2|e}	;Wait trigger from engine 2 and external trigger.
	rst	;Reset program counter and start the program again.
.segment program2		;Program for engine 2.
	mux_map_start row1	;Map the first LED.
	mux_ld_end row9	;End address of the mapping data table.
loop2:	trigger w{1}	;Wait for trigger from engine1.
	ramp 1.0,-255	;Decrease PWM 100->0% in 1 second.
	mux_map_next	;Set the next row active in the mapping table.
	branch 8,loop2	;Loop 8 times.
	trigger s{1}	;Send trigger to engine1.
	rst	
.segment program3		

		;Program for engine 3(empty).
rst		



Schematic of the Evaluation Board Application Side



Schematic of the Evaluation Board USB Side

Bill of Materials

Designator	Qty.	Part Number	Description	Value	Footprint
U6	1	LP55231SQ	Lighting management unit		24-Pin WQFN
U3	1	FT232RL	USB to UART		28-SSOP
U2	1	MSP430F1612IPM	Micro controller		64-LQFP
U1	1	LP2985AIM5-3.3	LDO regulator		0603L
U5	1	BH1600FVC-TR	Light Sensor		
D1, D2, D3	3	ASMT-YTC2-0AA02	RGB LED		6-PLCC
D14	2	HSMS-A100-J00J1	Red LED		2-PLLC
D4, D5, D6, D7, D8, D9, D10, D11, D12	9	CLM3C-WKW-CWBYA453	White LED		2-PLCC
C1, C2, C3	3	LMK212B7475KG-T	Ceramic capacitor	4.7 µF, 10V	0805
C5	1	C0603C103J5RACTU	Ceramic capacitor	10 nF, 50V	0603
C6, C9, C10, C11, C12	5	C0603C104K8RACTU	Ceramic capacitor	0.1 µF, 10V	0603
C7	1	C2012Y5V1A106Z	Ceramic capacitor	10 µF, 10V	0805
C15, C14	2	LMK105BJ105KV-F	Ceramic capacitor	1 µF, 10V	0402
C16, C17	2	JMK105BJ474KV-F	Ceramic capacitor	0.47 µF, 6.3V	0402
C23	1	C0603C105Z8VACTU	Ceramic capacitor	1 µF, 10V	0603
C24	1	C0603C475K8PACTU	Ceramic capacitor	4.7 µF, 10V	0603
R1, R4, R5	3	ERJ-3GEYJ104V	Resistor	100k	0603
R2, R8, R13	3	ERJ-3GEYJ103V	Resistor	10k	0603
R3	1	RC0603JR-074K7L	Resistor	4.7k	0603
R9	1	ERJ-3GEYJ562V	Resistor	5.6k	0603
R10	1	ERJ-3GEYJ183V	Resistor	18k	0603
R12	1	ERJ-3GEYJ101V	Resistor	100	0603
R14	2	ERJ-3GEYJ331V	Resistor	330	0603
R6, R7	2	ERJ-3GEYJ152V	Resistor	1.5k	0603
L1	1	MMZ2012S400A	Ferrite		0805

Revision History

Changes from A Revision (November 2014) to B Revision **Page**

- Changed "(LED1 = D4, LED2 = D5, LED7 = D6)" to "(LED1 = D4, LED2 = D5, LED3 = D6)" 5
- Changed "(LED3 = D7, LED2 = D8, LED8 = D9)" to "(LED4 = D7, LED5 = D8, LED6 = D9)" 5
- Changed "(LED5 = D10, LED6 = D1, LED9 = D12)" to "(LED7 = D10, LED8 = D11, LED9 = D12)" 5

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

STANDARD TERMS AND CONDITIONS FOR EVALUATION MODULES

1. *Delivery:* TI delivers TI evaluation boards, kits, or modules, including any accompanying demonstration software, components, or documentation (collectively, an “EVM” or “EVMs”) to the User (“User”) in accordance with the terms and conditions set forth herein. Acceptance of the EVM is expressly subject to the following terms and conditions.

   1.1 EVMs are intended solely for product or software developers for use in a research and development setting to facilitate feasibility evaluation, experimentation, or scientific analysis of TI semiconductors products. EVMs have no direct function and are not finished products. EVMs shall not be directly or indirectly assembled as a part or subassembly in any finished product. For clarification, any software or software tools provided with the EVM (“Software”) shall not be subject to the terms and conditions set forth herein but rather shall be subject to the applicable terms and conditions that accompany such Software

   1.2 EVMs are not intended for consumer or household use. EVMs may not be sold, sublicensed, leased, rented, loaned, assigned, or otherwise distributed for commercial purposes by Users, in whole or in part, or used in any finished product or production system.

2 *Limited Warranty and Related Remedies/Disclaimers*:

   2.1 These terms and conditions do not apply to Software. The warranty, if any, for Software is covered in the applicable Software License Agreement.

   2.2 TI warrants that the TI EVM will conform to TI's published specifications for ninety (90) days after the date TI delivers such EVM to User. Notwithstanding the foregoing, TI shall not be liable for any defects that are caused by neglect, misuse or mistreatment by an entity other than TI, including improper installation or testing, or for any EVMs that have been altered or modified in any way by an entity other than TI. Moreover, TI shall not be liable for any defects that result from User's design, specifications or instructions for such EVMs. Testing and other quality control techniques are used to the extent TI deems necessary or as mandated by government requirements. TI does not test all parameters of each EVM.

   2.3 If any EVM fails to conform to the warranty set forth above, TI's sole liability shall be at its option to repair or replace such EVM, or credit User's account for such EVM. TI's liability under this warranty shall be limited to EVMs that are returned during the warranty period to the address designated by TI and that are determined by TI not to conform to such warranty. If TI elects to repair or replace such EVM, TI shall have a reasonable time to repair such EVM or provide replacements. Repaired EVMs shall be warranted for the remainder of the original warranty period. Replaced EVMs shall be warranted for a new full ninety (90) day warranty period.

3 *Regulatory Notices:*

   3.1 *United States*

      3.1.1 *Notice applicable to EVMs not FCC-Approved:*

      This kit is designed to allow product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and software developers to write software applications for use with the end product. This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter.

      3.1.2 *For EVMs annotated as FCC – FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant:*

      CAUTION

      This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

      Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

      FCC Interference Statement for Class A EVM devices

      NOTE: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Interference Statement for Class B EVM devices

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- *Reorient or relocate the receiving antenna.*
- *Increase the separation between the equipment and receiver.*
- *Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.*
- *Consult the dealer or an experienced radio/TV technician for help.*

3.2 *Canada*

3.2.1 *For EVMs issued with an Industry Canada Certificate of Conformance to RSS-210*

Concerning EVMs Including Radio Transmitters:

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concernant les EVMs avec appareils radio:

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concerning EVMs Including Detachable Antennas:

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication. This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante. Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur

3.3 *Japan*

3.3.1 *Notice for EVMs delivered in Japan:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_01.page 日本国内に輸入される評価用キット、ボードについては、次のところをご覧ください。
http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_01.page

3.3.2 *Notice for Users of EVMs Considered "Radio Frequency Products" in Japan:* EVMs entering Japan may not be certified by TI as conforming to Technical Regulations of Radio Law of Japan.

If User uses EVMs in Japan, not certified to Technical Regulations of Radio Law of Japan, User is required by Radio Law of Japan to follow the instructions below with respect to EVMs:

1. Use EVMs in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
2. Use EVMs only after User obtains the license of Test Radio Station as provided in Radio Law of Japan with respect to EVMs, or
3. Use of EVMs only after User obtains the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to EVMs. Also, do not transfer EVMs, unless User gives the same notice above to the transferee. Please note that if User does not follow the instructions above, User will be subject to penalties of Radio Law of Japan.

【無線電波を送信する製品の開発キットをお使いになる際の注意事項】 開発キットの中には技術基準適合証明を受けていないものがあります。 技術適合証明を受けていないもののご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
2. 実験局の免許を取得後ご使用いただく。
3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。 日本テキサス・ インスツルメンツ株式会社
東京都新宿区西新宿６丁目２４番１号
西新宿三井ビル

3.3.3 *Notice for EVMs for Power Line Communication:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_02.page
電力線搬送波通信についての開発キットをお使いになる際の注意事項については、次のところをご覧ください。http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_02.page

4 *EVM Use Restrictions and Warnings:*

4.1 EVMS ARE NOT FOR USE IN FUNCTIONAL SAFETY AND/OR SAFETY CRITICAL EVALUATIONS, INCLUDING BUT NOT LIMITED TO EVALUATIONS OF LIFE SUPPORT APPLICATIONS.

4.2 User must read and apply the user guide and other available documentation provided by TI regarding the EVM prior to handling or using the EVM, including without limitation any warning or restriction notices. The notices contain important safety information related to, for example, temperatures and voltages.

4.3 *Safety-Related Warnings and Restrictions:*

4.3.1 User shall operate the EVM within TI's recommended specifications and environmental considerations stated in the user guide, other available documentation provided by TI, and any other applicable requirements and employ reasonable and customary safeguards. Exceeding the specified performance ratings and specifications (including but not limited to input and output voltage, current, power, and environmental ranges) for the EVM may cause personal injury or death, or property damage. If there are questions concerning performance ratings and specifications, User should contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may also result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM user guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, even with the inputs and outputs kept within the specified allowable ranges, some circuit components may have elevated case temperatures. These components include but are not limited to linear regulators, switching transistors, pass transistors, current sense resistors, and heat sinks, which can be identified using the information in the associated documentation. When working with the EVM, please be aware that the EVM may become very warm.

4.3.2 EVMs are intended solely for use by technically qualified, professional electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems, and subsystems. User assumes all responsibility and liability for proper and safe handling and use of the EVM by User or its employees, affiliates, contractors or designees. User assumes all responsibility and liability to ensure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard. User assumes all responsibility and liability for any improper or unsafe handling or use of the EVM by User or its employees, affiliates, contractors or designees.

4.4 User assumes all responsibility and liability to determine whether the EVM is subject to any applicable international, federal, state, or local laws and regulations related to User's handling and use of the EVM and, if applicable, User assumes all responsibility and liability for compliance in all respects with such laws and regulations. User assumes all responsibility and liability for proper disposal and recycling of the EVM consistent with all applicable international, federal, state, and local requirements.

5. *Accuracy of Information:* To the extent TI provides information on the availability and function of EVMs, TI attempts to be as accurate as possible. However, TI does not warrant the accuracy of EVM descriptions, EVM availability or other information on its websites as accurate, complete, reliable, current, or error-free.

6. *Disclaimers:*

   6.1 EXCEPT AS SET FORTH ABOVE, EVMS AND ANY WRITTEN DESIGN MATERIALS PROVIDED WITH THE EVM (AND THE DESIGN OF THE EVM ITSELF) ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." TI DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING SUCH ITEMS, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

   6.2 EXCEPT FOR THE LIMITED RIGHT TO USE THE EVM SET FORTH HEREIN, NOTHING IN THESE TERMS AND CONDITIONS SHALL BE CONSTRUED AS GRANTING OR CONFERRING ANY RIGHTS BY LICENSE, PATENT, OR ANY OTHER INDUSTRIAL OR INTELLECTUAL PROPERTY RIGHT OF TI, ITS SUPPLIERS/LICENSORS OR ANY OTHER THIRD PARTY, TO USE THE EVM IN ANY FINISHED END-USER OR READY-TO-USE FINAL PRODUCT, OR FOR ANY INVENTION, DISCOVERY OR IMPROVEMENT MADE, CONCEIVED OR ACQUIRED PRIOR TO OR AFTER DELIVERY OF THE EVM.

7. *USER'S INDEMNITY OBLIGATIONS AND REPRESENTATIONS.* USER WILL DEFEND, INDEMNIFY AND HOLD TI, ITS LICENSORS AND THEIR REPRESENTATIVES HARMLESS FROM AND AGAINST ANY AND ALL CLAIMS, DAMAGES, LOSSES, EXPENSES, COSTS AND LIABILITIES (COLLECTIVELY, "CLAIMS") ARISING OUT OF OR IN CONNECTION WITH ANY HANDLING OR USE OF THE EVM THAT IS NOT IN ACCORDANCE WITH THESE TERMS AND CONDITIONS. THIS OBLIGATION SHALL APPLY WHETHER CLAIMS ARISE UNDER STATUTE, REGULATION, OR THE LAW OF TORT, CONTRACT OR ANY OTHER LEGAL THEORY, AND EVEN IF THE EVM FAILS TO PERFORM AS DESCRIBED OR EXPECTED.

8. *Limitations on Damages and Liability:*

   8.1 *General Limitations*. IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF THESE TERMS ANDCONDITIONS OR THE USE OF THE EVMS PROVIDED HEREUNDER, REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF REMOVAL OR REINSTALLATION, ANCILLARY COSTS TO THE PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, RETESTING, OUTSIDE COMPUTER TIME, LABOR COSTS, LOSS OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, LOSS OF USE, LOSS OF DATA, OR BUSINESS INTERRUPTION. NO CLAIM, SUIT OR ACTION SHALL BE BROUGHT AGAINST TI MORE THAN ONE YEAR AFTER THE RELATED CAUSE OF ACTION HAS OCCURRED.

   8.2 *Specific Limitations.* IN NO EVENT SHALL TI'S AGGREGATE LIABILITY FROM ANY WARRANTY OR OTHER OBLIGATION ARISING OUT OF OR IN CONNECTION WITH THESE TERMS AND CONDITIONS, OR ANY USE OF ANY TI EVM PROVIDED HEREUNDER, EXCEED THE TOTAL AMOUNT PAID TO TI FOR THE PARTICULAR UNITS SOLD UNDER THESE TERMS AND CONDITIONS WITH RESPECT TO WHICH LOSSES OR DAMAGES ARE CLAIMED. THE EXISTENCE OF MORE THAN ONE CLAIM AGAINST THE PARTICULAR UNITS SOLD TO USER UNDER THESE TERMS AND CONDITIONS SHALL NOT ENLARGE OR EXTEND THIS LIMIT.

9. *Return Policy.* Except as otherwise provided, TI does not offer any refunds, returns, or exchanges. Furthermore, no return of EVM(s) will be accepted if the package has been opened and no return of the EVM(s) will be accepted if they are damaged or otherwise not in a resalable condition. If User feels it has been incorrectly charged for the EVM(s) it ordered or that delivery violates the applicable order, User should contact TI. All refunds will be made in full within thirty (30) working days from the return of the components(s), excluding any postage or packaging costs.

10. *Governing Law:* These terms and conditions shall be governed by and interpreted in accordance with the laws of the State of Texas, without reference to conflict-of-laws principles. User agrees that non-exclusive jurisdiction for any dispute arising out of or relating to these terms and conditions lies within courts located in the State of Texas and consents to venue in Dallas County, Texas. Notwithstanding the foregoing, any judgment may be enforced in any United States or foreign court, and TI may seek injunctive relief in any United States or foreign court.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		