Application Report
SNVA205A–April 2008–Revised April 2013

# AN-1562 LP5521 Programming Considerations

**ABSTRACT**

This application report describes LP5521 programming commands and provides examples for the user.

**Contents**



**List of Figures**

# 1 Introduction

This application report describes LP5521 programming commands with examples. Most of the programs are presented with command compiler syntax. The Command compiler is described at the end of this document in Appendix A. Command compiler software is available with the evaluation kit.

# 2 Wait/Ramp Command

Ramp command generates either increasing or decreasing PWM ramp, which execution time and number of steps can be defined. In one ramp command PWM value can be incremented or decremented up to 128 steps from the present PWM value. Maximum PWM value is 255 that can be interpreted, that channel's current source is constantly active. Ramp command maximum execution time $t_{MAX}$ = (1000 ms x number of steps) - 1 ms.

Figure 1 illustrates ramp and wait command usage. The program has been made with command compiler. First ramp command increase PWM value from 0 to 100 in 150 ms. Second ramp command will increase PWM value from 100 to 200 in 150 ms. PWM value is kept constant during the next 300 ms wait cycle. Program then continues by ramping down PWM from 200 to 100 in 50ms. During the 450 ms wait cycle PWM value is kept constant and the last command decrease PWM value from 100 to 0 during 100 ms.

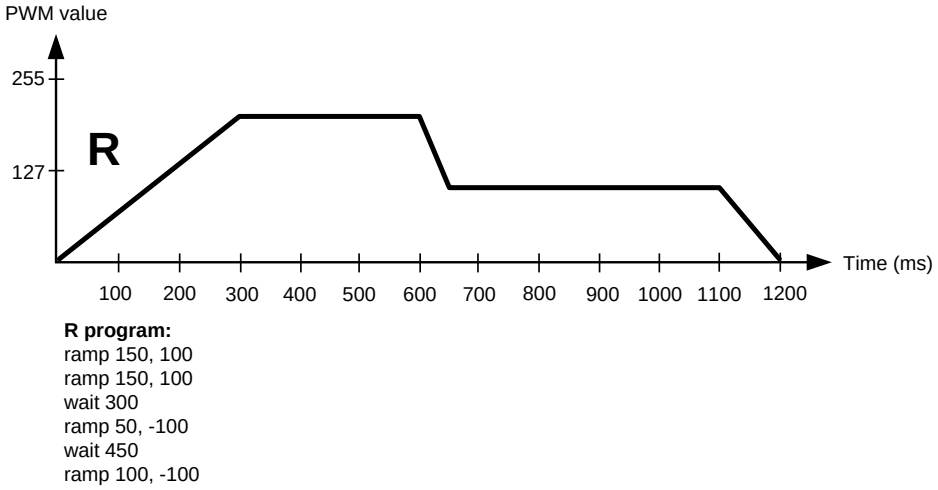



**Figure 1. Ramp and Wait Command Example**

Combining ramp and wait command can be used to reduce number of program commands. Figure 2 illustrates this situation. Ramp begins from PWM value 231 and saturates to 255 in 100 ms. After PWM is saturated to maximum, ramp command changes to wait command for the rest 450 ms. Falling ramp saturates similar way to 0 PWM value. After the saturation, the rest of the command execution will be wait time.

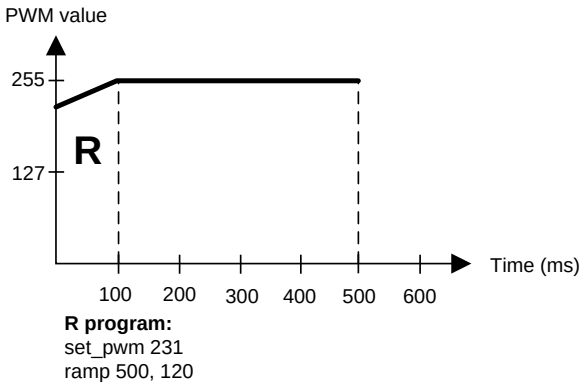



**Figure 2. Combined Ramp and Wait Command Example**

Wait command maximum time is 999 ms on the command compiler and if longer wait time is required, then branching (looping) can be used. Branch command is explained in this application note. Also ramp command can produce the desired wait time if PWM level during wait cycle is either 0 or 255. Ramp command maximum execution time $t_{MAX}$ = (1000 ms x number of steps) - 1 ms on the command compiler, so maximum wait time with ramp command is 127 999 ms. On Figure 3, PWM is first set to 255 and 1200 ms wait cycle is produced with ramp command that increases PWM level by 127 in 1200 ms. Since PWM is already at maximum command will only produce wait time. Similar way falling ramp can be used as wait command if PWM level is 0.

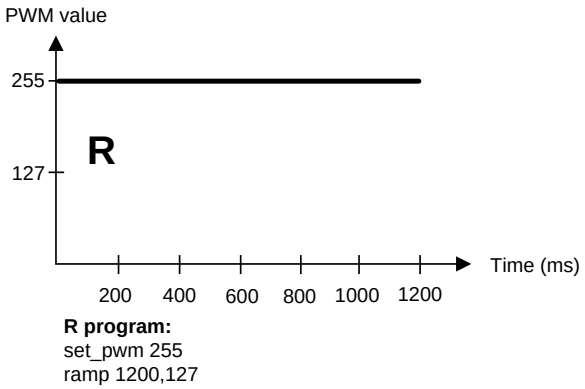



**Figure 3. Using Ramp Command as Wait**

## 3 Set PWM Command

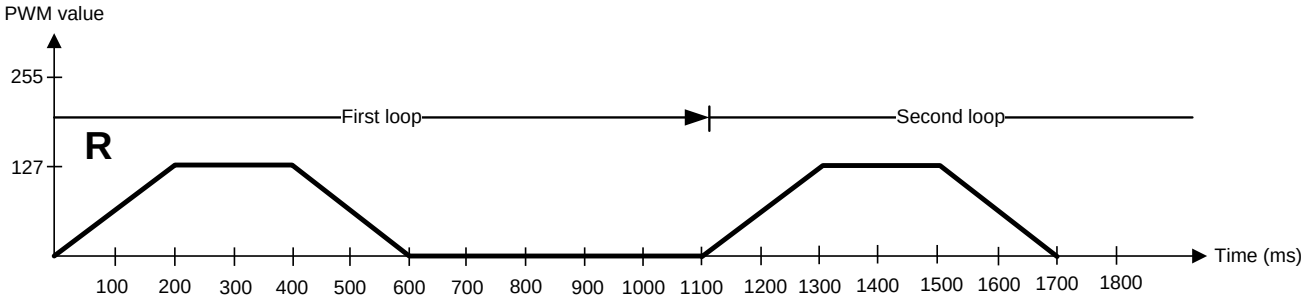
Set_pwm command adjusts PWM level with 8-bit control from 0 to 255. PWM level is adjusted to new value in 0.488 ms (typ.).

## 4 Go To Start Command

Go to start command resets program counter and program execution will be started from the beginning of the program. Go to start can be interpreted as infinite loop. By default, all program memory locations are reset to zeros, which implies to Go to the start command. In command compiler syntax this command is Start. If program memory is fully occupied, and last command is ramp, wait, set_pwm or trigger, program execution will be continued from the beginning of the program.

## 5 Branch Command

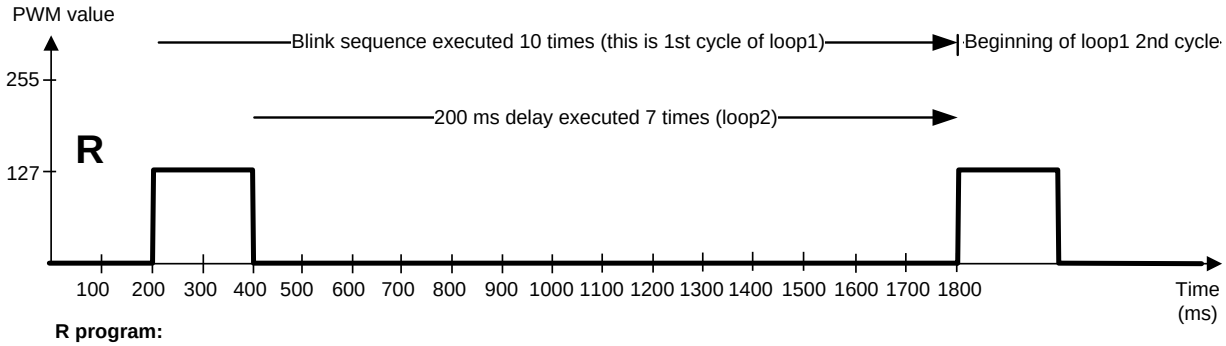
Branch command can be used to loop certain sequences in program. Figure 4 illustrates branch command. Program ramps up PWM level from 0 to 127 in 200 ms. Then PWM level is kept constant 200 ms, and then ramped down to 0 in 200 ms. PWM is kept at 0 during the next 500 ms. The whole sequence is executed 6 times. Loop start location is defined with label (loop1).




**R program:**
loop1:
ramp 200, 127
wait 200
ramp 200, -127
wait 500
branch 5, loop1

**Figure 4. Branch Command Example**

The maximum loop count is 63 in one branch command, but LP5521 supports loop inside loop (nested) looping. Nested looping is illustrated in Figure 5. 1600 ms blinking cycle is repeated 10 times. LED is active 200 ms during the cycle and rest of the cycle 1400 ms is wait time. The program has two loops loop1 and loop2. Loop1 repeats the whole sequence 10 times and loop2 creates 1400 ms wait time.




**R program:**
wait 200
loop1:
set_pwm 127
wait 200
set_pwm 0
loop2:
wait 200
branch 6, loop2
branch 9, loop1

**Figure 5. Branch Command Example**

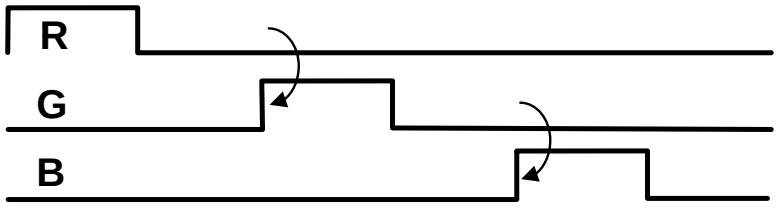
## 6 End Command

End command stops program execution. There are two parameters that can be defined with end command: interrupt and reset. Interrupt can be used to notify processor that program execution is at the end. Interrupt pulls INT pin low, and status bits in register address 0CH informs which channel (R, G, B) has caused the interrupt. Interrupt pin state and status bits will be cleared when status register 0CH is read. Reset parameter resets program counter to 0, changes channel to hold from run mode, and sets PWM output to 0. If no parameters are defined, channel will be changed to hold mode and PWM value will remain.

## 7 Triggering

Triggering is efficient way of controlling program execution between LP5521 channels (R,G,B) or multiple LP5521 devices in the same application. Trigger signal can also be connected to processor.

Figure 6 describes the basic triggering concept. Each channel (R,G,B) have identical 100 ms LED pulse and 100 ms delay period after the pulse. R channel begins the sequence by generating pulse and at the end sends trigger to G channel. G channel continues program execution and at the end of the program sends trigger to B channel program.



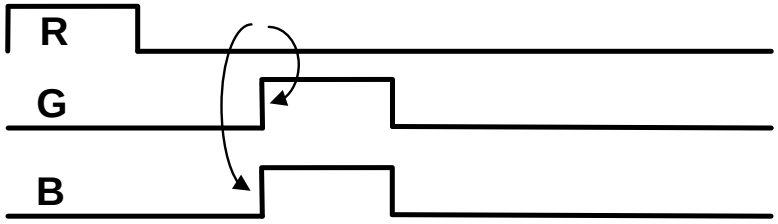

**R program:**
set_pwm 255
wait 100
set_pwm 0
wait 100
trigger sg
end

**G program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
wait 100
trigger sb
end

**B program:**
trigger wg
set_pwm 255
wait 100
set_pwm 0
end

**Figure 6. Basic Triggering Example**

One channel can send multiple triggers in one command. Figure 7 shows that R channels triggers both G and B channels. G and B channel programs are identical.




**R program:**
set_pwm 255
wait 100
set_pwm 0
wait 100
trigger sgb
end

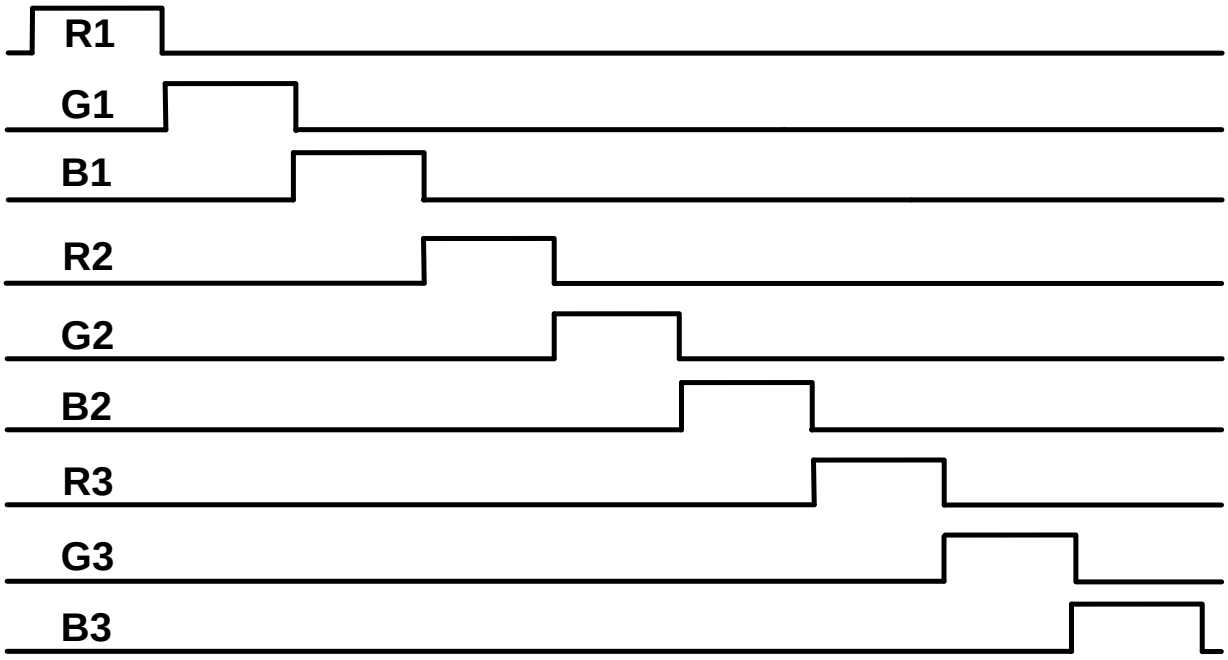
**G program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
end

**B program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
end

**Figure 7. Sending Multiple Triggers Example**

Figure 8 describes how to control multiple LP5521 devices through internal and external triggering. The example consists of three LP5521 devices that can produce LED circle. Only one LED is active at certain times and, therefore, the human eye sees a circulating bright LED. Figure 8 describes LED waveforms and program code. Figure 9 presents how the three LP5521s are connected together.

In order to start the sequence correctly, device1 needs to be configured to run program mode last. Each channel (R,G,B) programs have identical 100 ms LED pulse and the start of the pulse is controlled by triggering. R1 channel (device 1) will start sequence and after pulse has been sent, R1 triggers channel G1 (device 1) to proceed. R1 program will then wait for two external triggers from device 2 and device 3. G1 program generates pulse and triggers B1 program. After B1 program has been executed external trigger will be sent. Devices 2 and 3 receive external trigger which causes device 2 LED sequence to start. At the end of the program, device 2 will send external trigger to devices 1 and 3. Device 3 has now received two external triggers and starts LED sequence. At the end of the program device 3 sends external trigger to device 1 and device 2. The whole LED sequence will start from the beginning.




**R1 program:**
wait 30
set_pwm 255
wait 100
set_pwm 0
trigger sg,we
trigger we

**G1 program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
trigger sb

**B1 program:**
trigger wg
set_pwm 255
wait 100
set_pwm 0
trigger se

**R2 program:**
trigger we
set_pwm 255
wait 100
set_pwm 0
trigger sg,we

**G2 program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
trigger sb

**B2 program:**
trigger wg
set_pwm 255
wait 100
set_pwm 0
trigger se

**R3 program:**
trigger we
trigger we
set_pwm 255
wait 100
set_pwm 0
trigger sg

**G3 program:**
trigger wr
set_pwm 255
wait 100
set_pwm 0
trigger sb

**B3 program:**
trigger wg
set_pwm 255
wait 100
set_pwm 0
trigger se

**Figure 8. External and Internal Triggering With Three LP5521 Devices**

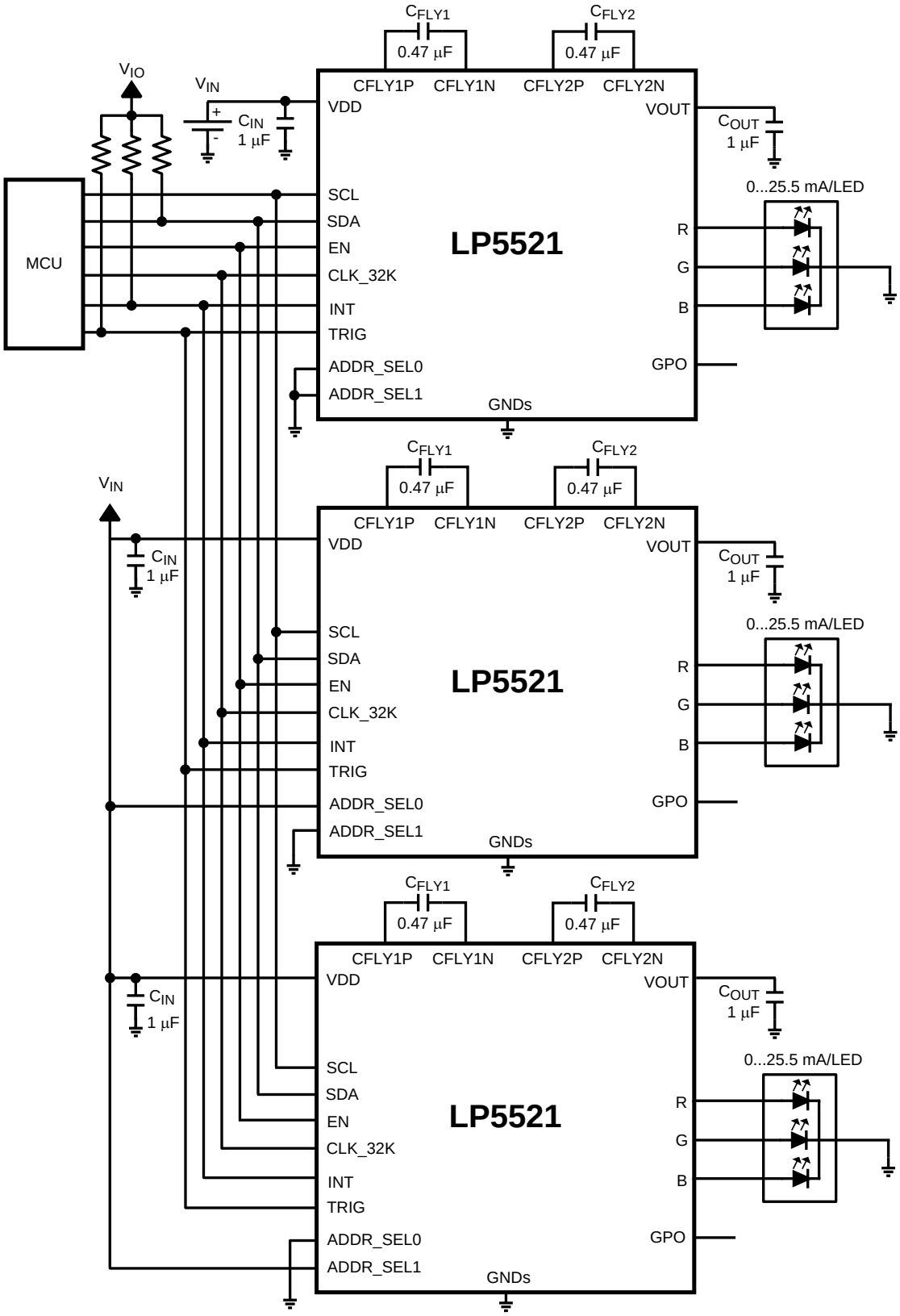



**Figure 9. Connecting Three LP5521 Devices Together**

## Appendix A Command Compiler

Command compiler is used to write LED sequences for the LP5521. Command compiler is a simple to use Windows program with graphic user interface.

User can write own memory files by using text editor (.src). Command compiler translates ASCII memory files in to binary file (.bin), or hex file (.hex). In the Format menu user can select between .bin and .hex formats. To start command compiler, double click the compiler icon (compiler.exe). Source file can be opened from File/Open menu. Source files have .src extension. When source tab has been selected source code is visible and it can be modified, compiled and saved. Below is explanation of the RGB driver commands. Program can be compiled to binary or hexadecimal formats with compile menu. After compilation command list can be seen by clicking List tab. Compilation generates binary/hex file and list file. Compiled file has 16 commands for each channel, which is a total of 48 commands. File represents the whole SRAM program memory content.

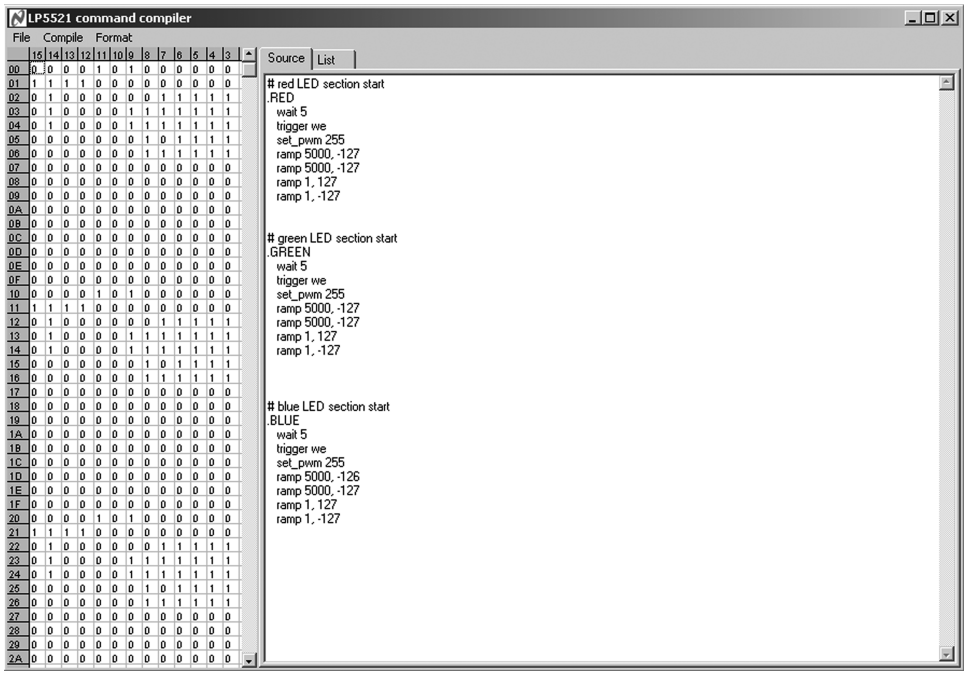



**Figure 10. Command Compiler User Interface**

## Appendix B Syntax

**Comment** – Line starting with “#” symbol.

Example: # red LED section start

**Sections** – Red, green or blue LED section starts with: .RED, .GREEN or .BLUE.

Example: .RED

**Labels** – Any words with colon (:) as ending symbol.

Example: label1:

## Appendix C Commands

### Ramp

Ramp command generates a PWM ramp from current value. Ramp command has two parameters – first is time in milliseconds (floating point format, maximum execution time $t_{MAX}$ = (1000 ms x number of steps) - 1 ms) and second is number of steps (positive or negative, integer 2-128) separated with comma.

Example: ramp 20.5,6

### Wait

With wait command program execution stops for time defined. Command has one parameter, time in milliseconds (floating point format, maximum 999).

Example: wait 50.5

### Branch

Branch command loads step number to program counter. Branch command has two parameters, loop counter (integer 0-63, 0 means infinite loop) and label separated with comma. Label must be predefined before using in a branch command. The following example loops 5 times commands between label1 and branch command:

Example: label1: … branch 5,label1

### Set_PWM

Set_pwm command sets PWM output value. Command has one parameter, PWM value (integer 0-255).

Example: set_pwm 23

### Start

(Go to) Start command resets program counter and continues executing from the beginning of section. No parameters used.

Example: start

### Trigger

Trigger command sets wait or send trigger. Command has two parameters, wait trigger channel and send trigger channel. Channels are defined as: r = red, g = green, b = blue, e = external.

Examples:

trigger sr => (Send trigger to red channel)

trigger sg => (Send trigger to green channel)

trigger sb => (Send trigger to blue channel)

trigger se => (Send trigger to external channel)

trigger sgb => (Send trigger to green and blue channel)

trigger wr => (Wait trigger from red channel)

trigger sr,wb => (Send trigger to red channel and wait trigger from blue channel)

### End

End command ends program execution. Also interrupt signal can be send or program counter can be reset. Command can have up to two parameters. I = interrupt send, R = reset program counter.

Examples:

end I

end R

end R,I

```
end I,R (same as earlier)
end
```

## Appendix D Errors

If there is an error during compilation error message is generated. Error messages are as follows:

1 = color section error

2 = syntax error

3 = ramp parameter error

4 = SRAM overflow error

6 = ramp step error

7 = branch error

9 = set_pwm parameter error

11 = wait parameter error

## Appendix E Files

Source file has .src extension. During compilation listing file with .lst extension and binary with .bin or hex file with .hex extension will be generated. File name is the same as for source file name.

## Appendix F Example Command File

```
# red LED section start
.RED
ramp 20.5,6
ramp 10,-15
wait 10
# green LED section start
.GREEN
ramp 10,-15
start
# blue LED section start
.BLUE
ramp 10,15
end R
```

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

| **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 | | |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265