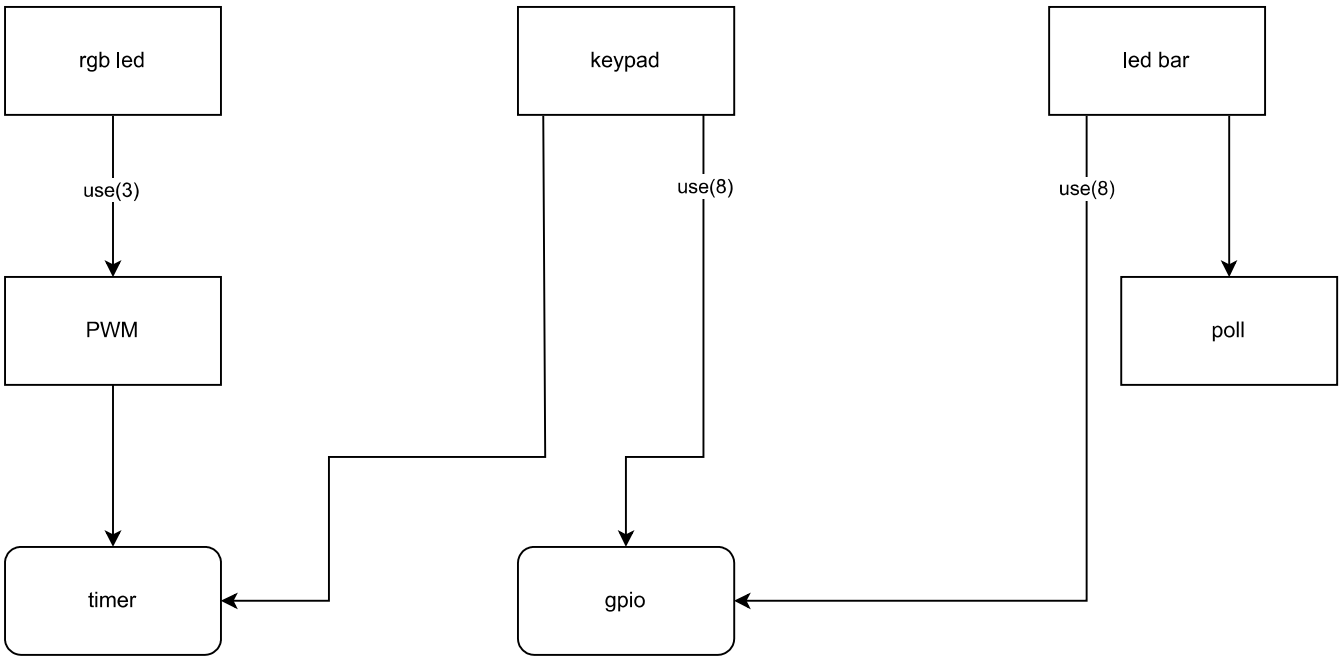
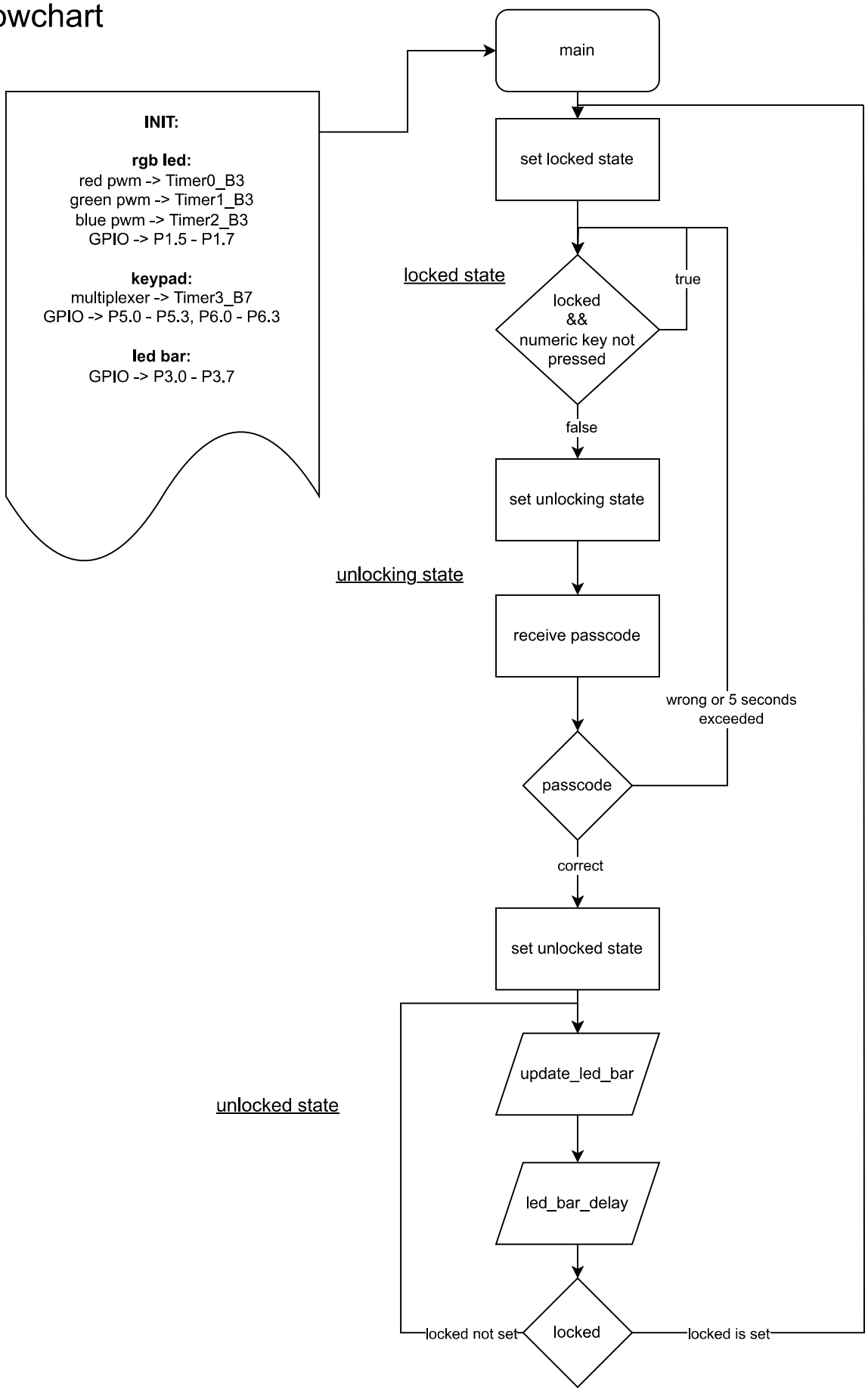
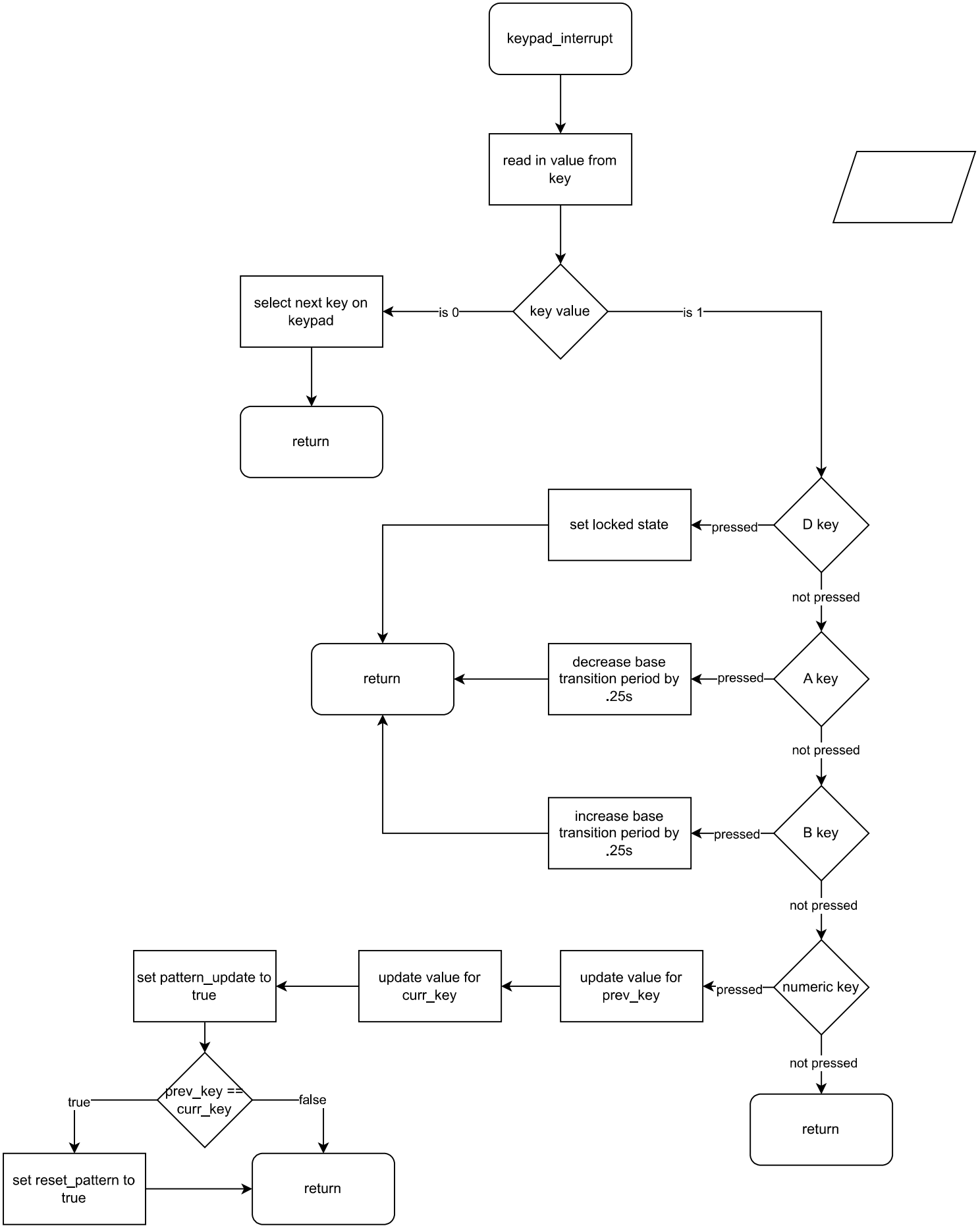
Software Architecture

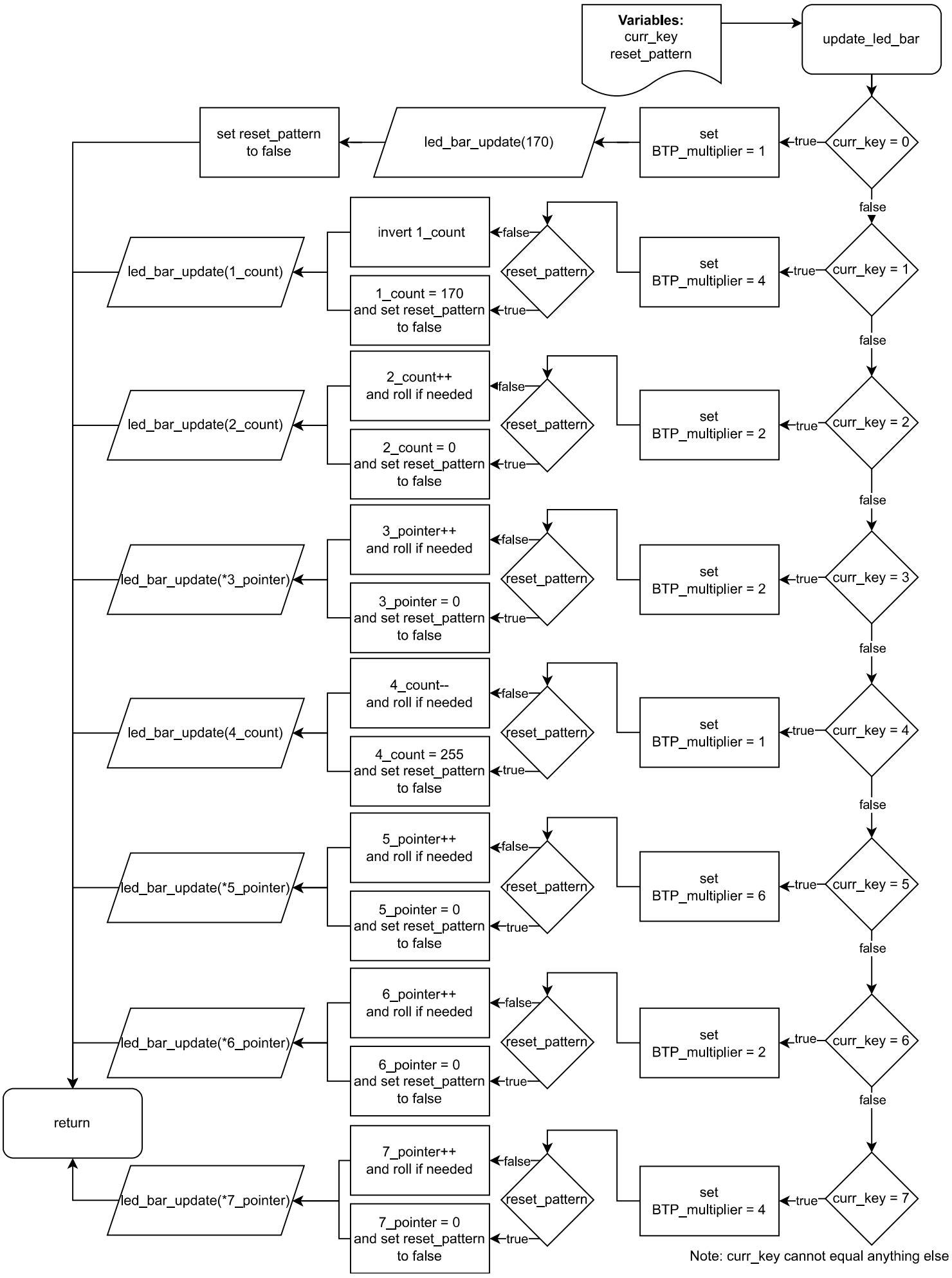


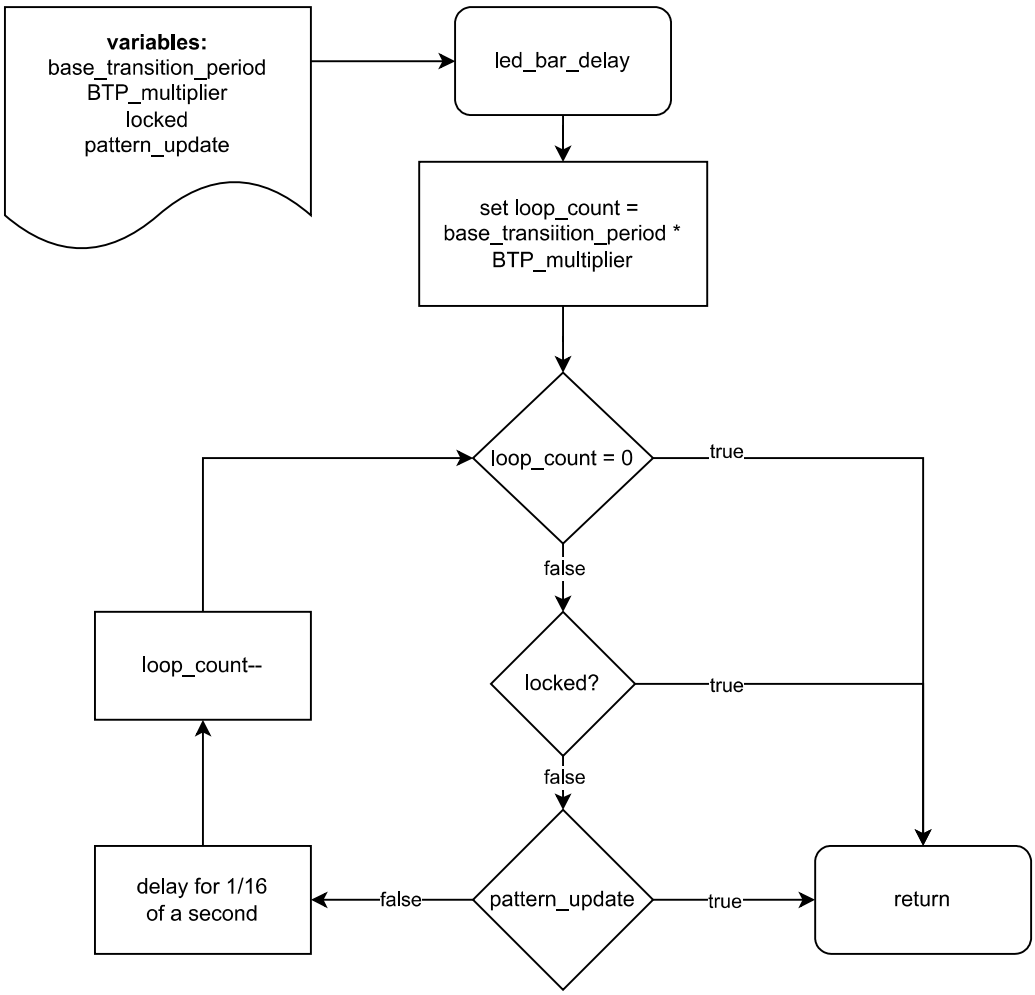
Flowchart

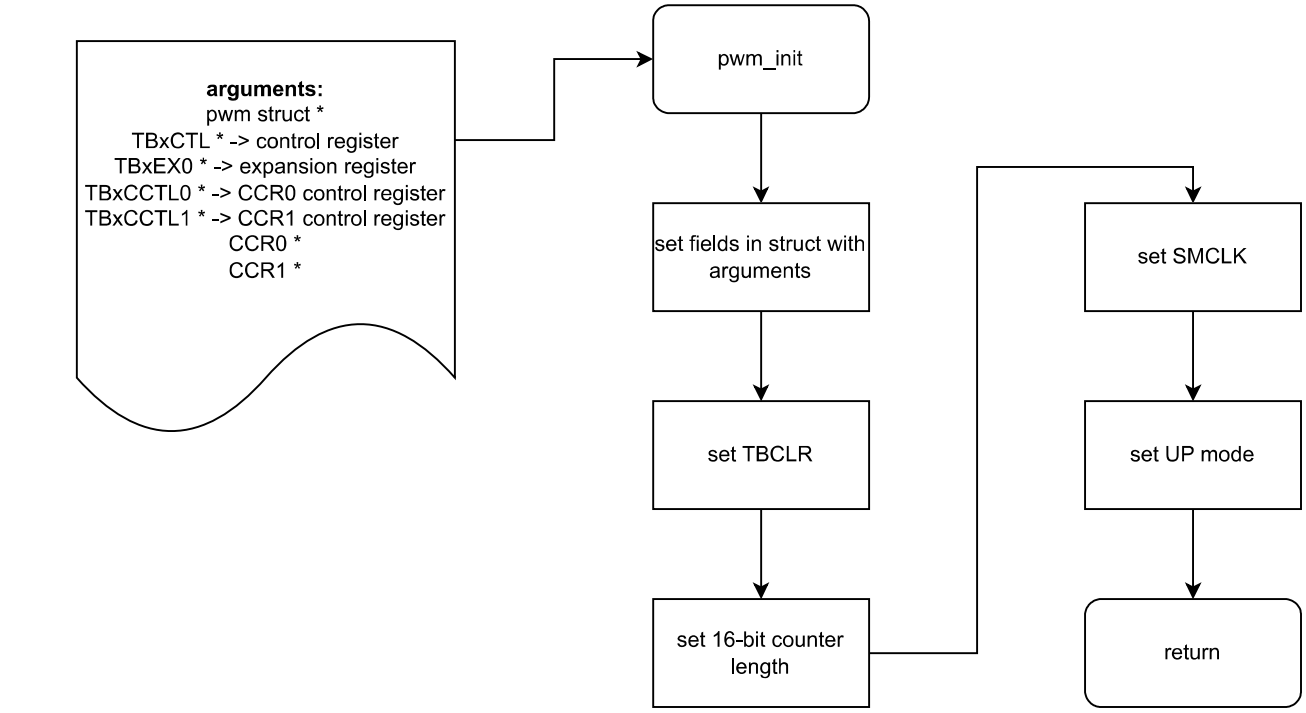

main
INIT:
rgb led:
red pwm -> Timer0_B3
green pwm -> Timer1_B3
blue pwm -> Timer2_B3
GPIO -> P1.5 - P1.7
keypad:
multiplexer -> Timer3_B7
GPIO -> P5.0 - P5.3, P6.0 - P6.3
led bar:
GPIO -> P3.0 - P3.7
set locked state
locked state
locked && numeric key not pressed
true
false
set unlocking state
unlocking state
receive passcode
wrong or 5 seconds exceeded
passcode
correct
set unlocked state
update_led_bar
unlocked state
led_bar_delay
locked
locked not set
locked is set

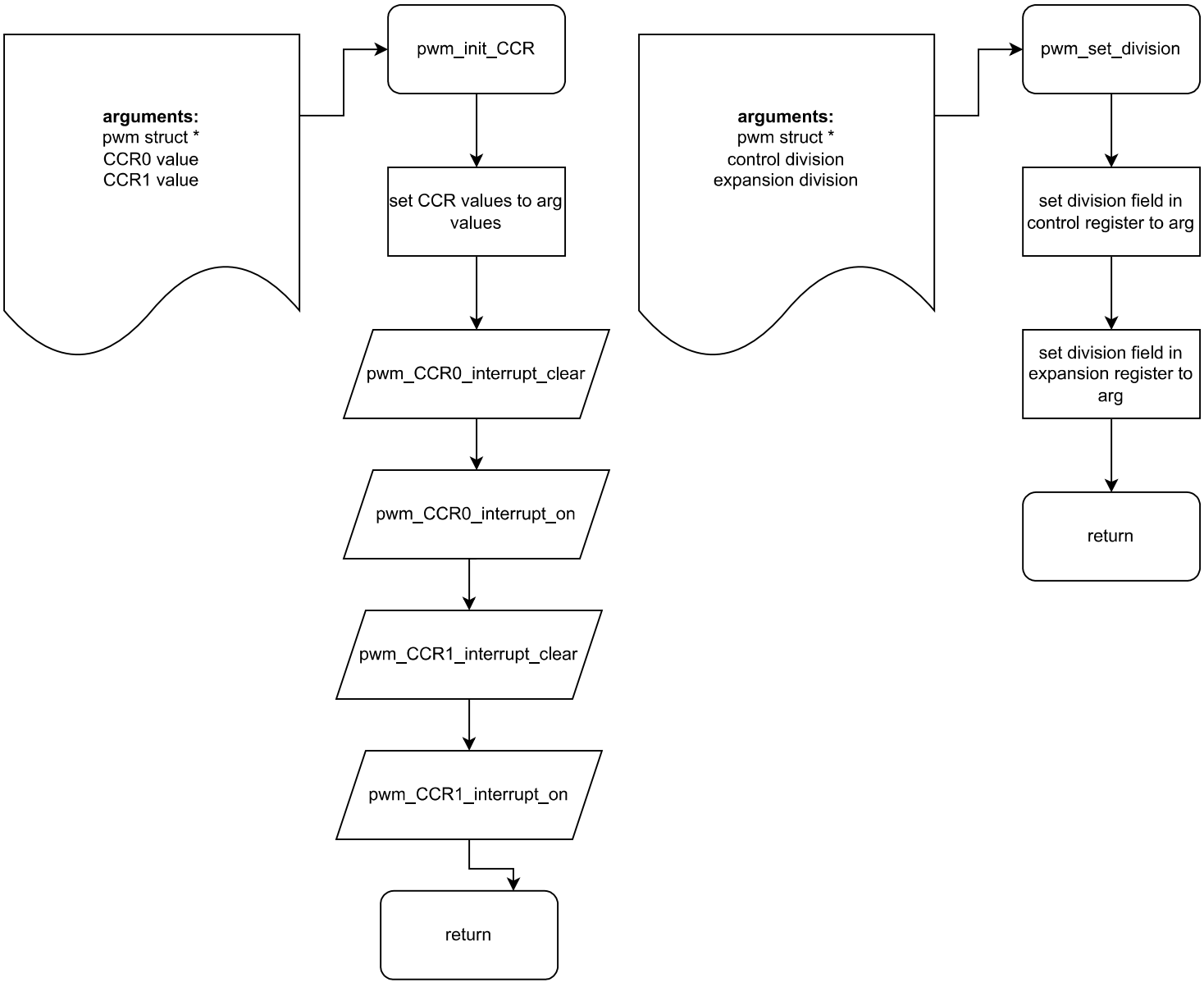
These are the interrupt sevice routines for the rgb led


keypad_interrupt
read in value from key
key value
is 0
is 1
select next key on keypad
return
D key
pressed
set locked state
not pressed
A key
pressed
decrease base transition period by .25s
return
not pressed
B key
pressed
increase base transition period by .25s
not pressed
numeric key
pressed
update value for prev_key
update value for curr_key
set pattern_update to true
not pressed
return
prev_key == curr_key
true
false
set reset_pattern to true
return

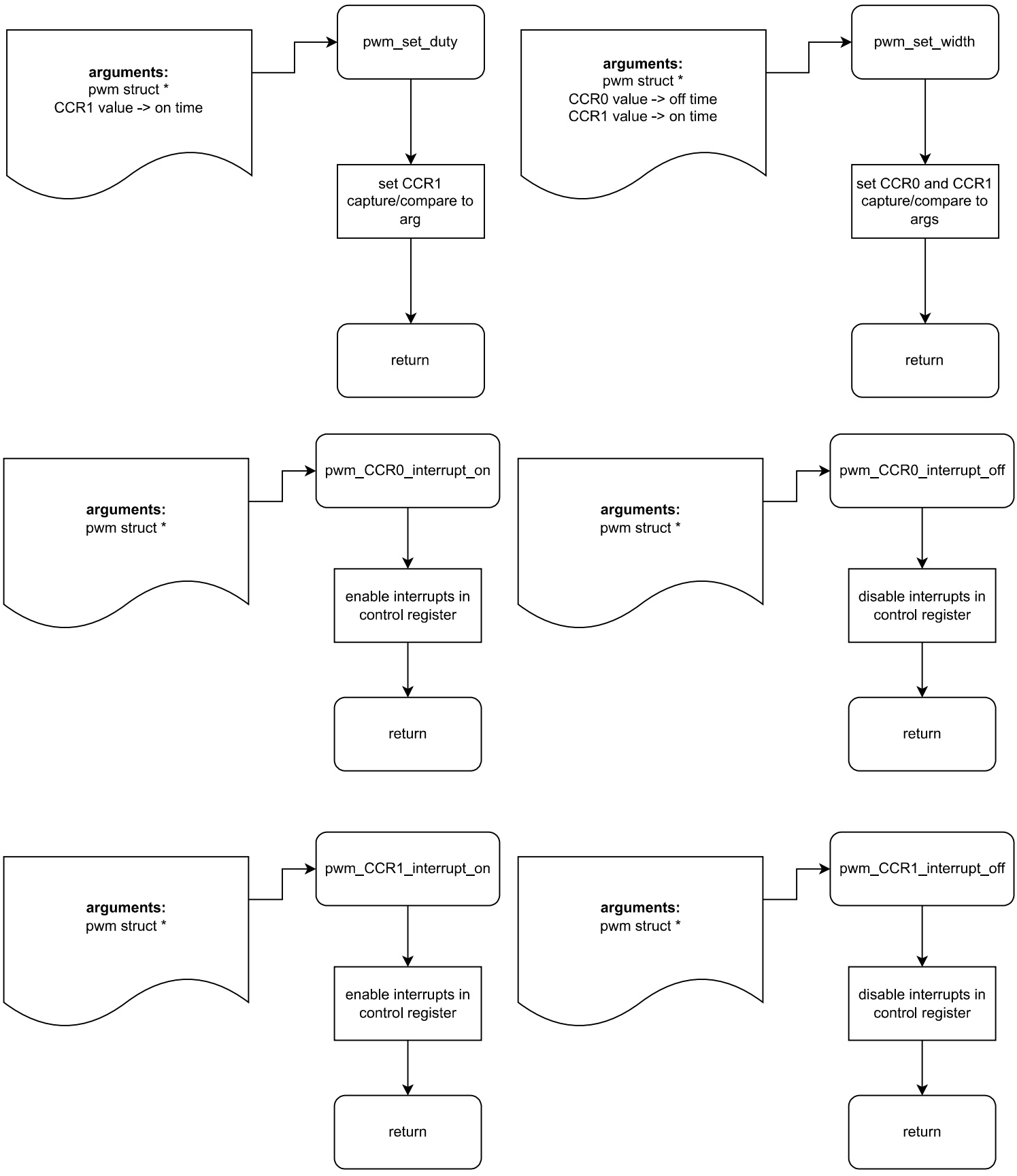

Variables:
curr_key
reset_pattern
update_led_bar
curr_key = 0
true
set
BTP_multiplier = 1
led_bar_update(170)
set reset_pattern
to false
false
curr_key = 1
true
set
BTP_multiplier = 4
reset_pattern
false
invert 1_count
led_bar_update(1_count)
true
1_count = 170
and set reset_pattern
to false
curr_key = 2
true
set
BTP_multiplier = 2
reset_pattern
false
2_count++
and roll if needed
led_bar_update(2_count)
true
2_count = 0
and set reset_pattern
to false
curr_key = 3
true
set
BTP_multiplier = 2
reset_pattern
false
3_pointer++
and roll if needed
led_bar_update(*3_pointer)
true
3_pointer = 0
and set reset_pattern
to false
curr_key = 4
true
set
BTP_multiplier = 1
reset_pattern
false
4_count--
and roll if needed
led_bar_update(4_count)
true
4_count = 255
and set reset_pattern
to false
curr_key = 5
true
set
BTP_multiplier = 6
reset_pattern
false
5_pointer++
and roll if needed
led_bar_update(*5_pointer)
true
5_pointer = 0
and set reset_pattern
to false
curr_key = 6
true
set
BTP_multiplier = 2
reset_pattern
false
6_pointer++
and roll if needed
led_bar_update(*6_pointer)
true
6_pointer = 0
and set reset_pattern
to false
curr_key = 7
true
set
BTP_multiplier = 4
reset_pattern
false
7_pointer++
and roll if needed
led_bar_update(*7_pointer)
true
7_pointer = 0
and set reset_pattern
to false
return
Note: curr_key cannot equal anything else

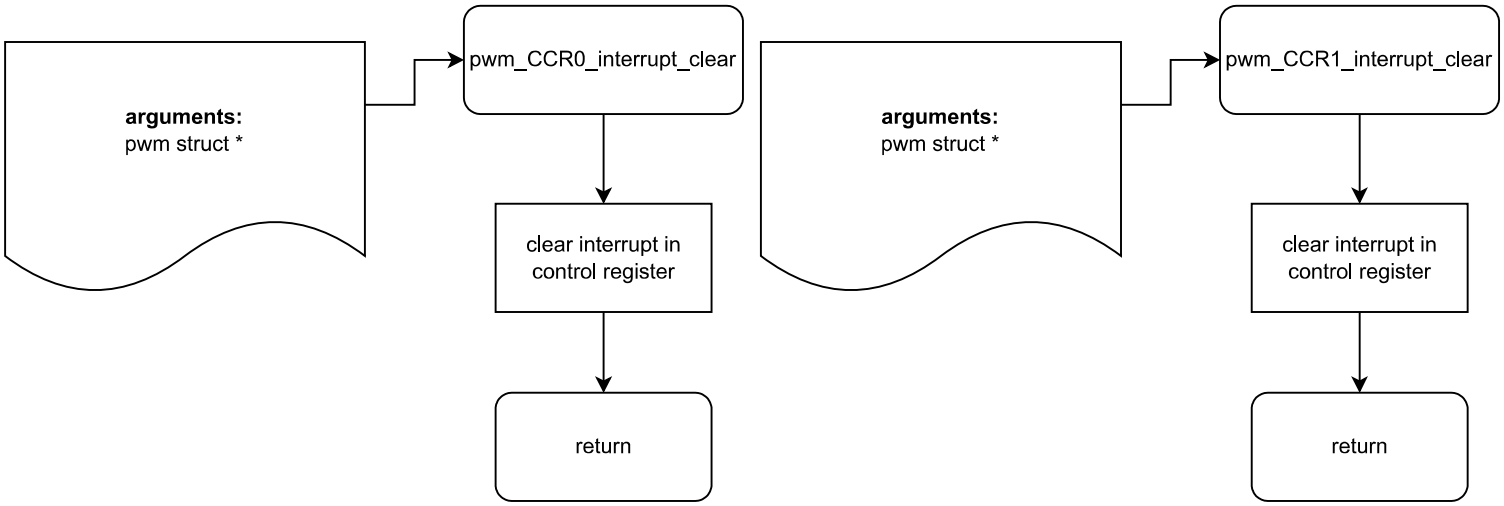

variables:
base_transition_period
BTP_multiplier
locked
pattern_update
led_bar_delay
set loop_count =
base_transiition_period *
BTP_multiplier
loop_count = 0
true
false
loop_count--
locked?
true
false
delay for 1/16
of a second
false
pattern_update
true
return

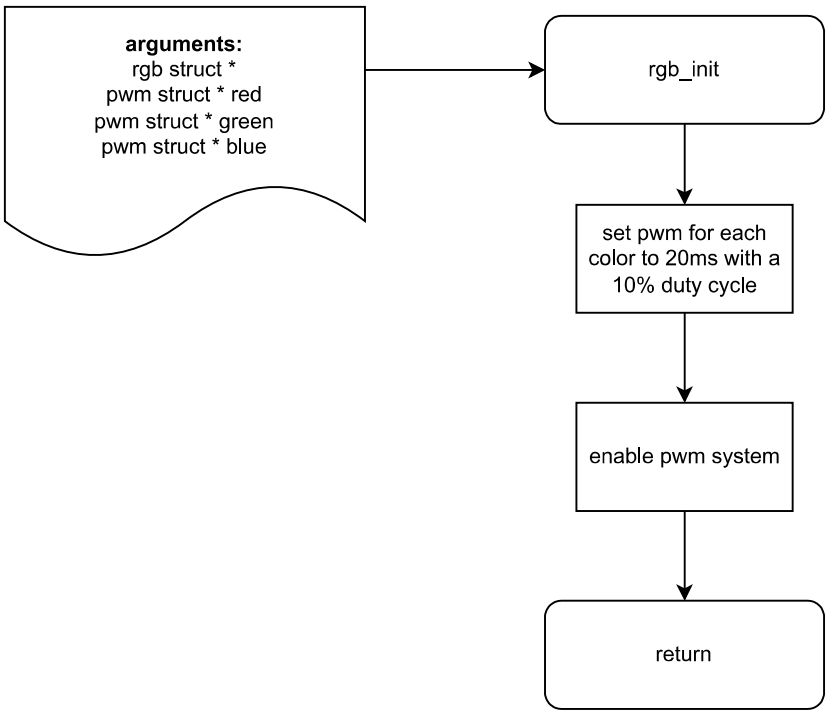

arguments:
pwm struct *
TBxCTL * -> control register
TBxEX0 * -> expansion register
TBxCCTL0 * -> CCR0 control register
TBxCCTL1 * -> CCR1 control register
CCR0 *
CCR1 *
pwm_init
set fields in struct with arguments
set TBCLR
set 16-bit counter length
set SMCLK
set UP mode
return


arguments:
pwm struct *
CCR0 value
CCR1 value
pwm_init_CCR
set CCR values to arg values
pwm_CCR0_interrupt_clear
pwm_CCR0_interrupt_on
pwm_CCR1_interrupt_clear
pwm_CCR1_interrupt_on
return
arguments:
pwm struct *
control division
expansion division
pwm_set_division
set division field in control register to arg
set division field in expansion register to arg
return

These should be implemented as macros

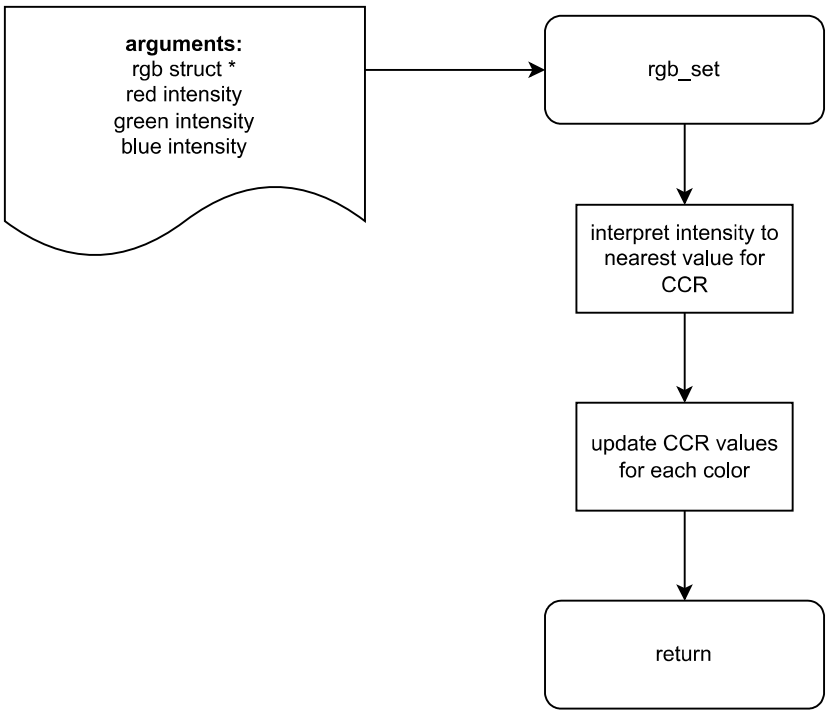



arguments:
pwm struct *
pwm_CCR0_interrupt_clear
clear interrupt in control register
return
arguments:
pwm struct *
pwm_CCR1_interrupt_clear
clear interrupt in control register
return

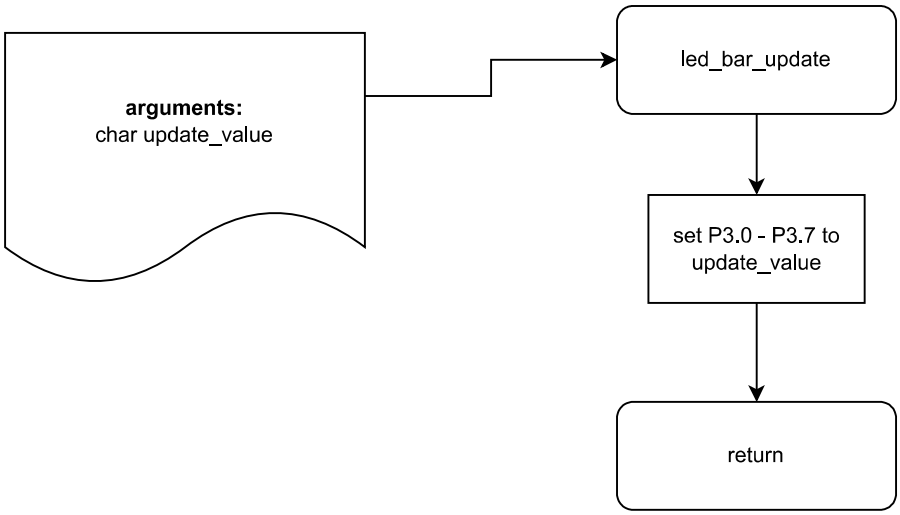


intensity should be from 0 - 10000

pwm
10000 = 100%
5000 = 50%



Macro


arguments:
char update_value
led_bar_update
set P3.0 - P3.7 to
update_value
return