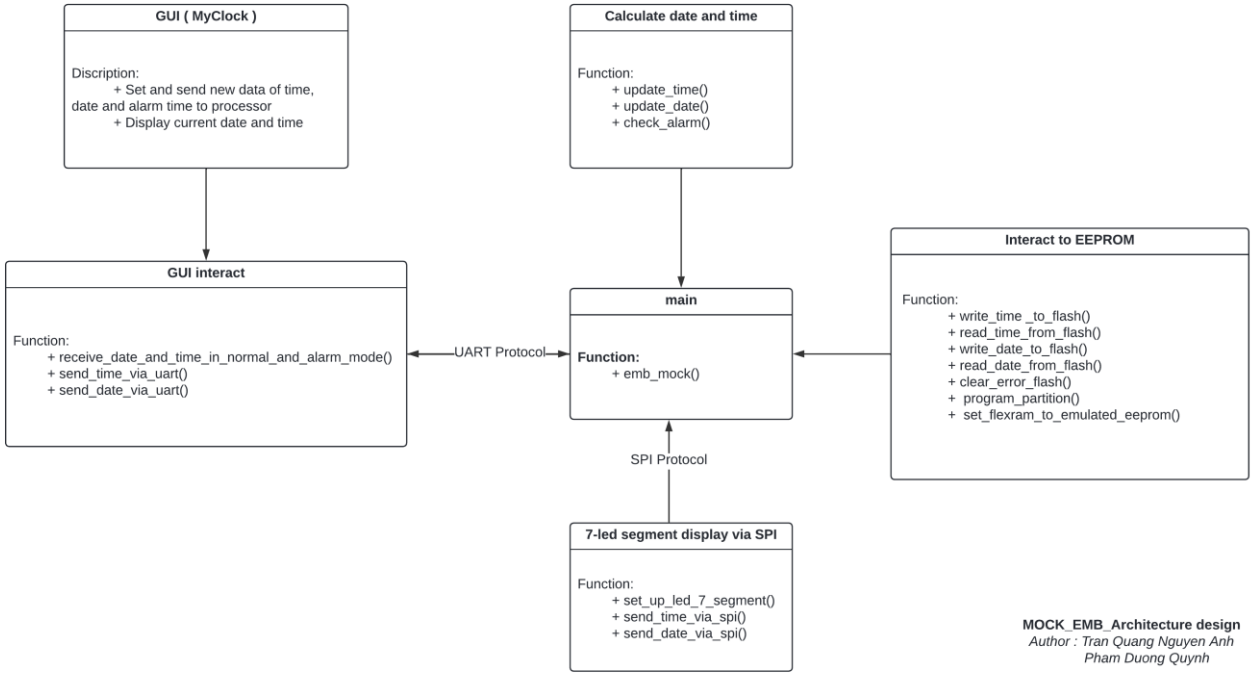
Author: Tran Quang Nguyen Anh, Pham Duong Quynh

THE DIGITAL CLOCK -

ARCHITECTURE AND DETAIL DESIGN

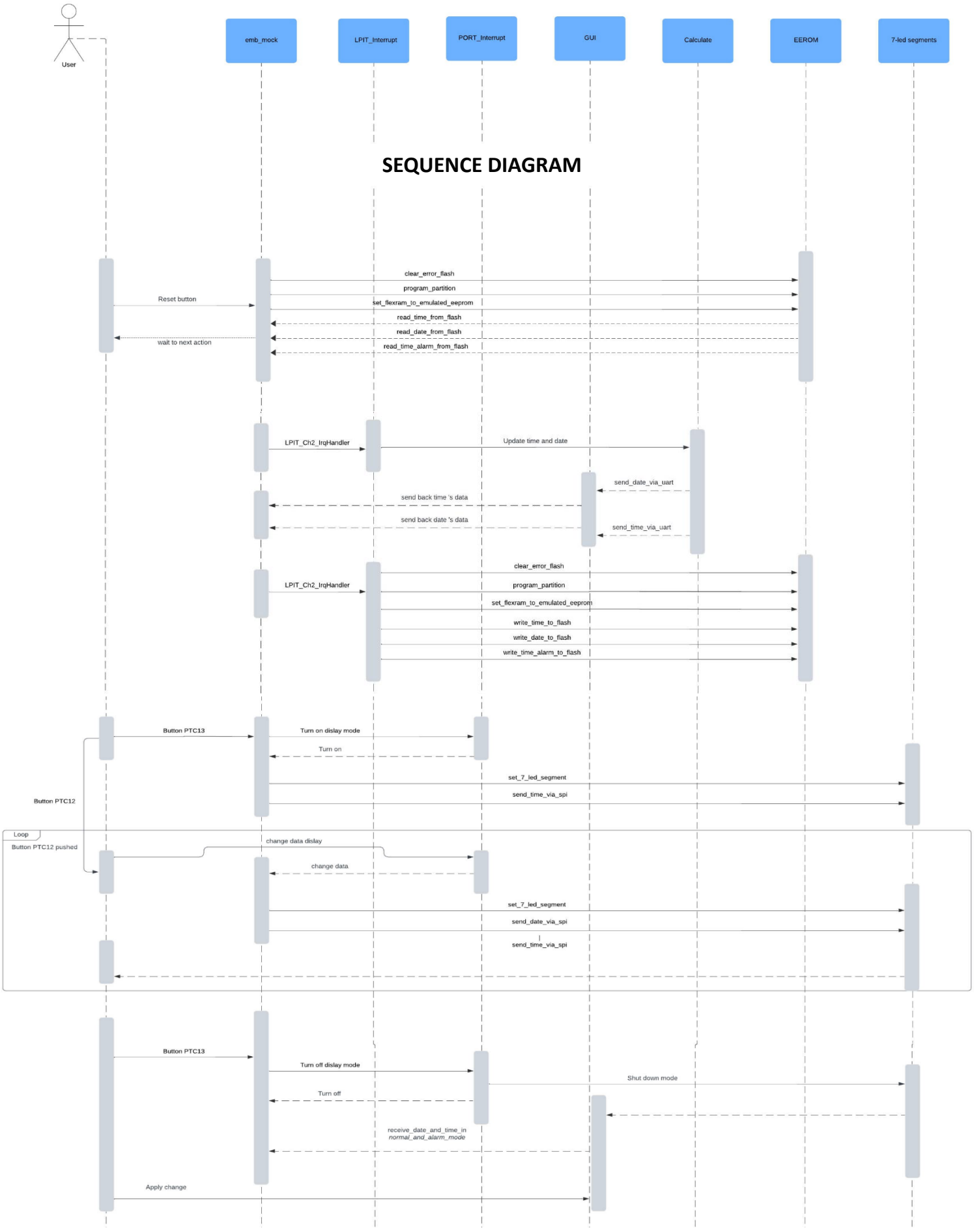
S32K144EVB

ARCHITECTURE DESIGN



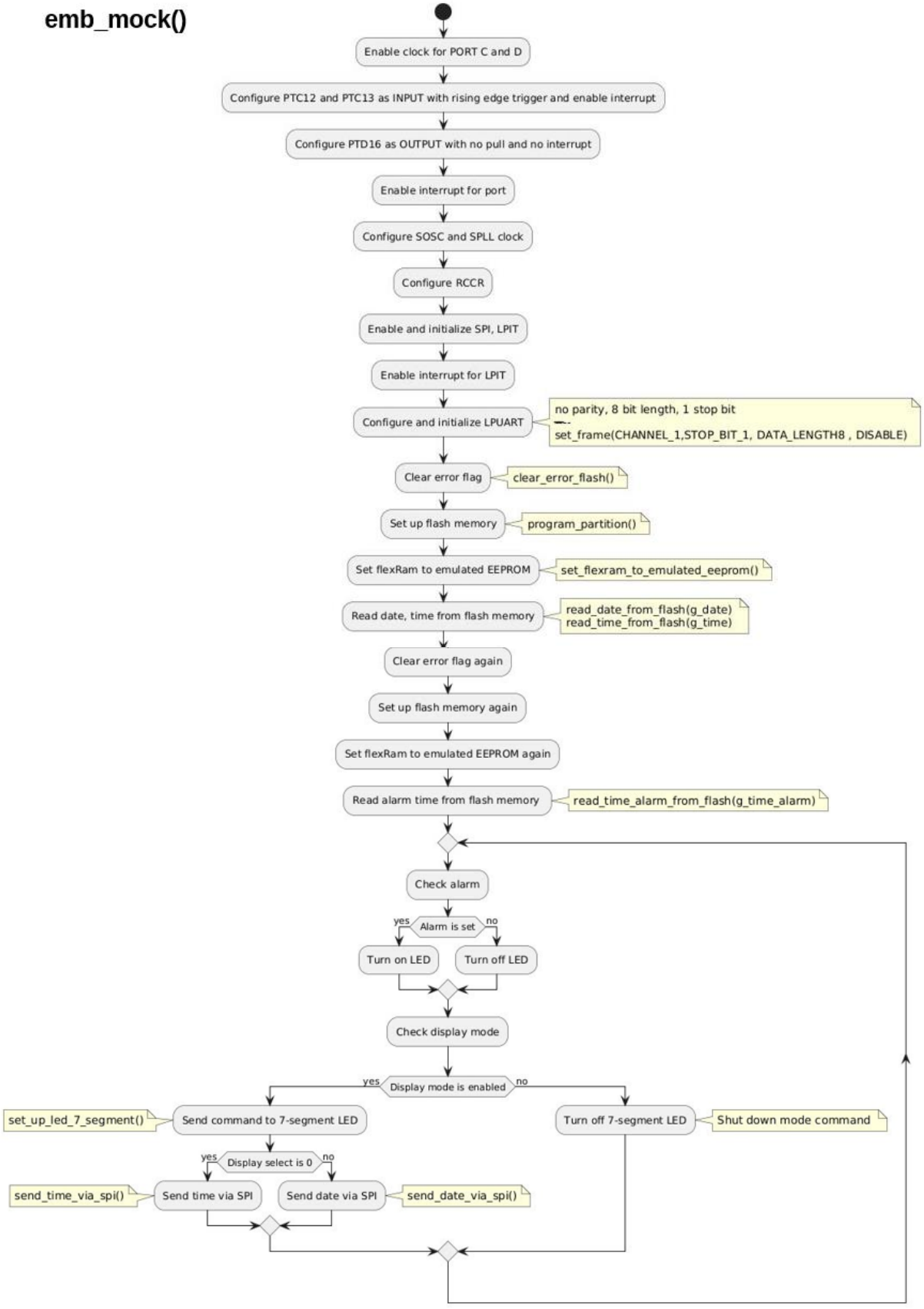
THE DIGITAL CLOCK DESIGN

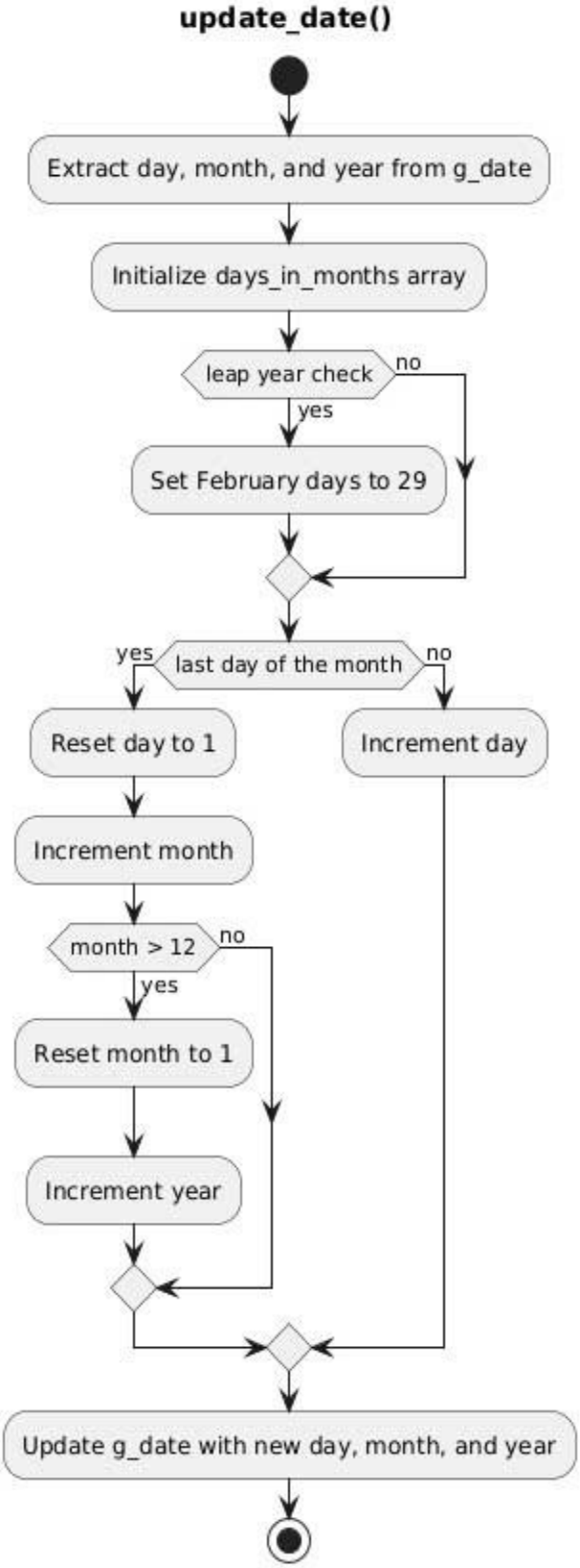
Author: Tran Quang Nguyen Anh, Pham Duong Quynh

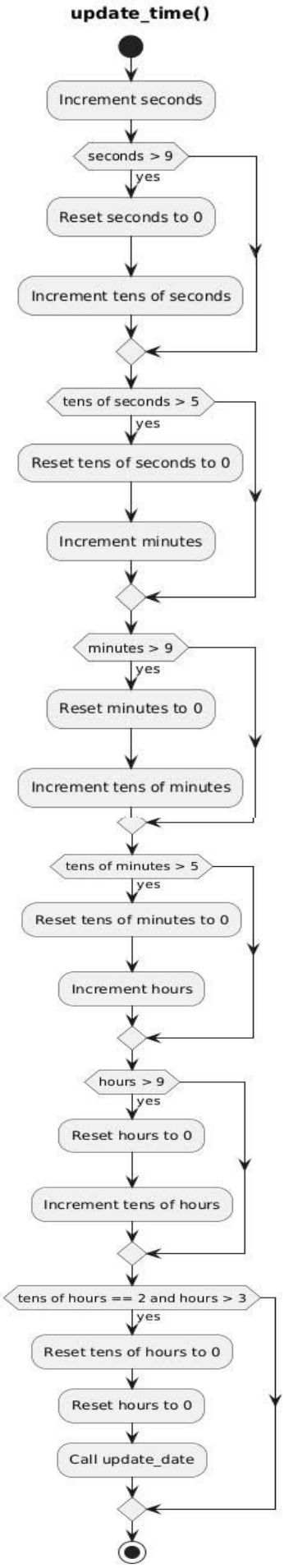


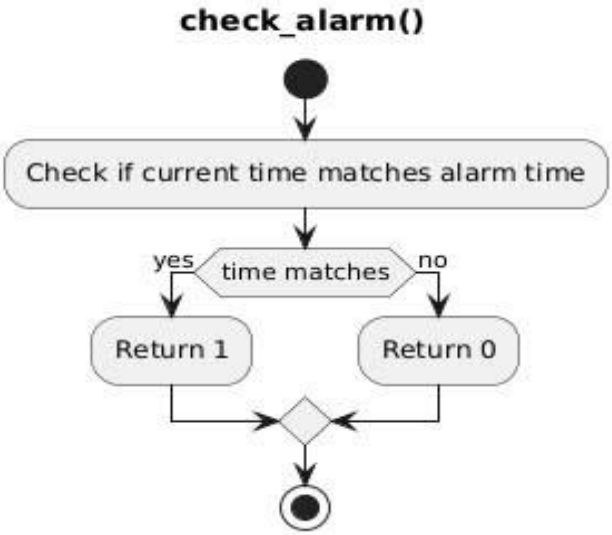
DETAIL DESIGN

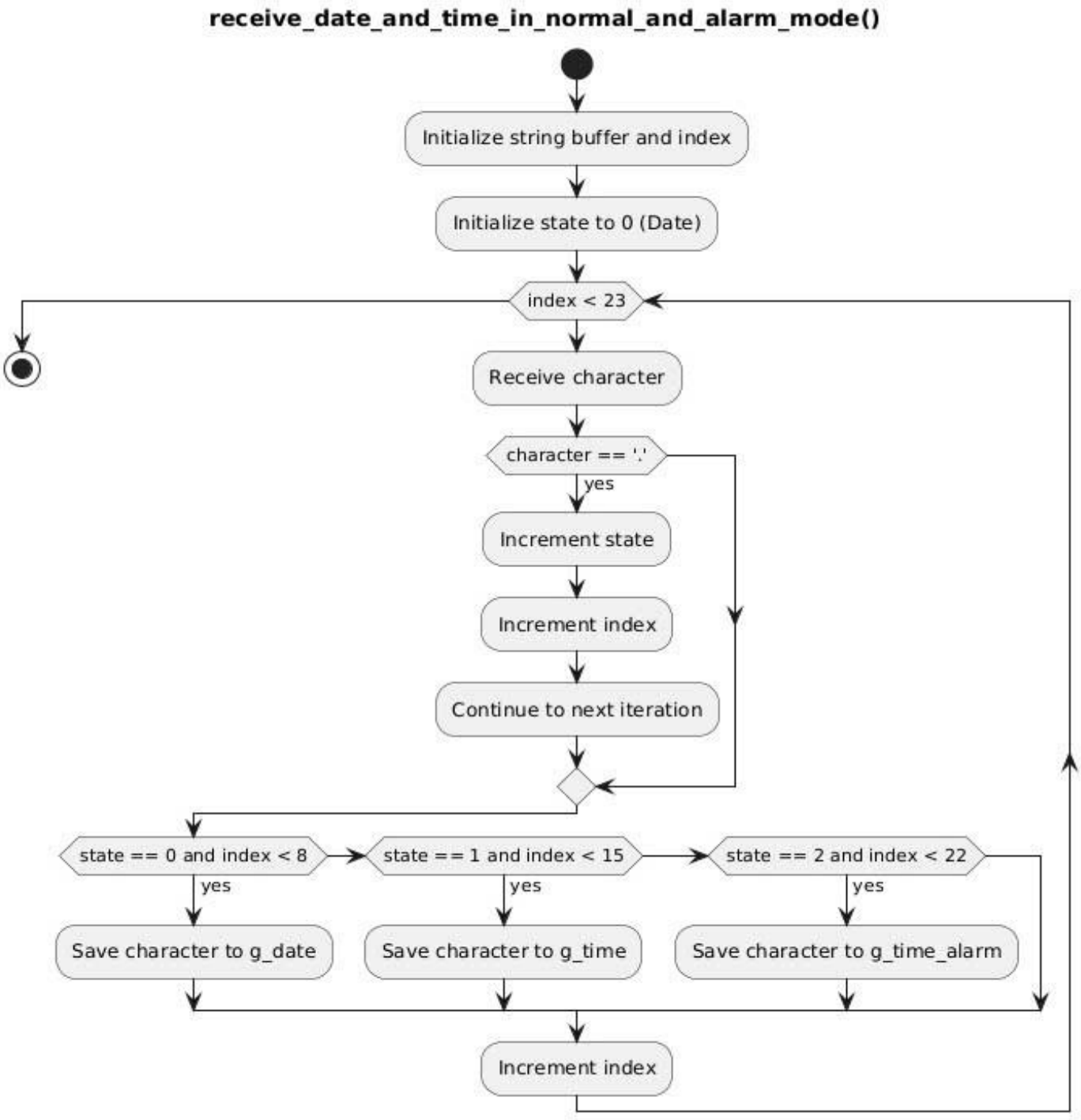
emb_mock()

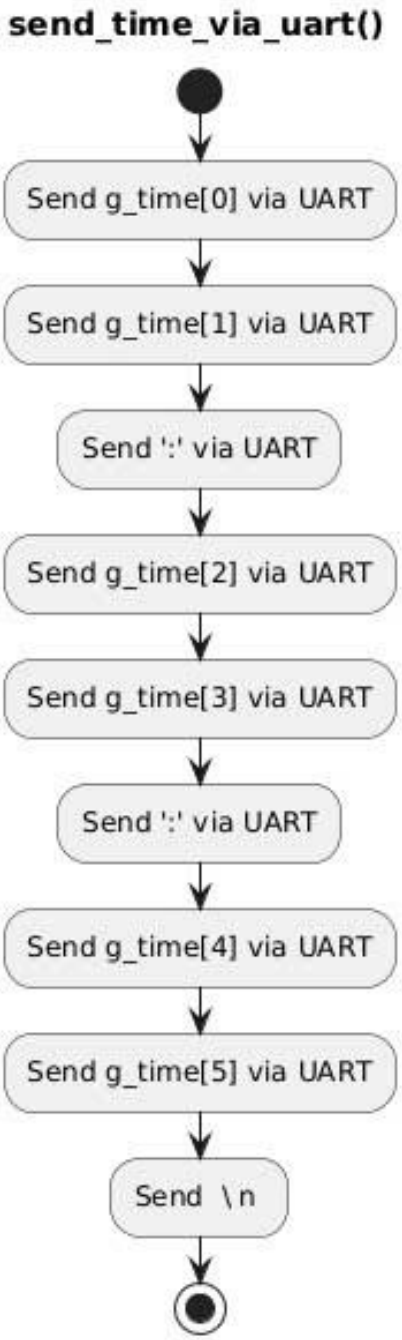


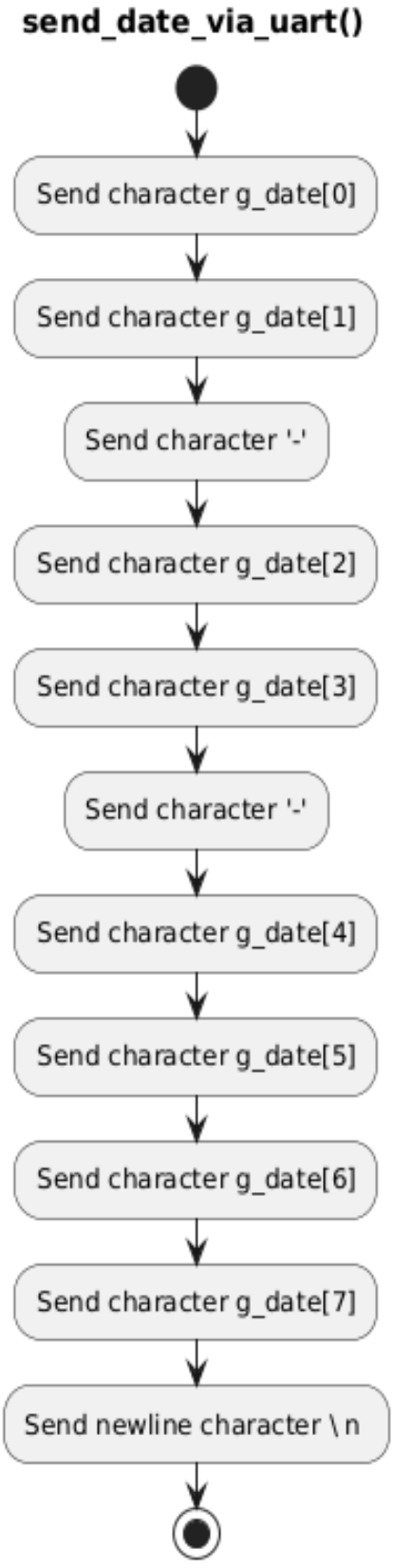

update_date()
Extract day, month, and year from g_date
Initialize days_in_months array
leap year check
no
yes
Set February days to 29
yes
last day of the month
no
Reset day to 1
Increment day
Increment month
month > 12
no
yes
Reset month to 1
Increment year
Update g_date with new day, month, and year

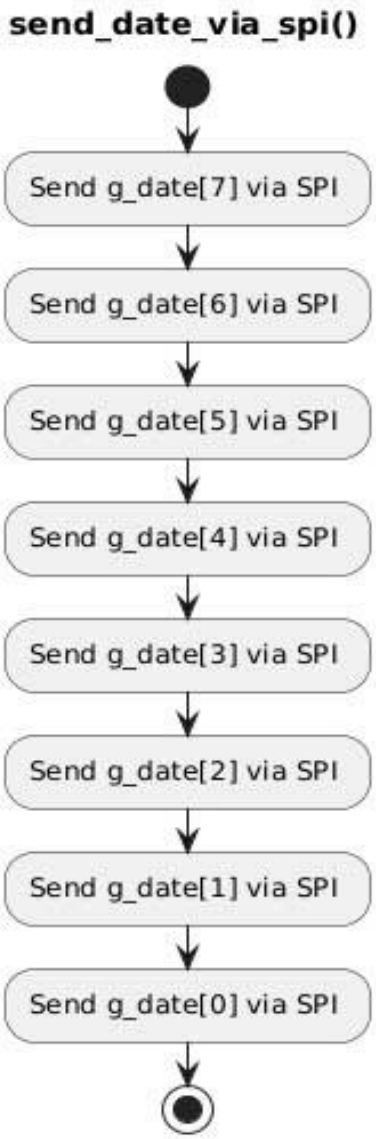

update_time()
Increment seconds
seconds > 9
yes
Reset seconds to 0
Increment tens of seconds
tens of seconds > 5
yes
Reset tens of seconds to 0
Increment minutes
minutes > 9
yes
Reset minutes to 0
Increment tens of minutes
tens of minutes > 5
yes
Reset tens of minutes to 0
Increment hours
hours > 9
yes
Reset hours to 0
Increment tens of hours
tens of hours == 2 and hours > 3
yes
Reset tens of hours to 0
Reset hours to 0
Call update_date

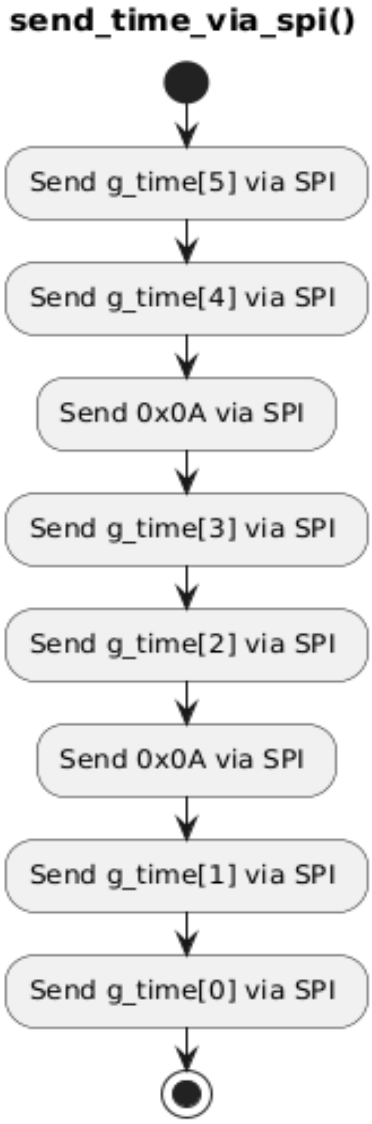

check_alarm()
Check if current time matches alarm time
yes
time matches
no
Return 1
Return 0

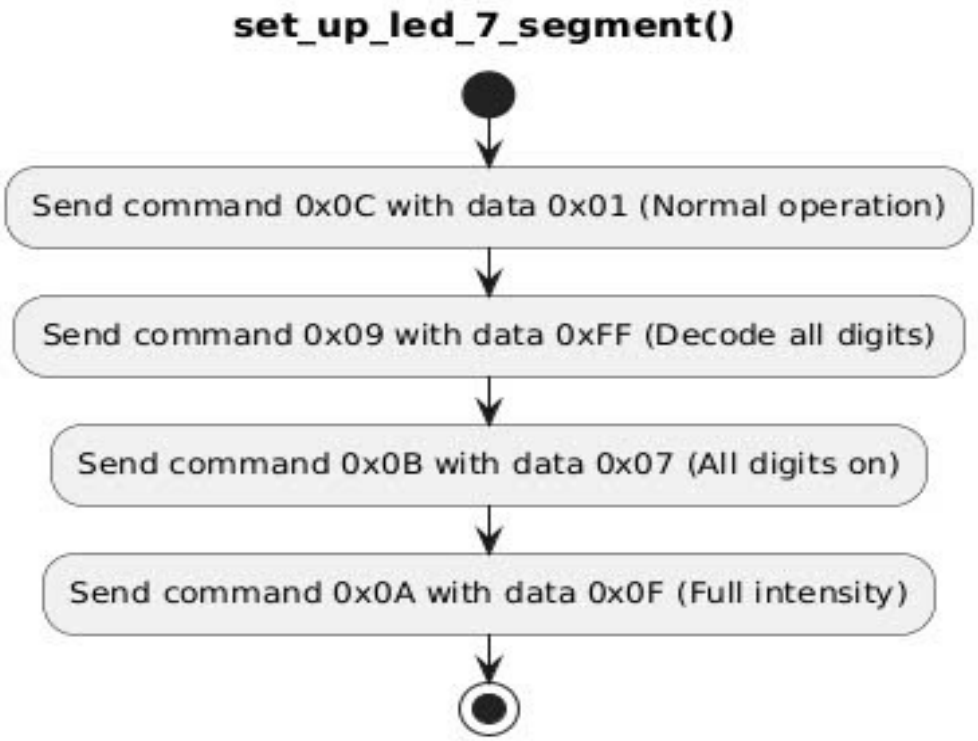

receive_date_and_time_in_normal_and_alarm_mode()
Initialize string buffer and index
Initialize state to 0 (Date)
index < 23
Receive character
character == '.'
yes
Increment state
Increment index
Continue to next iteration
state == 0 and index < 8
state == 1 and index < 15
state == 2 and index < 22
yes
yes
yes
Save character to g_date
Save character to g_time
Save character to g_time_alarm
Increment index

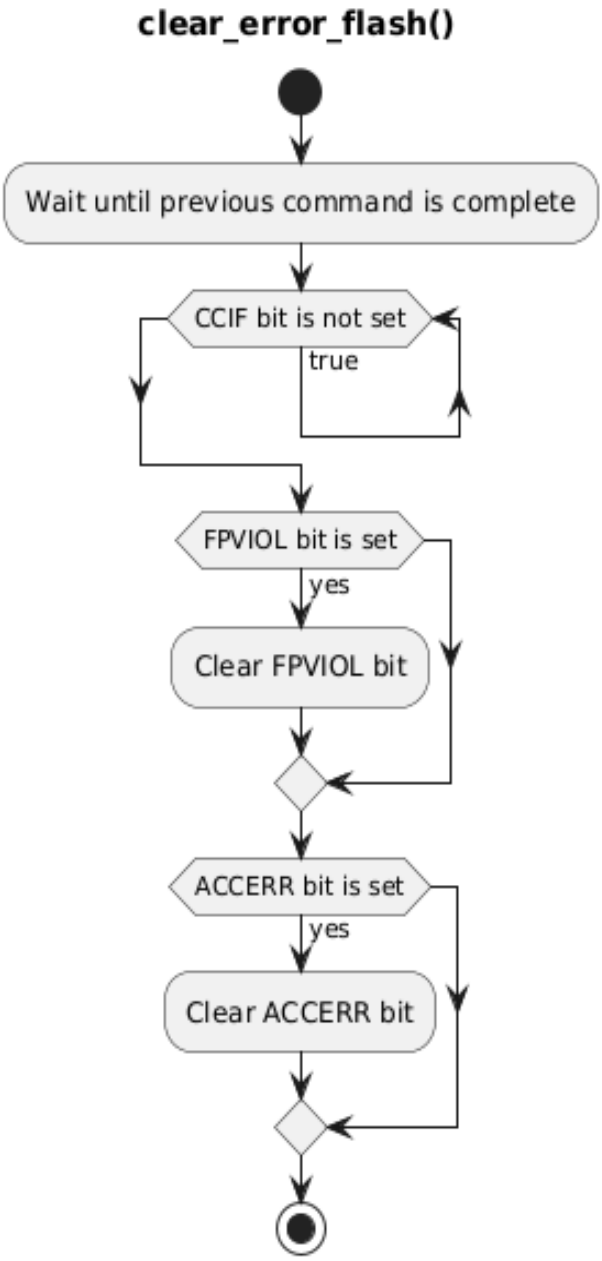

send_time_via_uart()
Send g_time[0] via UART
Send g_time[1] via UART
Send ':' via UART
Send g_time[2] via UART
Send g_time[3] via UART
Send ':' via UART
Send g_time[4] via UART
Send g_time[5] via UART
Send \n

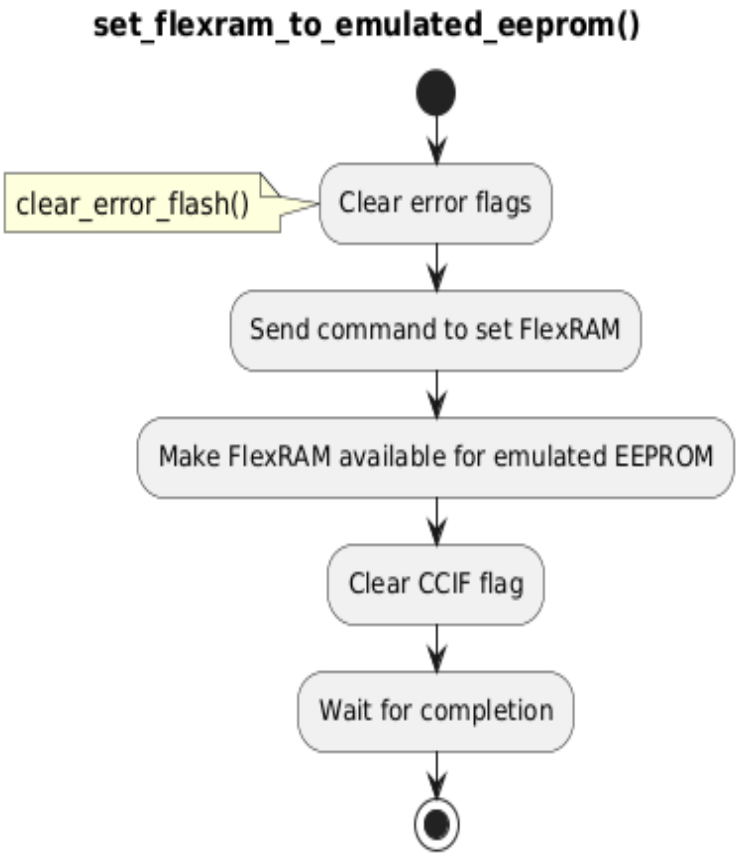

send_date_via_uart()
Send character g_date[0]
Send character g_date[1]
Send character '-'
Send character g_date[2]
Send character g_date[3]
Send character '-'
Send character g_date[4]
Send character g_date[5]
Send character g_date[6]
Send character g_date[7]
Send newline character \n

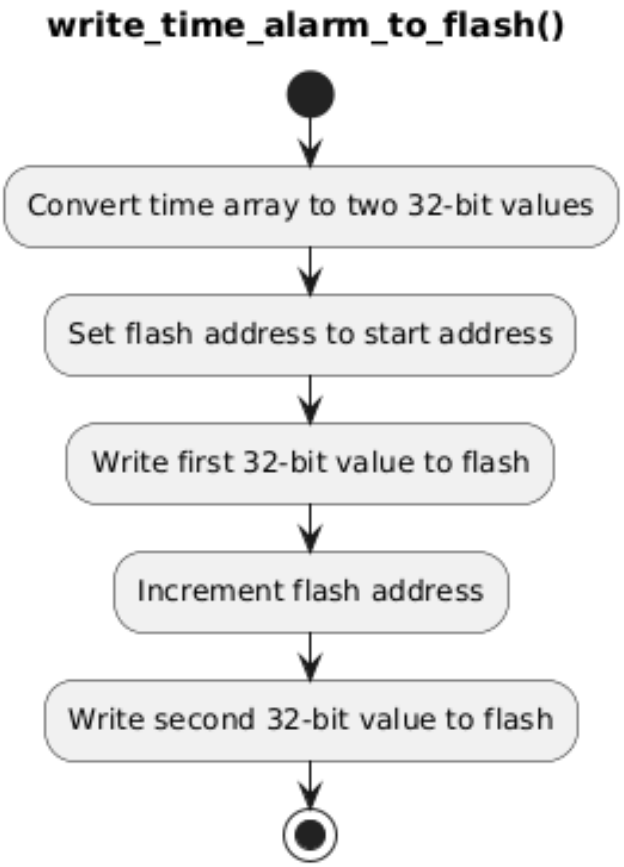

send_date_via_spi()
Send g_date[7] via SPI
Send g_date[6] via SPI
Send g_date[5] via SPI
Send g_date[4] via SPI
Send g_date[3] via SPI
Send g_date[2] via SPI
Send g_date[1] via SPI
Send g_date[0] via SPI

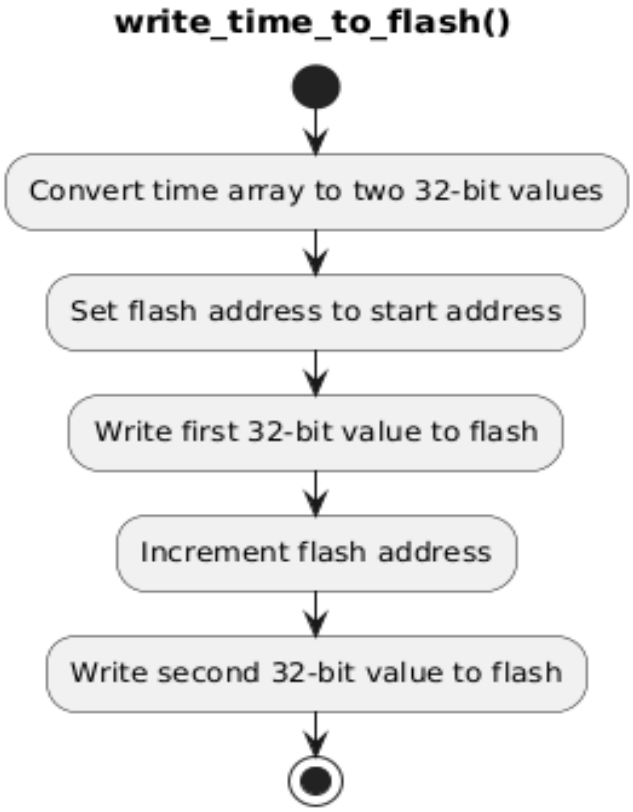

send_time_via_spi()
Send g_time[5] via SPI
Send g_time[4] via SPI
Send 0x0A via SPI
Send g_time[3] via SPI
Send g_time[2] via SPI
Send 0x0A via SPI
Send g_time[1] via SPI
Send g_time[0] via SPI

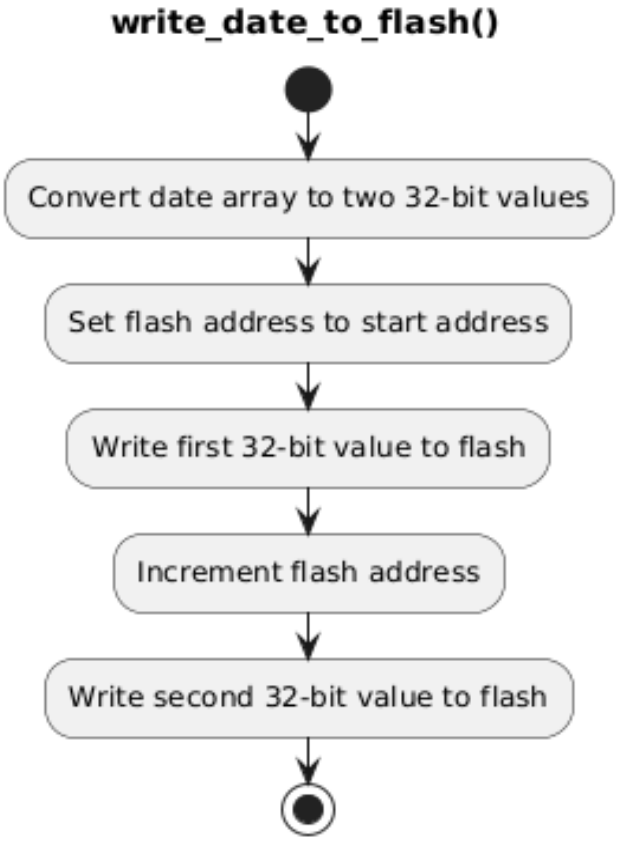

set_up_led_7_segment()
Send command 0x0C with data 0x01 (Normal operation)
Send command 0x09 with data 0xFF (Decode all digits)
Send command 0x0B with data 0x07 (All digits on)
Send command 0x0A with data 0x0F (Full intensity)

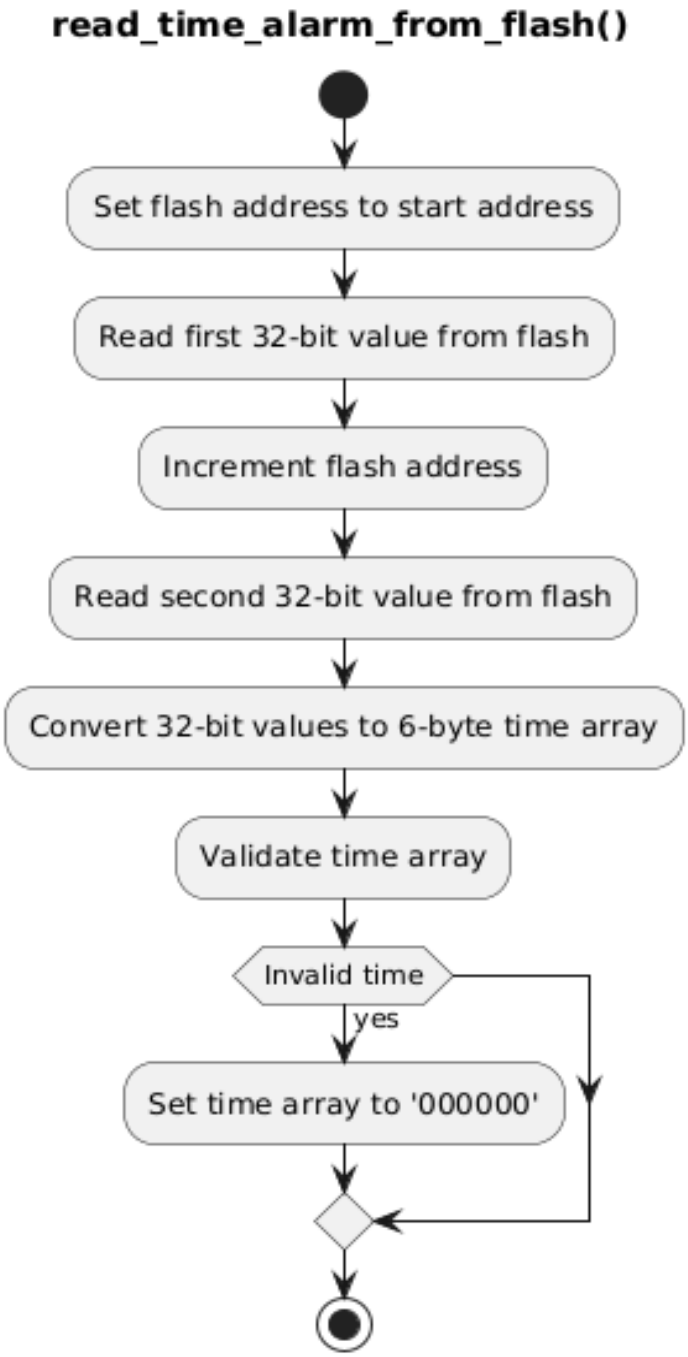

clear_error_flash()
Wait until previous command is complete
CCIF bit is not set
true
FPVIOL bit is set
yes
Clear FPVIOL bit
ACCERR bit is set
yes
Clear ACCERR bit

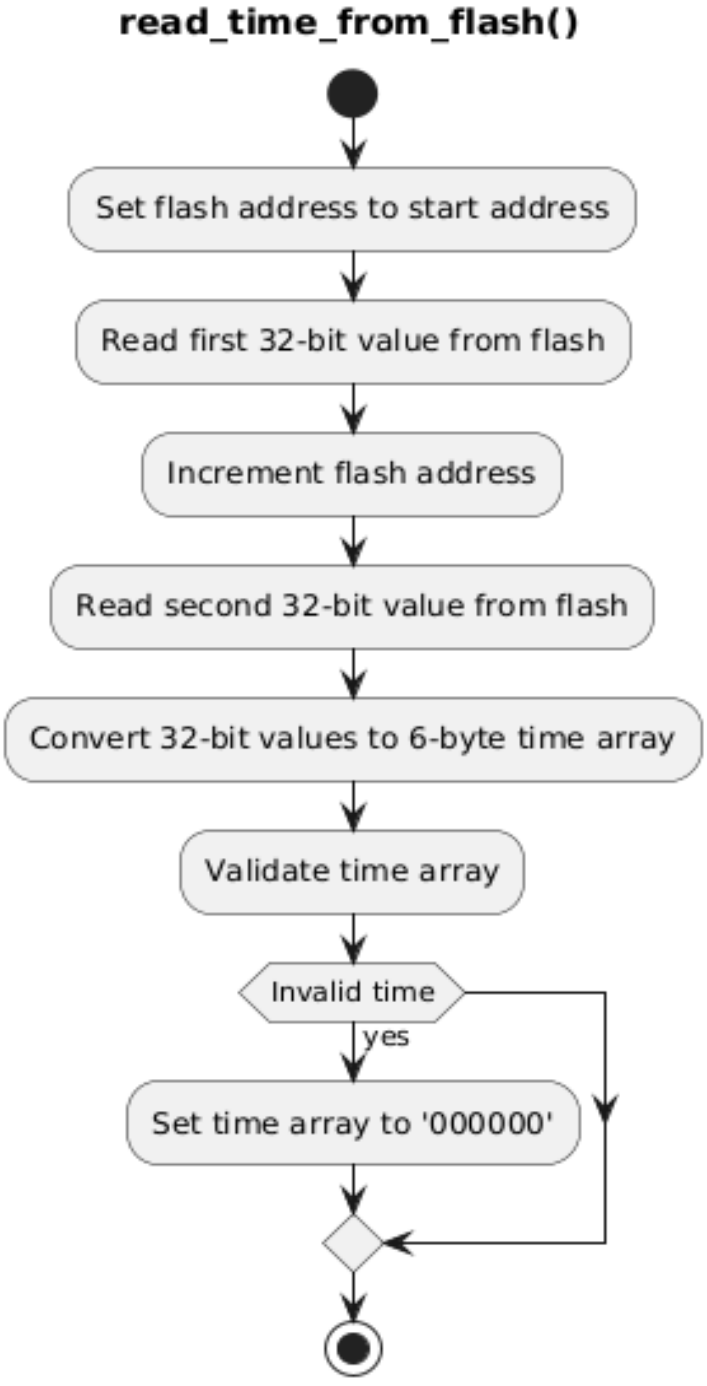

set_flexram_to_emulated_eeprom()
clear_error_flash()
Clear error flags
Send command to set FlexRAM
Make FlexRAM available for emulated EEPROM
Clear CCIF flag
Wait for completion

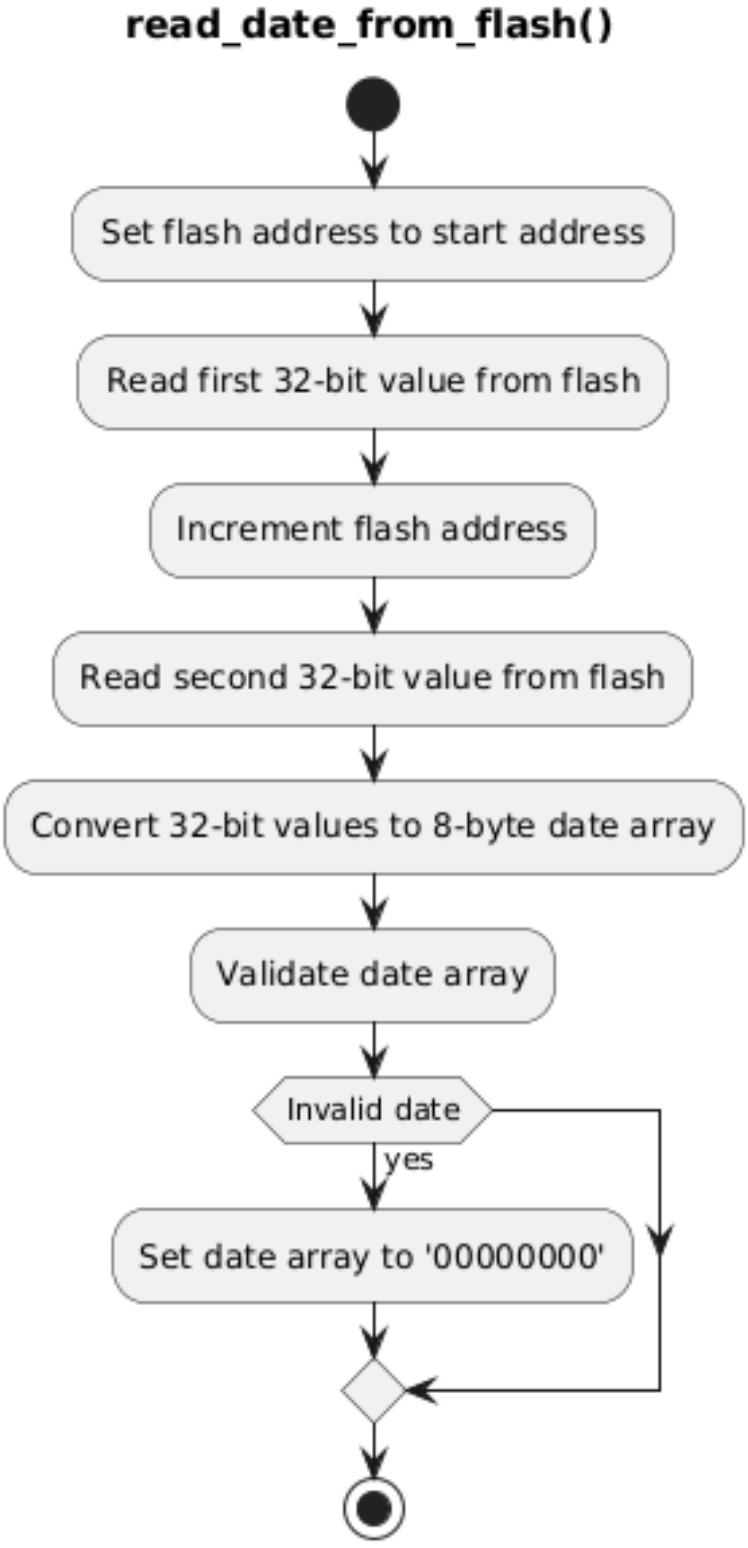

write_time_alarm_to_flash()
Convert time array to two 32-bit values
Set flash address to start address
Write first 32-bit value to flash
Increment flash address
Write second 32-bit value to flash


write_time_to_flash()
Convert time array to two 32-bit values
Set flash address to start address
Write first 32-bit value to flash
Increment flash address
Write second 32-bit value to flash


write_date_to_flash()
Convert date array to two 32-bit values
Set flash address to start address
Write first 32-bit value to flash
Increment flash address
Write second 32-bit value to flash


read_time_alarm_from_flash()
Set flash address to start address
Read first 32-bit value from flash
Increment flash address
Read second 32-bit value from flash
Convert 32-bit values to 6-byte time array
Validate time array
Invalid time
yes
Set time array to '000000'


read_time_from_flash()
Set flash address to start address
Read first 32-bit value from flash
Increment flash address
Read second 32-bit value from flash
Convert 32-bit values to 6-byte time array
Validate time array
Invalid time
yes
Set time array to '000000'


read_date_from_flash()
Set flash address to start address
Read first 32-bit value from flash
Increment flash address
Read second 32-bit value from flash
Convert 32-bit values to 8-byte date array
Validate date array
Invalid date
yes
Set date array to '00000000'

BLANK PAGE