MAINLINE

1. Initialize Timer7 clock (**Clock Module** base: =0x44E0_0000)
    a. Write #0x02 to **CM_PER_TIMER7_CLKCTRL** (offset: #0x7C)
    b. Select 32KHz by writing #0x02 to **PRCMCLKSEL_TIMER7** (=0x44E0_0504)
2. Initialize I2C2 clock
    a. Write #0x02 to **CM_PER_I2C2_CLKCTRL** (offset: #0x44)
3. Initialize GPIO1 clock
    a. Write #0x02 to **CM_PER_GPIO1_CLKCTRL** (offset: #0xAC)
4. Initialize GPIO1_31 for logic low (GPIO1 Base: =4804_C000)
    a. Write #0x8000_0000 to **GPIO_CLEARDATAOUT** (offset: #0x190)
5. Set up GPIO1_31 for output (GPIO1 Base: =4804_C000)
    a. Read **GPIO_OE** (offset: #0x134)
    b. Modify (set bit 31) =0x7FFF_FFFF
        i. Enable output GPIO1_31 (**pin 20 P8**)
    c. Write back
6. Initialize INTC for Timer7 and I2C2 (**INTC** base: =0x4820_0000)
    a. Reset INTC
        i. Write #0x2 to **INTC_SYSCONFIG** (offset: #0x10)
    b. Unmask INTC INT 95 (95 - 32 - 32 = 31) (DMTimer7)
        i. Write #0x8000_0000 to **INTC_MIR_CLEAR2** (offset: #0xC8)
    c. Unmask INTC INT 30 (30 = 30) (I2C2)
        i. Write #0x4000_0000 to **INTC_MIR_CLEAR0** (offset: #0x88)
7. Map I2C2
    a. Read **conf_uart1_rtsn** (Address: =0x44E1_097C)
    b. Modify (set bits 0,1, 4, 5) #0x33 for MODE3 (pin 19 of P9)
    c. Write back
    d. Read **conf_uart1_ctsn** (Address: =0x44E1_0978)
    e. Modify (set bits 0,1, 4, 5) #0x33 for MODE3 (pin 20 of P9)
    f. Write back
8. Configure I2C2 module (**I2C2** base: =0x4819_C000)
    a. Write #0x3 to **I2C_PSC** (offset: #0xB0)
        i. Sets clock divider to 4 so that $48 \div 4 = 12\ MHz$
        ii. ¿$= PSC + 1$
    b. Write #0x08 to **I2C_SCLL** (offset: #0xB4)
        i. Sets SCL low time to 50% duty with 400kbps and 12MHz clock
        ii. $tLOW = (SCLL + 7) * ICLK$
    c. Write #0x0A to **I2C_SCLH** (offset: #0xB8)
        i. Sets SCL high time to 50% duty with 400kbps and 12MHz clock
        ii. $tHIGH = (SCLH + 5) * ICLK$
    d. Write #0x40 to **I2C_OA** (offset: #0xA8)
        i. Sets it's own address to 0x40
    e. Write #0x8000 to **I2C_CON** (offset: #0xA4)
        i. Take I2C2 out of reset
9. Initialize I2C2 (**I2C2** base: =0x4819_C000)

    a. Read **I2C_CON** (offset: #0xA4)
    b. Modify (set bits 9 [**TRX**], 10 [**MST**]) #0x600
        i. Set to master transmitter
    c. Write back
10. Configure slave address and DATA counter (**I2C2** base: =0x4819_C000)
    a. Write #0x60 to **I2C_SA** (offset: #0xAC)
        i. Set slave address for current transmission
    b. Write #0x02 to **I2C_CNT** (offset: #0x98)
        i. Set data count to one byte
11. Initialize Timer7 count, overflow, etc (**Timer7** Base: =0x4804_A000)
    a. Reset Timer7
        i. Write #0x1 to **TIOCP_CFG** (offset: #0x10)
    b. Enable overflow interrupt
        i. Write #0x2 to **IRQENABLE_SET** (offset: #0x2C)
    c. Load TLDR and TCRR with count value
        i. Write 0xFFFF_0000 to **TLDR** (offset: #0x40)
        ii. Write 0xFFFF_0000 to **TCRR** (offset: #0x3C)
12. Enable IRQ interrupt
    a. Clear bit 7 in **CPSR**
13. Enable Timer7 interrupt signals
    a. Write #0x03 to **TCLR** (Address: =0x4804_A038)
14. Wait loop

INT_DIRECTOR

1. Save registers
2. Check if interrupt from Timer7
    a. Check bit 31 in **INTC_PENDING_IRQ2** (Address: =0x4820_00D8)
    b. If bit 31 = 0 go check I2C2
    c. If bit 31 = 1 check bit 1 of **IRQSTATUS** (Address: =0x4804_A028)
        i. If bit 1 = 1, go to TIMER7_SVC
        ii. If bit 1 = 0, go to RETURN_SVC
3. Check if interrupt from I2C2
    a. Check bit 30 in **INTC_PENDING_IRQ0** (Address: =0x4820_0098)
    b. If bit 30 = 0 go to RETURN_SVC
    c. If bit 30 = 1 check bit 4 of **I2C_IRQSTATUS** (Address: =0x4819_C028)
        i. If bit 4 = 1, go to I2C2_SVC
        ii. If bit 4 = 0, continue to check NACK
    d. Else if bit 30 = 1 and bit 4 = 0, check bit 1 of **I2C_IRQSTATUS**
        i. If bit 1 = 1, go to ERROR
        ii. If bit 1 = 0, go to RETURN_SVC

TIMER7_SVC

1. Disable Timer7 interrupt signals
    a. Write #0x00 to **TCLR** (Address: =0x4804_A038)

2. Initiate I2C2 transfer (**I2C2** base: =0x4819_C000)
    a. Poll **BB** (bit 12) of **I2C_IRQSTATUS_RAW** (offset: #0x24)
        i. See if line is busy; if bit = 0 continue, if bit = 1 keep polling
    b. Read **I2C_CON** (offset: #0xA4)
    c. Modify (set bits 0 [**STT**], 1 [**STP**]) #0x3
        i. Start transfer
    d. Write back
3. Enable I2C2 interrupt signals
    a. Write #0x12 to **I2C_IRQENABLE_SET** (Address: =0x4819_C02C)
4. Go to RETURN_SVC

I2C2_SVC

1. Write data
    a. Write data to **I2C_DATA** (Address: =0x4819_C09C)
2. Clear interrupt
    a. Write #0x10 to **I2C_IRQSTATUS** (Address: =0x4819_C028)
3. Disable I2C2 XRDY interrupt signal
    a. Write #0x10 to **I2C_IRQENABLE_CLR** (Address: =0x4819_C030)
4. Enable Timer7 interrupt signals (**Timer7** Base: =0x4804_A000)
    a. Load TLDR and TCRR with count value
        i. Write 0xFFFF_E000 to **TLDR** (offset: #0x40)
        ii. Write 0xFFFF_E000 to **TCRR** (offset: #0x3C)
    b. Write #0x03 to **TCLR** (Address: =0x4804_A038)
5. Go to RETURN_SVC

RETURN_SVC

1. Enable IRQ interrupt
    a. Write #0x1 to **INTC_CONTROL** (Address: =0x4820_0048)
2. Restore registers and return to wait loop

ERROR

1. Set GPIO1_31
    a. Write #0x8000_0000 to **GPIO_SETDATAOUT** (Address: =0x4804_C194)
    b.
2. Enter endless loop