STM32 Blue Pill Assembly Cheat Sheet (Peripherals)

1️⃣ GPIO (Digital I/O)

Example: Toggle PC13 LED

```
.syntax unified
.cpu cortex-m3
.thumb
.global main

.equ RCC_BASE, 0x40021000
.equ RCC_APB2ENR, 0x18
.equ IOPCEN, 4

.equ GPIOC_BASE, 0x40011000
.equ GPIOC_CRH, 0x04
.equ GPIOC_ODR, 0x0C
.equ PC13, 13

main:
    LDR R0, =RCC_BASE
    LDR R1, [R0,#RCC_APB2ENR]
    ORR R1,R1,#(1<<IOPCEN)
    STR R1,[R0,#RCC_APB2ENR]

    LDR R0,=GPIOC_BASE
    LDR R1,[R0,#GPIOC_CRH]
    BIC R1,R1,#(0xF<<20)
    ORR R1,R1,#(1<<20)
    STR R1,[R0,#GPIOC_CRH]

loop:
    LDR R1,[R0,#GPIOC_ODR]
    EOR R1,R1,#(1<<PC13)
    STR R1,[R0,#GPIOC_ODR]
    BL delay
    B loop

delay:
    MOV R2,#0xFF
outer: MOV R3,#0xFF
inner: SUBS R3,R3,#1
       BNE inner
       SUBS R2,R2,#1
       BNE outer
       BX LR
```

2️⃣ USART / UART (Transmit 'A')

```
.equ USART1_BASE,0x40013800
.equ USART_SR,0x00
.equ USART_DR,0x04
.equ TXE,7

uart_send:
    LDR R0,=USART1_BASE
wait_txe:
    LDR R1,[R0,#USART_SR]
    TST R1,#(1<<TXE)
    BEQ wait_txe
    MOV R2,#'A'
    STRB R2,[R0,#USART_DR]
    BX LR
```

3️⃣ Timer (Basic delay / periodic)

```
.equ TIM2_BASE,0x40000000
.equ TIM_CR1,0x00
.equ TIM_CNT,0x24
.equ TIM_PSC,0x28
```

```
.equ TIM_ARR,0x2C

timer_init:
    LDR R0,=TIM2_BASE
    MOV R1,#7999        @ prescaler
    STR R1,[R0,#TIM_PSC]
    MOV R1,#999         @ auto-reload
    STR R1,[R0,#TIM_ARR]
    MOV R1,#1           @ CEN
    STR R1,[R0,#TIM_CR1]
    BX LR
```

4️⃣ ADC (Single Conversion, PA0)

```
.equ ADC1_BASE,0x40012400
.equ ADC_SR,0x00
.equ ADC_CR2,0x08
.equ ADC_DR,0x4C
.equ ADC1_CH0,0

adc_read:
    LDR R0,=ADC1_BASE
    LDR R1,[R0,#ADC_CR2]
    ORR R1,R1,#1        @ ADON
    STR R1,[R0,#ADC_CR2]
    ORR R1,R1,#1<<30    @ SWSTART
    STR R1,[R0,#ADC_CR2]
wait_adc:
    LDR R2,[R0,#ADC_SR]
    TST R2,#1           @ EOC
    BEQ wait_adc
    LDR R3,[R0,#ADC_DR] @ result
    BX LR
```

5️⃣ SPI (Master Transmit)

```
.equ SPI1_BASE,0x40013000
.equ SPI_CR1,0x00
.equ SPI_DR,0x0C
.equ TXE,1

spi_send:
    LDR R0,=SPI1_BASE
wait_txe:
    LDR R1,[R0,#SPI_CR1]
    TST R1,#(1<<TXE)
    BEQ wait_txe
    MOV R2,#0xAA
    STRB R2,[R0,#SPI_DR]
    BX LR
```

6️⃣ I2C (Master Write)

```
.equ I2C1_BASE,0x40005400
.equ I2C_CR1,0x00
.equ I2C_DR,0x10
.equ I2C_SR1,0x14

i2c_send:
    LDR R0,=I2C1_BASE
    MOV R1,#0x50        @ device address
    STRB R1,[R0,#I2C_DR]
    BX LR
```

7️⃣ EXTI (External Interrupt on PA0)

```
.equ EXTI_BASE,0x40010400
.equ EXTI_IMR,0x00
.equ EXTI_PR,0x14
.equ EXTI0,0

enable_exti:
```

```
LDR R0,=EXTI_BASE
LDR R1,[R0,#EXTI_IMR]
ORR R1,R1,#(1<<EXTI0)
STR R1,[R0,#EXTI_IMR]
BX LR
```

STM32F103C8T6 Assembly Language Cheat Sheet (Summary + Examples)

1️⃣ GPIO (Digital I/O)

Purpose: LED, Buttons, Digital output/input control
Registers:

- `RCC_APB2ENR` → enable GPIO clock
- `GPIOx_CRH/CRL` → configure pin mode (input/output)
- `GPIOx_ODR` → write output
- `GPIOx_IDR` → read input

Example: Toggle PC13 LED

```
LDR R0,=GPIOC_BASE
LDR R1,[R0,#GPIOC_ODR]
EOR R1,R1,#(1<<13)
STR R1,[R0,#GPIOC_ODR]
```

Summary: Simple toggle for blinking LED, software delay needed.

2️⃣ UART / USART

Purpose: Serial communication (send/receive characters or strings)
Registers:

- `USART_SR` → status (TXE, RXNE)
- `USART_DR` → data register

Example: Transmit single char

```
uart_send:
    LDR R0,=USART1_BASE
wait_txe:
    LDR R1,[R0,#USART_SR]
    TST R1,#(1<<7)      @ TXE
    BEQ wait_txe
    MOV R2,#'A'
    STRB R2,[R0,#USART_DR]
    BX LR
```

Summary: Can be extended to send string character by character using loop.

3️⃣ Timer (Basic Delay / Periodic Event)

Purpose: Create precise timing, trigger events periodically
Registers:

- `TIMx_PSC` → prescaler
- `TIMx_ARR` → auto-reload
- `TIMx_CR1` → enable counter

Example: Basic periodic toggle

```
MOV R1,#1
STR R1,[R0,#TIM_CR1]  @ enable timer
```

Summary: Timer can generate interrupts for periodic tasks.

4️⃣ADC (Analog to Digital Conversion)

Purpose: Read analog sensors (0-3.3V)
Registers:

- `ADC_CR2` → enable + start conversion
- `ADC_SR` → conversion complete
- `ADC_DR` → read result

Example: Read PA0 analog input

```
LDR R3,[R0,#ADC_DR]   @ store ADC result
```

Summary: Can read potentiometers, sensors. Can use continuous mode + interrupts.

5️⃣SPI (Master / Slave Communication)

Purpose: High-speed communication with SPI devices (EEPROM, sensors, displays)
Registers:

- `SPI_CR1` → configuration
- `SPI_DR` → transmit/receive

Example: Transmit byte

```
STRB R2,[R0,#SPI_DR]
```

Summary: Works with external SPI devices, requires clock + mode setup.

6️⃣ I2C (Master / Slave Communication)

Purpose: Communicate with I2C devices (EEPROM, RTC, sensors)
Registers:

- `I2C_CR1` → control
- `I2C_DR` → data
- `I2C_SR1` → status

Example: Send device address

```
STRB R1,[R0,#I2C_DR]
```

Summary: Use for sensors, RTC, EEPROM. Can use interrupts for event-driven communication.

7️⃣ EXTI / Interrupts

Purpose: Handle external events like button presses asynchronously
Registers:

- `EXTI_IMR` → unmask interrupt
- `EXTI_PR` → pending flag

Example: Enable interrupt for PA0

```
LDR R0,=EXTI_BASE
ORR R1,R1,#(1<<0)
STR R1,[R0,#EXTI_IMR]
```

Summary: Can trigger ISR for buttons, sensors; reduces polling.

8️⃣UART String Transmission (Multiple Characters)

Example: Send string "HELLO"

```
uart_send_string:
    LDR R4,=string
next_char:
    LDRB R1,[R4],#1
    CMP R1,#0
    BEQ done
    BL uart_send
    B next_char
done:
    BX LR

string:
    .ascii "HELLO\0"
```

Summary: Combines UART transmit single char with loop for string.

9️⃣Software Delay

- Nested loops required for large delay in Assembly
- Thumb MOV allows only 8-bit immediate, so use nested loops

```
MOV R2,#0xFF
outer: MOV R3,#0xFF
inner: SUBS R3,R3,#1
       BNE inner
       SUBS R2,R2,#1
       BNE outer
       BX LR
```

Summary: Can adjust R2/R3 for blink speed or timing.

✅ Notes / Tips

1. All examples **Thumb / Cortex-M3 compatible**
2. GPIO / Timer / USART / ADC / SPI / I2C / EXTI covers main peripherals
3. UART string + interrupt examples show event-driven behavior
4. CubeIDE requires `main:` function for startup
5. PC13 LED is **low-active**