Microprocessors and Embedded Systems

Curated undergraduate study track (not accredited, no external exam). Original explanations and practice only, following the reviewed course handout.

Where hardware meets software

- EE03 built the logic; this course builds the machine that runs programs.
- Every register computation checked by bit widths; every timing by its **clock**; every code fragment by its state trace.

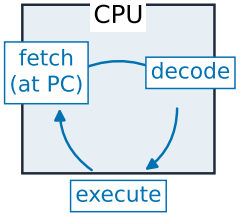
Architecture: the memory map is arithmetic

1. **Map** Flash 512K at 0x0000_0000-0x0007_FFFF; SRAM 96K at 0x2000_0000; periphery 64K at 0x4000_0000 - top nibbles 0, 2, 4 distinct, no overlap possible.
2. **Timer** ARR = 4999 gives 5000 counts (the count includes zero - N-1 is the rule); 8 MHz / 8 / 5000 = 200 Hz.
3. **Widths** 19 lines for 512K and 16 for 64K fit exactly; 96K needs 17 lines (128K of space), leaving 32K unused.

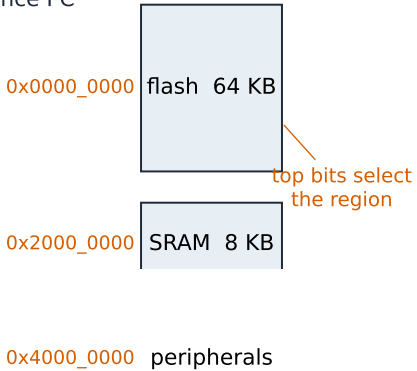
One chip, one loop

(1) one instruction per lap

fetch → decode → execute → advance PC



(2) the memory map



Flash at 0x0000_0000, SRAM at 0x2000_0000: the top bits keep them apart.

Embedded C: volatile and the register contract

1. **Trace** set bit 3: 0x00 -> 0x08; clear bit 5: 0x28 -> 0x08; toggle bit 3: back to 0x00; test bit 5: false - every before/after computable by hand.
2. **Wrap** A 32-bit counter at 84 MHz wraps in 51.1 s - the trace and the **clock** agree on the same program.
3. **Optimiser** -O2 may DELETE a volatile-free delay loop entirely - the compiler may compute its result at compile time.

Three views of one line

C

```
*GPIO_OUT |= (1u << 5);          set bit 5 only
```

assembly: load, mask, store

```
LDR  r0, =0x20000100     address of GPIO_OUT
LDR  r1, [r0]            bus transfer 1: read
ORR  r1, r1, #0x20       inside the CPU, no bus
STR  r1, [r0]            bus transfer 2: write
```

register transfer

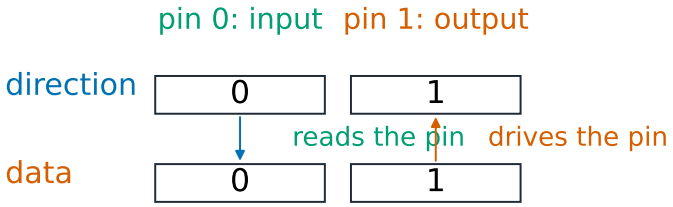
```
GPIO_OUT ← GPIO_OUT | 0x20       other bits unchanged
```

The OR-assign is load, OR, store: three operations, two **bus** transfers.

Memory and I/O: polling costs, DMA scales

1. **Compare** 64-byte **SPI** send: **polling** 896 cycles of CPU; DMA 40 cycles of CPU (plus 512 of **bus**) - 95.5% freed.
2. **Scale** Double the buffer: polling doubles to 1792; the DMA CPU cost stays 40 - only the bus time grows.
3. **Pull-up** The active-low button needs its pull-up: without it the pin floats and a 0 means nothing.

Direction says which way, data says what



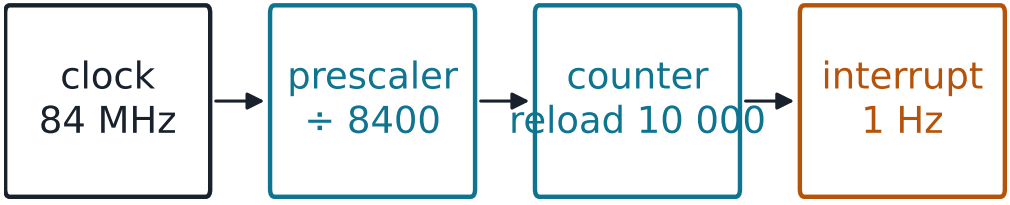
configure direction once, then read or write data

Configure the direction bit once; the data bit then reads or drives the pin.

Interrupts and timers: prescaler times reload

1. **Tick** 1 Hz from 84 MHz: factors 8400 x 10000 = 84 000 000, stored as PSC = 8399 and ARR = 9999 - both fit 16 bits (a factor of 84 000 would not: the width check is the design).
2. **Load** 10 kHz encoder ISR (2 us) + 2 kHz control ISR (8 us) = 3.6% of the CPU; the 100 us computation still fits its 500 us budget with 390 to spare.
3. **Debounce** 5 consecutive samples at 1 kHz: the bounce trace declares press at index 5 - 4 ms of bought immunity.

Two divisions to 1 Hz



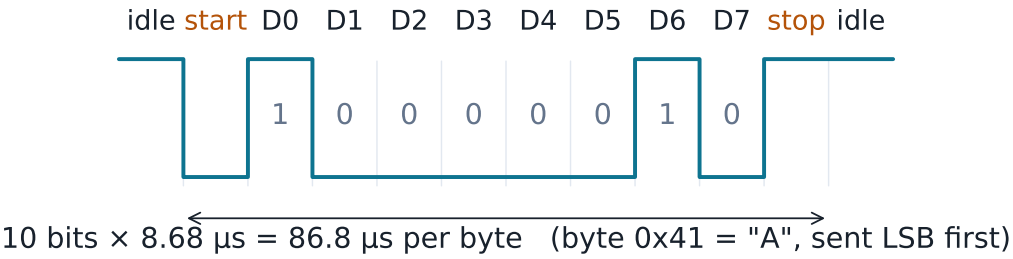
84 000 000 ÷ 8400 = 10 000 Hz; 10 000 ticks at 10 kHz = 1 s

Prescaler 8400 turns 84 MHz into a 10 kHz tick; reload 10 000 turns the tick into the 1 Hz **interrupt**.

Serial protocols: three points on one trade

1. **UART** 115200 baud 8N1: 86.8 us per byte; a 12-byte packet at 9600 takes 12.5 ms - over a 10 ms budget before the requirement even doubles.
2. **SPI** 10 MHz: 0.8 us per byte - about 108x this **UART**; a 16-bit **ADC** read at 2 MHz is 8 us plus CS overhead, 123k conversions/s.
3. **I2C** 400 kHz fast mode: 9 clocks per byte - the same 12-byte packet is 275 us, decisive margin over a 5 ms deadline.

Ten bit times per byte

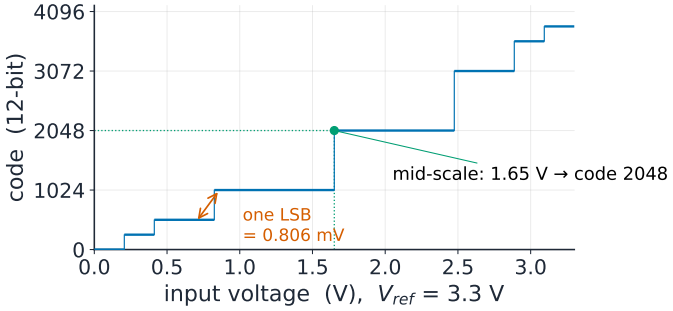


An 8N1 frame: 86.8 us per byte at 115 200 baud - the framing bits are part of the price.

ADC: resolution is pure arithmetic

1. **Mid-scale** 12-bit, 3.3 V: **LSB** 0.806 mV; 1.65 V reads exactly 2048 - mid-scale is the self-checking point.
2. **In units** Accelerometer 4 g over 4096 codes: 0.977 mg per LSB - 10x finer than a 10 mg requirement.
3. **Range** 100 uA through 10 k is 1.0 V - 30% of the range; 33 k uses it all and the LSB drops to 24 nA, if the source impedance still fits.

Every code owns one LSB

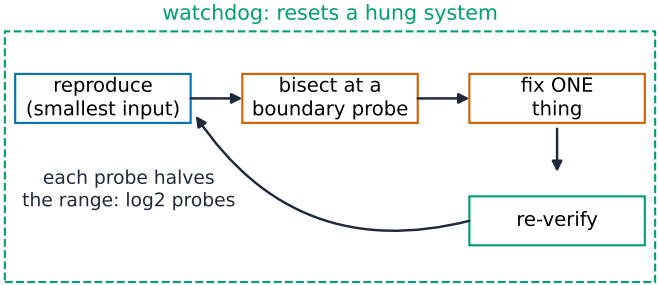


A 12-bit converter over 3.3 V: 0.806 mV per code, mid-scale 1.65 V reads exactly 2048.

Debugging: bisect, one fix, re-verify

1. **Hang** Main-loop LED stops, ISR LED ticks: the fault is in the main loop; a state print pins it to a timeout-less **I2C** wait - one-line fix, verified by the soak.
2. **Stack** Overwritten markers below the stack prove an overflow; the high-water mark finds what grew it.
3. **Contexts** 1 kHz control in the ISR (14%), **UART** bytes to a ring buffer parsed in the main loop (2.3%), display at 5 Hz - single-byte volatile flags need no mutex.

Reproduce, bisect, fix one, re-verify



Each probe halves the range: ten probes bound a thousand instructions; the **watchdog** is the outer net.

Check yourself

- **Timer**: 84 MHz to 1 Hz, 16-bit registers? (Factors 8400 and 10 000: PSC 8399, ARR 9999.)
- **UART**: 115200 8N1, one byte? (86.8 us.)
- **ADC**: 12-bit, 3.3 V, 1.65 V input? (Code 2048.)