Embedded C Cheat Sheet

AVR tinyAVR 0/1/2-Series • ATtiny1616 • MPLAB X • XC8 • KasperAero LLC

TYPES & ESSENTIAL INCLUDES

TYPE	MEANING
uint8_t / int8_t	8-bit Unsigned (0–255) / Signed (–128 to 127)
uint16_t / int16_t	16-bit Unsigned (0–65k) / Signed (–32k to 32k)
uint32_t	Unsigned 32-bit (0–4 billion)
bool	true / false — needs <stdbool.h>
float	32-bit floating point
static	Variable persists between calls
volatile	Tells compiler: value may change externally
const	Value cannot be changed after init

```
#include <avr/io.h>         // Port/register names
#include <util/delay.h>      // _delay_ms(), _delay_us()
#include <stdint.h>          // uint8_t, int16_t, etc.
#include <stdbool.h>         // bool, true, false
#define F_CPU 20000000UL     // MUST be before delay.h
```

PREPROCESSOR (TEXT SUBSTITUTION)

```
#define F_CPU 20000000UL // 20 MHz clock
#define PIN PIN3_bm      // bit mask for pin 3
#define ABS(x) ((x) < 0 ? -(x) : (x))
#define MAX(a,b) ((a) > (b) ? (a) : (b))
```

Runs before compilation. No memory allocated, no instructions generated. Replaces name with text value.

GPIO — PORT A, B, C PIN CONTROL

REGISTER / MACRO	ACTION
`PORTx.DIRSET = mask`	Set pin(s) as **OUTPUT**
`PORTx.DIRCLR = mask`	Set pin(s) as **INPUT**
`PORTx.OUTSET = mask`	Drive pin(s) **HIGH**
`PORTx.OUTCLR = mask`	Drive pin(s) **LOW**
`PORTx.OUTTGL = mask`	**Toggle** pin(s)
`PORTx.IN`	Read actual pin state (input)
`PORTx.PINnCTRL`	Per-pin config: pull-up, invert, etc.
`PORT_PULLUPEN_bm`	Enable internal pull-up resistor

```
// PIN naming — PIN0_bm is just 0x01
PORTB.OUTSET = PIN0_bm; // → PB0
PORTA.OUTSET = PIN0_bm; // → PA0

// Best practice: name both port AND pin
#define LED_PORT PORTB
#define LED_PIN PIN0_bm
LED_PORT.OUTSET = LED_PIN;

// Pull-ups (inputs):
PORTA.PIN1CTRL |= PORT_PULLUPEN_bm;  // PA1 pull-up ON
PORTA.PIN1CTRL &= ~PORT_PULLUPEN_bm; // PA1 pull-up OFF
```

MPLAB X & PICKIT 5 DEBUGGING

Variables	Window → Debug → Variables (Locals)
Watches	Add any var or expression (e.g. mT calc)
Visualizer	Tools → Data Visualizer (UART graph)
F7 / F8	Step Into / Step Over
F4 / Pause	Run to Cursor / Freeze execution

PICkit 5: Project Properties → PICkit 5 → Power target from PICkit 5. **Set to 3.3V for TMAG.**

BITWISE OPERATIONS

OP	PURPOSE	OP	PURPOSE
&	AND (mask/read)	<<	Shift Left (×2)
|	OR (set bits)	>>	Shift Right (÷2)
^	XOR (toggle)	&=	AND-assign (clear)
~	NOT (invert)	|=	OR-assign (set)

```
REG |= (1 << 4);  // Set bit 4
REG &= ~(1 << 4); // Clear bit 4
REG ^= (1 << 4);  // Toggle bit 4
if (REG & (1 << 4)) { ... } // Test bit 4

// Chained error check
bool ok = true;
ok &= write_reg(addr, REG1, 0x40);
ok &= write_reg(addr, REG2, 0x02);
```

TWI0 (I²C MASTER) — ATTINY1616

Pins: PA1 SDA, PA2 SCL. **MBAUD:** 95 → ~100kHz @ 20MHz

STATUS FLAGS	CHECK AFTER EACH BYTE
TWI_WIF_bm	Write complete (sent)
TWI_RIF_bm	Read complete (received)
TWI_RXACK_bm	Device NACKed (1=Bad)

```
// I2C Write Sequence
TWI0.MADDR = (dev_addr << 1) | 0x00;
while (!(TWI0.MSTATUS & TWI_WIF_bm));
if (TWI0.MSTATUS & TWI_RXACK_bm) return

TWI0.MDATA = reg_addr;
while (!(TWI0.MSTATUS & TWI_WIF_bm));

TWI0.MDATA = data;
while (!(TWI0.MSTATUS & TWI_WIF_bm));

TWI0.MCTRLB = TWI_MCMD_STOP_gc;
```

TMAG3001 HALL SENSOR REFERENCE

I²C Addresses: GND: 0x34 | VCC: 0x35 | SDA: 0x36 | SCL: 0x37

REG	PURPOSE	REG	PURPOSE
0x12/13	X MSB/LSB	0x18	Conv_Status
0x16/17	Z MSB/LSB	0x02	Sensor_Config

```
// 0x02 Config: 0x40 (Z only typical)
// Convert raw Z to milliTesla:
float z_mT = ((float)z_raw / 32767.0f) * 40.0f;
```

COMMON GOTCHAS

F_CPU mismatch	delay() wrong if F_CPU ≠ clock
Wrong PORT/PIN	PIN0_bm on PORTA ≠ PIN0_bm on PORTB
Read OUT vs IN	OUT shows drive, IN shows actual level
VCC > 3.6V (TMAG)	**5V will damage TMAG3001 sensor**
delay() in ISR	Never call _delay_ms in an interrupt