C Programming

Keys of Learning

- Syntax
  - Standard C
  - MCU-specific Extended C
    - Supported by their compilers
- Libraries
  - Standard C libraries
  - Extended processor C libraries
  - Peripheral libraries
  - Application libraries

Standard C

- Variables
  - Primitive: int, double, float, char
  - Pointers
- Operators: arithmetic, logic
- Statement
- Function
  - int main(void)
  - int main(int argc, char** argv)
- Preprocessing Directives (Macro)
  - #include
  - #define
- Good reference
  - https://www-s.acm.illinois.edu/webmonkeys/book/c_guide/

Standard C

- Often used in embedded
- Bitwise operators
- Variables
  - pointer
  - typedef
  - struct (with bit assignments)
  - union
  - enum

- Keywords for compilation
  - volatile
  - extern
  - const
  - register
  - inline
  - asm
- Preprocessing Directives (Macro)
  - #ifdef

Bitwise Operators

- Bitwise not: ~, ~=
- Bitwise or: |, |=
- Bitwise and: &, &=
- Bitwise xor: ^, ^=
- Left shift: <<, <<=
- Right shift: >>, >>=

Standard C

- pointer
  - Case 16
  - reference
    - &
  - dereference
    - *
  - array
  - NULL means 0
    - NULL is macro

```
int i;
int* j = &i;
int k[10];

int** a = &j;
int* b[10];
int c[10][6];

char* p=NULL; p++;
int* q=NULL; q++;


In atmega128,
int* x=(int*)0x20;
*x=100;
int* y=(int*)0x100;
*y=100;
int* z=(int*)0x10F0;
*z=100;
```

Standard C

- typedef
  - Case 16
  - For convenience
  - Mostly used with
    - struct
    - function pointer

```
typedef signed char int8_t;
int8_t foo;

typedef struct _sttype {
  int a;
  char b;
} STType;
STType foo;
foo.a = 10;
foo.b = 'x';

typedef char (*ftype)(int, char)
ftype foo;
char bar(int a, char b) {...}
foo=bar;
char y = bar(10, 'x');
char y = foo(10, 'x');
```

Standard C

- struct
  - Case 16
  - Bit assignment

```
typedef struct _sttype {
  int a;
  char b;
} STType;
STType foo;
foo.a = 10;
foo.b = 'x';

typedef struct tagCORCONBITS {
  unsigned :2;
  unsigned SFA:1;
  unsigned IPL3:1;
  unsigned :11;
  unsigned VAR:1;
} CORCONBITS;
CORCONBITS foo;
foo.SFA = 1;
foo.IPL3 = 1;
```

Standard C

- union
  - Case 16
  - Memory sharing

```
typedef struct tagIC1CON2BITS {
  union {
    struct {
      unsigned SYNCSEL:5;
      unsigned :1;
      unsigned TRIGSTAT:1;
      unsigned ICTRIG:1;
      unsigned IC32:1;
    };
    struct {
      unsigned SYNCSEL0:1;
      unsigned SYNCSEL1:1;
      unsigned SYNCSEL2:1;
      unsigned SYNCSEL3:1;
      unsigned SYNCSEL4:1;
    };
  };
} IC1CON2BITS;
IC1CON2BITS foo;
foo.SYNCSEL = 0x1F;
foo.SYNCSEL0 = 1;
```

Standard C

- enum
  - Case 16
  - Define a list of variables and their values
  - All variables are integers

```
typedef enum {
    ST_OFF = 0,
    ST_ON = 1,
    ST_SAMPLE = 2,
    ST_PROCESS = 3,
    ST_PAUSE = 4,
} STATUS;
STATUS foo;
foo = ST_OFF
```

Standard C

- extern
  - Case 16
  - Indicate the variable is defined outside of the current file

In any *.h

```
extern int foo;
```

In one and only one .c file and outside any function

```
int foo = 10;
```

Standard C

- volatile
  - Case 16
  - Indicate the variable may be modified by outside routines and thus shall not be omitted by optimization
    - Variables that could be changed by interrupts
  - `make volatile`

```
#include <stdint.h>

int main() {

    uint8_t status1 = 0;
    while (status1) return 1;

    uint8_t volatile status2 = 0;
    while (status2) return 2;

    return 0;
}
```

Standard C

- const
  - Case 16
  - type const variable
  - const arguments in function

```
const int i;
int const i;

const char * str;
char const * str;

char * const str;

const char * const str;
char const * const str;
```

Standard C

- register
  - Case 16
  - Requests that the variable be stored in register
  - But, not guaranteed by compiler
  - `make register`

```
#include <stdint.h>

int main() {
    register uint16_t i;
    uint16_t j;
    for (i=0; i<10; i++) j+=i;
    return j;
}
```

Standard C

- #ifdef
  - Case16

```
#define __PIC24EP512GU810__

#if defined(__PIC24EP512GP806__)
#include <p24EP512GP806.h>
#endif

#if defined(__PIC24EP512GU810__)
#include <p24EP512GU810.h>
#endif

#if defined(XXX)
#ifdef XXX
```

Embedded C

- Integer types
- Registers
  - Structure/union of registers
  - How to access registers
- Inline assembly
- MCU-specific libraries
  - How to access program memory

Example: XC16

- Constants

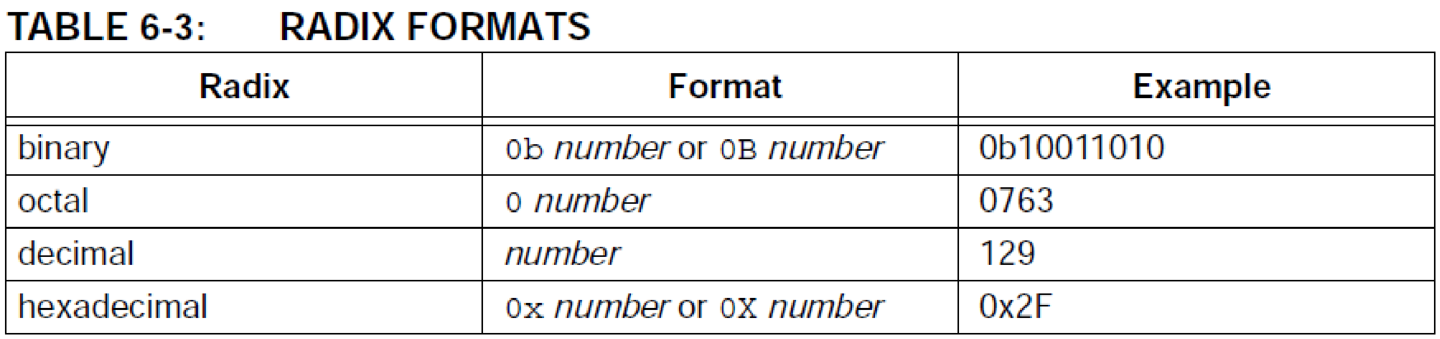

TABLE 6-3: RADIX FORMATS

Radix	Format	Example
binary	`0b` *number* or `0B` *number*	0b10011010
octal	`0` *number*	0763
decimal	*number*	129
hexadecimal	`0x` *number* or `0X` *number*	0x2F

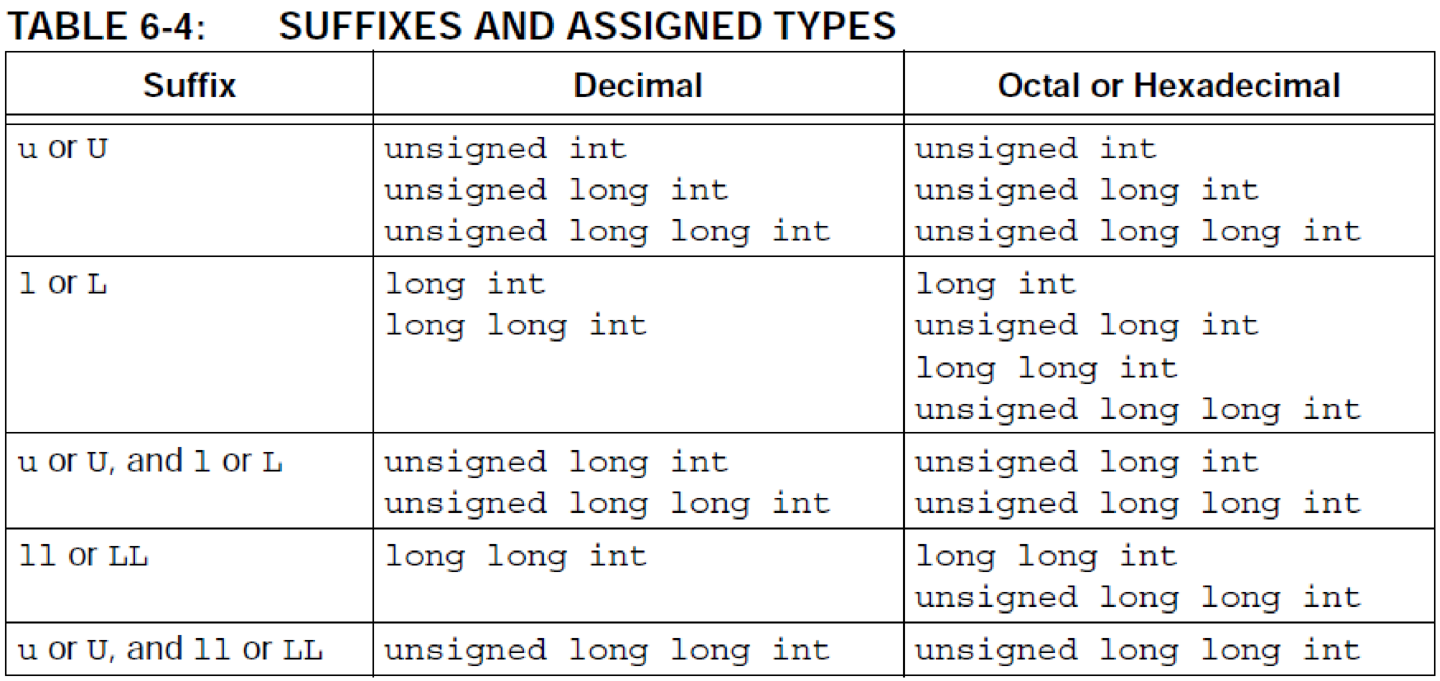

TABLE 6-4: SUFFIXES AND ASSIGNED TYPES

Suffix	Decimal	Octal or Hexadecimal

`u` or `U`	`unsigned int`
	`unsigned long int`
	`unsigned long long int`
		`unsigned int`
		`unsigned long int`
		`unsigned long long int`

`l` or `L`	`long int`
	`long long int`
		`long int`
		`unsigned long int`
		`long long int`
		`unsigned long long int`

`u` or `U`, and `l` or `L`	`unsigned long int`
	`unsigned long long int`
		`unsigned long int`
		`unsigned long long int`

`ll` or `LL`	`long long int`	`long long int`
		`unsigned long long int`

`u` or `U`, and `ll` or `LL`	`unsigned long long int`	`unsigned long long int`

Example: XC16

- Implementation-defined Behavior
  - Integers: int in C is 32-bit in x86.
  - But, ...

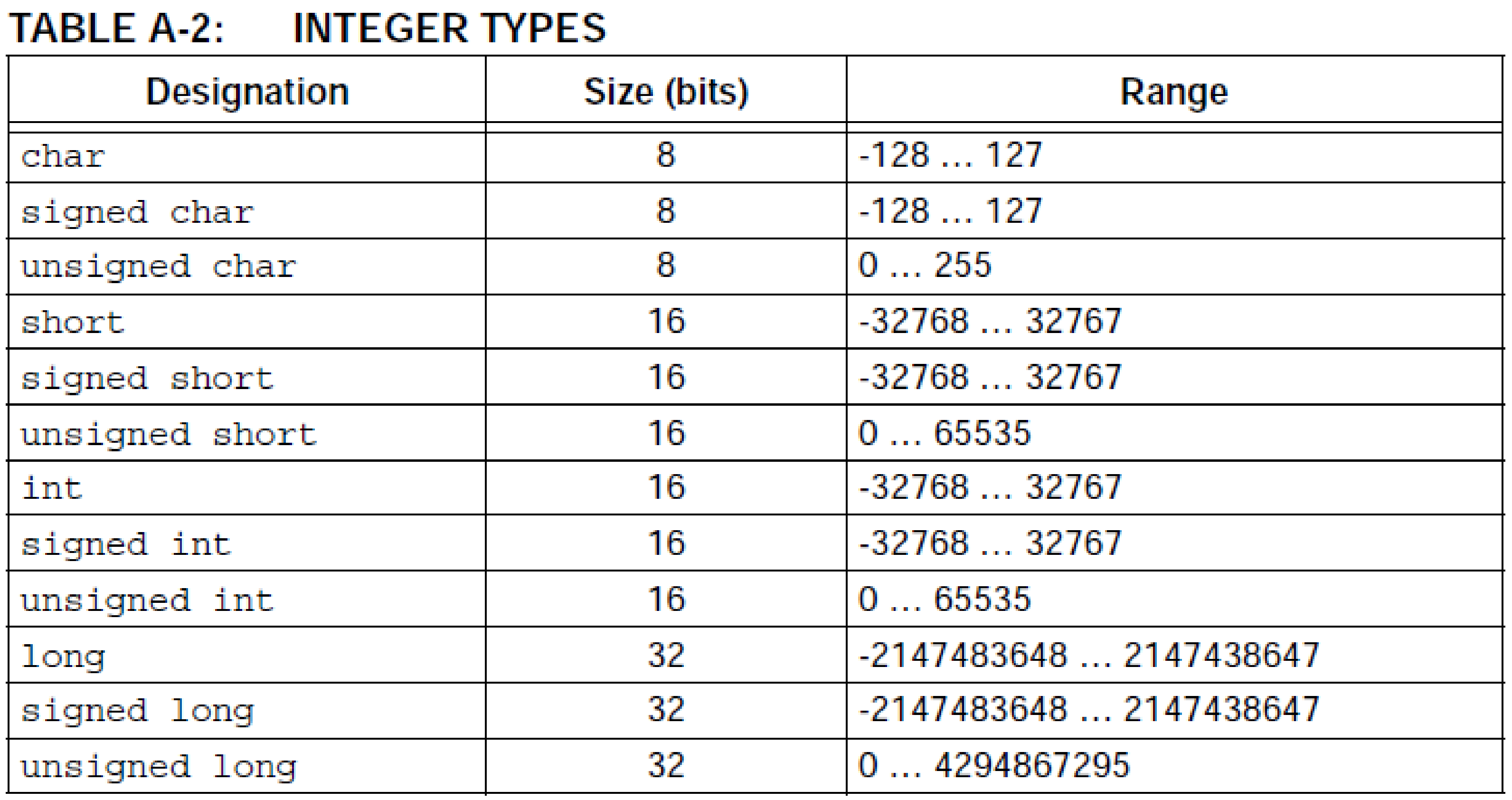

TABLE A-2: INTEGER TYPES

Designation	Size (bits)	Range
`char`	8	-128 … 127
`signed char`	8	-128 … 127
`unsigned char`	8	0 … 255
`short`	16	-32768 … 32767
`signed short`	16	-32768 … 32767
`unsigned short`	16	0 … 65535
`int`	16	-32768 … 32767
`signed int`	16	-32768 … 32767
`unsigned int`	16	0 … 65535
`long`	32	-2147483648 … 2147438647
`signed long`	32	-2147483648 … 2147438647
`unsigned long`	32	0 … 4294867295

Explicit Integer Type

- Signed and unsigned
- Number of bits
- int8_t, uint8_t
- int16_t, uint16_t
- int32_t, uint32_t
- int64_t, uint64_t
- Rather than using
  - char, short, int, long

MCU-specific C

- Example: Avr-gcc
  - https://www.nongnu.org/avr-libc/user-manual/index.html
- Data in program space
  - #include <avr/pgmspace.h>
  - var_declaration PROGMEM = ...;
  - byte = pgm_read_byte(&(var[index]));
- Inline
  - inline void foo();
- Assembly
  - asm(code : output operand list : input operand list [: clobber list]);

MCU-specific C Library

- CRC: Cyclic redundancy check
- https://www.nongnu.org/avr-libc/user-manual/group__util__crc.html

```
#include <util/crc16.h>
uint16_t getcrc(uint8_t data[], uint8_t size) {
    uint16_t crc = 0
    uint8_t i;
    for (i=0; i<size; i++)
        crc = _crc16_update(crc, data[i]);
    return crc;
}
```

MCU-specific C

- Example: XC16
- Variable attributes
  - __attribute__
  - int x __attribute__ ((aligned (16))) = 0;
- Function attributes
  - void __attribute__ ((address(0x100))) foo();
  - void __attribute__((interrupt,auto_psv)) isr0(void);

Example: XC16

- Implementation-defined Behavior
  - #pragma

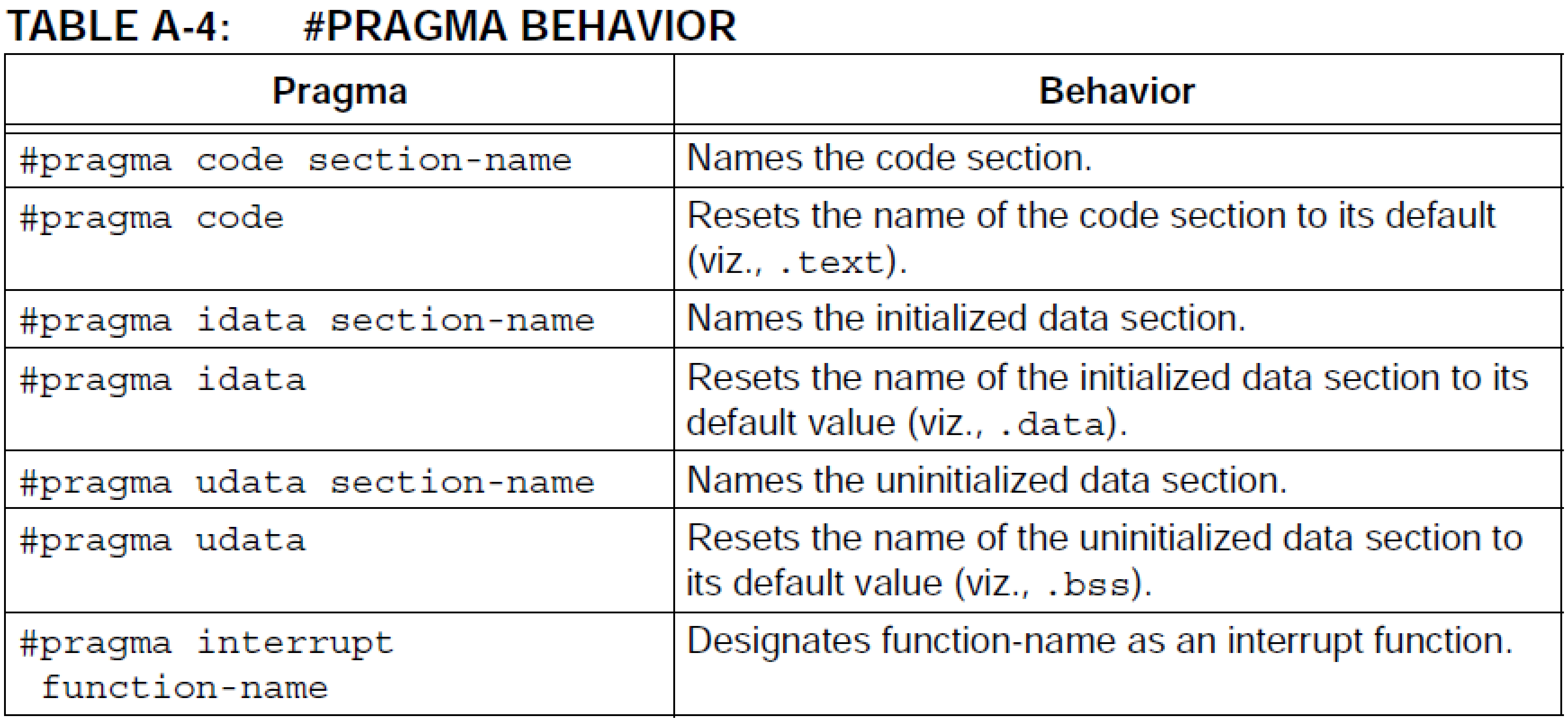

TABLE A-4: #PRAGMA BEHAVIOR

Pragma	Behavior
`#pragma code section-name`	Names the code section.
`#pragma code`	Resets the name of the code section to its default (viz., `.text`).
`#pragma idata section-name`	Names the initialized data section.
`#pragma idata`	Resets the name of the initialized data section to its default value (viz., `.data`).
`#pragma udata section-name`	Names the uninitialized data section.
`#pragma udata`	Resets the name of the uninitialized data section to its default value (viz., `.bss`).
`#pragma interrupt function-name`	Designates function-name as an interrupt function.

Libraries

- Example: Avr-gcc
  - C libs: stdxxx.h, string.h, math.h
  - Avr libs:
    - eeprom.h
    - interrupt.h
    - io.h
    - pgmspace.h
    - power.h
    - sleep.h
    - delay.h
    - crc16.h
    - atomic.h
    - sfr_defs.h

Libraries

- Example: xc16-gcc
- Standard C functions
  - Functions: stdio.h, stdlib.h, math.h, assert.h, string.h, time.h, ...
  - Types: stdbool.h, stdint.h
- Peripheral Libraries
  - Timer, I/O, UART, SPI, I2C, ADC, DMA, Reset, CRC, RTCC,
- Application libraries
  - USB, Graphics, Memory Disk Drive, TCP/IP Stack, mTouchCap, Smart Card, MiWi