Introduction to Microcontrollers

starring the dsPIC33EP128GP502

Peter Loux Sr.

Microprocessor vs. Microcontroller

- Microprocessor
  - General purpose computing
  - Need external components like memory, I/O
- Microcontroller
  - Integrated on single chip
  - Optimized for specific tasks in embedded computing
  - cheaper

Comparison of Systems

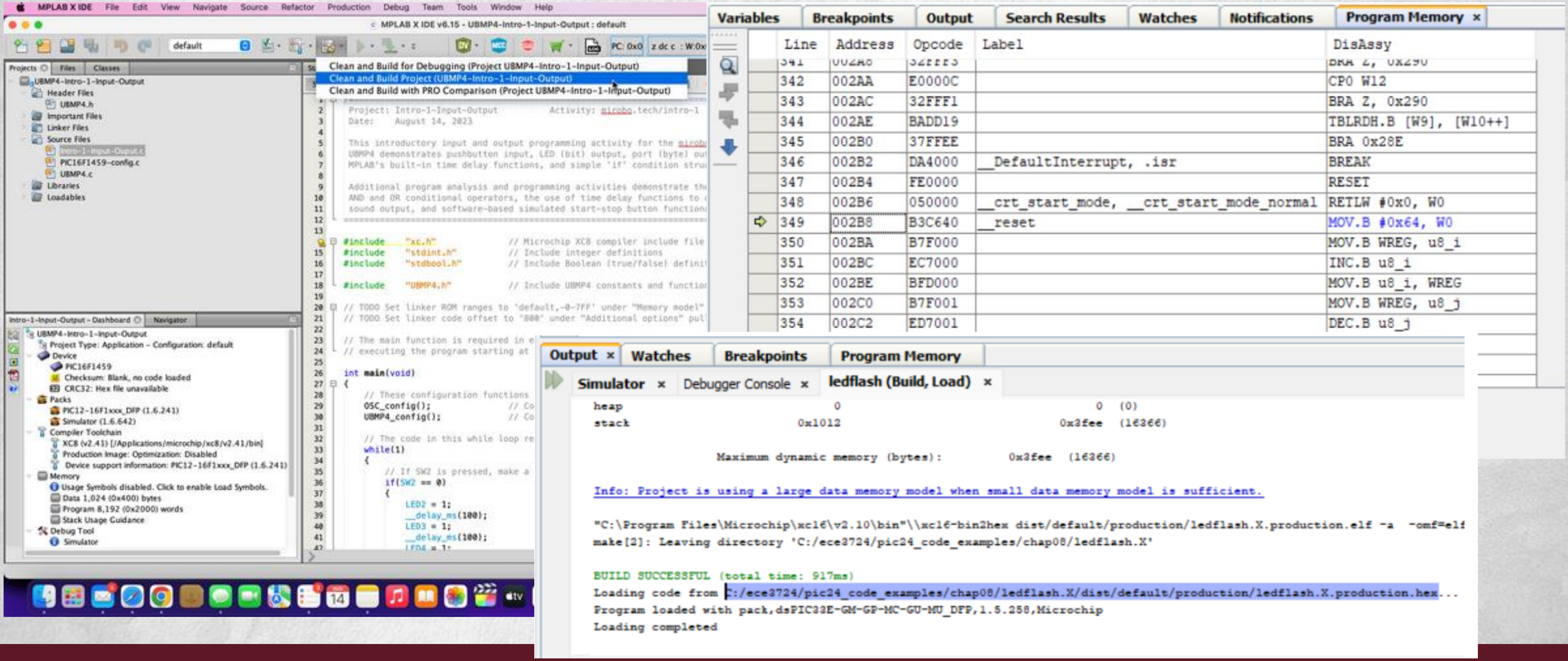
Name	Year	IPS	Memory*	Storage*	Price**
IBM System/360 Model 30	1964	34,500	64 KB	~113 MB per tape reel	??? / $133,000
MOS Technology 6502 (Apple II)	1977	430,000	20 KB	140 KB per floppy disk	$25 / $1298
Intel Pentium	1994	112 million	8 MB	500 MB HDD	$400 / $2500
Intel i5-11600K	2021	346 billion	16 GB	2 TB SSD	$200 / $1200 (as of 2025)
Espressif Systems ESP8266	2014	90 million	32 KB	80 KB	$5.46
Microchip dsPIC33EP128GP502	2011	70 million	128 KB	16 KB	$4.37

* For microprocessors, a typical configuration is given. ** CPU price / system price (not inflation-adjusted)

Why the PIC?

- Internals similar for purposes of this course
- Chip only vs other components
- Built-in bootloader vs Bully
- Downloadable libraries vs book code from 2011
- Variety
- Price
- Experience

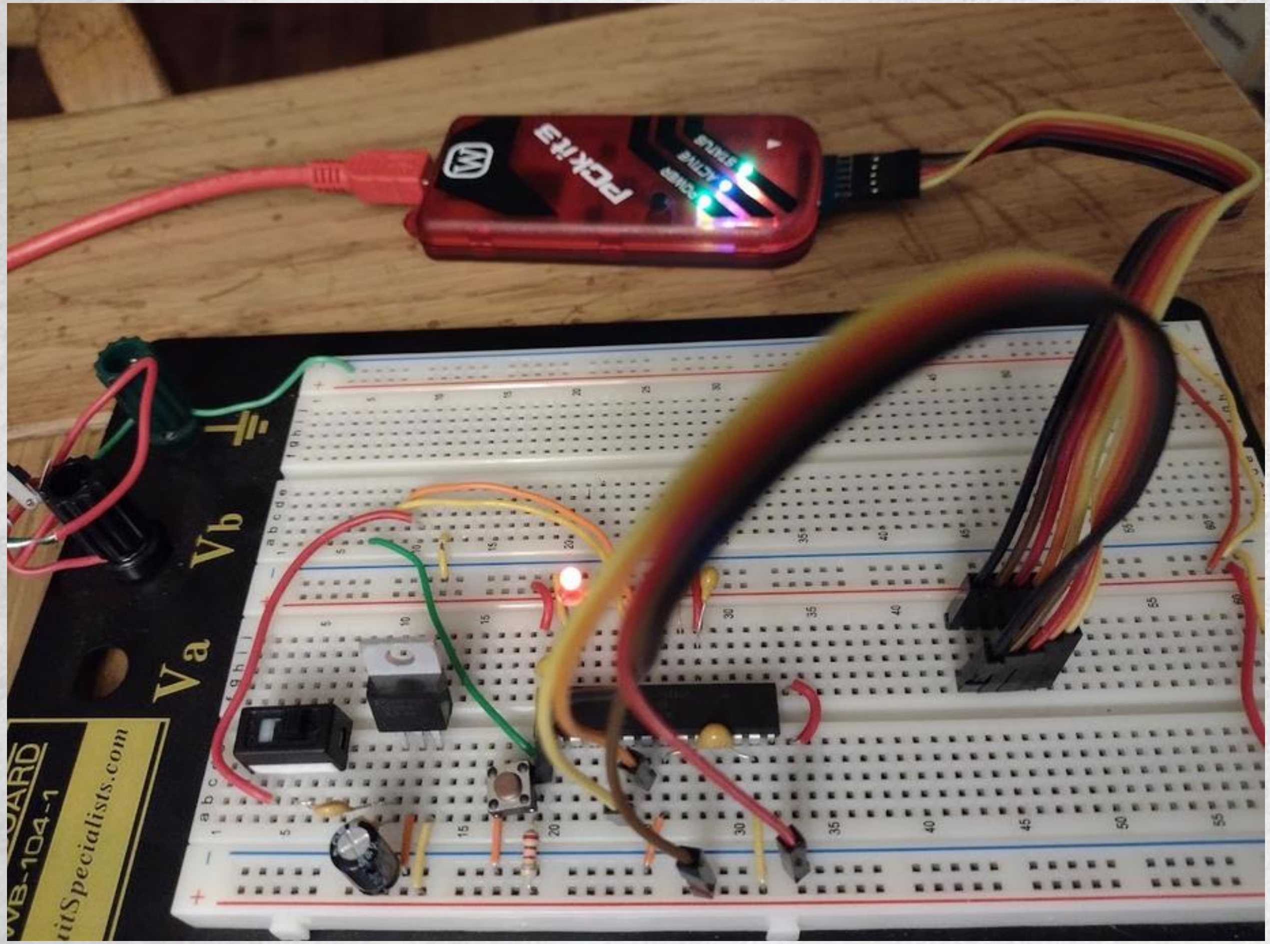
MPLAB

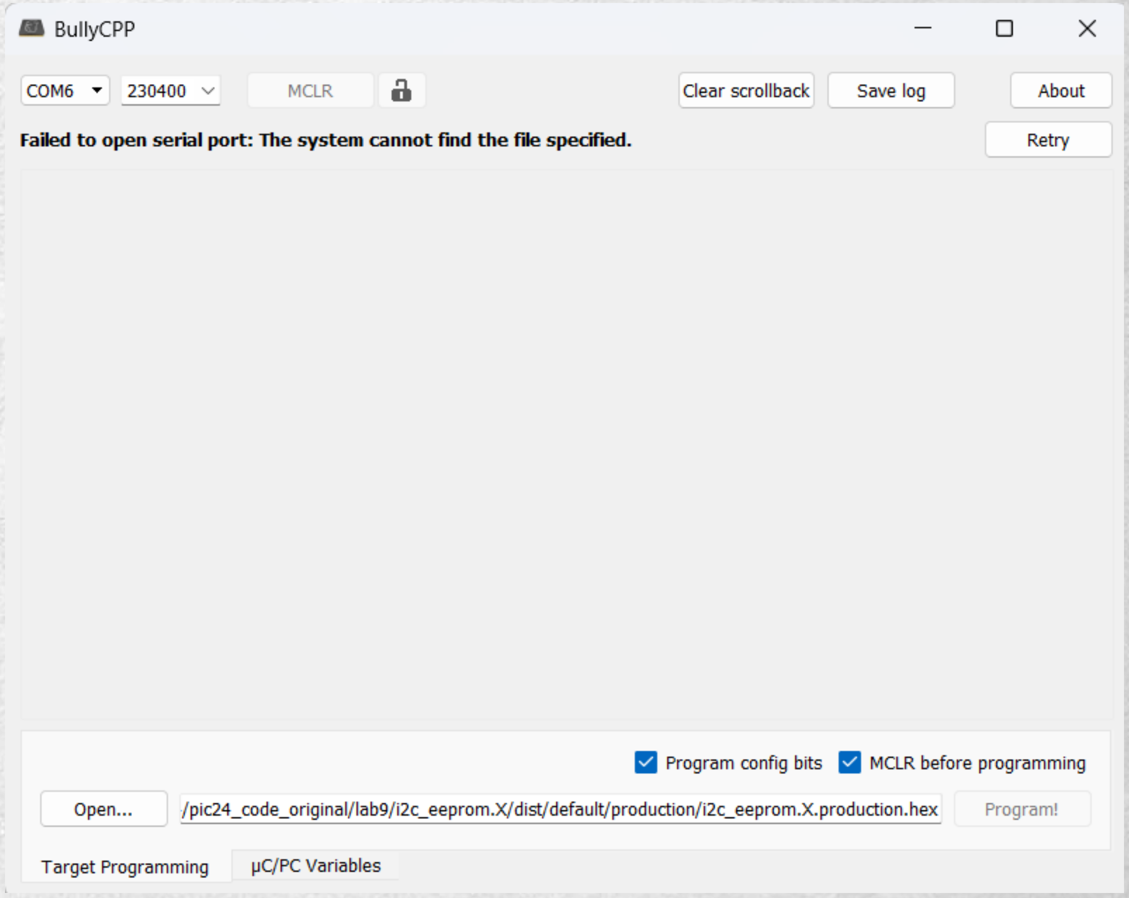

MPLAB X IDE
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
MPLAB X IDE v6.15 - UBMP4-Intro-1-Input-Output : default
Clean and Build for Debugging (Project UBMP4-Intro-1-Input-Output)
Clean and Build Project (UBMP4-Intro-1-Input-Output)
Clean and Build with PRO Comparison (Project UBMP4-Intro-1-Input-Output)
Projects Files Classes
UBMP4-Intro-1-Input-Output
Header Files
UBMP4.h
Important Files
Linker Files
Source Files
PIC16F1459-config.c
UBMP4.c
Libraries
Loadables
Variables Breakpoints Output Search Results Watches Notifications Program Memory
Line Address Opcode Label DisAssy
342 002AA E0000C CP0 W12
343 002AC 32FFF1 BRA Z, 0x290
344 002AE BADD19 TBLRDH.B [W9], [W10++]
345 002B0 37FFEE BRA 0x28E
346 002B2 DA4000 __DefaultInterrupt, .isr BREAK
347 002B4 FE0000 RESET
348 002B6 050000 __crt_start_mode, __crt_start_mode_normal RETLW #0x0, W0
349 002B8 B3C640 __reset MOV.B #0x64, W0
350 002BA B7F000 MOV.B WREG, u8_i
351 002BC EC7000 INC.B u8_i
352 002BE BFD000 MOV.B u8_i, WREG
353 002C0 B7F001 MOV.B WREG, u8_j
354 002C2 ED7001 DEC.B u8_j
Output Watches Breakpoints Program Memory
Simulator Debugger Console ledflash (Build, Load)
heap 0 0 (0)
stack 0x1012 0x3fee (16366)
Maximum dynamic memory (bytes): 0x3fee (16366)
Info: Project is using a large data memory model when small data memory model is sufficient.
"C:\Program Files\Microchip\xc16\v2.10\bin"\\xc16-bin2hex dist/default/production/ledflash.X.production.elf -a -omf=elf
make[2]: Leaving directory 'C:/ece3724/pic24_code_examples/chap08/ledflash.X'
BUILD SUCCESSFUL (total time: 917ms)
Loading code from C:/ece3724/pic24_code_examples/chap08/ledflash.X/dist/default/production/ledflash.X.production.hex...
Program loaded with pack,dsPIC33E-GM-GP-MC-GU-MU_DFP,1.5.258,Microchip
Loading completed

The Compiler

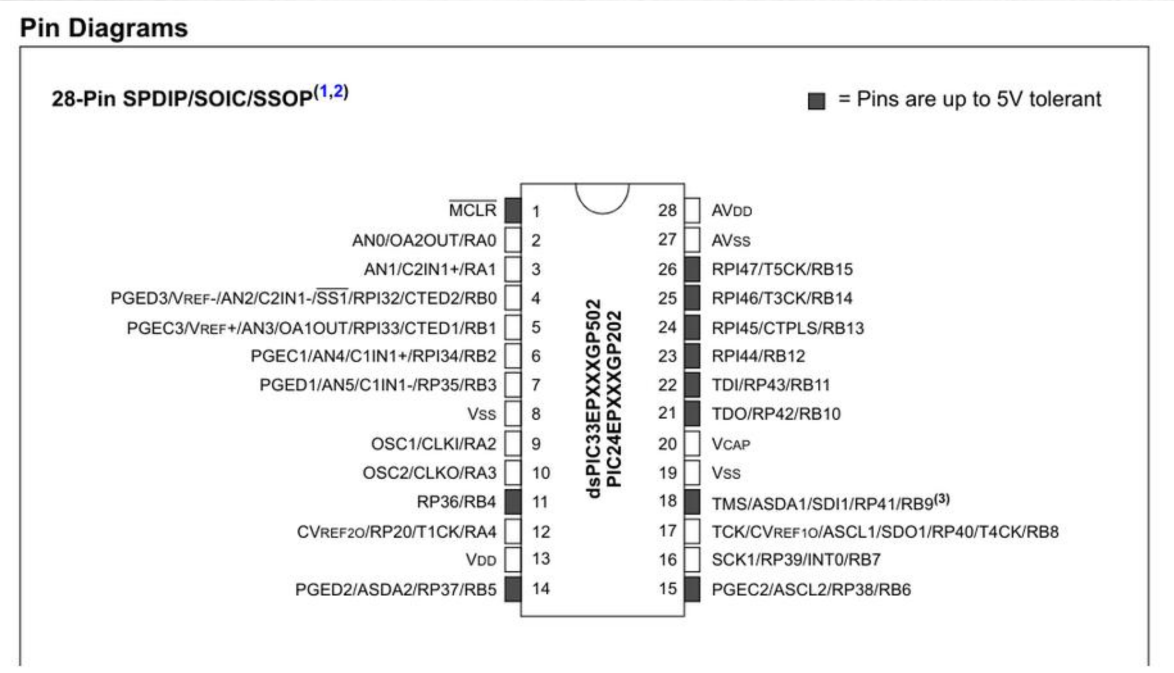
-

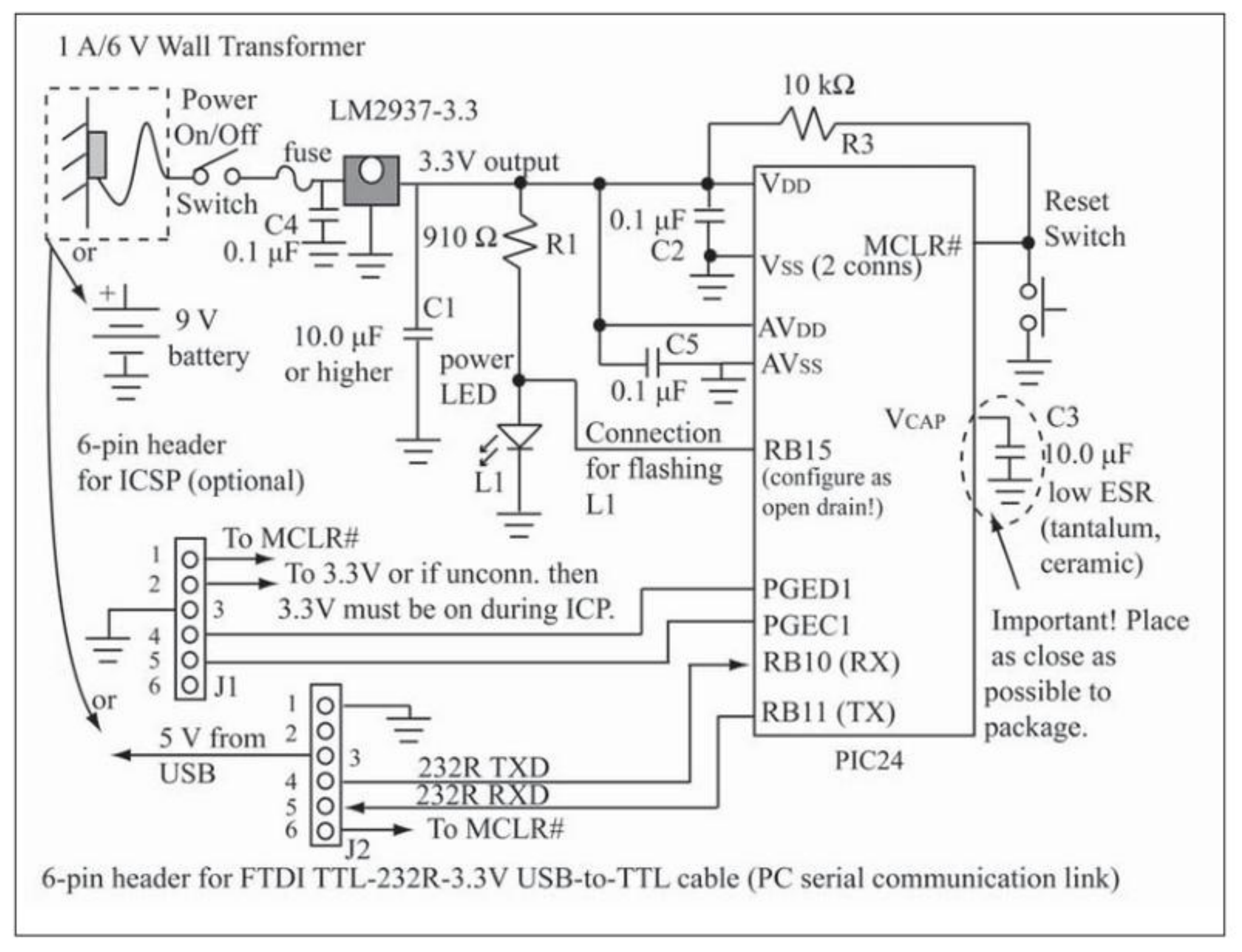
The Bootloader





The Circuit


Pin Diagrams
28-Pin SPDIP/SOIC/SSOP(1,2)
= Pins are up to 5V tolerant
MCLR 1
AN0/OA2OUT/RA0 2
AN1/C2IN1+/RA1 3
PGED3/VREF-/AN2/C2IN1-/SS1/RPI32/CTED2/RB0 4
PGEC3/VREF+/AN3/OA1OUT/RPI33/CTED1/RB1 5
PGEC1/AN4/C1IN1+/RPI34/RB2 6
PGED1/AN5/C1IN1-/RP35/RB3 7
VSS 8
OSC1/CLKI/RA2 9
OSC2/CLKO/RA3 10
RP36/RB4 11
CVREF2O/RP20/T1CK/RA4 12
VDD 13
PGED2/ASDA2/RP37/RB5 14
28 AVDD
27 AVSS
26 RPI47/T5CK/RB15
25 RPI46/T3CK/RB14
24 RPI45/CTPLS/RB13
23 RPI44/RB12
22 TDI/RP43/RB11
21 TDO/RP42/RB10
20 VCAP
19 VSS
18 TMS/ASDA1/SDI1/RP41/RB9(3)
17 TCK/CVREF1O/ASCL1/SDO1/RP40/T4CK/RB8
16 SCK1/RP39/INT0/RB7
15 PGEC2/ASCL2/RP38/RB6
dsPIC33EPXXXGP502
PIC24EPXXXGP202


1 A/6 V Wall Transformer
Power On/Off Switch
LM2937-3.3
fuse
3.3V output
10 kΩ
R3
Reset Switch
VDD
MCLR#
Vss (2 conns)
AVDD
AVSS
or
C4
0.1 µF
910 Ω
R1
0.1 µF
C2
9 V battery
10.0 µF or higher
C1
power LED
C5
0.1 µF
L1
Connection for flashing L1
RB15 (configure as open drain!)
VCAP
C3
10.0 µF low ESR (tantalum, ceramic)
Important! Place as close as possible to package.
6-pin header for ICSP (optional)
To MCLR#
To 3.3V or if unconn. then 3.3V must be on during ICP.
J1
PGED1
PGEC1
RB10 (RX)
RB11 (TX)
PIC24
5 V from USB
232R TXD
232R RXD
To MCLR#
J2
6-pin header for FTDI TTL-232R-3.3V USB-to-TTL cable (PC serial communication link)

Flashing an LED

```
#include "pic24_all.h"

/**
A simple program that flashes the Power LED.
*/

// A naive software delay function
void a_delay(void){
  uint16_t u16_i,u16_k;
  // change count values to alter delay
  for (u16_k = 1800; --u16_k;) {
    for (u16_i = 1200 ; --u16_i ;);
  }
}

int main(void) {
  configClock();     //clock configuration
  /********** PIO config **********/
  _ODCB15 = 1;          //enable open drain
  _TRISB15 = 0;         //Config RB15 as output
  _LATB15 = 0;          //RB15 initially low
  while (1) {           //infinite while loop
   a_delay();           //call delay function
   _LATB15 = !_LATB15;  //Toggle LED attached to RB15
  } // end while (1)
}
```

Includes several header files, discussed later in this chapter.

A subroutine for a software delay. Change `u16_i`, `u16_k` initial values to change delay.

Infinite loop that blinks the LED. Only exit is through MCLR# reset or power cycle.

Figure 8.4
chap08\ledflash_nomacros.c: C code for flashing an LED

```
#include "pic24_all.h"

void config_led1(void) {
  CONFIG_RB15_AS_DIG_OUTPUT();
  ENABLE_RB15_OPENDRAIN();
}

// _LATB15 is port register for RB15.
#define LED1 (_LATB15)

int main(void) {
  configClock();
  config_led1();
  LED1 = 0;
  while (1) {
    DELAY_MS(250);
    LED1 = !LED1;
  } // end while (1)
}
```

LED1 macro makes changing of LED1 pin assignment easier, also improves code clarity.

Figure 8.5
chap08\ledflash.c: Improved code example for flashing an LED

UART

```
#include "pic24_all.h"
// "Echo" program which waits for UART RX character and echos it back +1.
// Use the echo program to test your UART connection.

int main(void) {
  uint8_t u8_c;

  configClock();
  configHeartbeat();
  configDefaultUART(DEFAULT_BAUDRATE);
  printResetCause();
  outString(HELLO_MSG);

  /** Echo code ********/
  // Echo character + 1
  while (1) {
    u8_c = inChar();    //get character
    u8_c++;             //increment the character
    outChar(u8_c);      //echo the character
  } // end while (1)
}
```

`configHeartbeat(void)` function defined in *lib\common\pic24_util.c*. Configures heartbeat LED by default on RB15.

`configDefaultUART(uint32_t u32_baudRate)` function defined in *lib\common\pic24_uart.c*. This initializes the UART1 module for our reference system.

`printResetCause(void)` function defined in *common\pic24_util.c*. Prints info string about reset source.

`outString(char* psz_s)` function defined in *common\pic24_serial.c*. Sends string to UART. `HELLO_MSG` macro default is file name, build date.

Figure 8.6
chap08\echo.c: Program for testing the serial link

Power Modes

Power Saving Modes

- PIC provides several different power saving modes.
- Sleep mode:
  A. CPU and all peripherals stop working.
  B. Can be awoke by the WDT timeout and external interrupt.
  C. Use the pwrsav #0 instruction to enter the sleep mode.
- Idle mode:
  A. CPU stop working.
  B. Peripherals can still work (e.g., receive data through UART).
  C. Use the pwrsav #1 instruction to enter the idle mode.
- Doze mode:
  A. CPU and peripherals still work.
  B. Main clock to CPU is divided by doze prescaler (2, 4, ..., 128)
  C. Peripheral clocks unaffected. CPU runs slower, but peripherals run at full speed

Reset Types

- There are many reasons can cause a reset. We can check each bit in RCON register to know what type reset occurred
- All Reset flag bit in RCON may be set or cleared by the user software

Flag Bit	Set by:	Cleared by:
TRAPR (RCON<15>)	Trap conflict event	POR, BOR
IOPUWR (RCON<14>)	Illegal opcode or initialized W register access	POR, BOR
CM (RCON<9>)	Configuration Mismatch	POR,BOR
EXTR (RCON<7>)	MCLR# Reset	POR
SWR (RCON<6>)	`reset` instruction	POR, BOR
WDTO (RCON<4>)	WDT time-out	`pwrsav` instruction, `clrwdt` instruction, POR,BOR
SLEEP (RCON<3>)	`pwrsav #0` instruction	POR,BOR
IDLE (RCON<2>)	`pwrsav #1` instruction	POR,BOR
BOR (RCON<1>)	BOR	n/a
POR (RCON<0>)	POR	n/a

Power on reset

Brown-out reset (When Vdd drops below a threshold)