LCDs in C

ECE 376 Embedded Systems

Jake Glower - Lecture #9

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

LCDs in C

LEDs

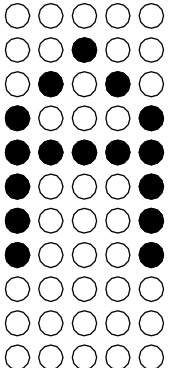
- Output binary data (on/off)
- Fast
- Hard to convey info

Graphic LCD

- Control over each pixel
- More versitile
- Requires more coding

Character LCD

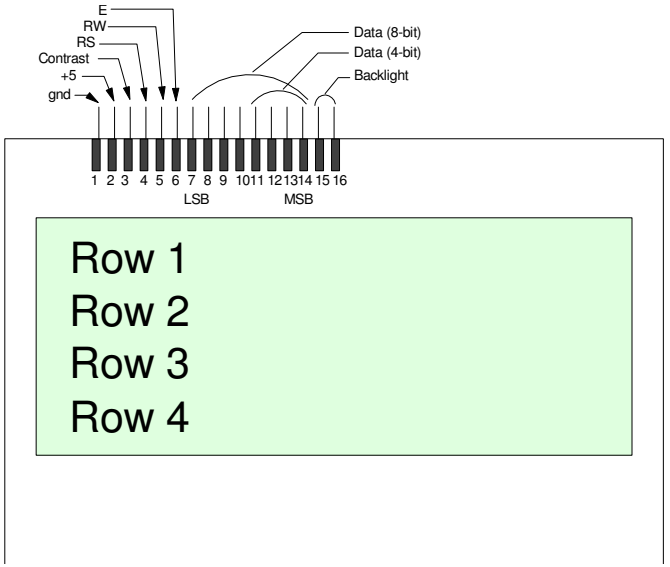
- On-board computer translates characters to graphics
- Easier to display info
- Slower



LCD I/O

- Pretty much standard for all LCD displays

Pin	Description
Ground, +5	Power for the LCD. Note: connecting these backwards will destroy the LCD.
Contrast:	0 to 5V signal for the 'brightness' of the display.
RS	Register Select. 1 = an instruction (such as blink the cursor) 0 = data (such as display 'A')
RW	Read / Write 0 = write to the LCD 1 = read data from the LCD
E	Clock. Data or instructions are read in when E is pulsed.
Data 0:7	in 8-bit mode, each byte is read in 8-bits at a time
Data 4:7	in 4-bit mode, each byte is read in two nibbles: left nibble first (MSB), right nibble last (LSB)
Backlight:	0 to 5V (sometimes 12V) to turn on the backlight (if available)



LCD Instrucion Set:

Instruction	RS	R/W	Data msb ..... lsb	Description
Clear Display	0	0	0000 0001	Clears display and returns cursor to home position (address 0). Execution time: 1.64ms
Home Cursor	0	0	0000 001x	Returns cursor to home position, returns a shifted display to original position. Display data RAM (DD RAM) is unaffected. Execution time: 40us to 1.64ms
Entry Mode Set	0	0	0000 01is	Sets cursor move direction and specifies whether or not to shift display. Execution time: 40us
On / Off Control	0	0	0000 1dcb	Turn display on or off, turn cursor on or off, blink character at cursor on or off. Execution time: 40us
Cursor Shift	0	0	0001 srxx	Move cursor or scroll display without changing display data RAM. Execution time: 40us
Function Set	0	0	001d nfxx	Set interface data length, mode, font.
Write Data to CD or DD RAM	1	0	dddd dddd	Data is written to current cursor position and (DD/CG) RAM address

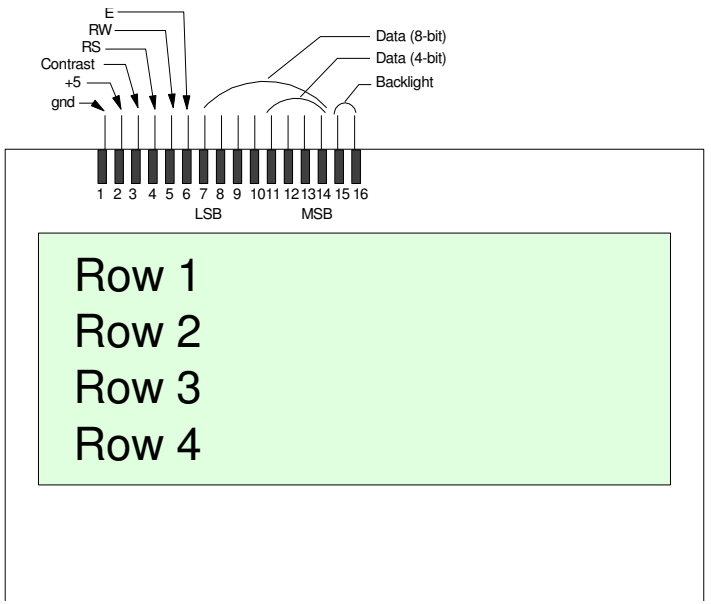
Initialization: 4-Bit Mode

Step	RS	R/W	D7:D4	then wait...	Comment
				15ms	Power On
1	0	0	0011	4.1 ms	
2	0	0	0011	100us	
3	0	0	0011	4.1ms	
4	0	0	0010	40us	4-bit mode
5	0	0	0010		
6	0	0	1Fxx	40us	F = font. 1 = 5x11, 0 for 5x8
7	0	0	0000		
8	0	0	1000	40us	Display off, cursor off, blink off
9	0	0	0000		
10	0	0	0001	1.6ms	Clear screen, cursor home
11	0	0	0000		
12	0	0	0110	40us	Increment cursor to the right when writing. Don't shift the screen.
Initialization Complete					

Place an 'AB' at location (2,6) (address 0x96)

Address = Row + Column

Row	Address of Col #0 (16xN LCD)	Address of Col #0 (20xN LCD)
0	0x80	0x80
1	0xC0	0xC0
2	0x90	0x94
3	0xD0	0xD4



Procedure:

Step	RS	R/W	D7:D4	then wait...	Comment
1	0	0	9		
2	0	0	6	40us	Move the cursor to row 2 column 6
3	1	0	4		
4	1	0	1	40us	Write 'A' (ascii 65 or 0x41) to the current position. Move one column to the right.
5	1	0	4		
6	1	0	2	40us	Write 'B' (ascii 66 of 0x42) to the current position. Move one column to the right.

LCD Routines:

Assume PORTD is used with

PORTD	7	6	5	4	3	2	1	0
LCD	D7	D6	D5	D4	E	R/S	R/W	-

void Wait_ms(unsigned int X)

- Pause Xms then return.

void LCD_Init(void)

- Initialize the LCD display, set the cursor to go from left to right, set the cursor to blink, move to top left corner.

void LCD_Move(unsigned char R, C)

- Move the cursor to row R column C.

void LCD_Write(unsigned char DATA)

- DATA written to the present position on the LCD display. Move the cursor one to the right.

void LCD_Out(unsigned int DATA, unsigned char D, unisgned char N)

- DATA written to the display
- D: The number of digits to display
- N: The number of digits to the right of the decimal point to display

For example,

LCD_Out(12345, 6, 3)

outputs

```
012.345
```

void LCD_Inst(0x01);

- clear the LCD display

Example 1: Write a routine to send

- 0..19 to the LCD display starting at row 0, column 0.
- 48..67 to the LCD display starting at row 1, column 0

```
// Global Variables

// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include         "lcd_portd.c"

void main(void)
{
   unsigned char i;

   LCD_Init();

   LCD_Move(0,0);
   for (i=0; i<20; i++) LCD_Write(i);
   LCD_Move(1,0);
   for (i=48; i<68; i++) LCD_Write(i);
   while(1);
   }
```

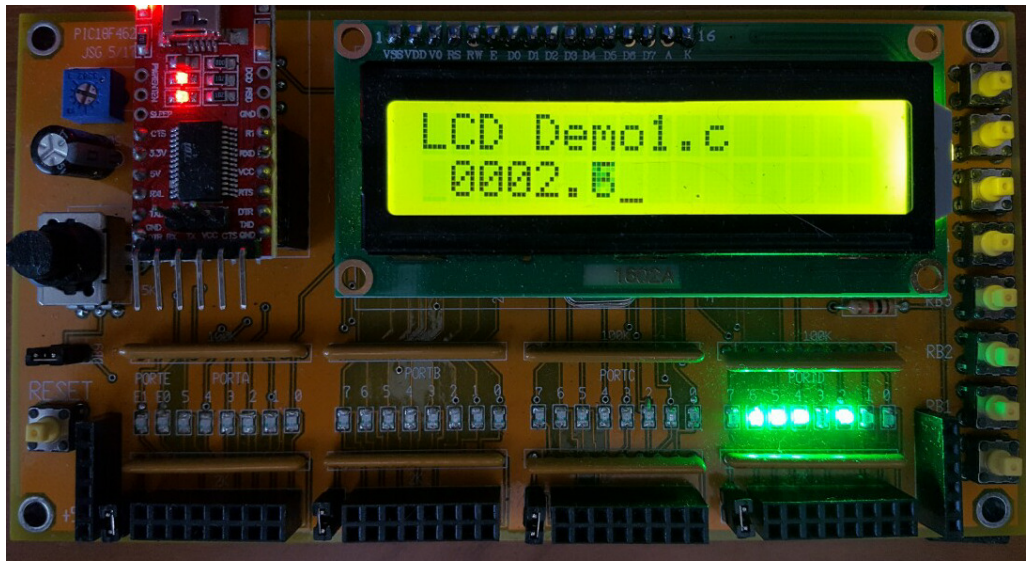
ASCII Table

- **www.ASCIITable.com**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T

Example 2: Count as fast as you can

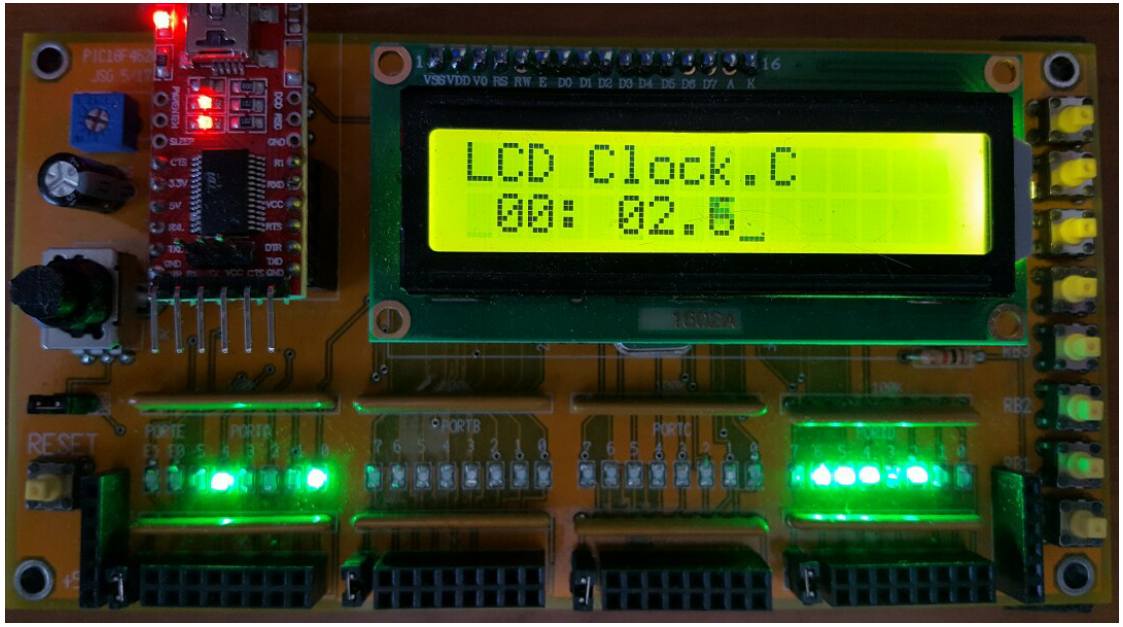
```
#include <pic18.h>
#include         "lcd_portd.c"

void main(void)
{
    unsigned int COUNT;
    unsigned int i;

    ADCON1 = 0x0F;

    LCD_Init();

    COUNT = 0;

    Wait_ms(100);

    while(1) {
        COUNT = COUNT + 1;
        LCD_Move(1,0);
        LCD_Out(COUNT, 5, 1);
        }
}
```



Example 3: Count every 100ms. Display time in seconds

Count every 100ms

```
    while(1) {
        LCD_Move(1,0);
        LCD_Out(MIN,5,0);
        LCD_Write(':');
        LCD_Out(SEC,5,1);

        SEC = SEC + 1;
        RA0 = 1;
        Wait_ms(100);
        RA0 = 0;
        }
}
```



Timing:

RA1 = 1

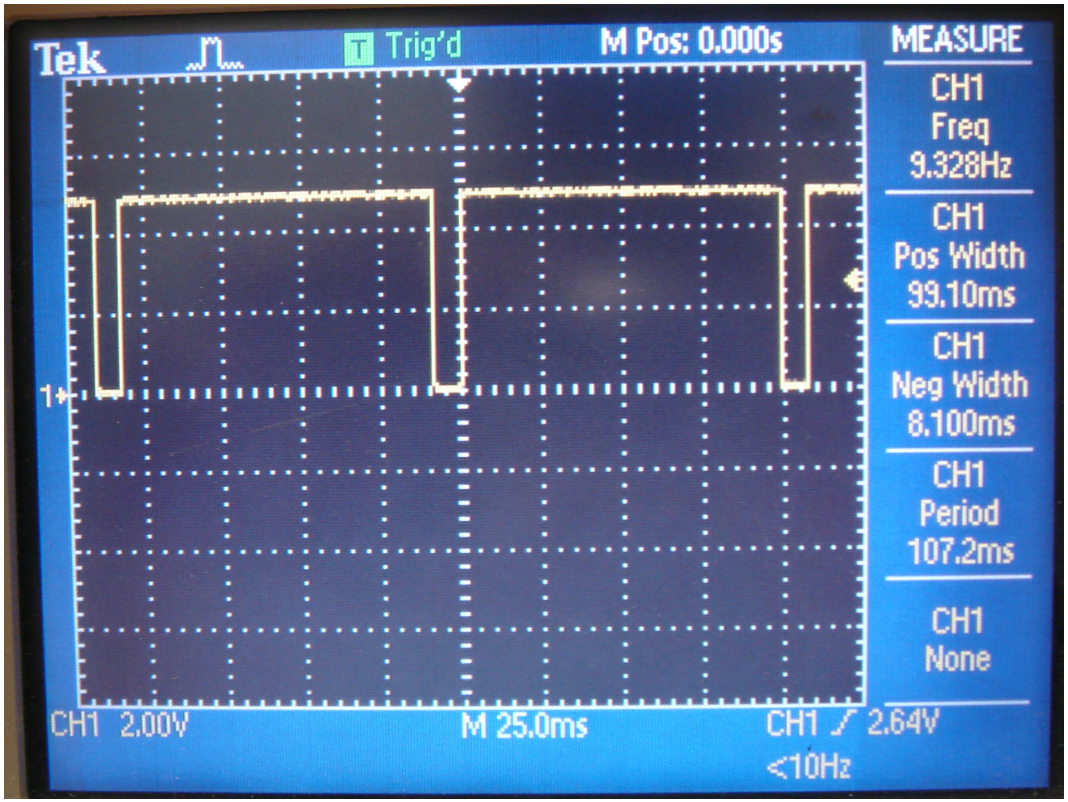
- Wait loop
- 100ms

RA1 = 0

- Rest of code
- 8.1ms

To make the loop take 100ms

- Rest of code = 8.1ms
- Wait_ms(92) waits 92ms
- Total = 100ms



The Power of C

Write the code for a roulette wheel:

- Press RB0 to place your bet (winning number is always 0)
- Display your bet on row #1
- Generate a random number N from 0..7
- Count 42 + N times (mod 8)
  - Display that number on the LCD dispay
  - Pause 100ms between counts
  - Beep at 220Hz each count
- If you stop on 0, you win!
  - Add $8 to your bank total
  - Otherwise you lose $1
- Start again when you press RB0

Note: Use the LCD display to check your code as you write it

Bottom-Up Programming

Step 1: Generate a random number from 0..7

- Wait for a button press
- When pressed, count really fast mod 8
- When released, the count is your random number
- Display the number on the LCD display

Winning Numbers:

- 2, 3, 0, 4, 5, 5, 5, 4, 7, 1, 2, 5, 7, 0, 3, 6, 6, 6, 5, 0, 0, 1, 4, 6
- Is this random???

```
unsigned int i, j;
unsigned int DIE;
unsigned int BANK;
unsigned int N;

TRISA = 0;
TRISB = 0xFF;
TRISC = 0;
TRISD = 0;
TRISE = 0;
ADCON1 = 0x0F;

BANK= 100;

LCD_Init();                    // initialize the LCD

LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MSG0[i]);
LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MSG1[i]);

while(1) {
   while(!RB0);
   while(RB0) N = (N + 1)%8;

   DIE = N;

   if(DIE == 0) BANK += 8;
   else BANK -= 1;

   LCD_Move(0,8);  LCD_Out(DIE, 1, 0);
   LCD_Move(1,8);  LCD_Out(BANK, 4, 0);

   }
```

Step 2: Rotate the marble 42+N times

- Play with the user
- Pause 100ms between
- The computer knows the winning number as soon as you release the button

Display the ball position on the LCD display

- You should see the number changing rapidly (every 100ms)
- Slow down the code if you want to see what's happening
- Wait_ms(100); >>>> Wait_ms(500);

\09_LCD\Roulette.c

```
    TRISD = 0;
    TRISE = 0;
    ADCON1 = 0x0F;

    BANK= 100;

    LCD_Init();                          // initializ

    LCD_Move(0,0);  for (i=0; i<20; i++) LCD_
    LCD_Move(1,0);  for (i=0; i<20; i++) LCD_

    while(1) {
       while(!RB0);
       while(RB0) N = (N + 1)%8;

       for(i=0; i<32+N; i++) {
          DIE = (DIE + 1)%8;
          LCD_Move(0,8);  LCD_Out(DIE, 1, 0);
          Wait_ms(100);
          }

       if(DIE == 0) BANK += 8;
       else BANK -= 1;

       LCD_Move(0,8);  LCD_Out(DIE, 1, 0);
       LCD_Move(1,8);  LCD_Out(BANK, 4, 0);

       }
}
```

Step 3: Beep with each number

- i = number of toggles
  - 50 toggles
  - 25 pulses
- j sets the period & frequency

```
// Subroutine Declarations
#include <pic18.h>

// Subroutines
#include        "lcd_portd.c"

void Beep(void) {

   unsigned int i, j;
   for(i=0; i<50; i++) {
      RA1 = !RA1;
      for(j=0; j<400; j++);
   }
}

// Main Routine

void main(void)
{
   unsigned int i, j;
   unsigned int DIE;
   unsigned int BANK;
   unsigned int N;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   TRISD = 0;
   TRISE = 0;
```

Step 4: Slow down as you get close to the end

- Count down from 32+N to 0
- Wait 50ms / number for N large
- Wait 1000ms at the end (N==1)

Use the LCD display to show the current ball position

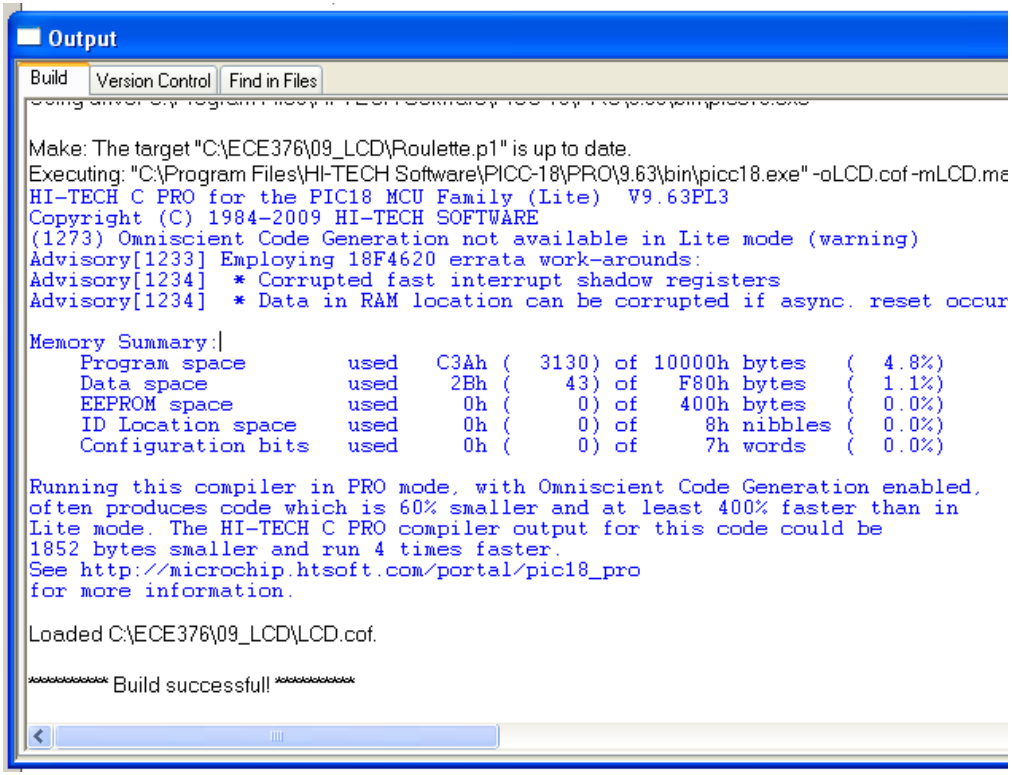
```
while(1) {
   while(!RB0);
   while(RB0) N = (N + 1)%8;

   for(i=32+N; i>0; i--) {
      DIE = (DIE + 1)%8;
      LCD_Move(0,8);  LCD_Out(DIE, 1, 0);
      Beep();
      Wait_ms(50 + 1000/i);
      }

   if(DIE == 0) BANK += 8;
   else BANK -= 1;

   LCD_Move(0,8);  LCD_Out(DIE, 1, 0);
   LCD_Move(1,8);  LCD_Out(BANK, 4, 0);

   }
```

Resulting Code:

- 3130 Bytes (1565 lines of assembler)

Improvements:

- Display the marble or PORTC
  - Each LED is a winning number, 0..7
- Allow bets on other numbers
  - Use buttons RB0..RB7
- Rig the game
  - Fair game 90% of the time
  - Always lose by one 10% of the time



Summary

LCD displays are really useful:

- You can proved a lot more information than just LEDs

Bottom-Up Programming helps with writing complicated routines

- Build the code step-by-step
- Check your code as you write it
- LCD display is a good way of seeing what your code is doing

You can write fairly complicated routines in C fairly quickly

- Roulette wheel code was written in under 30 minutes
- Result was 1565 lines of assembler
- (Assembler would take a *lot* longer to write).