# LED Cube 16×16×16 on PIC18F46K22

**// File: LED\_Cube.c**

**// Project: 16×16×16 LED Cube on PIC18F46K22**

**// Student: Mahmoud Ali Hassan**

**// Supervisor: Dr. Muhammad Sbih**

**// Tishreen University, 2013**

**#include <xc.h>**

**#include <stdint.h>**

**#include <string.h>**

**// Configuration bits**

**#pragma config FOSC = HS // High-speed crystal/resonator**

**#pragma config WDT = OFF // Watchdog Timer off**

**#pragma config LVP = OFF // Low-voltage programming off**

**#pragma config MCLRE = ON // MCLR pin enabled**

**#pragma config PBADEN = OFF // PORTB<4:0> as digital**

**#pragma config XINST = OFF // Extended instruction set off**

**#define \_XTAL\_FREQ 20000000UL // 20 MHz crystal**

**// Cube geometry**

**#define NUM\_LAYERS 16**

**#define NUM\_COLUMNS (NUM\_LAYERS\*NUM\_LAYERS) // 256 columns total**

**#define SR\_BYTES (NUM\_COLUMNS/8) // 32 shift-register bytes**

**// Shift-register latch pin (74HC595 RCLK)**

**#define SR\_LATCH\_LAT LATBbits.LATB2**

**#define SR\_LATCH\_TRIS TRISBbits.TRISB2**

**// Layer drivers: 16 MOSFET gates on PORTC (0–7) and PORTA (0–7)**

**#define LAYERS\_PORT1 LATC**

**#define LAYERS\_TRIS1 TRISC**

**#define LAYERS\_PORT2 LATA**

**#define LAYERS\_TRIS2 TRISA**

**// Layer dwell time in microseconds (0.5 ms)**

**#define LAYER\_DELAY\_US 500**

**// Number of patterns**

**#define NUM\_PATTERNS 2**

**// Forward declarations**

**void init\_ports(void);**

**void init\_spi(void);**

**void disable\_all\_layers(void);**

**void enable\_layer(uint8\_t layer);**

**void shift\_out\_layer(uint8\_t \*buf);**

**void pattern\_rain(uint8\_t layer, uint8\_t \*buf);**

**void pattern\_demo(uint8\_t layer, uint8\_t \*buf);**

**// LFSR for pseudo-random**

**static uint16\_t lfsr = 0xACE1;**

**static uint8\_t rain\_state[NUM\_COLUMNS];**

**// Pattern function table**

**void (\*patterns[NUM\_PATTERNS])(uint8\_t, uint8\_t\*) = {**

**pattern\_rain,**

**pattern\_demo**

**};**

**int main(void) {**

**uint8\_t layer, current\_pattern = 0;**

**uint8\_t sr\_buffer[SR\_BYTES];**

**init\_ports();**

**init\_spi();**

**disable\_all\_layers();**

**memset(rain\_state, 0, sizeof(rain\_state));**

**while (1) {**

**// Cycle through layers**

**for (layer = 0; layer < NUM\_LAYERS; layer++) {**

**// Generate pattern data for this layer**

**patterns[current\_pattern](layer, sr\_buffer);**

**// Output to shift registers**

**shift\_out\_layer(sr\_buffer);**

**// Drive only this layer**

**disable\_all\_layers();**

**enable\_layer(layer);**

**// Short dwell**

**\_\_delay\_us(LAYER\_DELAY\_US);**

**}**

**// Switch pattern every full cube update**

**current\_pattern++;**

**if (current\_pattern >= NUM\_PATTERNS) {**

**current\_pattern = 0;**

**}**

**}**

**return 0;**

**}**

**// Initialize port directions**

**void init\_ports(void) {**

**// Shift-register latch pin**

**SR\_LATCH\_TRIS = 0;**

**SR\_LATCH\_LAT = 0;**

**// Layer driver pins**

**LAYERS\_TRIS1 = 0x00;**

**LAYERS\_TRIS2 = 0x00;**

**LAYERS\_PORT1 = 0x00;**

**LAYERS\_PORT2 = 0x00;**

**// SPI pins: SDO1 (RC7), SCK1 (RC6), SDI1 unused**

**TRISCbits.TRISC7 = 0; // SDO1 output**

**TRISCbits.TRISC6 = 0; // SCK1 output**

**}**

**// Initialize MSSP1 as SPI Master**

**void init\_spi(void) {**

**SSP1CON1 = 0b00100001; // SSPEN=1, CKP=0, SSPM=0001 (Fosc/16)**

**SSP1STAT = 0b00000000; // CKE=0**

**}**

**// Turn off all layer MOSFETs**

**void disable\_all\_layers(void) {**

**LAYERS\_PORT1 = 0x00;**

**LAYERS\_PORT2 = 0x00;**

**}**

**// Enable a single layer (0–15)**

**void enable\_layer(uint8\_t layer) {**

**if (layer < 8) {**

**LAYERS\_PORT1 = (1 << layer);**

**} else {**

**LAYERS\_PORT2 = (1 << (layer - 8));**

**}**

**}**

**// Shift out SR\_BYTES bytes MSB-first and latch**

**void shift\_out\_layer(uint8\_t \*buf) {**

**uint8\_t i;**

**SR\_LATCH\_LAT = 0;**

**for (i = 0; i < SR\_BYTES; i++) {**

**// send highest‐order byte first**

**SSP1BUF = buf[SR\_BYTES - 1 - i];**

**while (!SSP1STATbits.BF); // wait until transmitted**

**}**

**SR\_LATCH\_LAT = 1;**

**}**

**// Generate pseudo-random bit from 16-bit LFSR**

**static uint8\_t get\_random\_bit(void) {**

**uint16\_t bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5)) & 1;**

**lfsr = (lfsr >> 1) | (bit << 15);**

**return (uint8\_t)(lfsr & 0xFF);**

**}**

**// Pattern 1: Rain simulation**

**void pattern\_rain(uint8\_t layer, uint8\_t \*buf) {**

**uint16\_t col;**

**// Clear buffer**

**memset(buf, 0x00, SR\_BYTES);**

**// Update drop positions only at layer 0**

**if (layer == 0) {**

**for (col = 0; col < NUM\_COLUMNS; col++) {**

**if (rain\_state[col] > 0) {**

**rain\_state[col]--;**

**} else if ((get\_random\_bit() & 0x3F) == 0) {**

**rain\_state[col] = NUM\_LAYERS;**

**}**

**}**

**}**

**// Light LED where drop height equals this layer index + 1**

**for (col = 0; col < NUM\_COLUMNS; col++) {**

**if (rain\_state[col] == (NUM\_LAYERS - layer)) {**

**uint16\_t byte\_index = col >> 3;**

**uint8\_t bit\_pos = col & 0x07;**

**buf[byte\_index] |= (1 << bit\_pos);**

**}**

**}**

**}**

**// Pattern 2: Simple demo (alternating checkerboard slices)**

**void pattern\_demo(uint8\_t layer, uint8\_t \*buf) {**

**uint16\_t row, col;**

**// For even layers: horizontal stripes, odd layers: vertical stripes**

**if ((layer & 1) == 0) {**

**// Horizontal: each row on/off in 16-bit pattern**

**for (row = 0; row < NUM\_LAYERS; row++) {**

**uint16\_t pattern = (row & 1) ? 0xFFFF : 0x0000;**

**for (col = 0; col < NUM\_LAYERS; col++) {**

**if ((pattern >> (col & 0x0F)) & 1) {**

**uint16\_t idx = row \* NUM\_LAYERS + col;**

**buf[idx >> 3] |= (1 << (idx & 0x07));**

**}**

**}**

**}**

**} else {**

**// Vertical: each column on/off**

**for (col = 0; col < NUM\_LAYERS; col++) {**

**uint16\_t pattern = (col & 1) ? 0xFFFF : 0x0000;**

**for (row = 0; row < NUM\_LAYERS; row++) {**

**if ((pattern >> (row & 0x0F)) & 1) {**

**uint16\_t idx = row \* NUM\_LAYERS + col;**

**buf[idx >> 3] |= (1 << (idx & 0x07));**

**}**

**}**

**}**

**}**

**}**