To enhance your existing code so that it reads both **date** and **time** from the DS1302 RTC and displays them on a 16x2 LCD with descriptive labels, follow the steps below. This modification will ensure that the LCD shows "Time: hh:mm

" on the first line and "Date: dd/mm/yy" on the second line, updating every second.

**Key Enhancements:**

1. **Reading Additional Registers:** We'll read the date, month, and year from the DS1302's registers in addition to the existing time (hours, minutes, seconds).
2. **BCD to Binary Conversion:** The DS1302 stores data in Binary-Coded Decimal (BCD) format. We'll implement a function to convert BCD values to binary for correct display.
3. **Formatted Display:** We'll create separate functions to format and display the time and date with appropriate labels on the LCD.

**Complete Modified Code:**

c

Copy code

#include <reg52.h>

#include <intrins.h>

// Pin Definitions

sbit SCLK = P3^6; // Serial Clock

sbit IO = P3^4; // Data I/O

sbit CE = P3^5; // Chip Enable

sbit RS = P2^6; // LCD Register Select

sbit RW = P2^5; // LCD Read/Write

sbit EN = P2^7; // LCD Enable

// Function Prototypes

unsigned char BCD\_to\_BIN(unsigned char val);

void delay\_ms(unsigned int ms);

void LCD\_Command(unsigned char command);

void LCD\_Init(void);

void LCD\_WriteChar(char c);

void LCD\_WriteString(char \*str);

void DS1302\_Init(void);

void DS1302\_WriteByte(unsigned char dat);

unsigned char DS1302\_ReadByte(void);

void DS1302\_WriteTime(unsigned char address, unsigned char dataa);

unsigned char DS1302\_ReadTime(unsigned char address);

void Display\_Time(unsigned char hour, unsigned char min, unsigned char sec);

void Display\_Date(unsigned char date, unsigned char month, unsigned char year);

// Function to convert BCD to Binary

unsigned char BCD\_to\_BIN(unsigned char val) {

return ((val >> 4) \* 10) + (val & 0x0F);

}

// Simple Delay Function (Approximate 1 ms per call)

void delay\_ms(unsigned int ms) {

unsigned int i, j;

for(i = 0; i < ms; i++)

for(j = 0; j < 120; j++); // Adjust inner loop for accurate delay if necessary

}

// LCD Command Function

void LCD\_Command(unsigned char command) {

RS = 0;

RW = 0;

P0 = command;

EN = 1;

delay\_ms(1);

EN = 0;

}

// Initialize LCD

void LCD\_Init(void) {

LCD\_Command(0x38); // 8-bit mode, 2 lines, 5x7 dots

LCD\_Command(0x0C); // Display ON, Cursor OFF

LCD\_Command(0x06); // Entry mode: Increment cursor

LCD\_Command(0x01); // Clear display

delay\_ms(2);

}

// Write Character to LCD

void LCD\_WriteChar(char c) {

RS = 1;

RW = 0;

P0 = c;

EN = 1;

delay\_ms(1);

EN = 0;

}

// Write String to LCD

void LCD\_WriteString(char \*str) {

while(\*str) {

LCD\_WriteChar(\*str++);

}

}

// Initialize DS1302

void DS1302\_Init(void) {

CE = 0; // Disable DS1302

SCLK = 0; // Initialize clock line

delay\_ms(1);

// Optionally, disable write protection by writing to control register (0x8E)

DS1302\_WriteTime(0x8E, 0x00); // Disable write protection

}

// Write Byte to DS1302

void DS1302\_WriteByte(unsigned char dat) {

unsigned char i;

for (i = 0; i < 8; i++) {

IO = dat & 0x01;

SCLK = 1;

delay\_ms(1);

SCLK = 0;

dat >>= 1;

}

}

// Read Byte from DS1302

unsigned char DS1302\_ReadByte(void) {

unsigned char dat = 0;

unsigned char i;

IO = 1; // Configure IO as input (open-drain high)

for (i = 0; i < 8; i++) {

dat >>= 1;

if (IO) dat |= 0x80;

SCLK = 1;

delay\_ms(1);

SCLK = 0;

}

return dat;

}

// Write Time/Date to DS1302

void DS1302\_WriteTime(unsigned char address, unsigned char dataa) {

CE = 1; // Enable DS1302

delay\_ms(1);

DS1302\_WriteByte(address); // Send address

DS1302\_WriteByte(dataa); // Send data

CE = 0; // Disable DS1302

delay\_ms(1);

}

// Read Time/Date from DS1302

unsigned char DS1302\_ReadTime(unsigned char address) {

unsigned char dataa;

CE = 1; // Enable DS1302

delay\_ms(1);

DS1302\_WriteByte(address | 0x01); // Send address with read command

dataa = DS1302\_ReadByte(); // Read data

CE = 0; // Disable DS1302

delay\_ms(1);

return dataa;

}

// Display Time on LCD

void Display\_Time(unsigned char hour, unsigned char min, unsigned char sec) {

char timeStr[17];

unsigned char index = 0;

// Format: "Time: hh:mm:ss"

timeStr[index++] = 'T';

timeStr[index++] = 'i';

timeStr[index++] = 'm';

timeStr[index++] = 'e';

timeStr[index++] = ':';

timeStr[index++] = ' ';

timeStr[index++] = (hour / 10) + '0';

timeStr[index++] = (hour % 10) + '0';

timeStr[index++] = ':';

timeStr[index++] = (min / 10) + '0';

timeStr[index++] = (min % 10) + '0';

timeStr[index++] = ':';

timeStr[index++] = (sec / 10) + '0';

timeStr[index++] = (sec % 10) + '0';

timeStr[index++] = ' ';

timeStr[index++] = ' ';

timeStr[index] = '\0';

LCD\_Command(0x80); // Move cursor to the first line

LCD\_WriteString(timeStr);

}

// Display Date on LCD

void Display\_Date(unsigned char date, unsigned char month, unsigned char year) {

char dateStr[17];

unsigned char index = 0;

// Format: "Date: dd/mm/yy"

dateStr[index++] = 'D';

dateStr[index++] = 'a';

dateStr[index++] = 't';

dateStr[index++] = 'e';

dateStr[index++] = ':';

dateStr[index++] = ' ';

dateStr[index++] = (date / 10) + '0';

dateStr[index++] = (date % 10) + '0';

dateStr[index++] = '/';

dateStr[index++] = (month / 10) + '0';

dateStr[index++] = (month % 10) + '0';

dateStr[index++] = '/';

dateStr[index++] = (year / 10) + '0';

dateStr[index++] = (year % 10) + '0';

dateStr[index++] = ' ';

dateStr[index++] = ' ';

dateStr[index] = '\0';

LCD\_Command(0xC0); // Move cursor to the second line

LCD\_WriteString(dateStr);

}

void main(void) {

unsigned char sec\_bcd, min\_bcd, hour\_bcd;

unsigned char date\_bcd, month\_bcd, year\_bcd;

unsigned char sec, min, hour;

unsigned char date, month, year;

DS1302\_Init();

LCD\_Init();

while(1) {

// Read BCD values from DS1302

sec\_bcd = DS1302\_ReadTime(0x81); // Seconds

min\_bcd = DS1302\_ReadTime(0x83); // Minutes

hour\_bcd = DS1302\_ReadTime(0x85); // Hours

date\_bcd = DS1302\_ReadTime(0x87); // Date (Day of Month)

month\_bcd = DS1302\_ReadTime(0x89); // Month

year\_bcd = DS1302\_ReadTime(0x8D); // Year

// Convert BCD to Binary

sec = BCD\_to\_BIN(sec\_bcd);

min = BCD\_to\_BIN(min\_bcd);

hour = BCD\_to\_BIN(hour\_bcd);

date = BCD\_to\_BIN(date\_bcd);

month = BCD\_to\_BIN(month\_bcd);

year = BCD\_to\_BIN(year\_bcd);

// Display Time and Date

Display\_Time(hour, min, sec);

Display\_Date(date, month, year);

delay\_ms(1000); // Wait for 1 second

}

}

**Explanation of Modifications:**

1. **BCD to Binary Conversion:**
   * **Function:** BCD\_to\_BIN converts BCD values to their binary equivalents, essential for correct numerical representation on the LCD.
   * **Usage:** After reading each time and date component from the DS1302, we convert them using this function before displaying.
2. **Reading Date Components:**
   * **Registers:**
     + **Date (Day of Month):** Address 0x87
     + **Month:** Address 0x89
     + **Year:** Address 0x8D
   * **Reading Process:** Similar to how hours, minutes, and seconds are read, we read these additional registers and convert their BCD values.
3. **Formatted Display Functions:**
   * **Display\_Time:** Formats and displays the current time with the prefix "Time: " on the first line of the LCD.
   * **Display\_Date:** Formats and displays the current date with the prefix "Date: " on the second line of the LCD.
   * **String Formatting:** Both functions construct strings character by character, appending appropriate labels and formatting separators (: and /).
4. **LCD Initialization and Communication:**
   * **Initialization:** Ensures the LCD is set to 8-bit mode, 2 lines, and initializes display settings.
   * **Commands and Data:** The LCD\_Command and LCD\_WriteChar functions handle sending commands and data to the LCD.
5. **DS1302 Initialization:**
   * **Write Protection:** The DS1302 has a write protection feature. To allow writing to its registers, we disable this protection by writing 0x00 to the control register (0x8E).
6. **Delay Function:**
   * **Accuracy:** The delay\_ms function provides an approximate delay. For more accurate timing, especially in applications requiring precise intervals, consider implementing a timer-based delay or using interrupt-driven timing.

**Additional Considerations:**

* **Hardware Connections:**
  + **Ensure** that all connections between the DS1302, 8051 microcontroller, and LCD are correctly established as per the pin definitions in the code.
* **IO Pin Direction:**
  + The IO pin on the DS1302 is bidirectional. In the code, setting IO = 1 effectively configures it as an input due to the open-drain nature of the DS1302's IO pin.
* **Power Supply and Backup Battery:**
  + The DS1302 requires a backup battery (typically a CR2032 coin cell) connected to its VBAT pin to maintain timekeeping when the main power is off.
* **Testing and Debugging:**
  + **Initial Testing:** It's advisable to test the time and date reading functions separately before integrating them with the LCD display to ensure accurate communication with the DS1302.
  + **LCD Display:** Verify that the LCD is correctly initialized and that characters are being displayed as expected.

By implementing these modifications, your system will successfully read both the date and time from the DS1302 RTC module and display them on the 16x2 LCD with clear labels, updating every second.