Multiplication of Backward Differences (MOBD) Algorithm

```
. . . . . .

int dummy, d0, d1, d2, mobd, threshold, rri_count, hr;
. . . . . .

void interrupt isr() (
. . . . . .
        case 8:                     // Function 8: Heart rate meter
            TMR0H = 0xED;           // Reload TMR0 for 5 ms count, sampling rate = 200 Hz
            TMR0L = 0x44;           // 0xFFFF-0xED44 = 0x12BB = 4795, (205 us delay)
            data1 = data0;          // Move old ECG sample to data1
            data0 =  ReadADC();     // Store new ECG sample from ADC to data0
            PORTBbits.RB2 = !PORTBbits.RB2; // Toggle RB2 for sampling rate check
            do_MOBD = 1;            // Flag main() to do MOBD for QRS detection
            break;
. . . . . .

void main() {
    threshold = 50;             // Threshold for the MOBD QRS-detection algorithm
    . . . . . .

    while (1) {
        . . . . . .
        case 8:             // Function 9: Multiplication of Backward Differences (MOBD)
            if (do_MOBD){                   // MOBD = Multiplication of Backwards Differences
                INTCONbits.TMR0IE = 0;      // Disable TMR0 interrupt
                do_MOBD = 0;                // Reset new_data flag
                d2 = d1;                    // Move oldest difference to d2
                d1 = d0;                    // Move older difference to d1
                d0 = (int)data0 - data1;    // Store new difference in d0, (int) casting important
                rri_count++;                // Increment RR-interval
                mobd = 0;                   // mobd = 0, unless sign consistency is met:
                if (d0 > 0 && d1 > 0 && d2 > 0){    // (1) If 3 consecutive positive differences
                    mobd = d0 * d1;         // Multiply first two differences
                    mobd = mobd >> 3;       // Scale down (divide by 8)
                    mobd = mobd * d2;       // Multiply the oldest difference
                }
                if (d0 < 0 && d1 < 0 && d2 < 0){    // (2) If 3 consecutive negative differences
                    d0 = -d0;               // Take absolute value of differences
                    d1 = -d1;
                    d2 = -d2;
                    mobd = d0 * d1;         // Multiply first two differences
                    mobd = mobd >> 3;       // Scale down (divide by 8)
                    mobd = mobd * d2;       // Multiply the oldest difference
                }
                if (refractory){            // Avoid detecting extraneous peaks after QRS
                    refractory++;
                    if (refractory == 40){  // Delay for 200 ms
                        refractory = 0;     // Reset refractory flag to 0
                        PORTBbits.RB3 = 0;  // Turn buzzer/LED off (Pin 36)
                    }
                }
                else if (mobd > threshold){ // If a peak is detected,
                    refractory = 1;         // Set refractory flag
                    PORTBbits.RB3 = 1;      // Turn buzzer/LED on (Pin 36)
                    display = 1;            // Set display flag
                }
                PORTD =(unsigned char)mobd; // Output mobd value to Port D
                INTCONbits.TMR0IE = 1;      // Enable TMR0 interrupt
            }
            if (display){                   // Display Heart Rate in 3 digits
                hr = 12000/rri_count;       // 60/0.005 = 12000
                rri_count = 0;              // Reset RRI counter
                PrintNum(hr, 71);           // Isolates each digit and displays
                display = 0;                // Reset display flag
            }
            break;
```