SUMMARY
This application note describes two experiments. One experiment allows the user to check that the low battery bit is working correctly. The other allows the user to generate a 10 ms periodic interrupt.

A program named RTC, written in Microsoft® Quick C version 1, is used for both experiments. The code works with the circuit shown in Figure 3. This circuit is a general purpose interface for use with an IBM® PC-XT® or PC-AT® (or equivalent). Keyboard entries may be either upper or lower case, but the underscores must be included.

Type rtc to execute the program, then follow the instructions on the monitor.

LOW BATTERY BIT EXPERIMENT
Equipment: Variable lab supply or 20k pot, center tapped to the VBB pin.
An oscilloscope with 10 MΩ, 10 pF probe or higher impedance.
The initial screen output is shown in Message 1. Before selecting the LOW_BATT_BIT mode, set the voltage at the VBB pin to about 2.5V. If VBB is GND or too low a voltage, a message should appear (see Message 2). Once the LOW_BATT_BIT mode is running, you should see screen output (see Message 3). The status of the low battery bit is displayed in the lower left of the monitor. A value of 0 indicates VBB is higher than the internal threshold detector. A value of 40 indicates the low battery bit is set.

Monitor the waveform at the OSC OUT pin. Observing the peak-to-peak voltage of this waveform is the only way to know that the DP8570A is in the battery backed mode, unless the test mode is selected. The waveform is sinusoidal in form and swings within 0.6V of VBB and ground. Refer to Figures 1a, 1b, 1c.

Vary the VBB voltage slowly as you approach 2.3V. If the VBB voltage gets too low (less than 1.8V) the oscillator may stop. During the low battery bit mode, if VBB is grounded or a too low a voltage, you will not get any indication on screen. If you hit the spacebar and re-enter the LOW_BATT_BIT mode, Message 1 warning will appear on the monitor.

10 mS INTERRUPT EXPERIMENT
Equipment: An oscilloscope is needed to monitor the INTR pin.

Before starting this section of the program, connect VBB to ground. The INTR_10 ms code configures the DP8570A in the Single Supply mode. Message 4 is output to the monitor indicating you are in the 10 ms Interrupt mode. Figure 2a shows expected waveforms for a PC-XT (4.77 MHz); Figure 2b a 386/33 MHz AT.
Watch out! VBB is at Ground or some illegal value

VBB voltage should be between 2.2V and VCC = 0.4V

Message 2: VBB Warning

Battery backed mode selected.
Check waveform at osc out to see if referenced
to the battery voltage
Peak value should be less than the battery voltage

Adjust voltage on VBB pin while monitoring screen.
The bottom left side of the screen will display zero
if VBB > threshold (about 2.1 volts),
or 40 if VBB < threshold.
This test may be ended by hitting the space bar.

Message 3: Normal Message after Selecting LOW_BATT_BIT
FIGURE 1a

Conditions:
LOW_BATT_BIT chosen
$V_{BB} = 3V$
$V_{CC} = 5V$

FIGURE 1b

Conditions:
LOW_BATT_BIT chosen
$V_{BB} = 2.5V$
$V_{CC} = 5V$

FIGURE 1c

Conditions:
INTR_10 ms chosen
$V_{BB} = 3V$
$V_{CC} = 5V$
(default to single supply mode)

FIGURE 1d

Conditions:
LOW_BATT_BIT chosen
$V_{BB} = 2.5V$
$V_{CC} = 5V$
The Oscillator is running. The Clock is started.
Now you are in the 10ms Interrupt mode
Use an oscilloscope to view the waveform at the INTR pin
Hit spacebar to return to 'Selection Menu'

Message 4: Normal Message after Selecting INTR_10ms

FIGURE 2a: Expected Waveforms at INTR Pin

FIGURE 2b: Expected Waveforms at INTR Pin
Component and Placement List

- **C1**: 0.047 μF ceramic placed at U1
- **C2**: 0.047 μF ceramic placed at U2
- **C3**: 0.047 μF ceramic placed at U3
- **C4**: 0.047 μF ceramic placed at U4
- **C5**: 5.6 μF solid tantalum placed at card edge
- **D1, D2, D3**: 1N914 or 1N4001 or 1N4933
- **R1, R5**: 2.7 kΩ
- **R2**: 3.9 kΩ
- **R3**: 10 kΩ
- **R4**: 10 kΩ
- **R6**: 39 kΩ
- **R7**: 20 kΩ
- **R8**: 0.1 μF

**Table I**

<table>
<thead>
<tr>
<th>CR1</th>
<th>C IN</th>
<th>COUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.768/32.0 kHz</td>
<td>15</td>
<td>47</td>
</tr>
<tr>
<td>4.19404 MHz</td>
<td>36</td>
<td>68</td>
</tr>
<tr>
<td>4.9152 MHz</td>
<td>36</td>
<td>68</td>
</tr>
</tbody>
</table>

**FIGURE 3. General Purpose Interface to IBM PC-XT or PC-AT for Experiments**

*Note: The circuit diagram includes various components such as resistors, capacitors, diodes, and a lithium battery.*
/******************************************************************************
 * This program RTC.C is designed to work with the DP857x family of
 * Real Time Clocks. It works with a demo board interfaced to a PC/XT
 * or PC/AT. This 'C' program is written in Microsoft C (version 6.0).
 * This program has two parts:
 * Part 1 allows testing the Low Battery Bit, in the Batt_Back Mode.
 * Part 2 initializes the 10 millisecond periodic interrupt.
 * Also, it delays clearing the INT, after polling the INT flag in
 * the 'MSR' for the purpose of observing the output on an Oscilloscope.
 ***************************************************************************/

#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <graph.h>

#define MSR 0x300 /* main status register */
#define PFR 0x303 /* periodic flag register */
#define RTMR 0x301 /* real time mode register */
#define IRR 0x304 /* interrupt routing register */
#define CMR 0x302 /* output mode register */
#define ICRO 0x303 /* interrupt control register 0 */
#define ICR1 0x304 /* interrupt control register 1 */
#define TCR0 0x301 /* timing control register 0 */
#define TCR1 0x302 /* timing control register 1 */
#define TESTR 0x31F /* Test Mode register */

enum { LOW_BATT_BIT, INTR_10ms, END } mode;
char buf[80];
char *mode_str[] = { "LOW_BATT_BIT", "INTR_10ms", "END" };

main()
{
    char *input;
    int i;

    /* Initialize the RTC, select 32.768KHz. */
    /* The following while loop tries to start the clock */
    /* and tests the osc fail bit to see that the oscillator */
    /* is running. The oscillator must be running in order */
    /* to configure the DP857X for battery back mode */
    _clearscreen( GCLEARSCREEN);
    printf("\n\nCheck that the oscillator has started.\n");
    printf("\nIf you don't get osc running in 5 seconds, the program will abort\n")
    ;
    init();

    printf("\n\nYou may choose the 'Low Batry Bit' Test\n");
    printf("\nthe 10ms Interrupt Test\n");
    printf("\n'END to return to DOS'\n");
    outpt(MSR, 0);
    mode = -1;
    do {
        while (mode != LOW_BATT_BIT && mode != INTR_10ms && mode != END)
        {
            printf("\n\nType in your choice in the following format, then hit ENTER:");
            
TL/F/11847-8
printf("\n\nChoices are:\tLOW_BATT_BIT	 or \n\tINTR_10ms or \n\n");
printf("\t\tEND to exit the Program\n");
printf("\n\nEnter your choice now: \n\n");

    input = gets(buf);
    for (i=0; i<3; i++)
        if (!strcmp(buf, mode_str[i]))
            mode = i;
    }
}

switch(mode)
{
    case LOW_BATT_BIT:
        init();    /* Call 'init' function */
        batbak();  /* Call 'batbak' function */
        mode = -1;
        break;
    case INTR_10ms:
        init();    /* Call init routine */
        intr();    /* Call intr routine */
        mode = -1;
        break;
    case END:
        printf("\nThis is the END of the Program\n");
        break;
    }
} while (mode != END);

batbak()

/* This program configures the DP857X 32.768KHz oscillator
   Conditions: Vcc = 5V, VBB = 3.0V (adjustable),
   T = ambient temperature */
{
    outp(MSR,0x40);    /* select bank 1 */
    outp(ICR1,0x80);   /* set PFAIL enable in ICR1 */
    outp(MSR,0);       /* select battery backed mode */
    outp(PFR,0);

    if(inp(IRR) & 0x40)
    {
        _clearscreen(_GCLEARSCREEN);
        _settextposition(11,15);
        printf("Watch out! VBB is at Ground or some illegal value");
        _settextposition(13,15);
        printf("VBB voltage should be between 2.2V and VCC - 0.4V");
        _settextposition(24,0);
        mode = -1;
        exit();
    }
    _clearscreen(_GCLEARSCREEN);
    printf("\n\tBattery backed mode selected.\n");
    printf("\n\tCheck waveform at OSC out to see if referenced\n");
    printf("\n\tto the battery voltage.\n\tPeak value should be less.");
    printf("\n\tthan the battery voltage\n");
printf("\n\n\n");
printf("\n\tAdjust voltage on VBB pin while monitoring screen."\n);  
printf("\n\tThe bottom left side of the screen will display zero\n");
printf("\n\tIf VBB > threshold (about 2.1volts),\n");
printf("\n\tor 40 if VBB < threshold.\n");
printf("\n\n\n");
outp(MSR,0);  /* select bank 0 */

while(!kbhit())
{
    _settextposition(24,0);
    printf("%x",inp(IRR) & 0x40);  /* display low batt bit */
    getch();
}

intr()
{
    int i;
    i = 0;

    outp(MSR,0x3E);  /* clear all pending interrupts */
    outp(PPR,0x40);
    outp(TCR0,0);
    outp(TCR1,0);
    outp(IRR,0x1D);  /* select per. intr to intr pin */
    outp(MSR,0x40);  /* select register bank 1 */
    outp(GMR,0x8);   /* intr = push pull active lo */
    outp(ICR0,0x10); /* select 10ms periodic intr */
    outp(TCR1,0x80);

    printf("\n\nNow you are in the 10ms Interrupt mode\n");
    printf("\nYou can use Oscilloscope to view the waveform\n");
    printf("\nHit spacebar to return to 'Selection Menu'\n");

    do {
        for (i=0; i < 1300) & & ((inp(MSR) & 0x05) != 5); i++
    ;

        if (i == 1300)
        {
            printf("\nThere is something WRONG !!!\n");
            printf("\nPlease check the Voltage at the VBB pin\n");
            exit();
        }
        else
        {
            for (i=0; i < 300; i++) /* this loop is for */
            /* viewing the waveform */
            /* The value in the 'FOR' loop is dependent on the speed of */
            /* the Processor. The value '200' in this example is for */
            /* the PC/XT running at 4.7 MHz. */
            outp(MSR,0x3E);  /* clear per intr */
        }
    } while (!kbhit());
    getch();
}

TLF/11647-10
init () /* function initialization */
{
/* This function selects 32 KHz Oscillator and attempts to start the *
  * clock. Check for 'OSC Running'. If not running, output message *
  * 'Hardware Problem, not running'. Return to DOS. */

  unsigned long int dt;
  int pfr=0x40, rtrr=0, irr;
  time_t t1, t2;
  dt = 0;
  /* delta time. difference between */
  /* the start & stop time. */
  /* start time. */

  outp(MSR, 0);
  /* select page 0, register bank 0 */
  outp(PFR, 0x0C0);
  /* select test mode */
  outp(TESTR, 0);
  /* clear test register */
  outp(PFR, 0x40);
  /* deselect test mode */
  outp(MSR, 0x40);
  outp(IRC1, 0x80);
  outp(RTMR, 8);
  /* issue start clock command */
  time(&t1);

  while((!(pfr == 0x40) || (rtrr == 0)) && dt < 5 )
  {
  /* if I stay in while loop */

  outp(MSR, 0x40);
  /* select bank 1 */
  outp(RTMR, 0x08);
  /* select 32KHz, start clock */
  rtrr = inp(RTMR) & 8;
  /* get start/stop bit */
  outp(MSR, 0);
  /* select bank 0 */
  pfr = (inp(PFR)&0x40);
  /* get osc fail bit */
  irr = inp(IRR);
  time(&t2);
  dt = t2 - t1;
  } if (dt == 5)
  {
  printf("\nThere is something wrong with the Hardware !");
  exit(0);
  }
  else
  printf("\n\t The Oscillator is running. The Clock is started.");
}

TL/F/11847-11
LIFE SUPPORT POLICY

NATIONAL’S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.