PUMPKIN™
REAL-TIME SOFTWARE

AN-7
Application Note

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

# Ninety-Day Countdown Timer

## Introduction

Some embedded applications have actions that occur on month-long or greater timescales. This Application Notes illustrates an easy way to create and use a three-month delay in an application using the Salvo RTOS.

## Application

Modern residential HVAC[1] systems are commonly driven by low-cost programmable electronic thermostats. The homeowner can set the desired temperature based on time of day, day of week, interior zone, etc. through a simple user interface consisting of a few buttons and a custom LCD display.

Home furnaces and air conditioners are outfitted with filters designed to trap nuisance particles such as pollen, plant spores, lint, pet hair and household dust. These filters have a finite life and should be replaced, on average, every ninety days.

A thermostat with a feature to alert the homeowner as to when the filter needs replacing will display "Replace Filter" after ninety days of operation.[2] A button is provided so that the user can reset this indicator after having replaced the filter.

## Salvo Code

The Salvo code to implement a ninety-day timer is shown in Listing 1 below. The system timer is configured for 10ms ticks.[3] The code specific to the countdown timer is shown in **bold**.

```
main()
{
  …
  OSInit();
  …
```

*created by Andrew E. Kalman on May 2, 2001 updated on May 3, 2001*

```
  OSCreateTask(TaskFilterLife,
    TASK_FILTER_LIFE_P, 9);
  …
  filterNeedsChanging = FALSE;
  …
  for (;;) {
    OSSched();
  }
}


void TaskFilterLife ( void )
{
  static unsigned long int longCounter;


  for (;;) {
    longCounter = 3888000;
    do {
      OS_Delay(200, TaskFilterLife1);
    } while ( --longCounter );

    filterNeedsChanging = TRUE;

    OSStop(TaskFilterLife2);
  }
}


void TaskKeypad ( void )
{
  for (;;) {
    …
    switch ( key ) {
      …
      case RESET_FILTER:
        filterNeedsChanging = FALSE;
        OSStartTask(TASK_FILTER_LIFE_P);
        break;
      …
      }
    …
    }
}
```

**Listing 1: Code for Ninety-Day Countdown Timer**

## Timer Issues

Salvo can be configured for 8-, 16- and 32-bit delays. Delays are measured in units of system ticks. At 10ms per tick, 90 days is 777,600,000 system ticks, less than 2^32. We could configure our application for 32-bit delays and delay `TaskFilterLife()` (above) via `OS_Delay(777600000)`. However, by utilizing 8-bit delays we can minimize the size of all of the task control blocks,[4] thereby minimizing RAM usage.

In order to delay for longer than 255 system ticks with Salvo configured for 8-bit delays, we can use a loop construct around the call to `OS_Delay()`. The loop variable must be declared as `static`. In Listing 1 above, 10ms x 200 x 3,888,000 is the number of seconds in ninety days.

## Application Behavior

`TaskFilterLife()` is created with a low priority. At startup, it begins running as soon as it is the highest-priority eligible task. It delays itself immediately, and every 2 seconds thereafter it decrements `longCounter` until it reaches zero ninety days later. Then, it sets the global flag `filterNeedsChanging`[5] and stops itself. It will remain stopped indefinitely, until … the homeowner replaces the filter and resets the indicator by pressing the `RESET_FILTER` key. Then `TaskFilterLife()` will begin counting down again.

## RAM usage

One task control block is required for `TaskFilterLife()`, and four bytes are required for `longCounter`, for a total of 9 bytes[6] to implement this functionality.

## Saving RAM

With an initial value of 3,888,000, the uppermost byte of `longCounter` is always zero. This means that only 24 bits are needed, and instead of a single 32-bit counter we could cascade a 16-bit and an 8-bit counter, as shown in Listing 2 below:

```
void TaskFilterLife ( void )
{
  static unsigned int intCounter;
  static unsigned char charCounter;


  for (;;) {
    intCounter = 19440;
    do {
      charCounter = 200;
      do {
        OS_Delay(200, TaskFilterLife1);
      } while ( --charCounter );
    } while ( --intCounter );

    filterNeedsChanging = TRUE;
```

```
            OSStop(TaskFilterLife2);
        }
    }
```

**Listing 2: Saving One Byte of RAM via Cascaded Counters**

The time to count down remains the same (10ms x 200 x 200 x 19440), but we use one fewer byte of RAM.

## Conclusion

By having `TaskFilterLife()` stop itself when the timer reaches zero, there's no need to further manage `longCounter` – all related activity stops until the task is restarted in `TaskKeyPad()`. Thus, the "Replace Filter" indicator, the ninety-day countdown timer and the desired application behavior are all easily implemented via a single Salvo task and a few bytes of RAM.

---

1 Heating, Ventilation and Air Conditioning.

2 An HVAC system with the ability to sense pressure (vacuum) on either size of the filter(s) could make a more intelligent assessment of the filter's useful life. But a three-month timer is much less expensive.

3 A typical value.

4 When configured for delays, each task's task control block contains RAM dedicated for delays, whether or not the task calls `OS_Delay()`. Hence it's advantageous to configure Salvo for the smallest possible delays.

5 A global flag is used here for simplicity only. `OSSignalBinSem()` could be used in place of the global flag as a (better) signaling mechanism to the display task.

6 Microchip PIC16 family of PICmicro MCUs.