Application Note AN2137

Subscriber Pulse-Metering Detector

By: Djessas Zoheir
Associated Project: Yes
Associated Part Family: CY8C26xxx
PSoC Designer Version: 4.1

Summary
This Application Note describes a low-cost, single-chip solution for detecting and monitoring the Subscriber Pulse Meter (Teletaxe) signal used for telephone call charges with the PSoC™ (Programmable System-on-Chip).

Introduction
The subscriber pulse-metering (SPM or Teletaxe) signal detector monitors and detects 12 kHz or 16 kHz call-charge pulses placed on the telephone line by the local telephone exchange.

Each SPM pulse represents "one unit" of charge. This unit-charge is a fixed rate in each country:

Cost = # Received SPM Pulses * Unit Charge.

Objectives
The objective of this design is implementation of an SPM detector in a PSoC for detection of 12 kHz SPM tones employed in the French payphone/card phone system. A system block diagram for an SPM detector is shown in Figure 1.

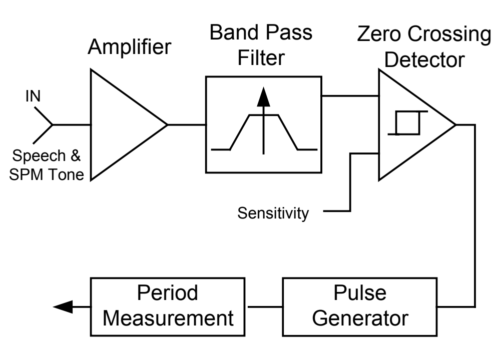


Figure 1. System Block Diagram

The amplifier, band-pass filter and zero-crossing detector can be implemented using PSoC analog blocks. The pulse generator and the period measurement logic can be implemented in software.

The resulting PSoC configuration is shown in Figure 2. The schematic is shown in Figure 3.

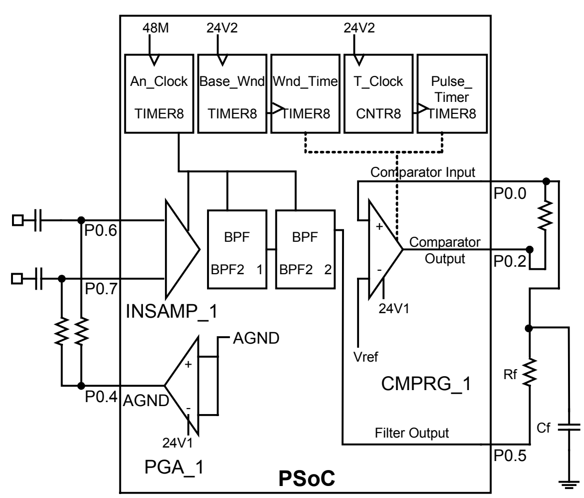


Figure 2. PSoC Configuration Diagram

Hardware

Voltage from the telephone line is often outside the operational linear range of the differential instrumentation amplifier inputs. To avoid damage to the device, there are two considerations when applying input protection:

- Continuous input current must be limited to less than 50 mA (R1 to R4).
- The input voltages must not exceed the supply by more than 0.5V (D1 to D4). [See Absolute Maximum Ratings from device data sheet.]

A hysteresis at the positive input of the comparator is implemented to accelerate the swing and confirm the state of the input, avoiding floating swing.

Telephone line sensor U5A detects the off_hook/on_hook state.

LEDs 0-7 are used for debugging and display. They can be replaced by a 2x16 LCD for displaying the units or associated call charge in the appropriate format.

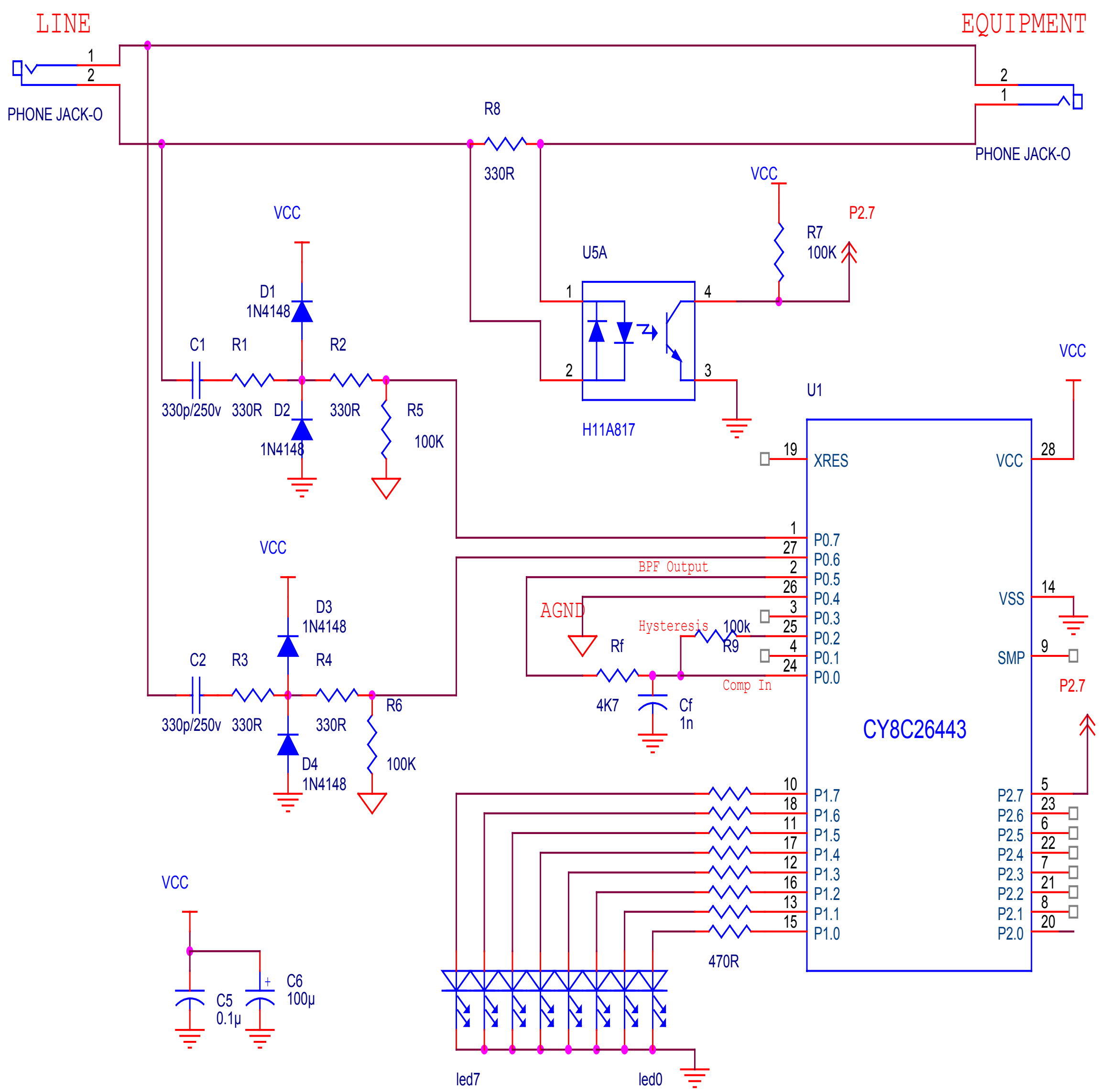


Figure 3. Electrical Diagram

Software

The time between the assertion and de-assertion of the output is 40 ms minimum and 48 ms maximum.

The SPM detector responds when 40 ms of valid tone is received within any 48-ms window, signifying receipt of a valid packet of SPM tone. Once it has responded, within any 48 ms of no-tone, tone outside the chosen bandwidth, or below the level threshold, the output will be de-asserted.

The main program first sets the device to powersave mode (sleep at 1 Hz) and waits for the telephone line to be in off_hook state. Once this occurs, it wakes up the processor, which sets the analog and digital blocks, and waits for an SPM tone.

The Program Counter (PC) stays in a main loop waiting for events (interrupts) to be performed. These events can be:

- A rising edge of tone with required level (comparator interrupt: comp_sub).
- A pulse measurement timeout (timer interrupt). Note: Pulse = 32 cycles [12 kHz].
- The Response/De_response timer timeout (units_wnd interrupt).

The first cycle of an SPM tone, with the required level, starts the wnd_timer for a 48-ms time out. Each time the comparator triggers, it interrupts the main loop for the period measurement procedure (see Comp_sub subroutine ahead). Figure 4 shows the main and comparator interrupt flowcharts.

When the pulse count reaches 15, before the wnd_timer expires (Response Time: 15 x 32 pulses = 40 ms), the last pulse is stopped, the pulse timer interrupt is enabled, and one unit-call charge is valid.

As long as the 12 kHz signal exists, the comparator trigger interrupts the main program for the pulse measurement procedure (Comp_Sub subroutine). In this case, the pulse measurement is not taken into account and the Pulse_Timer interrupt is enabled and used to detect the end of the SPM tone. See the following source.

Each measure of the pulse is proceeded by:
In *Comp_sub.asm*:

```
Pulse_on:
    ;one pulse = 32 cyles of 12 kHz signal
    call  bPulse_Timer_CaptureCounter
    mov   [rate],A
    call  Pulse_Timer_Stop.
```

The

```
    call  bPulse_Timer_CaptureCounter
```

instruction interrupts the procedure (Pulse_Timer interrupt is enabled) for:

```
Pulse_TimerINT:
    push  a
    call  bPulse_Timer_CaptureCounter
    cmp   a,255
    jc    tim_loop
    mov   [error_cnt],10  ;any value >4
    call  Pulse_Timer_DisableInt
tim_loop:
    pop   a
    reti
```

The value of Pulse_Timer_CaptureCounter register is the gap between each 32 cycles until the last cycle (< 32) is interrupted by:

```
    Call   bPulse_Timer_CaptureCounter
```

The instruction does not occur (end of 12 kHz signal) and the Pulse_Timer continues to run until it is timed out. Then the terminal count interrupts the program for the Pulse_TimerINT procedure. The Pulse_Timer counter is reloaded by the value 255 and the main program enters the De_response evaluation loop.

```
; in _main:

loop2:
    mov   a, [error_cnt]
    cmp   a,4  ;see error_cnt above :10
    jc    loop1
    tst   [flag], F_WND_TIMER_DERESP_OUT
    jnz   main_loop .
```

The wnd_timer is restarted again for de-response time. When it expires the process is restarted.

Comp_sub subroutine:
The first few cycles are not processed in order to avoid spurious noise pulses. The subroutine reads and restarts the pulse timer every 32 cycles (one pulse). At each pulse it measures the repetition pulse rate against a predetermined maximum and minimum, depending on the frequency within the legal bandwidth (12 kHz ±3% or ±5% or ± 7.5%).

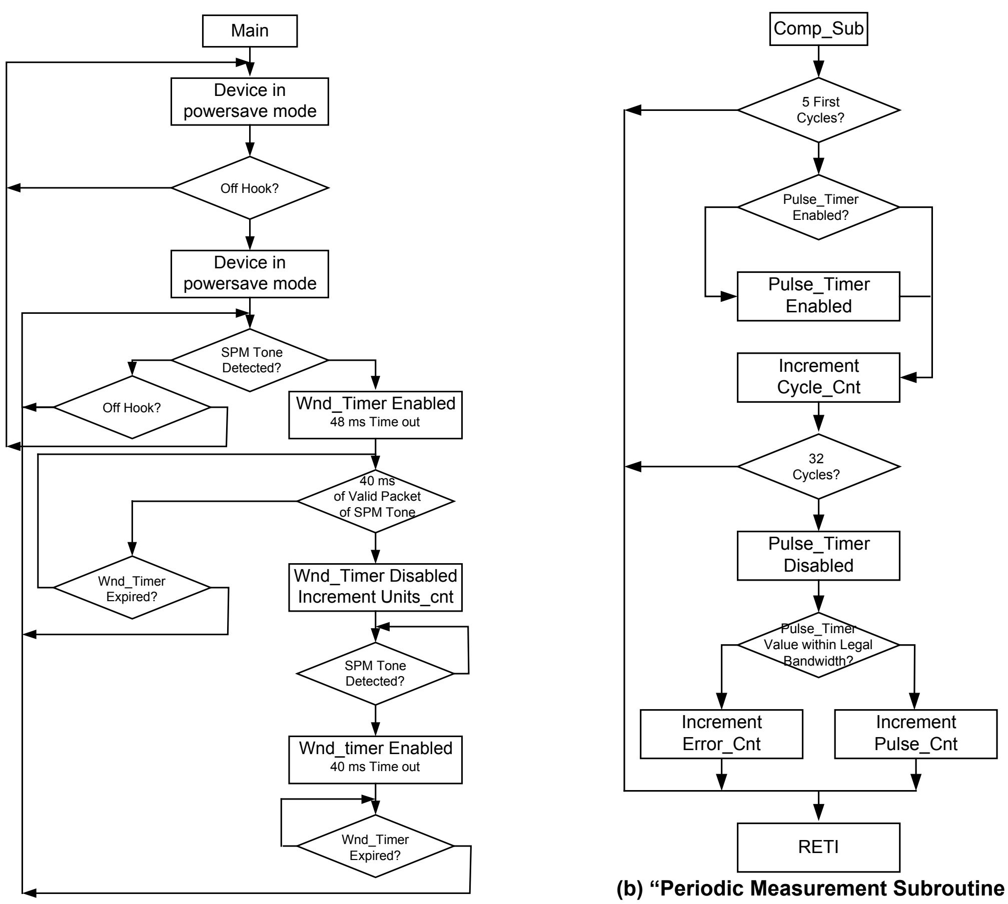
In *Comp_sub.asm*:

```
pulse_on:
;one pulse = 32 cyles of 12 kHz signal
    call   bPulse_Timer_CaptureCounter
    mov    [rate],A
    call   Pulse_Timer_Stop
    ;reload and restart the Pulse_Timer
    mov    A, 255
    call   Pulse_Timer_WritePeriod
    call   Pulse_Timer_Start
    mov    [pulse_cnt],32
    cmp    [rate],  LIMIT_H
    jnc    limit_out
    cmp    [rate], LIMIT_L
    jc     limit_out
    dec    [pack_cnt]
    jmp    pop_ret
limit_out:
    inc    [error_cnt]
pop_ret:
    pop    A
    reti
```

If the pulse period is out of range, the error counter is incremented. The main loop resets the process if it reaches three errors.

Conclusion

Presented has been a low-cost, PSoC solution for detecting and monitoring the Subscriber Pulse Meter signal used for telephone call charges.



(a) "Main Program"

(b) "Periodic Measurement Subroutine

Figure 4. Software Flowchart

About the Author

Name: Djessas Zoheir
Contact: djzoheir@caramail.com

Cypress MicroSystems, Inc.
2700 162nd Street SW, Building D
Lynnwood, WA 98037
Phone: 800.669.0557
Fax: 425.787.4641
http://www.cypress.com/ / http://www.cypress.com/support/mysupport.cfm