February 6, 2018

v1.1

Polling Routine for Push-Button Switches

This is a timer-based periodic interrupt routine for polling the state of one or more momentary push-button switches. At an appropriate polling frequency, the same interrupt could be used to service both the momentary switches and a quadrature rotary encoder.

The routine does not require any hardware debouncing of the switches, nor does it wait for switches to settle during any interrupt. The algorithm inherently recognizes as a pressed button any button which is pressed at any time during the servicing period. It also distinguishes between a short press and a long press, and will register the press of a single button or any combination of multiple buttons. With N buttons, counting the press of any single button, or any two buttons, and with short or long for each, the total number of recognized patterns is N(N+1). So with five buttons, there would be 30 possible unique press patterns which could theoretically be used.

The coder must select values for two down-counters used in the routine, which will be based on the polling frequency. The first we will call LNGCNT, or long count, which is the number of interrupts it takes to recognize a button press as "long". So if a long press must be held for at least two seconds, and the polling frequency is 500 interrupts per second, then LNGCNT would be 1000. The second value is RELCNT, or release count, which is the minimum number of consecutive interrupts, during which all switches are open, needed to recognize a valid release. Typically RELCNT covers 1/10 to 1/4 second, depending on the release bounce behavior of the switches. If the time is set too short, bouncing may continue past the release count, and the release may register as another press.

The actions taken by the routine depend on what MODE it is in. Setup initializes the routine in MODE 1, in which there is no pending activity, and the routine is just waiting to detect a button press of some kind. When it does so, it switches the routine into MODE 2 for subsequent interrupts, during which it determines which buttons were pressed, and whether short or long, and then executes whatever that button-press is intended to do.

For a long press, it will then shift the routine into MODE 3, which simply awaits the release of all buttons, not recognizing any button presses as valid until that release is completed, and then shifts the routine back into MODE 1 where the process starts all over again. The MODE 3 transition allows the appropriate long button press action to be performed as soon as the LNGCNT time has passed even though the buttons are still being held down at that point. Eventually, all buttons must be released before any button presses will be recognized.

A short press is one that is released before the LNGCNT has expired. It is recognized only after a full release has occurred, so there is no need for MODE 3, and it can simply transition directly into MODE 1.

Since the service routine shifts back and forth a good bit between various decision points, a flow chart diagram is most useful in understanding how everything works. The following constants and variables are used in the diagram:

MODE The processing mode the routine is currently in. Initialized at MODE 1 in setup.

PINS The current state of the port pins as read on this interrupt. All bits not connected to buttons are set to 1. Any buttons pressed will read as 0.

PUSHED All buttons which have been pressed at any time during MODE 1 or MODE 2.

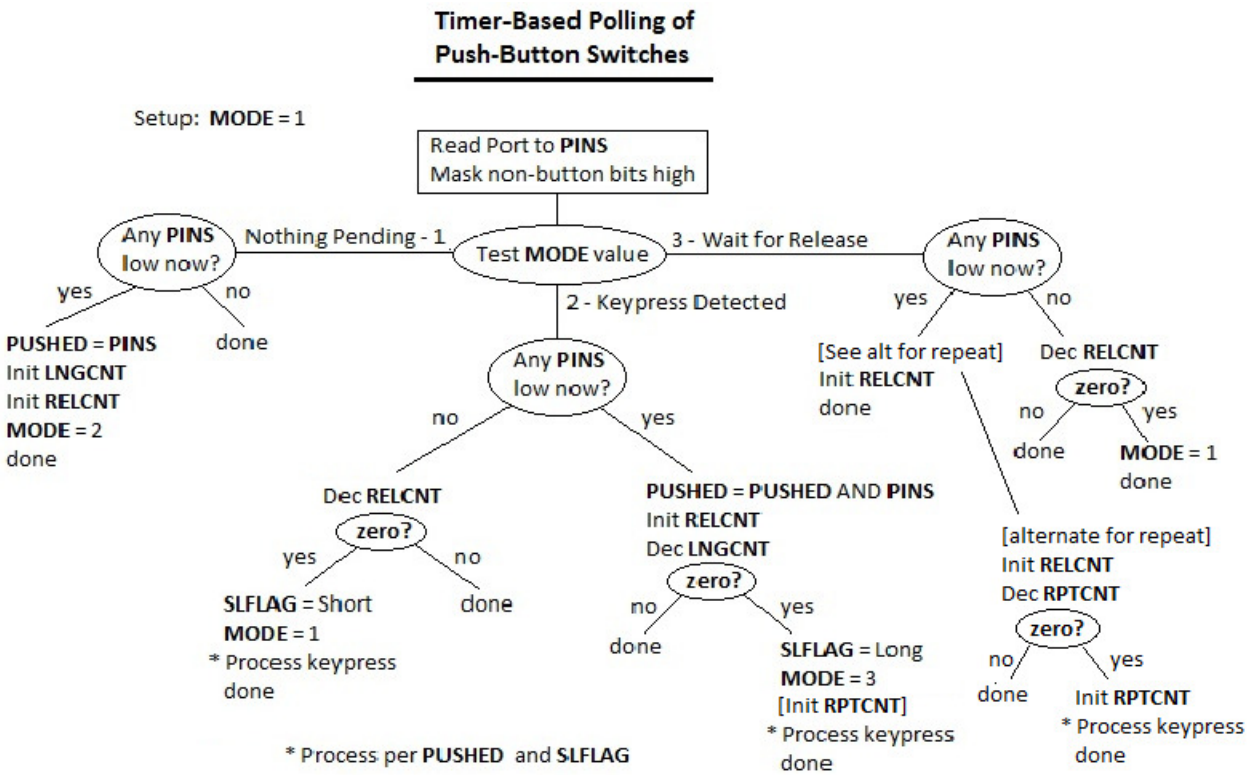
LNGCNT Long count. The minimum number of interrupts needed for a press to be long.

RELCNT Release count. The minimun number of consecutive all-high readings for a release.

[RPTCNT] The number of interrupts between repeats on continuing to hold a long press, if used.

SLFLAG Short/long flag. Indicates whether the press was short or long.

done Return from interrupt



If continuing to hold a long press is meant to go into repeat mode, an alternative to handle that is shown in **MODE 3** using **RPTCNT**, which is the number of interrupts between repeats. If repeat is only for some presses, an additional test involving something like RPTFLAG would be needed.

The combination of the **PUSHED** and **SLFLAG** determine the button press(es) detected. Find the **PUSHED** value in a table of values being used, and that provides the index into a table of vectors to the corresponding processing routines. If no match is found, just ignore that unused press pattern. A processing routine should not be so long that the next button press, or repeat, will overrun it.