Application Note: A392-001
Using the JSB392 to detect contact states

J-Works, Inc.
www.j-works.com

1. Overview

This application note provides information about how to use the J-Works model JSB392 USB Switch / Sensor Module in a typical user application when attaching to contact closures (NO normally open and NC normally closed).

The JSB392 input provides a IN+ terminal that has an internal pull-up so when not connected will be at a known state. It is intended that a contact closure is wired between this input (IN+) and the IN- (Ground) so no external power source is required. (see figure 3 & 4) The correct polarity or meaning of the contact state is determined by the type of contact closure and that should be used in the monitoring software.

When a mechanical switch or contact is closed and sometimes even when opened, it will bounce briefly and produce multiple electrical edges (spurious signals) rather than a single high/low transition. You want to avoid false reading of these values due to this. The process of eliminating spurious reading is called debouncing. Shown in this document is a simple software solution to remove the multiple edges and only report when the contact is in a steady state.

If the switch being monitored is being pushed by a person, code can also detect and decide what to do if the person holds down the switch for a longer period of time. Sometimes an application may want to provide additional feedback to the user, like help screens.

Wiring detail is shown for a simple switch that has a normally closed and normally open contact, but can be adapted to any type of contact closures not limited to switches or relays. The samples in this document are coded in C++ and using the DLL provided on the JSB392 CD (or download). Any programming language that can access our device, provided DLL's, Class Libraries, Linux software, or various third party software packages can also be use by translated the algorithms from the C++ example to the required language. Sample code is shown in a straight forward manner, not always using advance features of the language or OS. A user may also want to add additional error checking for their own application as required.

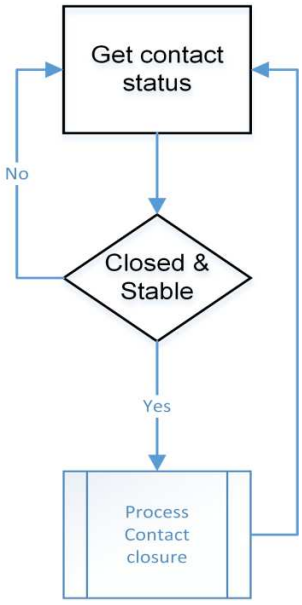

Figure 1 Basic Flow

2. Detail Flow Diagram of sample code (only switch detection)

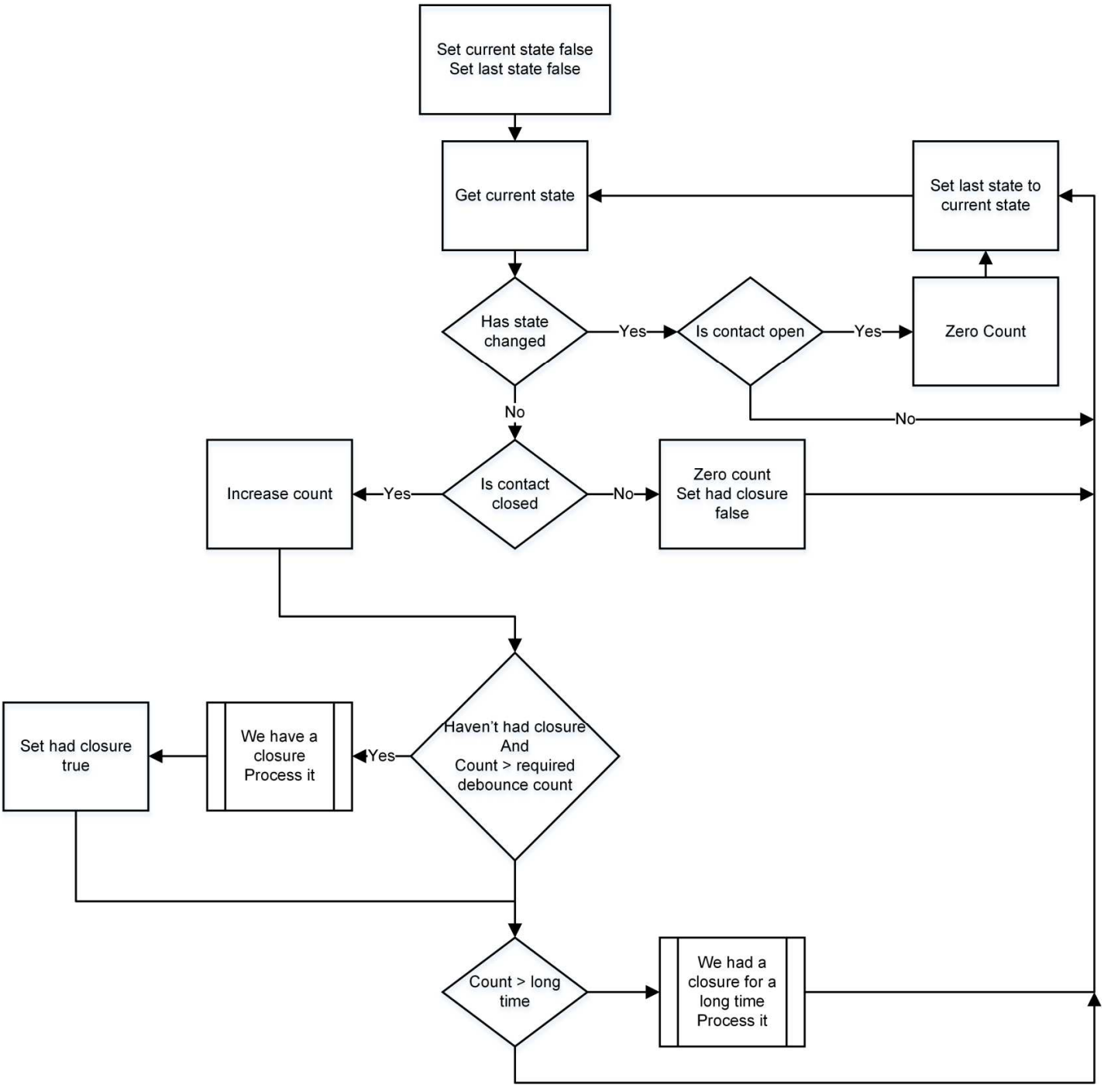

Figure 2 Sample code flow

3. Sample Code to find and start communication with JSB392

```
// see how many modules are present
int nNumberOfModules = Jsb383NumberOfModules();
// get the serial number of the 1st module, need serial number to know which module are accessing
CString serialNumber;
char * pChar = serialNumber.GetBuffer(32);
Jsb383SerialNumber(1,pChar,32);
serialNumber.ReleaseBuffer();
// now any future commands can use this serial number to access the module
// the full switch sample code displays this in a list box, and also obtain other
// module information that isn't always needed, but is there.
```

4. How to Connect using a switch with NO (normally open contacts)

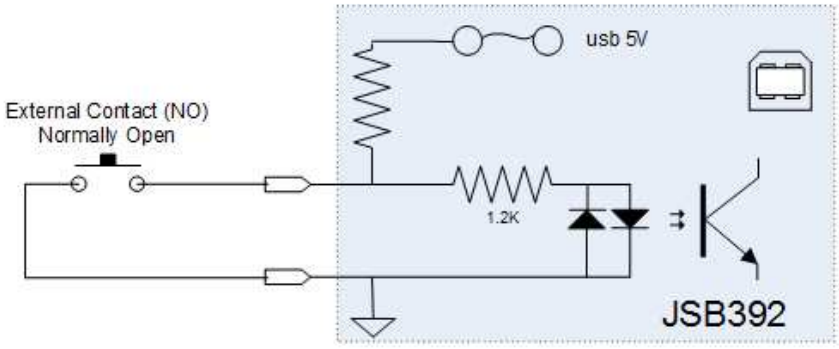

Figure 3 Wiring Diagram

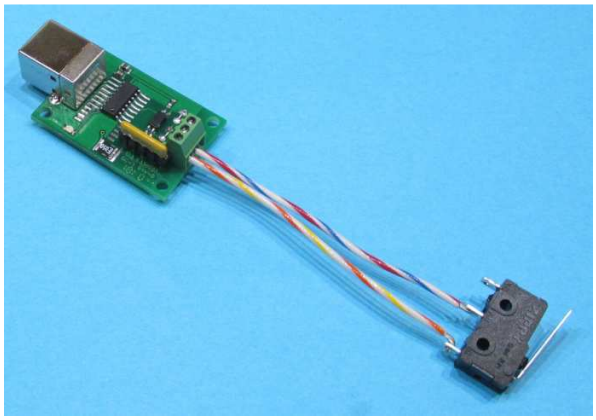

Figure 4 Actual Test Configuration

Note that any type of switch or contact other than the one used can be use instead.

5. Overview of Code for NO detection

```
// these are globals for switch debounce and edge detection

// last state read from module
bool lastState = false;
// had a closure, need to now wait for open before detecting another closure
bool hadClosure;
// how long state has been stable
unsigned int cntClosed = 0;
// how long to wait for switch debounce based on typically switch and 25ms poll interval (user selected)
#define DEBOUNCE_COUNT  2
// sometimes you may want to do another action if switch is down for a long time, user selected
#define LONG_TIME_DOWN  40
// what the input means is based on what type of contact used (NO or NC)
// for NO,  there value will be opposite when using nc
#define CONTACT_OPEN false
#define CONTACT_CLOSED true

// In the sample the following code is in a onTimer callback that is setup for a 25ms interval.
// The code can also be done in a thread that sets a event that can be used by another thread
// application

// get the current state of the only input of the Jsb392
bool currentState = Jsb383IsInputOn(serialNumber,1);
bool changedState = (lastState == currentState);
if (changeState) //we have a new state
        {
        if (currentState == CONTACT_OPEN) // contact is open, could be bounce or opening, don't care
                {
                cntClosed = 0; //start debounce count over
                }
        }
else  // contact is the same as last time, so if it closed then check debounce
        {
        if ( (currentState == CONTACT_CLOSED) && (!hadClosure))
                {
                cntClosed++;
                }
        else
                {
                cntClosed = 0;// whenever switch is open, don't count
                hadClosure = false; // since we are open, any closures are done
                }

        if ( (cntClosed >= DEBOUNCE_COUNT) && (!hadClosure))
                { // we have a valid switch closure after debounce
                // do what you need to do on valid switch closure just happening
                }
        if (cntClosed >= LONG_TIME_DOWN)
                { // switch has been down for quite awhile now
                  // maybe do something else because of this,
                }
        }
```

6. How to Connect using a switch with NC (normally open contacts)

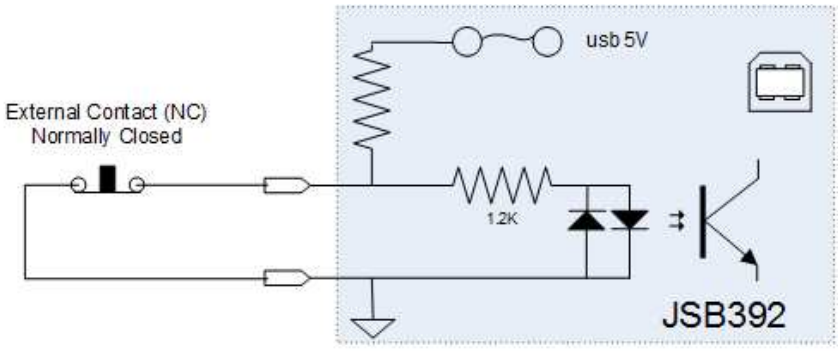

Figure 5 Wiring Diagram

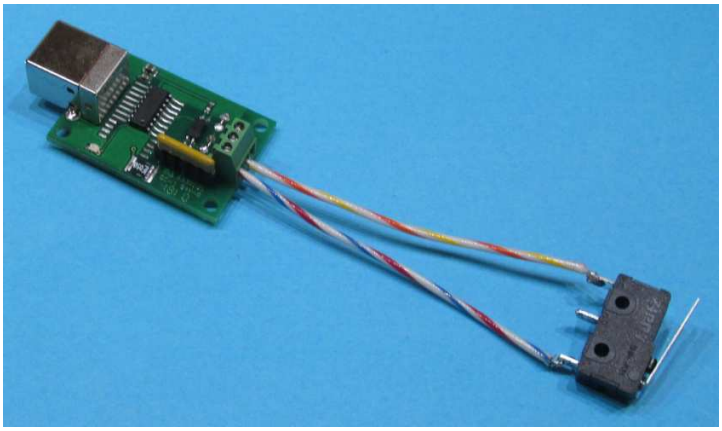

Figure 6 Actual Test Configuration

Note that any type of switch or contact other than the one used can be use instead.

7. Overview of Code for NC detection

```
// these are globals for switch debounce and edge detection

// last state read from module
bool lastState = false;
// had a closure, need to now wait for open before detecting another closure
bool hadClosure;
// how long state has been stable
unsigned int cntClosed = 0;
// how long to wait for switch debounce based on typically switch and 25ms poll interval (user selected)
#define DEBOUNCE_COUNT  2
// sometimes you may want to do another action if switch is down for a long time, user selected
#define LONG_TIME_DOWN  40
// what the input means is based on what type of contact used (NO or NC)
// for NO,  there value will be opposite when using nc
#define CONTACT_OPEN true
#define CONTACT_CLOSED false

// In the sample the following code is in a onTimer callback that is setup for a 25ms interval.
// The code can also be done in a thread that sets a event that can be used by another thread
// application

// get the current state of the only input of the Jsb392
bool currentState = Jsb383IsInputOn(serialNumber,1);
bool changedState = (lastState == currentState);
if (changeState) //we have a new state
        {
        if (currentState == CONTACT_OPEN) // contact is open, could be bounce or opening, don't care
                {
                cntClosed = 0; //start debounce count over
                }
        }
else  // contact is the same as last time, so if it closed then check debounce
        {
        if ( (currentState == CONTACT_CLOSED) && (!hadClosure))
                {
                cntClosed++;
                }
        else
                {
                cntClosed = 0;// whenever switch is open, don't count
                hadClosure = false; // since we are open, any closures are done
                }

        if ( (cntClosed >= DEBOUNCE_COUNT) && (!hadClosure))
                { // we have a valid switch closure after debounce
                // do what you need to do on valid switch closure just happening
                }
        if (cntClosed >= LONG_TIME_DOWN)
                { // switch has been down for quite awhile now
                  // maybe do something else because of this,
                }
        }
```