Semaphore (class)

A semaphore is a means of controlling access to the connection, so that we don’t have concurrent sessions. It’s a “traffic light”.

The constructor of the semaphore object simply starts an empty object with a flag stating the semaphore is unlocked.

Because we don’t want to have more than a single semaphore instance, the invocation of the semaphore isn’t done by mySemaphore=Semaphore. The constructor is a private function. Invocation is done by calling the me() method of the semaphore class. This is a static function which means it can be directly invoked without an active instance: mySemaphore=Semaphore.me()

Me()

This static method uses a persistent variable to make sure there’s only a single instance of the class (I think this is called the singleton design pattern)

If the uniqeInstacne persistent variable is populated, the return value of the function is the value. If not, a new semaphore object is instantiated and returned.

If an argument is given to the me() method, the name of the object is set to the arguent. (I don’t know what the name field of the semaphore object is used for at the moment)

Locked()

bool, checking if the object is locked.

Lock()

Locking the objected. If it’s already locked, it waits 10 seconds for release.

Release()

Unlocking the object. What if someone else already locked this object? Can I just release it at will?

Tcp2Labview (class)

This is the class that enables communications between labview and matlab. (TCP) important!

establishConnection(obj)

the main function in the ctor is this one. The tcpip() class is used to establish a new tcpip connection if one doesn’t already exist. The connection need the ip address of the target and the port.

A new connection has an input and output connection buffer size set to 10kB.

The timeout time is set to 60seconds.

Delete

Closes the tcp connection and releases the semaphore.

Uploadcode()

Generates a semaphore and locks it.

It then tries to write a 0 to to a binary ‘file’ whose file ID is the TCP connection ID. If that fails, it tries to re-establish the connection.

Blocksize is initiated at 3 times the number of code lines. It doesn’t say why and I don’t understand why it might be.

The function does some type arithmetic that I am not used to. If you are trying to go over the code, it’s a good idea to read the typecast function documentation in matlab. Example 3 inside that code helped my especially as in hex notation you can see how matlab collects terms together when switching from int8 to int16.

The code seems to work like this:

Make an array containing cols 2,3,4 of the code block. This is just a place holder for what comes later.

Now take cols 1,2 where command and subcommand are stored, use the magic of typecast to take 2 elements of type int8 and combine them to a single int16 element. Put that in the first col of the new code matrix. Cols 2 and 3 of the new code matrix already have the correct data stored in them.

The code matrix eventually is converted to a long vector and written to the TCP connection. A trailing zero is written. (int16(0) this might be a EOF indicator, I don’t know. ASK NITZAN)

The rest of the function deals with the datablock. CHECK WHAT THIS IS

Updatefpga()

Sets the FPGA to download program

Execute

Can set the number of repetitions of the program