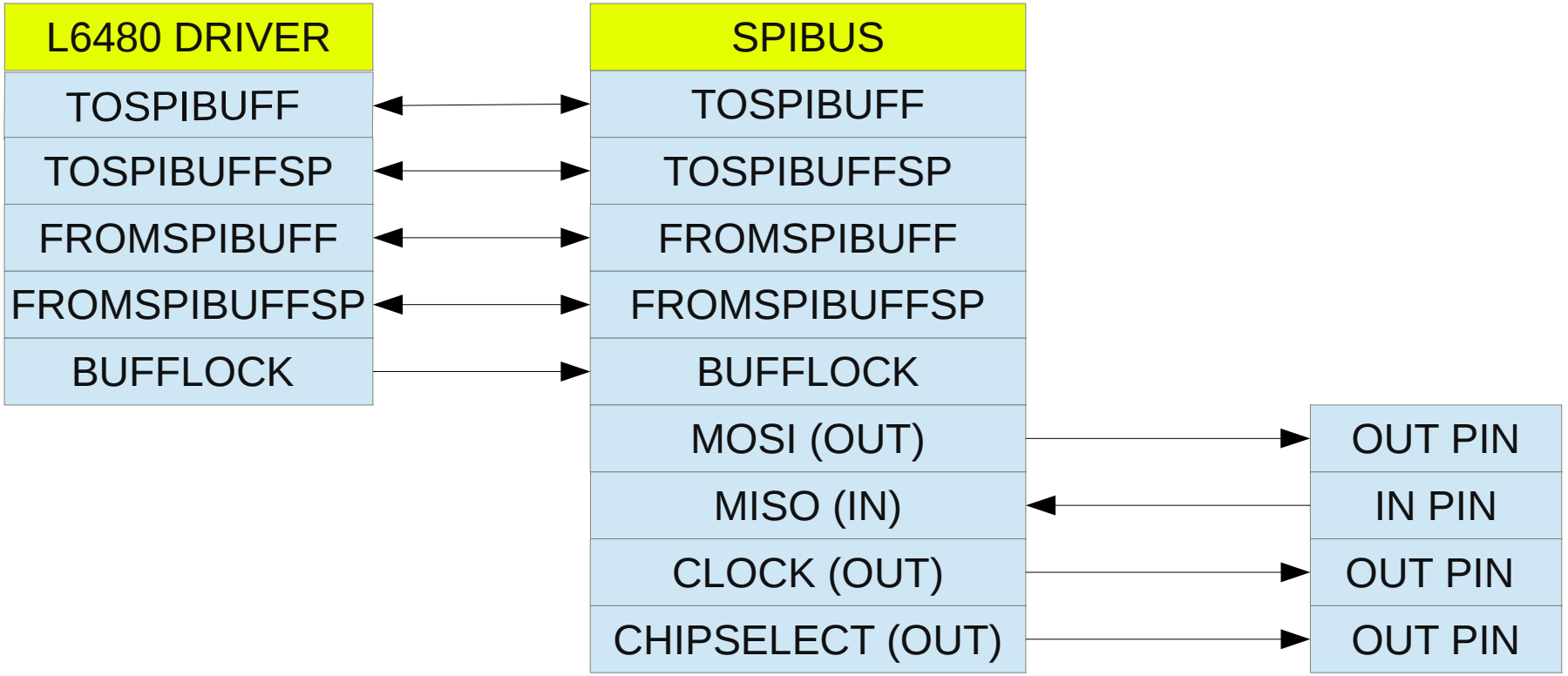
SPIBUS HAL Driver for LinuxCNC

- SPI (Serial Peripheral Interface) is a serial protocol for talking with memory, sensors, and other digital devices. It is Synchronous, meaning it has a separate clock line that tells the slaves on the network when data is valid and when they may write to the bus. It requires four lines: Clock, Master Out Slave In (MOSI), Master In Slave Out (MISO), and CS (Chip Select).
- This particular implementation was written out of a desire to talk with the ST Microelectronics L6480 stepper motor controller. Another module called L6480STEPPER is being written as a compliment to this generic SPI driver.
- The SPIBUS driver can map its pins to ANY available HAL pin through nets.



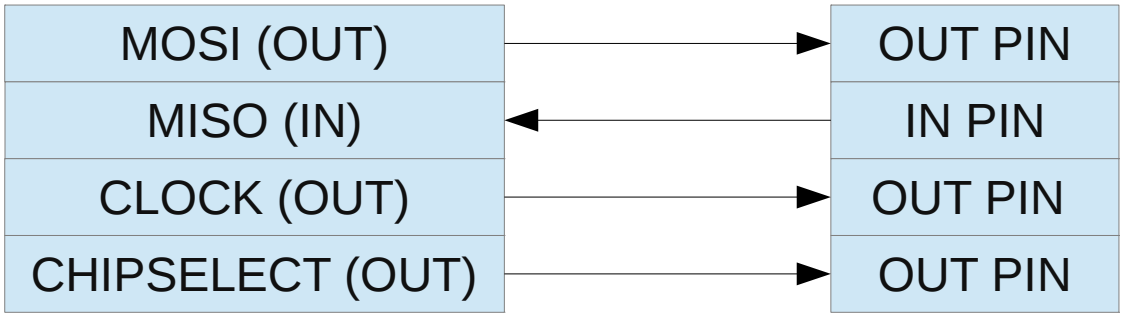
BUFFLOCK – PIN will be used to let the SPIBUS component know that it is being written to and that it cannot change the state of the buffers NOR the position of the stack pointers. The SPIBUS component will work off of a private copy of the buffers keeping track of how much data it has sent or received since the buffer was locked. After the lock is removed, it will update the buffers based on what transcribed since the lock was put in place. This is to prevent a race condition when SPIBUS is in the base thread and preempts the software talking to it. Before writing to the buffers or reading from them, BUFFLOCK should be and the other component should not change ANYTHING until the component is called again. Then it can modify the buffers, snapshot the FROMSPIBUFF/reste FROMSPIBUFFSP etc. After the modifications, BUFFLOCK can be set back to zero again releasing control of the buffers back to the SPIBUS component.

TOSPI Connections

- TOSPIBUFF – An array of unsigned [32]. Only 8 bits of the unsigned are used. Holds the bytes that will be sent to the SPI bus. The data in the buffer will be shifted down automatically as the SPIBUS component sends the data out. [0] is the oldest data, [31] will contain the newest.
- TOSPIBUFFSP – unsigned bi-directional – Contains how many bytes of data are in the buffer. When 0, the buffer is empty. When at 31, it is full.

FROMSPI Connections

- FROMSPIBUFF – An array of unsigned [32]. Only 8 bits of each unsigned are used. Holds the bytes that came from the SPI bus. The consumer of the data in the buffer is responsible for shifting the data in the buffer. The SPIBUS component puts the newest data at the highest slot in the buffer. [0] is the oldest data, [31] will contain the newest.
- FROMSPIBUFFSP – unsigned bi-directional – Contains how many bytes of data are in the buffer. When 0, the buffer is empty. When at 31, it is full. It is the responsibility of the consumer of the data to update this stack pointer.
- Important: before trying to read the data in the FROM buffers, use BUFFLOCK to freeze it. Wait a cycle. Read BUFFLOCK to make sure it is locked. Then modify the buffers to show you have read them and clear BUFFLOCK.



Those familiar with the SPI bus protocol will recognize the pin names.

Master out, slave in (MOSI) is used for the SPI driver to send data to some physical pins. It should be connected to an output pin. It will be connected to the data input on the slave devices.

Master In Slave out (MISO) is used to read responses from the slaves. It must be an input pin. It will be connected to the slave data output.

CLOCK – This is connected to an output pin on your LinuxCNC box and will be connected to a clock input on your SPI bus parts.

CHIPSELECT – This line (usually active LOW) goes to a low or zero state to signal the chip connected to it that it is being talked to. This allows multiple chips to talk on the same SPI bus with only a single additional line for each. Four wires for one chip, Five for two, etc.