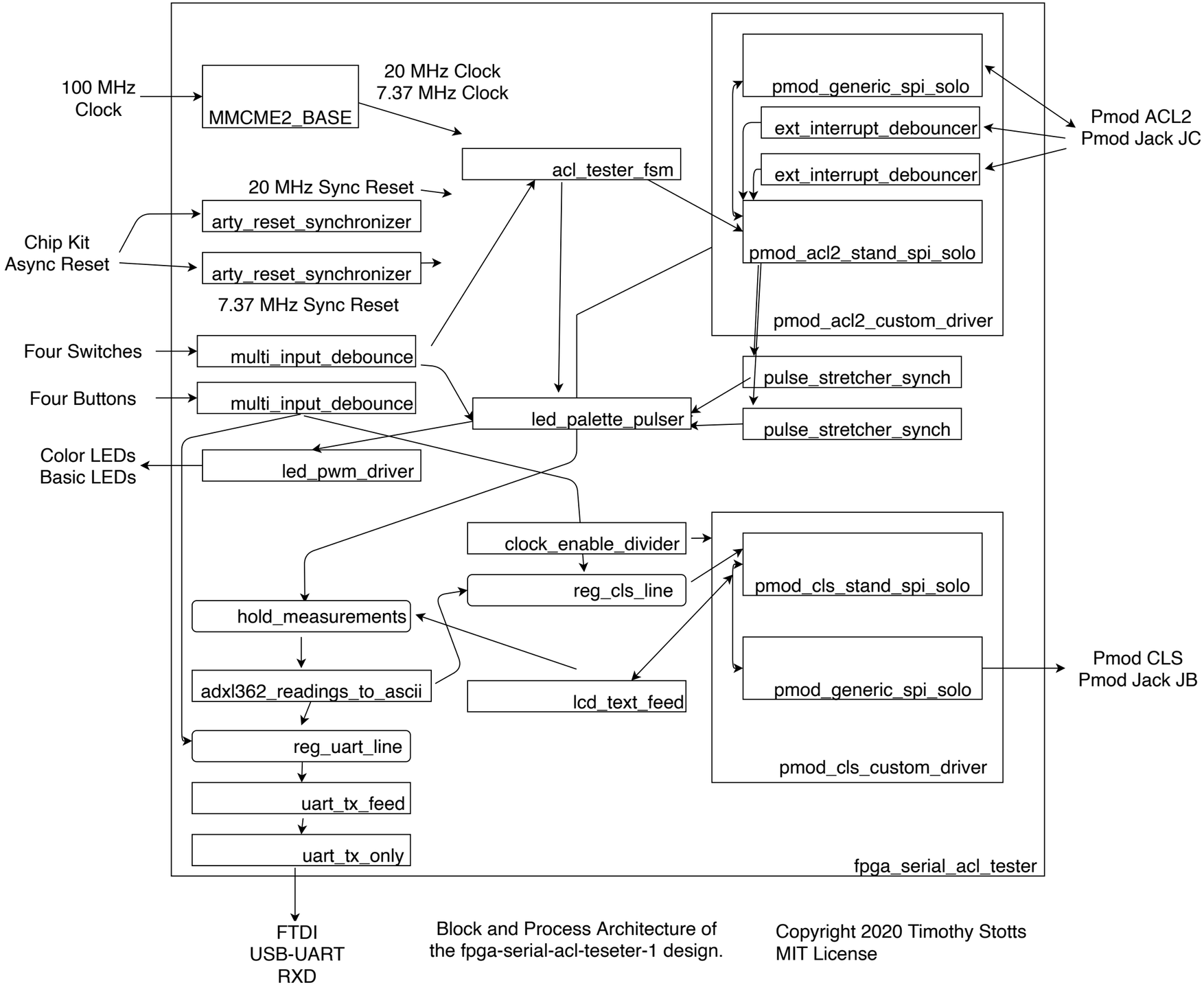
FPGA Serial Accelerometer Tester, Version 1

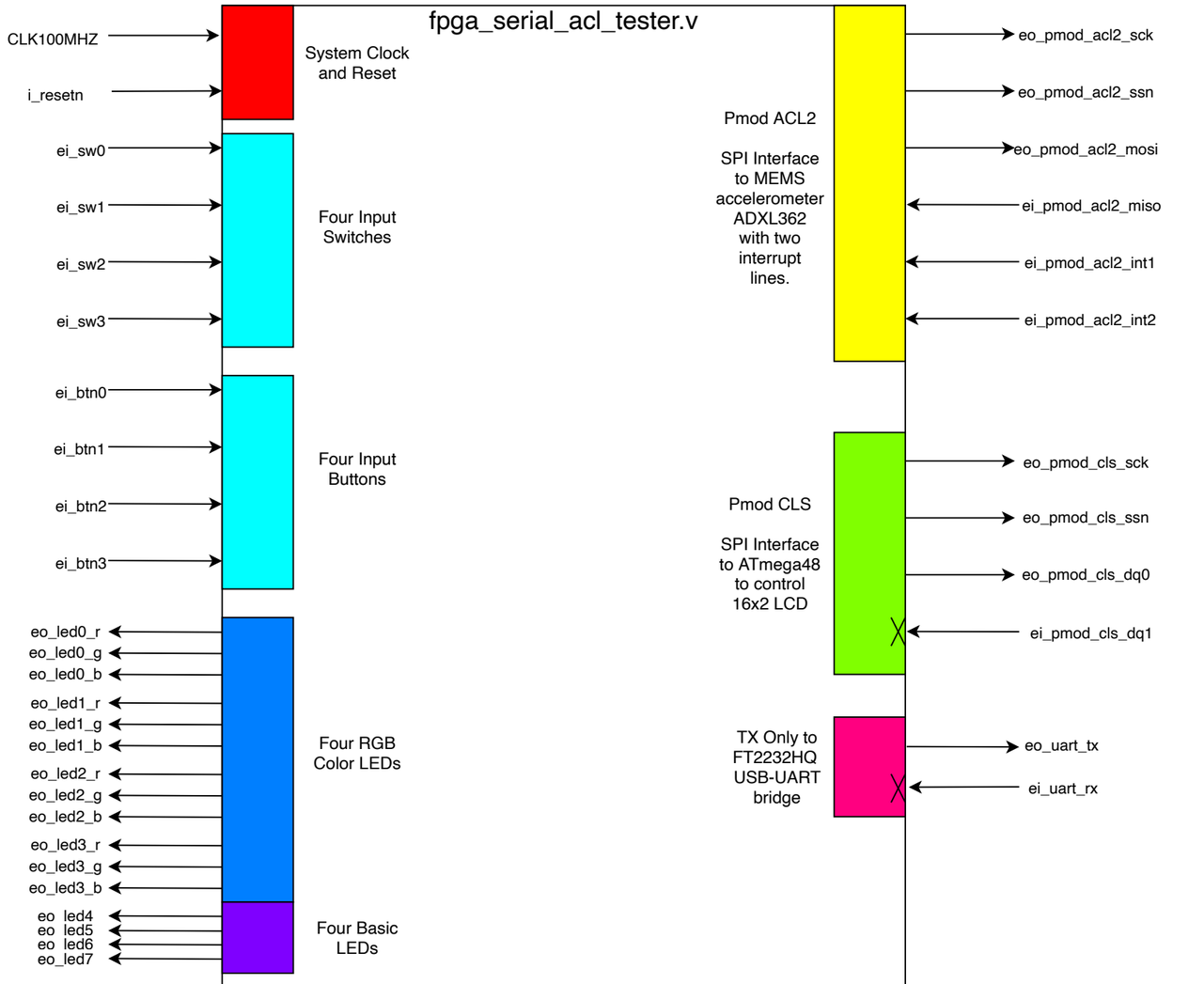
by Timothy Stotts

Hosted at:
https://github.com/timothystotts/fpga-serial-acl-tester-1

ACL-Tester-Design-Diagrams document revision 8A

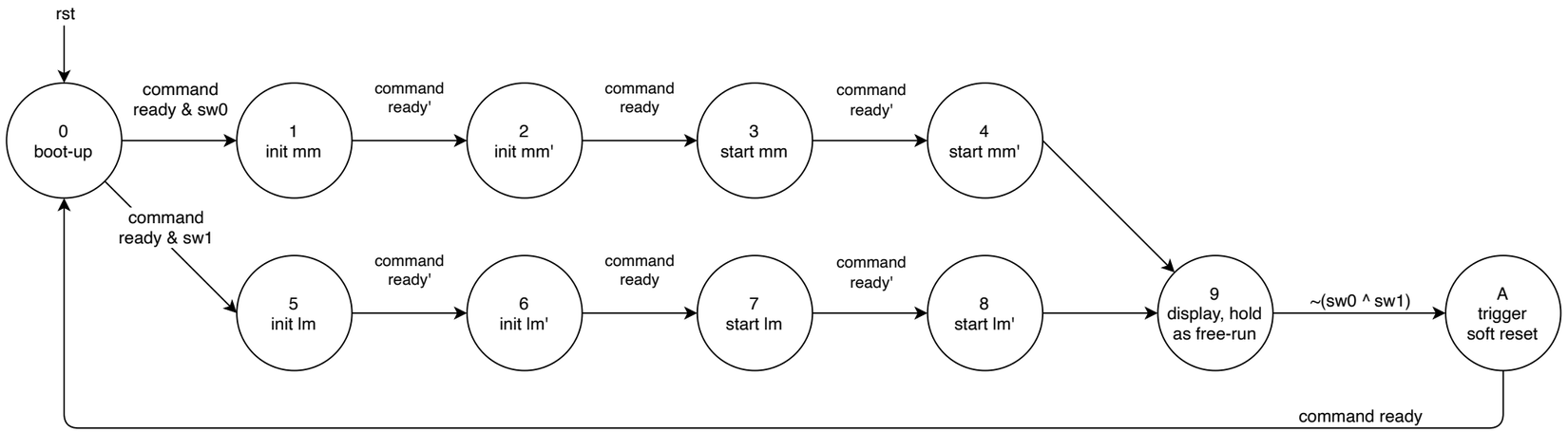


Block and Process Architecture of the fpga-serial-acl-teseter-1 design.

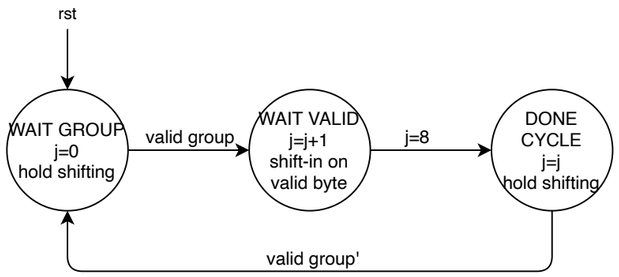


This top-level FPGA module operates the Pmod ACL2 to obtain measurement readings, converts these readings to ASCII data and text, and then outputs the ASCII data on the UART TX and the ASCII text on the Pmod CLS LCD. The switches select one of two possible operational modes: Stream Measurements to Display, or Detect Activity/Inactivity events and display event-trigger measurements. The event triggering is coded into the Pmod ACL2 driver with constants that define operation and thresholds. The color LEDs LD0 and LD1 display information about internal FSM states to indicate mode of operation and if the top-level FSMs are operating data reception from the Pmod ACL2. The color LEDs LD2 ad LD3 indicate Activity Event and Inactivity Event respectively. The basic LED LD4 indicates Awake status of the Pmod ACL2; and basic LD6 and LD7 indicate selection of SW0 and SW1 respectively.

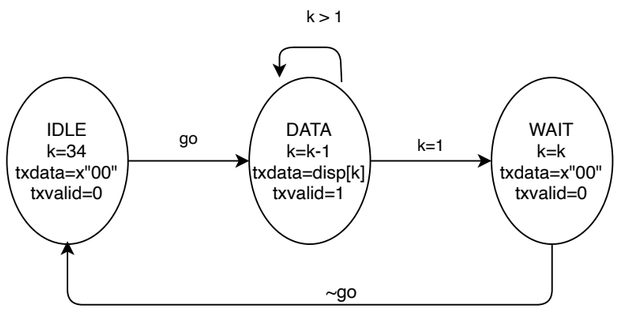
The Pmod CLS is selected to operate with SPI output and no SPI input, even though the Pmod CLS additionally supports UART and I2C modes.



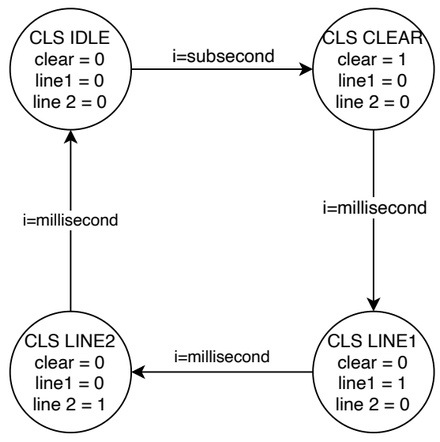
Tester FSM for operating the PMOD ACL2 driver commands.



Tester FSM to receive the streamed measurements and shift them into a bit vector.

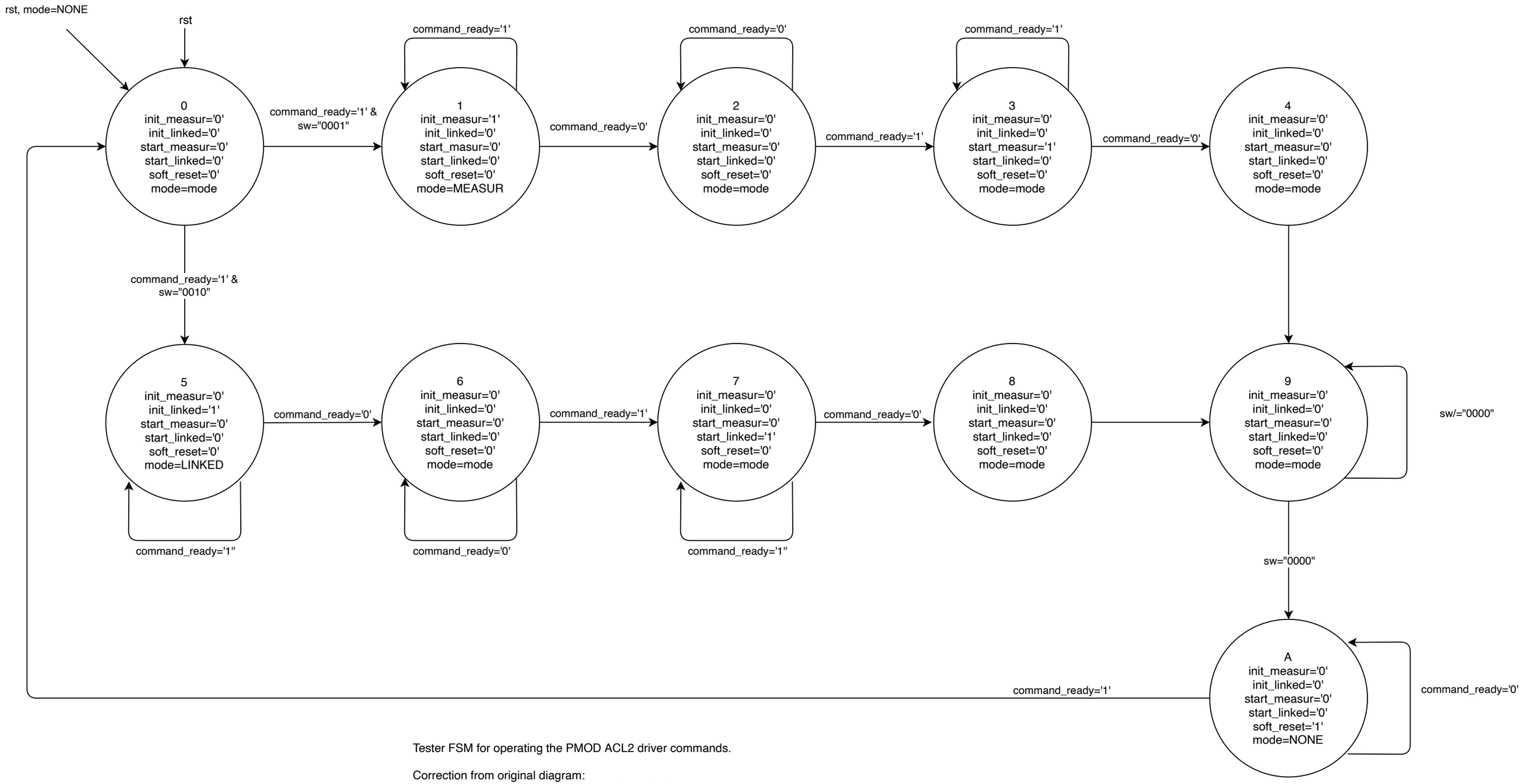


Tester FSM to load the TX ONLY UART with a 32 character text line, plus carriage return and new line.



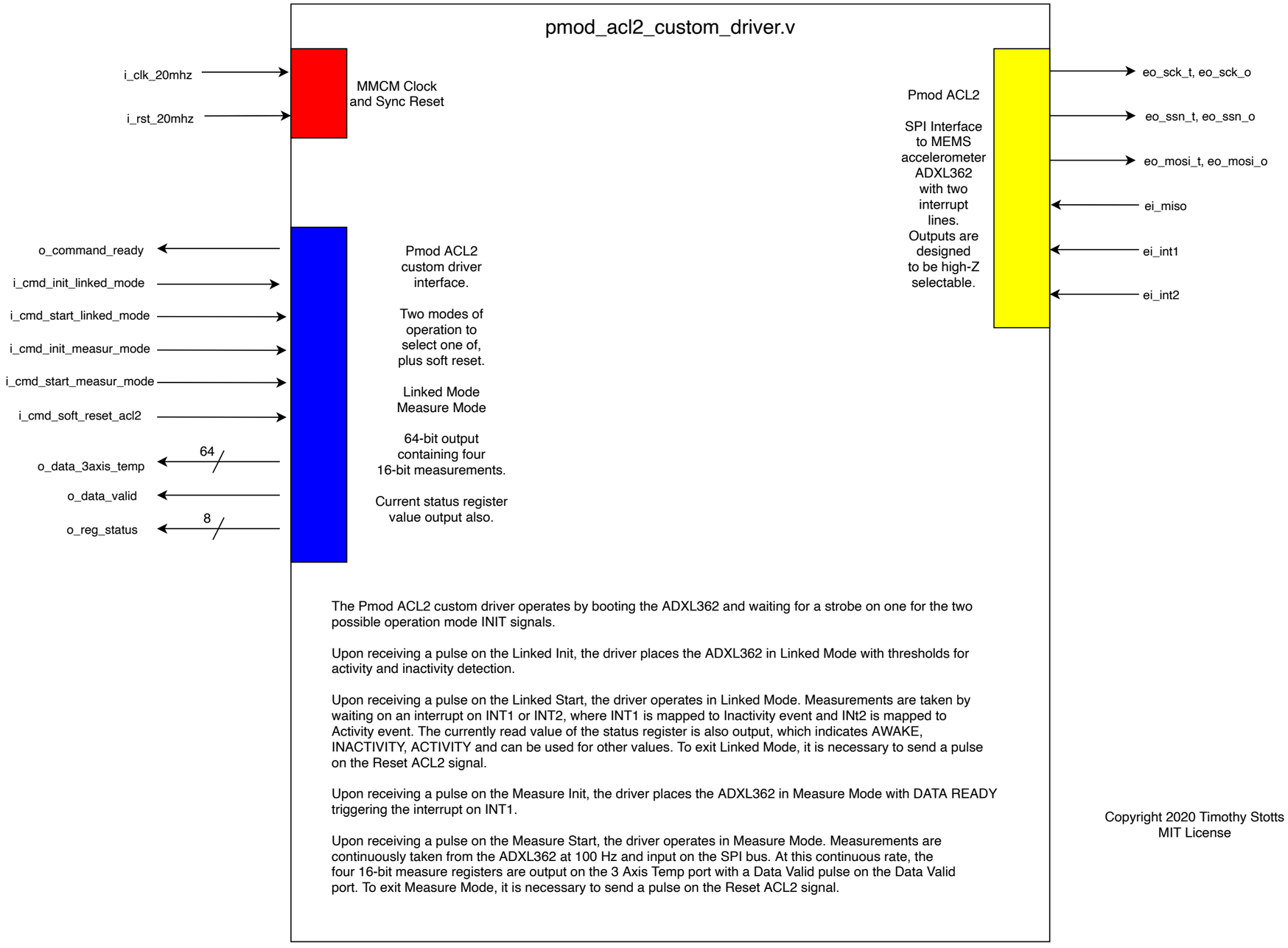
Tester FSM for updating the PMOD CLS display.

These diagrams are
incomplete. Some
state-bypass preventions
and iterations may not be shown.
These are the original diagram
draft prior to implementation.



Tester FSM for operating the PMOD ACL2 driver commands.

Correction from original diagram:
- details of the FSM outputs and recursive output
- transitions 9-A and A-0 are a unique sensitivity to input command_ready in order to pulse the output soft_reset for width of as many clock cycles as required, instead of only one clock cycle. The custom ACL2 driver inputs the soft_reset signal as a level interrupt to transition to SOFTRESET_CMD after completing the current SPI operations. The command_ready remains zero before and after assertion of the soft_reset signal, and changes to one when the soft reset is complete. Preventing state bypass for these two transitions is different than the other transitions of this FSM.



The Pmod ACL2 custom driver operates by booting the ADXL362 and waiting for a strobe on one for the two possible operation mode INIT signals.

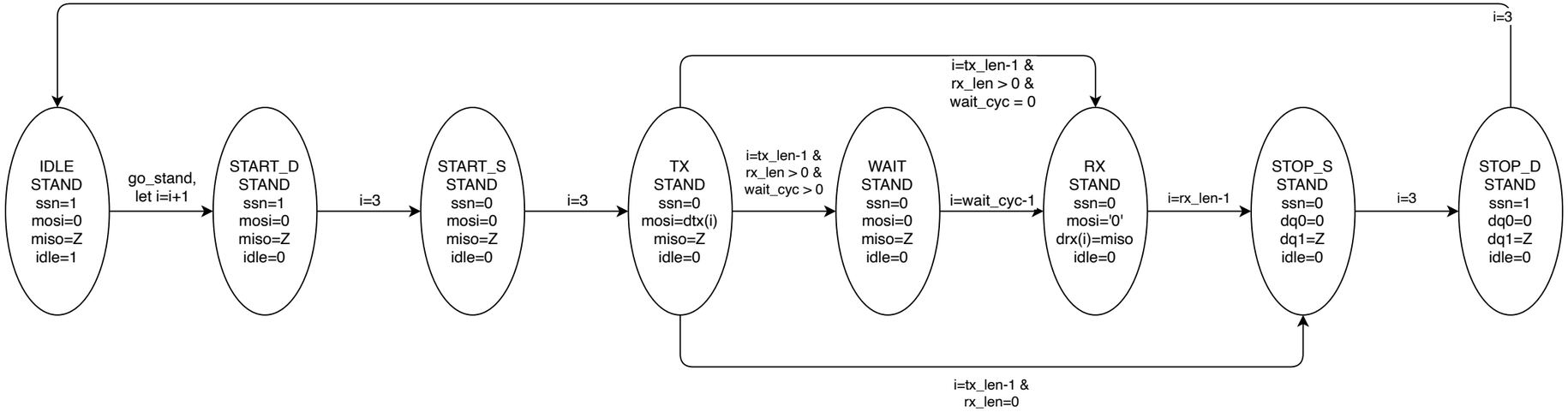
Upon receiving a pulse on the Linked Init, the driver places the ADXL362 in Linked Mode with thresholds for activity and inactivity detection.

Upon receiving a pulse on the Linked Start, the driver operates in Linked Mode. Measurements are taken by waiting on an interrupt on INT1 or INT2, where INT1 is mapped to Inactivity event and INt2 is mapped to Activity event. The currently read value of the status register is also output, which indicates AWAKE, INACTIVITY, ACTIVITY and can be used for other values. To exit Linked Mode, it is necessary to send a pulse on the Reset ACL2 signal.

Upon receiving a pulse on the Measure Init, the driver places the ADXL362 in Measure Mode with DATA READY triggering the interrupt on INT1.

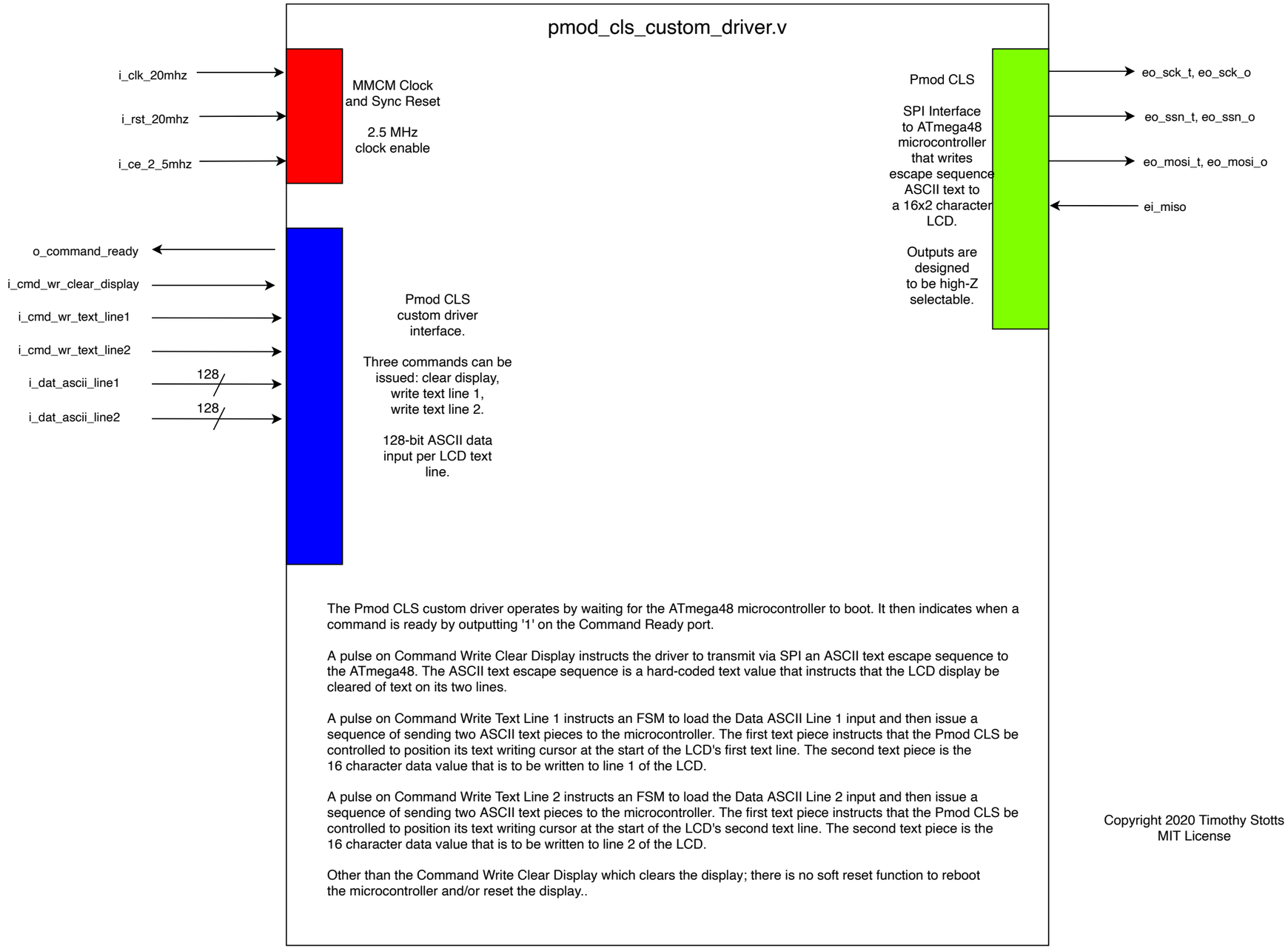
Upon receiving a pulse on the Measure Start, the driver operates in Measure Mode. Measurements are continuously taken from the ADXL362 at 100 Hz and input on the SPI bus. At this continuous rate, the four 16-bit measure registers are output on the 3 Axis Temp port with a Data Valid pulse on the Data Valid port. To exit Measure Mode, it is necessary to send a pulse on the Reset ACL2 signal.

Operational driver for the Digilent PMOD ACL2, that drives the Standard SPI FSM. This diagram is incomplete and does not show Soft Reset operation, or boot-time delay. Also, not all state-bypass preventions and not all iterations are show.



In each transition, tx_len and rx_len are to be multiplied by 8 from the FSM input signals, as it only makes sense to input into the FSM a byte count, while the FSM requires transitioning based upon a bit count.

Generic SPI FSM, with only one SPI slave on the bus.



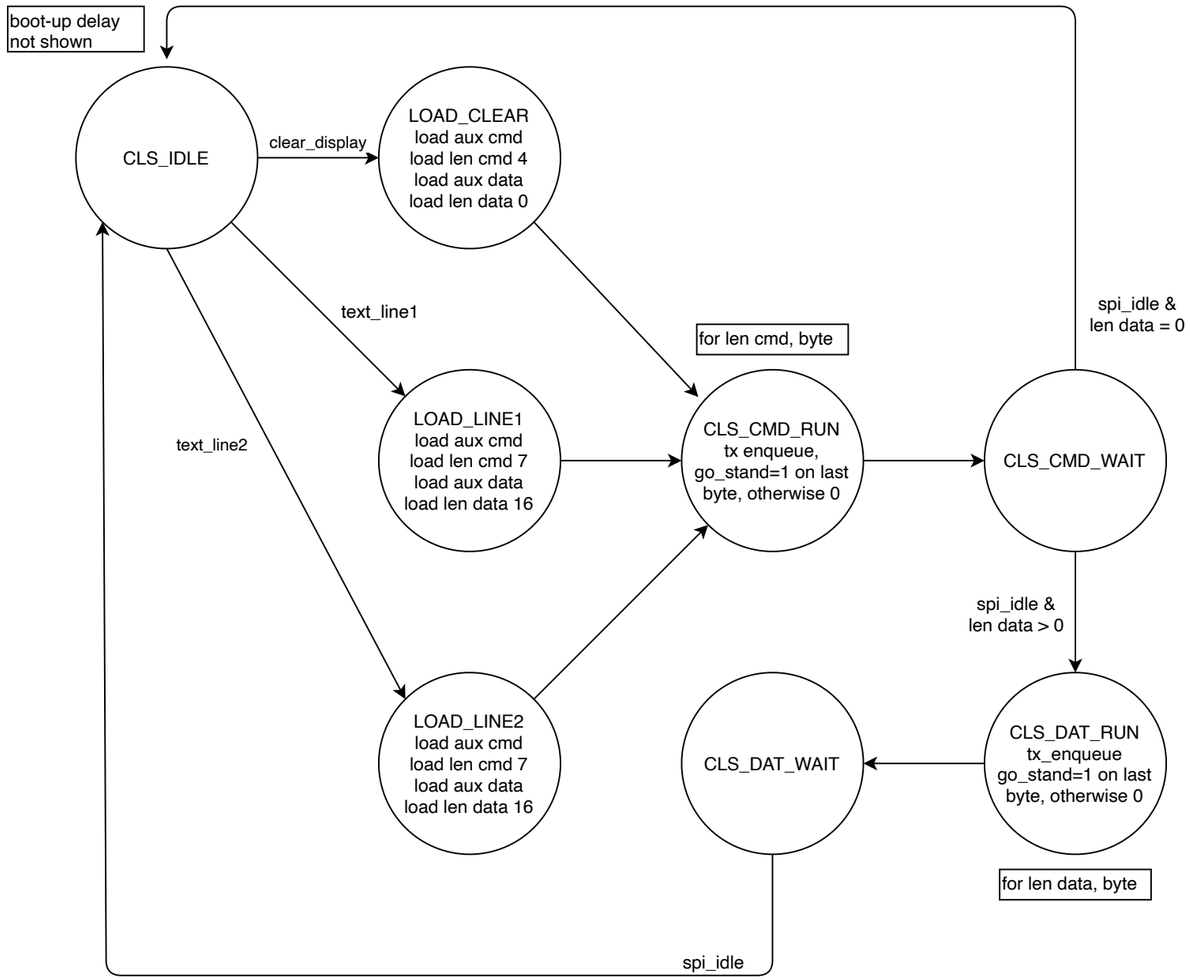
The Pmod CLS custom driver operates by waiting for the ATmega48 microcontroller to boot. It then indicates when a command is ready by outputting '1' on the Command Ready port.

A pulse on Command Write Clear Display instructs the driver to transmit via SPI an ASCII text escape sequence to the ATmega48. The ASCII text escape sequence is a hard-coded text value that instructs that the LCD display be cleared of text on its two lines.

A pulse on Command Write Text Line 1 instructs an FSM to load the Data ASCII Line 1 input and then issue a sequence of sending two ASCII text pieces to the microcontroller. The first text piece instructs that the Pmod CLS be controlled to position its text writing cursor at the start of the LCD's first text line. The second text piece is the 16 character data value that is to be written to line 1 of the LCD.

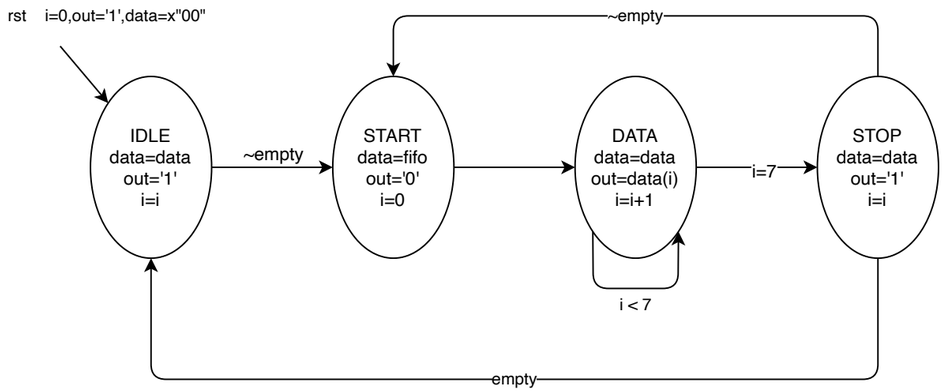
A pulse on Command Write Text Line 2 instructs an FSM to load the Data ASCII Line 2 input and then issue a sequence of sending two ASCII text pieces to the microcontroller. The first text piece instructs that the Pmod CLS be controlled to position its text writing cursor at the start of the LCD's second text line. The second text piece is the 16 character data value that is to be written to line 2 of the LCD.

Other than the Command Write Clear Display which clears the display; there is no soft reset function to reboot the microcontroller and/or reset the display..

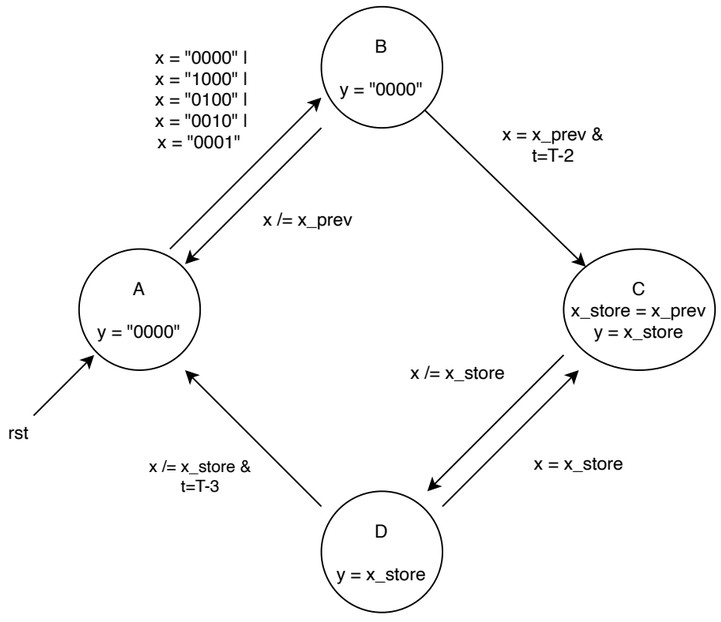


A FSM to operate the Digilent Inc. PMOD CLS LCD display communication via the single slave SPI-machine FSM of this document.

This diagram is
incomplete and does not
show boot-time delay. Also, some
state-bypass preventions
and iterations may not be shown.



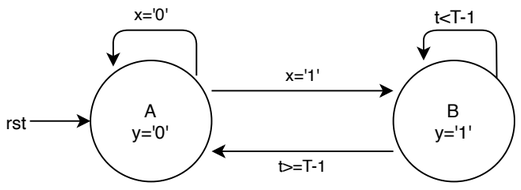
A TX ONLY UART output to UART chip from the FPGA, with the FSM executing at BAUD rate as its clock enable.



Full 4-button combined debouncer.

x is defined as a four-bit value.
x_prev is defined as a four-bit value that holds the previous clock cycle value of x.
x_store is defined as a four-bit value that holds the value of x and updates the
debouncer FSM entered state C during the transition BC..

The registers x_prev and x_store could be combined into one register, with its capture of X
being a clock-enable during transitions and states of a more complex diagram.



Moore FSM for a synchronous pulse stretcher of signal X that lasts for a duration less than T, with Y lasting exactly T cycles.

Textbook Figure 8.28. quoted from:

Finite State Machines in Hardware: Theory and Design (with VHDL and SystemVerilog)
by Volnei A. Pedroni,
reprinted courtesy of The MIT Press