SYSTEM BUS TIMING

EE3232 DIGITAL SYSTEMS III
CLASS NOTES CHAPTER 4

Department of Electrical Engineering
University of New Brunswick

 January 5, 2000

SUMMARY

- Objectives.
- Instruction cycles and bus cycles.
- The 80C188XL system bus interface.
- 80C188XL timing.
- Interfacing switches (review).
- Debouncing.
- Interfacing LED's.
- Decoding Circuits.
- Troubleshooting in the Lab.

OBJECTIVES

- To be able to draw a logic diagram of the interface between a system bus and,
  - Memory devices,
  - IO devices,
  - An 80188 microprocessor.
- To explain the operation of the interfaces above using timing diagrams as instructions are fetched and executed.
- To explain the need for demultiplexing the address-data pins of the 80188.
- To design decoding circuits for,
  - Full decoding,
  - Block decoding,
  - Incomplete decoding,

INSTRUCTION CYCLES AND BUS CYCLES

- The system bus is comprised of:
  1. The 20-bit ***address bus***, a_0 - a_{19},
  2. The 8-bit ***data bus***, d_0 - d_7, and
  3. The ***control bus*** :

/MEMRD	Memory Read
/MEMWR	Memory Write
/IORD	IO Read
/IOWR	IO Write
/RESIN	Reset In
CLKOUT	Clock Out
READY	Wait States
INTi	Interrupt Inputs
/INTAi	Interrupt Acknowledges
NMI	Non-maskable Interrupt

- An instruction is encoded in one or more bytes. The first byte is called the ***op-code***.
- ***Instruction cycle*** : refers to the activity that occurs when an instruction is fetched and executed.
- An instruction cycle results in the execution of one or more bus cycles.
- ***Bus cycle*** : is executed whenever a byte is transferred between memory (or IO) and the CPU.
- Bus cycles may be decomposed into T-states, (T_1, T_2, T_3 ...), each synchronized to the CPU clock.

THE 80C188XL SYSTEM BUS INTERFACE

- Refer to the diagram on the following page.

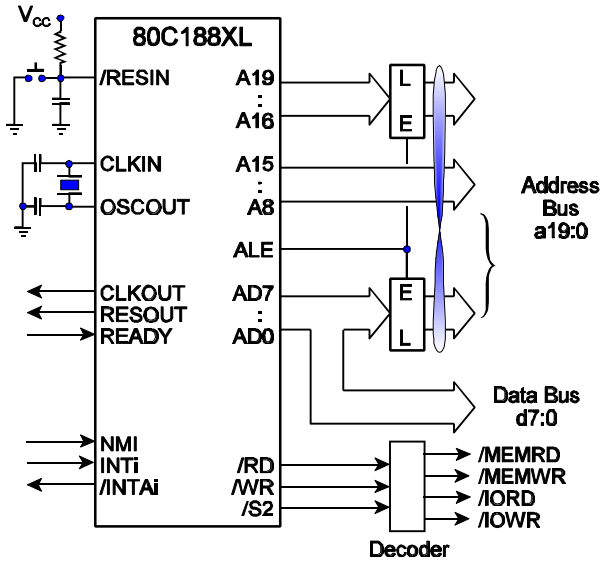
Address/Data Bus Interface

- Addresses and data are (time) ***multiplexed*** on the AD_0 - AD_7 pins.
- Addresses and status are (time) ***multiplexed*** on the A_{16} - A_{19} pins.
- AD_0 - AD_7 contains 8-bit data bus during T_2, T_3.

- AD_0 - AD_7 , A_8 - A_{15} , A_{16} - A_{19} contain a 20-bit address during T_1 .

THE 80C188XL SYSTEM BUS INTERFACE (CONT'D)

- ***External Latches*** enabled by the ALE, ***demultiplex*** the multiplexed address-data-status at AD_0 - AD_7 and A_{16} - A_{19}.

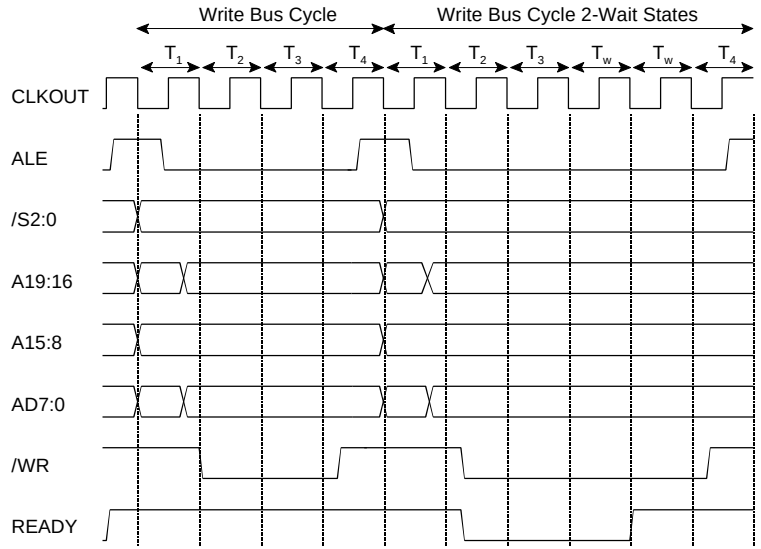


Control Bus Interface

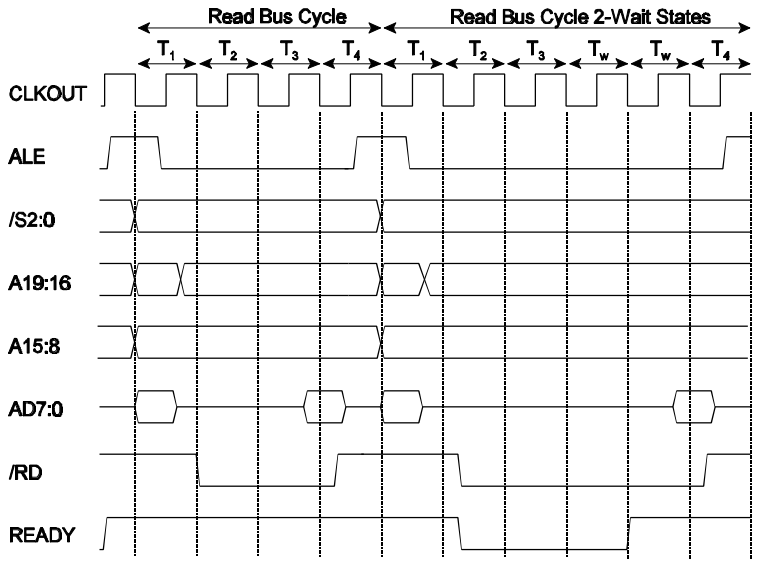
- /RD is activated during a read bus cycle,
- /WR is activated during a write bus cycle.
- /S2 = '0' during IO (read or write) bus cycles,
- /S2 = '1' during memory bus cycles.
- /RESIN resets IP = 0000H, CS = FFFFH.
- A crystal is connected between CLKIN and OSCOUT for setting the CLKOUT frequency, f = $f_{crystal}$ /2.
- INTi, /INTAi, NMI are used to interface external interrupts.
- READY is an input used to lengthen the /RD and /WR pulses (by inserting ***wait states***) to permit interfaces with slow memory and IO devices.

80C188XL TIMING

- Each 80C188XL bus cycle consists of 4 or more T-states.
- The AD_0 - AD_7 and A_{16} - A_{19} pins contain address information during T_1 . AD_0 - AD_7 contains data during T_3 .
- ALE is activated during T_1 and is used to trigger external address latches during T_1 so that only address information appears on the address bus during T_2 , T_3 and T_4 .
- Deactivating READY during T_2 (or T_w) causes an additional clock cycle (identical to T_3) called a ***wait state***, T_w, to be inserted between T_3 and T_4 .
- Wait states, T_w, lengthen the duration of the /RD or /WR pulse.
- This allows read or write bus cycles with slow memory and IO devices.



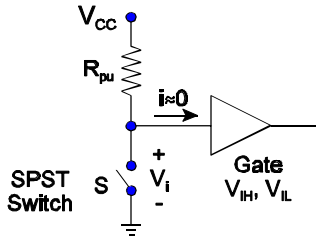
80C188 Write Cycle Timing



80C188 Read Cycle Timing

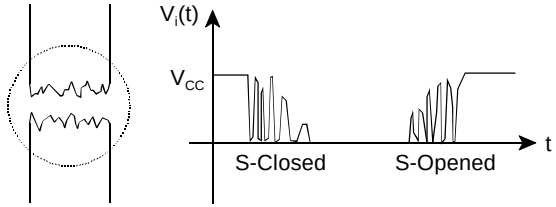
INTERFACING SWITCHES (REVIEW)

- Need a circuit to transform position of a SPST switch into a logic output.
- Use a single pull-up resistor, R_{pu}, as shown.
- R_{pu} is selected such that a voltage that represents a logic *HIGH* appears at V_i when the switch is open.
- When closed, V_i = 0V ➜ logic *LO*.



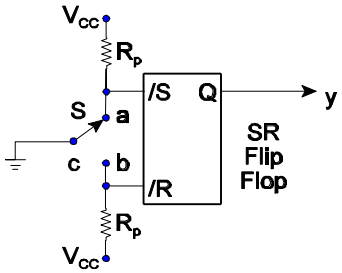
DEBOUNCING (REVIEW)

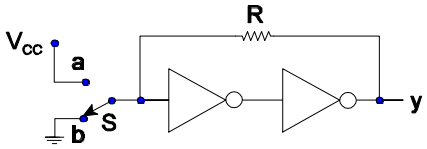
- When a mechanical switch opens or closes, the contacts bounce for 10's of milliseconds.
- Bouncing is caused by surface irregularities and the mechanical spring constant of the switch.
- Mechanical bouncing can wreak havoc in a fast digital circuit by (unknowingly) causing an input signal to change logical state many times with a single throw of the switch.



- One way to eliminate mechanical bouncing is to use hardware debouncer circuits such as those below.

- Sketch the waveforms, b(t) and y(t), as the switch, S, changes position.

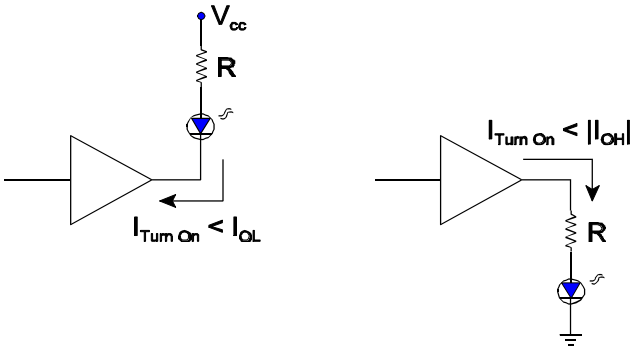




INTERFACING LED'S (REVIEW)

- Device specs include the maximum currents that a gate can sink when driving an output at logic HI and at logic LO.
- If the gate drives an LED then the gate must be able to supply the minimum turn-on current for the LED.
- Two forms of interface are possible,

INTERFACING LED'S (CONT'D)

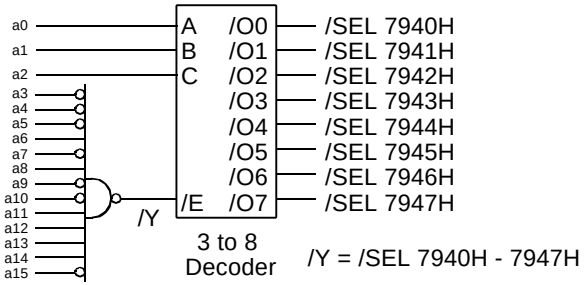


DECODING CIRCUITS

- Decoding circuits are used to select IO or memory devices by decoding address and control bus signals.
- Decoding circuits may be classified as :
  - Full decoding,
  - Block decoding,
  - Incomplete decoding.

FULL DECODING FOR MEMORY DEVICES

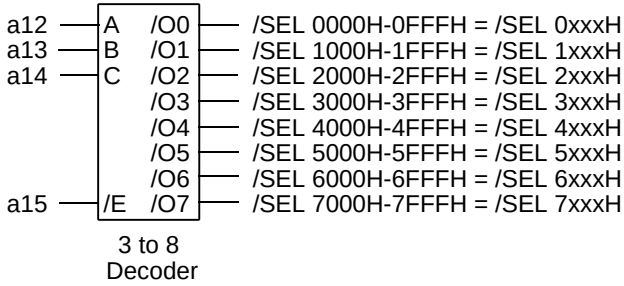
- All address lines on the system bus are decoded.
- Used as enables on port devices.



/Y = /SEL 7940H - 7947H

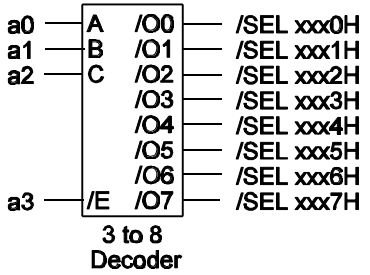
BLOCK DECODING FOR MEMORY DEVICES

- Partial decoding of the upper address lines of the system bus.
- The upper address line are decoded.
- The un-decoded lower address lines are don't cares.
- Commonly used to activate enables on memory devices.



INCOMPLETE DECODING FOR MEMORY DEVICES

- Partial decoding of lower address lines of the system bus.
- The lower address line are decoded.
- The un-decoded higher address lines are don't cares.
- Commonly used to activate the enables on IO devices.



TROUBLESHOOTING IN THE LAB

1. Be systematic.

2. Ensure that power to all components is turned on.

3. If the CPU board is dead then go to Step 7.

4. If the CPU board is ***live*** then verify that all peripheral or IO boards that your program uses are functional.
   - Use port commands to exercise the peripherals.
   - Write FFH / 00H to the OP port.
   - Measure and verify the output.

5. If the peripheral is not functional then go to Step 7.

6. If the hardware is functional but the program is not then debug the logical errors in your program:
   - Program is gracefully terminated?
   - A procedure is not terminated by RET or RET n?
   - An interrupt handler is not terminated by IRET?
   - Equal number of PUSH and POP's in all procedures?

7. Check that the correct power supply voltages are getting to the power pins of each component.

8. Do a visual and tactile check of the board:
   - Are wire clippings, pencil lead, solder ... shorting IC pins or circuit board traces?
   - Are all jumper blocks in the correct locations?
   - Touch all of the IC's to see if any are excessively hot.
   - Are any IC's inserted backwards in their sockets?
   - Are any IC pins bent or misaligned in their sockets?

9. If the problem still persists, perform a ***signal roll call*** of the main CPU signals with a scope :
   - The CPU clock.
   - The address and data lines for bus activity.
   - ALE and the /RD and /WR control signals.
   - Are enables being generated for the EPROM's?
   - If there is no activity on the address or data bus then check,

     READY input (it should be active hi)
     /RESIN input (it should be inactive hi)
     INTR (it should be inactive low)
     HOLD (it should be inactive low).