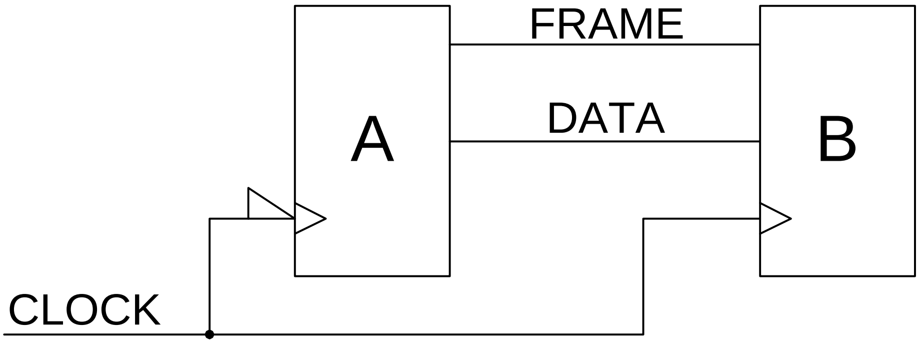
Lecture 2

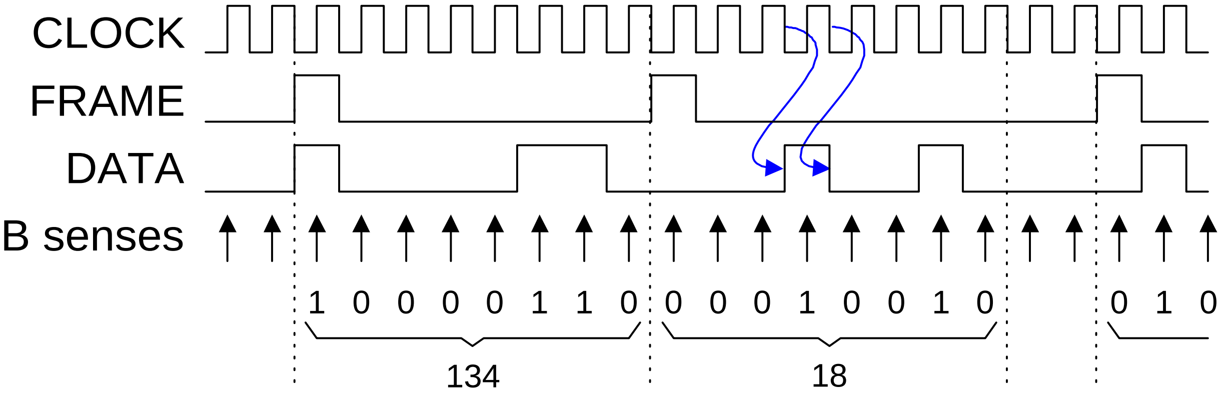
Synchronous Bit-Serial Interfacing

Objectives

- Explain how data is sent between two digital systems using a synchronous bit-serial protocol
    - Synchronous: same clock at transmitter & receiver
    - Bit-serial: Only one bit sent at a time
    - Protocol: The procedure for exchanging information
- Explain the meaning of setup and hold times
- Investigate the timing constraints in a transmission system

Synchronous Bit-Serial Transmission





Transmitting 8 bit values from A to B:

– FRAME indicates the first bit of each value; the other 7 bits follow on consecutive clock cycles. The FRAME signal is often called a *frame sync* pulse.

– DATA changes on the *falling* CLOCK edge

– Propagation delays are often omitted from diagram.

– DATA is sensed by system B on the *rising* CLOCK edge to maximise tolerance to timing errors. We must always clock a flipflop at a time when its DATA input is not changing.

Transmission Delays

Propagation speed = $(L_0C_0)^{-½}$ where L_0 and C_0 are inductance and capacitance per unit length.

For a uniform line this gives a total delay of $(LC)^{½}$ where L and C are the total inductance and capacitance. Any additional load capacitance will increase delay.

Signal speed can be expressed in terms of:

- the speed of light (c = 30 cm/ns)
- the geometry of the wiring
- the relative permittivity of the insulator:

Examples:

- Coax cable:

  $c \times \varepsilon_r^{-½} \Rightarrow$ 20 cm/ns for ε_r =2.3 (teflon)

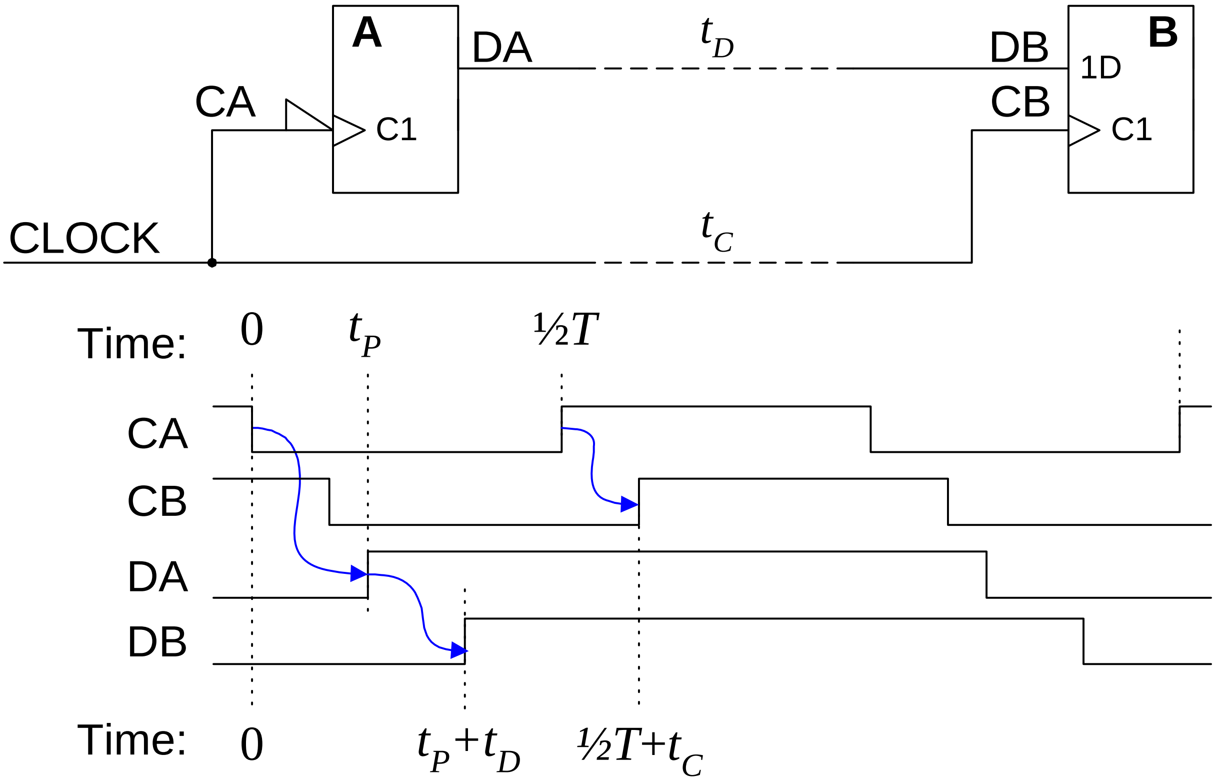
- PCB with ground plane:

  $1.4c \times (1.4+\varepsilon_r)^{-½}$ cm/ns $\Rightarrow$ 17 cm/ns for ε_r =5 (fibreglass)

Rule-of-thumb:

Data travels along typical wires and circuit board tracks at about 15 cm/ns: half the speed of light.

Timing Specifications



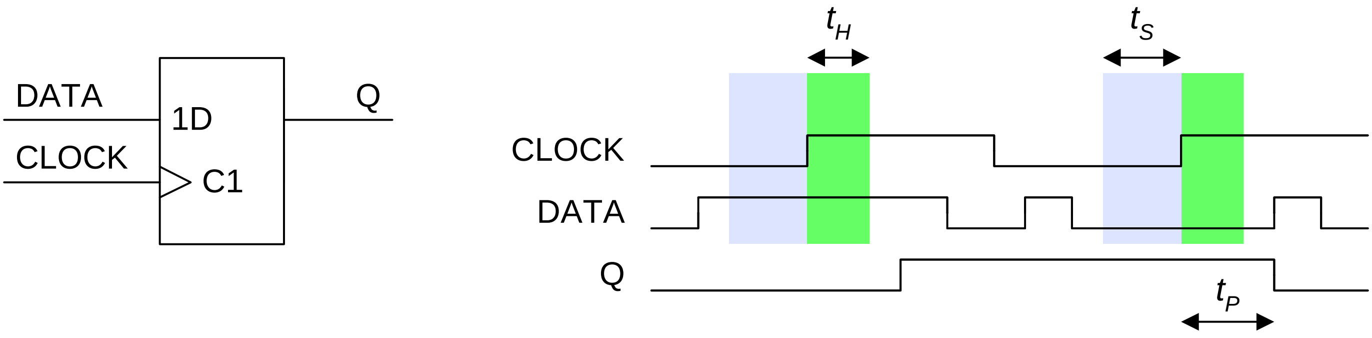
t_P	*Propagation* delay for device A.
T	Clock Period.
t_C, t_D	Transmission line delays for CLOCK and DATA

For Device B:

- Data input changes at time t_P+t_D
- Clock input changes ↑ at time ½$T+t_C$

Setup and Hold Times

The DATA input to a flipflop or register must not change at the same time as the CLOCK.



Setup Time: DATA must reach its new value at least t_S before the CLOCK↑ edge.

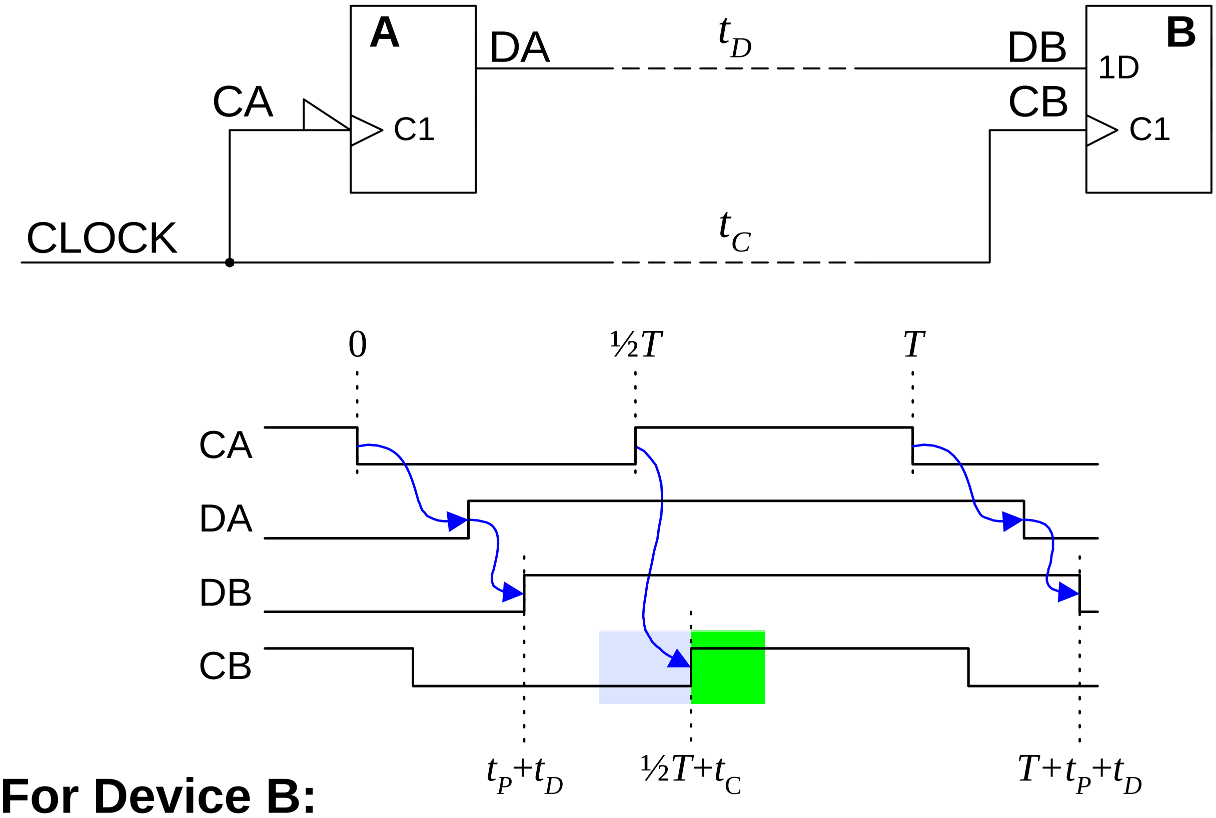
Hold Time: DATA must be held constant for at least t_H after the CLOCK↑ edge.

Typical values for a register: $t_S = 5$ ns, $t_H = 3$ ns

The setup and hold time define a window around each CLOCK ↑ edge within which the DATA **must not change**.

If these requirements are not met, the Q output may oscillate for many nanoseconds before settling to a stable value.

Timing Constraints



For Device B:

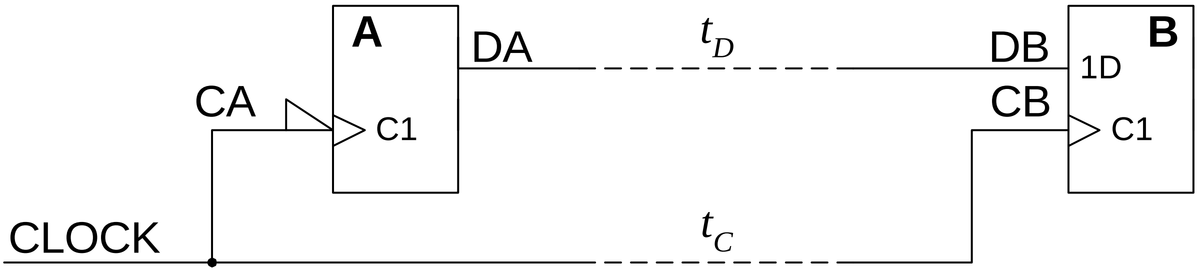
- Data input (DB) changes at t_P+t_D (and $T+ t_P+t_D$)
- Clock↑ (CB) at time $½T+t_C$

For reliable operation:

- Setup Requirement: $t_P + t_D + t_S < ½T + t_C$
- Hold Requirement: $½T+t_C + t_H < T + t_P + t_D$

Get a pair of inequalities for each flipflop/register in a circuit.
You never get both t_S and t_H in the same inequality.

Example Values



For Motorola 56001 27MHz DSP processor:

$0 < t_P < 50$ ns, $t_S = 12$ ns, $t_H = 27$ ns

Suppose differential delay: $-10 < (t_D - t_C) < +10$

Find maximum CLOCK frequency (min CLOCK period):

- $\max(t_P + t_D) + t_S < \min(\frac{1}{2}T + t_C)$

  $50 + 10 + 12 < \frac{1}{2}T + 0$ $(t_D = 10, t_C = 0)$

  $\frac{1}{2}T > 12 + 50 + (+10) = 72 \Rightarrow T > 144$ ns

- $\max(\frac{1}{2}T + t_C) + t_H < \min(T + t_P + t_D)$

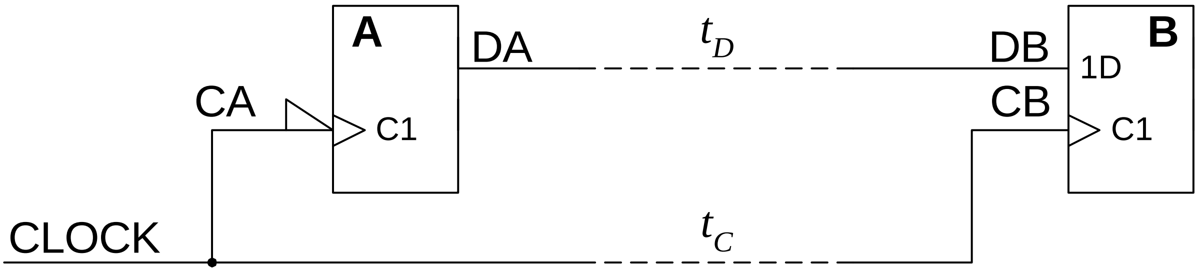
  $\frac{1}{2}T + 10 + 27 < T + 0 + 0$ $(t_D = 0, t_C = 10)$

  $\frac{1}{2}T > 27 + 10 = 37 \Rightarrow T > 74$ ns

Hence $f_{\text{CLOCK}} < 1/144 = 7$ MHz

To test for worst case: make the left side of the inequality as big as possible and the right side as small as possible.

Propagation Delay Constraint Inequalities



When do they arise

- Whenever a flipflop’s clock and data input signals originate from the same ultimate source. Here CB and DB both originate from CLOCK. You normally get two inequalities for each flipflop in a circuit.

Relationship beween setup and hold inequalities:

- Setup Requirement: $t_P + t_D + t_S < ½T + t_C$
- Hold Requirement: $½T + t_C + t_H < t_P + t_D + T$
- To get the Hold inequality you change t_S to t_H , swap the sides of the other terms and add T onto the right side.

Are both t_S and t_H ever in the same inequality?

- No.

How do you decide to take the max or the min?

- For a <, take max of everything on the left and min of everything on the right.
- max = most positive: for example, max(–7,–2) = –2

Quiz Questions

1. What is a *bit-serial* transmission system?
2. What is a *synchronous* transmission system?
3. In a synchronous transmission system in which the transmitted data changes on the rising edge of the CLOCK, why is it normal for the receiver to sense the data on the falling edge of the CLOCK ?
4. What is the purpose of the *frame sync* signal In a synchronous bit-serial transmission system?
5. How far does a signal travel along a typical wire in one nanosecond?
6. What do the terms *setup time* and *hold time* mean?
7. Why do you get a pair of timing inequalities for each flipflop or register in a circuit?
8. In formulating the timing inequalities, how do you choose what to use for a quantity whose value may lie anywhere within a particular range?

Answers are all in the notes.

Lecture 3

Asynchronous Bit-Serial Interfacing

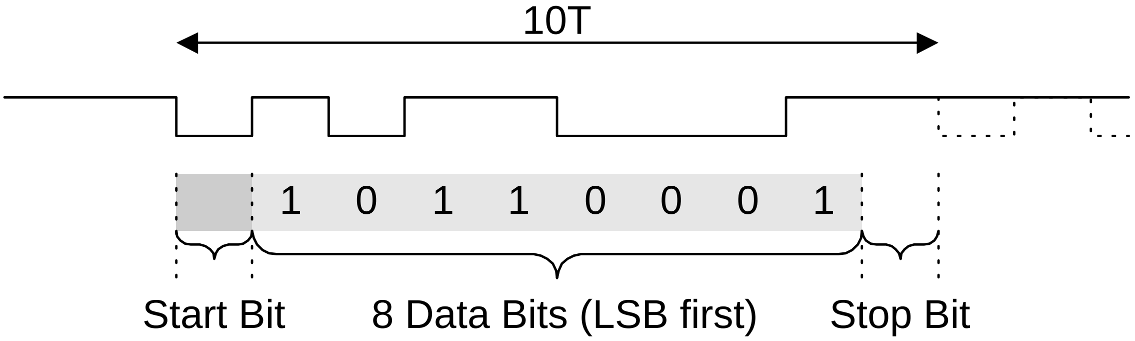
Objectives

- Explain how data is sent between two digital systems using an asynchronous bit-serial protocol
- Explain why it is necessary to include START and STOP bits in an asynchronous protocol.
- Explain the circuitry needed for an asynchronous bit-serial receiver
- Derive the tolerances for the transmitter and receiver clocks in an asynchronous bit-serial system

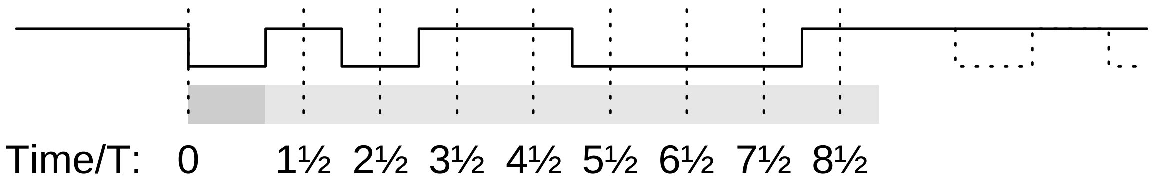
Asynchronous Bit-Serial Transmission

Combine timing and data into a single signal to circumvent differential delays over long distances (saves wires too).

RS232: Serial Port on a PC

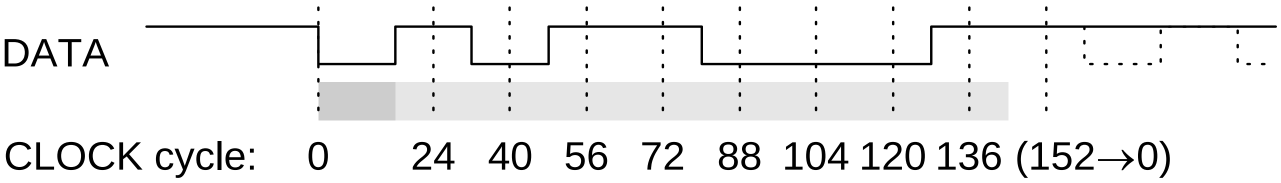


- Idle state has signal = 1. START bit indicates new byte.
- Data transmitted LSB first (above example equals 141_{10})
- Bit cell duration is T: e.g. T=52 µs, $1/T$ = 19200 baud
- STOP bit needed to ensure signal goes to 1 before the next START bit which might follow immediately.
- Signal is decoded by sampling each bit in the centre of its cell an appropriate time after the start bit:

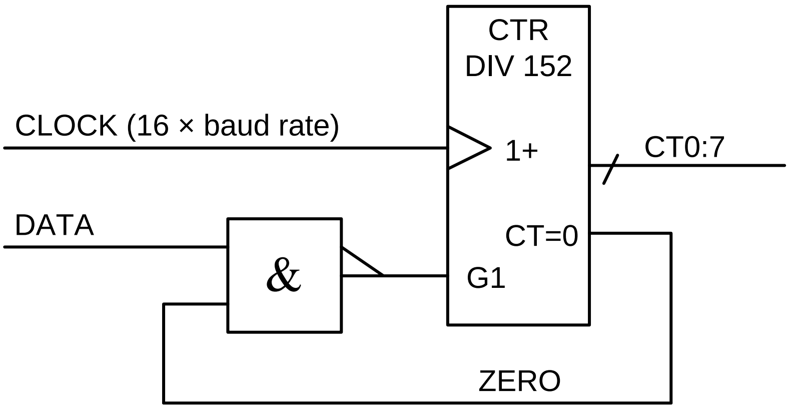


RS232 Receiver Timing

Good time resolution ⇒ use a master clock period of T/16:

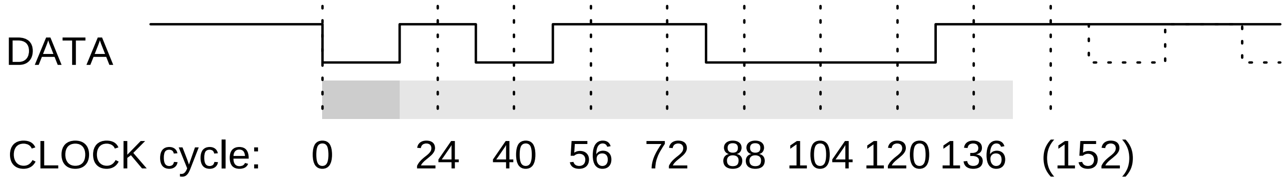


- Middle of STOP bit is after 9½ bitcells ⇒ 9½ × 16 = 152 master clock cycles.
- Use ÷152 counter but hold it at 0 until the START bit arrives ⇒ inhibit counting whenever CT=0 and DATA=1.
- Counter will increment either if DATA=0 or if CT is already non-zero.
- We use a *clock enable* input, G1, to control whether or not the counter increments; much better design than using gates to modify the clock signal.

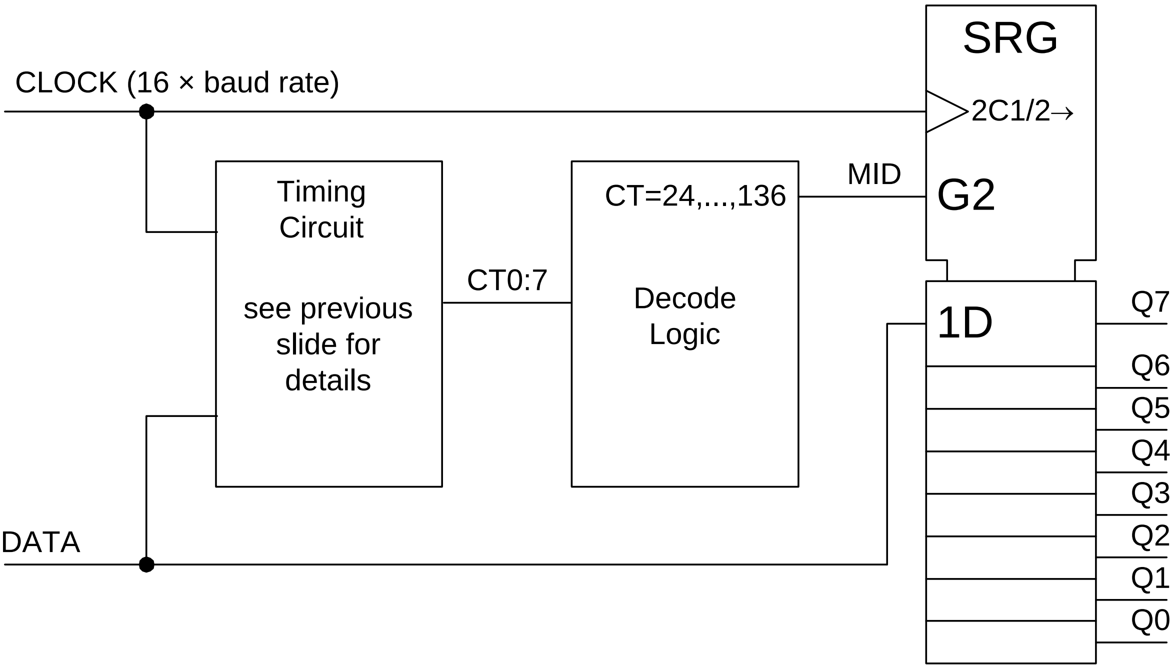


The CT=0 output from counter goes high when the contents of the counter, CT, are zero. Generate this signal using a NOR gate connected to all 8 counter outputs.

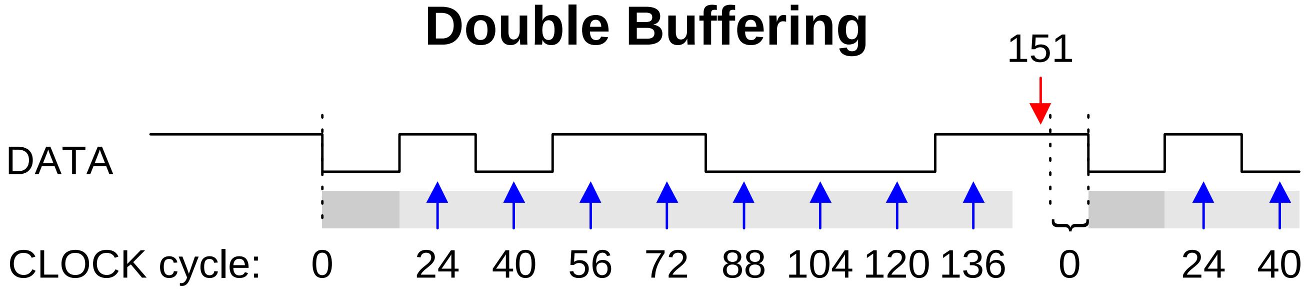
RS232 Receiver



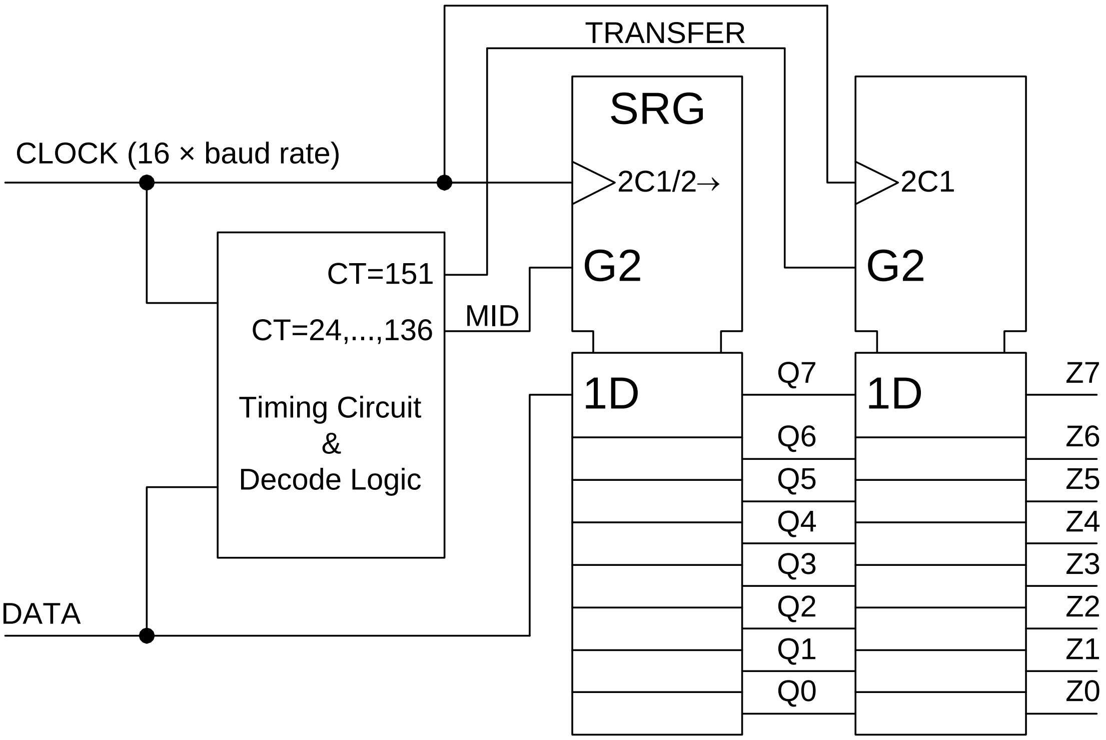
- Use an 8-bit shift register to store the data value. Only allow it to shift when CT = 24, 40, … , 136.
- The decode logic output, MID, goes high when the counter has one of these values (all odd multiples of 8 ⇒ four LSBs = 1000_2).
- Notation:
  - The shift register clock has two functions separated by a /: 2C1 clocks first bit, 2→ shifts the rest.
  - Both these functions are controlled by the *clock enable* input, G2.



Double Buffering

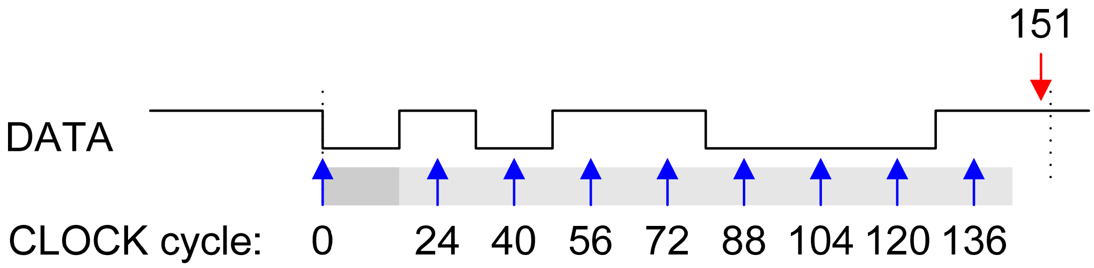


- The 8 data bits only stay in the shift register for $3T$ before they get shifted out again by the next data byte.
- Host microprocessor must respond to an interrupt within this time and retrieve the data.
- Use a second register to grab the data at T=151 and keep it for a whole $10T$. This gives the µP more time.



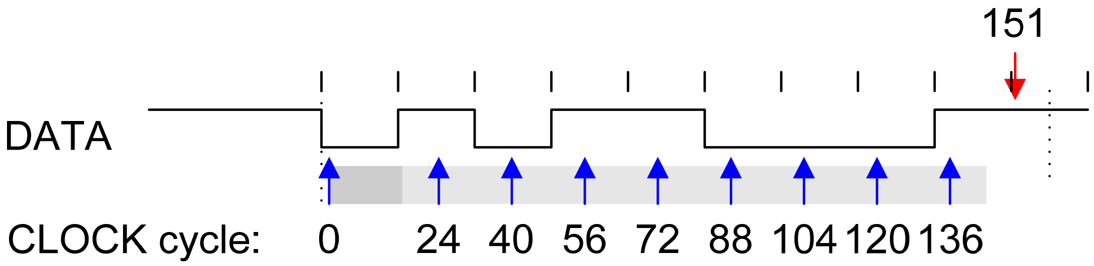
Timing Errors

- **Ideal situation:**
  - Receiver clock period P = T/16
  - Counter starts counting exactly on DATA falling edge

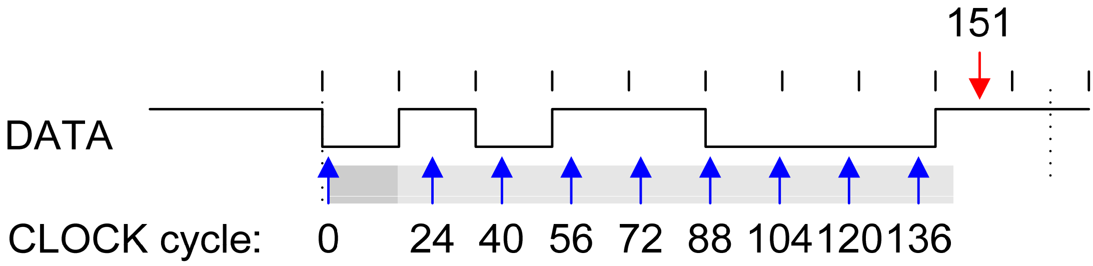


- **Real situation:**
  - Receiver clock period not exactly T/16
  - Counter starts with some delay
    - On first rising edge of P after DATA goes low

P slightly too small

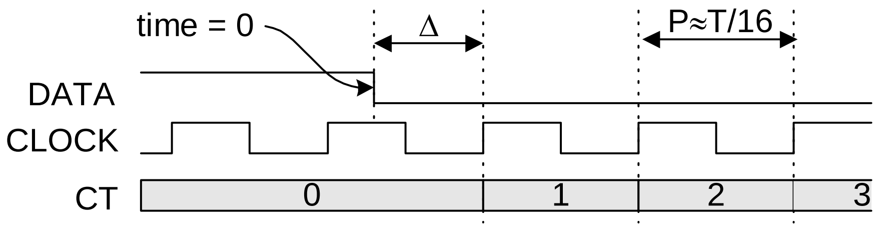


P much too small



RS232 Receiver Timing

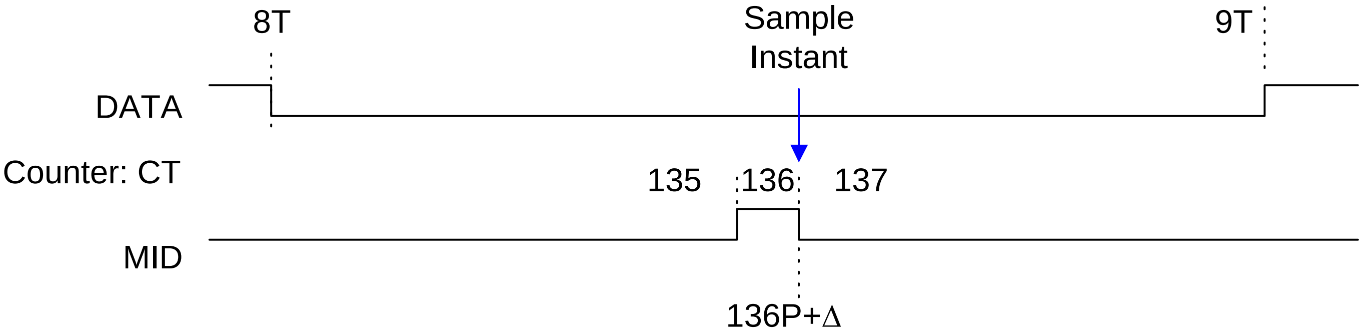
CT will change to 1 on the first CLOCK ↑ edge after DATA goes to 0:



Neglecting logic propagation delays, $0 < \Delta < P$ where P is receiver clock period.

- Count 0→1 a time Δ after the START bit
- Count $n \rightarrow n+1$ a time $nP+\Delta$ after the START bit

Timing in the last (MSB) bit cell:



We will sample the correct bit cell if: $8T < 136P + \Delta < 9T$

$$8T < 136P+0 \quad \Rightarrow \quad T/P < 17$$

$$136P+P < 9T \quad \Rightarrow \quad T/P > 15.2$$

Hence $T/P = 16 + 6.3\% - 5.0\%$ which implies a clock accuracy of around ± 2.5% at transmitter and receiver.

Quiz Questions

1. How can you be sure that in the RS232 protocol there will always be a high-to-low transition at the beginning of each transmitted byte ?
2. What is the function of the *clock enable* input on a counter or register?
3. What logic gate is needed to detect when the contents of a counter is equal to zero?
4. If an asynchronous protocol has one START bit, eight data bits and one STOP bit, how may bitcell periods is it from the beginning of the START bit until the centre of the STOP bit?
5. What is the function of an input pin that is labelled 2C1/2→?
6. If the CLOCK input of a counter has period P, what is the range of possible delays between the counter's enable pin going high and the counter incrementing?
7. What is the purpose of double buffering the data in an asynchronous bit-serial receiver?
8. How can you tell if a binary number is an odd multiple of 16?

Answers are all in the notes.

Lecture 4

Microprocessor to Memory Interface

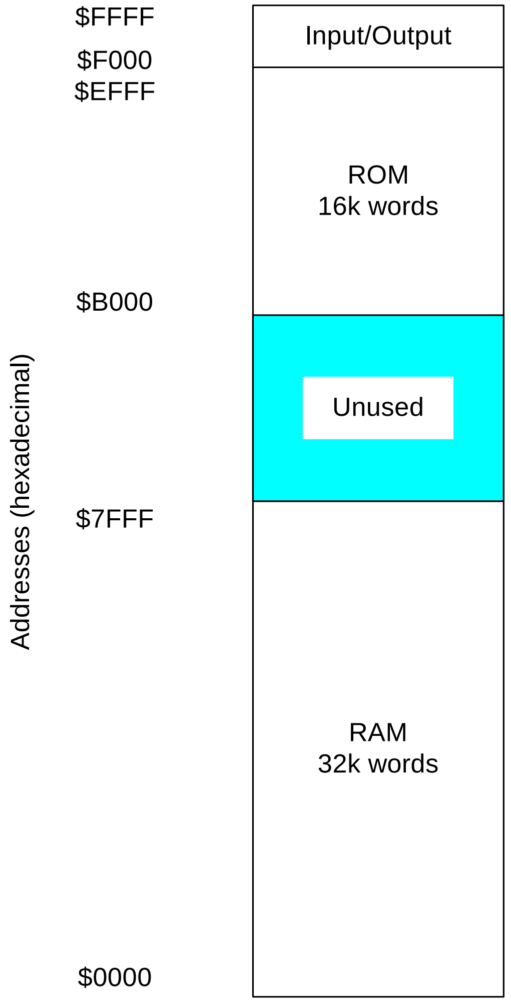
Objectives

- Explain how memory is connected to a microprocessor
- Describe the sequence of events in reading from and writing to a static RAM
- Describe the structure and input/output signals of a static RAM

Microprocessor Memory Map

A typical 8-bit microprocessor has

- A 16-bit address bus, A15:0
  - Can have up to 2^{16}=65536 memory locations
  - Value is usually written in hexadecimal often with $ prefix:
    - e.g. $1000 = 2^{12} = 4k = 4096
- An 8-bit data bus, D7:0
  - Each data word in memory has 2^8 = 256 possible values

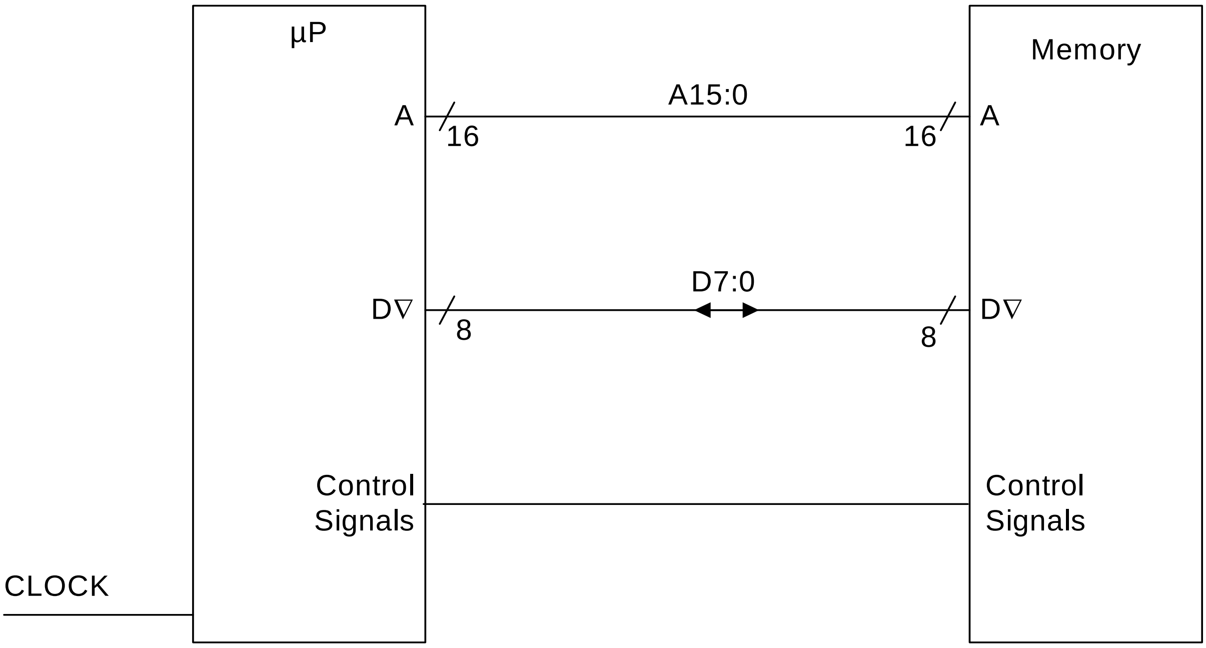


We can tell which region of memory an address is in by inspecting the top few bits:

	A15:12	
F:	1111	Input/Output
E:	1110	ROM
D:	1101	ROM
C:	1100	ROM
B:	1011	ROM
A:	1010	
9:	1001	
8:	1000	
7:	0111	RAM
6:	0110	RAM
5:	0101	RAM
4:	0100	RAM
3:	0011	RAM
2:	0010	RAM
1:	0001	RAM
0:	0000	RAM

INOUT = A15·A14 ·A13 ·A12
ROM = A15·A14 ·!(A13 ·A12) + A15·!A14·A13·A12
RAM = !A15

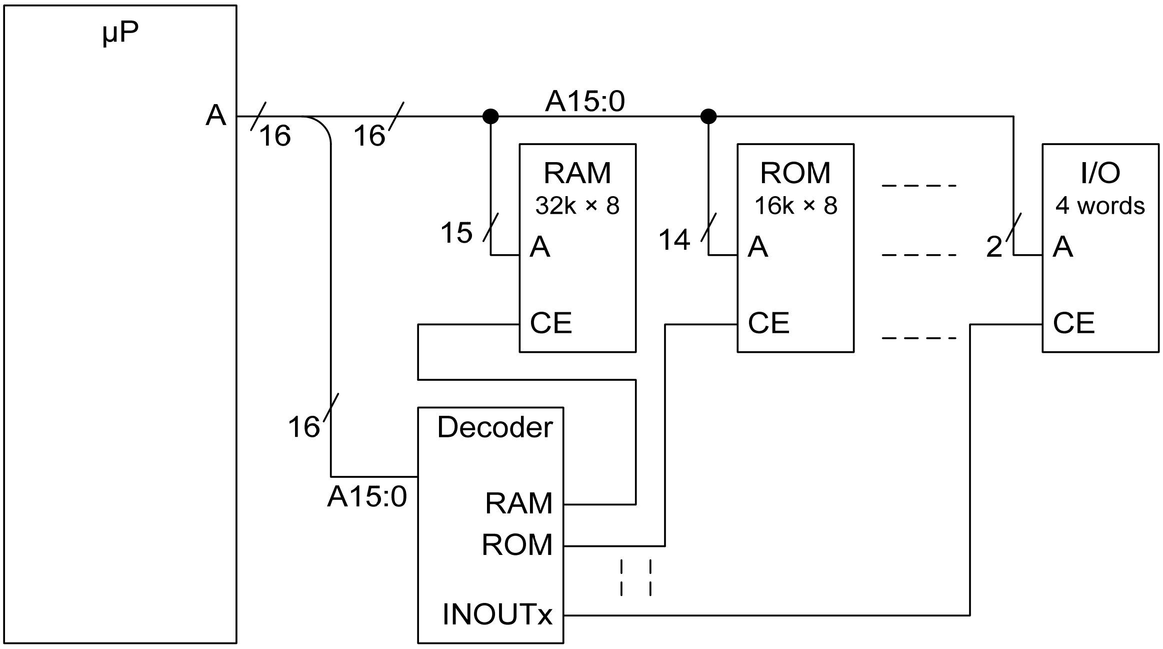
Microprocessor ↔ Memory Interface



During each memory cycle:

- A15:0 selects one of 2^{16} possible memory locations
- D7:0 transfer one word (8 bits) of information either to the memory (write) or to the microprocessor (read).
- D7:0 connections to the microprocessor are tri-state (∇): they can be:
  - “logic 0”, “logic 1” or “high impedance” (inputs)
- The control signals tell the memory what to do and when to do it.

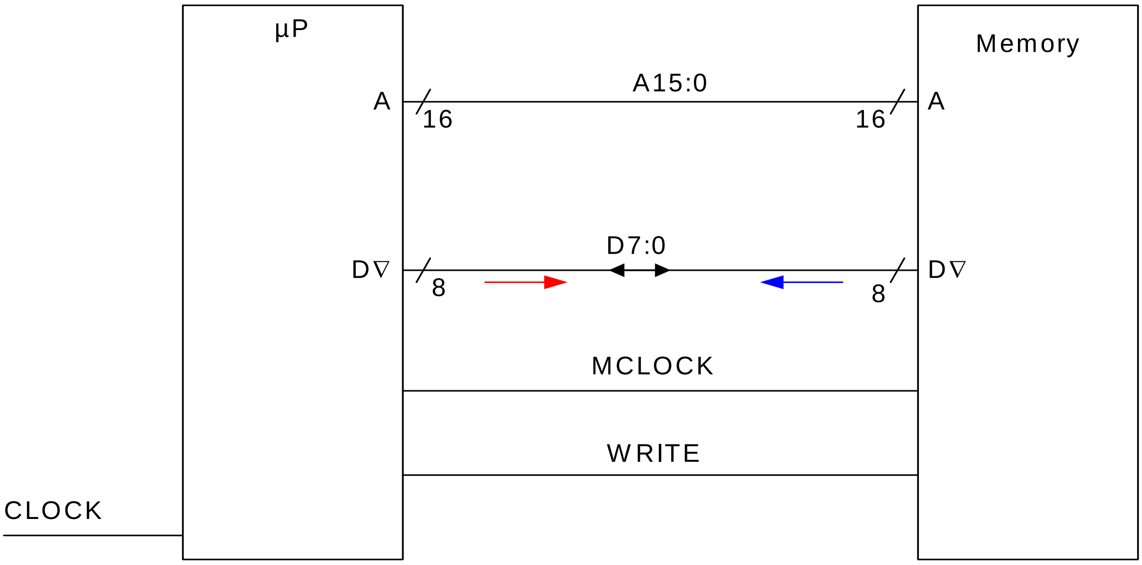
Memory Chip Selection



- Each memory circuit has a “chip enable” input (CE)
- The “Decoder” uses the top few address bits to decide which memory circuit should be enabled. Each one is enabled only for the correct address range:

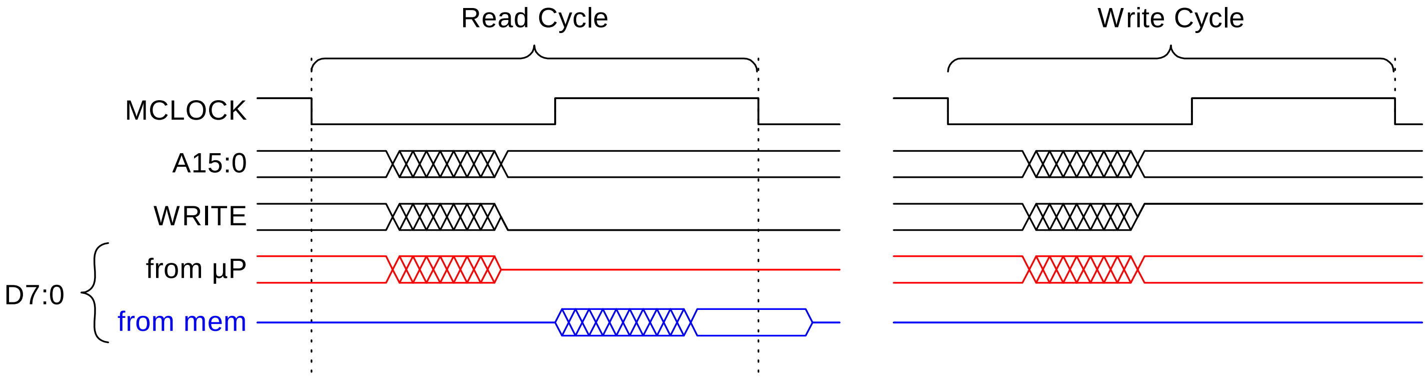
  RAM = !A15
  ROM = A15·(A14 ·!(A13 ·A12) +!A14·A13·A12)
  INOUTx = A15·A14·A13·A12·!A11·A10·!A9·A8·
  !A7·A6·A5·A4·!A3·A2
- INOUTx responds to addresses: $F574 to $F577
  other I/O circuits will have different addresses
- Low n address bits select one of 2^n locations within each memory circuit (value of n depends on memory size)

Memory Interface Control Signals



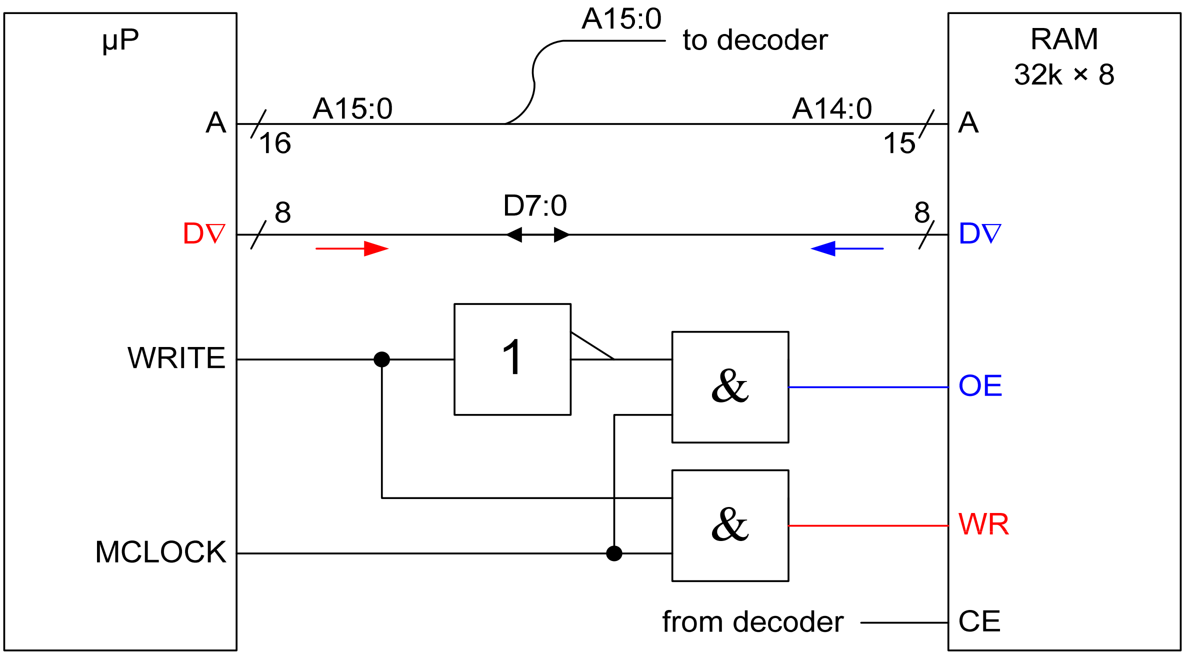
Control signals vary between microprocessors but all have:

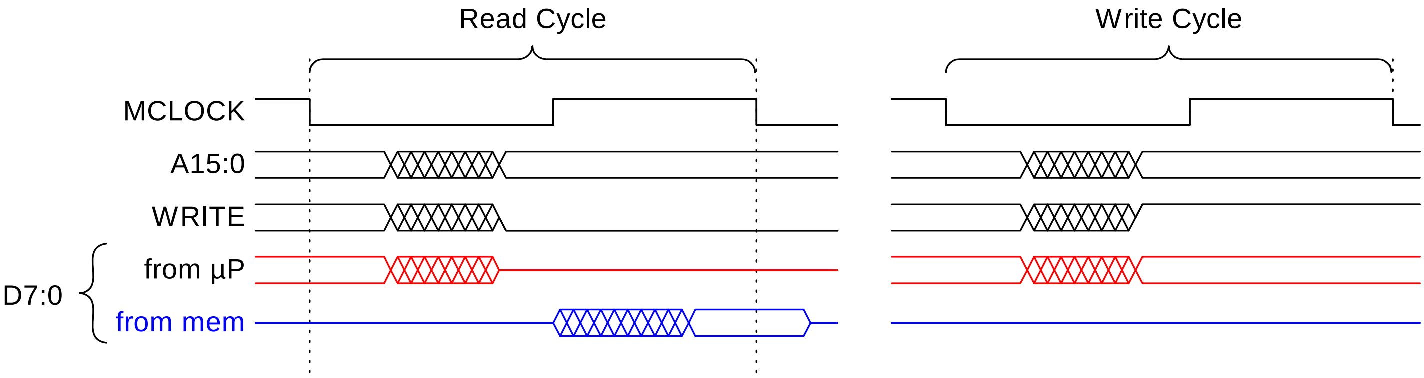
- A clock signal to control the timing (can be the same as the system CLOCK)
- A signal to say whether the microprocessor wants to read from memory or to write to memory
  - Must make sure that D7:0 is only driven at one end



D7:0 from memory only allowed when MCLOCK·!WRITE true

Memory Circuit Control Signals





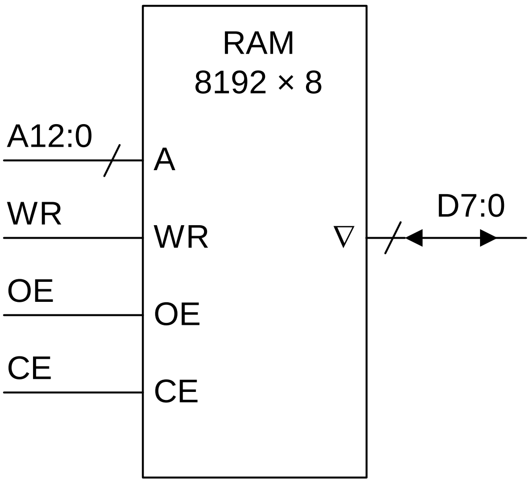
- **Output enable**: OE = MCLOCK·!WRITE turns on the D7:0 output from the memory
- **Write enable**: WE = MCLOCK·WRITE writes new information into the selected memory location with data coming frommicroprocessor
- **Chip enable**: comes from the decoder and makes sure the memory only responds to the correct addresses

RAM: Read/Write Memory

Static RAM: Data stored in bistable latches

Dynamic RAM: Data stored in charged capacitors:
retained for only 2ms.
Less circuitry ⇒ denser ⇒ cheaper.

8k × 8 Static RAM



CE	OE	WR	D0:7	Action
0	?	?	Hi Z	Disabled
1	0	0	Hi Z	Idle
1	1	0	Out	Read
1	?	1	In	Write

∇ Tri-state output: Low, High or Off (High Impedance).
Allows outputs from several chips to be connected;
Designer <u>must</u> ensure only one is enabled at a time.

CE Chip Enable: disabling chip cuts power by 80%.

OE Output Enable: Turns the tri-state outputs on/off.

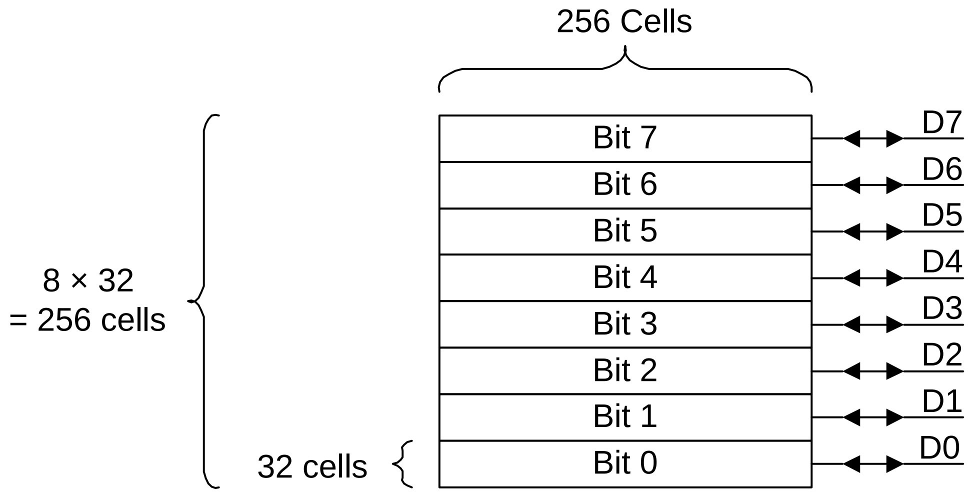
A12:0 Address: selects one of the 2^{13} 8-bit locations.

WR Write: stores new data in selected location

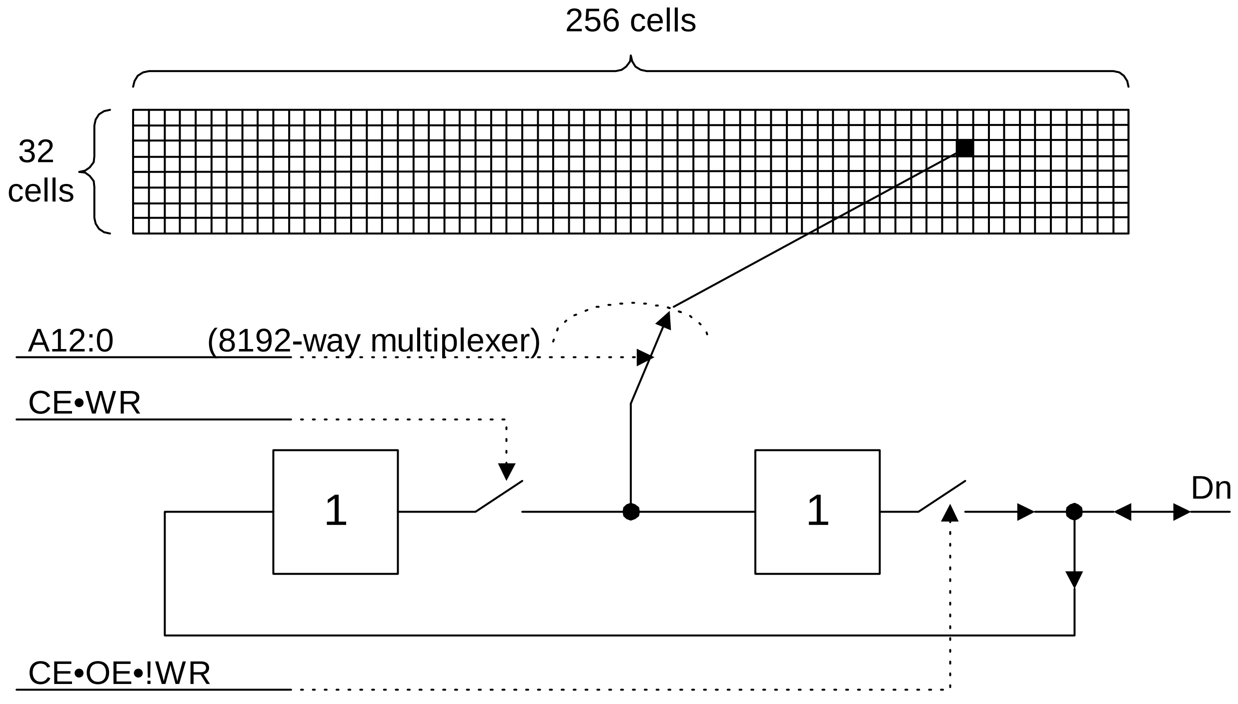
D7:0 Data in for write cycles or out for read cycles.

8k × 8 Static RAM

The 64k memory cells are arranged in a square array:



For each output bit, an 8192-way multiplexer selects one of the cells. The control signals, **OE**, **CE** and **WR** determine how it connects to the output pin via buffers:



Occasionally DIN and DOUT are separate but ⇒ more pins

Quiz Questions

1. What is the *memory map* of a microprocessor system
2. Why do all microprocessor systems include some read-only memory (ROM)
3. What does it mean if a digital device has a *tri-state* output? When are such outputs necessary ?
4. What is the difference between the *chip enable* and the *output enable* inputs of a static RAM?
5. If a static RAM has *n* address inputs and *m* data outputs, how many bits of information does it store?
6. What is the binary value of the three most significant address bits for the hexadecimal address $BC37 ?

Answers are all in the notes.

Lecture 5

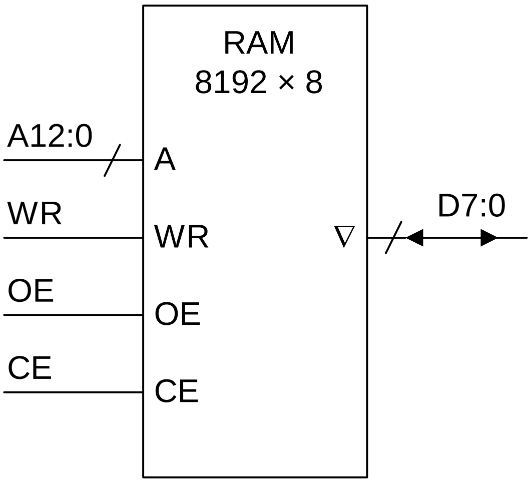
Microprocessor to Memory Interface

Objectives

- Investigate the timing constraints for a microprocessor when reading from or writing to memory.

RAM: Read/Write Memory

8k × 8 Static RAM



CE	OE	WR	D0:7	Action
0	?	?	Hi Z	Disabled
1	0	0	Hi Z	Idle
1	1	0	Out	Read
1	?	1	In	Write

∇ Tri-state output: Low, High or Off (High Impedance). Allows outputs from several chips to be connected; Designer must ensure only one is enabled at a time.

A12:0 Address: selects one of the 2^{13} 8-bit locations.

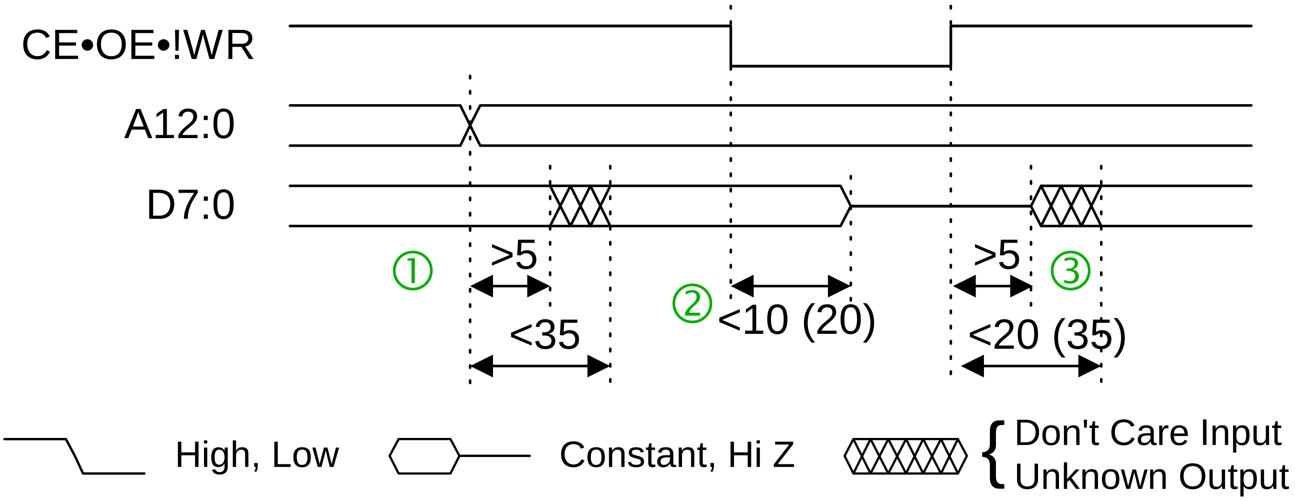
D7:0 Data in for write cycles or out for read cycles.

CE **Chip Enable**: disabling chip cuts power by 80%.

OE **Output Enable**: Turns the tri-state outputs on/off.

WR **Write**: stores new data in selected location

Memory Read Cycle



Note: Time axis not to scale

A *read cycle* happens when CE•OE•!WR is true.

① If A12:0 changes, D7:0 remains for at least 5 ns and goes to new value within 35 ns. Rubbish in between <u>even if new and old locations contain the same value</u>.

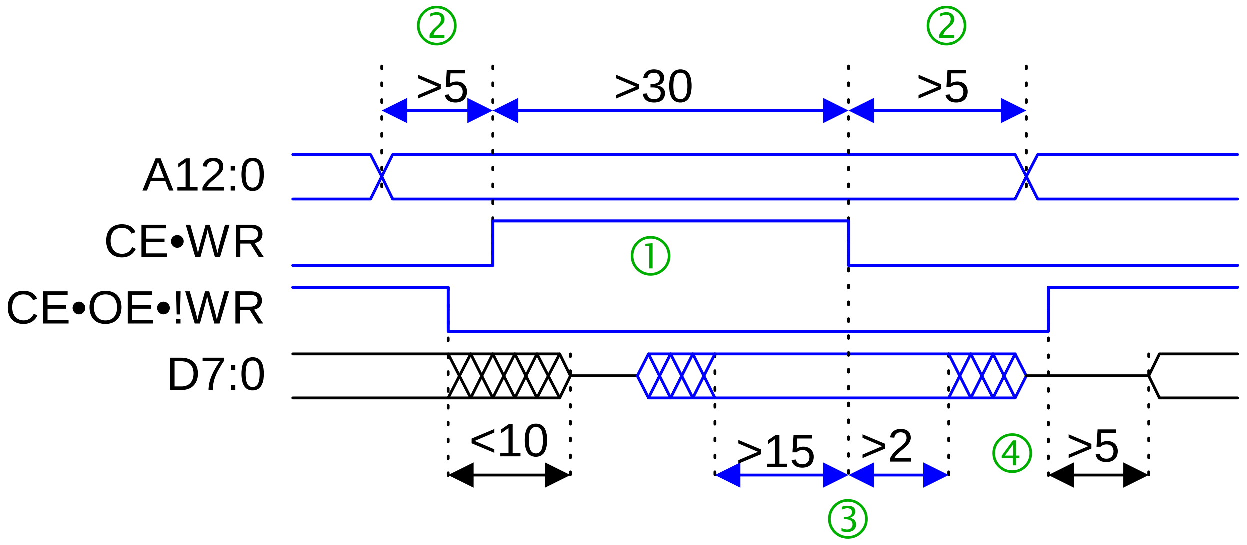
② If a read cycle ends due to OE going low, the outputs go Hi-Z within 10 ns

③ If a read cycle starts due to OE going high, D7:0 stays Hi-Z for at least another 5 ns and the selected word appears within 20ns

You can use CE instead of OE but it is slower: 20 ns to turn off and 35 ns to turn on (in parentheses on timing diagram).

When reading data, the propagation delay to the D7:0 outputs is called the RAM's *access time*: 35 ns from A12:0 and 20 ns from OE.

Memory Write Cycle



A *write cycle* happens whenever CE•WR is true.

① CE•WR must go high for at least 30 ns.

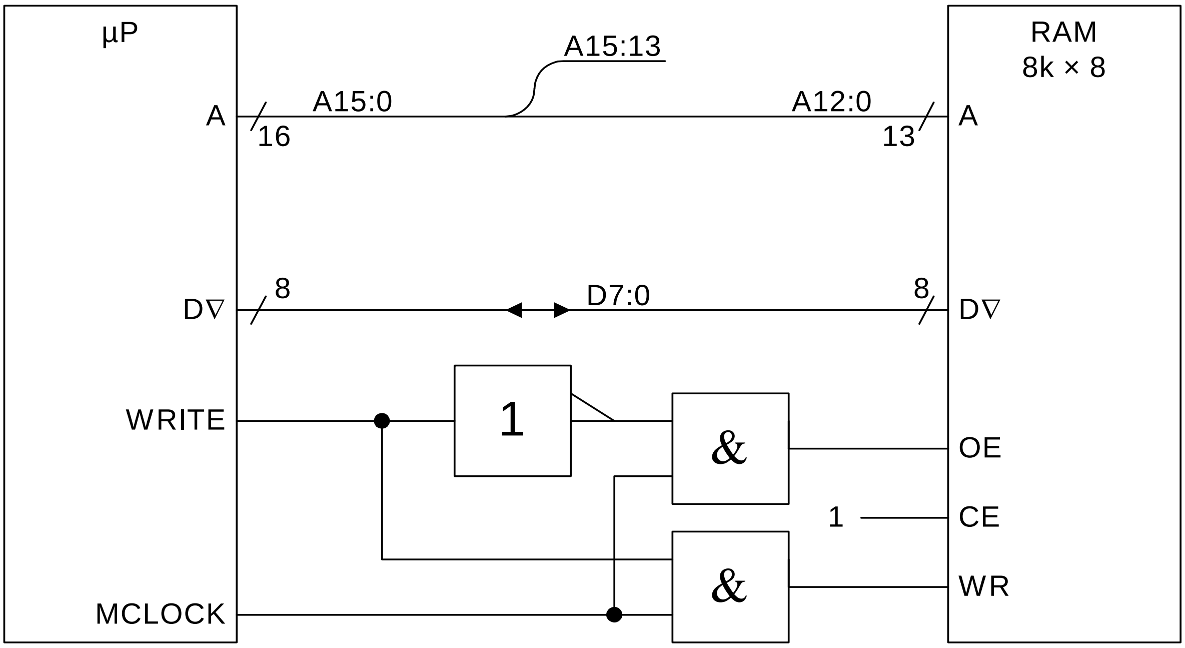
② To avoid writing to unwanted locations, the address, A12:0 must remain constant for at least 5 ns at both ends of the write pulse.

③ Input data D0:7 only matters at the *end* of the write pulse. Setup & hold times of 15 ns & 2 ns define a window within which it must not change.

④ Input When CE•OE•!WR goes high, the memory reverts to read mode. The input data must be removed from D7:0 before this happens.

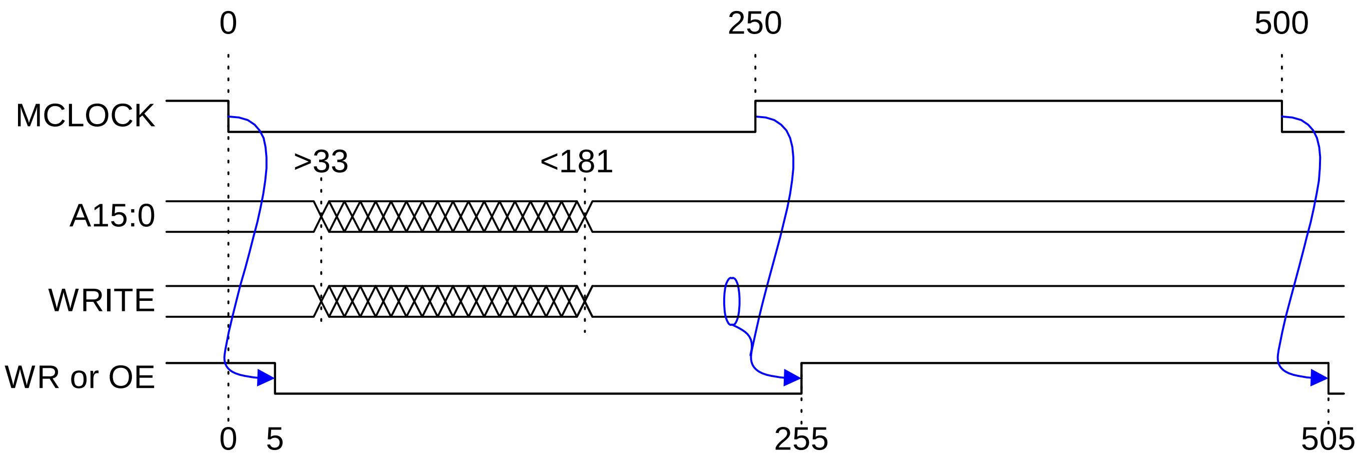
- Timing specifications that end on an output are guarantees from the chip manufacturer (shown in black).
- Timing specifications that end on an input are requirements that the designer must meet (shown blue).

Microprocessor ↔ Memory Interface

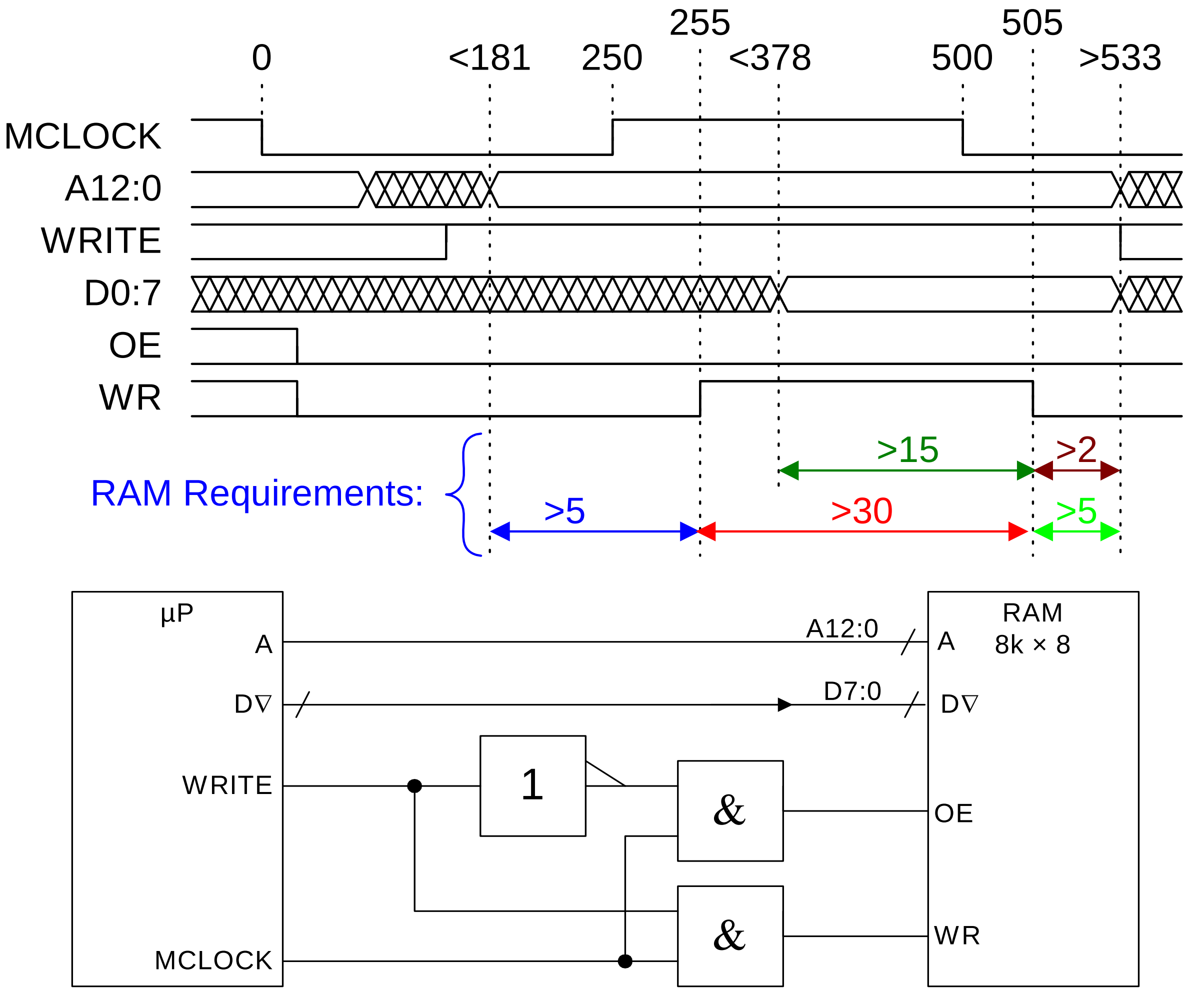


OE = MCLOCK • !WRITE WR = MCLOCK • WRITE

- Reading or writing takes place during the **second half of the clock cycle** when MCLOCK is high.
- WRITE output from µP determines whether WR or OE goes high. Assume NAND gate delay = 5 ns.

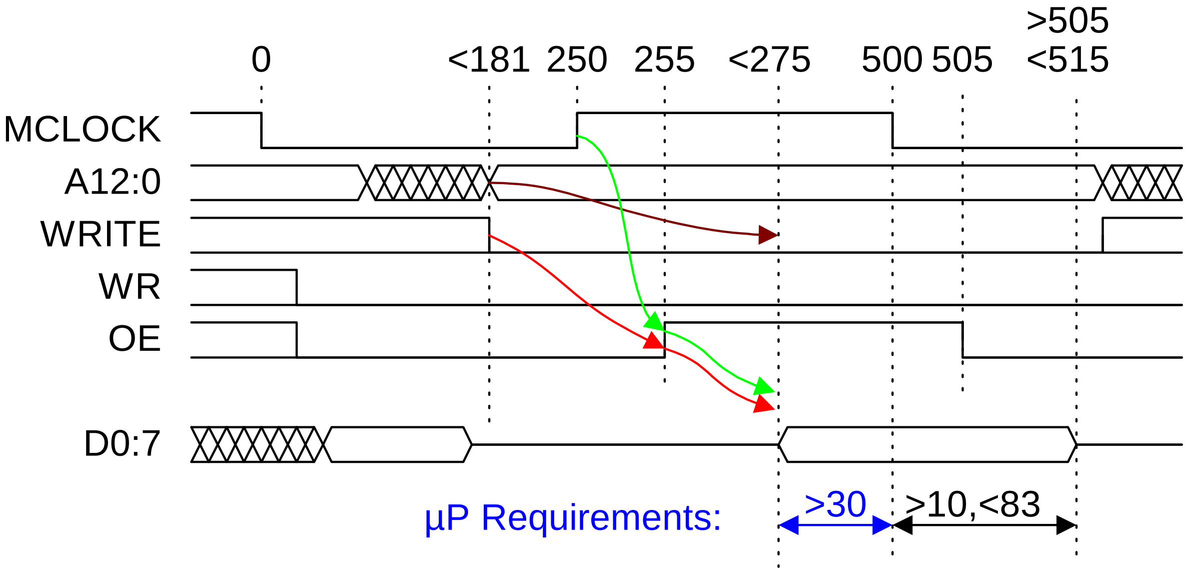


Microprocessor Write to Memory

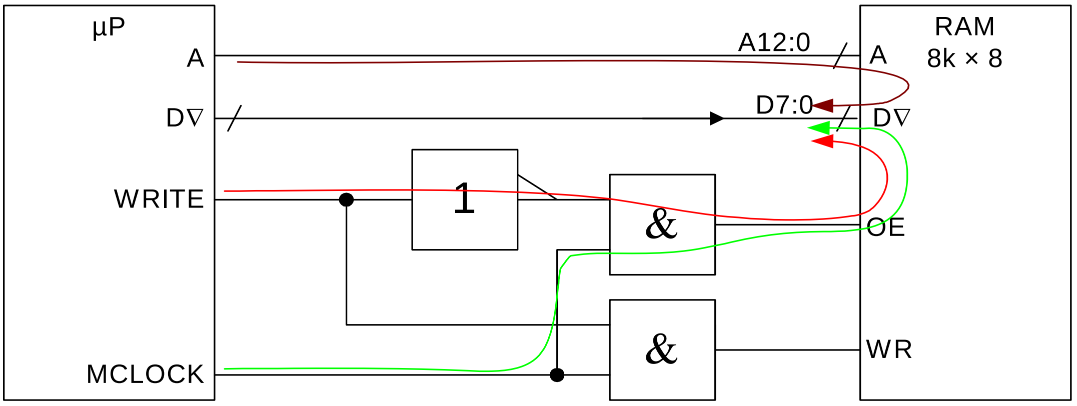


- µP emits data within 128ns of MCLOCK↑
- Requirements:
  - Addr→WR setup: 181+5 < 255 ✓
  - Data →!WR setup: 378+15 < 505 ✓
  - WR pulse: 255+30 < 505 ✓
  - Addr hold: 505+5 < 533 ✓
  - Data hold: 505+2 < 533 ✓

Microprocessor Read Setup Time

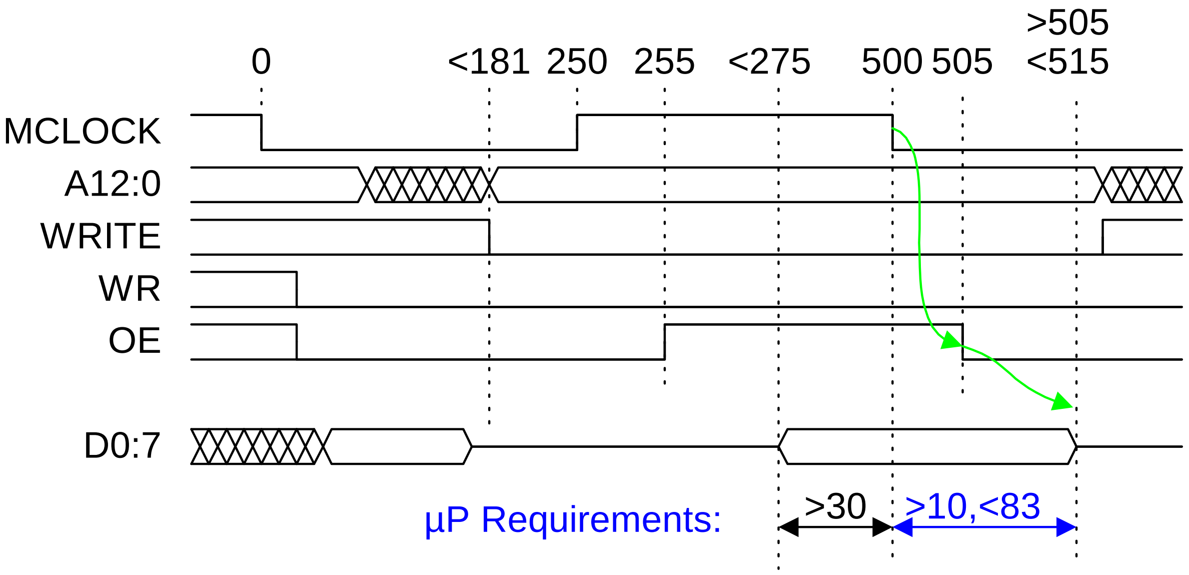


- Data Setup time: 30 ns before MCLOCK↓.
  - Three paths must be satisfied
  - Check each one individually

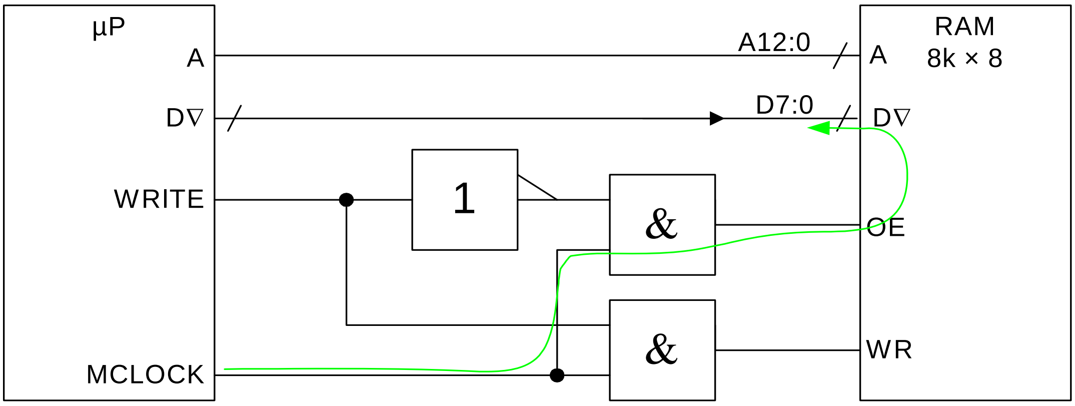


- Requirements:
  - Addr to Data setup: 181+35+30 < 500 ✓
  - WRITE to Data setup: 181+5+5+20+30 < 500 ✓
  - MCLOCK to Data setup: 250+5+20+30 < 500 ✓

Microprocessor Read Hold Time



- D7:0 must go tristate 10 to 83 ns after MCLOCK↓.
  - MCLOCK path is the only relevant one
  - OE↓ to tristate delay varies between 0 and 10 ns



- Requirements:
  - Min hold: 505 > 500+10 ☜
  - Max hold: 515 < 500+83 ✓

May need to add some delay to !OE signal to meet min hold

Quiz Questions

1. What is the *access time* of a static RAM?
2. When writing to a static RAM, why is does the state of the data inputs matter only at the end of the write pulse?
3. How do you check timing constraints if the manufacturer specifies a maximum propagation delay but no minimum ?
4. How do you check timing constraints if the validity of an output depends on several of the input signals ?

Answers are all in the notes.