Unsynchronized I/O Fails in the Classroom

How does I/O work in the classroom?
1. You blurt out a question.
2. The professor stops and looks around.
3. The professor makes an intelligent response. For example: “Eh? What? Huh? Was there a question?”
4. Then the professor goes back to writing.

The problem? No synchronization!

I/O Requires Synchronization to Succeed

So you try again...
1. You raise your hand.
2. Eventually, the professor notices.
3. The professor asks, “Yes?”
4. You ask your question.
5. The professor says it’s a good question.
6. And assigns it as a homework problem.
7. Then the professor goes back to writing.

Handshaking in Hardware Similar to Q/A in Class

In a classroom, we use our
◦ our hand to indicate status (have a question), and
◦ our voice to deliver data (the question).

When hardware devices
◦ lack a common clock,
◦ they need synchronization to communicate.

The simplest form is handshaking.
Status Signals Used to Synchronize Actions

In a (hardware) **handshaking protocol**,
- each side sends a **1-bit status signal** to the other (constantly—there's no shared clock)
- and data producer sends **N bits of data** to consumer.

To send data, producer
1. sets the values of the **data bits**, then
2. flips its **status bit** to indicate new data, then
3. waits for consumer to finish reading (at which point the consumer flips its bit).

Status Signals Combine to Obtain Ready/Not Ready

With processors and I/O devices,
- processor much faster than I/O devices
- (as students are to professors).

In LC-3 model, the **two status signals are combined into a single bit**
- with XOR in datapath (assume both bits start at 0)
- before presenting to the processor, so...
- **1 means 'ready'**
- **0 means 'not ready'**

LC-3 Changes Status Signal Implicitly with Keyboard

Consider the keyboard and the LC-3.
After the human presses a key:
1. Keyboard makes key value (in ASCII) available as data.
2. Keyboard changes status bit to 1 (ready).
3. LC-3 processor observes status bit.
4. Processor reads key value, **implicitly changing status bit back to 0** (not ready).

Status and Data Bits Stored in Registers

**So where are the ‘status’ and ‘data’ bits?**
In registers, of course!
LC-3 has 16-bit word size, so uses **16-bit registers** for convenience.

For the keyboard...
- **KBDR: Keyboard Data Register** (the key)
- **KBSR: Keyboard Status Register** (the ready signal)
KBDR is 8-bit ASCII Zero-Extended to 16 Bits

The KBDR holds a single keystroke as an extended ASCII (8-bit) character. The character is in KBDR[7:0]. The upper 8 bits, KBDR[15:8], are not used, and are filled with 0s when read.

Where is the Status Bit in KBSR?

Where should the status bit go in KBSR?

I like 42, and 4+2 = 6, so maybe in bit 6?

Is there a better reason for another bit?

Only Ready Bit (KBSR[15]) is Usable by Software

KBSR[15] is the ready bit for the keyboard.
- When KBSR is loaded,
- the N condition holds the ready bit.

Protocol Violations Considered Harmful

What happens if a second key is pressed before the processor reads the first key?

No place to store a second key value, so one key is lost (which key depends on the implementation).
Protocol Violations Considered Harmful

What happens if the processor reads KBDR before a key is ready?

The bits read have no meaning.
(In some implementations, they may not even be bits!)

LC-3 Output Also Uses Two Registers

LC-3 has only one input device: the keyboard. And one output device: the display.
In LC-3, output is also based on ASCII.*
And LC-3 uses two more 16-bit registers…
◦ DDR: Display Data Register (the character)
◦ DSR: Display Status Register (the ready signal)
*Not generally true of actual devices; take ECE391.

LC-3 Changes Status Signal Implicitly with Display

Display is initially “ready” to display a character.
1. LC-3 processor writes character into DDR, implicitly changing status bit to 0 (not ready).
2. Display reads and displays the character.
3. Display changes status bit to 1 (ready).
4. LC-3 processor waits to observe ‘ready’ before writing another character (step 1).

Processors May Spend Many Cycles Waiting

In practice, LC-3 code
◦ must begin by waiting for ‘ready,’ (Step 4)
◦ as the code cannot know whether other code has already used the display.
Generally,
◦ the display is slower than the LC-3, so
◦ the LC-3 may spend many cycles waiting.
Humans are even slower, so waiting for keystrokes takes even longer.
Only Ready Bit (DSR[15]) is Usable by Software

DSR is similar to KBSR: only one bit is used.
- When DSR is loaded,
- the N condition holds the ready bit.

Other bits are undefined and must be ignored by software.

Write 8-bit ASCII Zero-Extended to 16 Bits to DDR

The DDR accepts a single character as an extended ASCII (8-bit) character.
The character is in DDR[7:0].
The upper 8 bits, DDR[15:8], are undefined, and should be filled with 0s when written.

Protocol Violations Considered Harmful

What happens if a second character is sent before the display handles the first character?

No place to store a second character, so one character is lost (which character depends on the implementation).