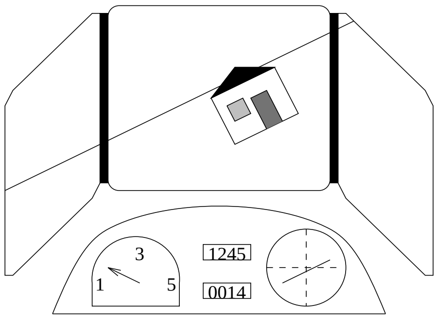
Display Systems / Simulations

- Flight Simulator

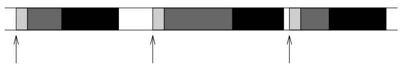


Regular Tasks

- *Time Driven*

  Tasks are synchronized to a system (*real-time*) clock. This keeps the operations regular regardless of the variation in computational complexity.

  - Display update
    Display should be updated once every 20ms to appear smooth.
  - Calculation of new flight parameters
    For each update we will recalculate the flight parameters.



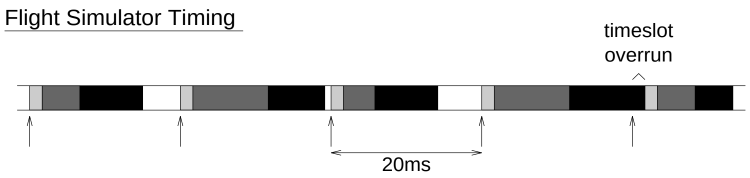
Swap real and shadow screens
Recalculate flight parameters
Create new display on shadow screen

Irregular Tasks

- *Event Driven*
  - Response to operator input

    The simulator must update certain program parameters in response to operator input.
    Operator input may cause execution of additional routines to recalculate flap position or to adjust active joystick feedback.
  - Asynchronous & unpredictable

    These tasks are asynchronous & unpredictable, the requirement for instant response will *interrupt* other processes and may upset timing for time driven tasks.



Swap real and shadow screens
Recalculate flight parameters
Create new display on shadow screen

- Timing Problems
  A timeslot overrun may be caused by overflying a flock of virtual sheep or by a novice pilot who keeps making radical adjustments to the controls.

Real-Time Applications and Issues

Task Priority

Where we have less time available than we might need, we must prioritize tasks.

- Essential - Hard real-time components.

  Since this is not a real plane, no task has *life or death* importance, but failures in certain tasks will effect the program usability.

- Desirable - Soft real-time components.

  Some tasks may have to be delayed or omitted when timing is tight.

Real-Time Applications and Issues

System specification

A system specification for the simulator may look something like this:

- Hard real-time requirements
  - the flight parameters *must* be recalculated every 20ms.
  - the system *must* respond to operator input within 50 ms.

- Soft real-time requirements
  - The screen display should be updated every 20ms
    occasional update gaps of 40ms are acceptable

Key Issues - Time

Correctness

Whereas a batch program has completed correctly provided that it produces the correct outputs for a given set of inputs, we require more from a real time system:

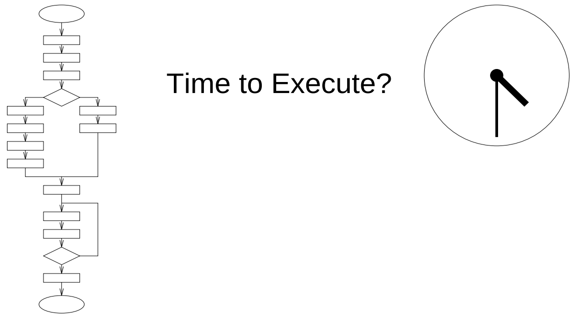
Temporal Determinism[1]

- *The response to external events must be deterministic with known response times.*

- *The knowledge is absolute that information will not be lost, nor will the response be too slow for the environment under control.*

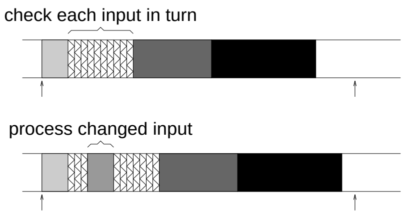
[1] a.k.a. Predictable Response Times

Key Issues - Time



Provided that we can calculate the worst possible execution times, we can provide guaranteed predictable response times.

Input processing

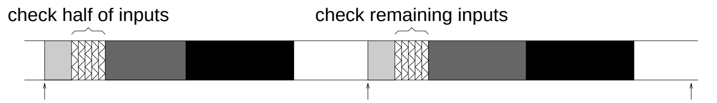


Before recalculating the flight parameters, we poll each input in turn to check if it has changed. Changed inputs require input processing.

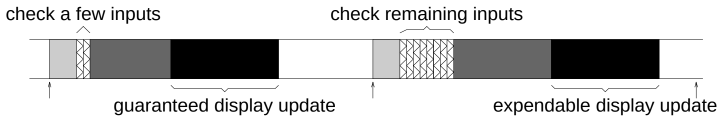
In this way we can see that the simulator should respond to a change in input within about 20ms.

In order to guarantee response times under severe conditions we may have to reschedule the tasks:

- Since constraints on input response are less restrictive.

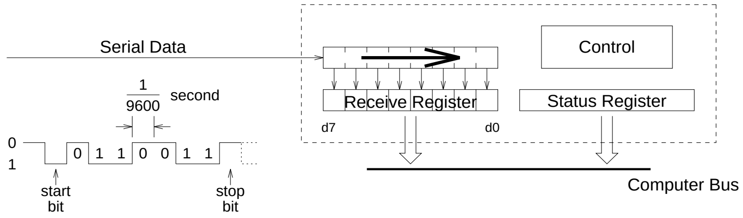


- Allow for the sacrifice of display update where many inputs coincide.



Faster Response Times

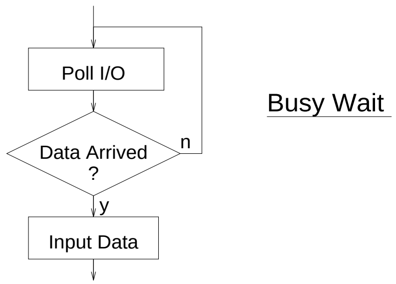
- RS232 data input requires a much faster response time.



Data must be extracted from the receive register before the next byte is received.

Continuous Polling

Continuous polling offers fastest response time.

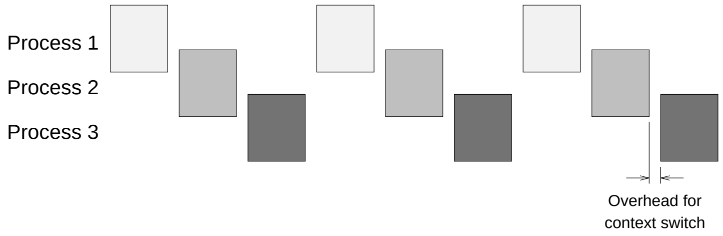


We describe this as *busy wait* since the processor is busy, yet no work is done.

Real-Time Applications and Issues

Faster Response Times With Multiple Tasks

Continuous polling is seldom the best solution for real-time systems since most will have to perform other tasks while waiting.



In a simple multi-tasking system, the response time is limited by the number of other tasks and the duration of a timeslice.

Key Issues - Events

Event Driven Software

With interrupts we can allow the order of execution to be partially determined by external events.

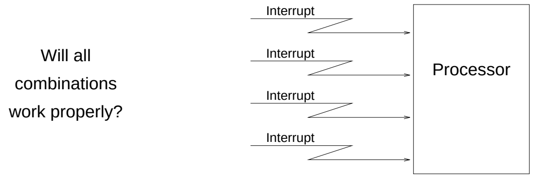
When a processor receives an interrupt signal from an external device, normal processing is suspended in order to run a service routine which deals with the external event.

- Priority routines can interrupt others.
- Very fast response to external events.
- No *busy wait*

Key Issues - Events

Correctness

Since we no longer control the order of program execution we must consider:



Event Determinism

- *Absolute knowledge of the states that the system and processor will go through in all cases.*
- *Knowledge that there are no combinations that will cause failure or lockup.*