EECE 499/693: Computers and Safety Critical Systems

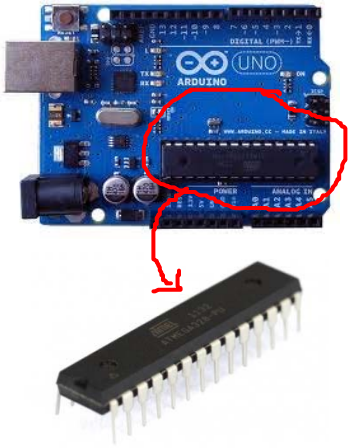
4 Design of Fail-Safe Computer System
A. Simplex System

Instructor: Dr. Charles Kim

Electrical and Computer Engineering
Howard University

www.mwftr.com/CS2.html

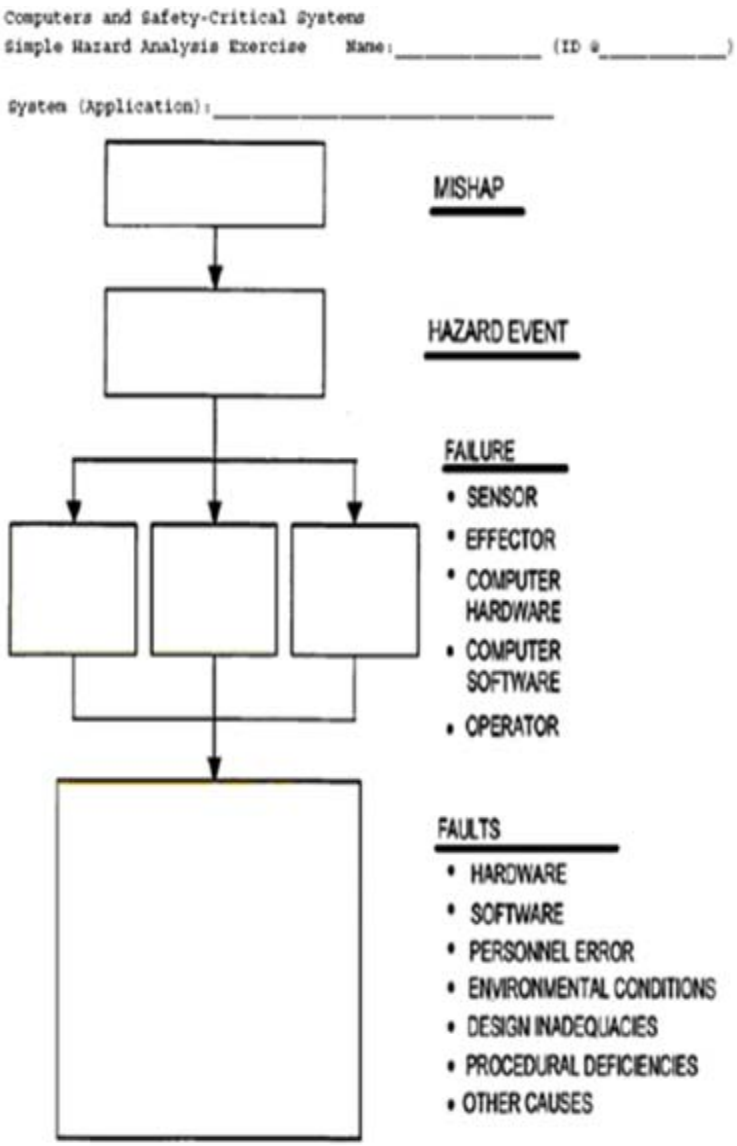
REMINDER -- Failure Rate Determination – Class Project



- Failure Rate Calculations:
  - 1. The popular microcontroller board ***Arduino UNO*** is built on Atmel microcontroller ATmega328. Referring the Atmel Microcontroller datasheet and the **MIL-HDBK-217 manual**, determine **the failure rate of the ATmega328 microcontroller**
  - 2. Texas Instrument's TLC2254M is Quad micro-power operational amplifier, and is QML certified for Military and Defense Application. Determine the **failure rate of TLC2254M** by referring MIL-HDBK-217 and TLC2254M datasheet from Texas Instrument. Note that TLC2254M is a Hybrid IC with numerous resistors, transistors, diodes, and capacitors, which all are to be considered in determining the failure rate
- Report should have details steps with explanations and justifications.
- Report Submission Due: **Nov 4, 2014**
- **NOTE: <u>Oct 28 and Oct 30 --- Project Week</u>**

Background

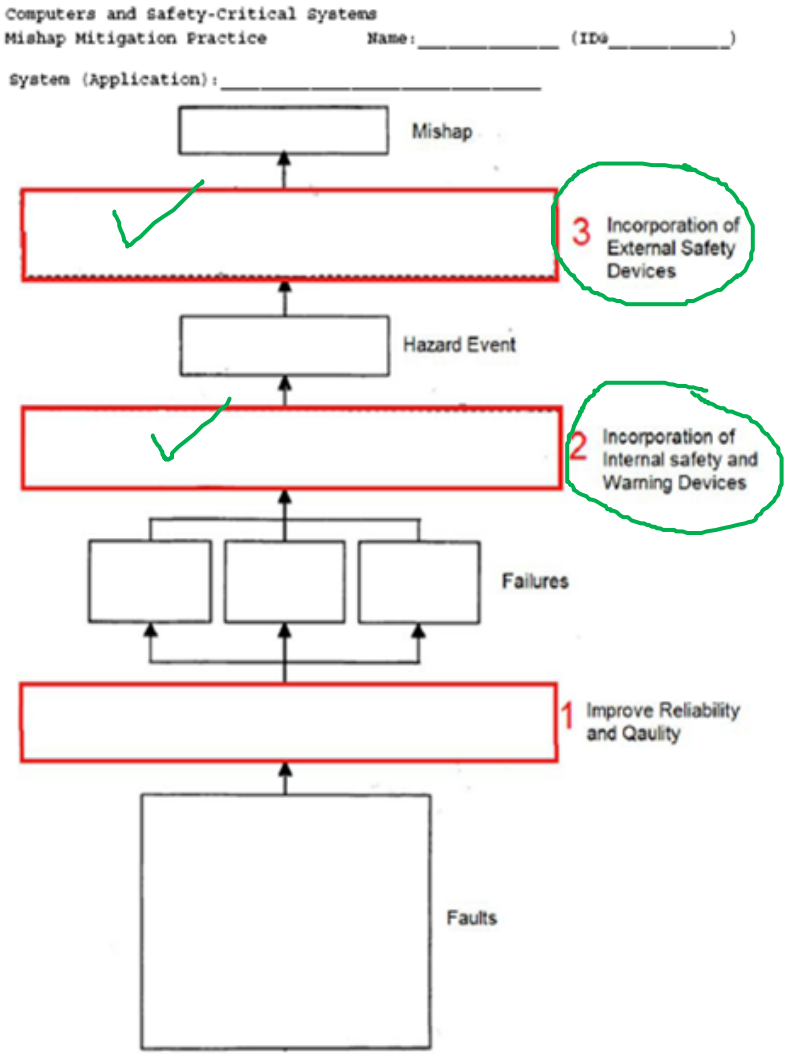
- **Chapter 2**: Computer Systems
  - **Basic computer system** with H/W, S/W, and Operator actions (without safety features)
  - Mishaps and Hazards in the computer systems
  - 5-Step system design for a selected computer control system *
- **Chapter 3**: How Computers Fail
  - Component Failure Modes and Effects
  - Operator Failures
  - Component Failure Rate Determination
- **Chapter 4**: Design of Fail-Safe Computer System
  - Design steps to make the Basic Computer System **fail-safe**



*** Redo option – by Thursday**

General Consideration

- Remember Hazard Mitigation steps?
  - 1 Improve component reliability and quality
  - 2 Incorporate internal safety and warning devices
  - 3 Incorporate external safety devices
- Focus
  - 2 and 3 above
  - Incorporation of **internal** and **external safety devices** in to a basic computer system
    - Simplex Systems
    - Duplex (Redundancy) Systems

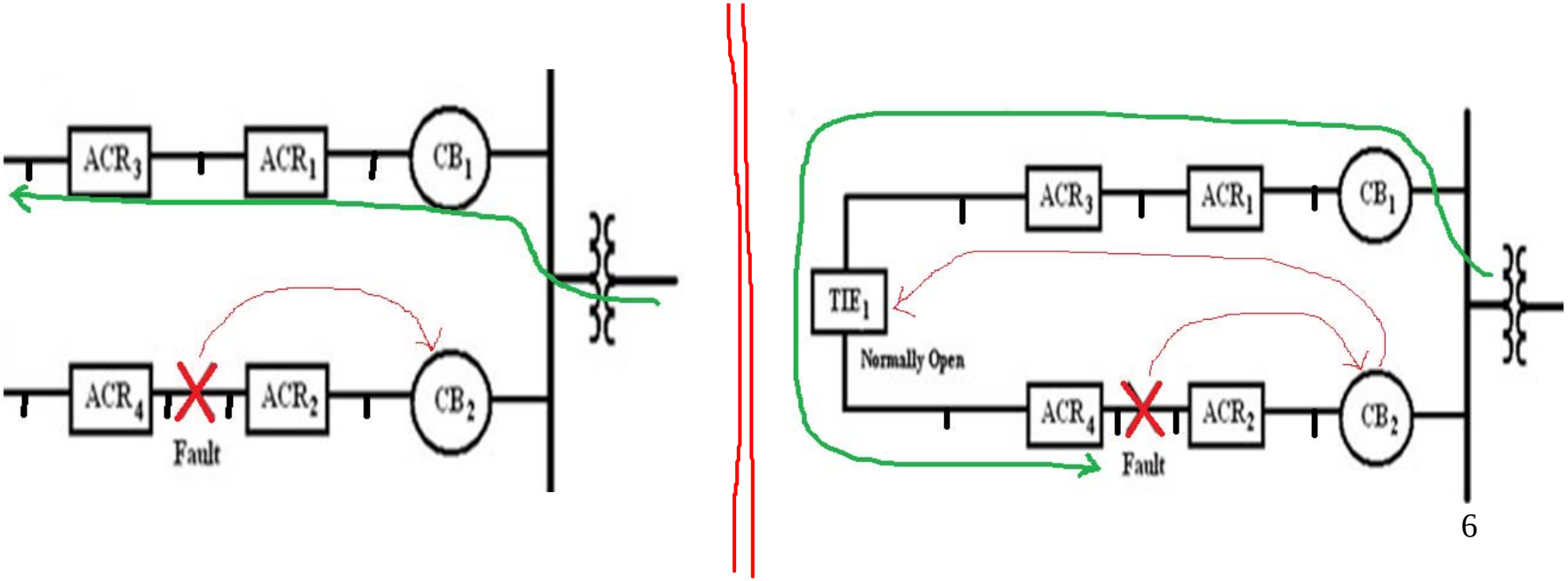


Fail-Safe vs. Fail-Operate

- Fail-Safe vs. Fail-Operate
  - Fail-Safe System:
    - In the event of failure, a system will revert to a non-operating state that will not cause a mishap.
    - A system must be able to detect faults or failures, and reconfigure itself to the safe, non-operating state
  - Fail-Operate System:
    - In the event of failure, a system will reconfigure itself so that safe operation will continue without noticeable interruption
    - A system must detect faults and failures, and reconfigure to the safe, normal operational state with unnoticeable interruption
- What's the current trend in industry
  - **Mix** of Fail-Safe and Fail-Operate approaches
  - Fail-operate system is preferred but **price** of such a system is not preferred
  - In cost, a fail-operate safety-critical system exceed x10 or x100 of a fail-safe counterpart.
  - In either system, **failure detection capability is essential**

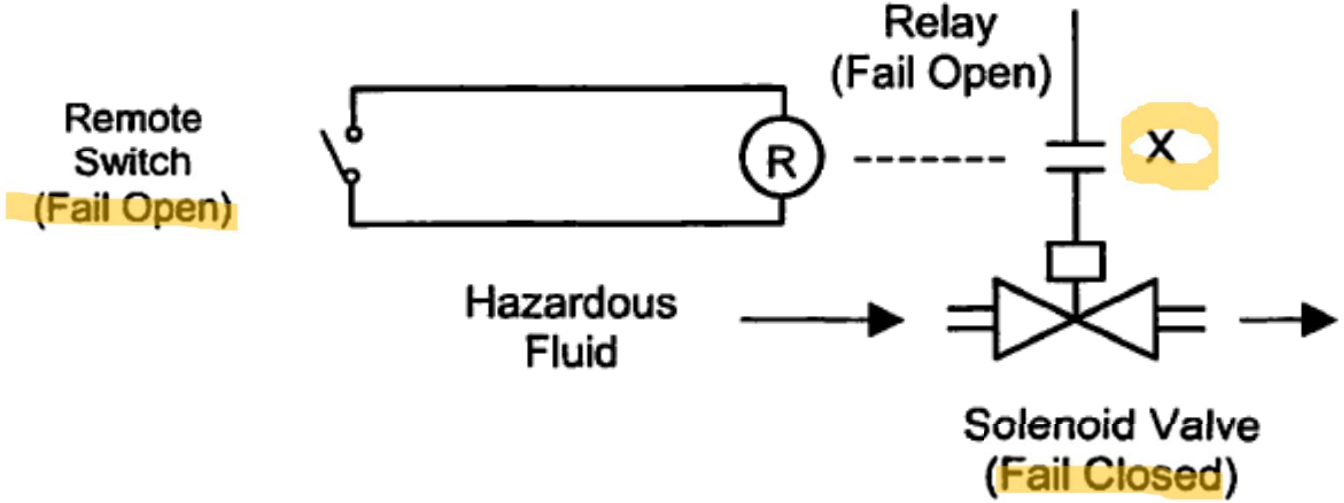
Fail-Safe and Fail-Operate in Power Utility

Circuit Reconfiguration



Inherently Fail-Safe System

- Use and connection of components, by which, when the failure of any component, automatically causes the system to revert to a fail-safe state.
- Example: ("Closed Valve" is Fail-Safe)
  - Failure of remote switch opens the relay → valve is closed {for Normal Close (NS) type: **Default** position is Close}
  - Failure of Relay closes the valve
  - Failure of Valve closes itself



Everyday Inherently Fail-Safe System

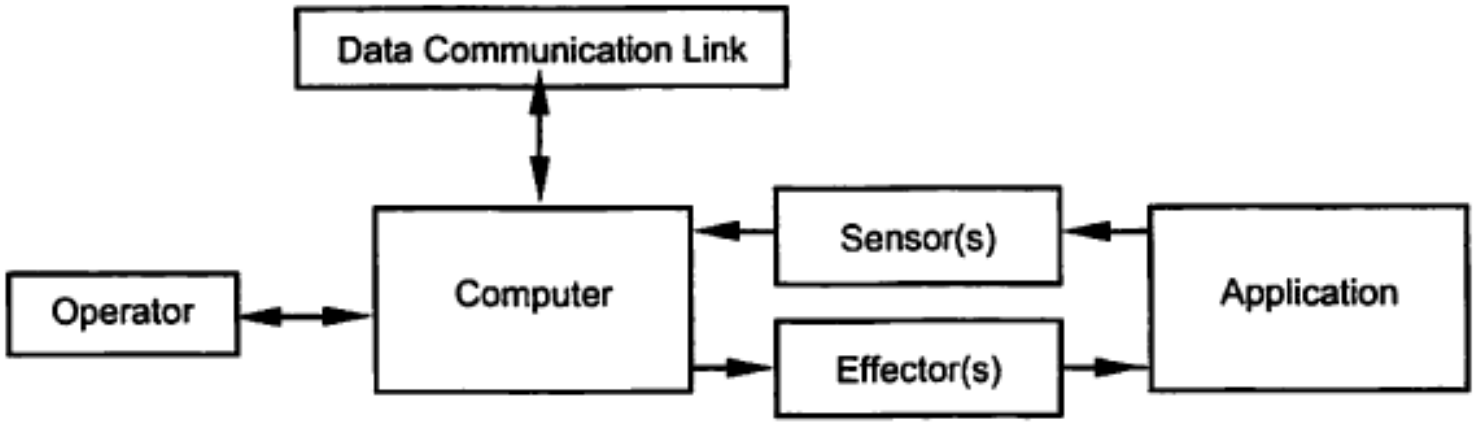
- Other Examples
  - Lawn Mower
  - Dead Man's Switch
  - Others

Inherently Fail-Safe System

- Can we make entire computer system inherently fail-safe?
- No. Why?
  - Computer hardware and software separates sensors and operator inputs from actuator and operator outputs
  - Failure modes of components are not well determined
- So, what approach?
  - Use of computer (or engineer) “intelligence” to detect faults and failures not readily detected by conventional electromechanical or analog methods.
- Fail-Safe Design Approach
  - Modification of H/W and S/W in the Basic Compute System so that it
    - 1 Can detect the presence of faults or occurrence of failures, and
      - **Very difficult and challenging**
    - 2 Reconfigure itself to a safe state
      - **Rather straightforward → Change the actuator output accordingly**

Fail-Safe Computer System – Simplex Architecture

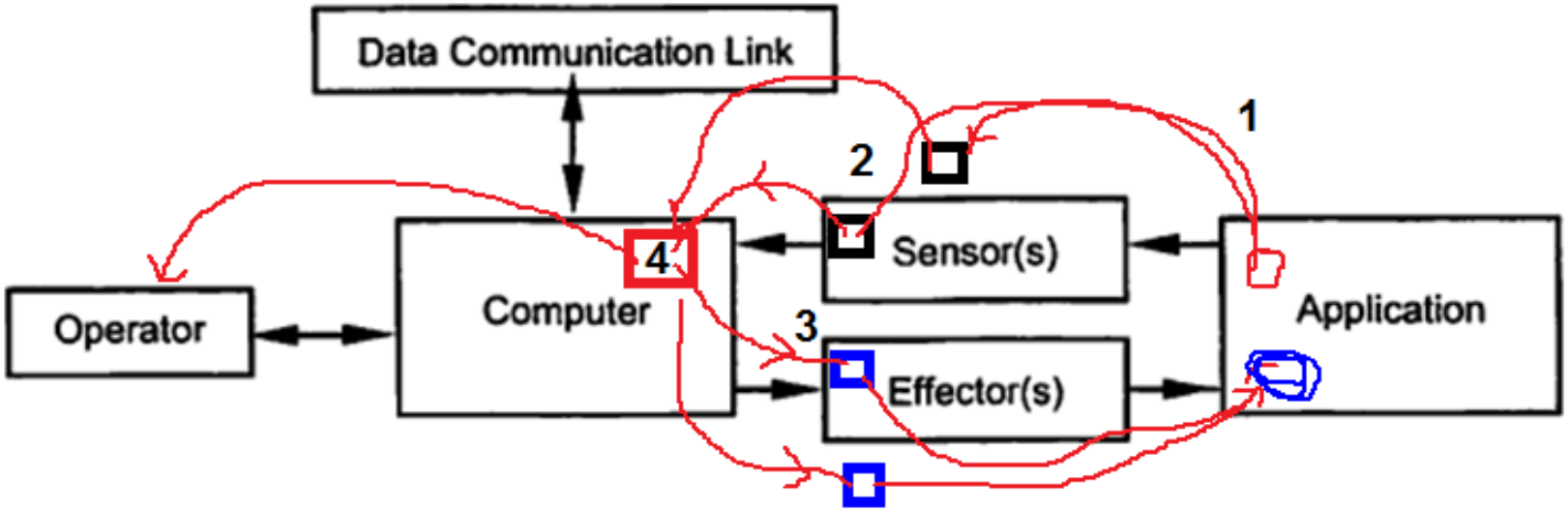
- A widely held belief – “Redundancy must be employed to be fail-safe.” Is this true?
  - What does HRO say?
  - What does NAT say?
- A simplex system: “a system which does not employ redundancy” whether it be a basic system or a fail-safe system
- A simplex System – Example
- We will discuss how this simplex system can behave fail-safe under fault and failure events in each of the component of the example system



Application Failure Control

- Type of application failures: collision, explosion, fire, etc.
- How do we modify the compute system so that application failures can be prevented from occurring?
- 4-Step Process ["Selection of an essential Input and Output" in avoiding the application failure and revert to a safe non-operating state]
  - **Step 1**: Define the physical measurements that can be made on the application which will **indicate it is approaching a failure condition**
  - **Step 2**: Select **appropriate sensors** for making these measurements and **interface them to the computer** (usually the sensors are already likely in place in the basic computer system)
  - **Step 3**: **Select actuators that can be commanded to eliminate** or arrest the conditions leading to the application failure and interface them to the computer (Usually the actuators are likely in place in the basic compute system)
  - **Step 4:** Design and **install software which continuously monitor the output** of the sensors (measurement), and if it detects a fault or onset of failure, signal the actuator to arrest the failure onset, and at the same time signal the operator for safety action based on the circumstances surrounding the application process or for emergency procedures.

Example of **Application** Failure Control

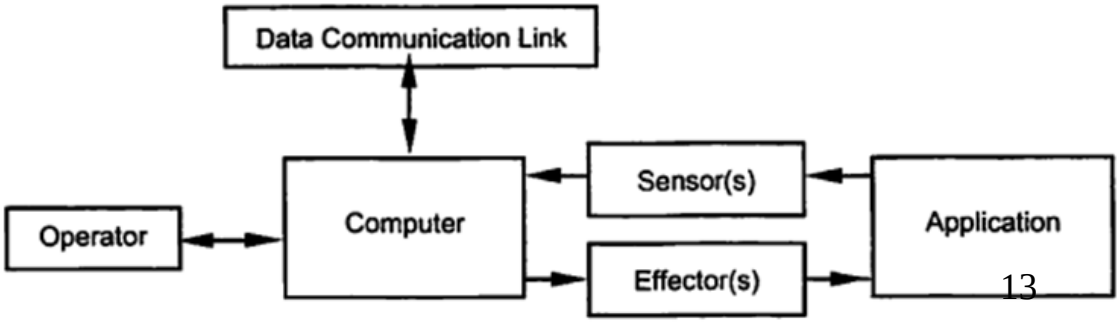


Application Failure Control Examples

Failure	1 Physical Measurement	2 Sensor	3 Effector Action
Fire	Temperature increase	Thermocouple	Reduce/eliminate heating
Over pressure	Pressure increase	Pressure switch Pressure transducer	Reduce pressure
Explosion	Leaking	Gas/vapor detector	Shut off sources

What do we investigate? – Other than Application Failures

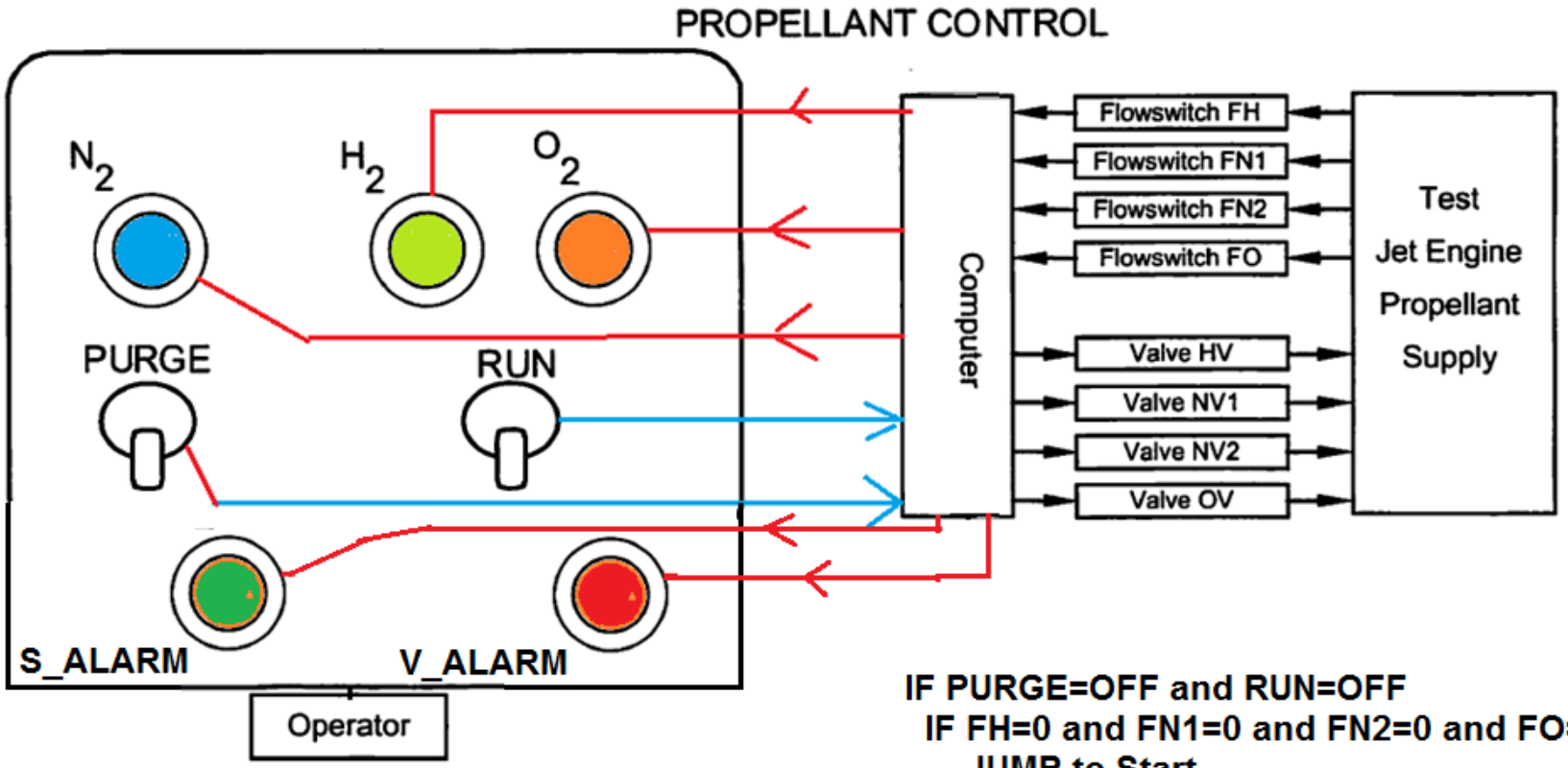
- Remember 2 essential elements for fail-safe system:
  - Failure Detection Capability
  - Reconfiguration to a non-operating safe state
- We will focus on
  - Sensor failure detection scheme
  - Actuator (effector) failure detection scheme
  - Computer Component failure detection scheme
  - System Reconfiguration
  - Handling Power/Interconnect failure
  - Handling Operators failure



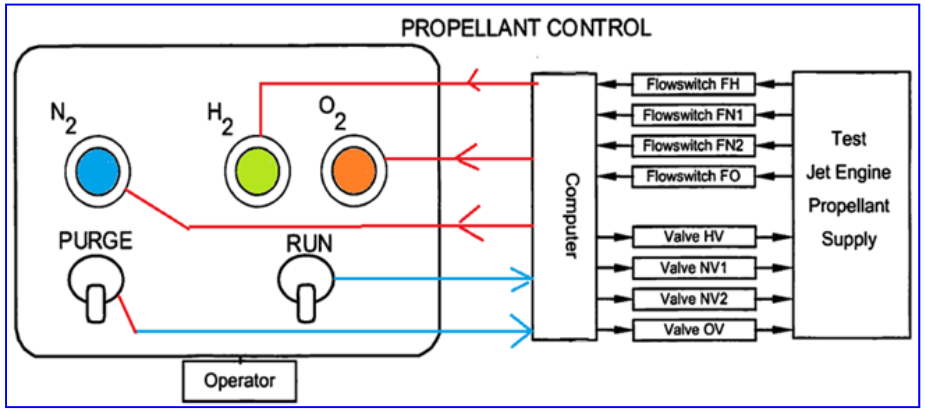
Sensor Failure Detection

- Designer should know, in advance, what the correct sensor output should be when the system is run in real time → Usually, correct sensor output can be predicted.
- Software can be made to measure the **expected sensor output by a given actuator output response** from a command.
  - No sensor failure when the commanded value matches with the actual value
  - Sensor failure if there is mismatch
  - Good only for 1 component [sensor] failure (while assuming that there is NO effector failure)
  - Software? "State Estimation" method

Example of Sensor Failure Detection

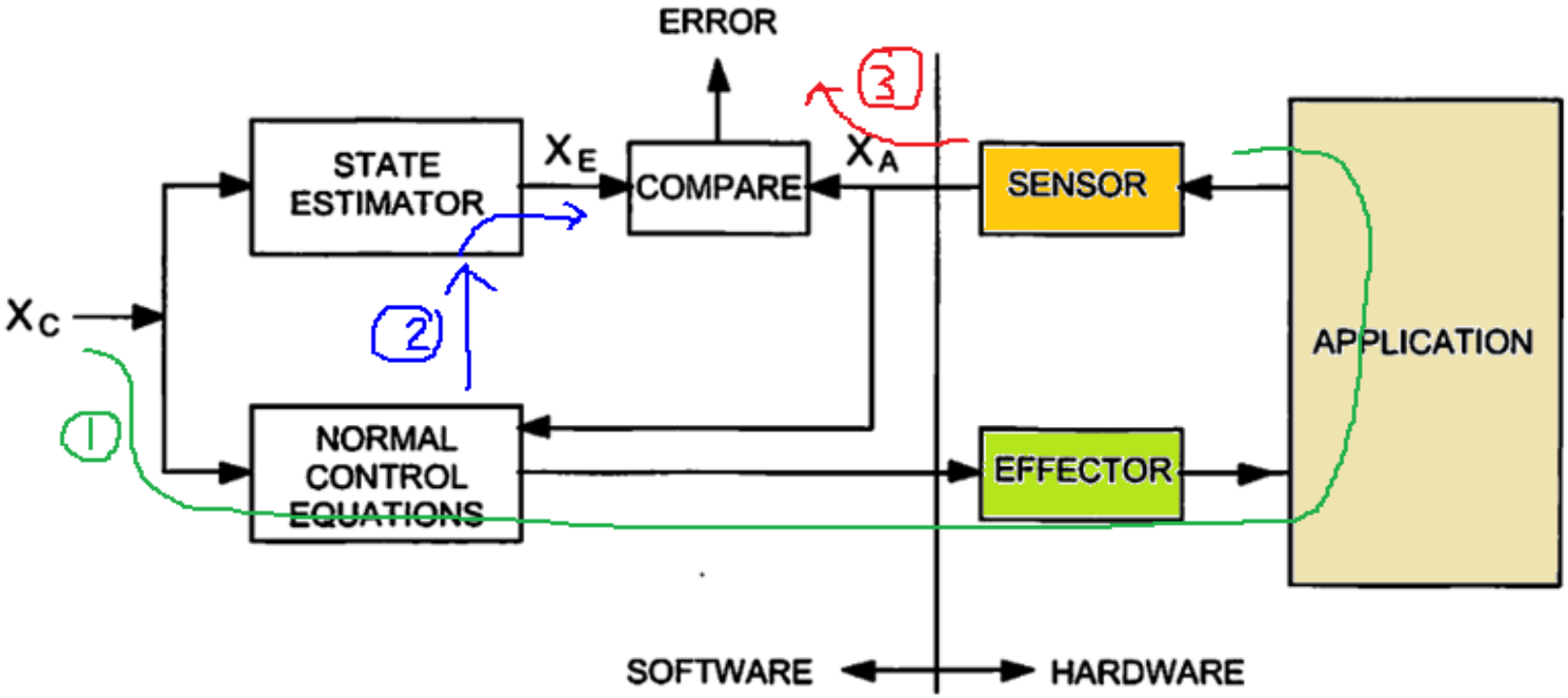


IF PURGE=OFF and RUN=OFF
IF FH=0 and FN1=0 and FN2=0 and FO=0
JUMP to Start
ELSE
S_ALARM =ON



Software - State Estimation

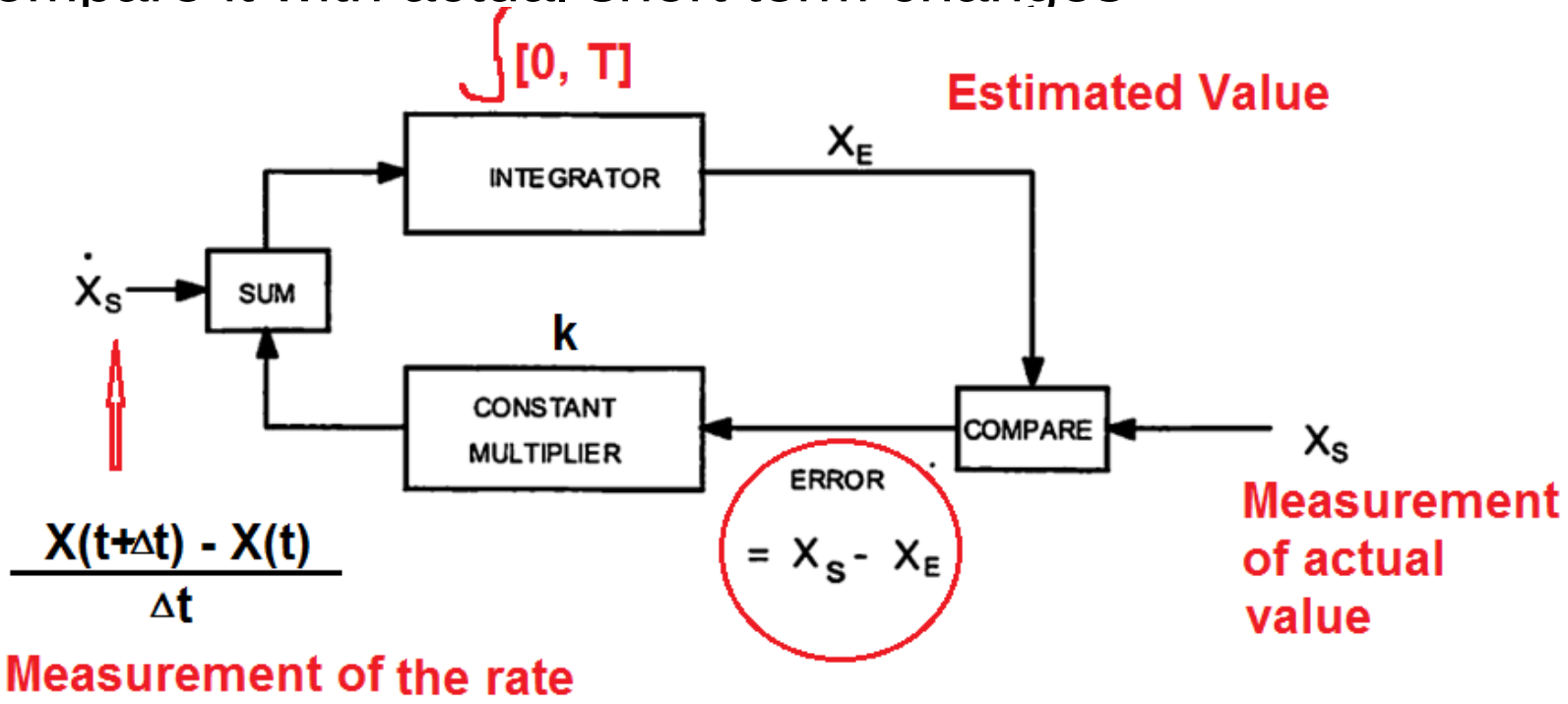
- Detecting Sensor Failure: State Estimation
  - Command (X_C) to normal control equation
  - Actuator feeds into physical system to a state X_A, which in turn will be reported by the sensor
  - Control Equation between command input and sensor output
  - Estimated value X_E that the sensor value exhibit if there is no failure



- **Question: How do we get the correct value from sensors?**

Complementary Filter for Getting Correct Values from Sensors

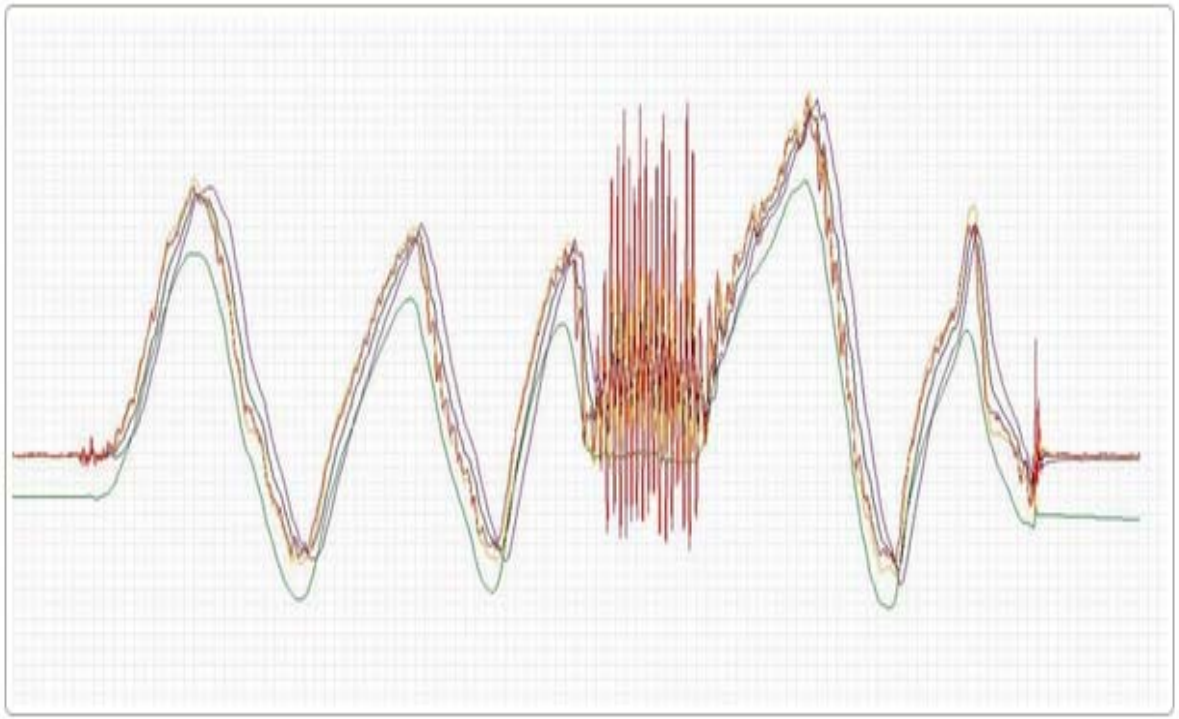
- **Background**: Under harsh physical conditions sensor outputs suffer from the short term change in the conditions → integration of the rates over the short time periods → compare it with actual short term changes

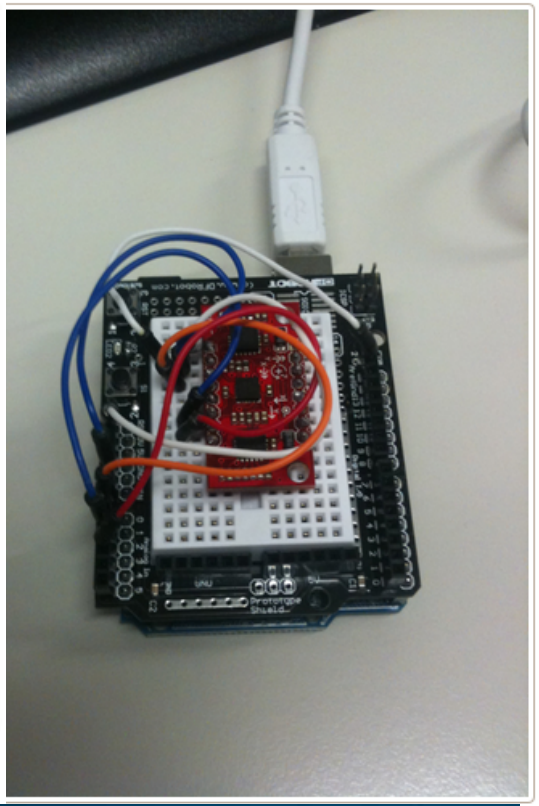


- **Why is this called complementary filter?** (hint: vibration and drift)

Complementary Filter [Example Case]

The hardware I used was composed of:
- Arduino 2009
- 6-axis IMU SparkFun Razor 6 DOF





Color lines:

- Red - accelerometer
- Green - Gyro
- Blue - Kalman filter
- Black - complementary filter
- Yellow - the second order complementary filter

robottini.altervista.org/tag/complementary-filter

ROBOTTINI | Little robots with Arduino

Posts Tagged complementary filter

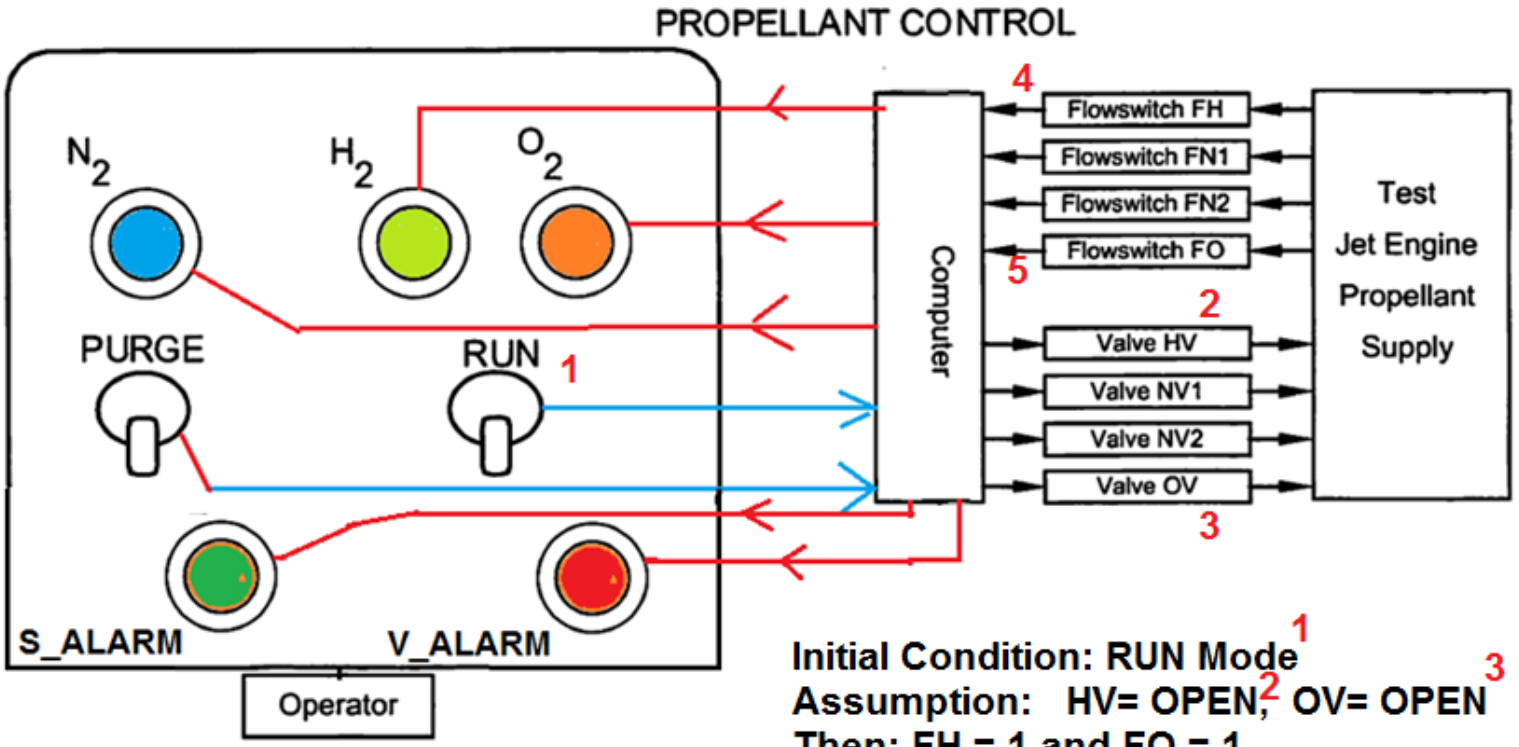
SEP 25

Posted by robottini in Tips | 39 Comments

Actuator (Effector) Failure Detection in Simplex Systems

- Background:
  - When S/W issues an actuator command, it inherently knows the expected actuator response.
- Method:
  - Apply **an instrument to measure the output of the actuator** and feed it back to the computer (and S/W).
  - Then S/W compares the expected actuator action against the actual action to test if the actuator is faulty or not.
  - This method is called a ***Wrap-around Test***.
- **Can we do this for the faulty Takata airbag?**

Actuator Failure Detection Example



Initial Condition: RUN Mode [1]
Assumption: HV= OPEN [2]**, OV= OPEN** [3]
Then: FH = 1 [4] **and FO = 1** [5]

Suppose: OV = CLOSED by mechnical problem [3']
Then: FH =1 [4] **and FO = 0** [5']

What do we do then for fail-safe?

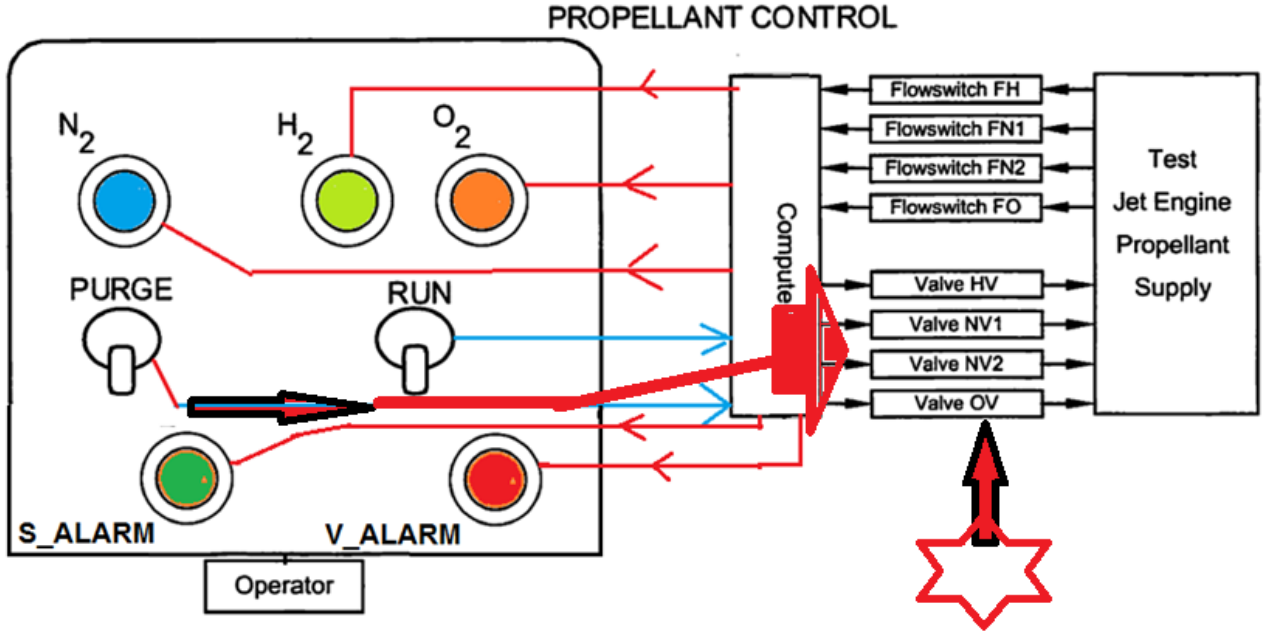
- Problems (when the mechanical problem causes OV open in an Close command): Detection may be made **only after** the unwanted release of gas

Problems in Sensor/Actuator Failure Detection

- Consider a condition:
  - when the mechanical problem causes OV open in PURGE command)
  - Detection (by the Sensor/Actuator Failure Detection Methods) may be made only after the unwanted release of gas
  - Ideally, we want to detect the **onset** of actuator failure, not the **failure itself** (or **after** the failure)
- So what would be a better option?
  - Detection of the mechanical movement of a valve, instead of detection of gas by sensing the gas flow sensor.
  - Cf. “Motion detector” vs “Presence detector”
  - Monitoring of the initial valve movement

System Reconfiguration by S/W Incorporation

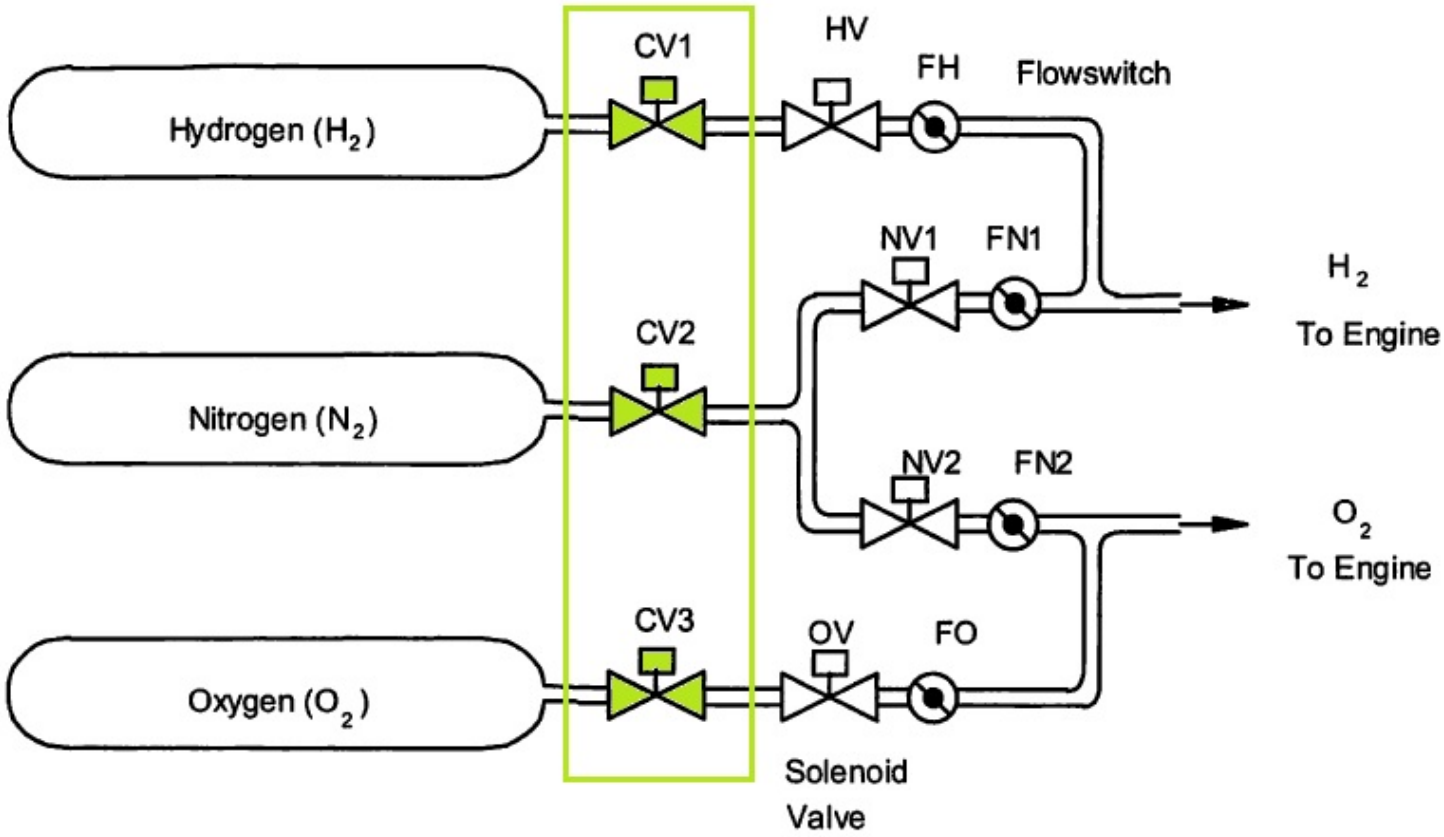
- How to reconfigure the system to a safe state?
- **Incorporation of S/W** so that it **changes the actuator output after a failure** has been detected such that the system will automatically assumes a safe state



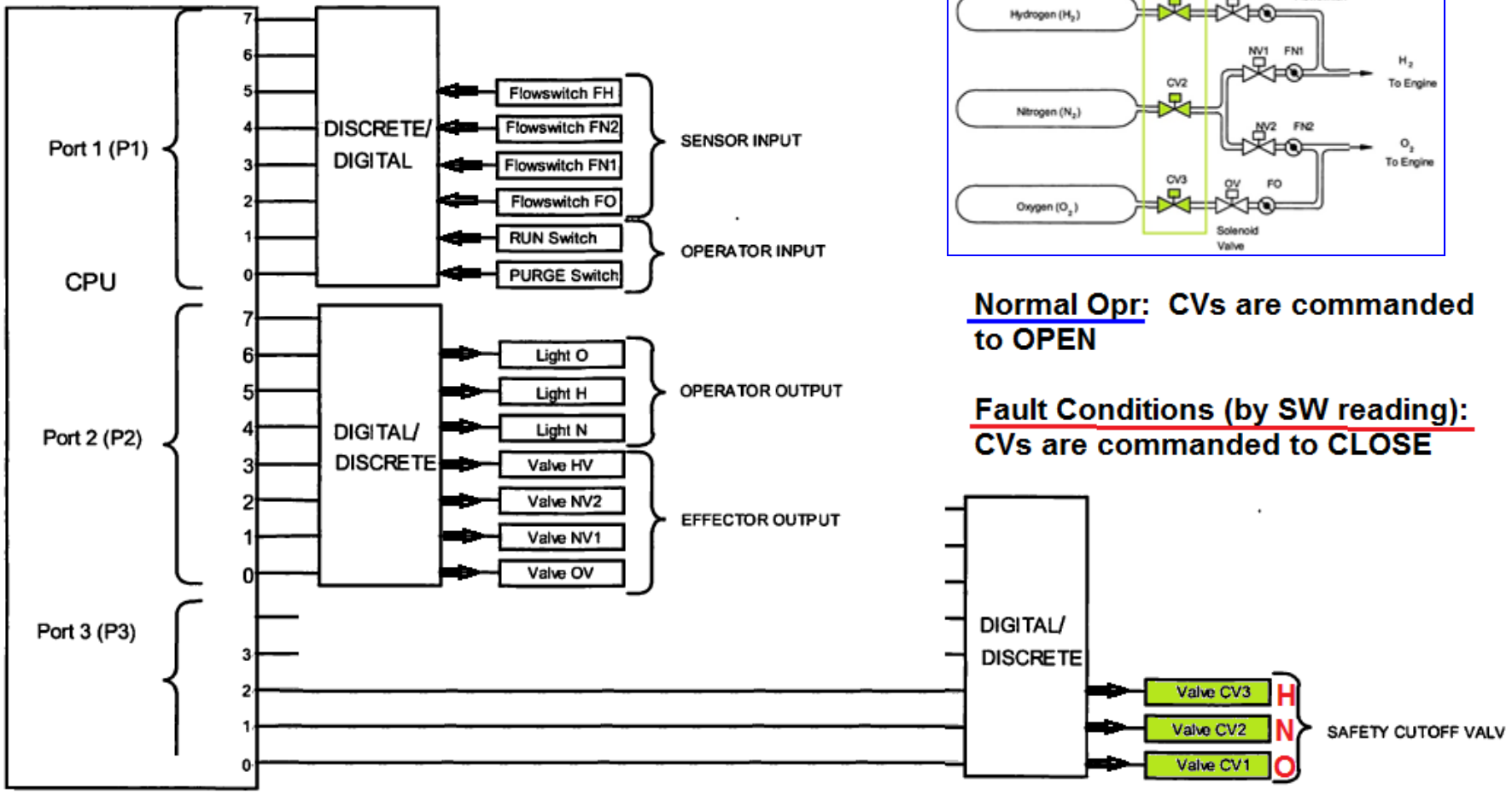
- Example:
  - **in PURGE command, when OV fails to close, then reconfigure the system by commanding all valves to OFF position**.
  - Question: Does this work?

Design (not just S/W) Modification for Safe Configuration

- Both H/W and S/W modification is required so that the resulting system can be safely reconfigured in the face of all possible failures.
- **NC** cutoff valve (CV) is placed upstream for each line
- What would HRO say? What would NAT say?



Modification – Programming Model



Normal Opr: CVs are commanded to OPEN

Fault Conditions (by SW reading): CVs are commanded to CLOSE

- **How do we prepare for cutoff valve (CV) failure? (CCF, CMF)**

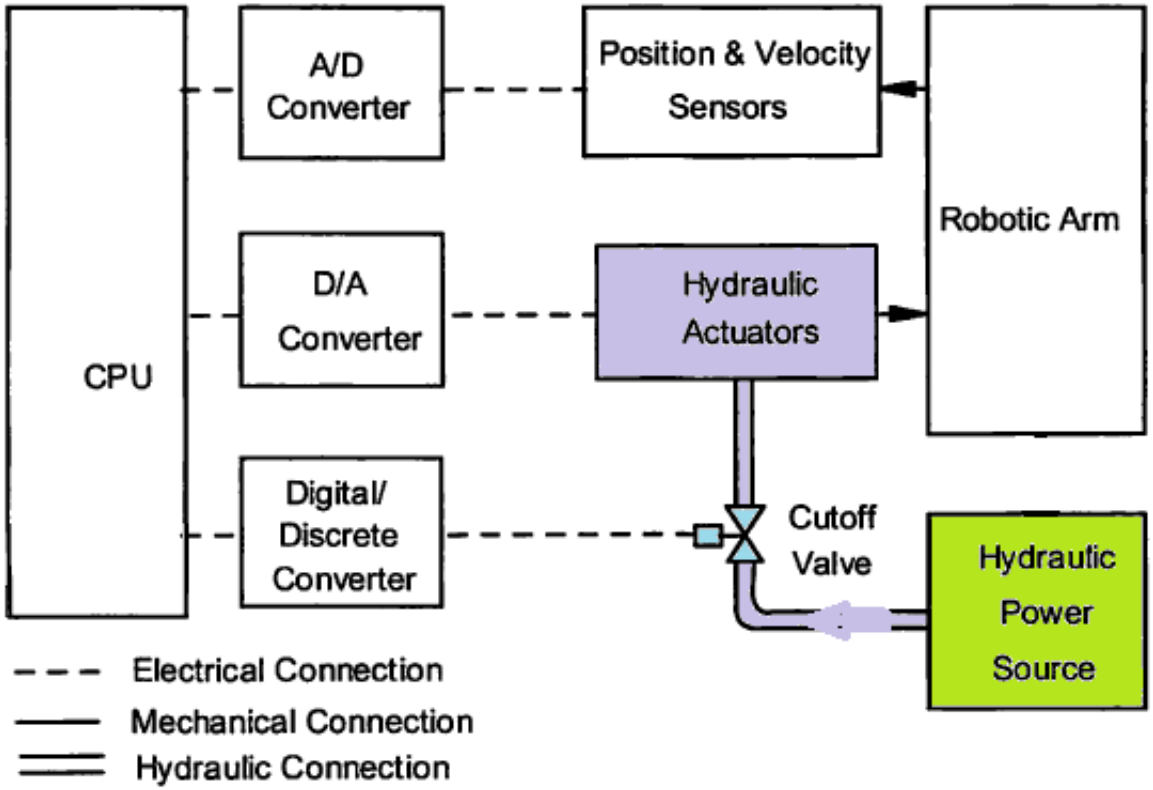
  → **Need to close ALL valves when any failure.**

Common-Cause Component Failures

- **Background**: The design approach to this point is based on the premise of single component failure occurrence only
- **But:** When similar components (of identical design and manufacture) are employed, they can fail as a group where they share a common defect or are put into a common environment → neither computer nor independent control can protect against
- How to address this CCF (Common Cause Failure) or CMF (Common Mode Failure)?
  - Use dissimilar components → diversification
  - (ex). Two dissimilar valve designs
  - (ex). Diversification of airbag procurement for a car maker
  - (ex). PCs with Windows, UNIX, and iOS
  - (ex). Desktops, Laptops, Tablets, etc

Achieving Fail-Safe by Disconnecting Effector Power

- **Effectors requires electric, hydraulic, or pneumatic power sources to function**
- **Simple disconnection of Power Source to the Effector**
  - **Commonly employed approach for achieving fail-safe in safety-critical system design**
  - **Robot Arm Case**

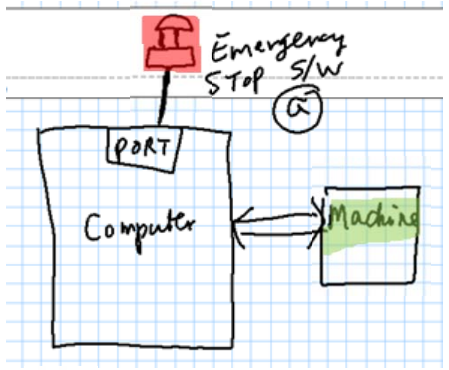


Example - Power Cutoff Approach

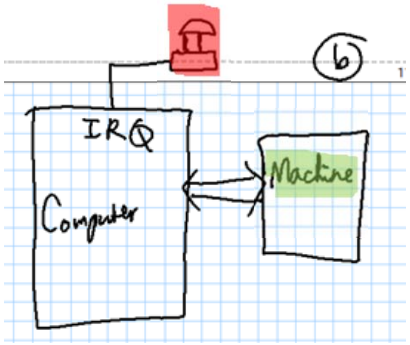
- Background:
  - Some aspects of designs may have safety implications which must be treated seriously to ensure that they operate correctly
- Design Focus:
  - How should an emergency stop button be interfaced to a microcomputer based machine control system to ensure its correct operation?

4 Design Approaches

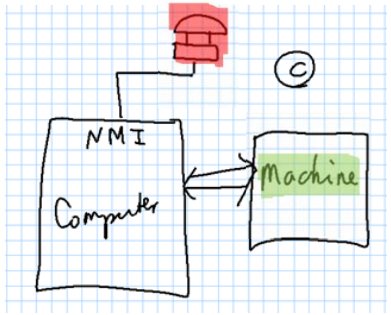
- (a) COM Port
  - Serial/Parallel input port to S/W
  - Poll periodically: sense and act



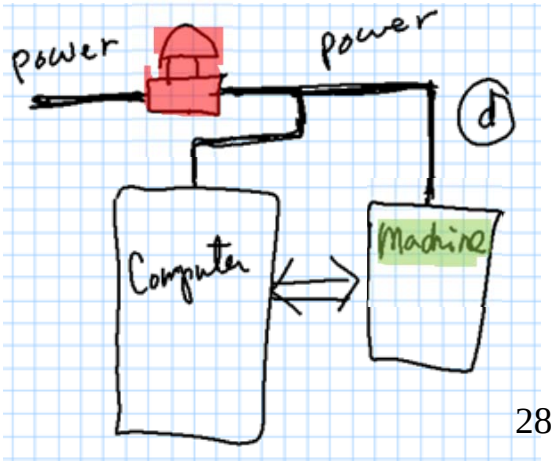
- (b) Interrupt
  - IRQ (Interrupt request) line to the S/W



- (c) Interrupt
  - NMI (non maskable interrupt) line to the S/W
  - IRQ always accepted

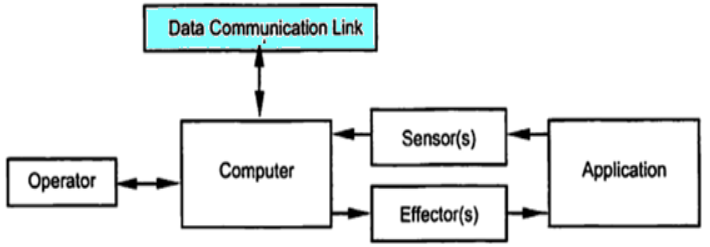


- **(d)Power Cutoff**
  - Main power supply line to switch operation
  - Safety function is provided by Power Switch



Data Communication Failure Detection

- Failure:
  - Corruption of transmitted/received information
  - Flipped state of a bit: 1 → 0; 0 → 1



- Failure Detection Techniques
  - **Parity generator/Checker**: addition of a parity bit so that the total number of "1" in a data is an odd or even number: odd or even parity
  - **Checksums**: Bytes check. Bytes in a data are summed and the summed value is transmitted
    - Example 1: Data {23, 16, 55} →Sum {23+16=55 = 94} → TX of Data + Sum {23, 16, 55, 94}→ RX of Sum of Data {23+16+55} against the last one in the Data {94}.
    - Example 2:

Character	Numerical Value	Transmitted Byte
e	01100101	01100101
x	01111000	01111000
a	01100001	01100001
m	01101101	01101101
p	01110000	01110000
l	01101100	01101100
e	01100101	01100101
Checksum	1011101100	11101100

Truncated

  - **Timeouts:** measure against No data – within a time window

Handling System Power/Interconnect Failures

- Electrical Power Source Failure

  UPS

  - Unable to command effectors to fail-safe state
  - So, effectors must be set and designed to go to a safe state when electrical power is lost
  - Normally Opened or Normally Closed valves
  - Normally Engaged Brakes (mechanical spring pressure against electrical current)
- Transient Electrical Power Failure
  - Affect computer function, effectors, and sensors
  - Power-up resent software must be designed to recognize the difference between normal power up and that following a transient failure
- Hydraulic Power Source Failure
  - Normally Closed Valve
- Pneumatic Power Source Failure
  - Normally Closed valve
- Power and Signal Interconnect Failure
  - Immediate functional failure
  - Detection and clearance of the first component/interconnect failure is essential

Prevention of Operator Failures

- Designing the Safe Operator Interface
  - No general rule for operator interface
  - Human being
- Monitoring Failures
  - **Perception:** Operator fails to perceive what compute system presents on Display → Audible Alarm may be necessary. Malfunction is most serious problem in causing intentional perception failures. (Ex) Fault indicators
  - **Cognition:** Operator fails to understand what computer system presented → Explanation S/W in operator's language
  - **Decision:** Operator mistakenly reacts after alerted and understood the problem presented by the computer system → Message with appropriate action and procedure suggested
- Failure to Follow Correct Operating Procedures: Measures
  - Automate the system as much as possible → limitation of computer
  - Validation S/W of operator action under given situation
  - Double verification system of an operator action which, if wrong, can lead to potentially dangerous actuator output

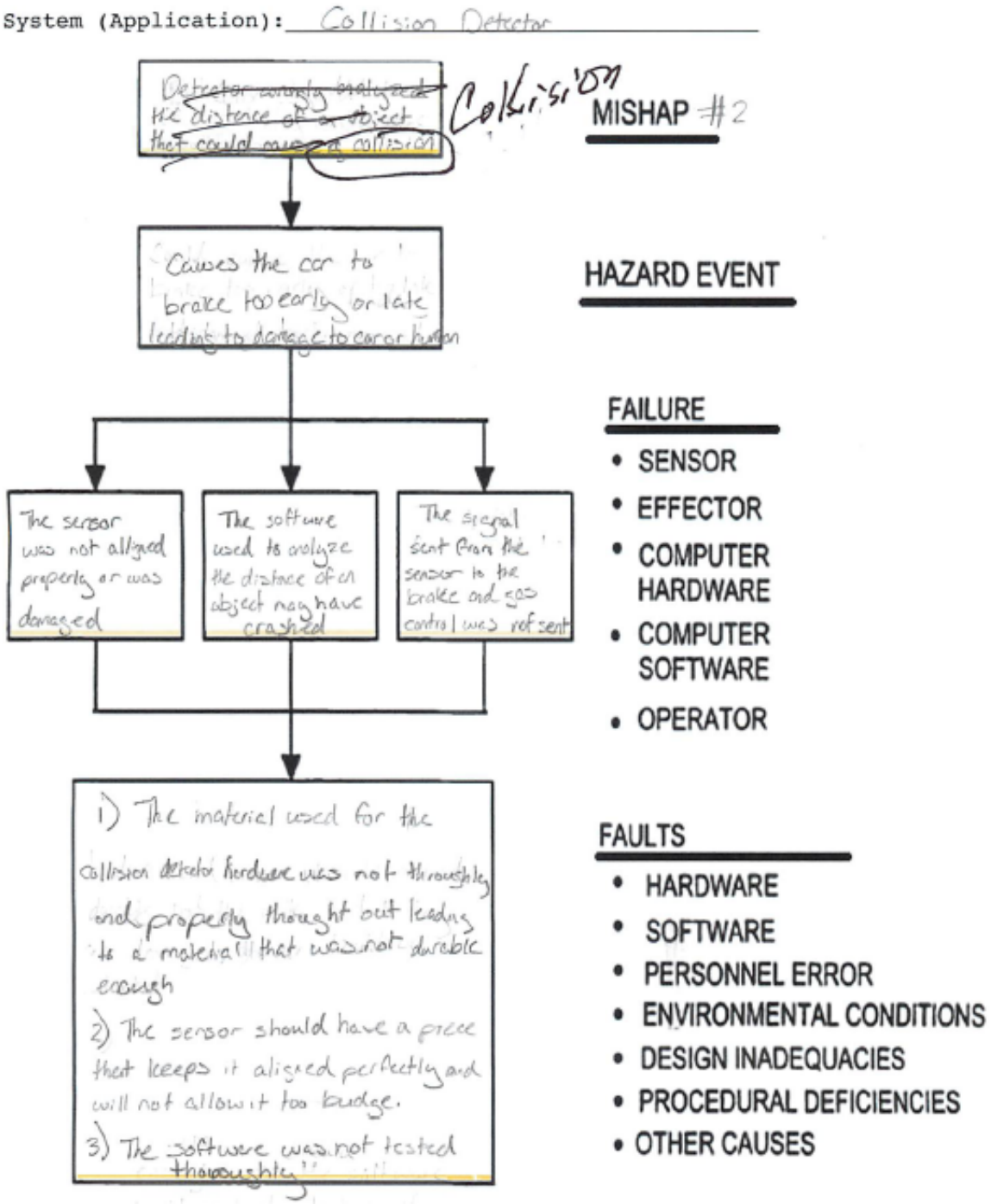
What have we covered so far?

- Simplex System
  - Fault detection and reconfiguration for a fail-safe state
    - Application failures
    - Sensor Failures
    - Actuator Failures
    - Power/Interconnect failures
    - Operator Failures
- Next step in simplex system
  - Computer hardware failure
  - Computer software failure
- Near Future:
  - Duplex/Redundancy System

Time to Apply Fail-Safe feature to Basic Computer System – Class Activity

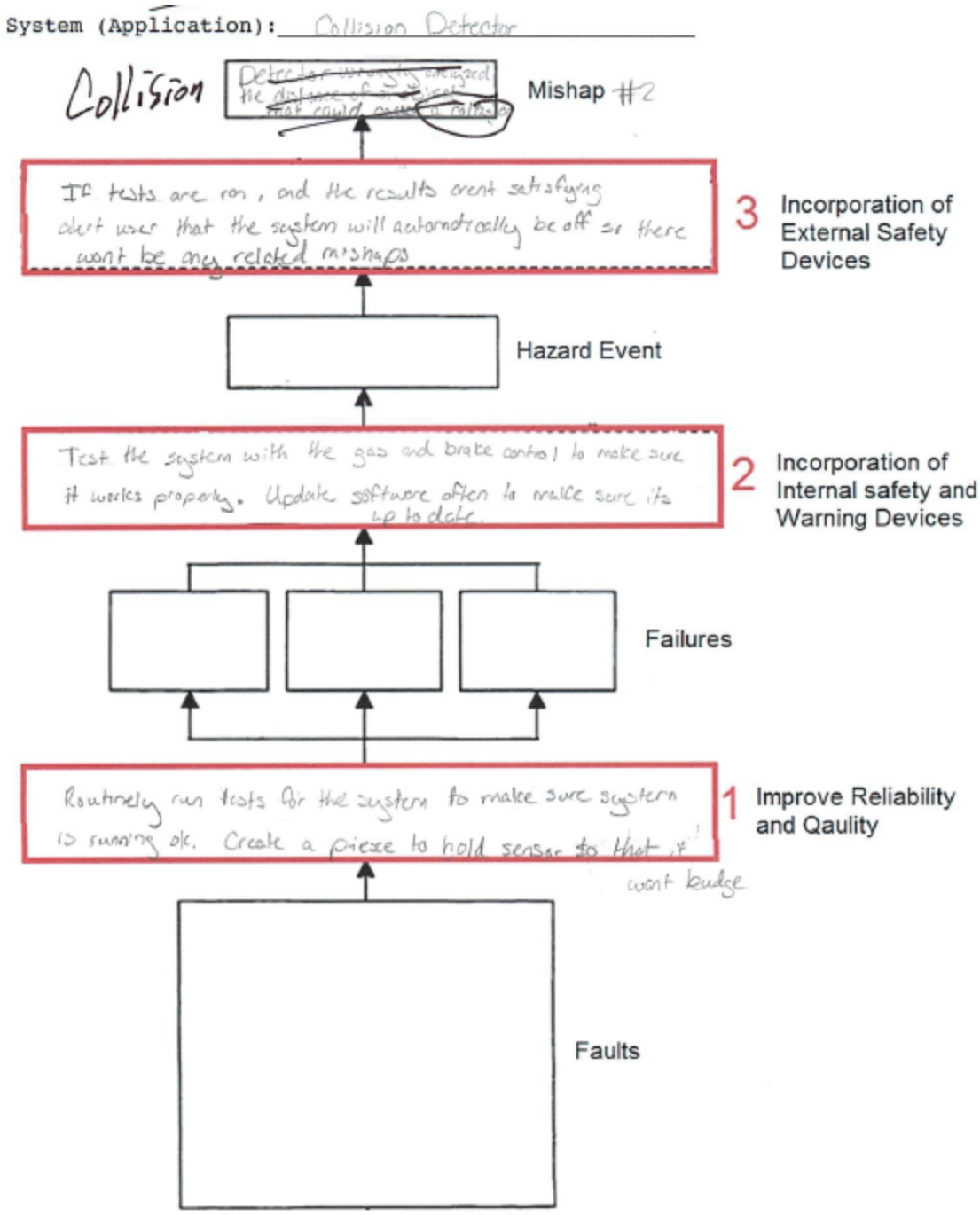
- 0 From your basic computer system for your _____ control system
- 1 Select a sensor (whose fault may lead to unsafe operation of the system)
  - Describe why/how the failure of the sensor leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe
- 2 Select an Effector /Actuator(whose fault may lead to unsafe operation of the system)
  - Describe why/how the failure of the actuator leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe
- 3 Check your 5-Step Design, and revise the steps required for Detection and Fail-Safe Reconfiguration

Hazard Analysis



Time to Add Fail-Safe feature to Basic Computer System

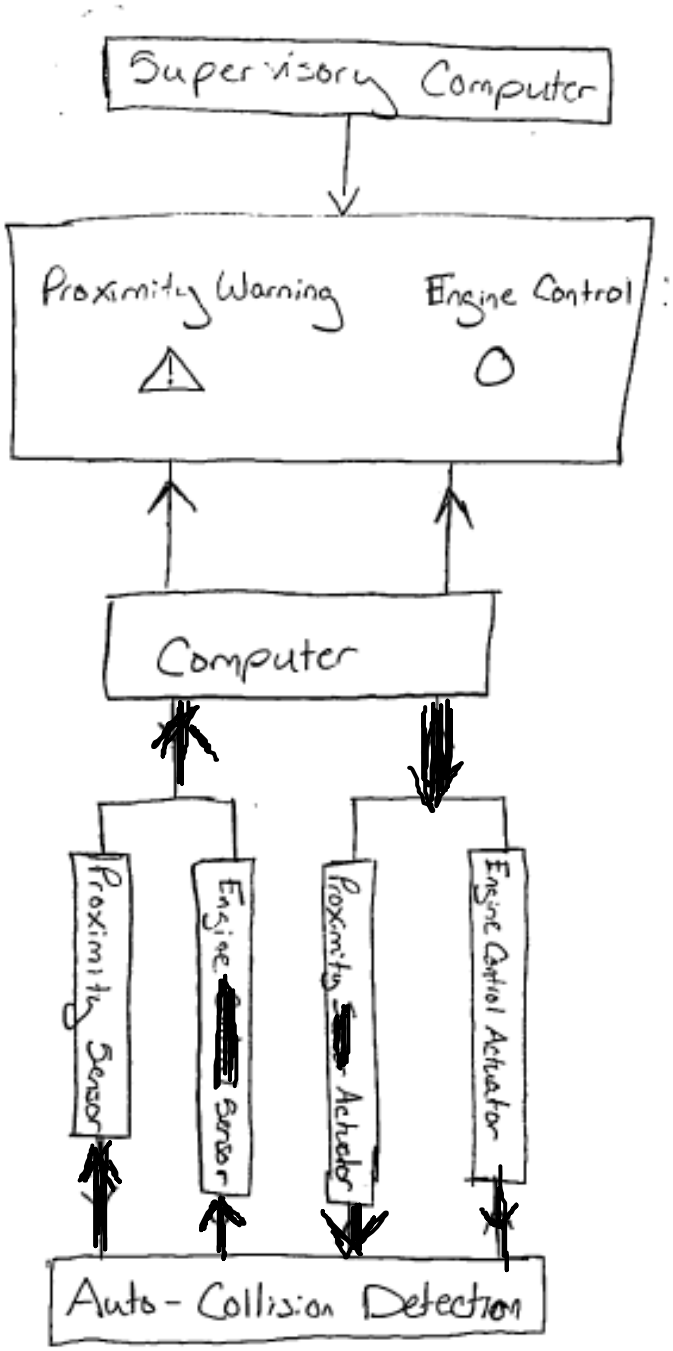
Detail 0



- **Mishap Mitigation**

Time to Add Fail-Safe feature to Basic Computer System

Detail 0



- Computer System Design – Step 1

Time to Add Fail-Safe feature to Basic Computer System

Detail 0

Step #2 Auto Collision Detection System

Proximity Warning

Sensor	Actuator	Warning Light
0	0	0
0	1	1
1	0	1
1	1	1

"0" – Means Off
"1" – Means On

Engine Control

Sensor	Actuator	Engine Control Light
0	0	0
0	1	1
1	0	1
1	1	1

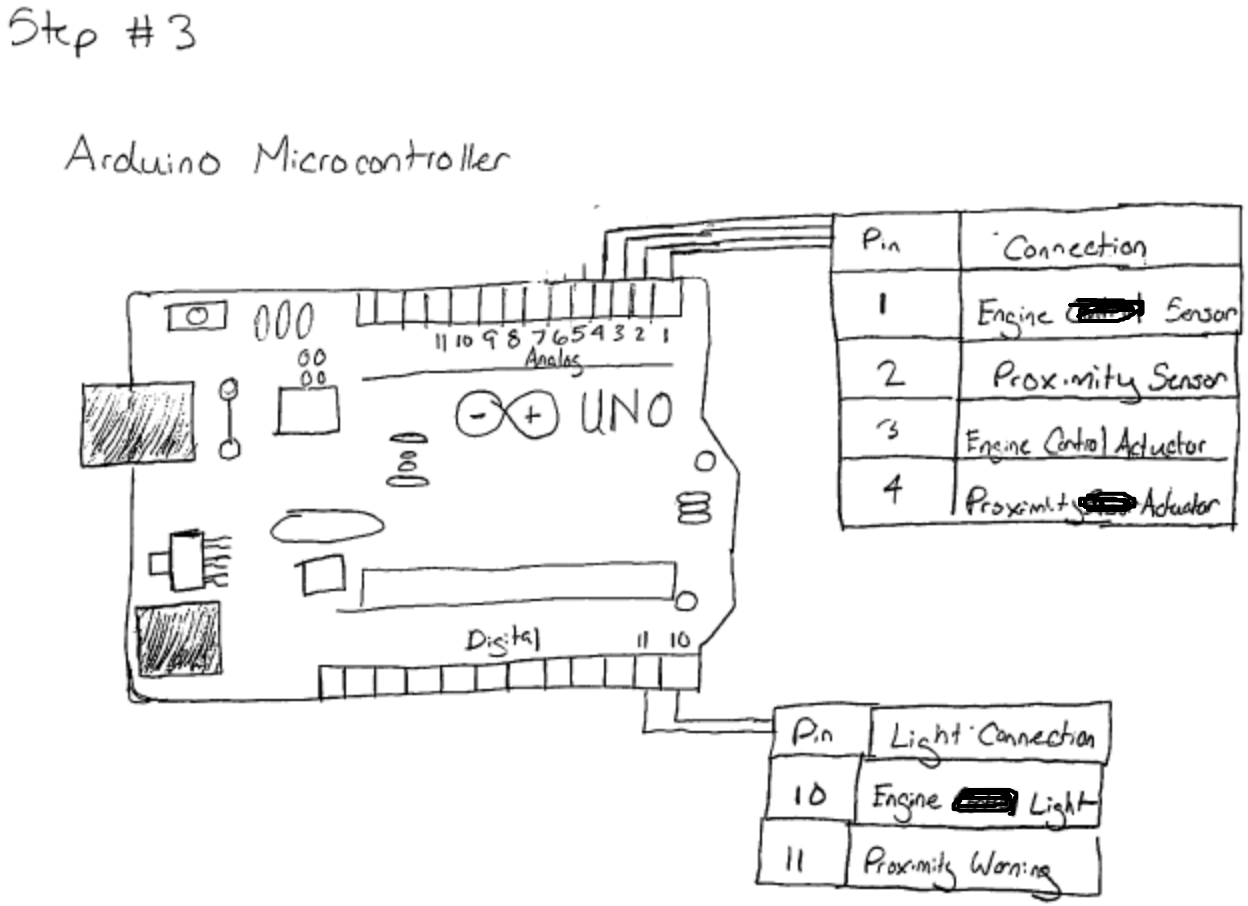
"0" – Means Off
"1" – Means On

- Computer System Design – Step 2

Time to Add Fail-Safe feature to Basic Computer System

Detail 0

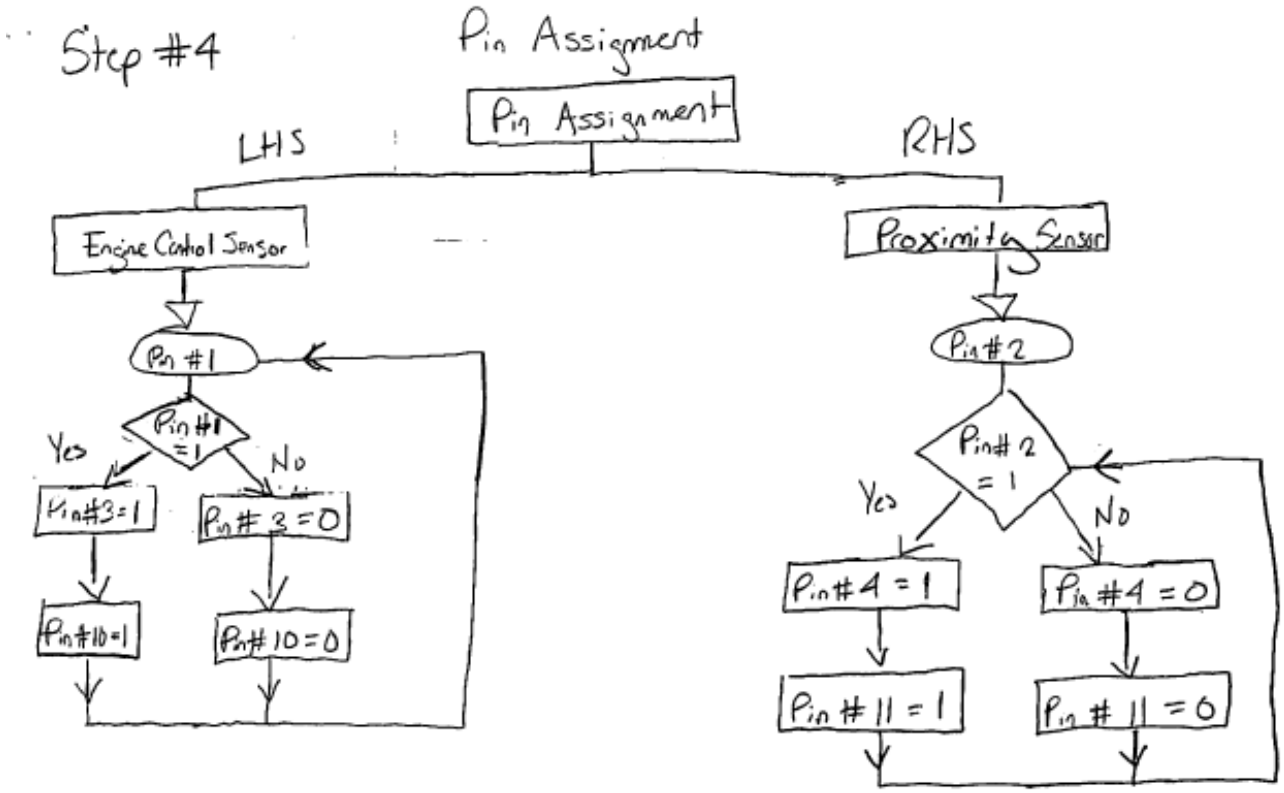
- Computer System Design – Step 3



Time to Add Fail-Safe feature to Basic Computer System

Detail 0

- Computer System Design – Step 4



Time to Add Fail-Safe feature to Basic Computer System

Detail 0

Step #5

1) Initialize Ports

Pin #1 = Engine [redacted] Sensor

Pin #2 = Proximity Sensor

Pin #3 = Engine Control Actuator

Pin #4 = Proximity [redacted] Actuator

Pin #10 = Engine [redacted] Light

Pin #11 = Proximity Warning Light

2)

```
If Pin #1 = 1
   Pin #3 = 1
   Pin #10 = 1
else
   Pin #3 = 0
   Pin #10 = 0
return Pin #3 & 10
```

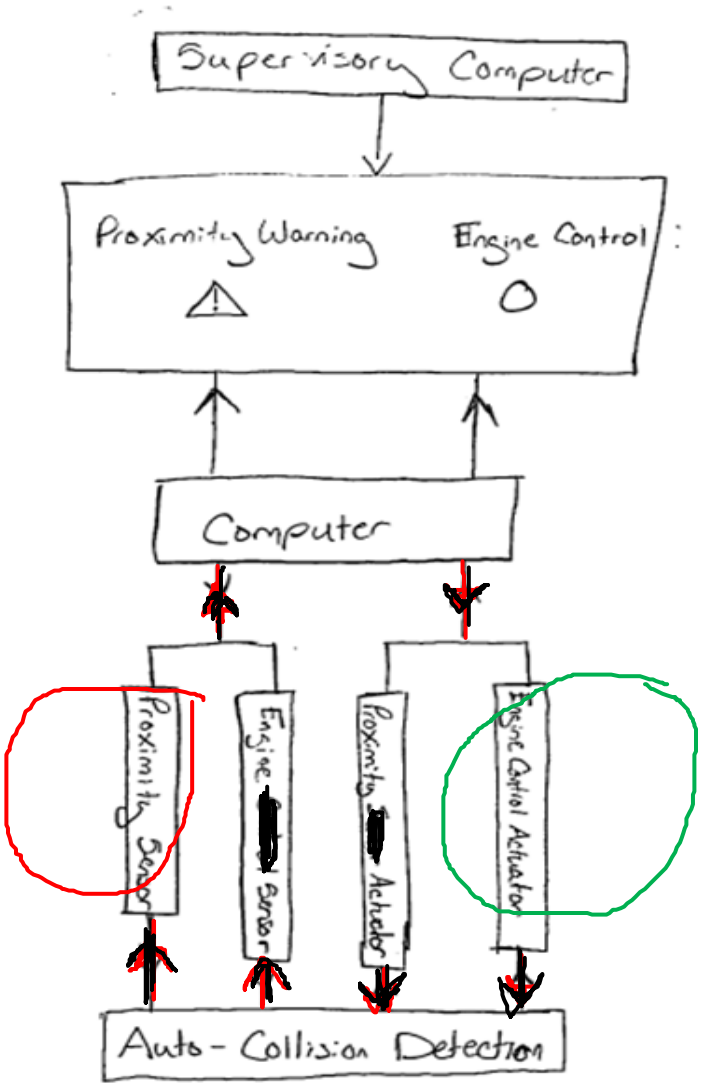
```
If Pin #2 = 1
   Pin #4 = 1
   Pin #11 = 1
else
   Pin #4 = 0
   Pin #11 = 0
return Pin #4 & 11
```

- Computer System Design – Step 5

Time to Add Fail-Safe feature to Basic Computer System

Detail 0

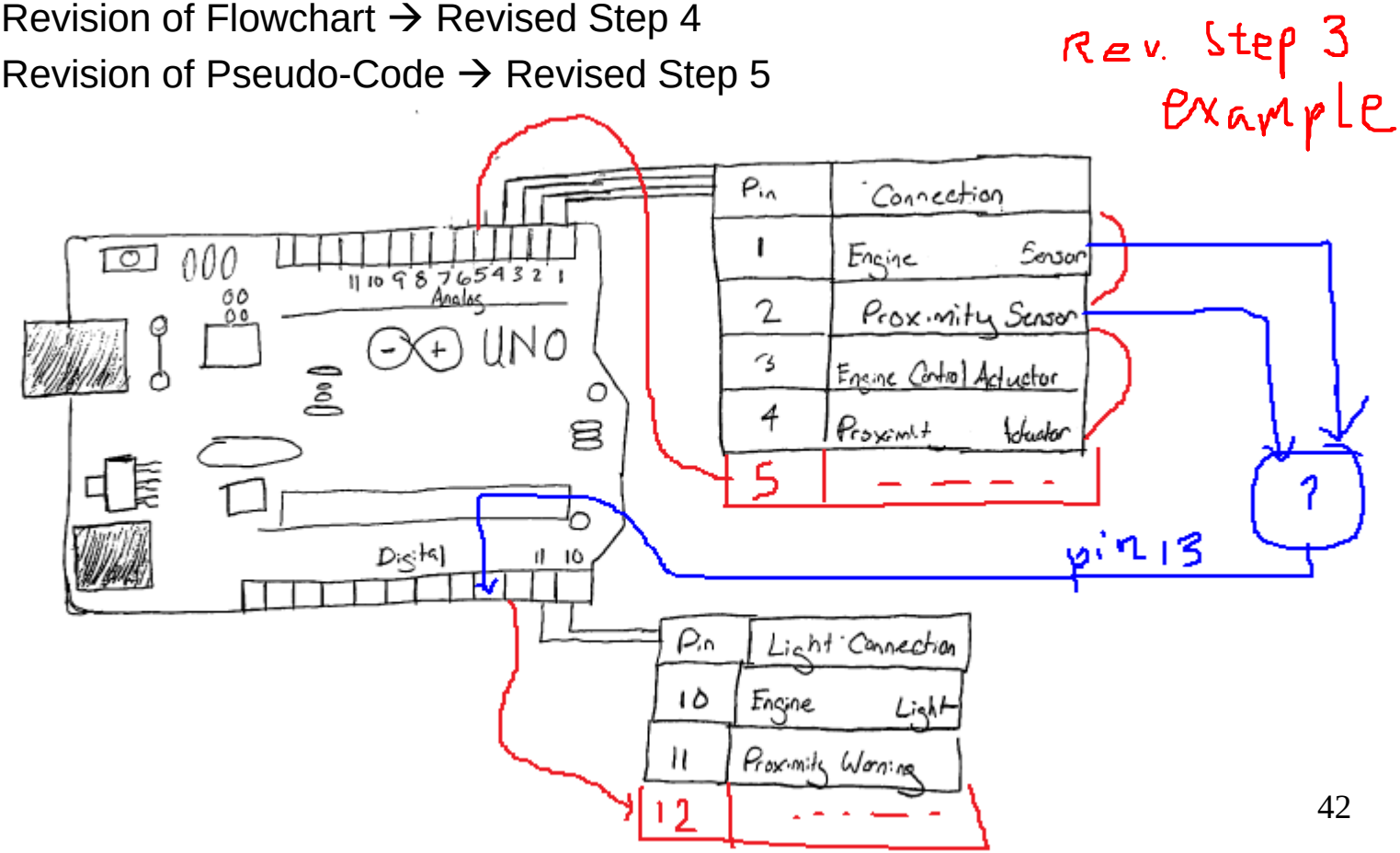
Stages 1 and 2 - Example



- 1 Select a sensor (whose fault may lead to unsafe operation of the system)
  - Describe why/how the failure of the sensor leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe
- 2 Select an Effector /Actuator(whose fault may lead to unsafe operation of the system)
  - Describe why/how the failure of the actuator leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe

Stage 3 - Example

- 3 Check your 5-Step Design, and revise the steps required for Detection and Fail-Safe Reconfiguration
  - Addition of sensors → Revised Step 1
  - Revision of S/W Requirement → Revised Step 2
  - Pin Assignment Change → Revised Step 3
  - Revision of Flowchart → Revised Step 4
  - Revision of Pseudo-Code → Revised Step 5



Class Activity for Fail-Safe Feature

- **Let's Start Now!**
- 0 From your basic computer system for your _____ control system
- 1 Select a sensor (whose fault may lead to unsafe operation of the system) – **SUBMISSION 1**
  - Describe why/how the failure of the sensor leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe
- 2 Select an Effector /Actuator(whose fault may lead to unsafe operation of the system) – **SUBMISSION 2**
  - Describe why/how the failure of the actuator leads to unsafe operation
  - Devise a detection system of the sensor in H/W design and S/W design
  - Devise a reconfiguration for fail-safe
- 3 Check your 5-Step Design, and revise the steps required for Detection and Fail-Safe Reconfiguration --- **SUBMISSION 3**

Activity Sheet 1

Computers and Safety-Critical Systems **Fall 2014**
Dr. Charles Kim

NAME:_____ ID:_____

Fail-Safe System Design Class Activity

1. Sensor Failure Detection and Reconfiguration

1a. Control system application:_____

1b. Sensor selected for failure detection:_____

1c. Description of why/how the failure of the sensor selected above leads to unsafe operation or state:

_____

_____

_____

_____

1d. Sensor failure detection method (with hardware and software):

1e. Reconfiguration method to a safe state.

Activity Sheet 2

Computers and Safety-Critical Systems **Fall 2014**
Dr. Charles Kim

NAME:_____ ID:_____

Fail-Safe System Design Class Activity

2. Actuator/Effector Failure Detection and Reconfiguration

2a. Control system application:_____

2b. Actuator/Effector selected for failure detection:_____

2c. Description of why/how the failure of the actuator selected above leads to unsafe operation or state:
_____
_____
_____
_____

2d. Actuator failure detection method (with hardware and software):

2e. Reconfiguration method to a safe state.

Activity Sheet 3

Computers and Safety-Critical Systems **Fall 2014**
Dr. Charles Kim

NAME:_____ ID:_____

Fail-Safe System Design Class Activity

3. Combination of Sensor and Actuator Failure Detection and Reconfiguration

3a. Control system application:_____

3b. Sensor selected for failure detection:_____

3c. Actuator selected for failure detection:_____

* Note: Attach the works of the following 5 steps to this sheet

3d. Revise your system design (Step 1)

3e. Revise your software requirement - truth table (Step 2)

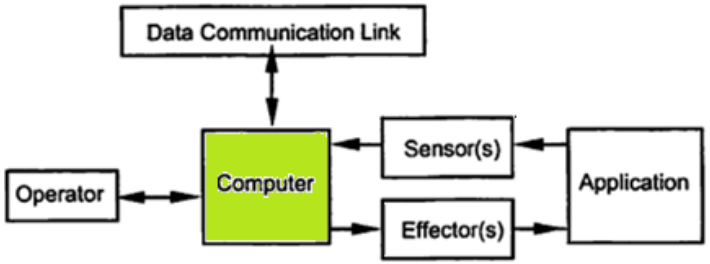
3f. Revise your programming model – pin/port assignment (Step 3)

3g. Redraw your flowchart (Step 4)

3h. Revise your pseudo-code (Step 5)

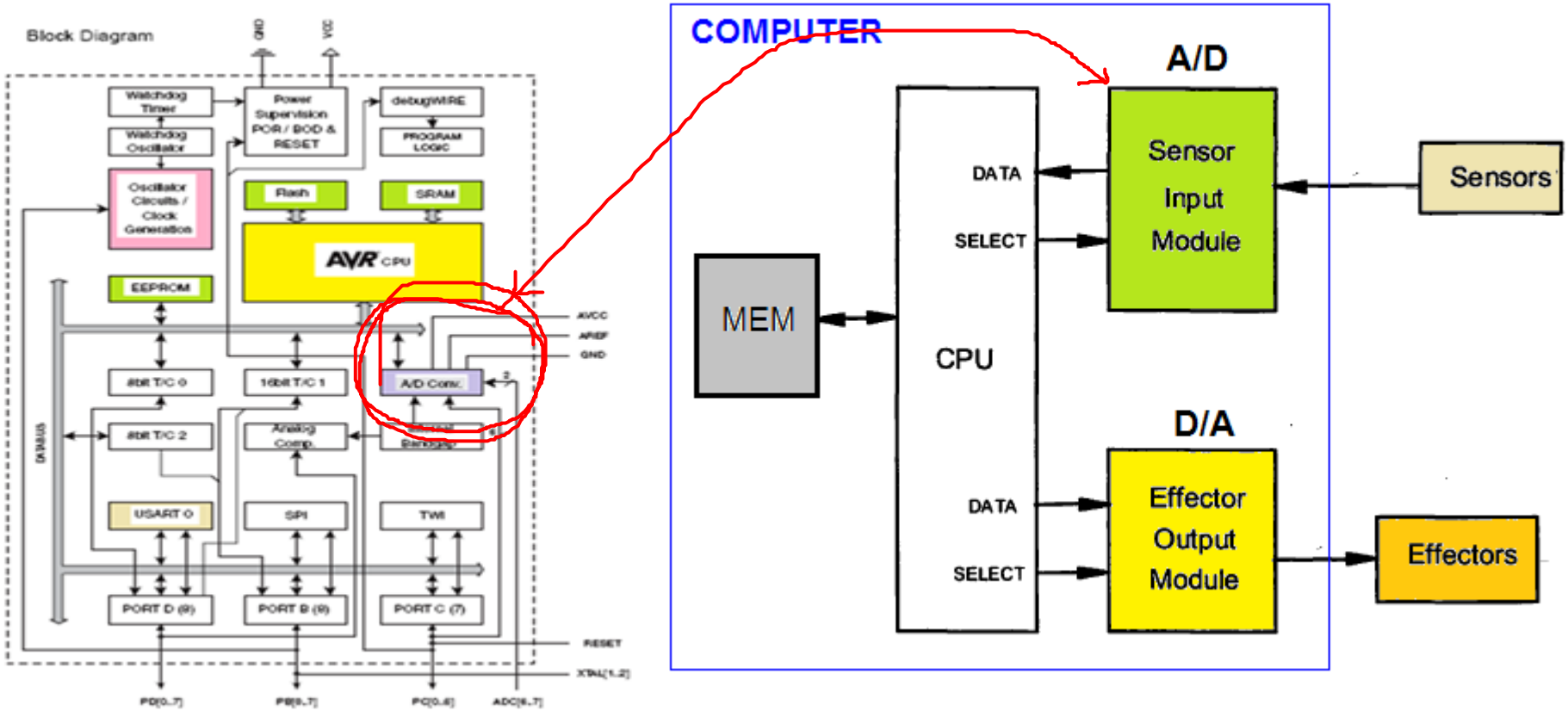
Next Stage

- So far we covered (for a simplex system)
  - Definition of fail-safe system
  - Inherent fail-safe system: lawn Mower Bar, Dead Man's Switch, etc
  - 2 essential components for a system to be fail-safe: **Fault Detection and Reconfiguration**
  - Sensor failure detection and reconfiguration
  - Actuator failure detection and reconfiguration
  - Data communication failure detection
  - Operator failure prevention
  - **Practice of Fail-Safe Design Involving Sensors and Actuators --- Class Activity**
- Next Step
  - Computer Failure Detection
    - Interface Hardware: Sensor Input Module, Actuator Output Module
    - CPU and Memory
    - Software
  - External Safety Devices and Controls
    - Safety Interlock
  - Summary for Simplex Fail-Safe System
  - Dual Redundant Architecture
  - Hardware and Software Reliability Improvement



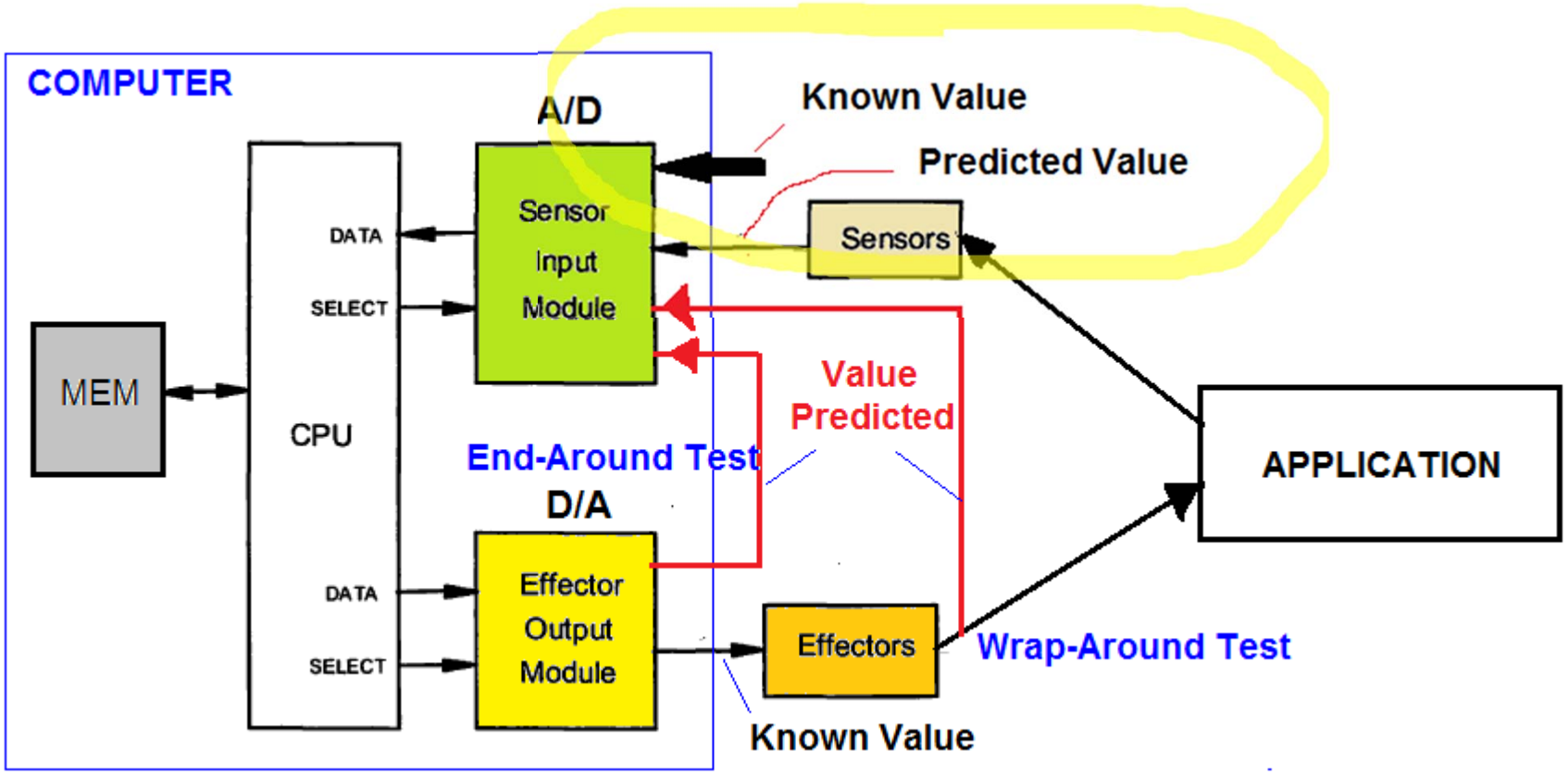
Computer Failure Detection – Interface Hardware

- Computer's internal hardware components such as Digital-to-Analog Converter (ADC) for **Actuator Output Module** or Analog-to-Digital Converter (DAC) for **Sensor Input Module**



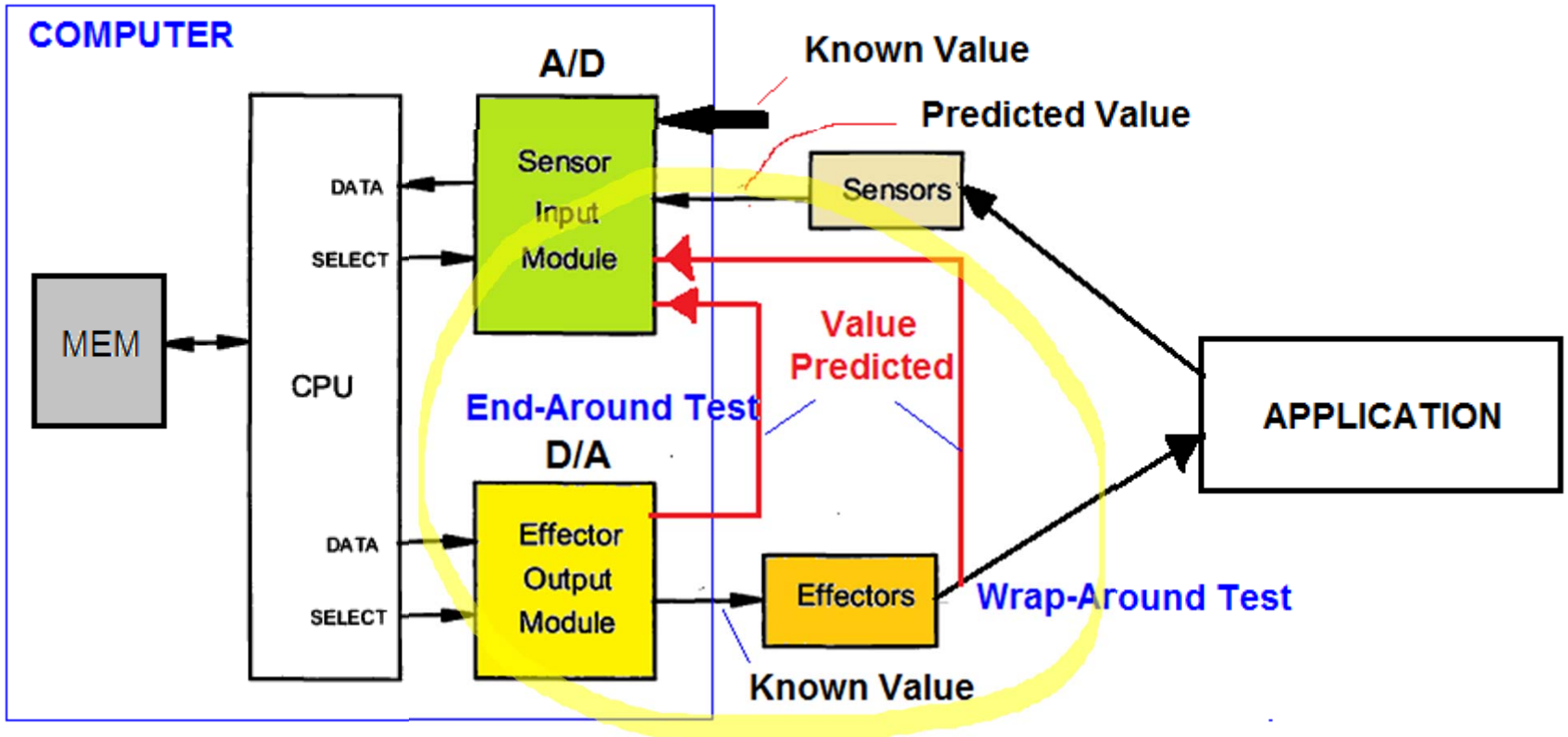
Detecting Sensor Input Module Failure

- **End-Around Test: Sensor Input Module Failure Detection** by putting known Value into a sensor input channel, and read the channel by S/W and compare them for match/mismatch



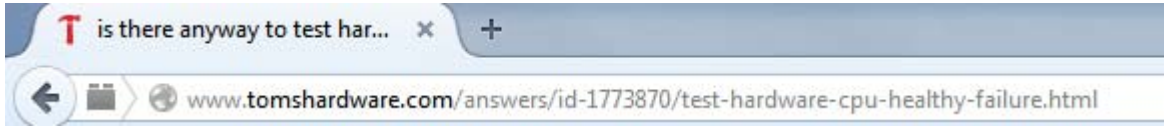
Detecting Sensor Input/Output Modules Failure

- Testing for Sensor Input (and **Effector Output**) modules
  - Output of the effector output module is wired back to a sensor input module (**End-Around Test**)
  - A known value is set out to the effector
  - Reading the effector output at the corresponding sensor input module (**Wrap-Around Test)**
  - This will detect **both** the sensor input module **and** effector output module failure



Computer Failure Detection – CPU and MEM

- Hardware failure causes software to cease functioning correctly
- Software **alone cannot** detect CPU and MEM hardware failures



spi_ajay August 20, 2013 4:44:05 AM

hello
i want to knwo if my pc's all components ram, graphics card, mother board, processor etc are working in good condition

Is there any software that can run diagnosie and tell me if some partially damaged component is exisitng even though its working , ty

Shaik Farooq CPUs August 20, 2013 4:50:12 AM

A bad computer motherboard or CPU can cause an assortment of different issues on your computer. Below are just a few of the possible issues you may encounter. It is important to remember that the issues below can also be caused by more than just a bad motherboard and CPU.

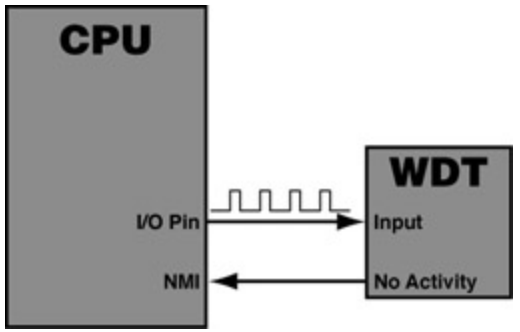
There are several different ways to test your computer's motherboard and CPU to determine if it's bad or has flaws that are causing issues with your computer. Below is a listing of these recommendations.

Software and Hardware solutions

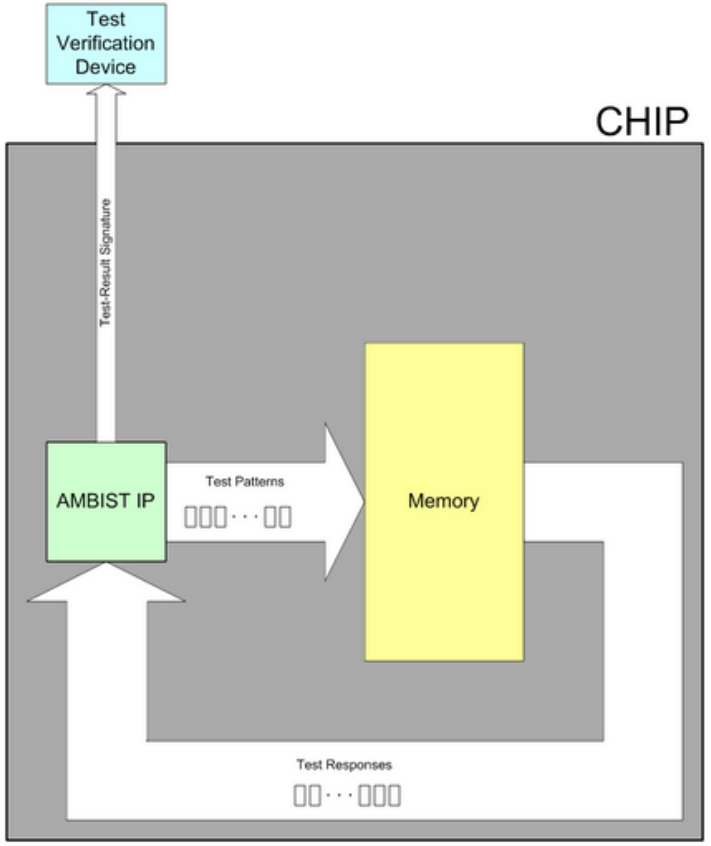
0

Computer Failure Detection – CPU and MEM

- Detection of “System Crash”: “CPU or MEM hardware failure leads to software malfunction or no-function” → Watchdog Timer Circuitry



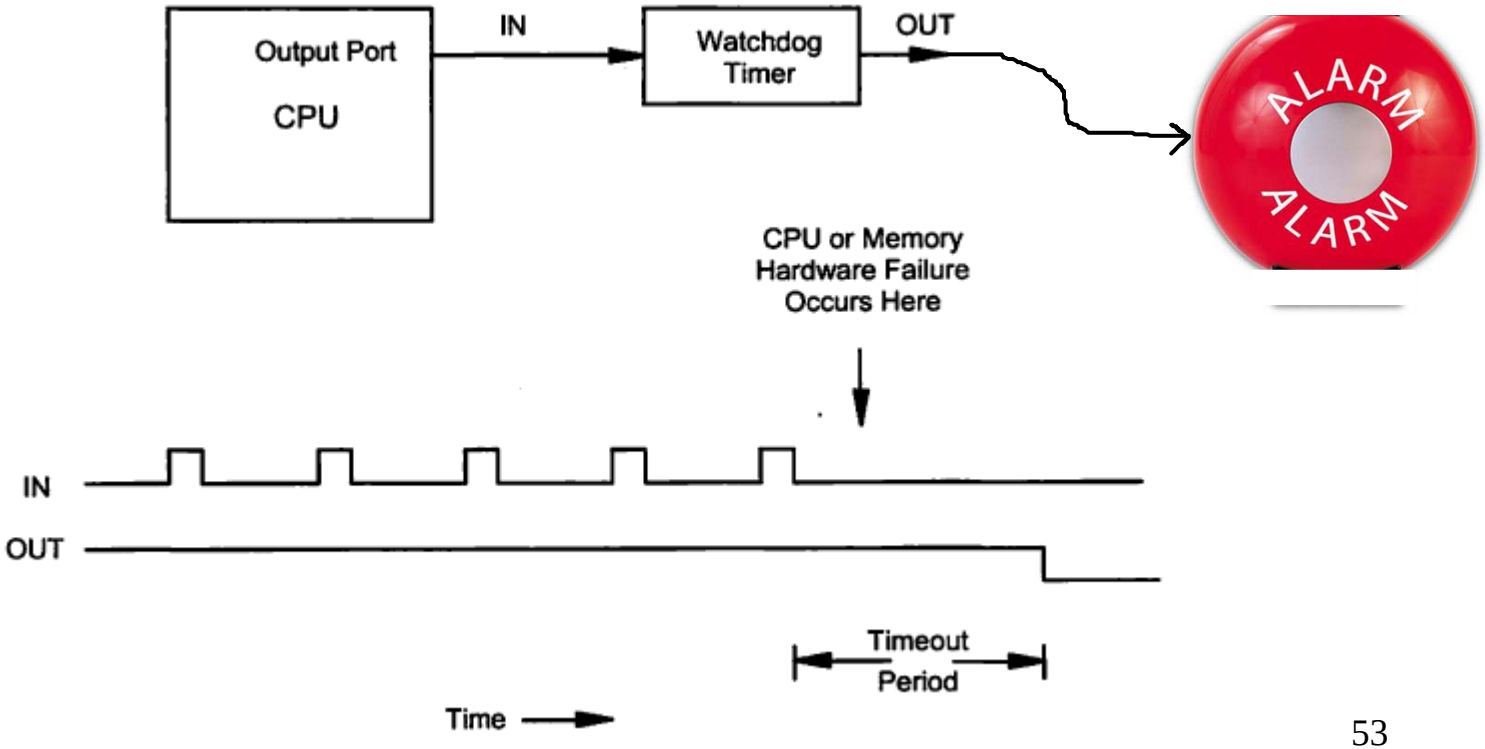
- Detection of “CPU and MEM failures, under which software still functions” → CPU/MEM Self-Test software program



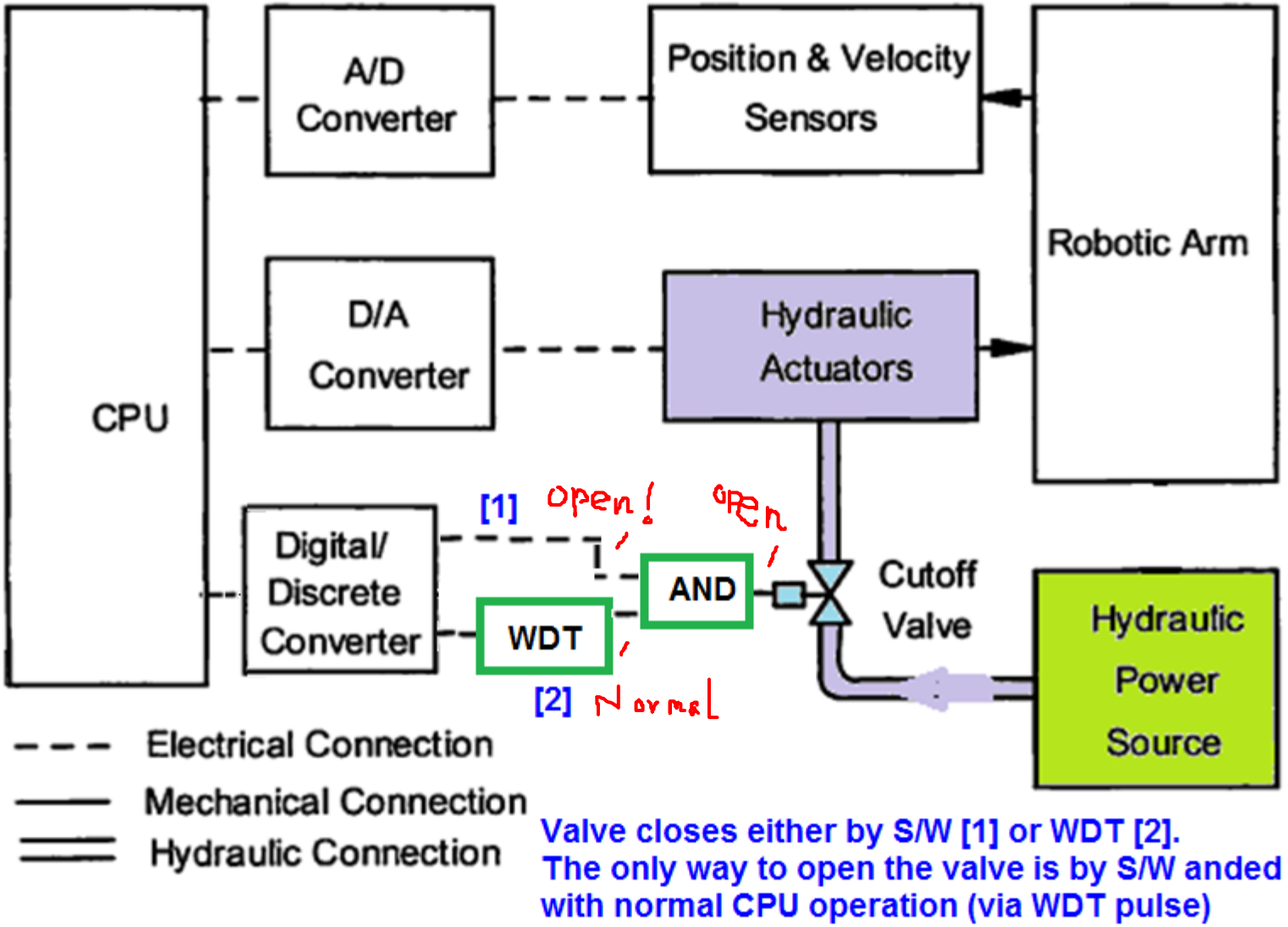
YFL elite Co., Ltd.
When best designs demand elite solutions.

Watchdog Timer

- Normally functioning software sends out a continuously varying signal to a special hardware circuit, Watchdog timer (WDT)
- The WDT sends out a discrete signal whose level is depending upon the running or non-running the software (Pulse Signal)
- The WDT output signal may be used to signal trip reconfiguration of the system to a safe state independent of CPU, or may be connected to a separate annunciator for the operator to know the situation.

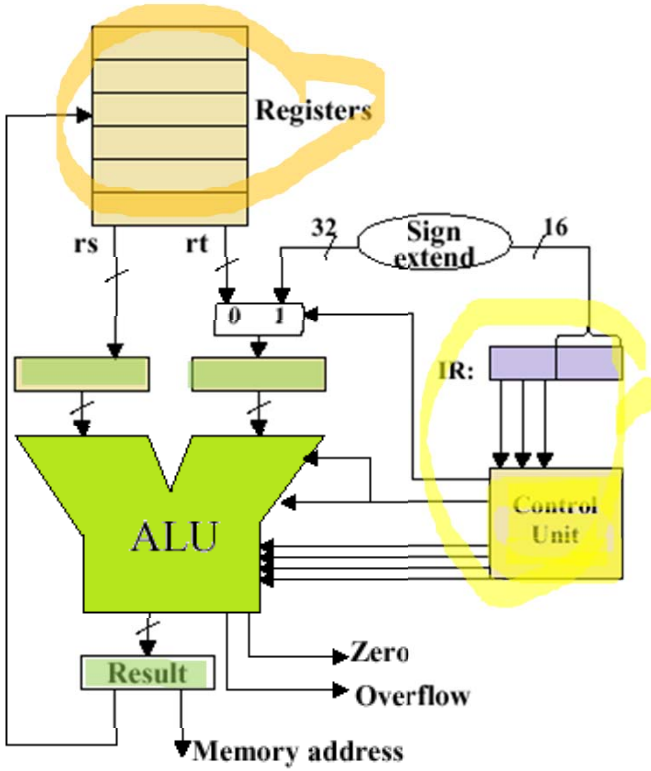


Watchdog Timer Example in Robotic Arm Controller



CPU Self-Test

- **CPU Self-Tet** supplements WDT
- Assumption: CPU may contain one or more faults but still executes S/W
- Objective: Detection of such faults before they surface as failures

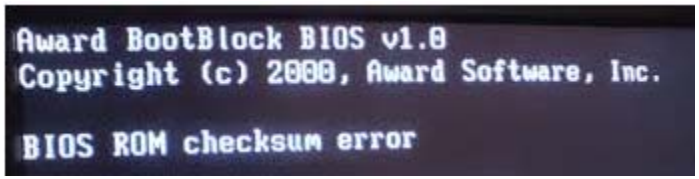


CPU Self-Test Examples

Failed CPU Component(s)	Name of Test	Test Description
ALU	ALU Test	Execute arithmetic or logical operations and compare computed results against known values.
Instruction decoder & pointer	None	Failure in this component means CPU test software probably won't run. Failure will probably result in WDT trip.
Accumulator & register(s)	Register Test	CPU registers are written with known data and read values compared.
Input and output port	End-around Test	Expected output in Input?
Memory data & address interface	None	Failure in this component means CPU test software probably won't run. Failure will probably result in WDT trip.

Memory Test

- Objective: Uncovering memory faults before they surface as memory failures
- Assumption: The memory test s/w is running with a correct instruction and data stream
- ROM test: memory blocks can be tested by calculating **checksums** (during the program development time), and continuously tested against the checksums in real time



- RAM test: **Reading** and **writing** each memory location with a "checkerboard" test pattern (e.g. 10101010 followed by a 01010101 to test stuck bits and stuck adjacent bits)

Clock Failure Detection

- Clock stoppage can be detected by WDT
- Clock variation results in **frame change**, which in turn alter control
- Use of timer to verify correct function of the resident computer clock

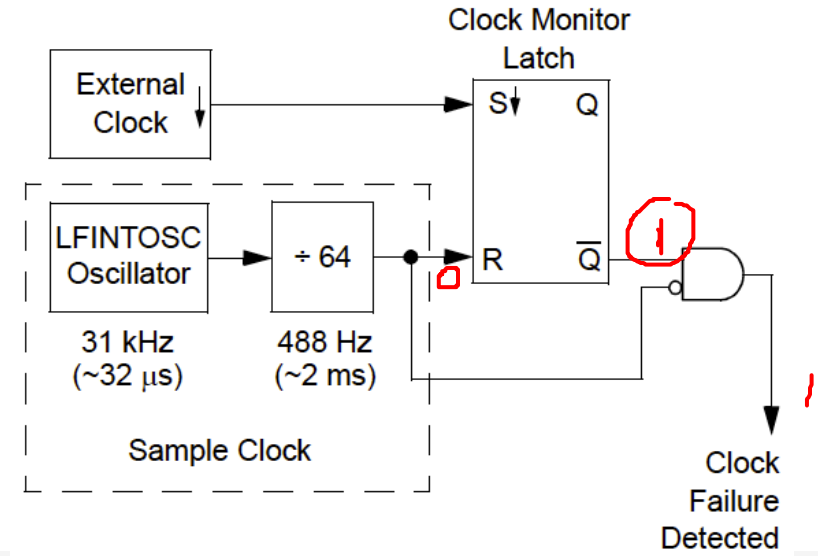
MICROCHIP

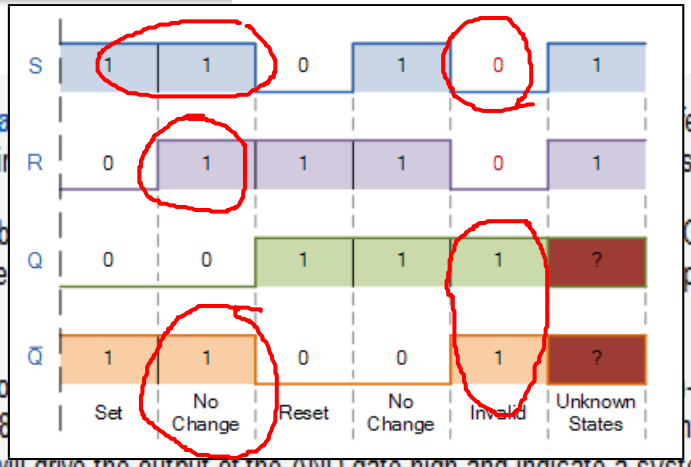
Developer Help

The right information, right now

Fail-Safe Clock Monitor

Fail-Safe Detection





The 31 Khz Interna e Clock monitoring. The INTOSC clock signal is sent through a divide by 64 circuit resulti s. This 488 Hz signal is the sample clock.

The system clock b On each pulse the Q output is set. The 488 Hz sample clock is sent to the Reset pin of the proper operating system, the Q and Q̄ outputs will toggle back and forth.

An AND gate with o -inverting input is connected to the Q̄ output and the inverting input is connected to the 48 g the Q̄ output will stay high when the 488 Hz signal also switches to a high state. This will drive the output of the AND gate high and indicate a system clock failure through the setting of the Oscillator Fail Interrupt Flag (OSFIF) bit. That system clock failure signal will trigger switch to the internal oscillator as the system clock.

Software Failure Detection

- There are software faults which might surface as failures in the operational environment
- Inadvertent entry of software failures that can crash the program may be detected by WDT
- Real problem is that a piece of code or routine which does not crash the system yet generates spurious, potentially hazardous output
- These types of faults are not easily detected and eliminated even in the final analysis step

Fault in Nuclear Software

Hitachi Finds Nuclear Software Fault; Undetected for 28 Years

By Shigeru Sato - April 10, 2008 22:57 EDT

April 11 (Bloomberg) -- Hitachi Ltd., Japan's third-largest builder of nuclear reactors, discovered a programming error in software used for almost three decades to measure the impact of earthquakes on pipes at atomic power stations.

The mistake, made by a Hitachi programmer, allows the software to underestimate the quake impact on steel pipes associated with eight nuclear reactors owned by six utilities, including Tokyo Electric Power Co., Hitachi spokesman Keisaku Shibatani said by telephone.

Confidence in the safety of Japan's nuclear power plants has been shaken after a 6.8-magnitude earthquake caused a fire and radiation leaks at a Tokyo Electric facility in Niigata prefecture last July. Twelve power producers, responding to a government request, revealed in March 2007 more than 300 cases of improper safety practices. Hitachi reported the software problem to the utilities this week, Shibatani said.

``It was a human error,'' he said. ``We're closely looking into this now.''

Recalls

Volvo Cars Recalled Following Software Bug Discovery

16 JUL 2012

Volvo Cars of North America, LLC, is reportedly recalling Volvo S80 vehicles with model years from 2011 to 2013. The cause of the recall is a software bug in the vehicle's computer causing the transmission to fail downshifting, which could lead to a fatal accident. Owners of said car will be notified or may call 1-800-458-1552. The computer repairs will be shouldered by the company.

In the automotive software industry, for example, software failure has led to expensive and embarrassing recalls. In May, 2008, auto manufacturer Chrysler recalled 24,461 Jeep Commanders, after it was found that embedded software could cause the engine to stall in some operating conditions.

Honda recalling 2.26M vehicles world-wide over automatic transmission failure

Posted by Vincent Van On August - 5 - 2011

Toyota Cites Brake Software Problems in New Prius Recall

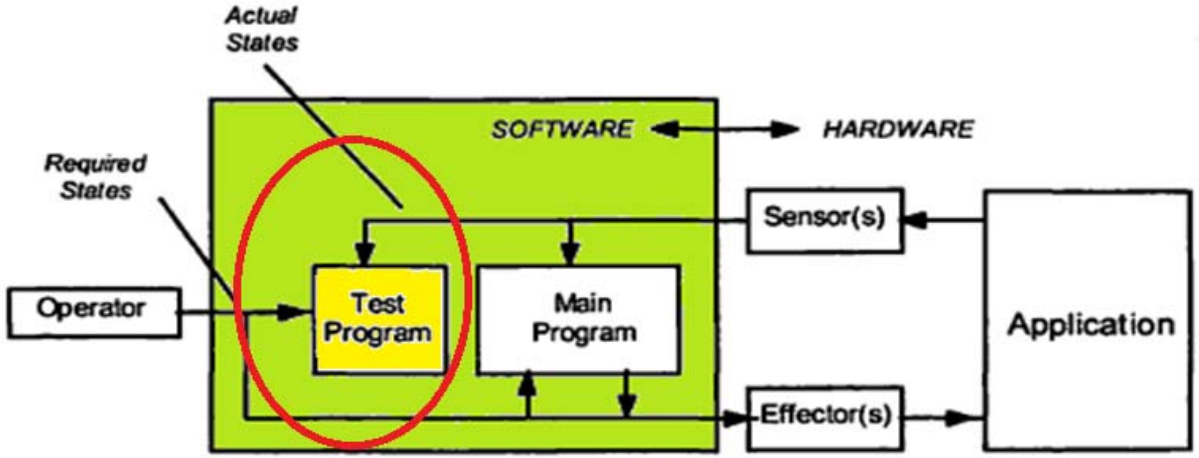
On Monday night, Toyota recalled its flagship high tech hybrid, the Prius, due to a brake software problem. The year that the company already wants to forget after unintented acceleration woes just got worse. Here are the details.

Quarter Of Medical Device Recalls Linked to Software Failures

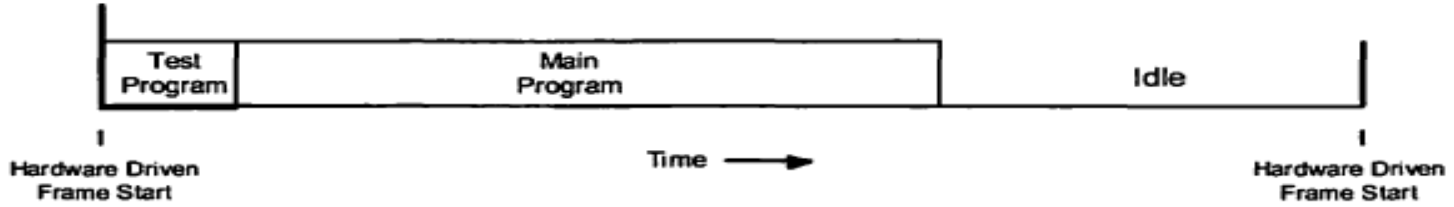
by Ryan L. Thompson on 07/11/2012

Software Failure Detection --- Detecting Incorrect Software Function

- The idea of using the software for detecting sensor failures (knowing in advance the expected sensor response) can be applied to detect software failure
- Build a simple, **stand-alone real-time software test system** that independently check the performance of the more complex functional software routine.



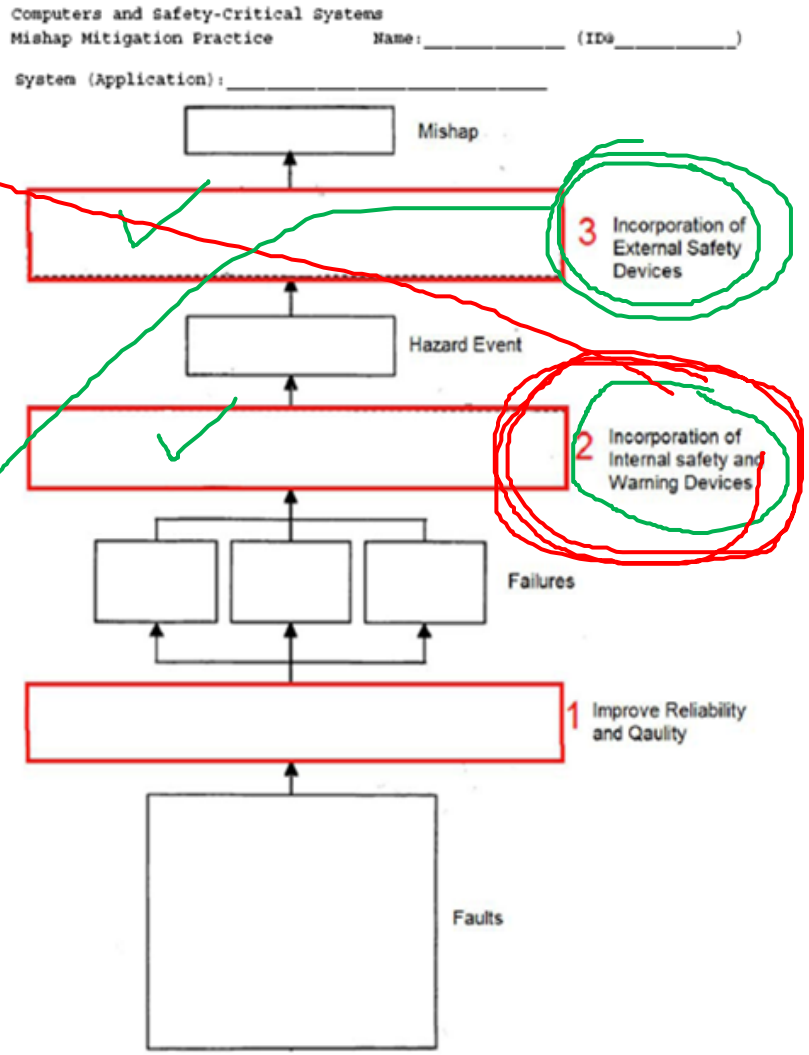
- Check if the actual and the required states match: Continued operation of this test software by hardware drive frame start (and does regularly)



So far

- Focus of Chapter 4
  - Incorporation of **internal** and **external safety devices** in to a basic computer system
    - **Simplex Systems**

- **Next Focus**
  - **External Safety and Controls**



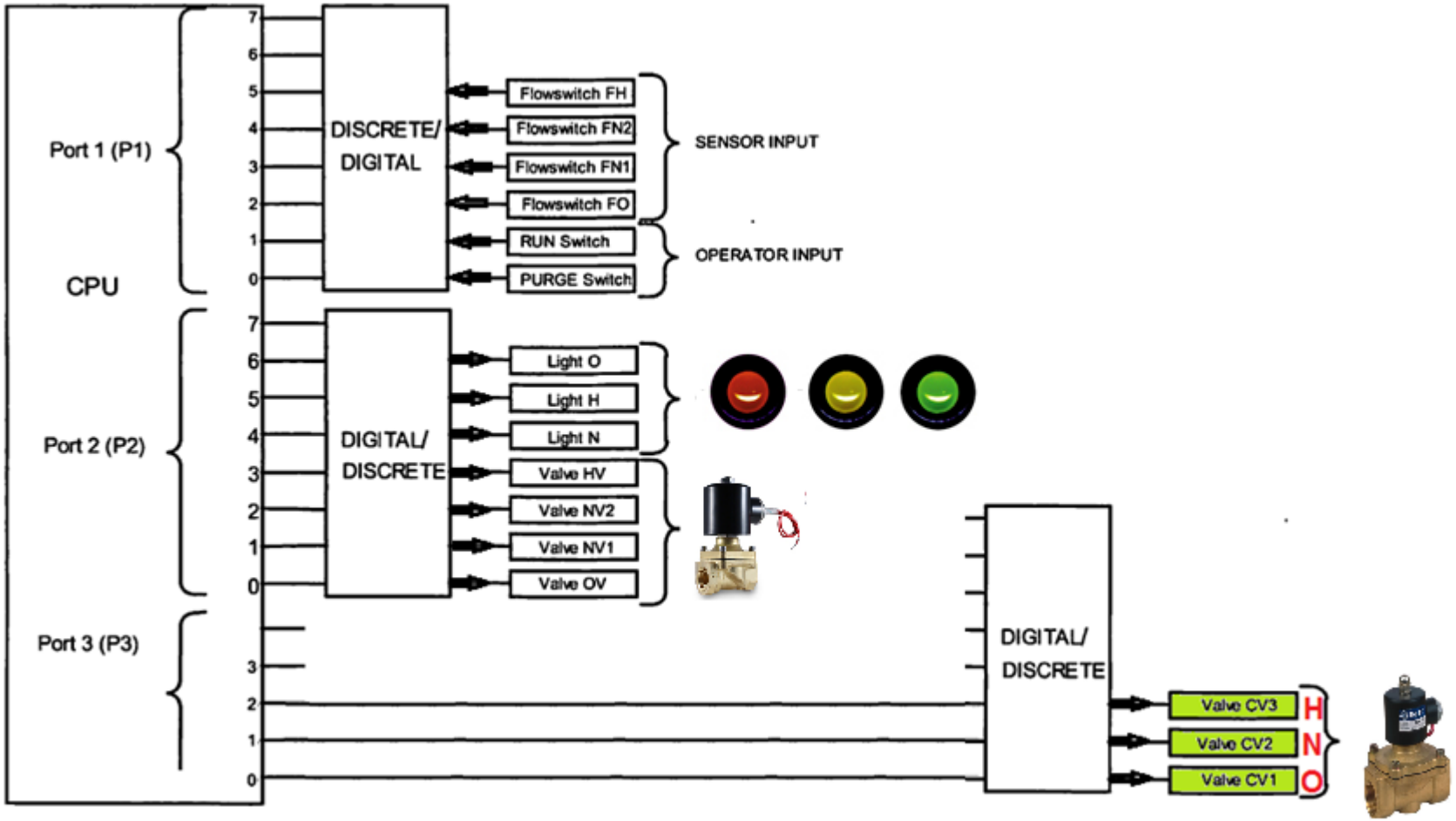
External Safety Devices and Controls

- These devices are added for further fail-safe in addition to the internal safety measures/devices
- These devices are placed outside of the computer control system
- These are to cover failures that elude the internal safety devices and to provide further risk reduction
- These must be independent from the internal safety devices and measures – should not be influenced by any failure in the system these devices are protecting
- Use of different design approach and dissimilar technology than those employed in the primary system --- to maintain the independence
- Use of top-down approach: Devices/Controls are applied at the Hazard level

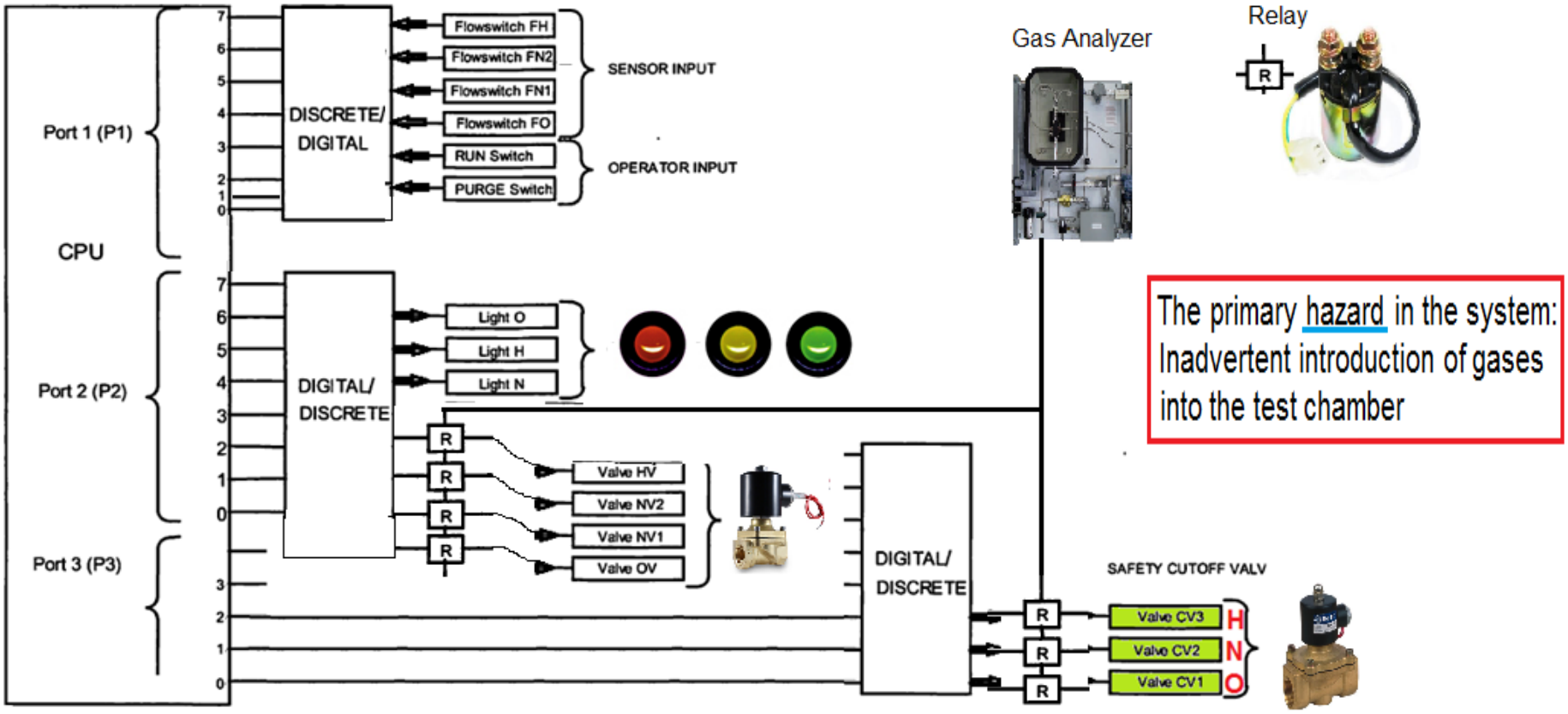
Example with Test Jet Engine Propellant Supply

- The primary hazard in the system: **Inadvertent introduction of gases into the test chamber**
- Introduction of a gas analyzer in the test chamber, connected to a set of relays
- The NO relays are connected between the solenoid valves and the digital/discrete outputs
- Operation
  - **IF** gas analyzer does not detect gas **THEN** activate the relay so that the digital output is connected to the valve coils → valves can be opened or closed
  - **IF** gas analyzer detects gas **THEN** deactivate the relay so that the digital output is disconnected to the valve coils → valves are automatically closed (NC)

Example with Test Jet Engine Propellant Supply - Before



Example with Test Jet Engine Propellant Supply - After



The primary hazard in the system: Inadvertent introduction of gases into the test chamber

- Operation
  - Gas analyzer does not detect gas – activate the relay so that the digital output is connected to the valve coils → valves can be opened or closed
  - Gas analyzer detects gas – deactivate the relay so that the digital output is disconnected to the valve coils → valves are automatically closed (NC)

External Safety Device Hierarchy

- From simplest to most dependable approach possible
- Choice should starts from the simplest to the most complex (only when the candidate control method is impossible or impractical to implement)
- Multiple layers of external safety devices, when feasible, are desirable

Independent Control Method	Characteristics	Examples
Physical barrier	Has no significant moving part.	Physical stop, pressure rupture disk, rubber bumper.
Mechanical device	Simple passive (unpowered) mechanism.	Pressure relief valve, spring and shock absorber, check valve.
Electromechanical power cutoff device	Physically actuated electrical switches.	Pressure switch, limit switch, proximity switch.
Analog or simple digital system	Simple electronic circuitry and I/O. Power supply required.	Gas detector, radiation detector.
Computer safety system	Complex electronic circuitry and I/O. Power supplies required.	Complex safety logic and/or calculations.

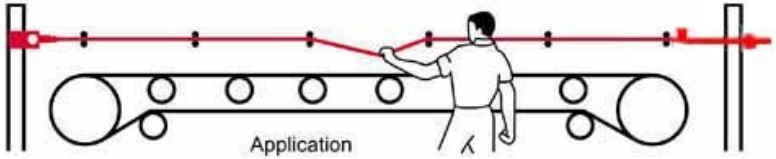
External Safety Device – Emergency Stop Circuits

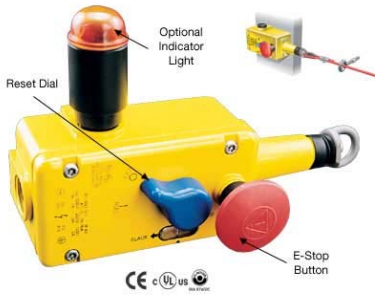
- The system operator is supplied with an **emergency stop button** or switch
- Emergency stop **immediately activates an independent system** which brings the system to a safe state
- Emergency Stop circuit must be independent and be not affected by the system
- Emergency Stop **may withdraw** electrical, hydraulic, and/or pneumatic power sources
- Multiple Emergency Stop Buttons may be wired in series in multiple operator stations so that any one button will safely shut the system down.

Cable and Push-Button E-Stop Assembly

Emergency-stop button.

Cable-tension indicator.





External Safety Device – Safety Interlocks

- "Safety Interlocks": A hard-wired device to inhibit actuator motion when external conditions make actuator motion unsafe.

www.microtechfactoryservice.com/how_work.html

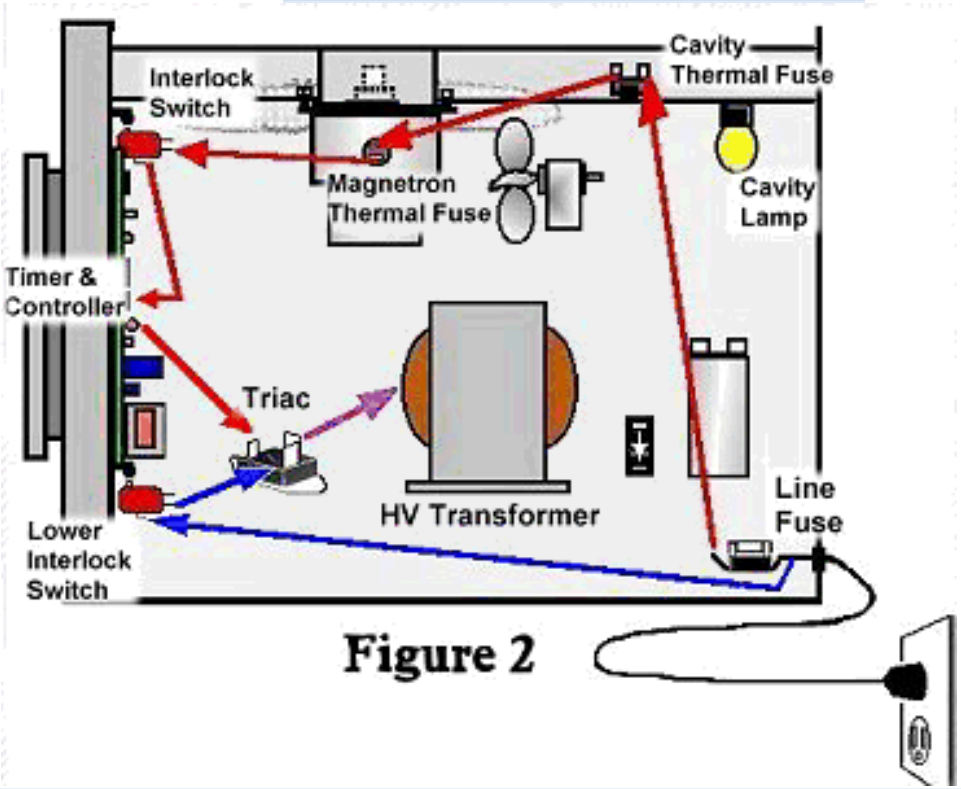


Figure 2

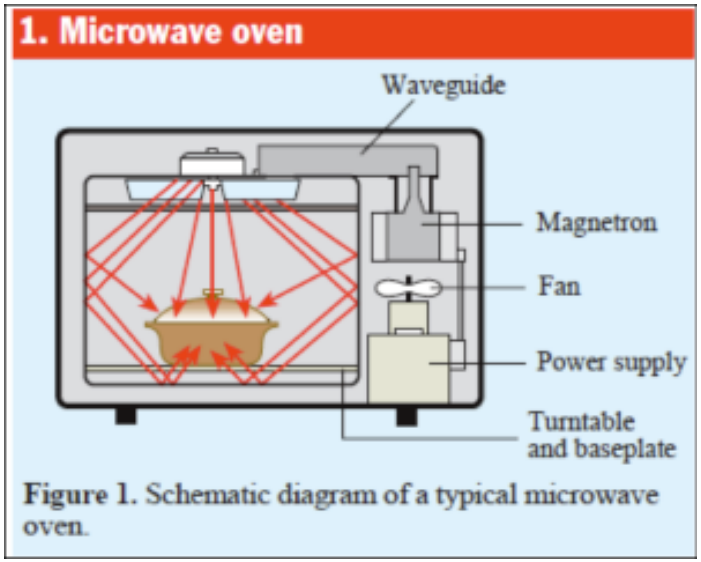
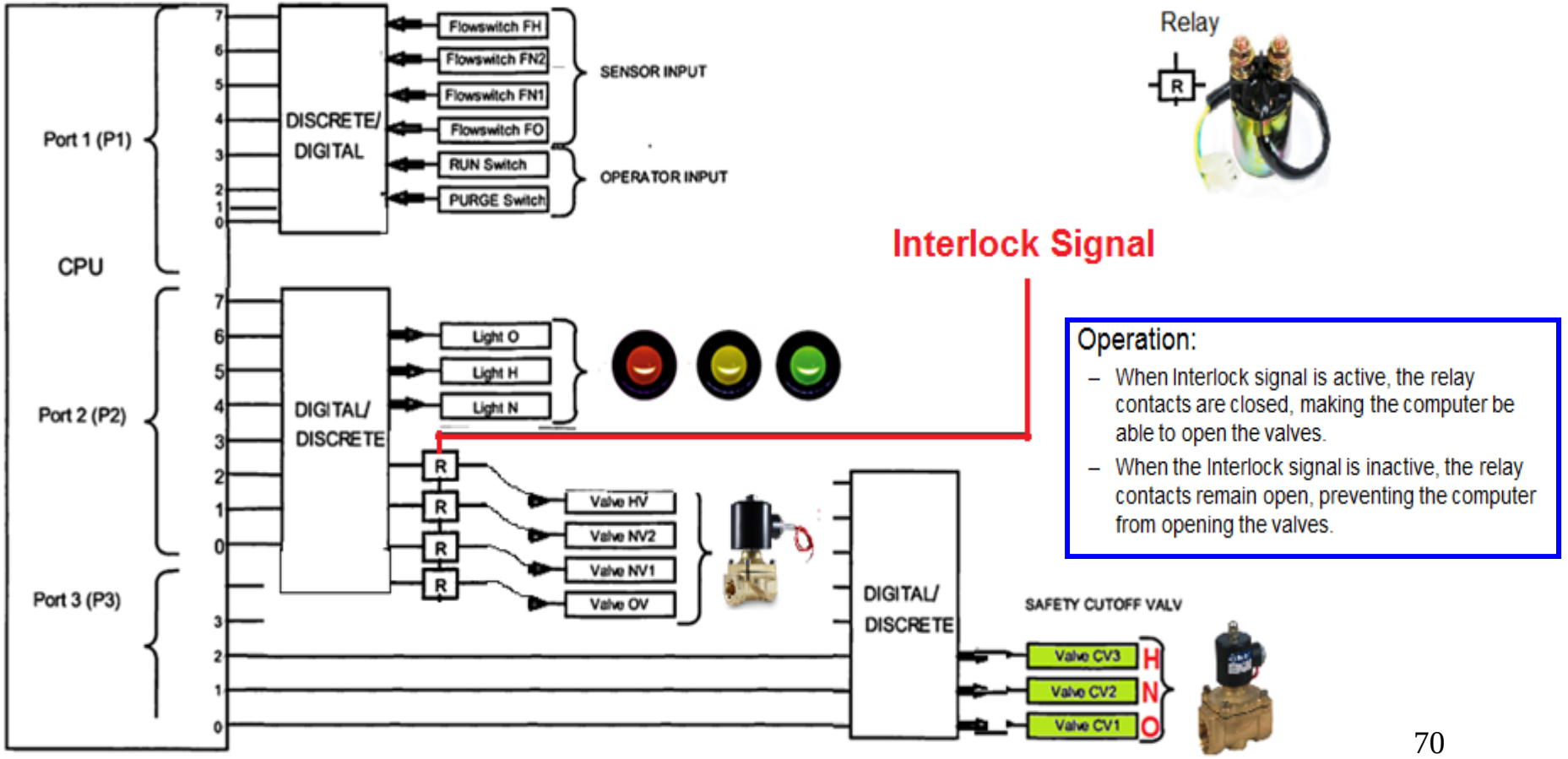


Figure 1. Schematic diagram of a typical microwave oven.

commonsensescience.wordpress.com/2011/11/16/easy-mac-spaghettios-and-left-overs-what-would-we-do-without-microv

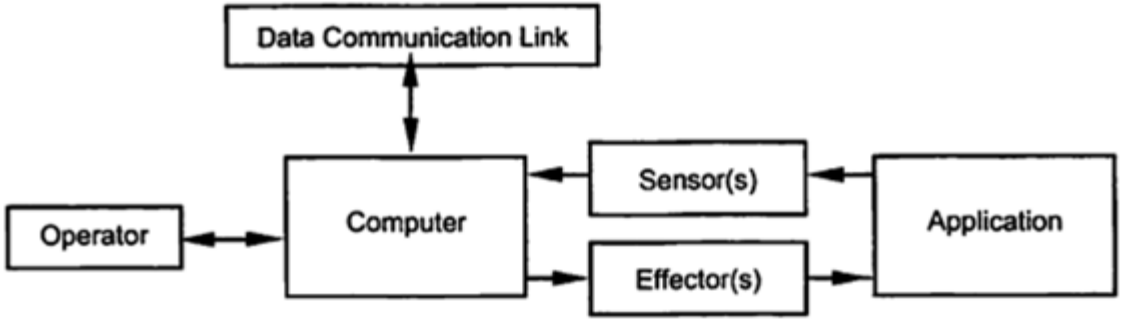
Safety Interlock Example

- Test Jet Engine Propellant System
- The signal from the engine control may inhibit the propellant software from opening the H2 and O2 valves given an incorrect RUN command from operator. NO relays are employed

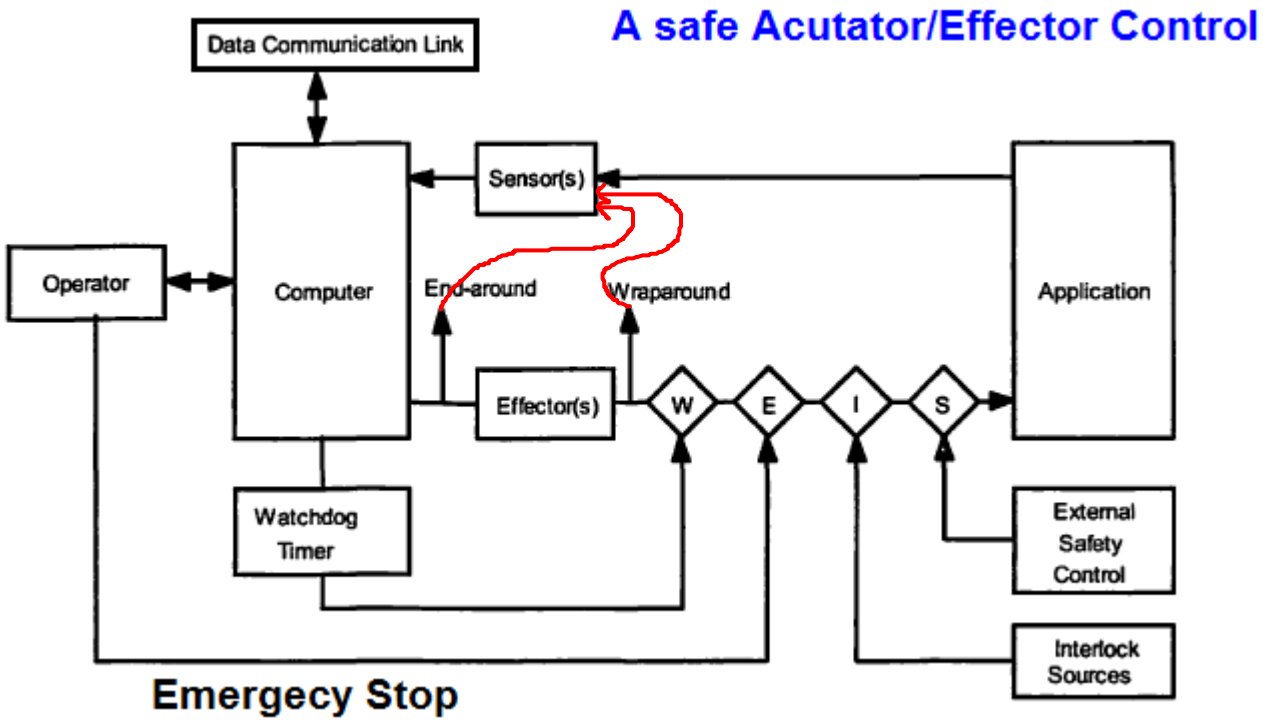


Summary --- Simplex Fail-Safe System

- Before



- After



Summary --- Simplex Fail-Safe System (S/W Functions)

Software Function	Summary Description
Application failure detection	Compare application parameters to normal values.
Sensor failure detection	Compare actual sensor value to that based on predictable discrete states. For analog and digital signals, compare actual value to that based on state estimator, reasonableness tests, informational redundancy, dependent sensor values, or analytical redundancy. This routine will also detect failures in resident software.
Effector failure detection	Wraparound test. Compare actual effector value to commanded value.
Write operator diagnostics	Detected faults/failures and associated software action is presented to operator output device.
Operator input check	Crosscheck against application variable limits and system operating modes. Verify potentially hazardous commands.
System power failure detection	Compare electrical, hydraulic, pneumatic power levels to normal range of values. (Note: system components should automatically transition to fail-safe positions on power loss.)
Communications failure detection	Detect failure based on parity or block checksum error or exceeding of timeout limit.

Summary --- Simplex Fail-Safe System (S/W Functions)

Software Function	Summary Description
CPU failure detection	Watchdog timer refresh. CPU self-tests. CPU hardware diagnostics.
Memory failure detection	Checksums. "Checkerboard" tests. Parity check (if implemented).
Sensor input module failure detection	Compare constant input to known value. Compare end-around received value to generated value.
Effector output module failure detection	End-around test. Compare received value to generated value.