Course Introduction

Hardware Software CoDesign

September 2011

Agenda

Course Introduction

1. Basic Themes that came out through the Historical Background

2. Basic Pitfalls – what to guard against

3. Problem Statement of a touch screen system

Course Introduction

Basic Themes

1. **Modeling** :: Because parallelism is important, the choice of an appropriate modeling paradigm is also important. Different modeling formalisms need to be developed that capture the various aspects of system behavior.

2. **Analysis and Estimation** :: Different models of the same system behavior need to be analyzed and estimated for performance. Performance would include tradeoffs for Area, Speed, Power. Cost to build, Time to Market are other factors.

3. **System-level partitioning, synthesis and interfacing** :: The basic steps of co-synthesis. A range of methodologies are applied for this.

4. **Implementation generation** :: Once the architecture has been generated and trade-offs known, the designs for the hardware and software components must be created.

5. **Co-simulation and Emulation** :: This helps designers to evaluate architectures and validate assumptions on implementations. Emulation uses FPGA / other emulation techniques to further speed up execution of system models.

Course Introduction

Basic Pitfalls – What to guard against

1. **Transient Overloads** :: Transient Overloads on multiple aspects like power, memory, out of bandwidth, data loss, missed deadlines of critical tasks.

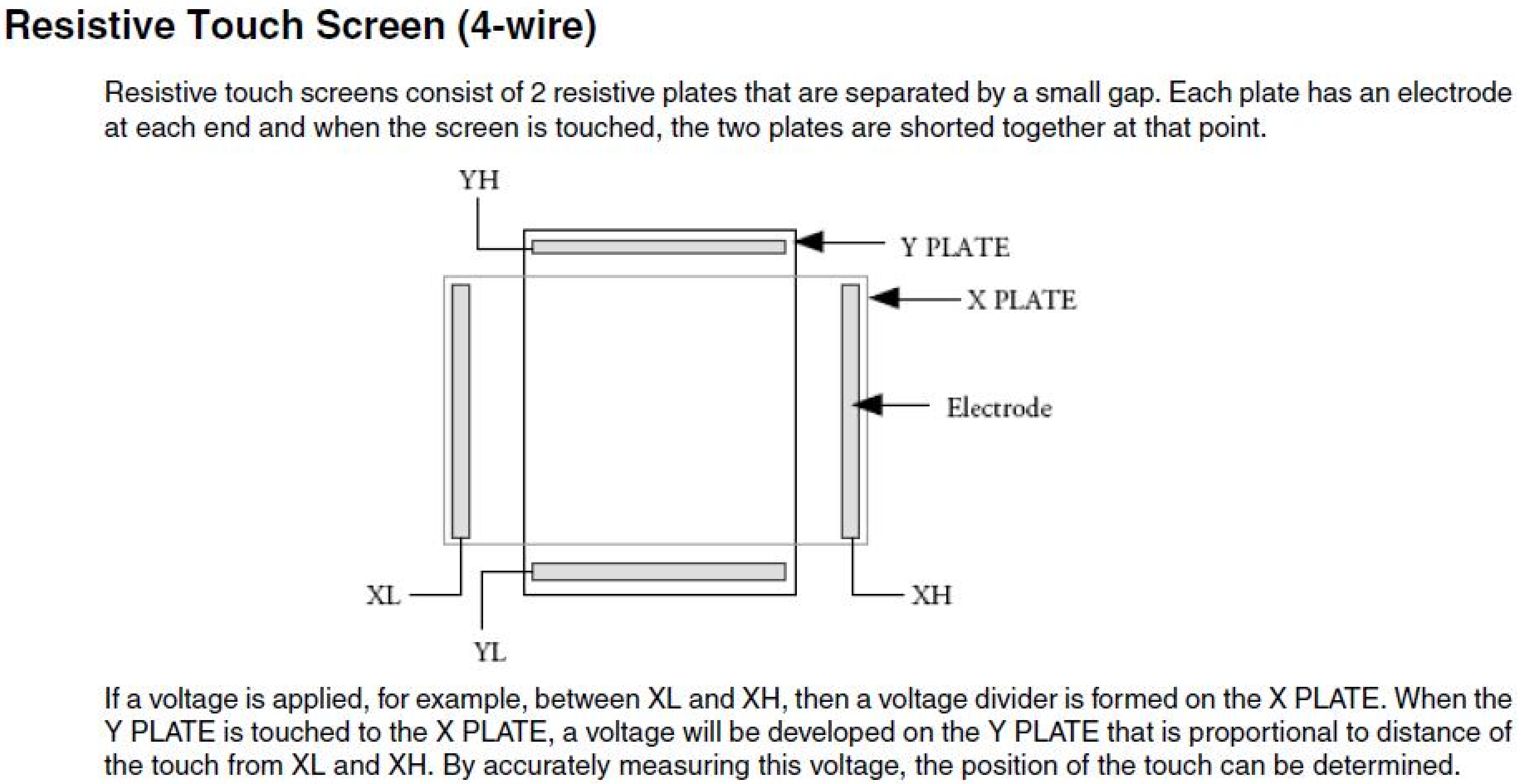
2. **Analysis Paralysis** :: Since there are many ways of doing it, there could be too many factors which do not allow the system designer to take decisions. A few ways of taking ''judgement'' decisions are based on fixing a set of topmost criteria and then working backwards.

3. **Simulation & Complexity** :: Simulation based performance verification, has a conceptual disadvantage that it becomes disabling as complexity increases. There is a great lot of dependence on the pattern provided for finding corner cases.

4. **Applications of the SoC** :: Generally an SoC would be implemented with a certain fixed application in focus. In some cases, the SoC could be used in non-typical or non-designed for applications. This would break some concurrency conditions which were not forseen earlier. Basically, for each new application, one needs to re-run the scenarios in case of using an existing SoC.

Course Introduction

Problem Statement of a touch screen system



Resistive Touch Screen (4-wire)

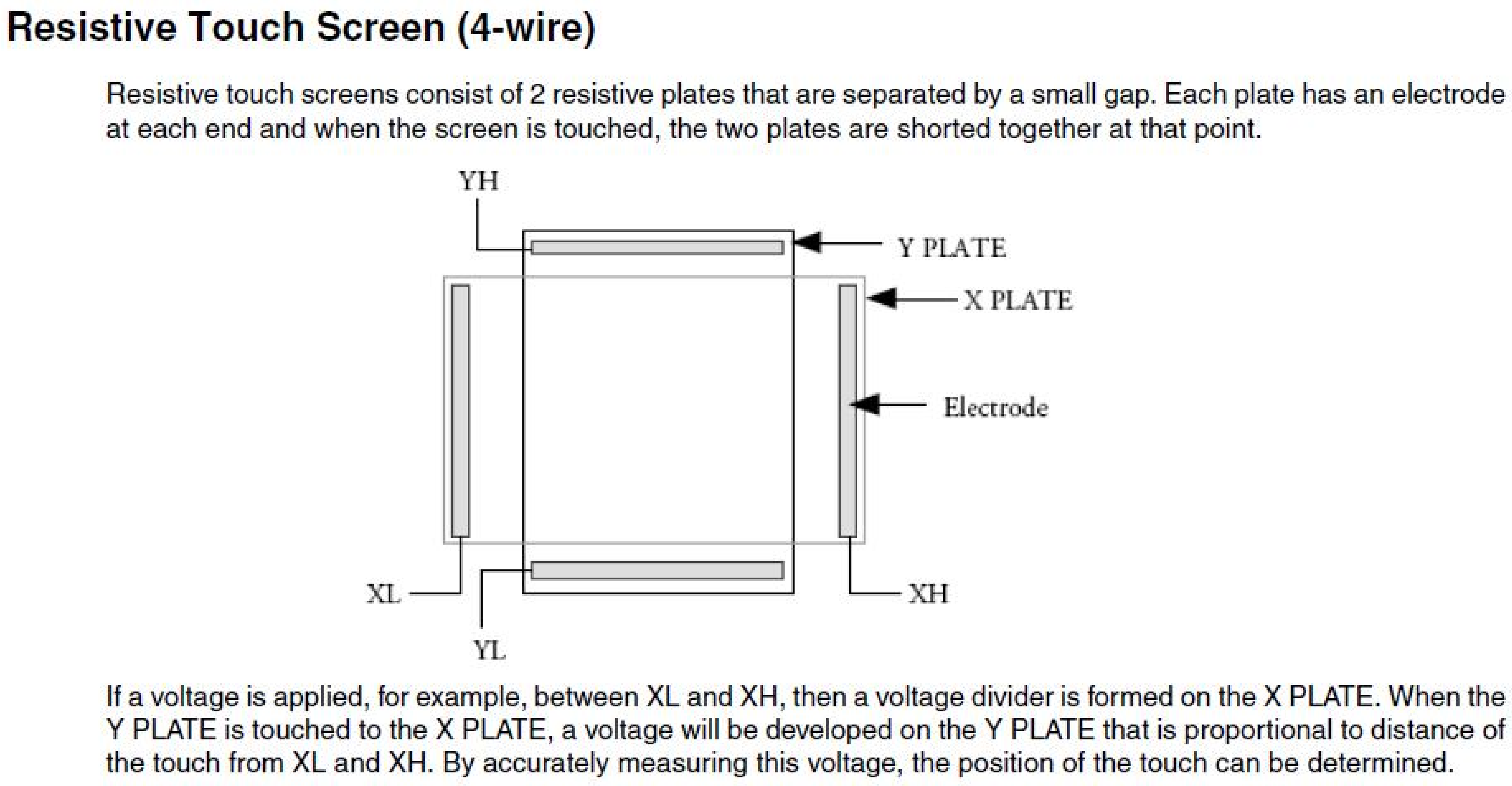
Resistive touch screens consist of 2 resistive plates that are separated by a small gap. Each plate has an electrode at each end and when the screen is touched, the two plates are shorted together at that point.

YH
Y PLATE
X PLATE
Electrode
XL
XH
YL

If a voltage is applied, for example, between XL and XH, then a voltage divider is formed on the X PLATE. When the Y PLATE is touched to the X PLATE, a voltage will be developed on the Y PLATE that is proportional to distance of the touch from XL and XH. By accurately measuring this voltage, the position of the touch can be determined.

Course Introduction

Problem Statement of a touch screen system – Requirements



Resistive Touch Screen (4-wire)

Resistive touch screens consist of 2 resistive plates that are separated by a small gap. Each plate has an electrode at each end and when the screen is touched, the two plates are shorted together at that point.

If a voltage is applied, for example, between XL and XH, then a voltage divider is formed on the X PLATE. When the Y PLATE is touched to the X PLATE, a voltage will be developed on the Y PLATE that is proportional to distance of the touch from XL and XH. By accurately measuring this voltage, the position of the touch can be determined.

1. Level 0 :: It should work at lowest cost.

Course Introduction

Problem Statement of a touch screen system – Requirements

1. Level 0 :: It should work at lowest cost.

2. Level 1 :: Some more details :-

   (a) System should continuously monitor for a touch.

   (b) When screen is touched, find out the X and Y coordinates.

   (c) Based on the event(s) of touch and location / pattern, tasks must be performed / activated.

Course Introduction

Problem Statement of a touch screen system – Requirements

1. Level 0 :: It should work at lowest cost.

2. Level 1 :: Some more details :-
   (a) System should continuously monitor for a touch.
   (b) When screen is touched, find out the X and Y coordinates.
   (c) Based on the event(s) of touch and location / pattern, tasks must be performed / activated.

3. Level 2 :: Some more details :-
   (a) Automated Screen Calibration
   (b) Automatically wakes up and goes back to standby to save power
   (c) Simple to write software and drivers
   (d) Programmable conversion rate

Course Introduction

Problem Statement of a touch screen system – Requirements

1. Level 0 :: What are the basic components required
   (a) An ADC which can do a ”voltage sensing”
   (b) A uP which can take in the Digital Input from ADC and do processing.
   (c) Some memory to store successive touch(s) and do a pattern recognition !
   (d) A ROM memory to store program, calibration related data.

Course Introduction

Problem Statement of a touch screen system – Requirements

1. Level 0 :: What are the basic components required

   (a) An ADC which can do a ”voltage sensing”

   i. What should be the accuracy, precision of the ADC? 8b, 12b

   ii. Should the ADC be a differential mode or single ended mode.

   iii. What should be the minimum acceptable frequency for the ADC? Depends on how ”Human Machine Interface” like touch screen works.

   iv. KBD controller does key debounce at 10 1̃5ms. Should this be a programmable frequency. In what steps should it be programmable. What complexity will be added to the system. Can we make the system more simple. What is the tradeoff.

   (b) A uP which can take in the Digital Input from ADC and do processing.

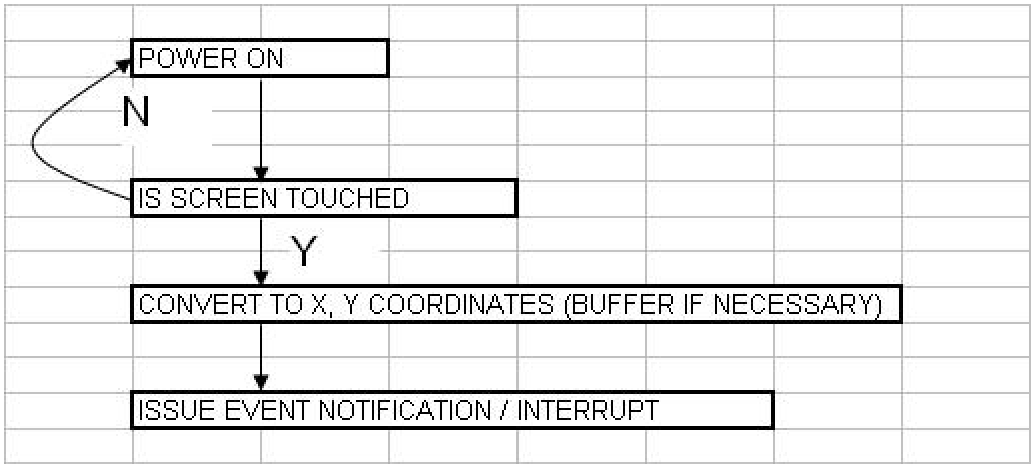
   (c) Some memory to store successive touch(s) and do a pattern recognition !

   (d) A ROM memory to store program, calibration related data.

Course Introduction

Problem Statement of a touch screen system – Requirements

1. Level 0 :: What are the basic components required
   (a) An ADC which can do a ”voltage sensing”
   (b) A uP which can take in the Digital Input from ADC and do processing.
      i. Write a small flow chart on how the data and control would flow from ADC to uP and control the events based on touch.



POWER ON

N

IS SCREEN TOUCHED

Y

CONVERT TO X, Y COORDINATES (BUFFER IF NECESSARY)

ISSUE EVENT NOTIFICATION / INTERRUPT

      ii. Write the software as if it was processor agnostic.
      iii. What is the complexity of the program.
      iv. Is an 8bit general purpose processor ok. What will require an upgrade to 16bit or 32bit general purpose processor.
      v. Is there anything in the program which can be hardware accelerated at low cost. Will a DSP suffice for this application.

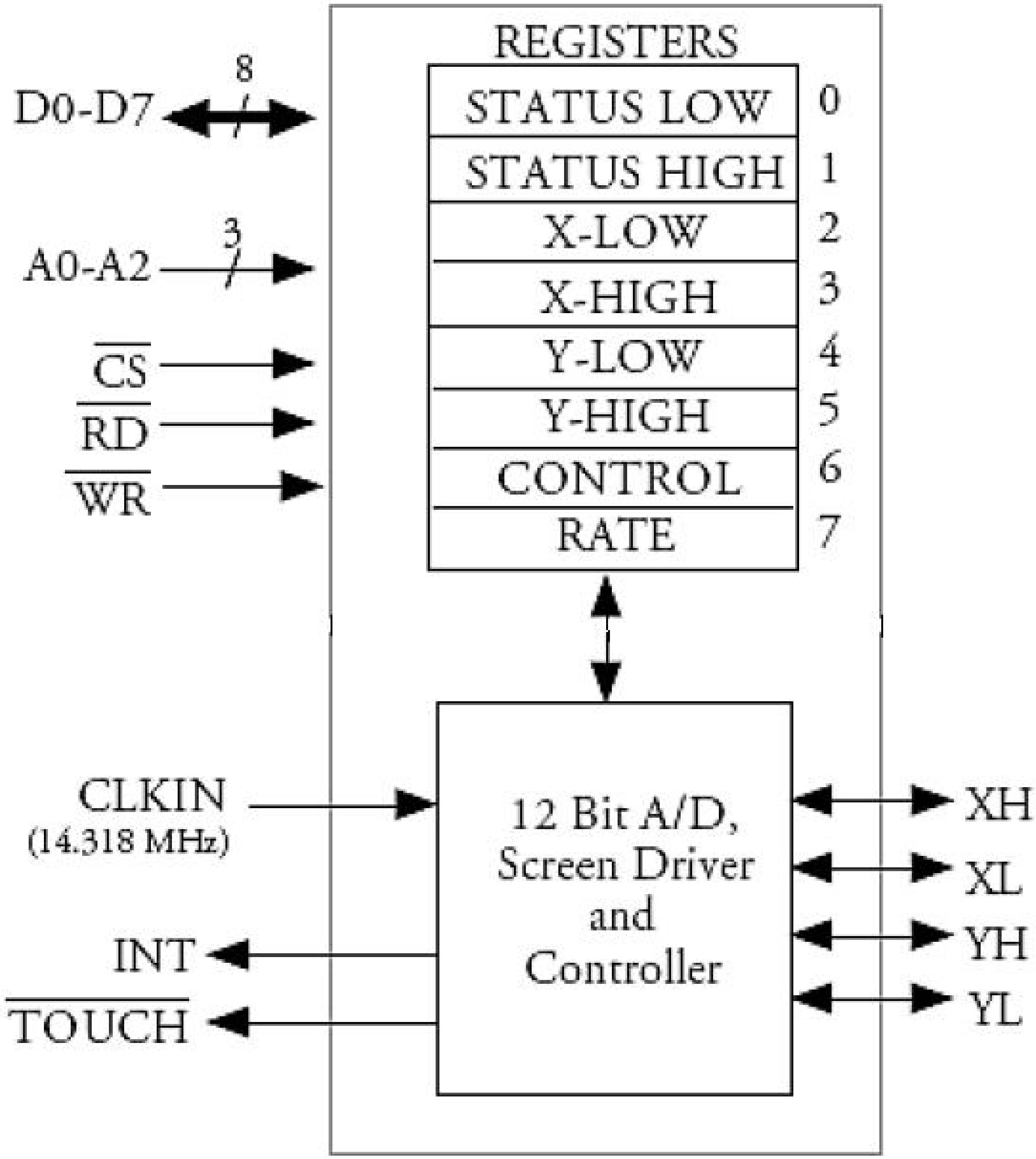
(c) Some memory to store successive touch(s) and do a pattern recognition !

(d) A ROM memory to store program, calibration related data.

Course Introduction

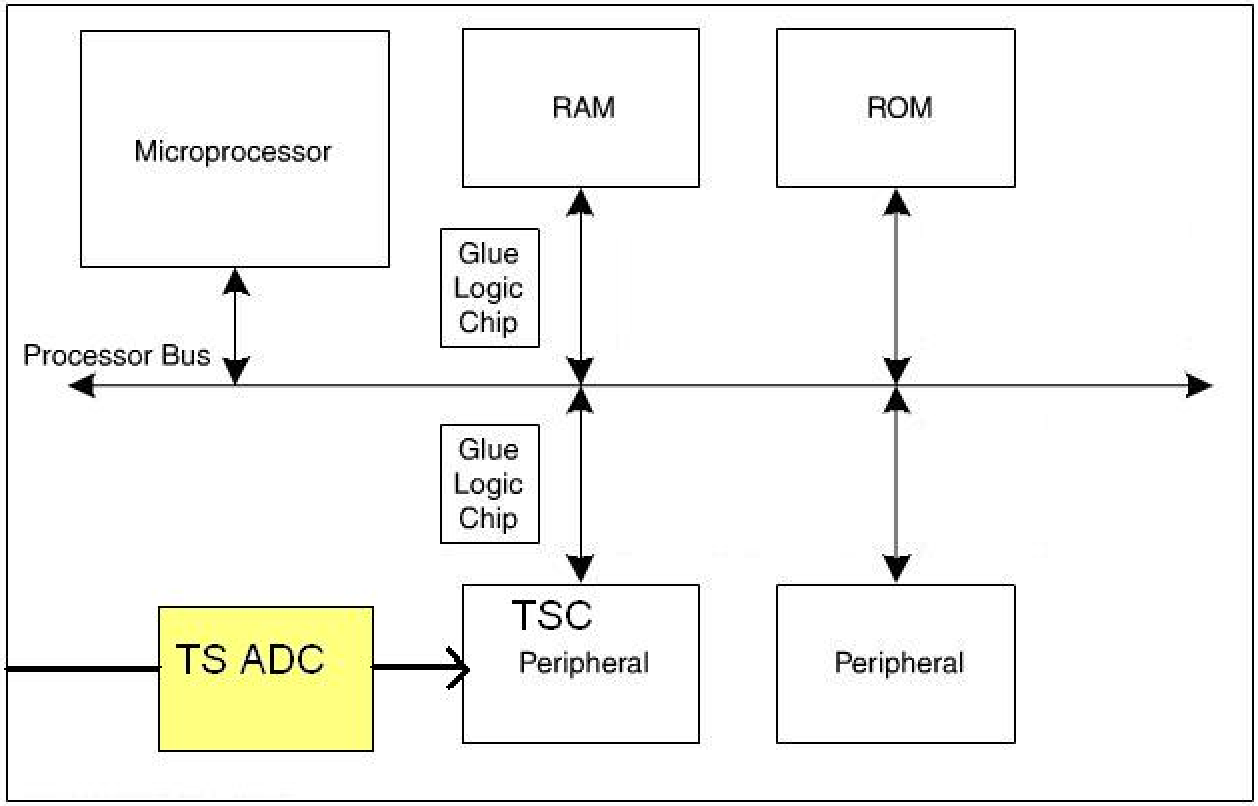
Problem Statement of a touch screen system – Design

1. Level 1 :: What are the basic components which we can fix (pin down)
   (a) The technology node based on requirements
   (b) The ADC
   (c) The uP or DSP
2. Level 2 :: Can we prototype this system using off the shelf components
   (a) Create the IP required to interface ADC with the uP/DSP busses. Add any hardware acceleration required.


REGISTERS
STATUS LOW 0
STATUS HIGH 1
X-LOW 2
X-HIGH 3
Y-LOW 4
Y-HIGH 5
CONTROL 6
RATE 7
D0-D7
8
A0-A2
3
$\overline{CS}$
$\overline{RD}$
$\overline{WR}$
CLKIN
(14.318 MHz)
INT
$\overline{TOUCH}$
12 Bit A/D, Screen Driver and Controller
XH
XL
YH
YL

Course Introduction

Problem Statement of a touch screen system – Design Block Diagram



Course Introduction

Problem Statement of a touch screen system – Software Design At the top level, the software provides for the following :-

1. Configure the controller hardware.
2. Determine if the screen is touched.
3. Acquire Stable, debounced position measurements.
4. Calibrate the touch screen.
5. Send changes in touch status and position to the higher level graphics software.

Course Introduction

Problem Statement of a touch screen system – Software Design

Configure the controller hardware.

1. Create a function named `TouchConfigureHardware()`
2. Decide – Should the driver be interrupt driven or polling driven.
3. In case of interrupts, the driver would actually use two :-
   (a) An interrupt to wake up when the screen is initially touched, known as the `PEN_DOWN` interrupt.
   (b) A second interrupt to signal when the ADC is available with the set of data conversions (X, Y).

Determine if the screen is touched.

1. Create a function named `WaitForTouchState ()`
2. When the controller is in the detection mode and a touch is detected, an internal interrupt can be generated called `PEN_DOWN IRQ`.
3. This detection is based on Y-axis touch plane tied high, X-axis touch plane tied low, and on the basis of touch the planes are shorted together and Y-axis plane is pulled low.
4. The driver task would not consume any CPU time until the `PEN_DOWN IRQ` event occurs. It would wake up and go into conversion mode only once the user touches screen.

Course Introduction

Problem Statement of a touch screen system – Software Design

Acquire Stable, debounced position measurements – Reading touch data

1. Create a function named `TouchScan()`. The outline of the procedure would be :-
   (a) Check to see if the screen is touched.
   (b) Take several raw readings on each axis for later filtering.
   (c) Check to see if the screen is still touched.
   (d) (Depending on H/W) store the readings to a memory block, or use FIFO entries.
   (e) (Depending on H/W) if the ADC output is 12bit, either pack data into 16bit (aligned) locations or chop/dither them to 8bit.
   (f) Taking Stable readings (filtering by using oversampling) is necessary for higher level drivers to act appropriately. NOTE :: This filtering could be a h/w function if CPU bandwidth is critical factor.

Calibrate the touch screen.

1. In an ideal scenario, the calibration would be run once during initial product power up and reference values saved in ”non-volatile” memory.

2. Create a function name `CalibrateTouchScreen ()` in case the user wants to calibrate using a graphical target on screen.

Course Introduction

Problem Statement of a touch screen system – Software Design

Send changes in touch status and position to the higher level graphics software.

1. Create a function named `GetScaledTouchPosition()`. This is a routine to read raw values and convert them to screen co-ordinates.

2. Create a function named `TouchTask ()`. This routine calls the actual task the user intended to be run while using the touch screen.

Course Introduction

Acknowledgements

1. Datasheet of IDT MK712 Touch Screen Controller
2. Application Bulletin "Burr Brown" now TI (public domain information) :: Touch Screen Controller Tips.
3. http://www.eetimes.com/design/embedded/4006455/Writing-drivers-fo common-touch-screen-interface-hardware