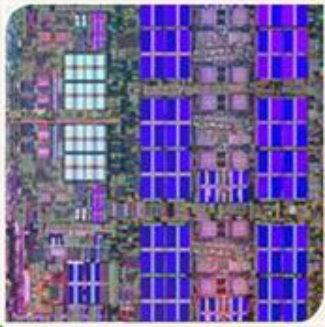
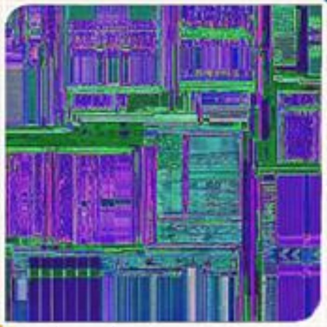



CS4803/CS8803 PGC Design and Programming of Game Console

Spring 2012

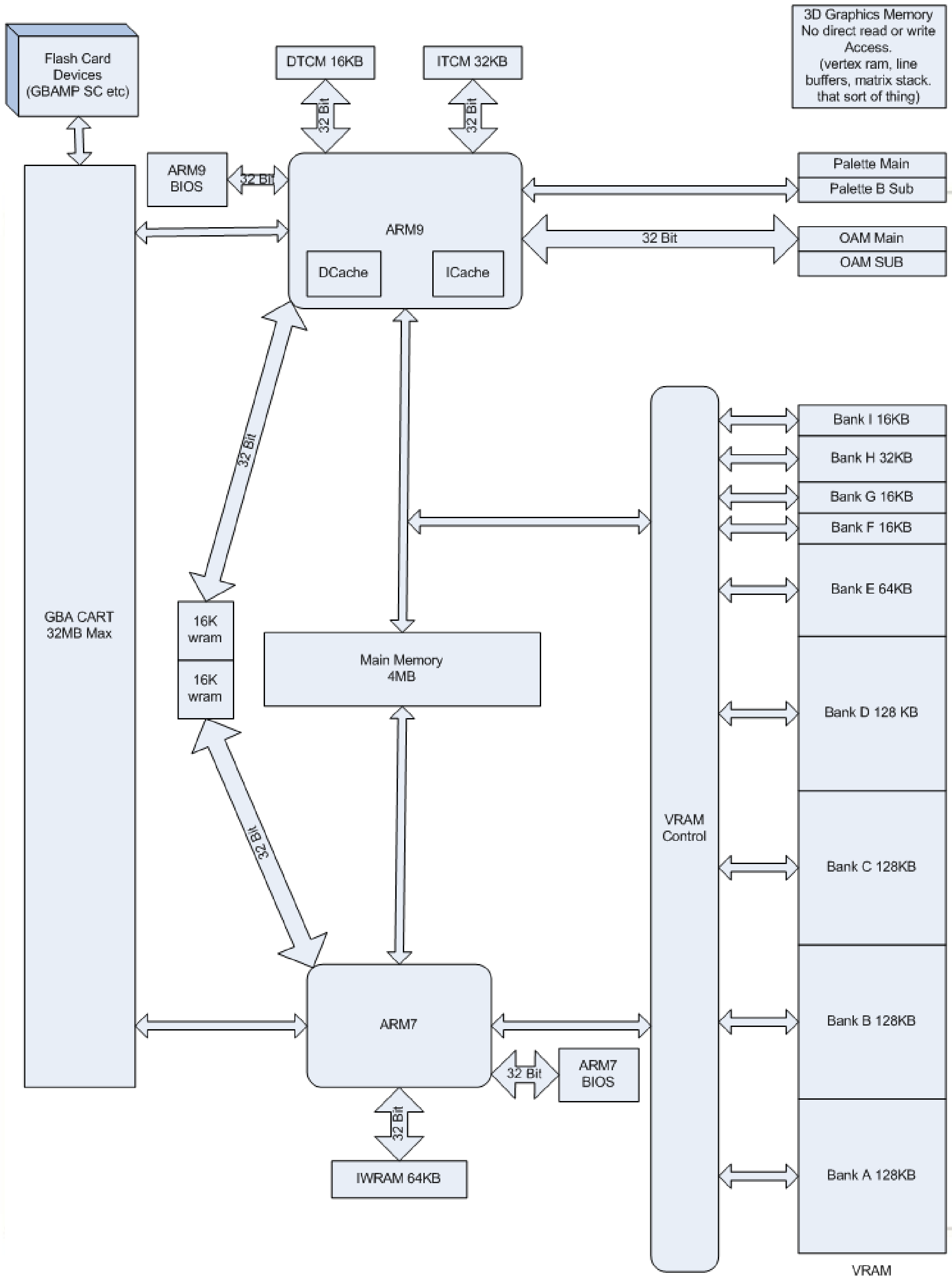
Prof. Hyesoon Kim

Georgia Tech College of Computing

Assignment #1

- Assembly code to read touch screen coordinates
- What to learn
  - How to handle peripheral interface in NDS
  - Memory layout & communication between ARM 7 & ARM 9
    - ARM 7 can access touch screen
    - ARM 9 can display

Nintendo DS Memory



TCM (tightly coupled memory)

WRAM: wireless Ram

Read

http://dev-scene.com/NDS/Tutorials_

Memory Space

ARM 9				
Name	Start Address	Stop Address	Size	Wait State
Main	0x02000000	0x023FFFFF	4MB	?
BIOS	0xFFFF0000	0xFFFF7FFF	32KB	?
ITCM	0x00000000	0x00007FFF	32KB	?
DTCM	0x0B000000	0x0B003FFF	16KB	?
Shared WRAM Bank 0	0x03000000	0x03003FFF	16KB	?
Shared WRAM Bank 1	0x03004000	0x03007FFF	16KB	?
ARM 7				
Main	0x02000000	0x023FFFFF	4MB	?
BIOS	0x00000000	0x00003FFF	16KB	?
IWRAM	0x03800000	0x0380FFFF	64KB	?
Shared WRAM Bank 0	0x03000000	0x03003FFF	16KB	?
Shared WRAM Bank 1	0x03004000	0x03007FFF	16KB	?
Video RAM				
Main OAM	0x07000000	0x070003FF	1KB	?
Sub OAM	0x07000400	0x070007FF	1KB	?
Main Palette	0x05000000	0x050003FF	1KB	?
Sub Palette	0x05000400	0x050007FF	1KB	?
Bank A	0x06800000	0x0681FFFF	128KB	?
Bank B	0x06820000	0x0683FFFF	128KB	?
Bank C	0x06840000	0x0685FFFF	128KB	?
Bank D	0x06860000	0x0687FFFF	128KB	?
Bank E	0x06880000	0x0688FFFF	64KB	?
Bank F	0x06890000	0x06983FFF	16KB	?
Bank G	0x06894000	0x06897FFF	16KB	?
Bank H	0x06898000	0x0689FFFF	32KB	?
Bank I	0x068A0000	0x068A3FFF	16KB	?

Virtual Video RAM				
Main Background	0x06000000	0x0607FFFF	512KB	?
Sub Background	0x06200000	0x0621FFFF	128KB	
Main Sprite	0x06400000	0x0643FFFF	256KB	?
Sub Sprite	0x06600000	0x0661FFFF	128KB	?

Memory

- Main memory (4MB)

  Start Address: 0x0200:0000

  End Address: 0x023F:FFFF

  Mirror : 0x0240:0000 // not cached

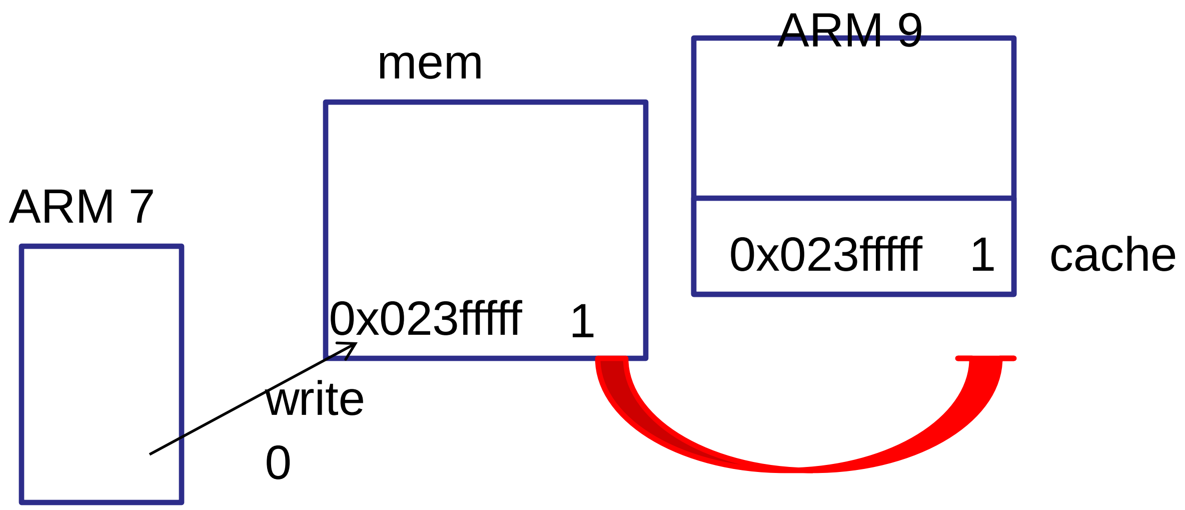
  - Both ARM7 and ARM9 can access the memory.
  - Usually ARM9's executable code and game data.
  - ARM 7's executable code is stored in IWRAM in DevKit Pro
  - (most games also use IWRAM first and then use main memory)
- IWRAM: ARM 7 Fast RAM
  - Start Address : 0x03800000 End Address : 0x0380FFFF
  - Only 64KB → so try to fit code here!

Important Tips for

- Two main functions- 1 for each processor
- Only Arm7 can access the coordinates and ARM 9 displays.
- The chosen address must be greater than 0x02400000. Because all addresses lower than this are cached in the arm9.
- No cache coherence support

Georgia Tech College of Computing

No Coherence Support



- ARM 9 has I-cache, D-cache
- Software flush and synch (libraries are provided in DevKitPro)

ARM9 code: Display data from ARM 7

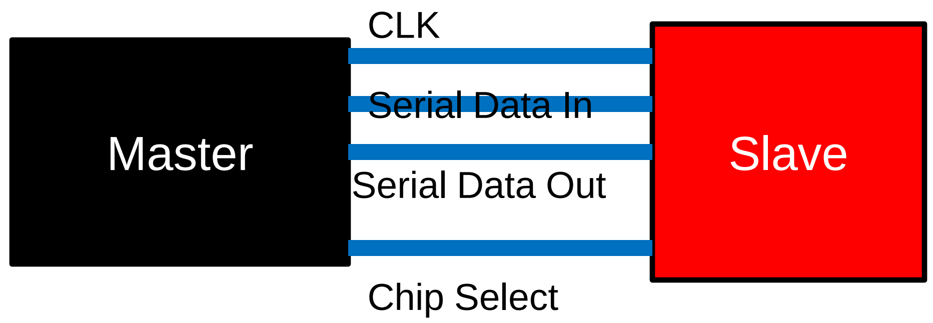
Here Xreg and Yreg are two randomly chosen mem locations (copyToAddressX, copytoAddressY) accessible to both processors

Your job to figure out possible addresses of copyToAddressX, copytoAddressY

```
#include <nds.h>
#include <stdio.h>
int main(void) {
    consoleDemoInit();
    volatile int* Xreg= (int *) copyToAddressX; //should be 32 bit aligned
    volatile int* Yreg= (int *) copyToAddressY;// should be 32 bit aligned

while(1) {
    iprintf ("\x1b[5;0H X: %2X", *Xreg>>20);
    iprintf ("\x1b[6;0H Y: %2X", *Yreg>>20);
    swiWaitForVBlank();
    }
    return 0;
    }
```

Serial Peripheral Interface



- Serial Peripheral Interface (SPI) provides communication with external devices in master or slave mode.
- 4 wires or 3 wires (bidirectional wires)
- Hand shaking protocols (e.g. RS-232) MSB first or LSB first? Check sum information? Beginning/ending?
- Store information in the Main memory

DS Serial Peripheral Interface (SPI)

- SPI Bus is a 4-wire
  - – Data in,
  - – Data out,
  - – Clock,
  - – Chip set
- 4 devices
  - (00)DS Firmware serial flash memory
  - (01) DS touch screen controller
  - (10) DS power management
  - (11) Nothing wired up

SPI Bus control/Status Register

40001C0h - NDS7 - SPICNT - SPI Bus Control/Status Register

0-1 Baudrate (0=4MHz/Firmware, 1=2MHz/Touchscr, 2=1MHz/Powerman., 3=512KHz)

2-6 Not used (Zero)

7 Busy Flag (0=Ready, 1=Busy) (presumably Read-only)

8-9 Device Select (0=Powerman., 1=Firmware, 2=Touchscr, 3=Reserved)

10 Transfer Size (0=8bit/Normal, 1=16bit/Bugged)

11 Chip select Hold (0=Deselect after transfer, 1=Keep selected)

12-13 Not used (Zero)

14 Interrupt Request (0=Disable, 1=Enable)

15 SPI Bus Enable (0=Disable, 1=Enable)

```
REG_SPICNT =      SPI_ENABLE |       SPI_BAUD_2MHz |
        SPI_DEVICE_TOUCH |       SPI_CONTINUOUS;
   //0x8A01;
```

SPI Read/Write

- Synchronous clock protocol
  - A single lock one data is sent to slave and one data is sent back to the master typically using the same register
  - 40001C2h - NDS7 - SPIDATA - SPI Bus Data/Strobe Register (R/W)
  - Even if you want to read, you must write a dummy value
  - 0-7 data, 8-15 NOT USED (even for 16-bit mode)
  - During transfer busy flag in SPICNT is set and when the transfer is done, busy flat is clear

Caution!

- 40001C0h :SPICNT
- 40001C2h : SPIDATA
- How many bytes in SPICNT ?
  - Use STRH

Example Timing Diagram

Data Transfer The following waveforms show examples of data transfers.

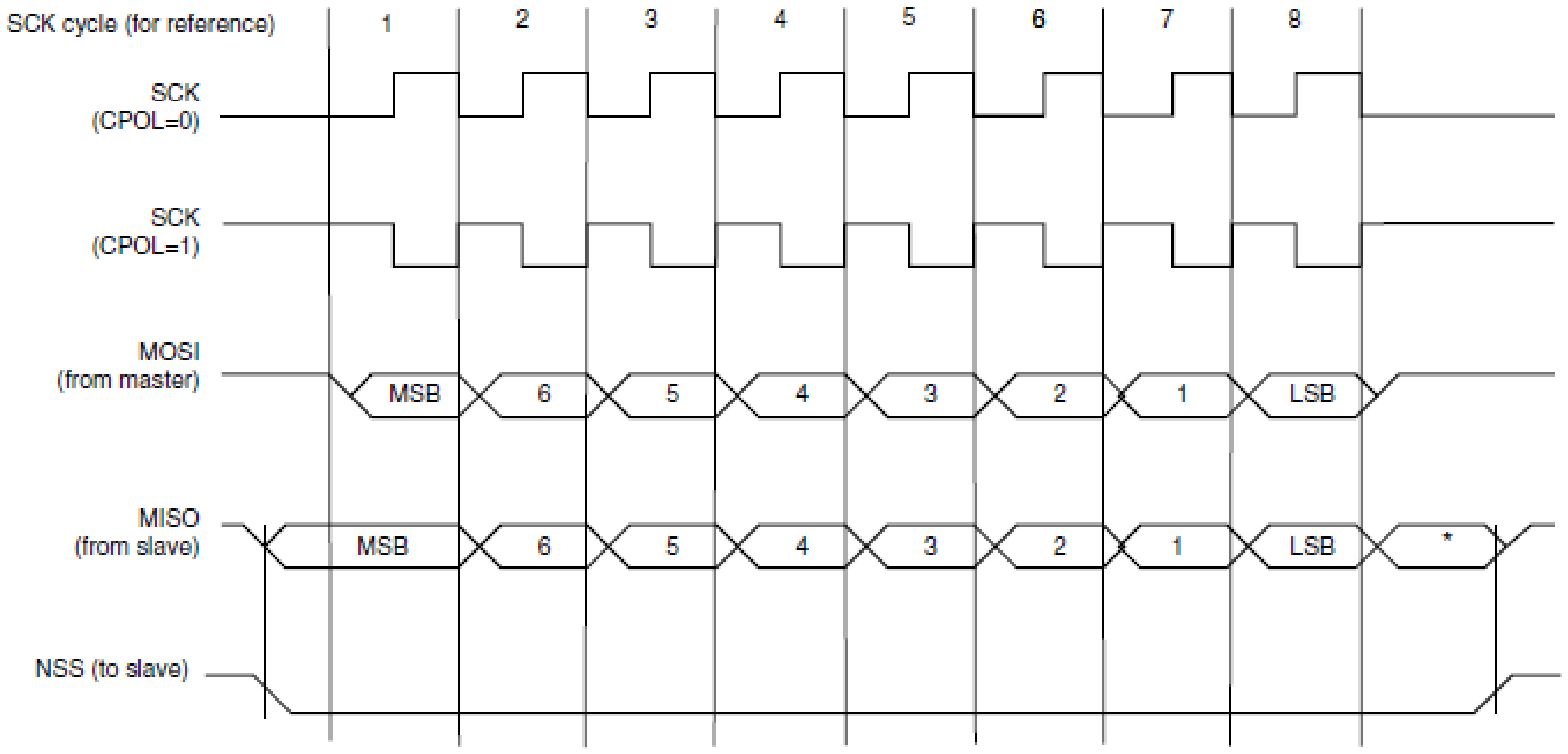


Figure 7. SPI Transfer Format (NCPHA equals One, 8 bits per transfer)

* Not defined, but normally MSB of previous character received.

Georgia Tech College of Computing

Steps

```
Loop{
    Set up REG_SPICNT
    Disable interrupt
    Wait until SPI is ready
    READ REG_SPIDATA
    Read once more after setting REG_SPICNT (continuous reading)
    READ REG_SPIDATA
    Recover interrupt setting
    Write data in a mem location where ARM 9 can read
    Wait to give some time to ARM 9 access MEM
    }
```

For some reason send
5 bit + 3 bit
So read two times

ARM 7 Code

```
while {
        Dummy loops //slow down the access // ARM 7 has a high priority
        oldIME = REG_IME; // create backup
        REG_IME = 0; //  turn interrupts off
        // REG_IME 0x04000208 16 bits Interrupt Master Enable Register
        REG_SPICNT = SPI_ENABLE | SPI_BAUD_2MHz | SPI_DEVICE_TOUCH |
             SPI_CONTINUOUS; //0x8A01
        REG_SPIDATA = TSC_MEASURE_X;
        SerialWaitBusy();
        REG_SPIDATA = 0;
        SerialWaitBusy();
        result = REG_SPIDATA;  // MSB bit first
// Clock in the rest of the data (last transfer)
      REG_SPICNT = SPI_ENABLE | 0x201;
      REG_SPIDATA = 0;
      SerialWaitBusy();
      result2 = REG_SPIDATA >>3;
      REG_IME = oldIME;  // restore interrupt enable information
        *copyToAddressX= ((result & 0x7F) << 5) | result2;   // write data so that ARM 9 can
      read
      }
```

MEM locations

REG_SPICNT = 0x040001C0

REG_SPIDATA= 0x040001C2

TSC_MEASURE_X= 0xD0
  TSC_MEASURE_Y=0x90
  TSC_MEASURE_Z1=0xB4
  TSC_MEASURE_Z2=0xC4

Touch Screen Controller

Control Byte (transferred MSB first)

0-1 Power Down Mode Select

2 Reference Select (0=Differential, 1=Single-Ended)

3 Conversion Mode (0=12bit, max CLK=2MHz, 1=8bit, max CLK=3MHz)

4-6 Channel Select (0-7, see below)

7 Start Bit (Must be set to access Control Byte)

Channel

0 Temperature 0 (requires calibration, step 2.1mV per 1'C accuracy)

1 Touchscreen Y-Position (somewhat 0B0h..F20h, or FFFh=released)

2 Battery Voltage (not used, connected to GND in NDS, always 000h)

3 Touchscreen Z1-Position (diagonal position for pressure meas.)

4 Touchscreen Z2-Position (diagonal position for pressure meas.)

5 Touchscreen X-Position (somewhat 100h..ED0h, or 000h=released)

6 AUX Input (connected to Microphone in the NDS)

7 Temperature 1

Touch Screen Controller

- Power down mode

Mode	/PENIRQ	VREF	ADC	Recommended use
0	Enabled	Auto	Auto	Differential Mode (Touchscreen, Penirq)
1	Disabled	Off	On	Single-Ended Mode (Temperature, Microphone)
2	Enabled	On	Off	Don't use
3	Disabled	On	On	Don't use

VREF : internal reference voltage NDS has external reference voltage that is always on

ADC: Analog to digital convert

Penirq: Pen interrupt

All channels are accessible from single-ended mode

Only 1,2,3,4,5 channels are accessible from differential mode

TSC2046

- Low-voltage I/O touch screen controller
- Power: 250mW

scr.x = (adc.x-adc.x1) * (scr.x2-scr.x1) / (adc.x2-adc.x1) + (scr.x1-1)
scr.y = (adc.y-adc.y1) * (scr.y2-scr.y1) / (adc.y2-adc.y1) + (scr.y1-1)

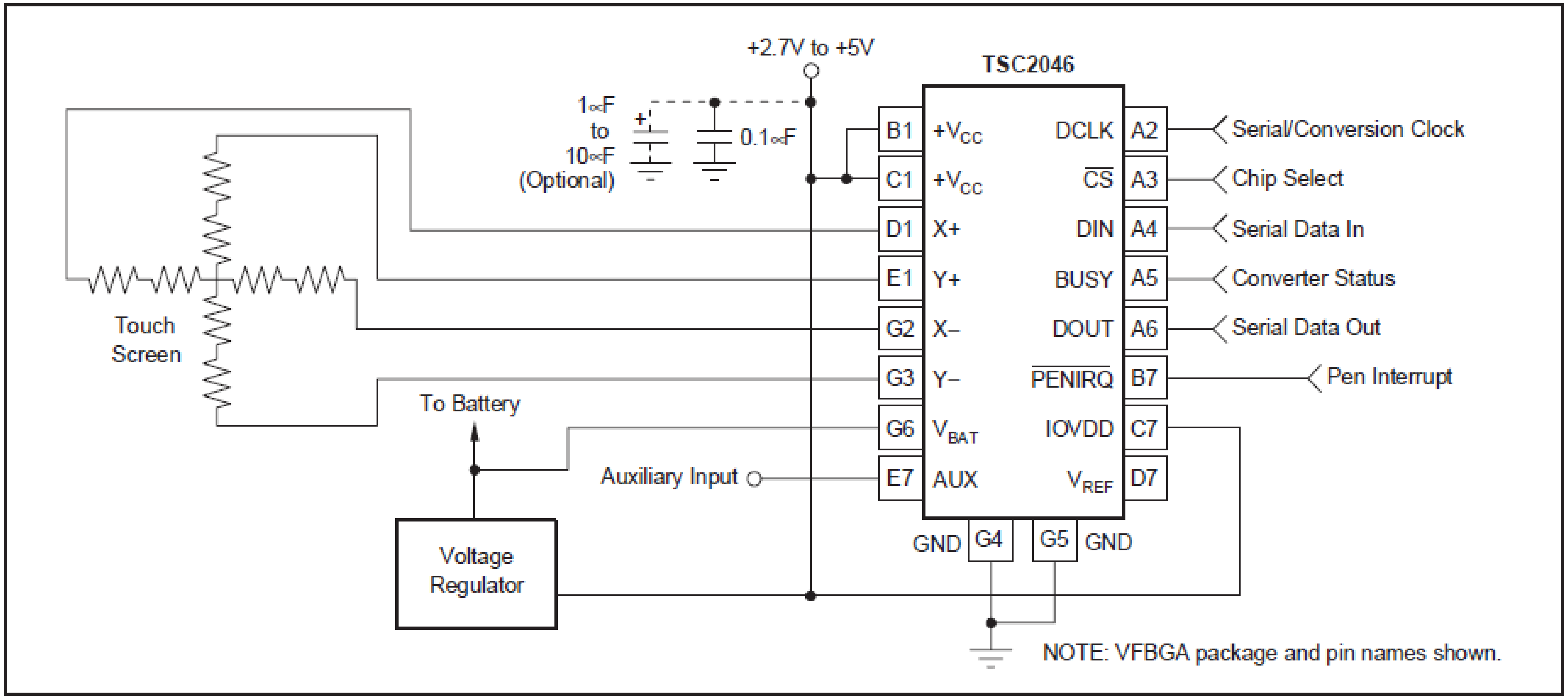


Figure 1. Basic Operation of the TSC2046

ARM 7 and ARM 9

- You have to compile separately.
- Building script will be provided in the assignment homepage

Assignment #1 Work Flow

- Start from “C” code
  - Submit “C” version of the code and “assembly code”
- Move on Assembly code (ARM 7 code) except
  - __SerialWaitBusy();
  - SPI set, Interrupt sets, read SPIDATA
- Report (10%): Each student
  - Commented code and simple descriptions
  - Any problems encountered and solutions
- Last assignment that uses ARM assembly coding

Framebuffer Mode

- 2 screens, 2 GPUs, only bottom has a touch screen
- A screen mode where the screen is mapped to a portion of memory
- Writing data to this memory area will result in data appearing on the screen
- Each screen’s pixel is represented by 2 B.
- Represented with 555 format
- 0123 4567 0123 4567

```
-rrr rr-- ---- ---- (bitmask: 0x7C00)
---- --gg ggg- ---- (bitmask: 0x3E0)
----    ----    --b bbbb (bitmask: 0x1F)
```

- But a simple macro

RGB15	Color
RGB15(31,0,0)	Red
RGB15(0,31,0)	Green
RGB15(0,0,31)	Blue
RGB15(0,0,0)	Black
RGB15(31,31,31)	White

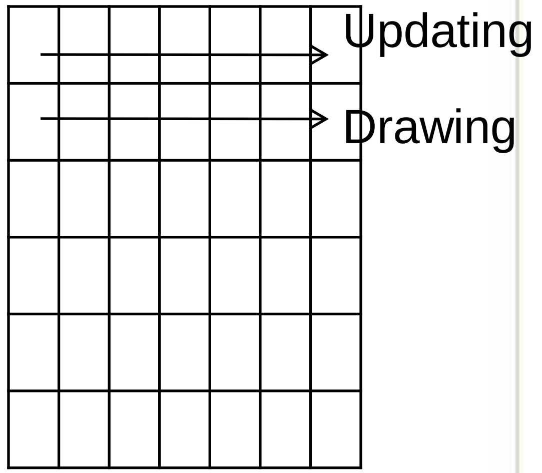
Vertical Blank Interrupt

Every 1/60th seconds, the hardware redraws.

Visiting each pixel row by row, copying the contents of the framebuffer for that pixel to the hardware screen pixel

Vertical blank interrupt: when it finishes drawing the screen

Use the interrupt to synchronize drawing.


Updating
Drawing