Computer Interfacing

LECTURE AND LABORATORY, 2/2560

Syllabus

1. Introduction to ETT STM32
2. General Purpose Input Output
3. Serial Communication (RS232)
4. Exception & Interrupt
5. ADC/DAC
6. Timer
7. SPI
8. LCD Display
9. Touch Screen Sensor
10. Ethernet
11. DMA
12. USB

สัดส่วนคะแนน Lecture

- Attendance 10% = 2% per Quiz
- Midterm 30%
- Final 30%
- Assignment 30%

สัดส่วนคะแนน Lab

- Attendance 20% = 2% per Lab
- Lab Submission 30%
- Final 25%
- Assignment 25%

Books for Laboratory

ปฏิบัติการไมโครคอนโทรลเลอร์ ARM Cortex-M3 กับ STM32

นคร ภักดีชาติ และ ชัยวัฒน์ ลิ้มพรจิตรวิไล

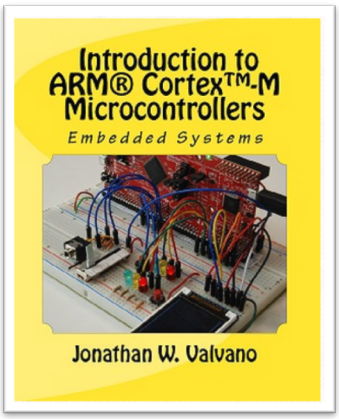
กรุงเทพ ฯ 2553

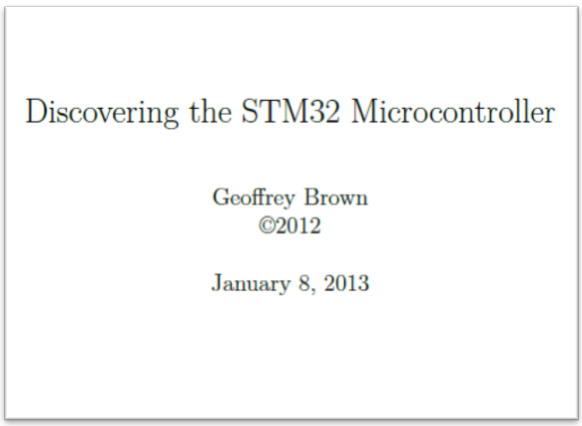
บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

http://www.inex.co.th/

ISBN: 9786169005933; 6169005939.

Books for Laboratory

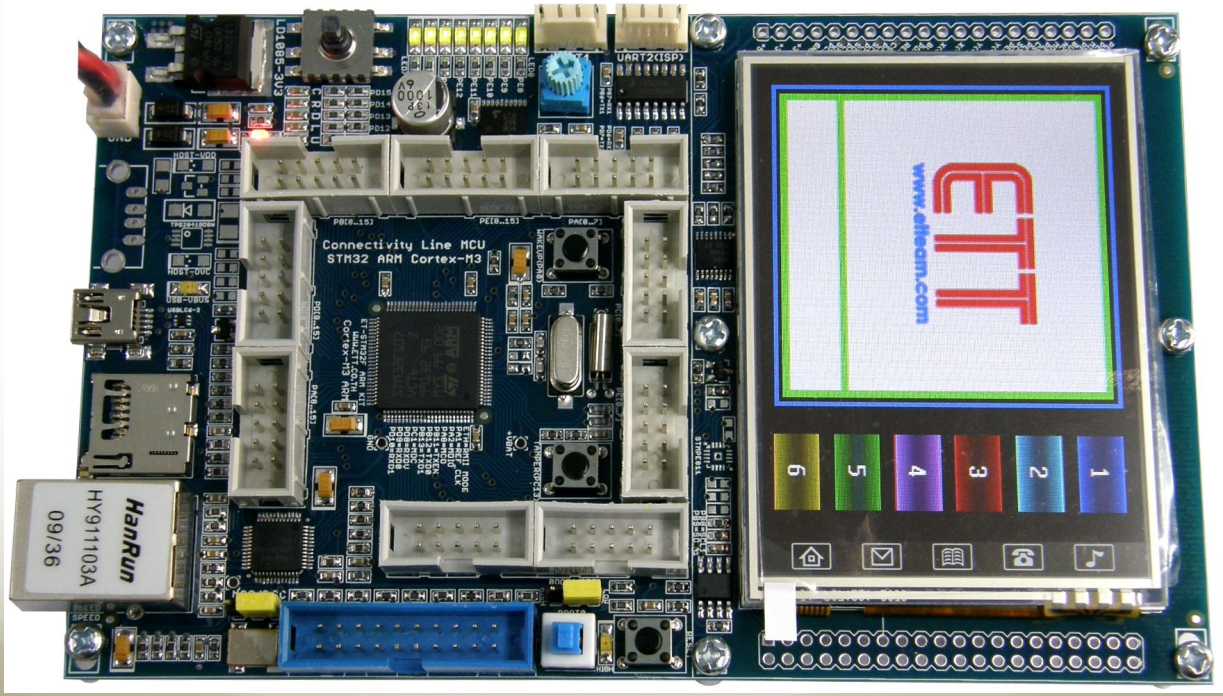




ETT STM32 Board

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ETT STM32 Board



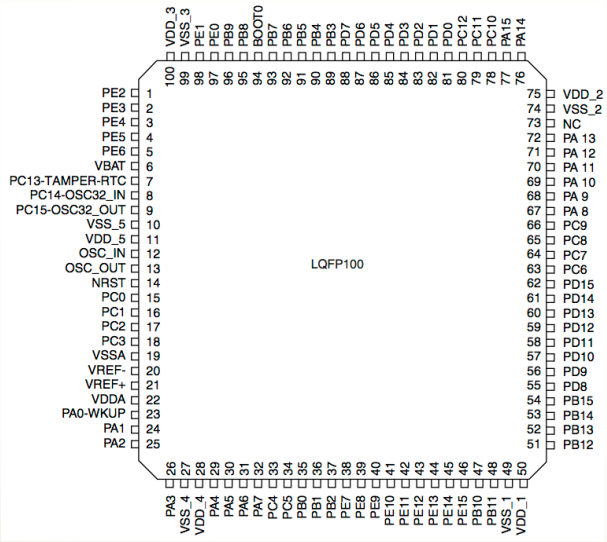
ETT STM32 Board

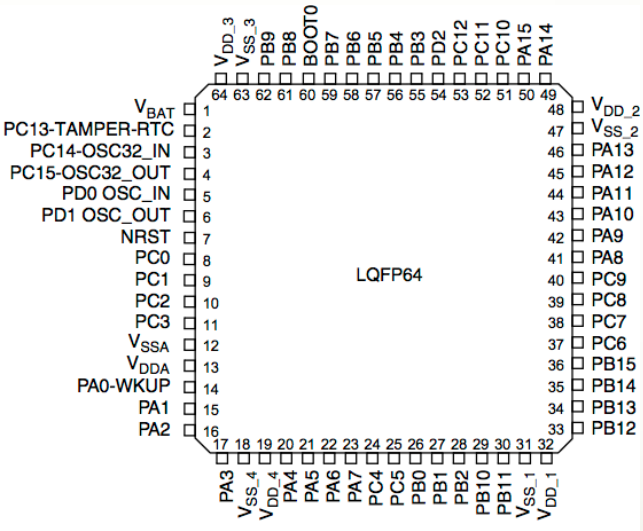
- CPU: STM32F107 microcontroller
  - ARM Cortex M3 LQFP100
  - Clock Frequency 25 MHz
- Memory (on chip)
  - Flash ROM 256 KByte
  - Static RAM 64 KByte
- I/O

ETT STM32 Board: I/O

- GPIO (General-Purpose Input/Output) Ports: I/O
- Joy Stick (5-way): I/P
- Buzzer: O/P
- 8 LEDs: O/P
- Touch Screen: I/P
- LCD display (320x240 pixels): O/P
- Ethernet port: I/O
- Micro Secure Digital (SD) Memory slot
- USB: Host and OTG

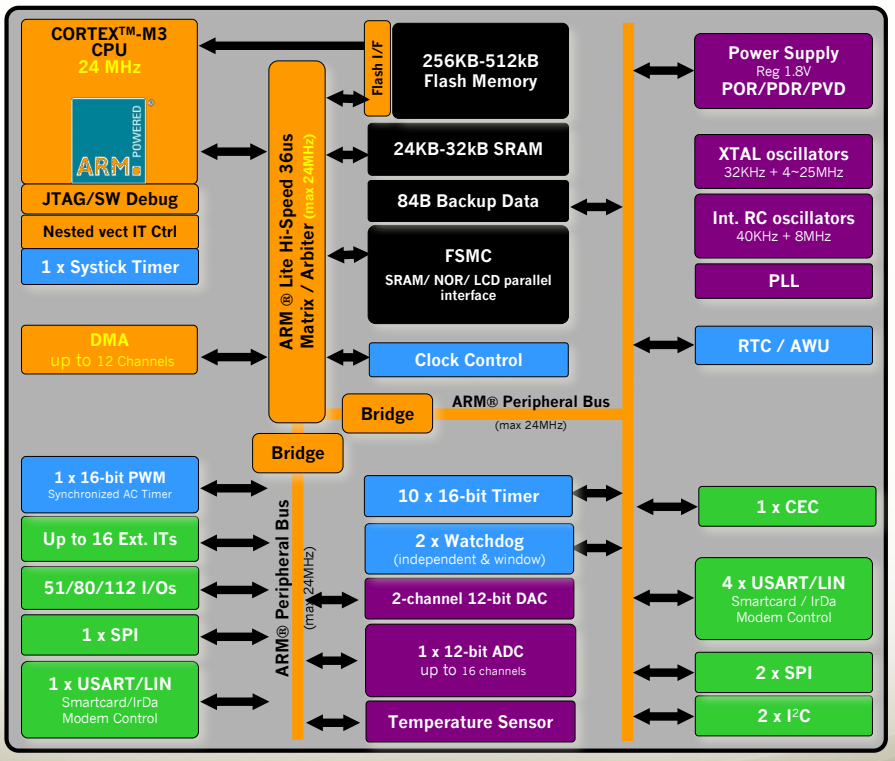
STM32F10x Packages: LQFP100 vs. LQFP64



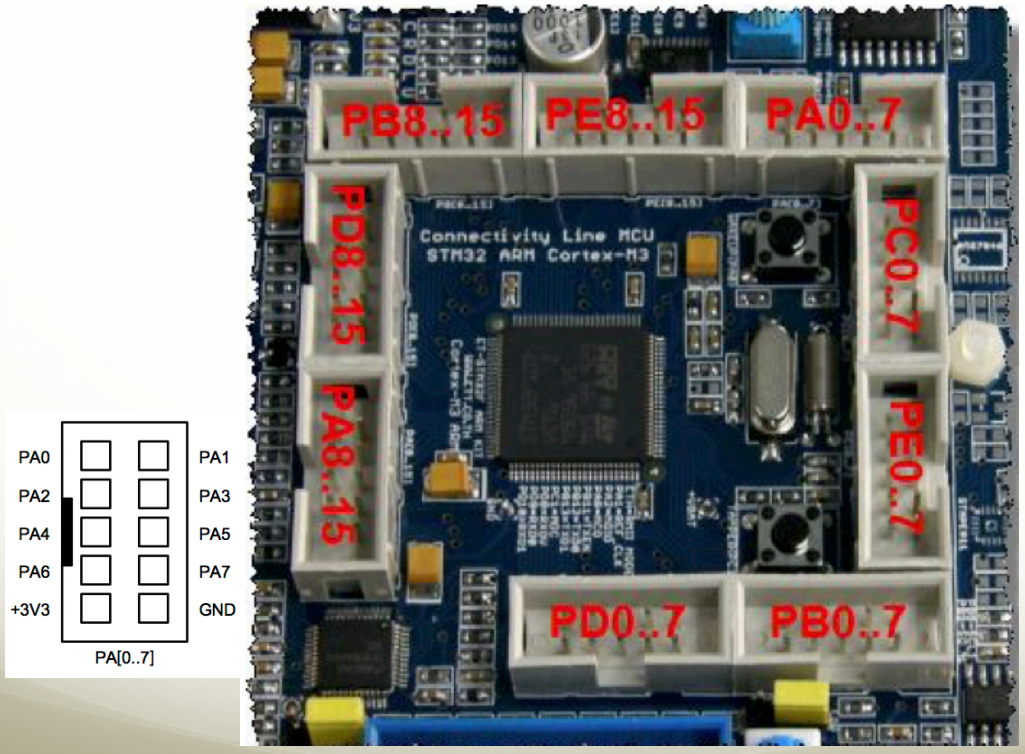


- 100 pins vs. 64 pins
- More pins, Higher Cost, Easier to Use

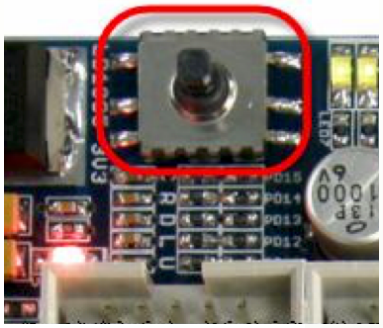
STM32F10x Series Block Diagram

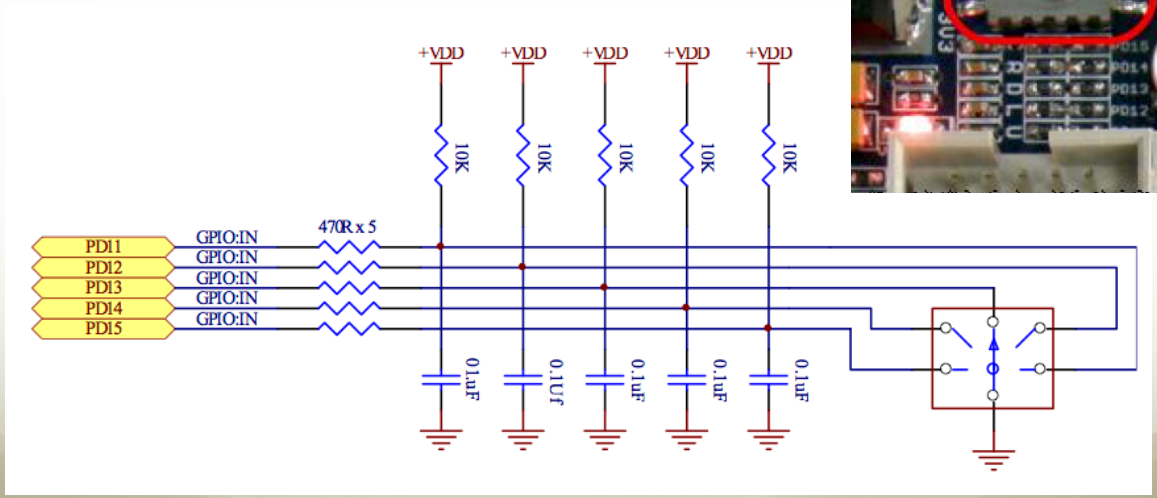


I/O Ports (Port A-Port E)

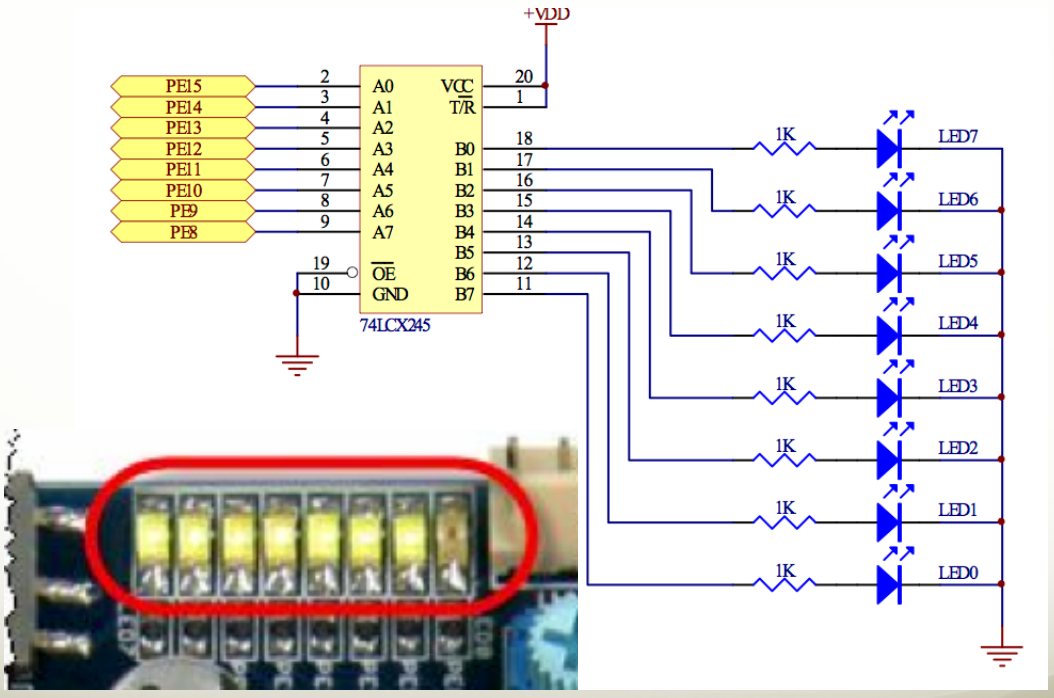


Joy Stick (5-way)

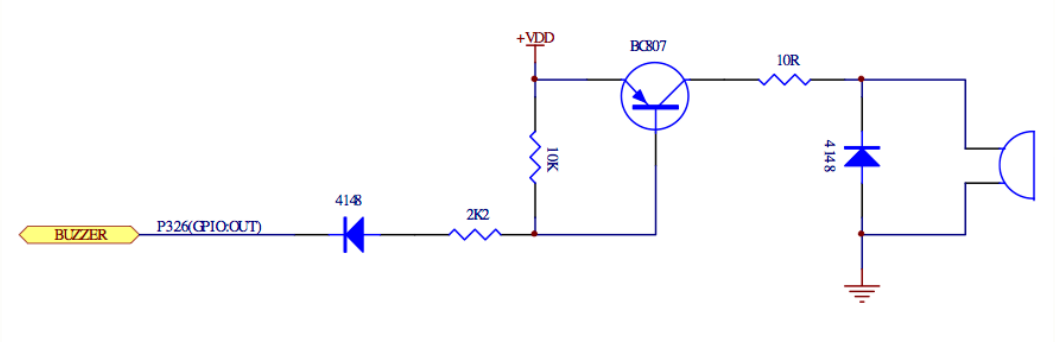


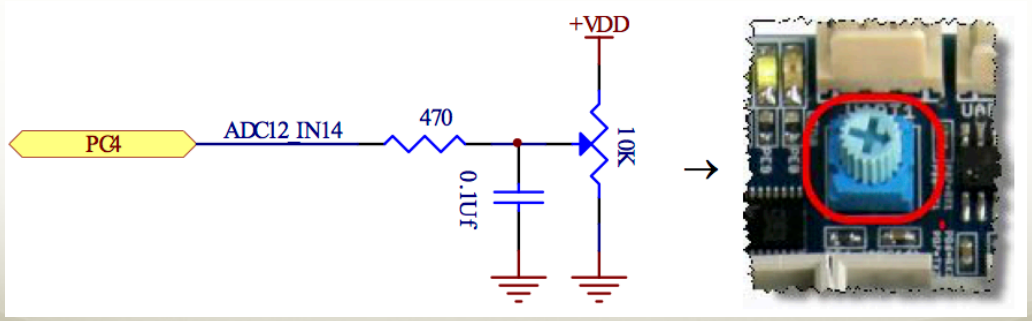


8 Light Emitting Diodes (LEDs)

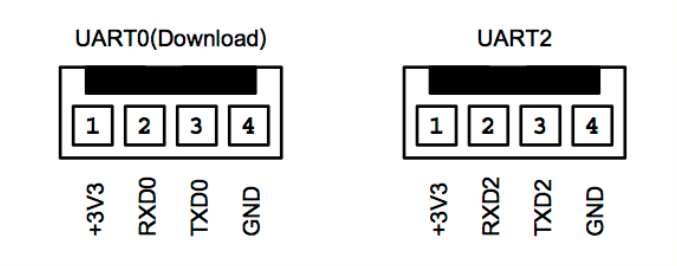


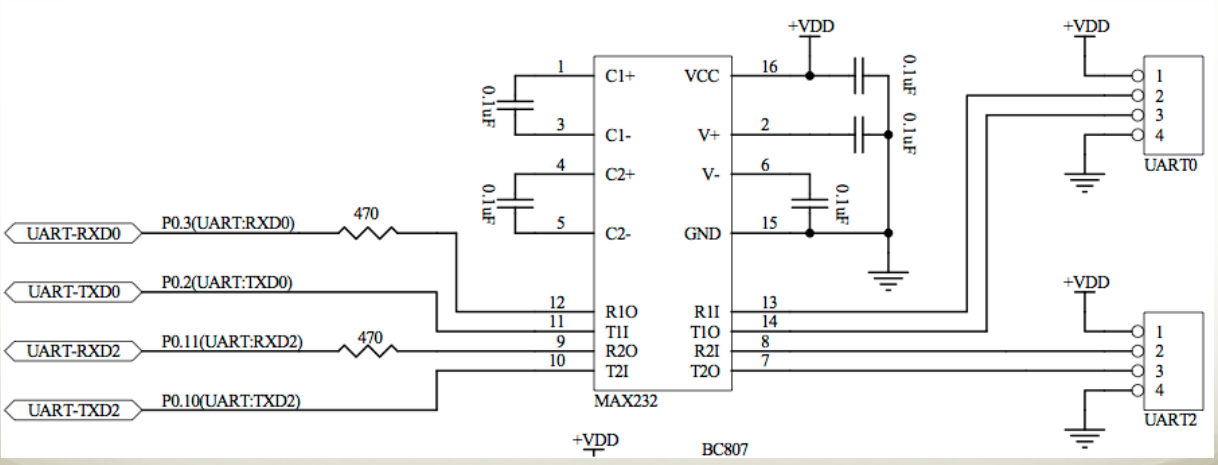
Buzzer and ADC (Volume Control)



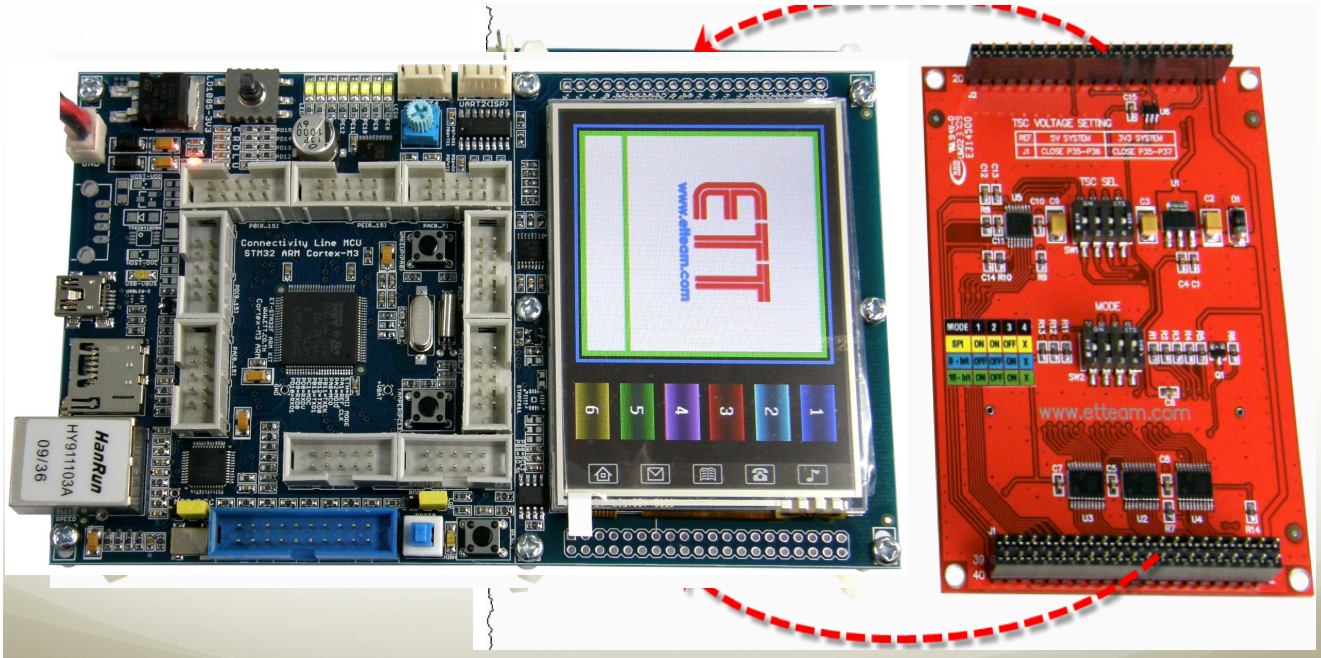


UART x2

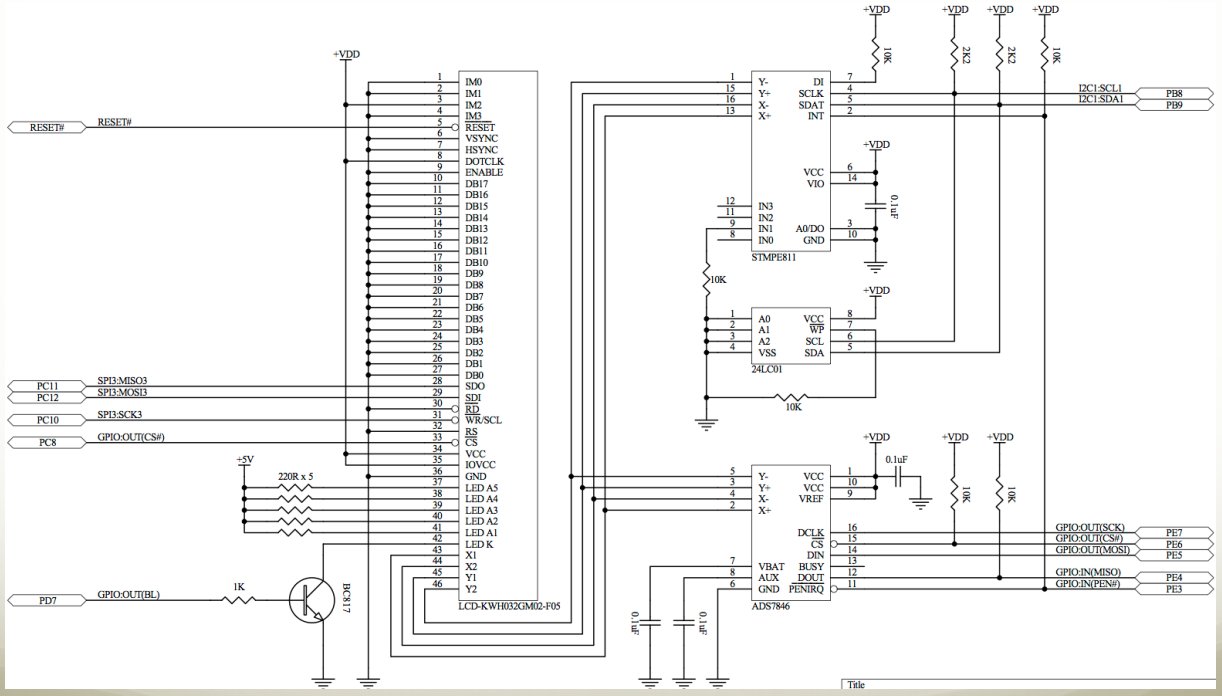




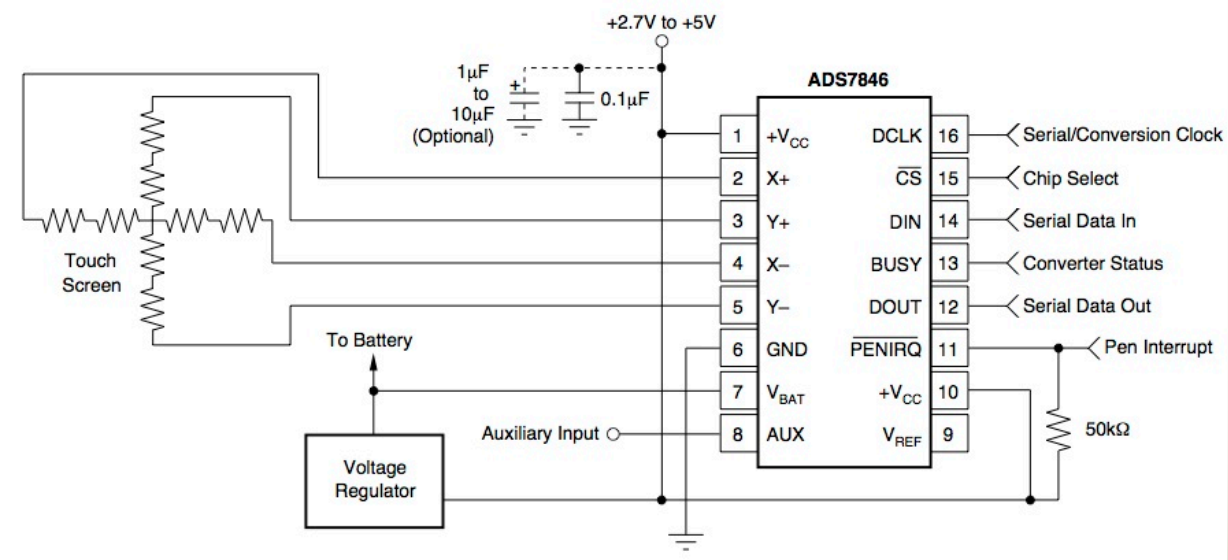
LCD Module



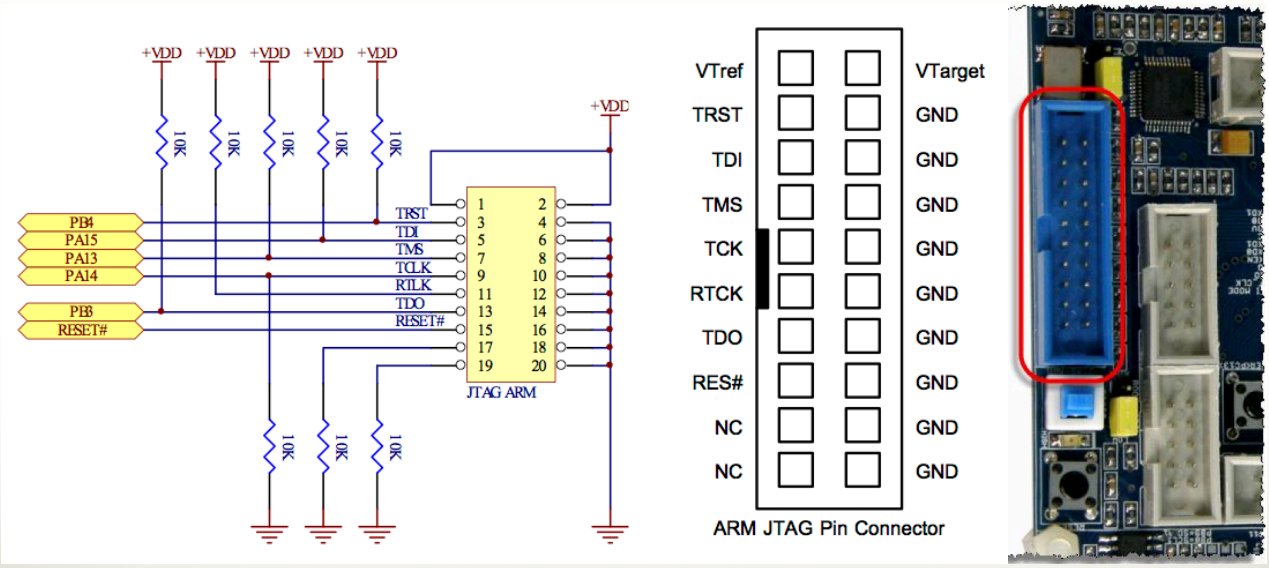
LCD Module



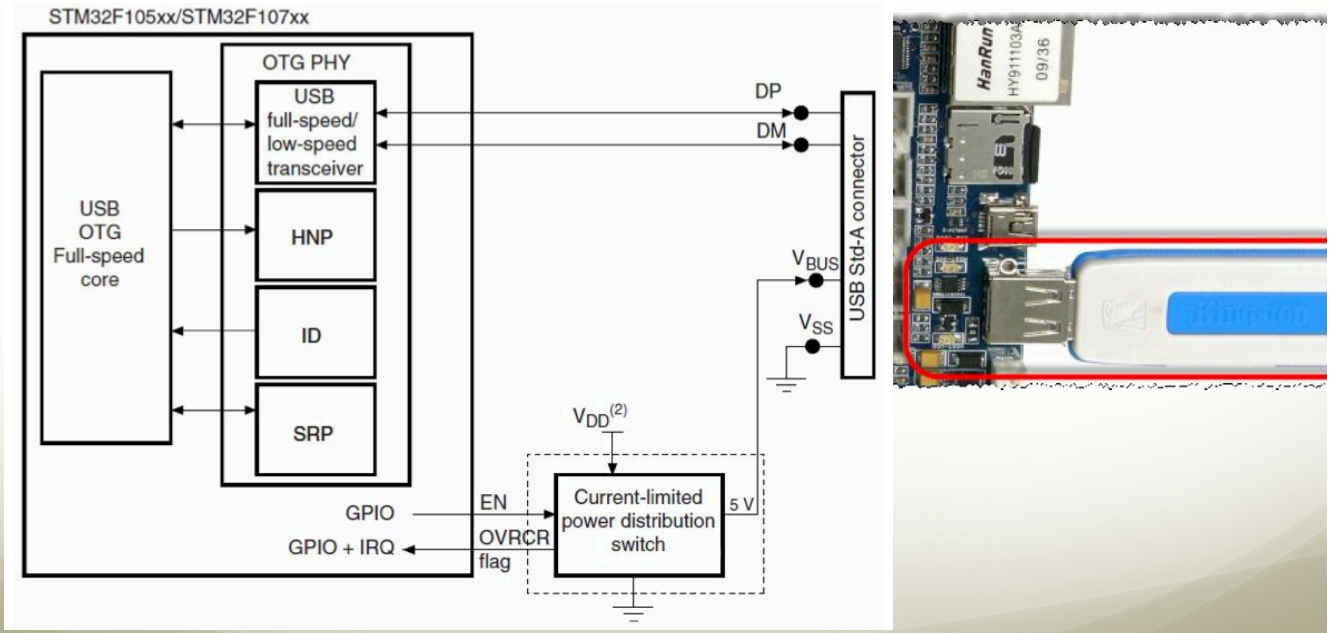
Touch Screen Controller ADS7846 Circuit Diagram



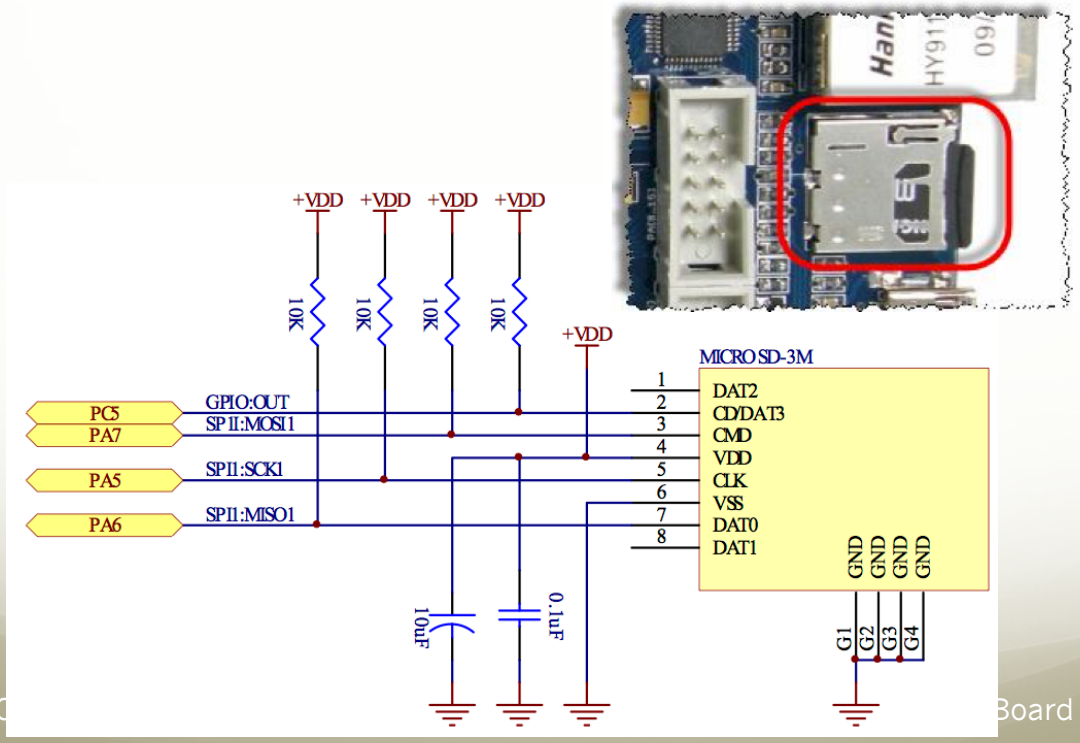
JTAG Connector



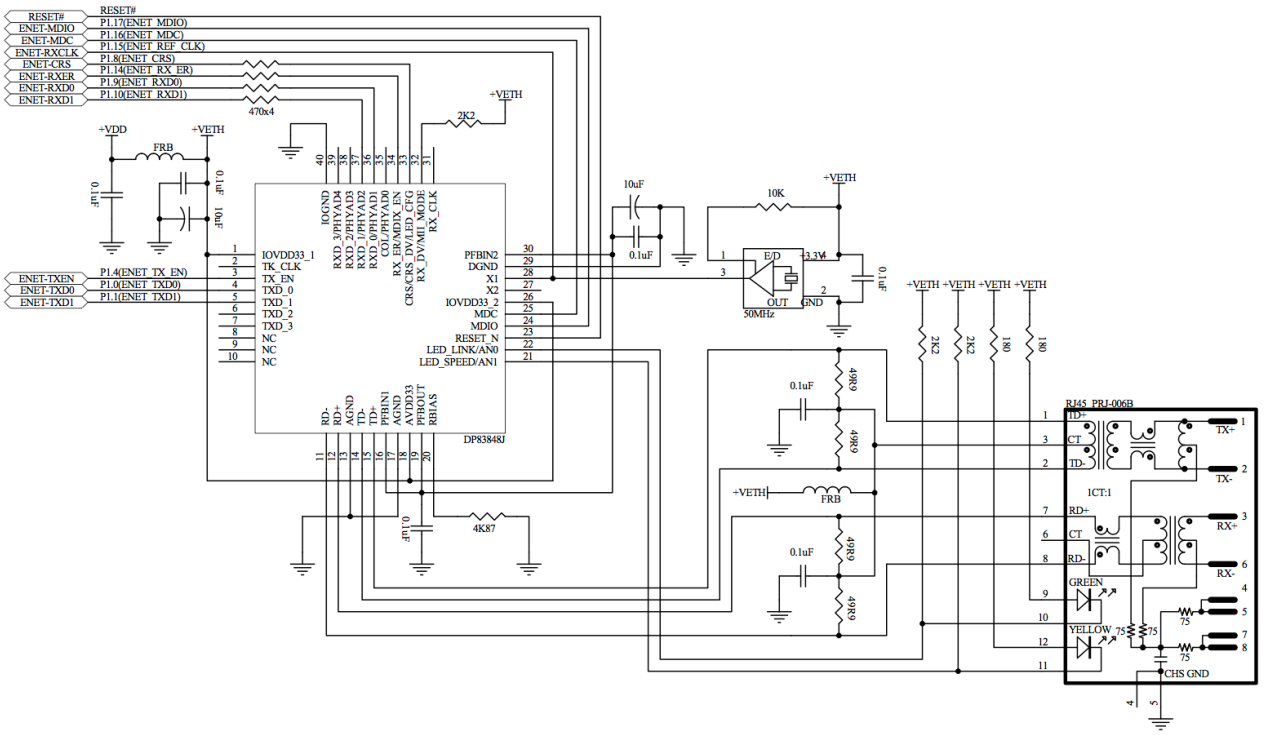
USB Device Port



Secure Digital Memory Slot

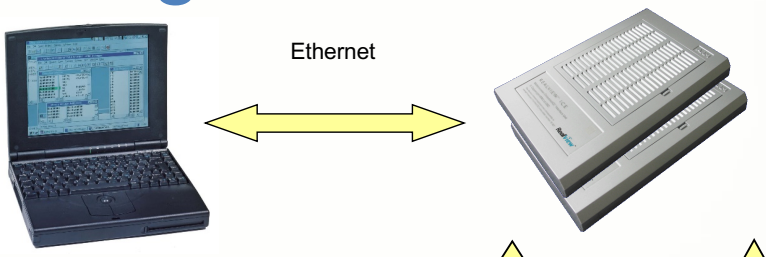


Ethernet Circuit Diagram

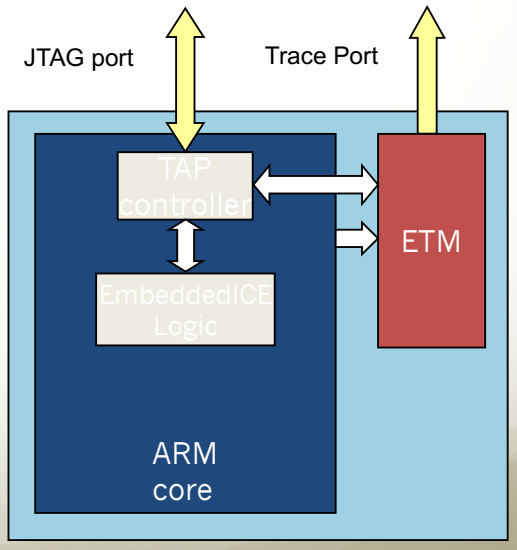


ARM Debug Architecture

Debugger (+ optional trace tools)



- EmbeddedICE Logic
  - Provides breakpoints and processor/system access
- JTAG interface (ICE)
  - Converts debugger commands to JTAG signals
- Embedded trace Macrocell (ETM)
  - Compresses real-time instruction and data access trace
  - Contains ICE features (trigger & filter logic)
- Trace port analyzer (TPA)
  - Captures trace in a deep buffer



Keil Development Tools for ARM

- Includes ARM macro assembler, compilers (ARM RealView C/C++ Compiler, Keil CARM Compiler, or GNU compiler), ARM linker, Keil uVision Debugger and Keil uVision IDE
- Keil uVision Debugger accurately simulates on-chip peripherals (I^2C, CAN, UART, SPI, Interrupts, I/O Ports, A/D and D/A converters, PWM, etc.)
- Evaluation Limitations
  - 16K byte object code + 16K data limitation
  - Some linker restrictions such as base addresses for code/constants
  - GNU tools provided are not restricted in any way
- http://www.keil.com/demo/

Keil Development Tools for ARM

Hello - µVision3 - [C:\Keil\ARM\Examples\Hello\Hello.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

LPC2100

Project Workspace

Register	Value
Current	
R0	0x0000000c
R1	0x0000000c
R2	0x00000020
R3	0x0000018d
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000

Symbols

Mask: * Case Sensitive

Name	Type
Simulator VTREG	
Peripheral SFR	
ALDOM	uchar
ALDOW	uchar
ALDOY	ushort
ALHOUR	uchar
ALMIN	uchar
ALMON	uchar
ALSEC	uchar

```
01 /*****************************************************************************/
02 /*  This file is part of the uVision/ARM development tools                  */
03 /*  Copyright KEIL ELEKTRONIK GmbH 2002-2004                                */
04 /*****************************************************************************/
05 /*                                                                           */
06 /*  HELLO.C:  Hello World Example                                            */
07 /*                                                                           */
08 /*****************************************************************************/
09
10 #include <stdio.h>                /* prototype declarations for I/O functions */
11 #include <LPC21xx.H>              /* LPC21xx definitions                      */
12
13
14 /****************/
15 /* main program */
16 /****************/
17 int main (void)  {                /* execution starts here                    */
18
19   /* initialize the serial interface   */
20   PINSEL0 = 0x00050000;          /* Enable RxD1 and TxD1                     */
21   U1LCR = 0x83;                  /* 8 bits, no Parity, 1 Stop bit            */
22   U1DLL = 97;                    /* 9600 Baud Rate @ 15MHz VPB Clock         */
23   U1LCR = 0x03;                  /* DLAB = 0                                 */
24
25   printf ("Hello World\n");      /* the 'printf' function call               */
26
27   while (1) {                    /* An embedded program does not stop and    */
```

Hello.c | Startup.s | Disassembly | Serial #2

```
MISSING DEVICE (R003: SECURITY KEY NOT FOUND)
Running in Eval Mode
Load "C:\\Keil\\ARM\\Examples\\Hello\\Obj\\Hello.ELF"

*** Restricted Version with 16384 Byte Code Size Limi
*** Currently used: 1980 Bytes (12%)

ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Output Window

Address: 0x4000

```
0x00004000: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000400D: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000401A: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004027: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004034: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00004041: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000404E: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000405B: 00 00 00 00 00 00 00 00 00 00 00 00 00
```

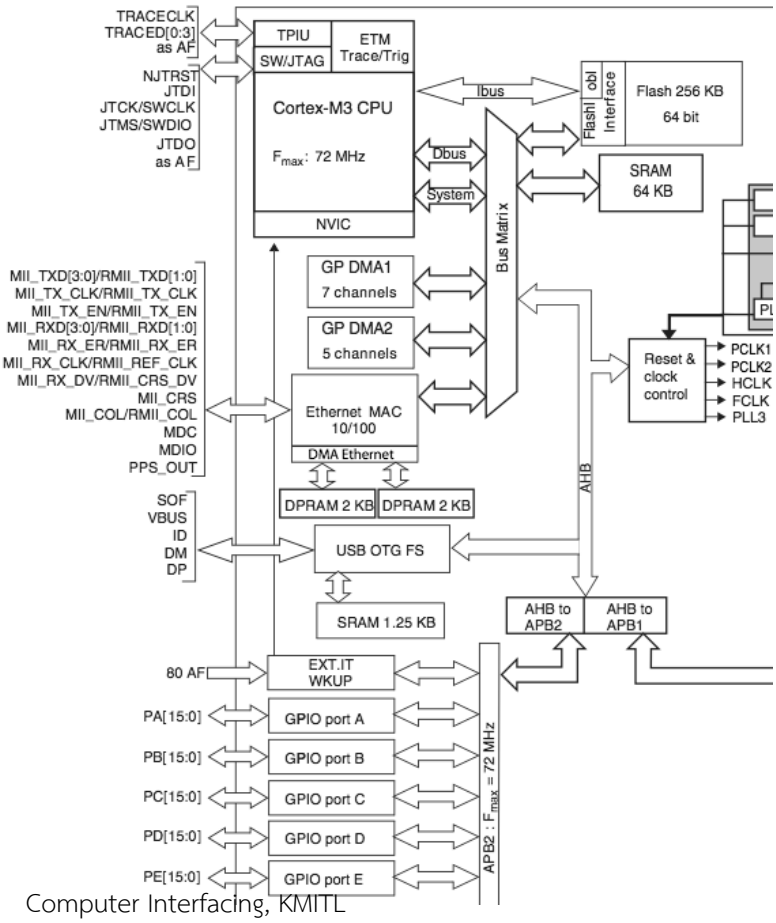
Memory

Memory #1 | Memory #2 | Memory #3 | Memory #4

Ready | Simulation | t1: 0.72642057 sec | L:29 C:1

ARM Cortex M3 & General Purpose Input/Output (GPIO)

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



STM32F10x & GPIO

- GPIO Port A – E
- Each port is 16 pins
- PA[0] – PA[15]
- PB[0] – PB[15]
- …
- PE[0] – PE[15]

Memory and Bus Architecture

- Three masters:

  Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)

  GP-DMA1 (general-purpose DMA)

- Three slaves:

  Internal SRAM

  Internal Flash memory

  AHB to APB bridges (AHB to APBx), which connect all the APB peripherals

Definitions

- ICode bus

  Connects the instruction bus of the Cortex™-M3 core to the Flash memory instruction interface. Instruction fetches are performed on this bus.

- DCode bus

  Connects the DCode bus (literal load and debug access) of the

- Cortex™-M3 core to the Flash memory data interface

- System bus

  Connects the system bus of the Cortex™-M3 core (peripherals bus) to a bus matrix which manages the arbitration between the core and the DMA

Definitions

- DMA bus

  Connects the AHB master interface of the DMA to the bus matrix which manages the access of CPU DCode and DMA to the SRAM, Flash memory and peripherals

- Bus matrix

  Manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a round robin algorithm

- AHB/APB bridges (APB)

  The two AHB/APB bridges provide full synchronous connections between the AHB and the two APB buses

  APB buses operate at full speed (up to 24 MHz)

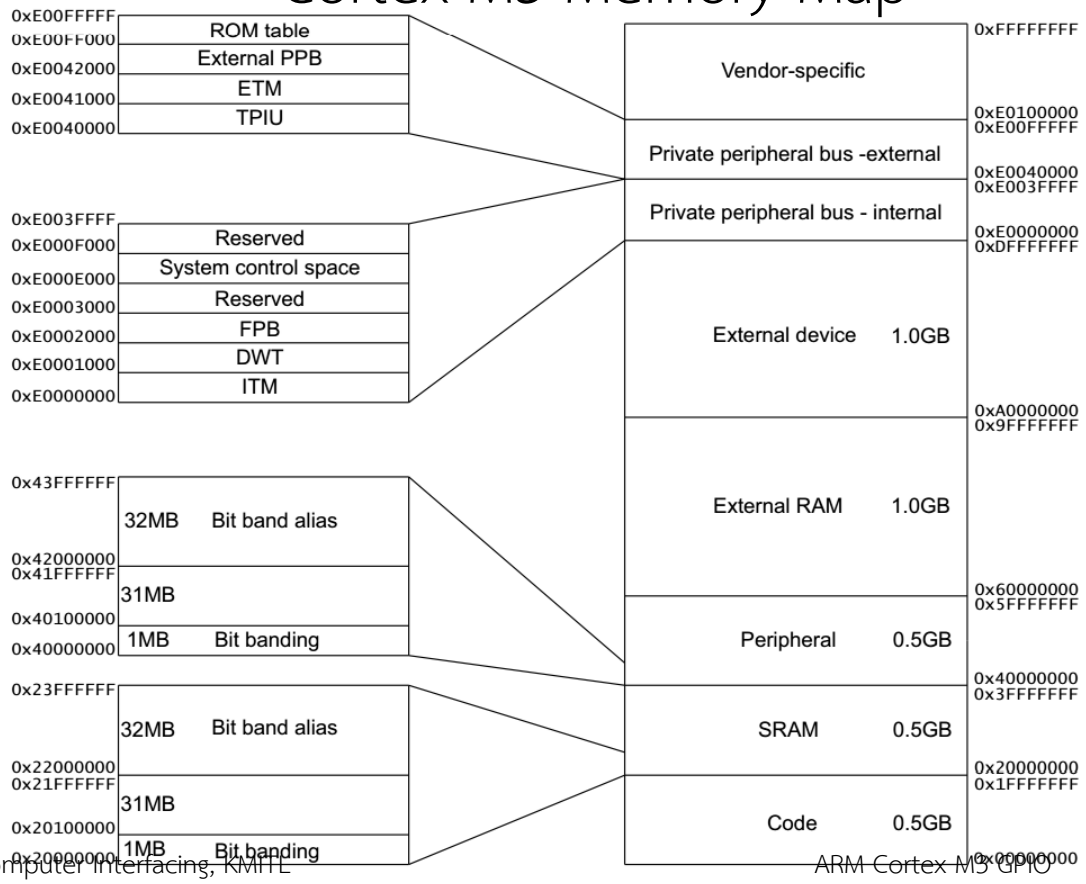
Memory Organization

- Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space
- Bytes are coded in memory in *little endian* format

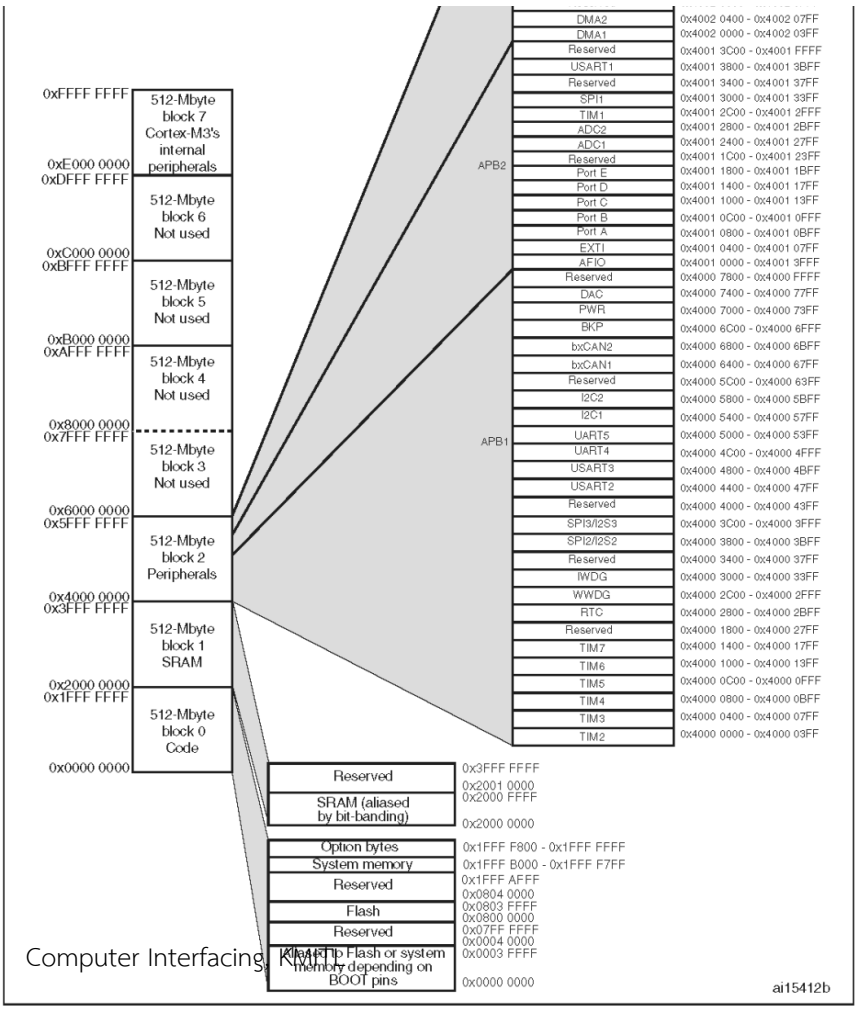
  The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte, the most significant

- Addressable memory space is divided into 8 main blocks, each of 512 MB

Cortex M3 Memory Map



Memory Map



Memory Map

Boundary address	Peripheral	Bus
0x4002 3000 - 0x4002 33FF	CRC	AHB
0x4002 2400 - 0x4002 2FFF	Reserved	
0x4002 2000 - 0x4002 23FF	Flash memory interface	
0x4002 1400 - 0x4002 1FFF	Reserved	
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC	
0x4002 0400 - 0x4002 0FFF	Reserved	
0x4002 0000 - 0x4002 03FF	DMA1	
0x4001 4C00 - 0x4001 FFFF	Reserved	APB2
0x4001 4800 - 0x4001 4BFF	TIM17 timer	
0x4001 4400 - 0x4001 47FF	TIM16 timer	
0x4001 4000 - 0x4001 43FF	TIM15 timer	
0x4001 3C00 - 0x4001 3FFF	Reserved	
0x4001 3800 - 0x4001 3BFF	USART1	
0x4001 3400 - 0x4001 37FF	Reserved	
0x4001 3000 - 0x4001 33FF	SPI1	
0x4001 2C00 - 0x4001 2FFF	TIM1 timer	
0x4001 2800 - 0x4001 2BFF	Reserved	
0x4001 2400 - 0x4001 27FF	ADC1	
0x4001 1C00 - 0x4001 23FF	Reserved	
0x4001 1800 - 0x4001 1BFF	GPIO Port E	
0x4001 1400 - 0x4001 17FF	GPIO Port D	
0x4001 1000 - 0x4001 13FF	GPIO Port C	
0x4001 0C00 - 0x4001 0FFF	GPIO Port B	
0x4001 0800 - 0x4001 0BFF	GPIO Port A	
0x4001 0400 - 0x4001 07FF	EXTI	
0x4001 0000 - 0x4001 03FF	AFIO	

Boundary address	Peripheral	Bus
0x4000 7C00 - 0x4000 FFFF	Reserved	APB1
0x4000 7800 - 0x4000 7BFF	CEC	
0x4000 7400 - 0x4000 77FF	DAC	
0x4000 7000 - 0x4000 73FF	Power control PWR	
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)	
0x4000 5C00 - 0x4000 6BFF	Reserved	
0x4000 5800 - 0x4000 5BFF	I2C2	
0x4000 5400 - 0x4000 57FF	I2C1	
0x4000 4C00 - 0x4000 53FF	Reserved	
0x4000 4800 - 0x4000 4BFF	USART3	
0x4000 4400 - 0x4000 47FF	USART2	
0x4000 3C00 - 0x4000 3FFF	Reserved	
0x4000 3800 - 0x4000 3BFF	SPI2	
0x4000 3400 - 0x4000 37FF	Reserved	
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)	
0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)	
0x4000 2800 - 0x4000 2BFF	RTC	
0x4000 1800 - 0x4000 27FF	Reserved	
0x4000 1400 - 0x4000 17FF	TIM7 timer	
0x4000 1000 - 0x4000 13FF	TIM6 timer	
0x4000 0C00 - 0x4000 0FFF	Reserved	
0x4000 0800 - 0x4000 0BFF	TIM4 timer	
0x4000 0400 - 0x4000 07FF	TIM3 timer	
0x4000 0000 - 0x4000 03FF	TIM2 timer	

Memory Organization

- Embedded Flash memory organization

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 31	0x0800 7C00 - 0x0800 8000	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Memory Organization

- Embedded SRAM

  The STM32F100xx features up to 32 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000

- Bit banding

  The Cortex™-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

  In the STM32F100xx, both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed.

General-Purpose I/Os (GPIOs)

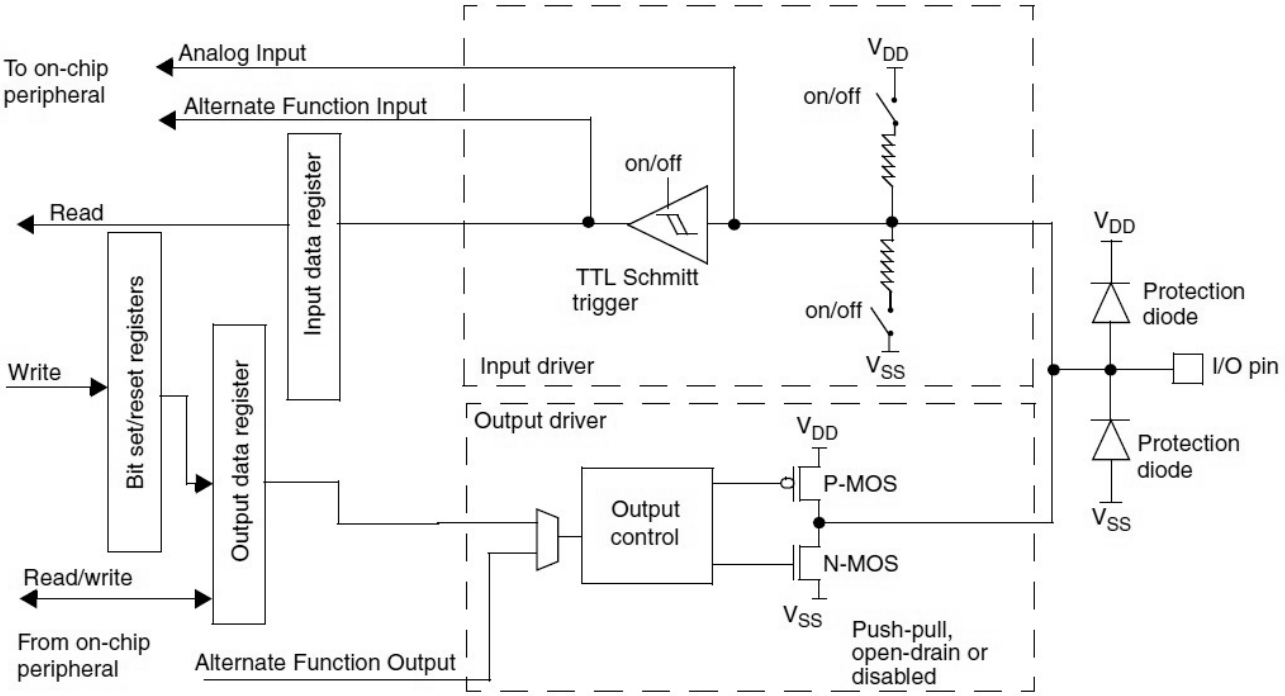
- Each of the general-purpose I/O ports has:
  - Two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH)
  - Two 32-bit data registers (GPIOx_IDR, GPIOx_ODR)
  - 32-bit set/reset register (GPIOx_BSRR)
  - 16-bit reset register (GPIOx_BRR)
  - 32-bit locking register (GPIOx_LCKR)
- Each port bit of GPIOs can be individually configured by software in several modes:
  - Input floating
  - Input pull-up
  - Input-pull-down
  - Analog
  - Output open-drain
  - Output push-pull
  - Alternate function push-pull
  - Alternate function open-drain

Pin Mapping Table

Table 5. Pin definitions (continued)

Pins			Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
BGA100	LQFP64	LQFP100					Default	Remap
A8	50	77	PA15	I/O	FT	JTDI	SPI3_NSS / I2S3_WS	TIM2_CH1_ETR / PA15 SPI1_NSS
B9	51	78	PC10	I/O	FT	PC10	UART4_TX	USART3_TX/ SPI3_SCK/I2S3_CK
B8	52	79	PC11	I/O	FT	PC11	UART4_RX	USART3_RX/ SPI3_MISO
C8	53	80	PC12	I/O	FT	PC12	UART5_TX	USART3_CK/ SPI3_MOSI/I2S3_SD
-	-	81	PD0	I/O	FT	PD0	-	OSC_IN(9)/CAN1_RX
-	-	82	PD1	I/O	FT	PD1	-	OSC_OUT(9)/CAN1_TX
B7	54	83	PD2	I/O	FT	PD2	TIM3_ETR / UART5_RX	
C7	-	84	PD3	I/O	FT	PD3	-	USART2_CTS
D7	-	85	PD4	I/O	FT	PD4	-	USART2_RTS
B6	-	86	PD5	I/O	FT	PD5	-	USART2_TX
C6	-	87	PD6	I/O	FT	PD6	-	USART2_RX
D6	-	88	PD7	I/O	FT	PD7	-	USART2_CK

Basic Structure of GPIO Bit



GPIO Bit Configuration Table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01		0 or 1
	Open-drain		1	10		0 or 1
Alternate Function output	Push-pull	1	0	11		don't care
	Open-drain		1	See Table on following page		don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

Note: During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1:0]=01b, MODEx[1:0]=00b)

GPIO Configuration: Output MODE Bits

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

GPIO Operation

- During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1:0]=01b, MODEx[1:0]=00b)
- When configured as output, the value written to the Output Data register (GPIOx_ODR) is output on the I/O pin.

  It is possible to use the output driver in Push-Pull mode or Open-Drain mode (only the N-MOS is activated when outputting 0).
- The Input Data register (GPIOx_IDR) captures the data present on the I/O pin at every APB2 clock cycle
- All GPIO pins have an internal weak pull-up and weak pull- down which can be activated or not when configured as input

GPIO Atomic Bit Set or Reset

- Atomic Read/Modify access

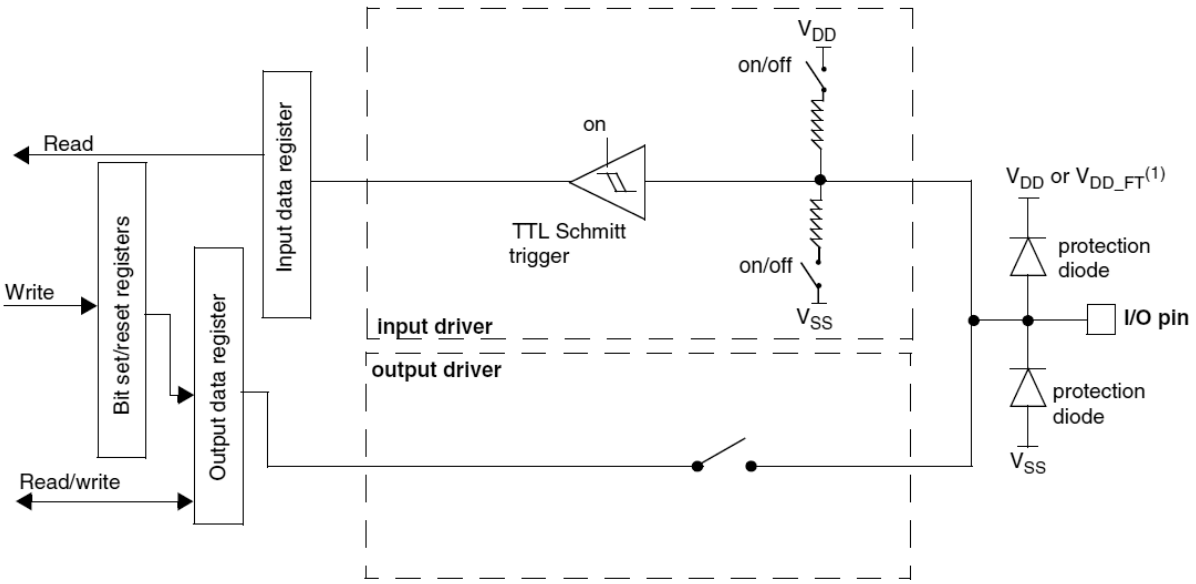
  No interruption in the middle to cause errors
- Atomic operations ensure that the desired change is not interrupted resulting in partial set/reset of GPIOs
- There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level:
- It is possible to modify only one or several bits in a single atomic APB2 write access
- This is achieved by programming to '1' the Bit Set/Reset Register (GPIOx_BSRR, or for reset only GPIOx_BRR) to select the bits you want to modify.

  Unselected bits will not be modified

Input Configuration

- When the I/O Port is programmed as Input:
  - The Output Buffer is disabled
  - The Schmitt Trigger Input is activated
  - The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating)
  - The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
  - A read access to the Input Data Register obtains the I/O State

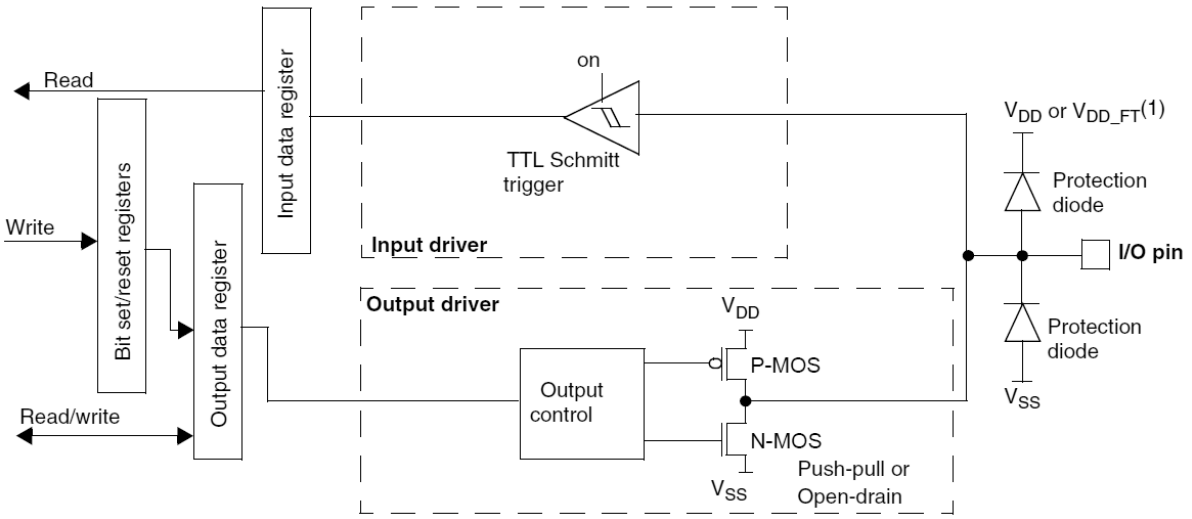
Input Configuration



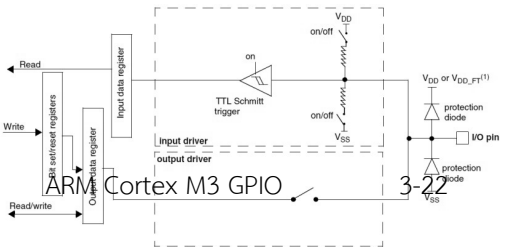
Output Configuration

- When the I/O Port is programmed as Output:
  - The Output Buffer is enabled:
    - Open Drain Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z. (the P-MOS is never activated)
    - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register activates the P-MOS
  - The Schmitt Trigger Input is activated.
  - The weak pull-up and pull-down resistors are disabled.
  - The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
  - Read access to Input Data Register gets the I/O state in open drain mode
  - Read access to Output Data register gets last written value in Push-Pull mode

Output Configuration



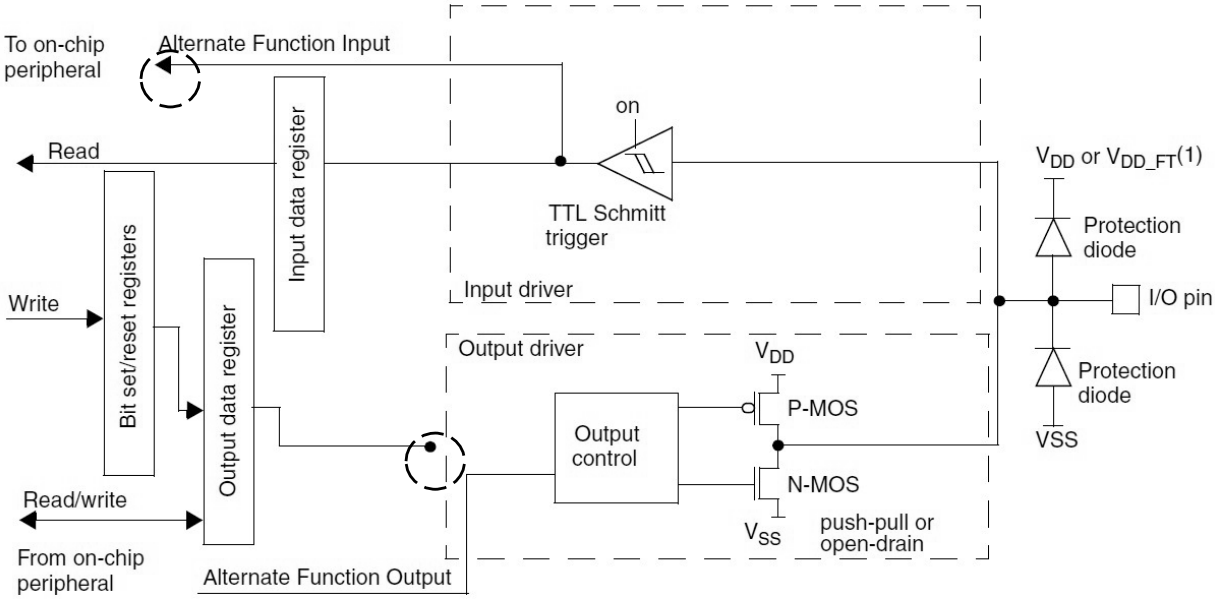
Compare to input configuration



Alternate Function Configuration

- When the I/O Port is programmed as Alternate Function:
  - The Output Buffer is turned on in Open Drain or Push-Pull configuration
  - The Output Buffer is driven by the signal coming from the peripheral (alternate function out)
  - The Schmitt Trigger Input is activated
  - The weak pull-up and pull-down resistors are disabled
  - The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
  - A read access to the Input Data Register gets the I/O state in open drain mode
  - A read access to the Output Data register gets the last written value in Push-Pull mode

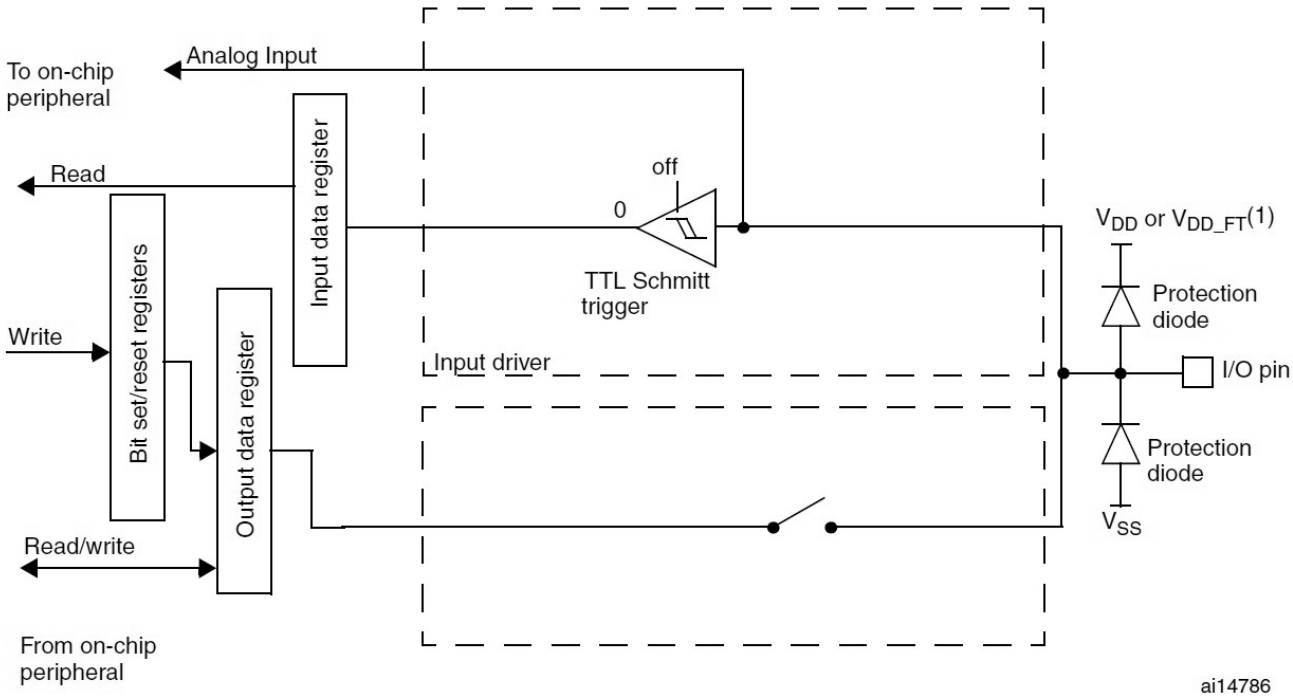
Alternate Function Configuration



Analog Configuration

- When the I/O Port is programmed as Analog configuration:
  - The Output Buffer is disabled.
  - The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin.
  - The output of the Schmitt Trigger is forced to a constant value (0).
  - The weak pull-up and pull-down resistors are disabled.
  - Read access to the Input Data Register gets the value “0”.

Analog I/O Configuration



GPIO Registers (Config Reg Low & High)

9.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

4 LSB of GPIOx_CRL (Config Reg Low)

3	2	1	0
CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw

CNFy[1:0]: Port x configuration bits (y= 0 .. 7)

In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved

In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

MODEy[1:0]: Port x mode bits (y= 0 .. 7)
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

I/P Data Register of Port x (A-E)

9.2.3 Port input data register (GPIOx_IDR) (x=A..G)

Address offset: 0x08h

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data (y= 0 .. 15)

These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

O/P Data Register of Port x (A-E)

9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).

Set/Reset Bit Register of Port x (A-E)

9.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

BSy: Port x *Set bit y (y= 0 .. 15)*

These bits are write-only and can be accessed in Word r

0: No action on the corresponding ODRx bit

1: Set the corresponding ODRx bit

Reset Bit Register of Port x (A-E)

9.2.6 Port bit reset register (GPIOx_BRR) (x=A..G)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

Config Lock Register of Port x (A-E)

9.2.7 Port configuration lock register (GPIOx_LCKR) (x=A..G)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit it is no longer possible to modify the value of the port bit until the next reset.

Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH).

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Config Lock Register of Port x (A-E)

Bits 31:17 Reserved

Bit 16 **LCKK[16]:** Lock key

This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence.
0: Port configuration lock key not active
1: Port configuration lock key active. GPIOx_LCKR register is locked until an MCU reset occurs.

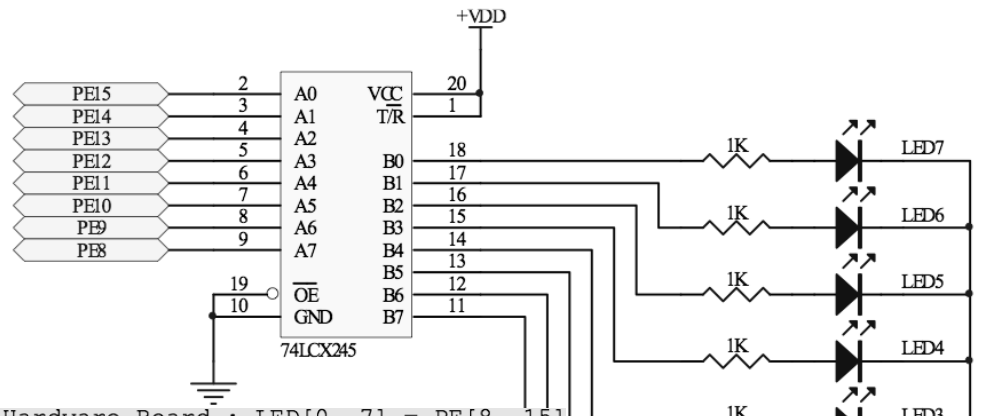
LOCK key writing sequence:
Write 1
Write 0
Write 1
Read 0
Read 1 (this read is optional but confirms that the lock is active)

Note: During the LOCK Key Writing sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence will abort the lock.

Bits 15:0 **LCKy:** Port x Lock bit y (y= 0 .. 15)

These bits are read write but can only be written when the LCKK bit is 0.
0: Port configuration not locked
1: Port configuration locked.

ETT Board & LEDs (Output)



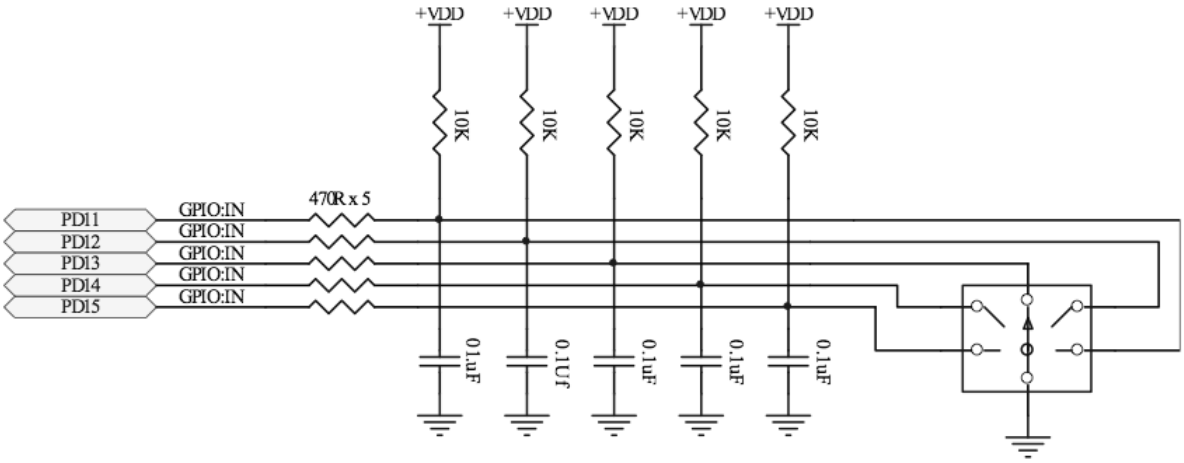

```
// ET-STM32F ARM KIT Hardware Board : LED[0..7] = PE[8..15]
#define LEDn                          8

#define LED0_GPIO_PORT                GPIOE
#define LED0_GPIO_CLK                 RCC_APB2Periph_GPIOE
#define LED0_GPIO_PIN                 GPIO_Pin_8

#define LED1_GPIO_PORT                GPIOE
#define LED1_GPIO_CLK                 RCC_APB2Periph_GPIOE
#define LED1_GPIO_PIN                 GPIO_Pin_9

#define LED2_GPIO_PORT                GPIOE
#define LED2_GPIO_CLK                 RCC_APB2Periph_GPIOE
#define LED2_GPIO_PIN                 GPIO_Pin_10
```


ETT Board & JoySwitch (Input)




```
// Joy Up(PD11)
#define UP_BUTTON_PORT              GPIOD
#define UP_BUTTON_CLK               RCC_APB2Periph_GPIOD
#define UP_BUTTON_PORT_SOURCE       GPIO_PortSourceGPIOD
#define UP_BUTTON_PIN               GPIO_Pin_11

#define UP_BUTTON_PIN_SOURCE        GPIO_PinSource11
#define UP_BUTTON_EXTI_LINE         EXTI_Line11
#define UP_BUTTON_IRQn              EXTI15_10_IRQn
```

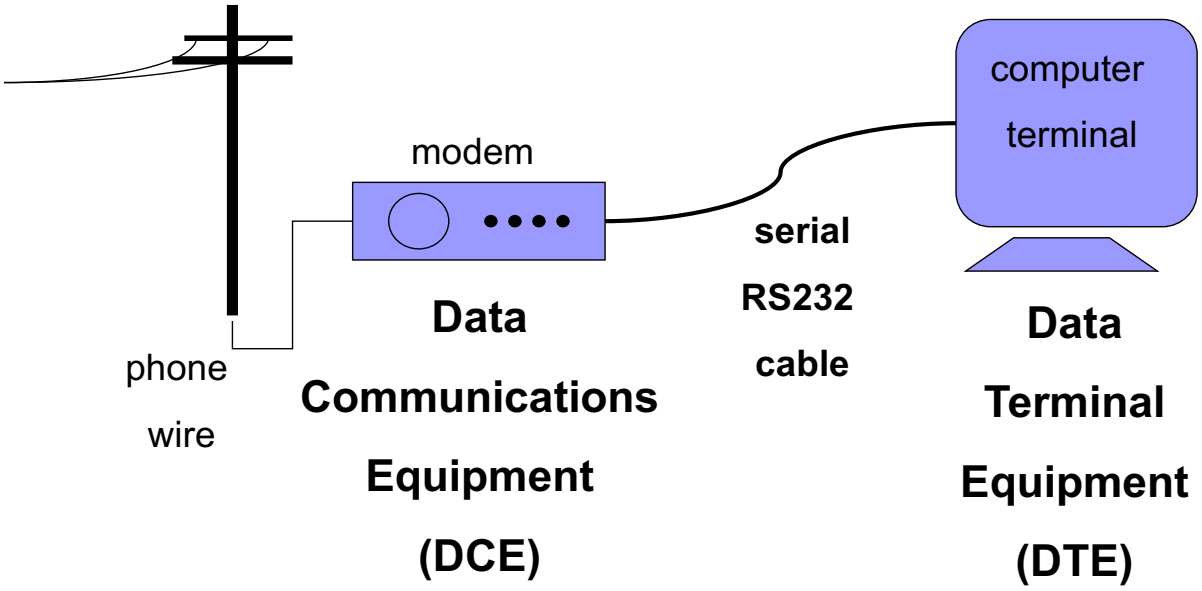
Universal Asynchronous Receiver/Transmitter (UART) & RS232

ผศ.ดร. สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

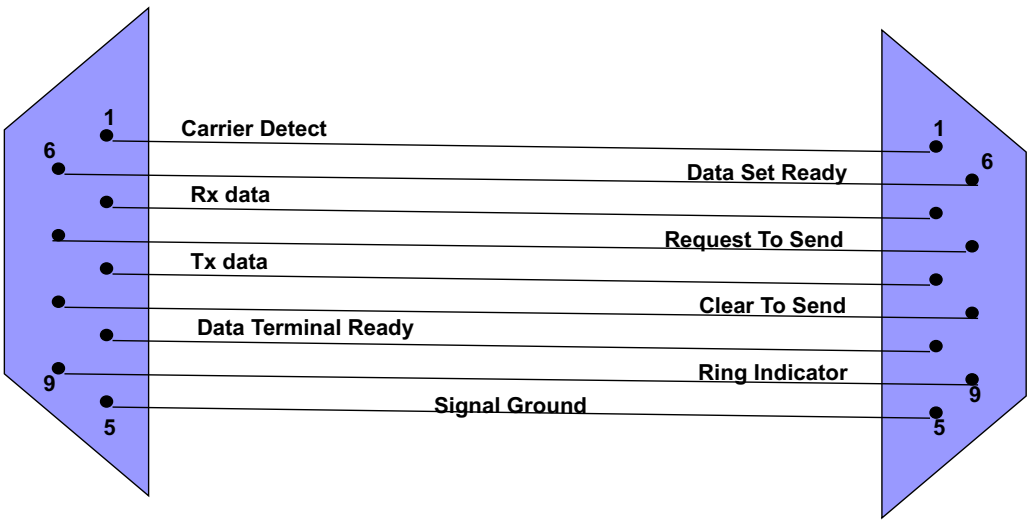
UARTs

- Typically point to point communications
- Normally 8-bit data format with one start and one stop bit
- Standards: E.g. RS232
  - defines connector type, pin assignments, voltage levels, max bit rate, cable length etc.
  - Min. 3 pins – TxD, RxD, Ground
  - Other pins for data flow control.
- Some common RS232 baud rates - 300,1200, 9600, 19200
- Handshaking: None,
  - Hardware - RTS, CTS, etc - simple logic levels
  - Software - Xon/Xoff protocol

PC with a modem



Normal 9-wire serial cable



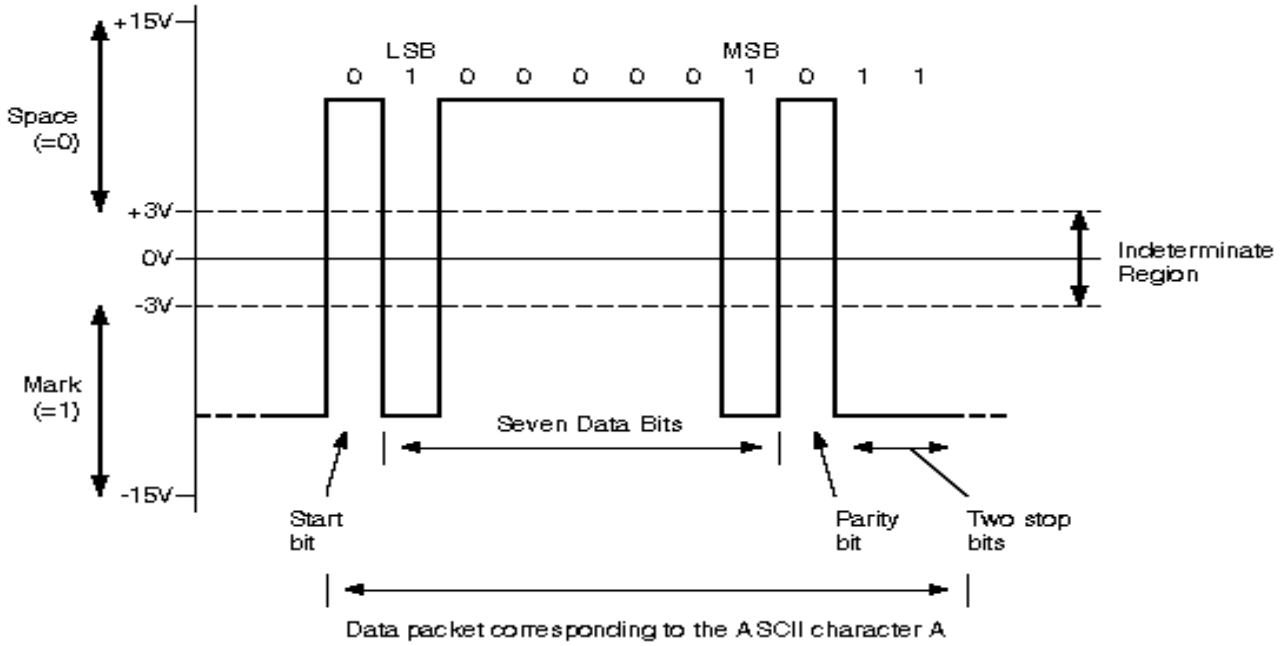
Signal functions

- CD: **Carrier Detect** The modem asserts this signal to indicate that it successfully made its connection to a remote device
- RI: **Ring Indicator** The modem asserts this signal to indicate that the phone is ringing at the other end of its connection
- DSR: **Data Set Ready** Modem to PC
- DTR: **Data Terminal Ready** PC to Modem

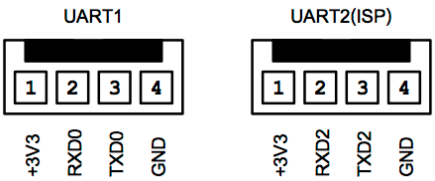
Signal functions (continued)

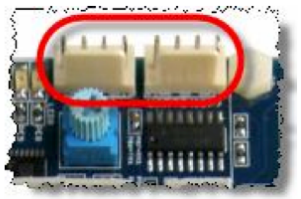
- RTS: **Request To Send** PC is ready for the modem to relay some received data
- CLS: **Clear To Send** Modem is ready for the PC to begin transmitting some data

RS232 Signal Voltage Levels (7 data bits, Even Parity, 2 stop bits)



ETT Board & RS232

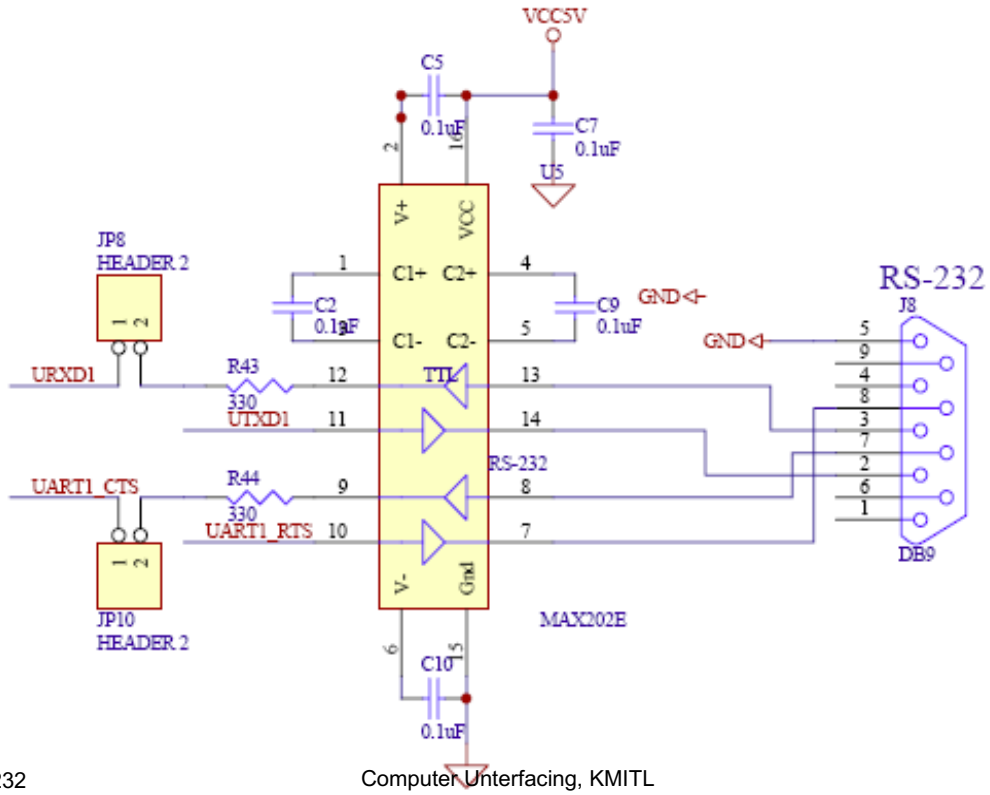




```
// ET-STM32F ARM KIT(STM32F107VCT6) Hardware Kit
// UART1(Remap) = PB7:RX1,PB6:TX1
#define EVAL_COM1           USART1                  //COM1 = USART1
#define EVAL_COM1_GPIO      GPIOB                   //USART1 Port = PB
#define EVAL_COM1_CLK       RCC_APB2Periph_USART1   //Enable USART1 Clock
#define EVAL_COM1_GPIO_CLK  RCC_APB2Periph_GPIOB    //Enable PB Clock
#define EVAL_COM1_RxPin     GPIO_Pin_7              //RX1=PB7
#define EVAL_COM1_TxPin     GPIO_Pin_6              //TX1=PB6

// UART2(Remap) = PD6:RX2,PD5:TX2
#define EVAL_COM2           USART2                  //COM2 = USART2
#define EVAL_COM2_GPIO      GPIOD                   //USART2 Port = PD
#define EVAL_COM2_CLK       RCC_APB1Periph_USART2   //Enable UART2 Clock
#define EVAL_COM2_GPIO_CLK  RCC_APB2Periph_GPIOD    //Enable PD Clock
#define EVAL_COM2_RxPin     GPIO_Pin_6              //RX2=PD6
#define EVAL_COM2_TxPin     GPIO_Pin_5              //TX2=PD5
```

UART (Digital 0-3.3 Volts) <=> RS232 (+/-12 Volts)

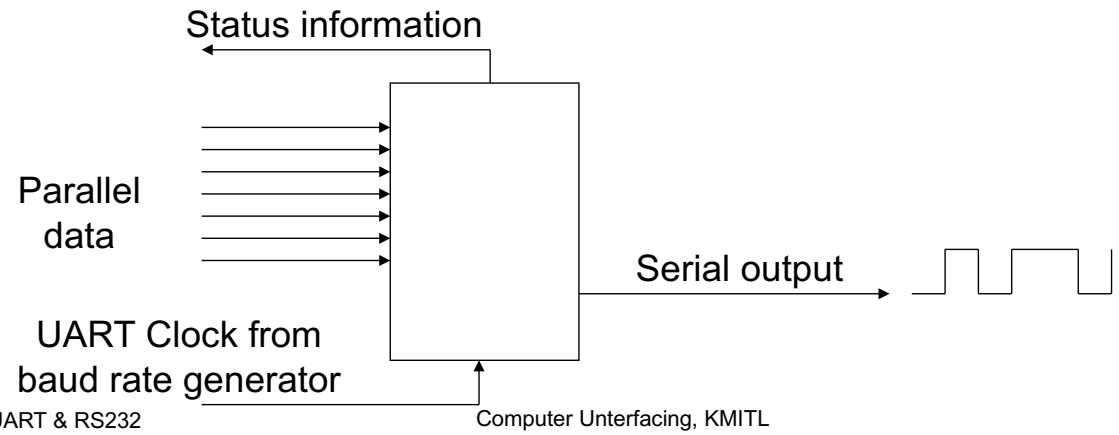


UART (Universal Asynchronous Receiver/Transmitter)

- Most UARTS are full duplex that allows serial output and serial input to take place simultaneously.
- Based on shift registers and a clock signal.
- UART clock determines baud rate (2400 – 112,500 bits/sec)
- UART frames the data bits with
  - a start bit to provide synchronisation to the receiver
  - one or more (usually one) stop bits to signal end of data
- Most UARTs can also optionally generate PARITY bit on transmission and PARITY checking on reception to provide simple error detection.
- UARTs often have receive and transmit buffers(FIFO's) as well as the serial shift registers

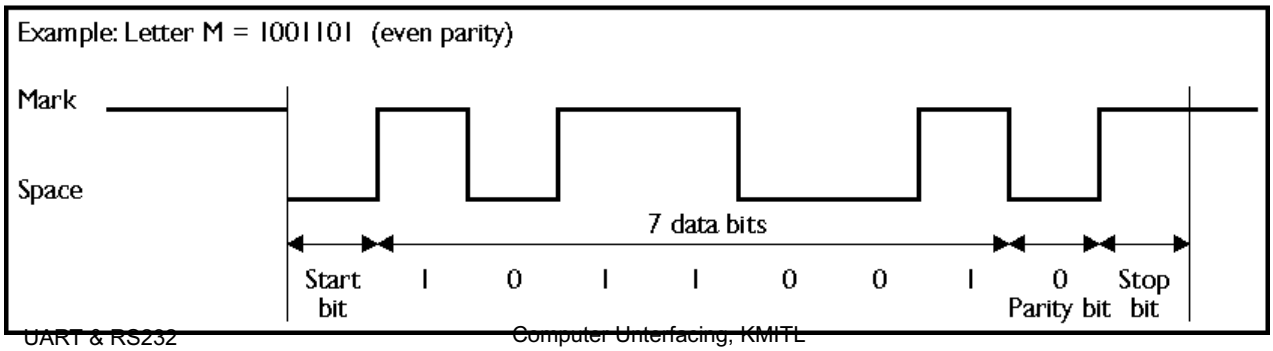
UART - Transmitter

- Transmitter (Tx) - converts data from parallel to serial format
  - inserts start and stop bits
  - calculates and inserts parity bit if required
  - output bit rate is determined by the UART clock



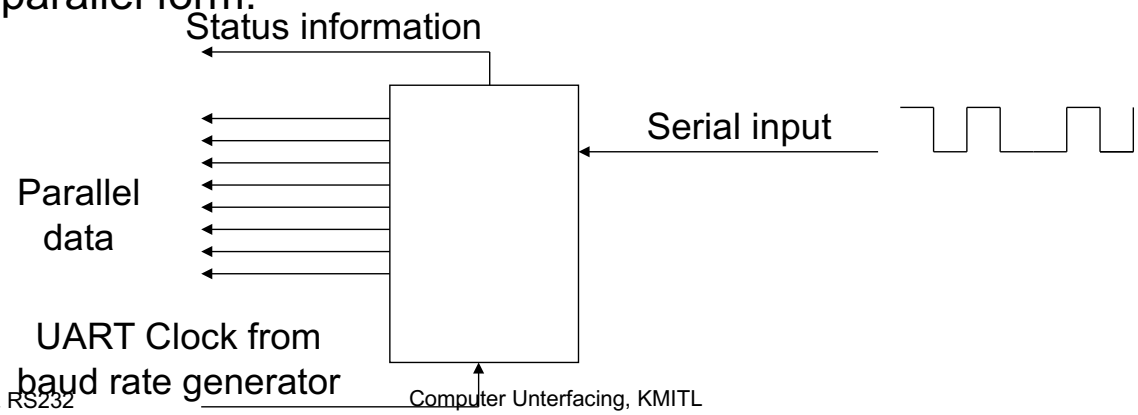
Asynchronous serial transmission

Serial transmission is little endian (least significant bit first)

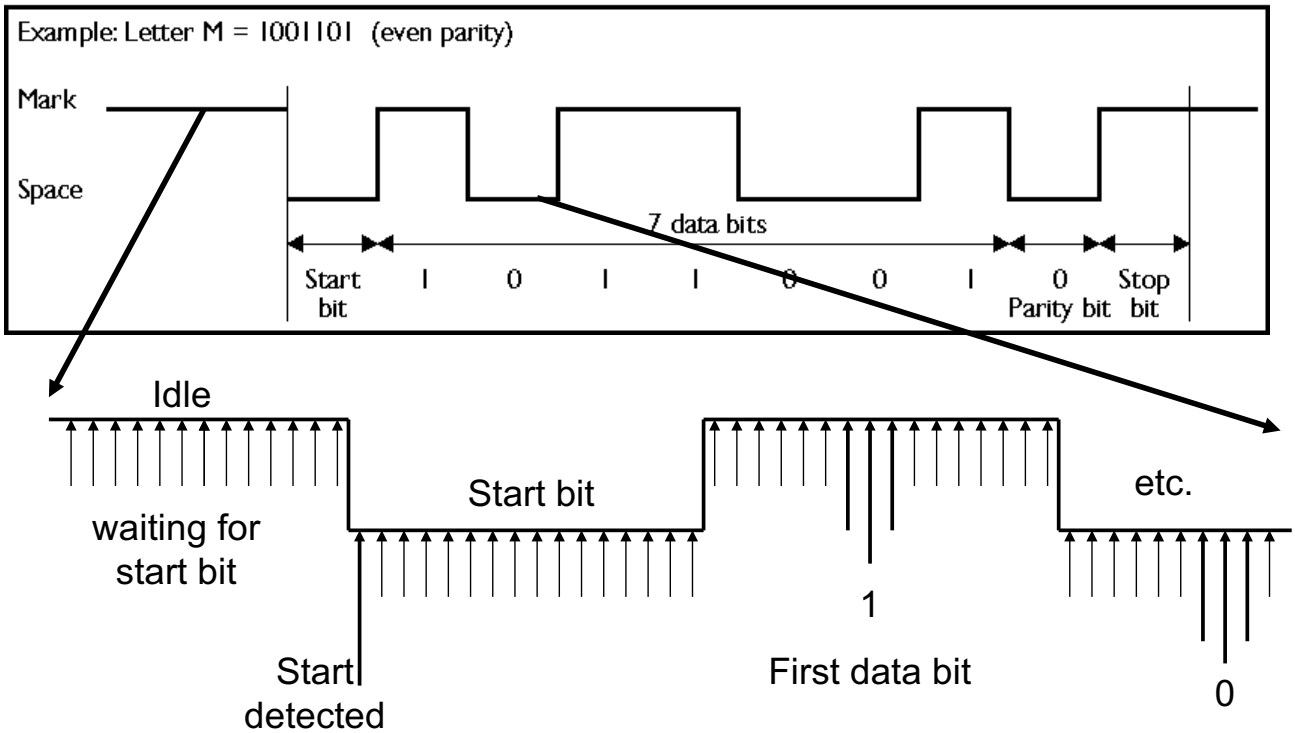


UART - The Receiver

- synchronises with transmitter using the falling edge of the start bit.
- samples the input data line at a clock rate that is normally a multiple of baud rate, typically 16 times the baud rate.
- removes the start and stop bits, optional calculates and checks the parity bit. Presents the received data value in parallel form.



Asynchronous serial reception



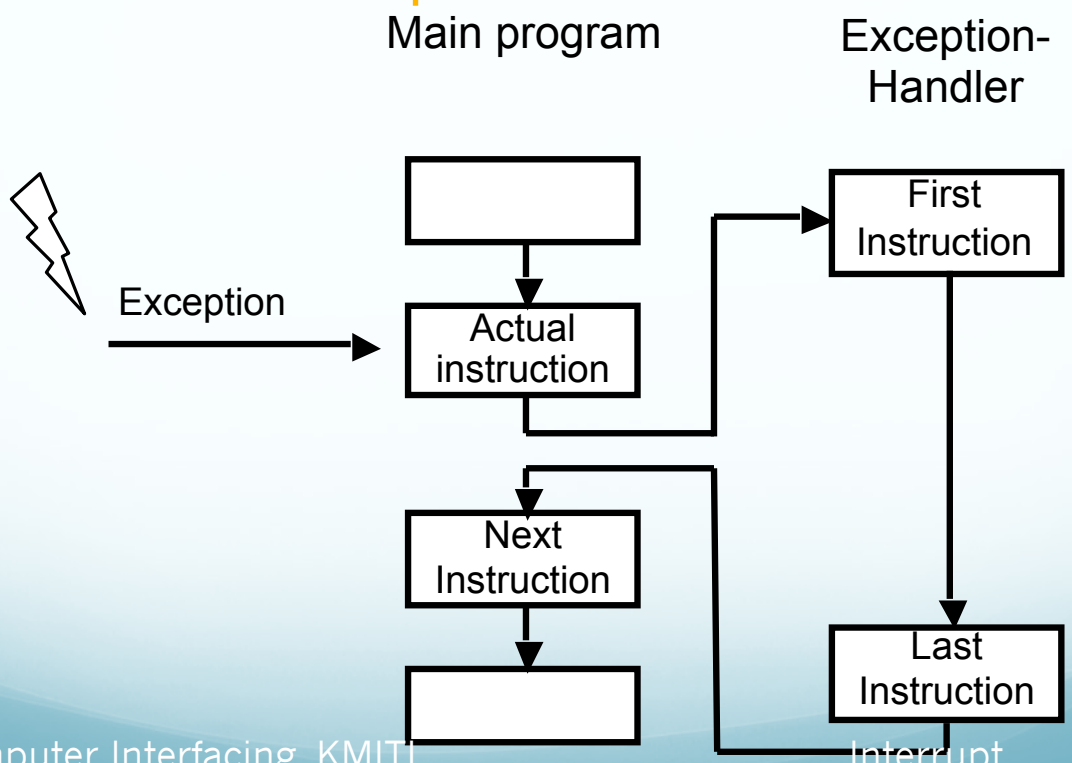
Exceptions and Interrupts ARM Cortex M3

ผศ.ดร. สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Introduction

- Exception are events
  - They occur during the execution of the program
- Type of ARM exceptions
  - Exceptions that result by a command
    - Software- Interrupt
    - Undefined instruction
    - Pre fetch Abort (memory access errors during the command reading)
  - Exceptions that result as a side effect of a command
    - Data Abort (memory access errors during the reading or writing of variables)
  - Externally generated exceptions (asynchronous)
    - Reset
    - Hardware- Interrupts: IRQ

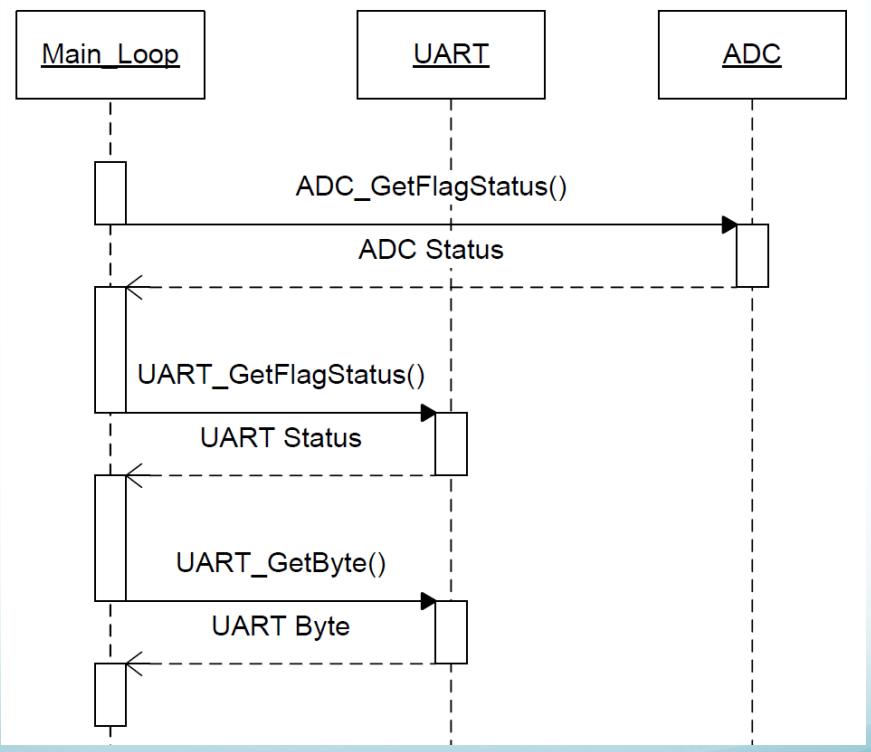
Program execution when an exception occurs



Properties of the exceptions

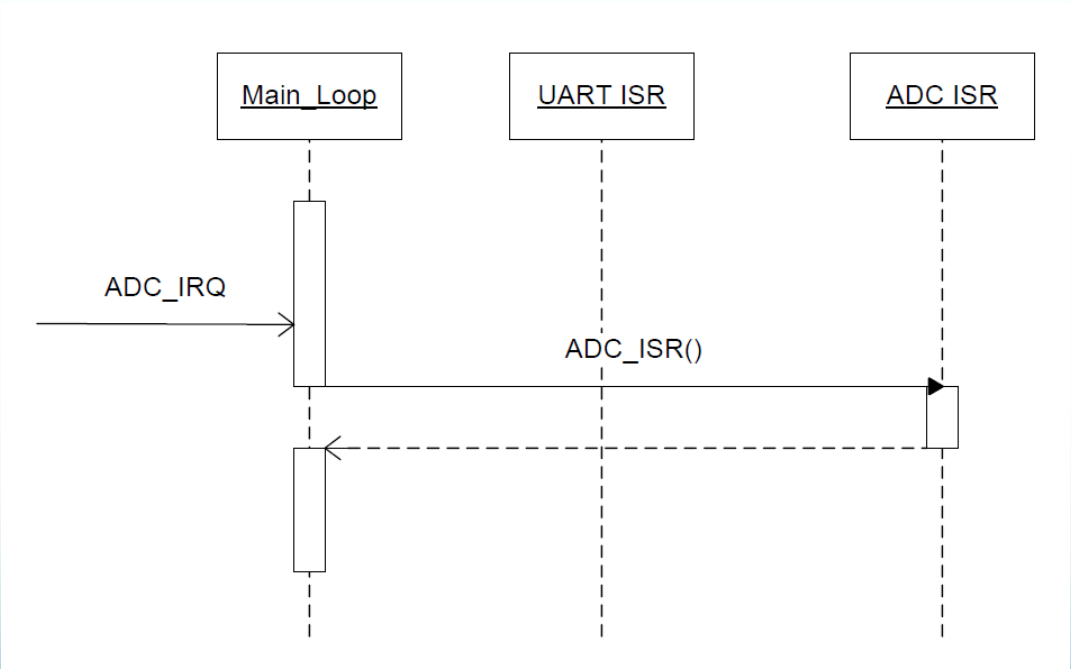
- Difference between the ISR and the standard function calls
  - The standard function calls are realized in a synchronous manner with branch instructions
  - Interrupt service routines are called when an exception signals occur
    - Vector table contains the addresses of the interrupt service routines

Programming without interrupts



- The main() function executes all peripheral calls in a fixed sequence

Programming with interrupts



Interrupt and exception vectors

- When an exception or an interrupt occurs
  - CPU interrupts the execution of the main program
  - CPU jumps to the vector address, which dependents on the exception type
- Base address of the vector table is usually 0
- Cortes M3 contains both an interrupt and an exception vector tables

Cortex-M3 Exception Types

No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
……	……..	……	settable	……..
256	Interrupt#240	247	settable	External Interrupt #240

Exception vector table of the Cortex M3 (Ref. RM0008 Reference manual)

Position	priority	Type of Priorty	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	UMI	Non maskable interrupt	0x0000_0008
	-1	settable	HardFault	All class of fault	0x0000_000C
	0	settable	MemManage	Memory management	0x0000_0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C 0x0000_002B
	3	settable	SCCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	Settable	SysTick	System tick timer	0x0000_003C

Interrupt vector table of the Cortex M3 (1)

Position	priority	Type of Priorty	Acronym	Description	Address
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080

Interrupt vector table of the Cortex M3 (2)

Position	priority	Type of Priorty	Acronym	Description	Address
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000_008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000_0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM	TIM1 Trigger & Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC TIM1	Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0

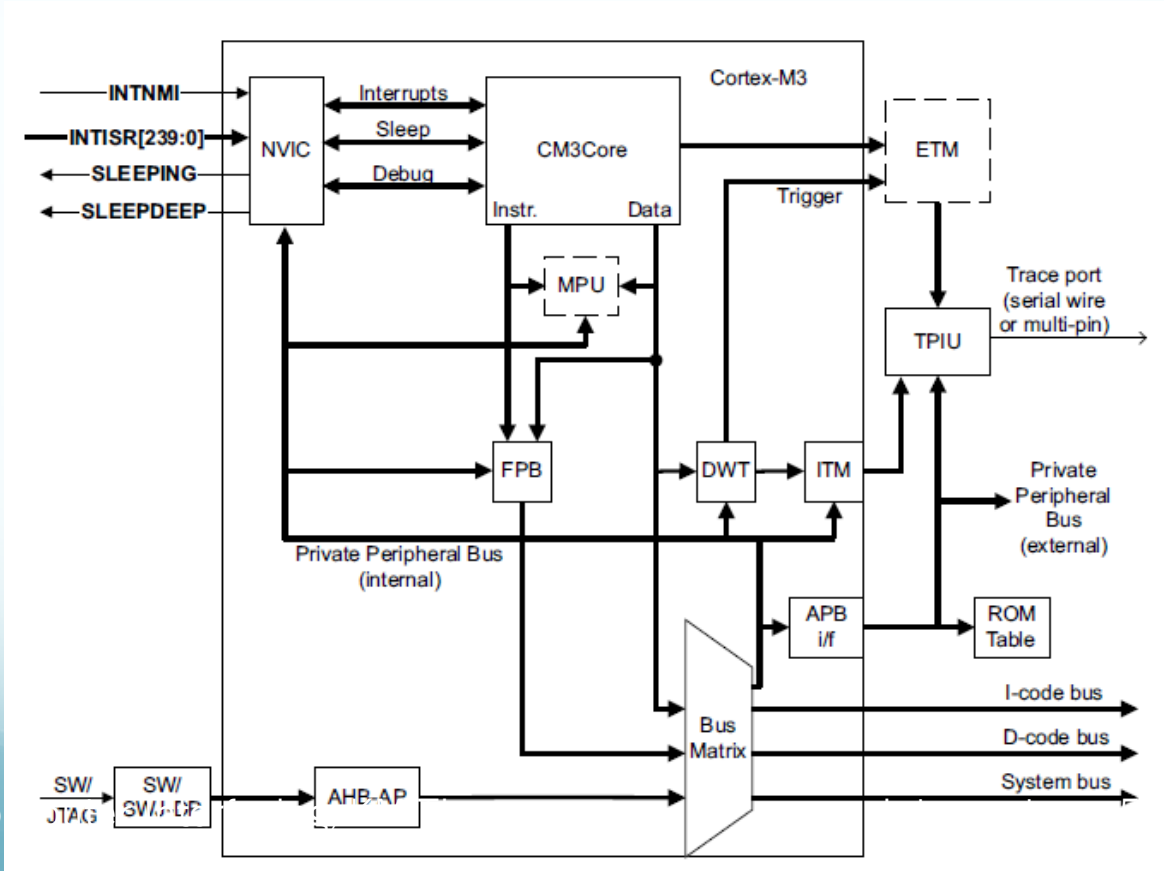
Interrupt vector table of the Cortex M3 (3)

Position	priority	Type of Priorty	Acronym	Description	Address
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8
-	-	-	-	Reserved	0x0000_00EC 0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114

Interrupt vector table of the Cortex M3 (4)

Position	priority	Type of Priorty	Acronym	Description	Address
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4	DMA2 Channel4 global interrupt	0x0000_012C
60	67	settable	DMA2_Channel5	DMA2 Channel5 global interrupt	0x0000_0130
61	68	settable	ETH	Ethernet global interrupt	0x0000_0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000_0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000_013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000_0140
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000_0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000_0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000_014C

Bloc Schematic of the NVIC (Ref. Cortex-M3 Technical Reference manual)

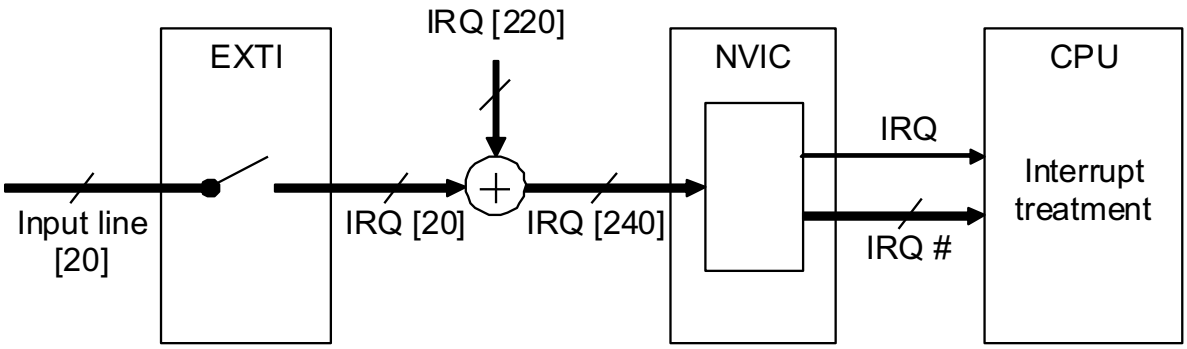


Nested Vectored Interrupt Controller (NVIC) (Ref. RM0008 Reference manual)

- Features
  - 68 (not including the sixteen Cortex™-M3 interrupt lines) exceptions
  - 16 programmable priority levels (4 bits of interrupt priority are used)
  - Low-latency exception and interrupt handling
  - Power management control
  - Implementation of System Control Registers
- The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts
- All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to *STM32F10xxx Cortex-M3 programming manual*

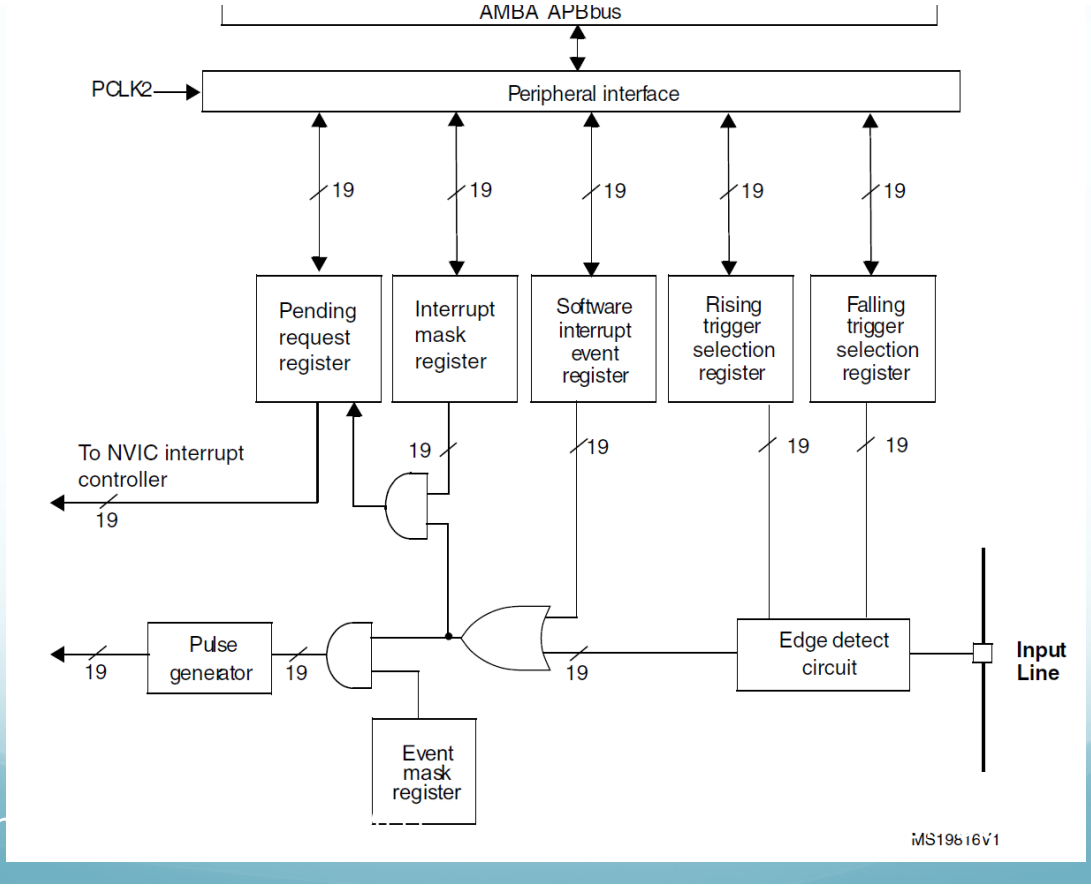
External interrupt/event controller (EXTI)

- EXTI consists of up to 20 edge detectors for generating event/interrupt requests

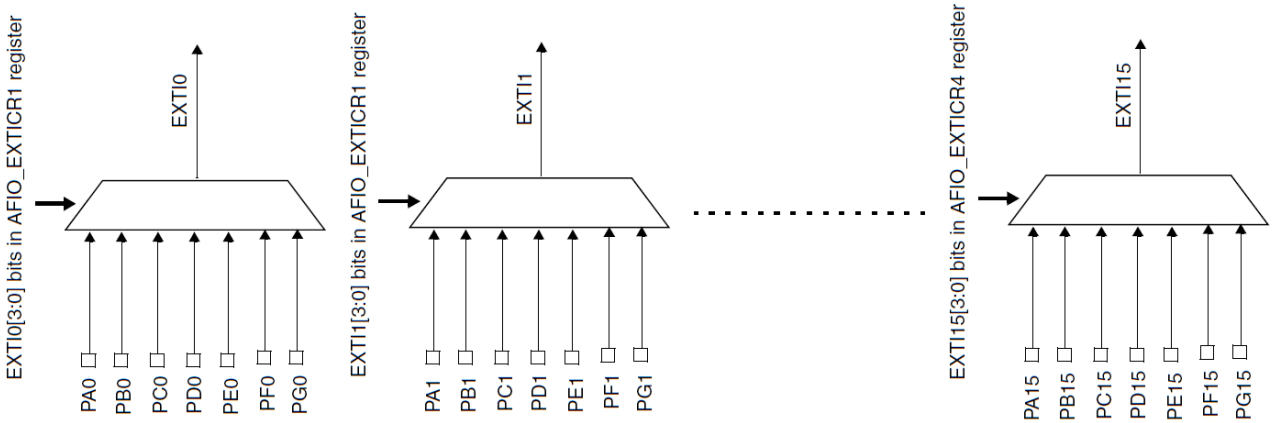


- Features
  - Independent trigger and mask on each interrupt/event line
  - Dedicated status bit for each interrupt line
  - Generation of up to 20 software event/interrupt request
  - Detection of external signal with pulse width lower than APB2 clock

EXTI Block diagram



External Interrupt/Event GPIO mapping



NVIC Registers

- Each interrupt input has several registers to control it
  - Enable/Disable Bit
    - Enable or disable the interrupt, Can be set, cleared or read
  - Pending Bit
    - If the pending bit is set, then the interrupt is pending
    - A pending interrupt can only be taken (become active) if it is enabled and it has sufficient priority to run
    - Pending bit can be set, cleared or read
  - Active Bit
    - A bit is set if the interrupt is executing or "active-stacked"
      - "Active-stacked" means the interrupt was executing, but was pre-empted by another higher-priority interrupt
    - Active register is normally read only
  - Priority field
    - priority management for each interrupt

Priorities of the exceptions

- The following questions must be answered for the case when several exceptions occur at the same time
  - Which exception should be treated at first?
  - Can an ISR be interrupted by an interruption signal?
- The exceptions have different priorities in most microprocessors
  - Their priorities can even be fixed specifically by software in some cases

NVIC Register Overview (Ref. Cortex-M3 Technical Reference manual)

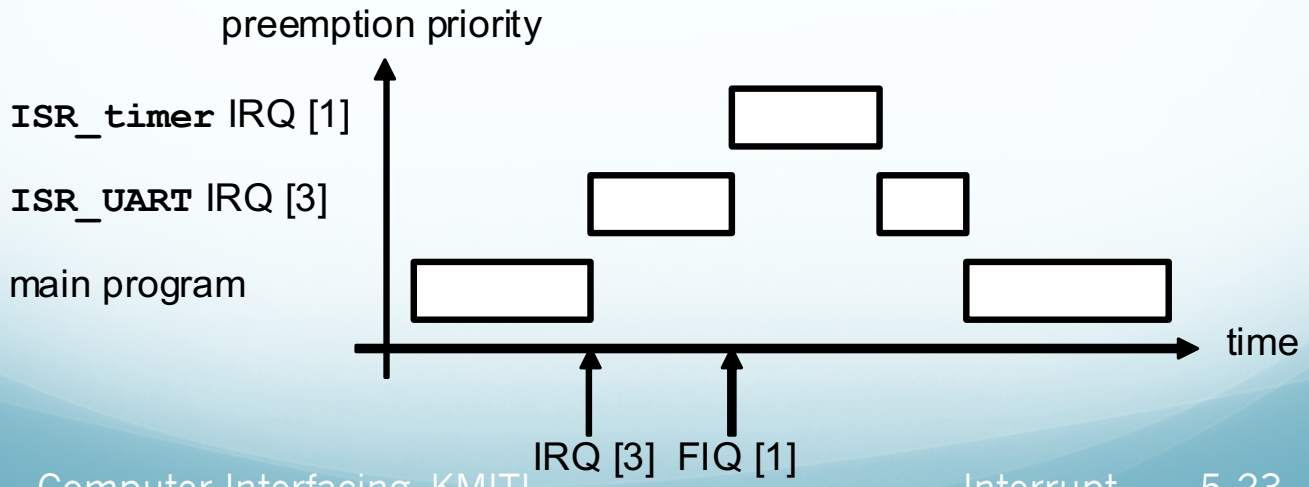
Name of the register	Type	Address
Interrupt Controller Type Register	Read-only	0xE000_E004
SysTick Control and Status Register	Read/write	0xE000_E010
SysTick Reload Value Register	Read/write	0xE000_E014
SysTick Current Value Register	Read/write clear	0xE000_E018
SysTick Calibration Value Register	Read-only	0xE000_E01C
Irq 0 to 239 Set Enable Register	Read/write	0xE000_E100 0xE000_E11C
Irq 0 to 239 Clear Enable Register	Read/write	0xE000_E100 0xE000_E11C
Irq 0 to 239 Set Pending Register	Read/write	0xE000_E200 0xE000_E21C
Irq 0 to 239 Active Bit Register	Read-only	0xE000_E300 0xE000_E31C
Irq 0 to 239 Priority Register	Read/write	0xE000_E400 0xE000_E4F0

NVIC Register Descriptions

- IRQ 0 to 239 Set-Enable Registers
  - Enable interrupts
  - Determine which interrupts are currently enabled
- IRQ 0 to 239 Clear-Enable Registers
  - Disable interrupts
  - Determine which interrupts are currently disabled
- IRQ 0 to 239 Set-Pending Register
  - Force interrupts into the pending state
  - Determine which interrupts are currently pending
- IRQ 0 to 239 Clear-Pending Register
  - Clear pending interrupts
  - Determine which interrupts are currently pending

Example

- The main program is interrupted by a hardware interrupt IRQ [3] (UART with preemption priority 3)
- During the treatment of the first hardware interrupt another hardware interrupt occurs (timer IRQ with preemption priority 1)
  - `ISR_UART` will be interrupted by the `ISR_Timer`



Priorities of the exceptions (Ref. Cortex-M3 Technical Reference Manual)

- NVIC supports software-assigned priority levels
  - Priority level from 0 to 255 can be assigned to each hardware Interrupt
    - `PRI_N` field of the *Interrupt Priority Register*
- All priority levels can be split into a preemption and a sub priorities
  - `PRIGROUP` field of the *Application Interrupt and Reset Control Register*

PRIGROUP[2:0]	Binary point position	Pre-emption field	Subpriority field	Number of pre-emption priorities	Number of subpriorities
b000	bxxxxxxx.y	[7:1]	[0]	128	2
b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
b011	bxxxx.yyyy	[7:4]	[3:0]	16	16
b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
b110	bx.yyyyyyy	[7]	[6:0]	2	128
b111	b.yyyyyyyy	None	[7:0]	0	256

NVIC Priority Register descriptions

- Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts

	31 – 24	23 – 16	15 – 8	7 – 0
E000E400	PRI_3	PRI_2	PRI_1	PRI_0
E000E404	PRI_7	PRI_6	PRI_5	PRI_4
E000E408	PRI_11	PRI_10	PRI_9	PRI_8
E000E40C	PRI_15	PRI_14	PRI_13	PRI_12
E000E410	PRI_19	PRI_18	PRI_17	PRI_16
E000E414	PRI_23	PRI_22	PRI_21	PRI_20
E000E418	PRI_27	PRI_26	PRI_25	PRI_24
E000E41C	PRI_31	PRI_30	PRI_29	PRI_28

NVIC Configuration

- Activation of an interrupt channel requires the following NVIC register configurations
  - Enable the interrupt channel by setting the its enable bit in the corresponding IRQ 0 to 239 Set-Enable Registers
  - Fix the priority of the interrupt channel in its Interrupt Priority Register

- Code example
  -
```
/* Enable the EXTI9_5 Interrupt */
NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

EXTI Configuration

- To configure the 20 lines as interrupt/event source
  - Configure the mask bits of the 20 Interrupt lines (EXTI_IMR ≡ `0x4001_04000`)
  - Configure the Trigger Selection bits of the Event lines (EXTI_RTSR ≡ `0x4001_0408` and EXTI_FTSR ≡ `0x4001_040C`)

- Code example

```
/* Configure EXTI interrupt on PIN PB7 (User button) */
EXTI_InitStructure.EXTI_Line = EXTI_Line7;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
```


GPIO Configuration

- Select the port pin in the corresponding `AFIO_EXTICRx` register

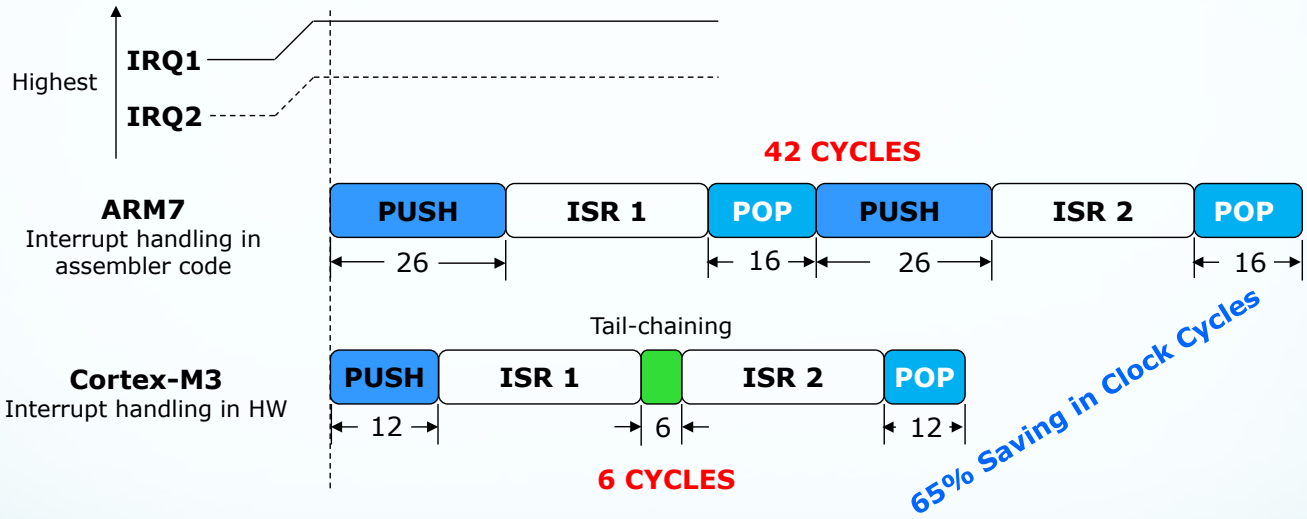
- Code example

```
/* Selects the pin PB7 as EXTI line */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOB,
                    GPIO_PinSource7);
```

Interrupt Handling

- **The Cortex-M3 processor integrates an advanced Nested Vectored Interrupt Controller (NVIC)**
  - The NVIC supports up to 240 dynamically reprioritizes interrupts each with up to 256 levels of priority
  - Supports advanced features for next generation real-time applications:
    - Tail-chaining of pending interrupts
    - Interrupt Pre-emption
    - Late Arrival

Interrupt Response- Tail Chaining(1/3)



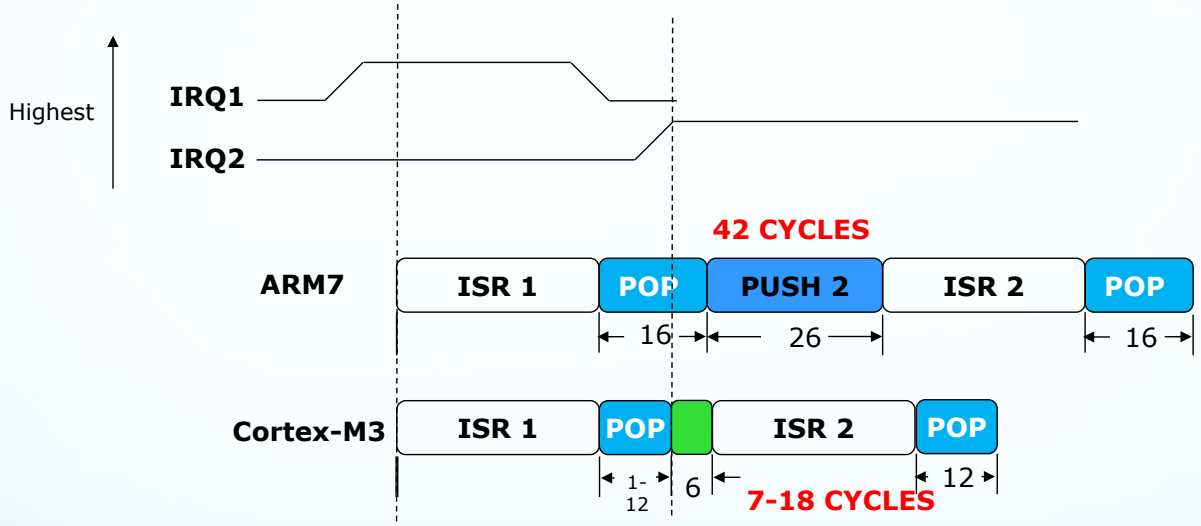
ARM7

- 26 cycles from IRQ1 to ISR1 entered
  - Up to 42 cycles
- 42 cycles from ISR1 exit to ISR2 entry
- 16 cycles to return from ISR2

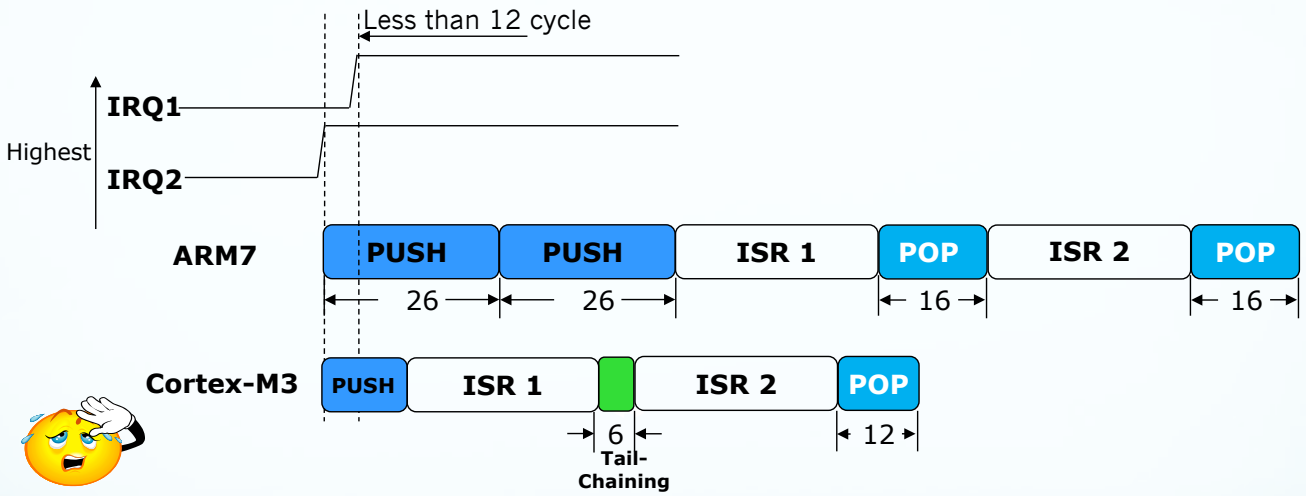
Cortex-M3

- 12 cycles from IRQ1 to ISR1 entered
  - 12 cycles
- 6 cycles from ISR1 exit to ISR2 entry
- 12 cycles to return from ISR2

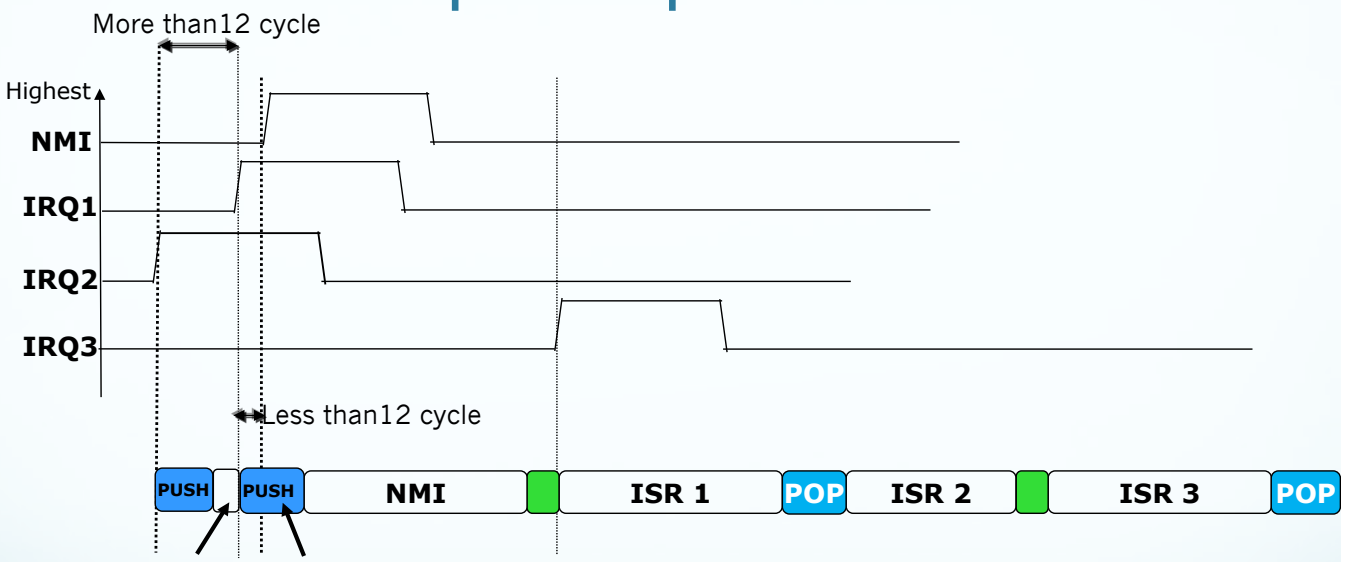
Interrupt Response – Preemption(2/3)



Interrupt Response – Late Arriving(3/3)



Interrupt Response – Lab



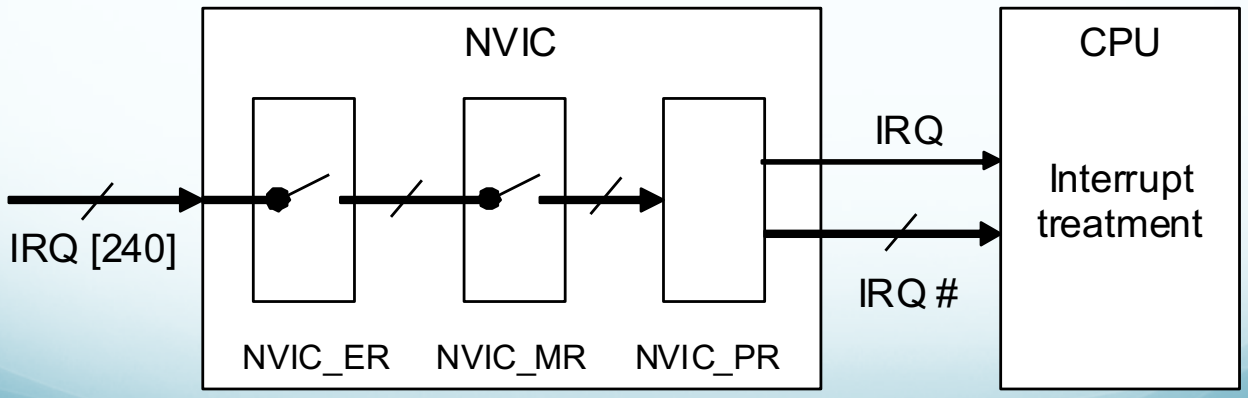
- Push for ISR1 begins
- Pre-empted by NMI
- New instruction fetch in parallel minimises time to NMI

Cortex-M3

- Following NMI processor tail-chains into ISR1
- ISR2 Completed
- Pop only occurs on return to "Main"

Masking of the Interrupts

- The interrupts can be activated or deactivated
- The Nested Vectored Interrupt Controller (NVIC) realize this operation within the Cortex M3 processors



ADC-DAC

ผศ.ดร. สุรินทร์ กิตติธรกุล และ
อ.สรยุทธ กลมกล่อม

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Analog-to-Digital Conversion

Terminology

analog: continuously valued signal, such as temperature or speed, with infinite possible values in between

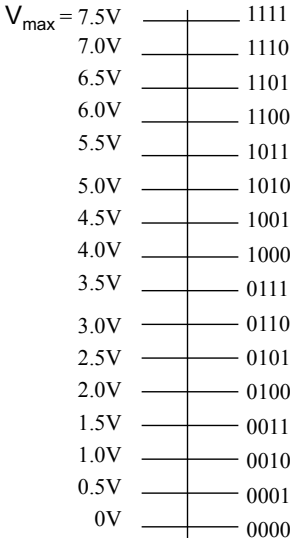
digital: discretely valued signal, such as integers, encoded in binary

analog-to-digital converter: ADC, A/D, A2D; converts an analog signal to a digital signal

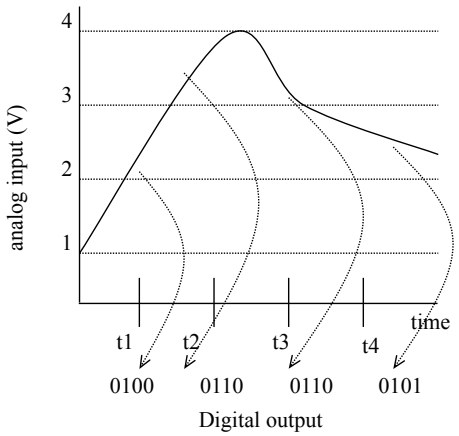
digital-to-analog converter: DAC, D/A, D2A

An embedded system's surroundings typically involve many analog signals.

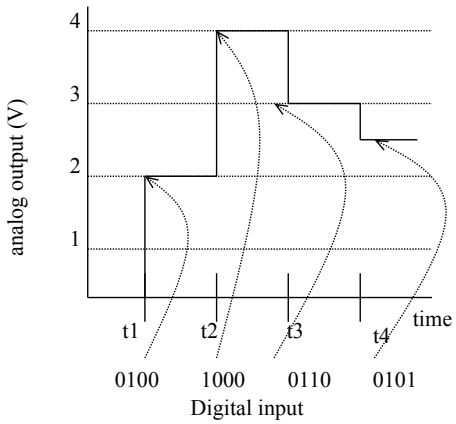
Analog-to-digital converters



proportionality



analog to digital



digital to analog

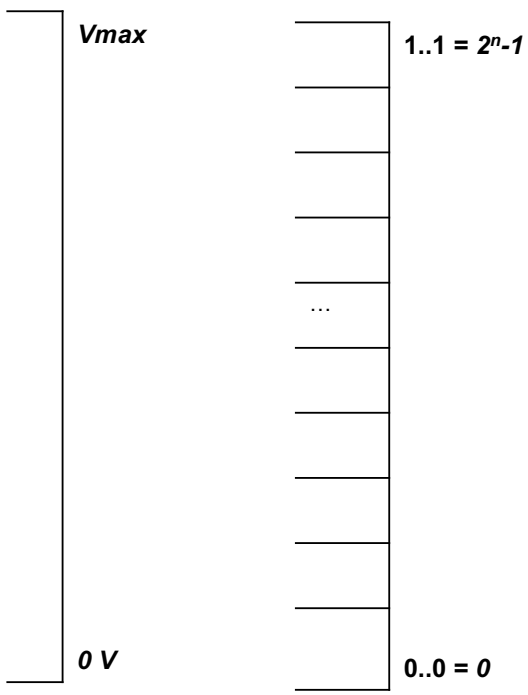
Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Proportional Signals

<u>Simple Equation</u>

Assume minimum voltage of 0 V.
Vmax = maximum voltage of the analog signal
a = analog value
n = number of bits for digital encoding
2^n = number of digital codes
M = number of steps, either 2^n or $2^n - 1$
d = digital encoding

$$a / Vmax = d / M$$



Resolution

Let $n = 2$

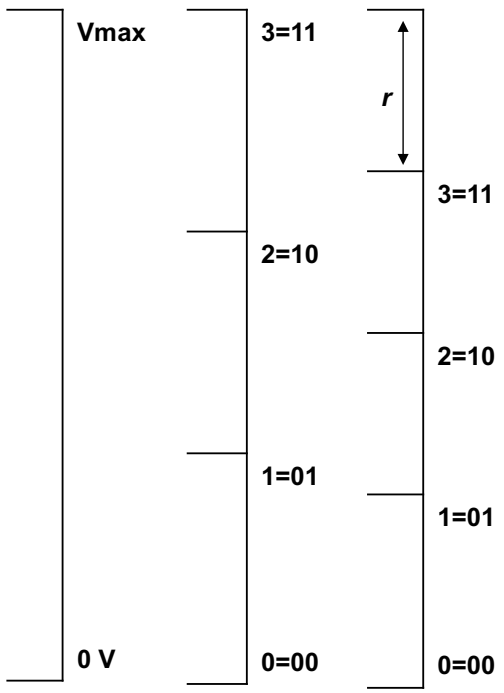
$\underline{M = 2^n - 1}$

3 steps on the digital scale
$d_0 = 0 = 0b00$
$d_{Vmax} = 3 = 0b11$

$\underline{M = 2^n}$

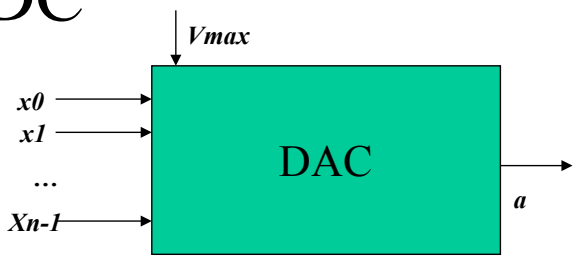
4 steps on the digital scale
$d_0 = 0 = 0b00$
$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax})

r, **resolution**: smallest analog change resulting from changing one bit



DAC vs. ADC

DAC:
n digital inputs for digital encoding *d*
analog input for *Vmax*
analog output *a*



ADC:

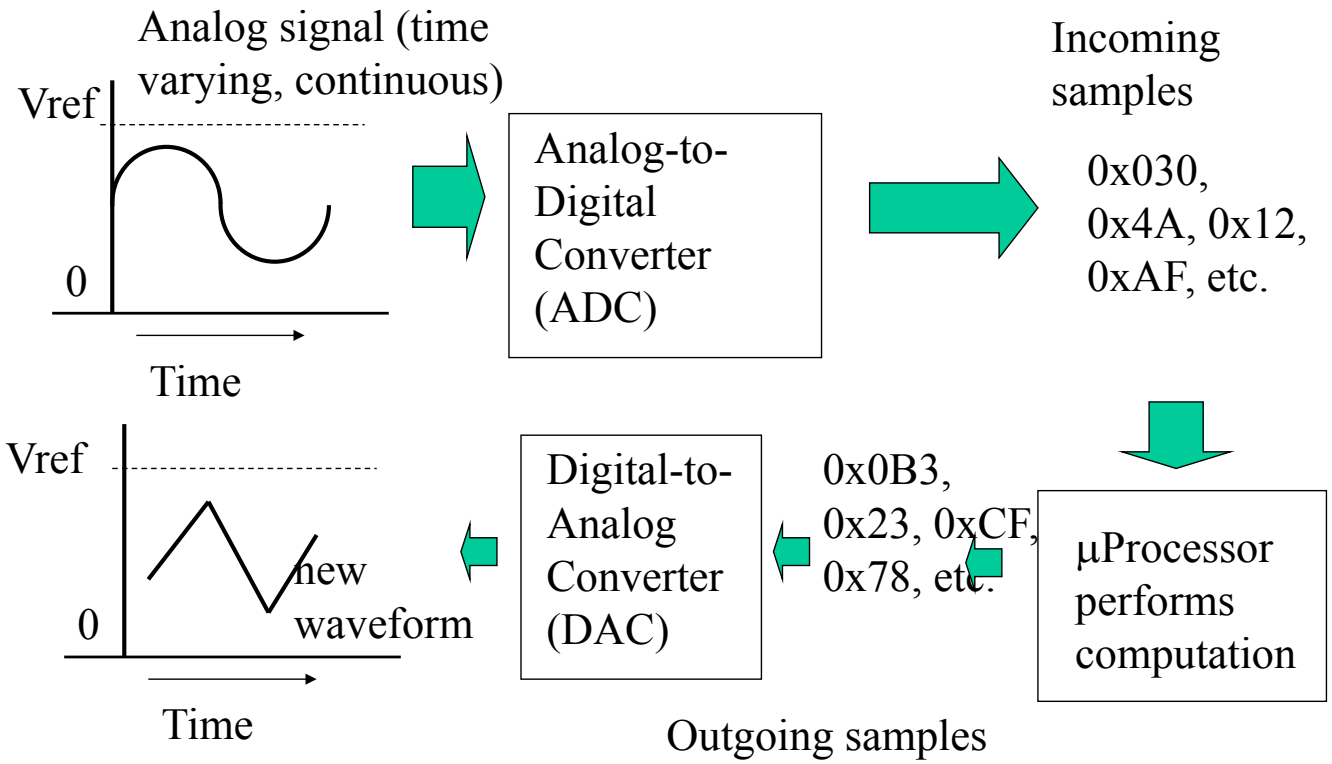
Given a *Vmax* analog input and an analog input *a*, how does the converter know what binary value to assign to *d* in order to satisfy the ratio?

- may use DAC to generate analog values for comparison with *a*
- ADC "guesses" an encoding *d*, then checks its guess by inputting *d* into the DAC and comparing the generated analog output *a'* with original analog input *a*
- How does the ADC guess the correct encoding?

Applications

- Audio
  - Speech recognition
  - special effects (reverb, noise cancellation, etc)
- Video
  - Filtering
  - Special effects
  - Compression
- Data logging

Digital Signal Processing



Vocabulary

- ADC (Analog-to-Digital Converter) – converts an analog signal (voltage/current) to a digital value
- DAC (Digital-to-Analog Converter) – converts a digital value to an analog value (voltage/current)
- Sample period – for ADC, time between each conversion
  - Typically, samples are taken at a fixed rate
- Vref (Reference Voltage) – analog signal varies between 0 and Vref, or between +/- Vref
- Resolution – number of bits used for conversion (8 bits, 10 bits, 12 bits, 16 bits, etc).
- Conversion Time – the time it takes for a analog-to-digital conversion

Bit Weight

Notice the concept of bit weight in the last example:
bit 7 = 7.5 V = 15/2
bit 6 = 3.75 V = 15/4

Each bit is weighted with an analog value, such that a 1 in that bit position adds its analog value to the total analog value represented by the digital encoding.

Example: -5 V to +5 V analog range, *n=8*

Digital Bit	Bit Weight (V)
7	10/2 = 5
6	10/4 = 2.5
5	10/8 = 1.25
4	10/16 = 0.625
3	10/32 = 0.313
2	10/64 = 0.157
1	10/128 = 0.078
0	10/256 = 0.039

Bit Weight

Example (continued): -5 V to +5 V analog range, *n=8*

Digital numbers for a few analog values

- Values shown increment by 6 bits (weight for bit position 5 is 1.25 V)
- Maximum digital number only approximates the maximum analog value in the range
  - Try (-5) + sum of all bit weights

Analog (V)	Digital (hex)
-5	00
-3.75	20
-2.5	40
-1.25	60
0	80
1.25	A0
2.5	C0
3.75	E0
5-0.039 = 4.961	FF

Sample ADC Computations

If Vref = 5V, and the 10-bit A/D output code is 0x12A, what is the ADC input voltage?

$$\text{output_code}/2^N * \text{Vref} = (0x12A)/2^{10} * 5\text{ V}$$
$$= 298/1024 * 5\text{ V} = 1.46\text{ V (Vin)}$$

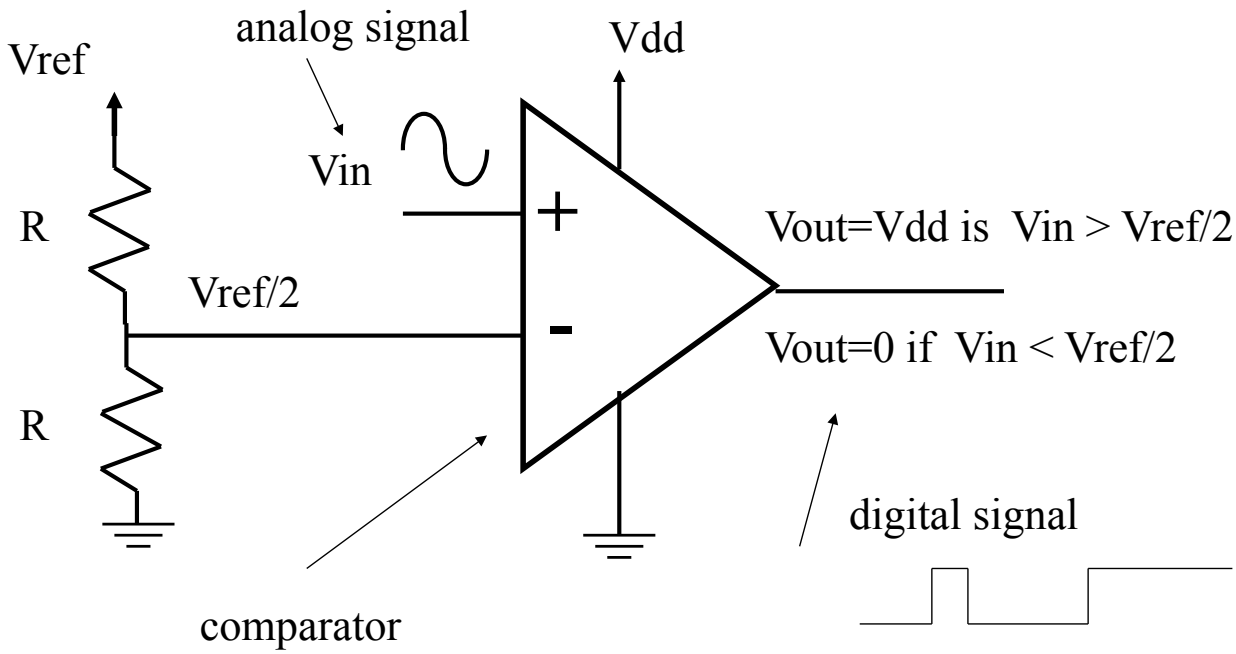
If Vref = 5V, and the upper 8 bits of the A/D output code is 0xA9, what is the ADC input voltage?

$$\text{output_code}/2^N * \text{Vref} = (0xA9)/2^8 * 5\text{ V}$$
$$= 169/256 * 5\text{ V} = 3.3\text{ V (Vin)}$$

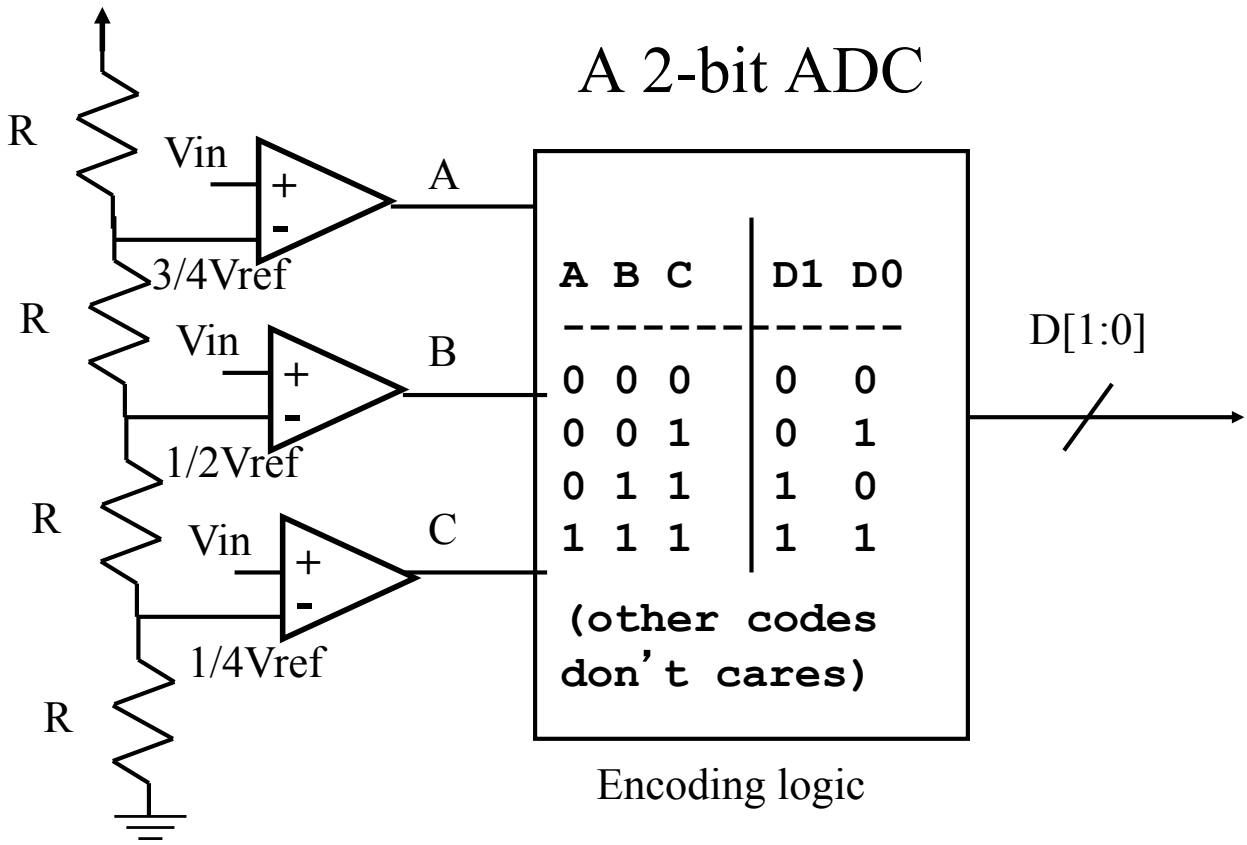
If Vref = 4V, and the A/D input voltage is 2.35 V, what is the ADC output code, upper 8-bits?

$$\text{Vin}/\text{Vref} * 2^N = 2.35\text{ V}/4\text{ V} * 2^8$$
$$= .5875 * 256 = 150.4 = 150 = 0x96$$

A 1-bit ADC



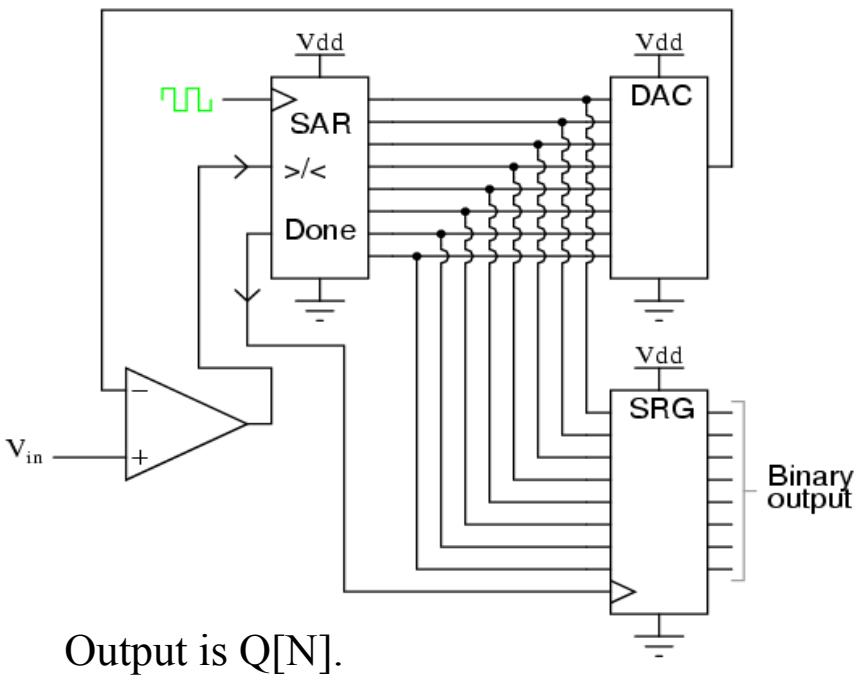
A 2-bit ADC



ADC Architectures

- The previous architectures are called ***Flash*** ADCs
  - Fastest possible conversion time
  - Requires the most transistors of any architecture
  - N-bit converter requires 2^N-1 comparators.
  - Commercially available flash converters up to 12 bits.
  - Conversion done in one clock cycle
- ***Successive approximation*** ADCs
  - Use only one comparator
  - Take one clock cycle per bit
  - High precision (16-bit converters are available)

Successive Approximation ADC



First, set DAC to produce Vref/2.

Output of Comparator is Q[N-1] (MSB)

If MSB =1 , then Vin between Vref and Vref/2, so set DAC to produce ¾ Vref.

If MSB=0, then Vin between Vref/2 and 0, so set DAC to ½ Vref.

Output of comparator is now Q[N-2].

Do this for each bit.

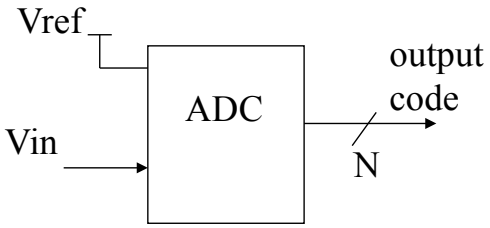
Takes N cycles.

Output is Q[N].

From http://www.analogcircuits.com

ADC, DAC Equations

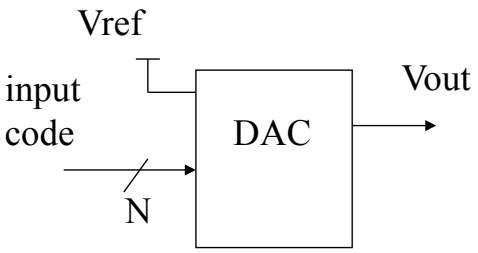
ADC: Vin = input voltage, Vref = reference voltage
N = number of bits of precision



$Vin/ Vref * 2^N = output_code$
$output_code/ 2^N * Vref = Vin$

1 LSB = $Vref/2^N$

DAC: Vout = output voltage, Vref = reference voltage,
N = number of bits of precision



$Vout/ Vref * 2^N = input_code$
$input_code/ 2^N * Vref = Vout$

1 LSB = $Vref/2^N$

ADC: Digital Encoding

Guessing the encoding is similar to finding an item in a list.

1. **Sequential search** – counting up: start with an encoding of 0, then 1, then 2, etc. until find a match.
   - 2^n comparisons: Slow!
2. **Binary search** – successive approximation: start with an encoding for half of maximum; then compare analog result with original analog input; if result is greater (less) than the original, set the new encoding to halfway between this one and the minimum (maximum); continue dividing encoding range in half until the compared voltages are equal
   - ***n*** comparisons: Faster, but more complex converter

→ Takes time to guess the encoding: start conversion input, conversion complete output

ADC using successive approximation

- Given an analog input signal whose voltage should range from 0 to 15 volts, and an 8-bit digital encoding, calculate the correct encoding for 5 volts. Then trace the successive-approximation approach to find the correct encoding.
- Assume $M = 2^n - 1$

$$a / Vmax = d / M$$

$$5 / 15 = d / (256 - 1)$$

$$d = 85 \text{ or binary } 01010101$$

ADC using successive approximation

Step 1-4: determine bits 0-3

$\frac{1}{2}(V_{max} - V_{min}) = 7.5$ volts
$V_{max} = 7.5$ volts.

0	0	0	0	0	0	0	0

$\frac{1}{2}(7.5 + 0) = 3.75$ volts
$V_{min} = 3.75$ volts.

0	1	0	0	0	0	0	0

$\frac{1}{2}(7.5 + 3.75) = 5.63$ volts
$V_{max} = 5.63$ volts

0	1	0	0	0	0	0	0

$\frac{1}{2}(5.63 + 3.75) = 4.69$ volts
$V_{min} = 4.69$ volts.

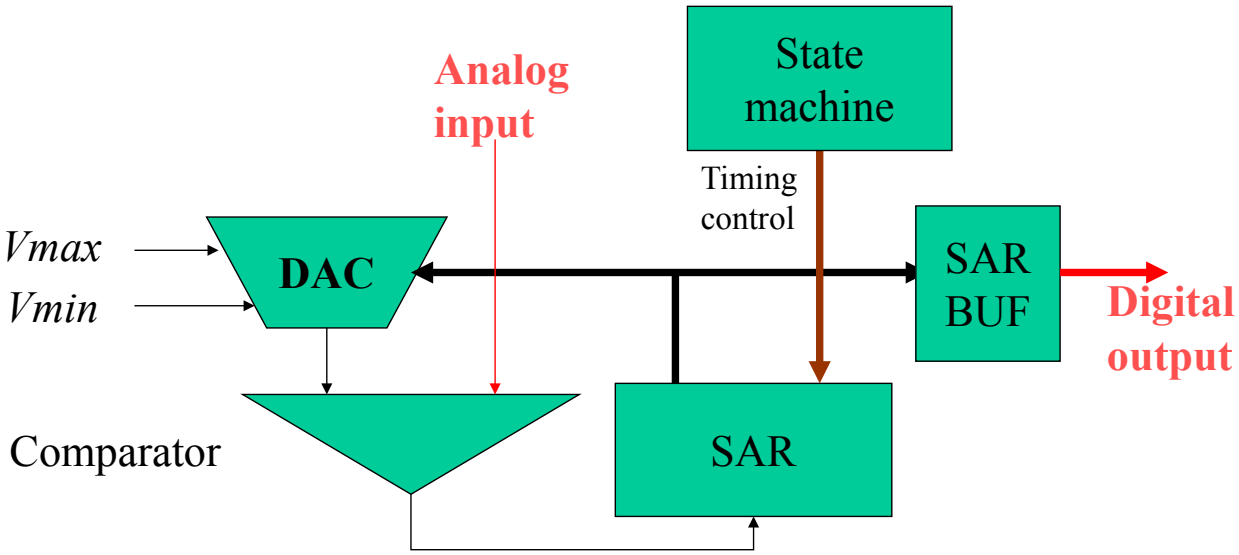
0	1	0	1	0	0	0	0

ADC using successive approximation

Step 5-8: Determine bits 4-7

½(5.63 + 4.69) = 5.16 volts V_{max} = 5.16 volts.	0	1	0	1	0	0	0	0
½(5.16 + 4.69) = 4.93 volts V_{min} = 4.93 volts.	0	1	0	1	0	1	0	0
½(5.16 + 4.93) = 5.05 volts V_{max} = 5.05 volts.	0	1	0	1	0	1	0	0
½(5.05 + 4.93) = 4.99 volts	0	1	0	1	0	1	0	1

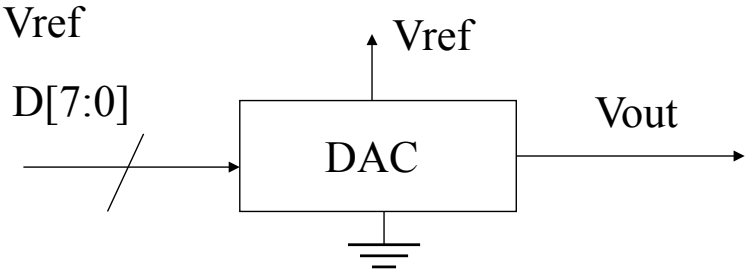
Constructing ADC



SAR: Successive approximation register

Digital-to-Analog Conversion

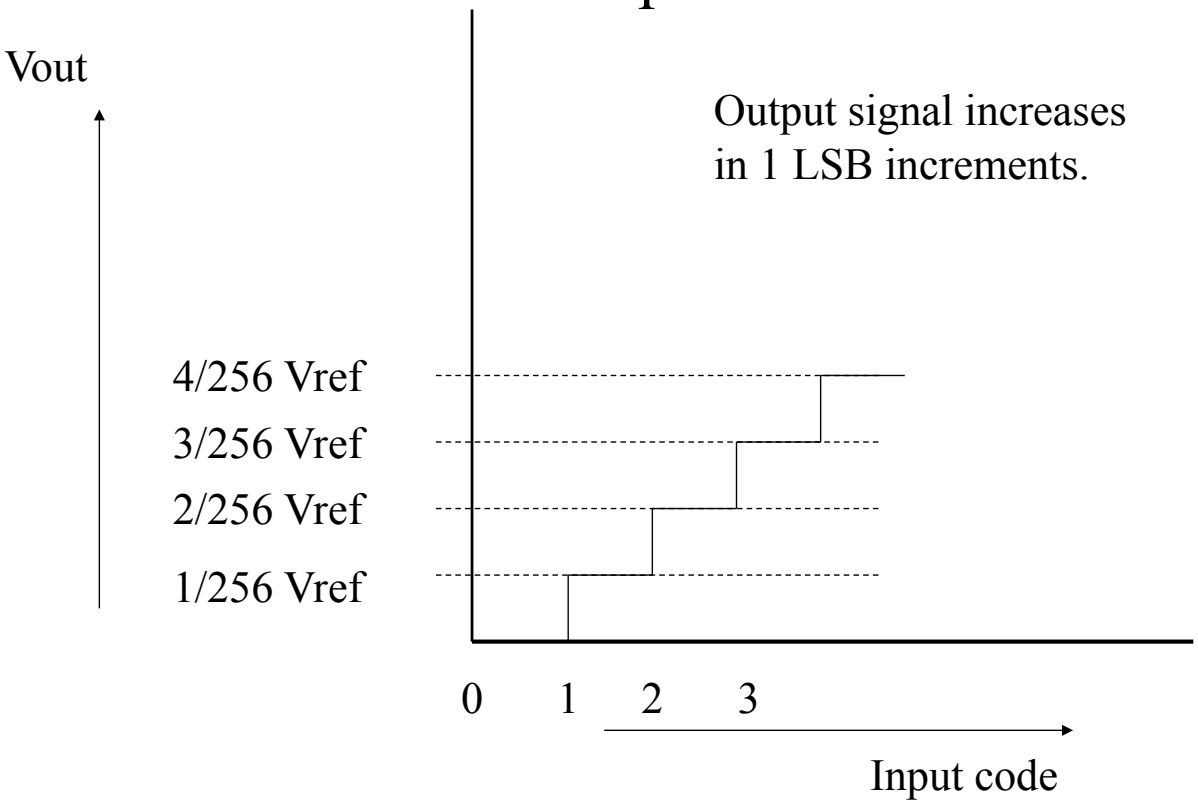
For a particular binary code, output a voltage between 0 and Vref



Assume a DAC that uses an unsigned binary input code, with 0 < Vout < Vref. Then

D= 0000 0000 Vout = 0V
D= 0000 0001 Vout = Vref(1/256) (one LSB)
D = 0000 0010 Vout = Vref(2/256)
...
D = 1111 1111 Vout = Vref(255/256) (full scale)

DAC Output Plot



Sample DAC Computations

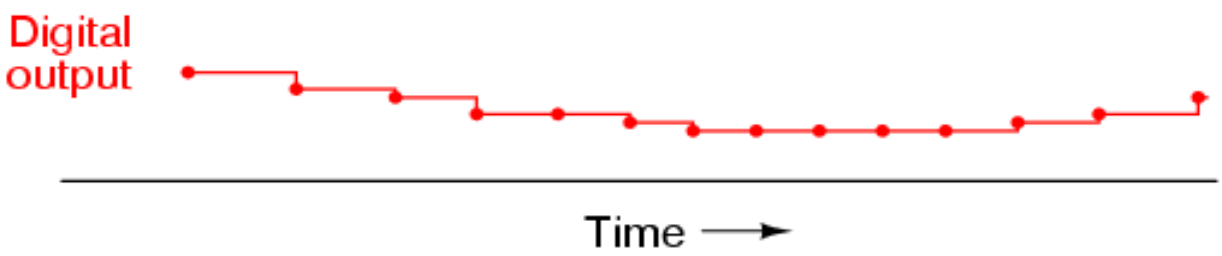
If Vref = 5V, and the 8-bit input code is is 0x8A, what is the DAC output voltage?

$$input_code/2^N * Vref = (0x8A)/2^8 * 5\ V$$
$$= 138/256 * 5\ V = 2.70\ V \quad (Vout)$$

If Vref = 4V, and the DAC output voltage is 1.25 V, what is the 8-bit input code?

$$Vout/\ Vref * 2^N = 1.25\ V/4\ V * 2^8$$
$$= 0.3125 * 256 = 80 = 0x50 \quad (input_code)$$

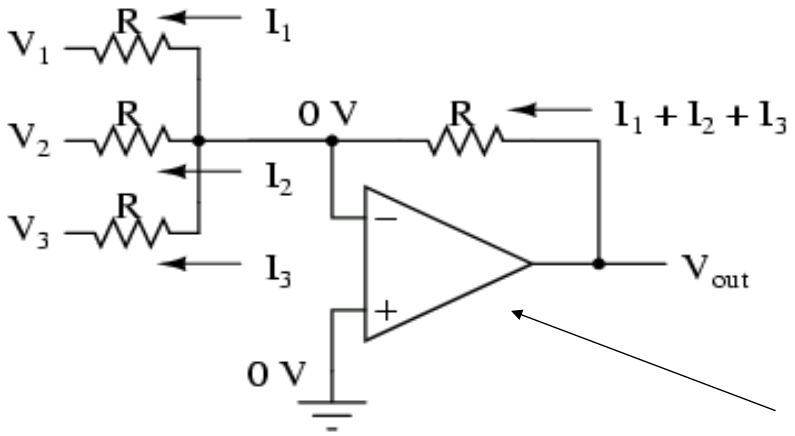
Typical DAC Output



From http://www.allaboutcircuits.com

DAC Architecture

Inverting summer circuit



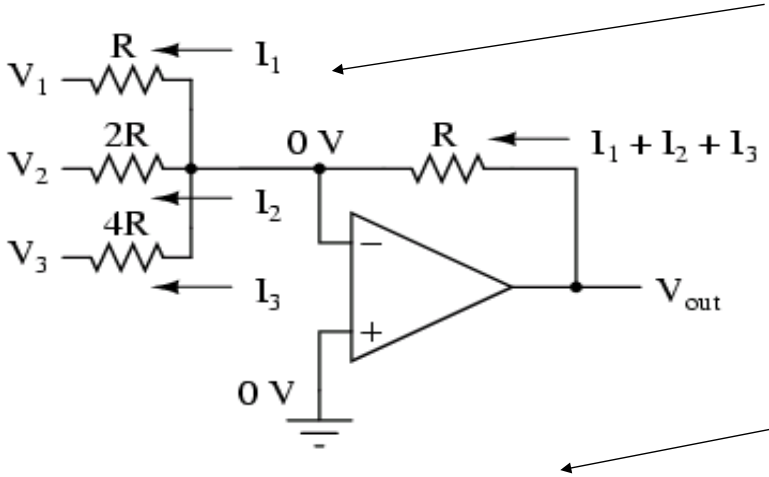
Operational Amplifier can be used to sum voltages.

$$V_{out} = -(V_1 + V_2 + V_3)$$

From http://www.allaboutcircuits.com

DAC Architecture (cont)

Note ratios of resistors

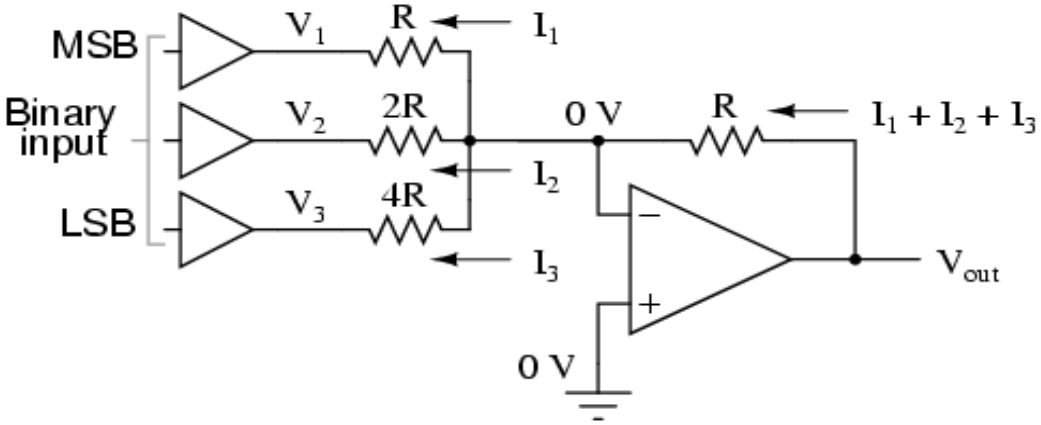


This is a binary code

$$V_{out} = -\left(V_1 + \frac{V_2}{2} + \frac{V_3}{4}\right)$$

From http://www.allaboutcircuits.com

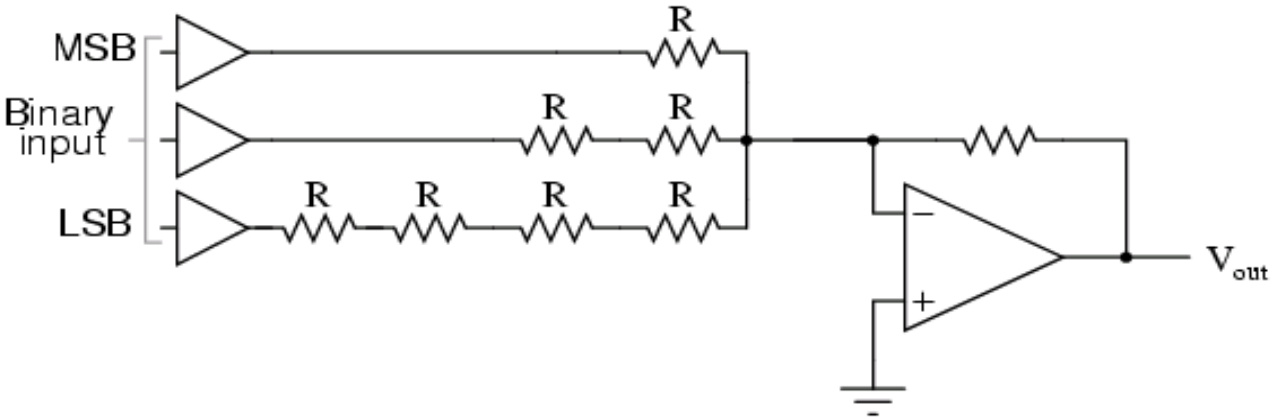
DAC Architecture (cont)



A 3-bit DAC, called an $R/2^N R$ DAC. Resistors are scaled by powers of 2 (this is hard to do in practice).

From http://www.allaboutcircuits.com

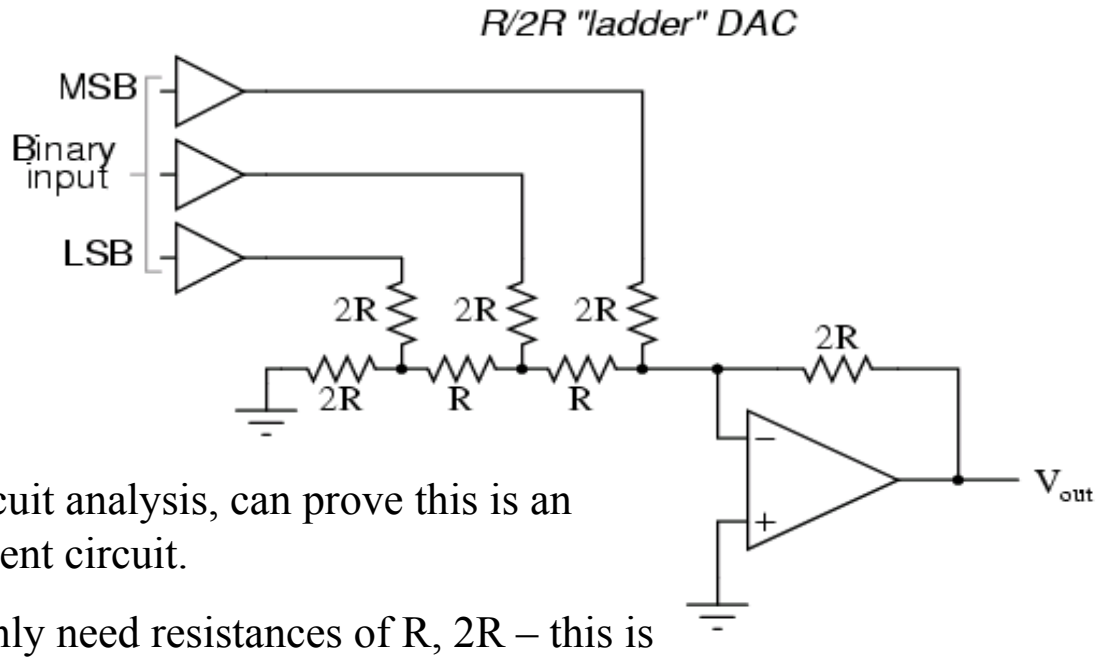
Another View



Resistance values are still R, 2R, 4R

From http://www.allaboutcircuits.com

R/2R DAC

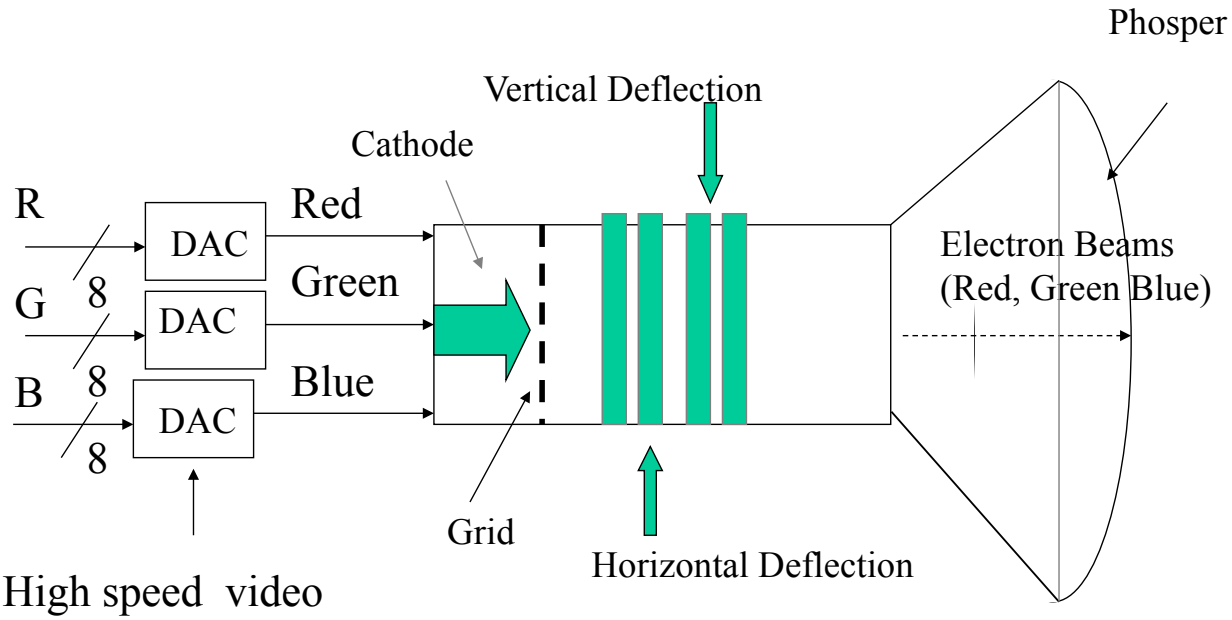


Via circuit analysis, can prove this is an equivalent circuit.

Now only need resistances of R, 2R – this is easy to do. This is the most common DAC architecture.

From http://www.allaboutcircuits.com

DAC Application

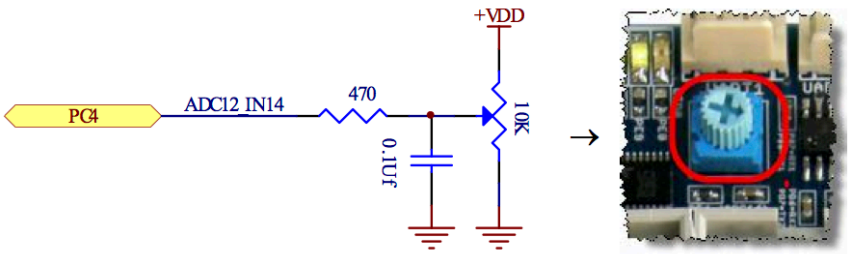


High speed video DACs produce RGB signals for color CRT

What do you have to know?

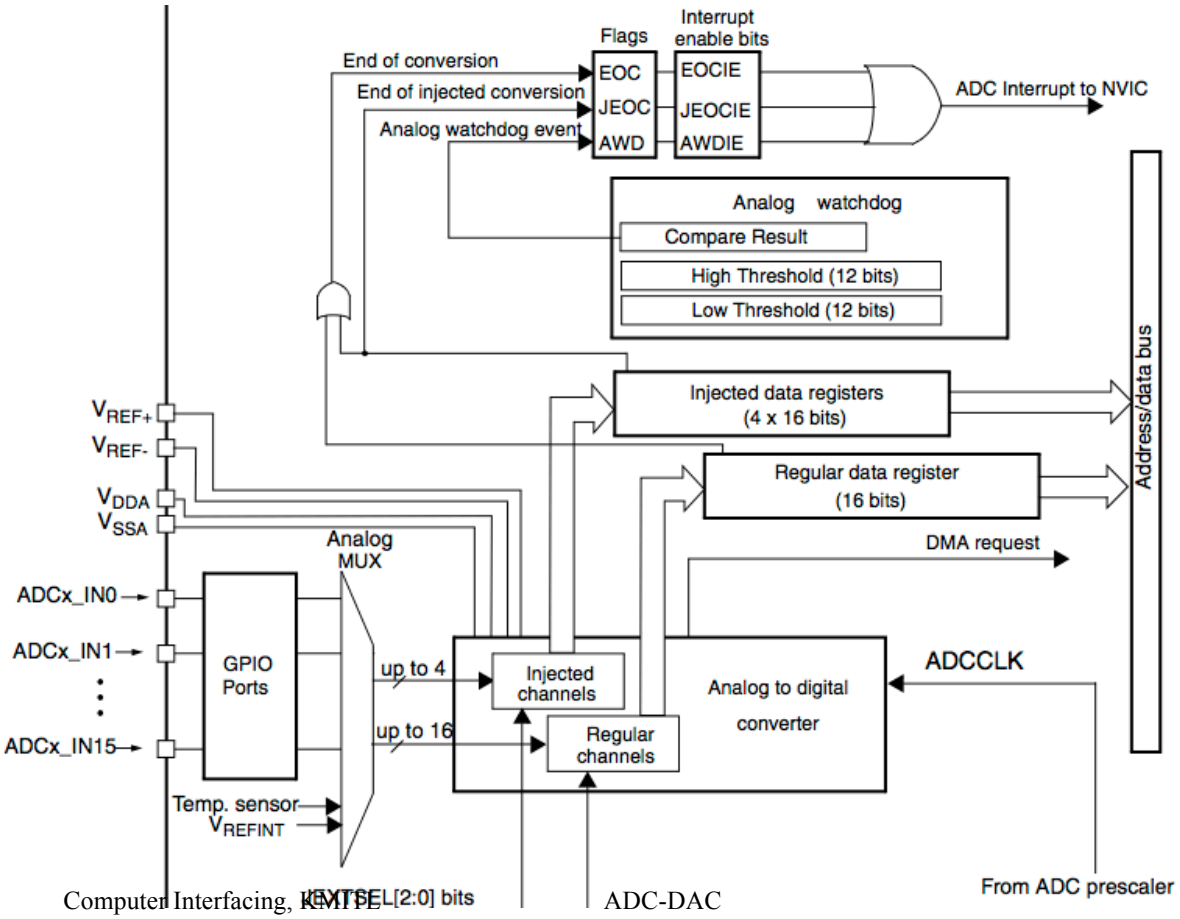
- Vocabulary
- DAC R/2N architecture
- ADC Flash, Successive approximation architectures
- STM32 A/D
  - How to configure
  - Acquisition, Conversion time
  - How to start do conversion, read result

ADC (Analog to Digital Converter) Input




```
/* Configure PC.04 (ADC Channel14) as analog input */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* ADC1 Configuration */
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_InitStructure.ADC_ScanConvMode = DISABLE;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_NbrOfChannel = 1;
ADC_Init(ADC1, &ADC_InitStructure);
```


Flags
Interrupt enable bits
End of conversion
End of injected conversion
Analog watchdog event
EOC
JEOC
AWD
EOCIE
JEOCIE
AWDIE
ADC Interrupt to NVIC
Analog watchdog
Compare Result
High Threshold (12 bits)
Low Threshold (12 bits)
Injected data registers
(4 x 16 bits)
Regular data register
(16 bits)
Address/data bus
VREF+
VREF-
VDDA
VSSA
Analog MUX
DMA request
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO Ports
up to 4
up to 16
Injected channels
Regular channels
Analog to digital converter
ADCCLK
Temp. sensor
VREFINT
EXTSEL[2:0] bits
From ADC prescaler

Timer, Watch Dog, PWM

ปรับแก้โดย

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

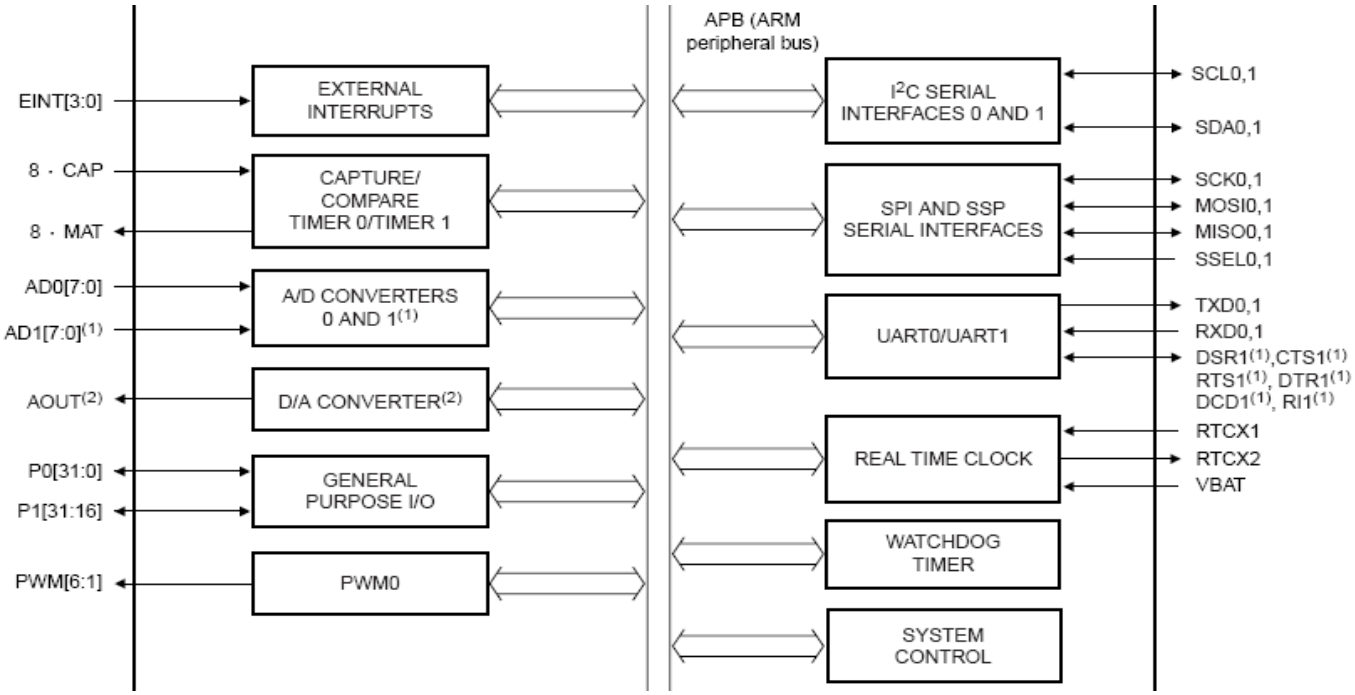
References

http://www.nxp.com/acrobat_download/usermanuals/UM10120_1.pdf
Trevor Martins , The insider's guide to the Philips ARM7 based microcontrollers, www.hitex.co.uk

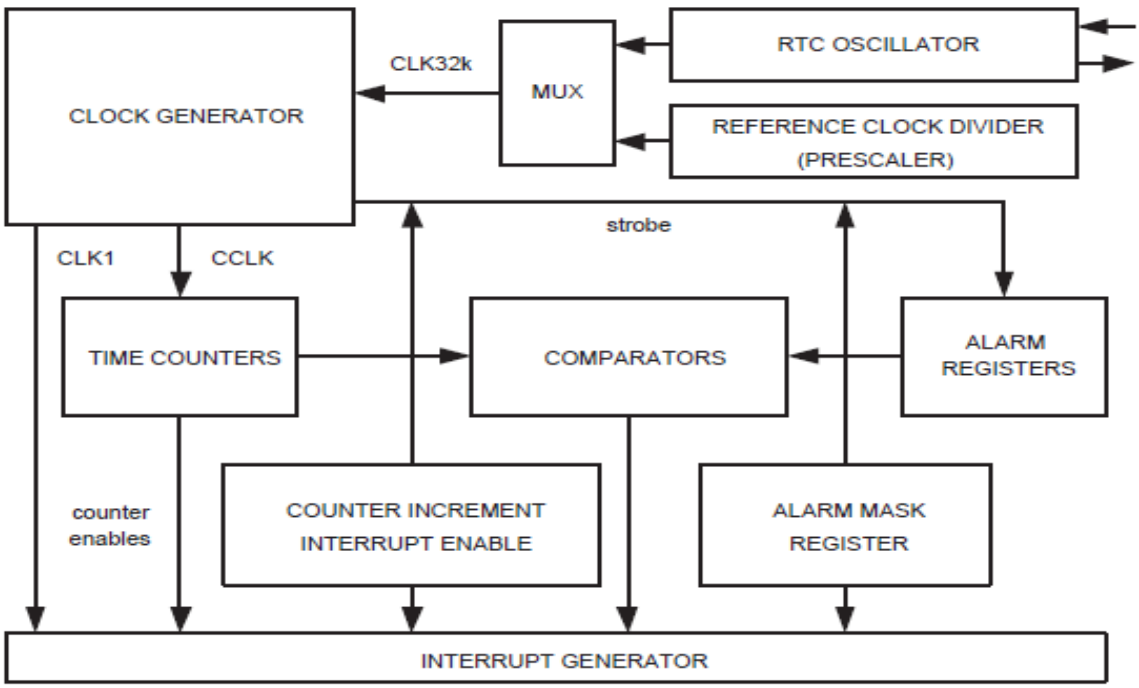
Introduction

1. Timer
2. Watchdog
3. Pulse Width Modulation (PWM)

Microcontroller peripherals



Timer/Counter Block Diagram

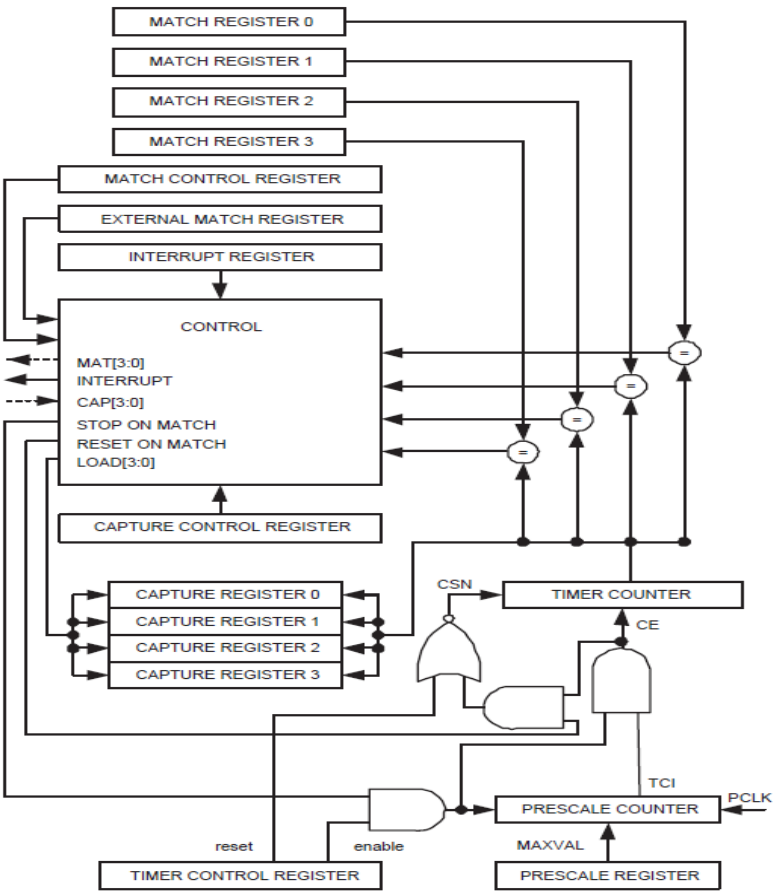


1) Timer

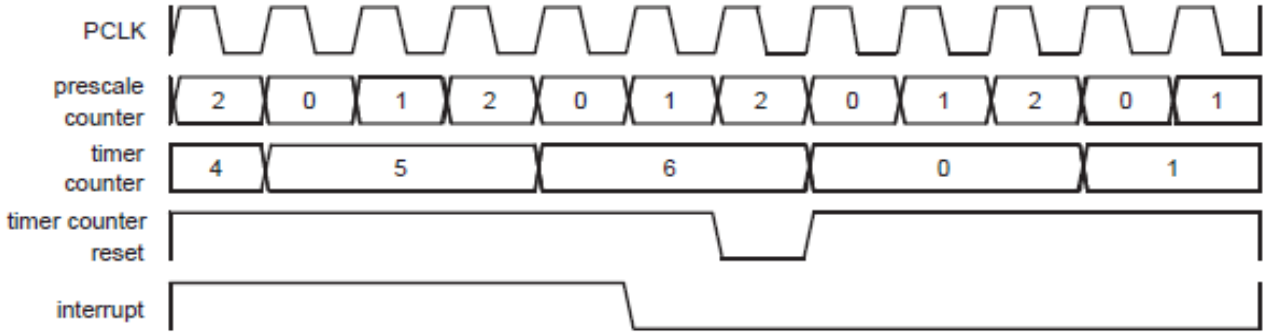
http://www.keilsoftware.com/dd/vtr/3735/8064.htm

- **Including these Features**
  - A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
  - Counter or Timer operation
  - Four 32-bit match registers that allow:
    - Set low on match, Set high on match, Toggle on match, Do nothing on match.
- **Applications**
  - Interval Timer for counting internal events.
  - Pulse Width Demodulator via Capture inputs.
  - Free running timer.

Match Logic

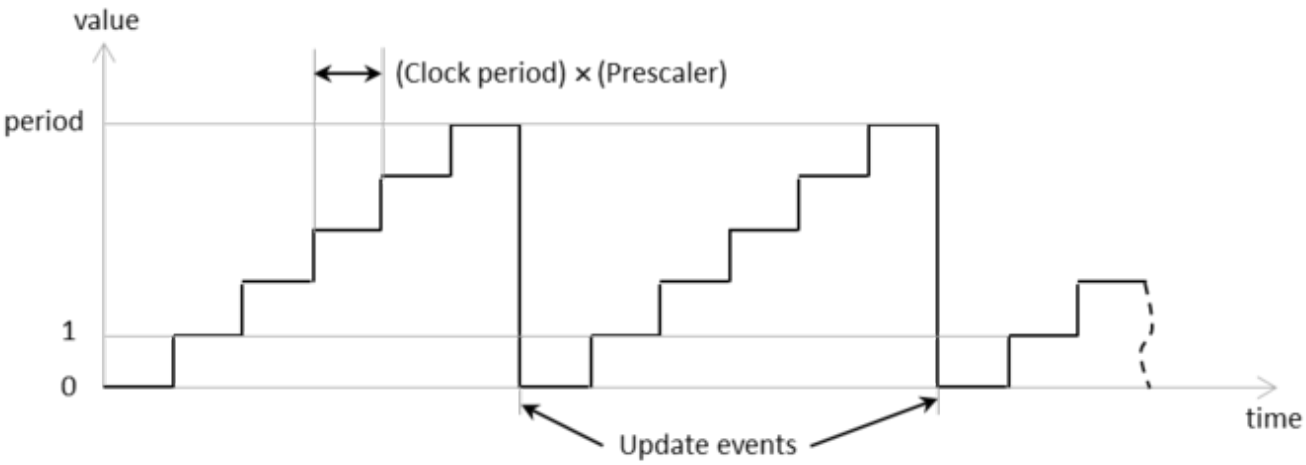


Timer & Interrupt

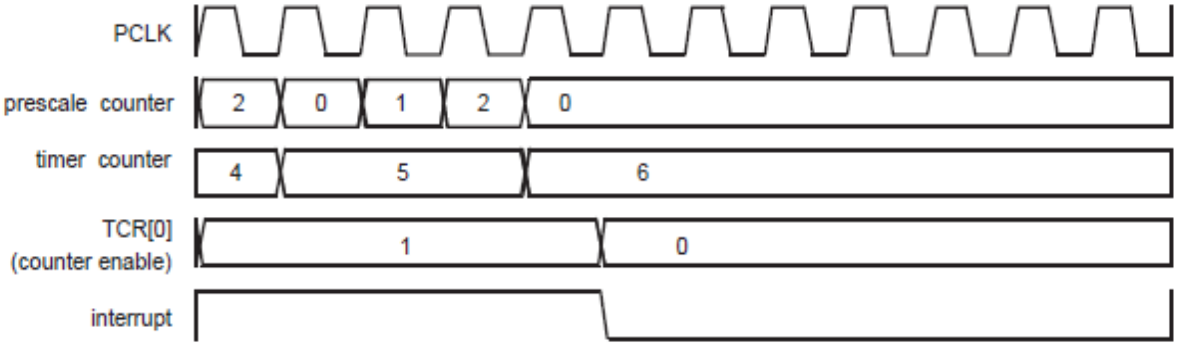


- a timer configured to reset the count and generate an interrupt on match.The prescaler is set to 2 and the match register set to 6. At the end of the timer cyclewhere the match occurs, the timer count is reset.

Timer & Interrupt



Timer & Interrupt then Stop



- a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

----------RECALL----------

Part 1 of void init_timer_Eint() of EINT.c (interrupt rate =1KHz)

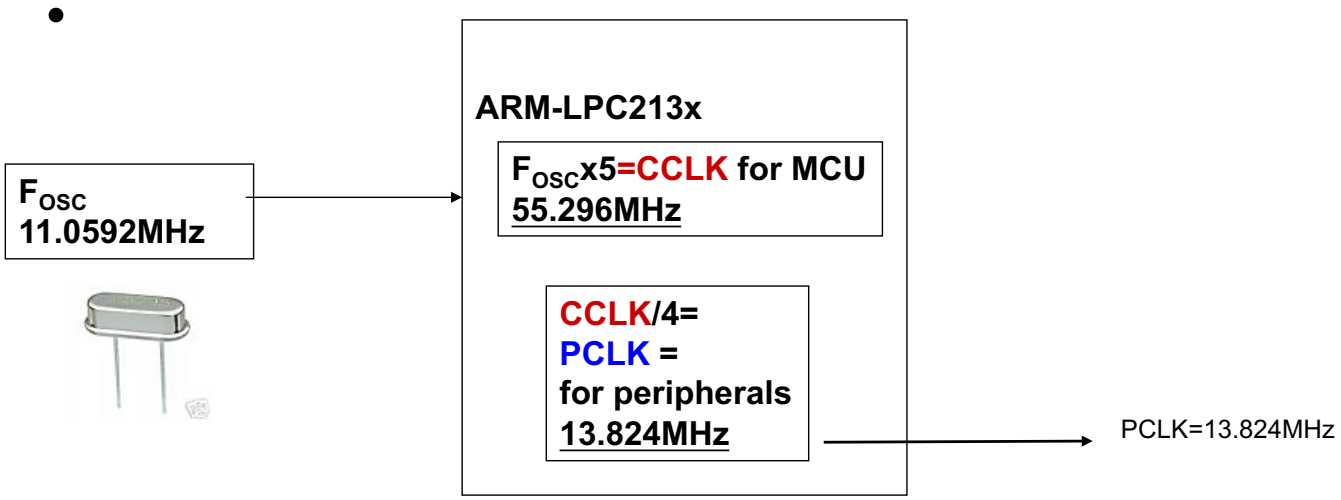
(for init timer , use VICVectAddr0

- /* Setup the Timer Counter 0 Interrupt */
- void init_timer_Eint (void) {
- T0PR = 0;
- // set prescaler
- T0MR0 =13824; // set interrupt interval to 1mS
- // since pclk/1KHz = (11059200 x 5)/(4 x 1000)=13824

cclk=M*Fosc, M=5
pclk=cclk/4
Pclk=11059200*5/4

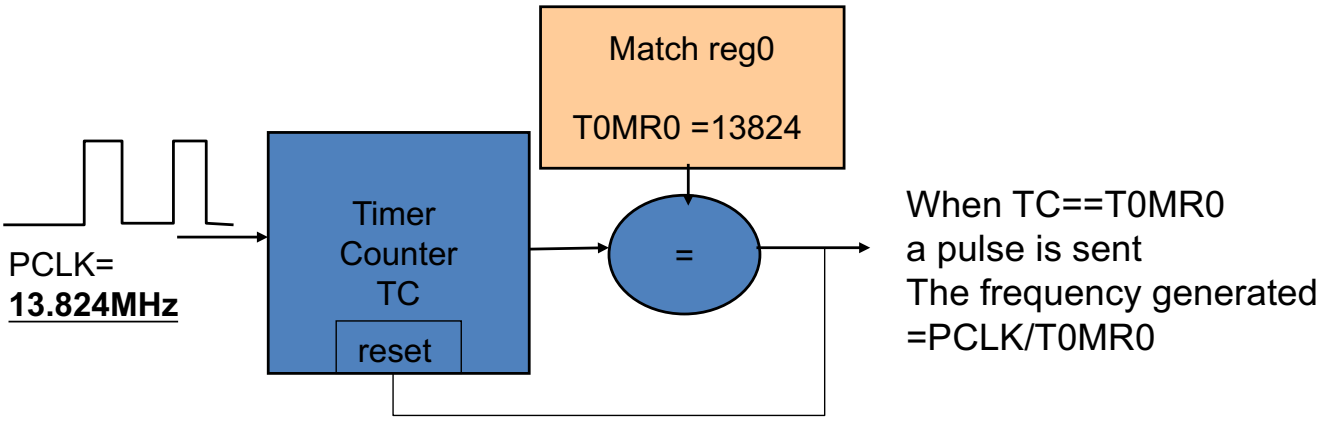
- T0MCR = 3; // Interrupt and Reset on MR0
- T0TCR = 1; // Timer0 Enable
- VICVectAddr0 = (unsigned long)IRQ_Exception;
- // set interrupt vector in 0 **(This becomes the highest priory interrupt)**
- VICVectCntl0 = 0x20 | 4; // use it for Timer 0 Interrupt
- VICIntEnable = 0x00000010; // Enable Timer0 Interrupt

Summary of Clocks
One oscillator generates two outputs CCLK, PCLK



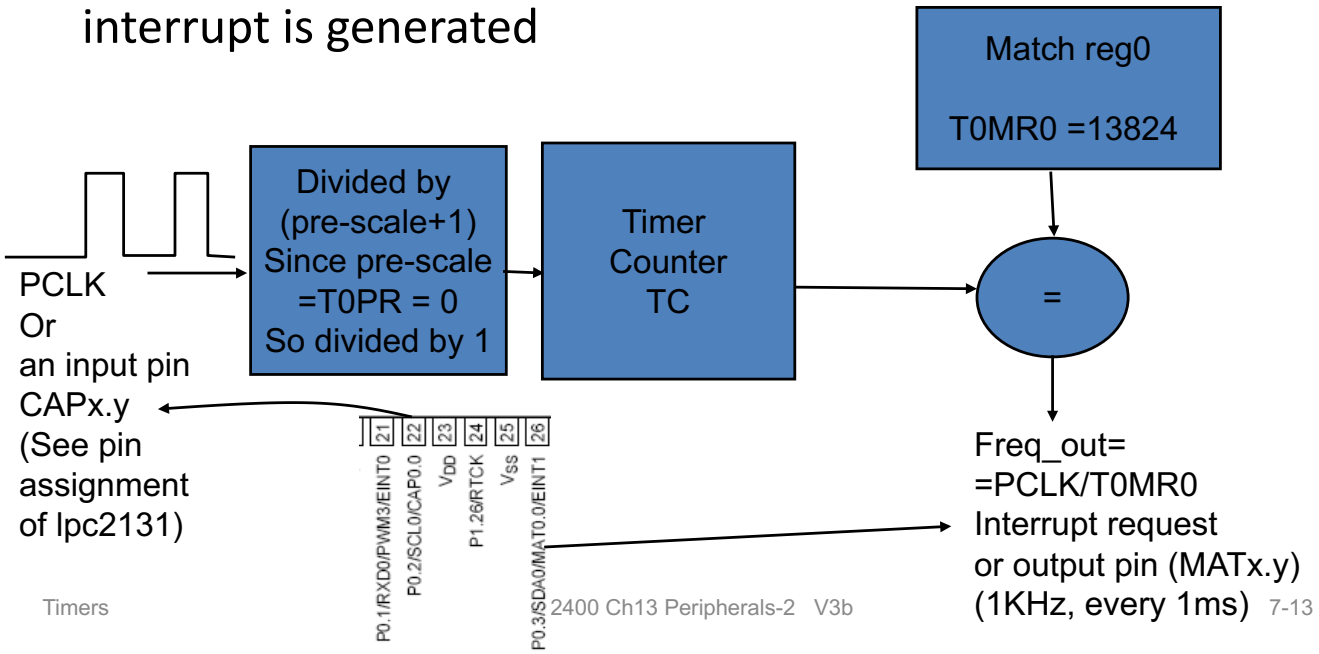
Concept of the timer Operation

- PCLK /freq_out=(11059200 x 5/4)/freq_out
- =13.824MHz /freq_out
- When timer counter (TC)=match reg0 (T0MR0), an pulse is generated, the the timer counter is reset



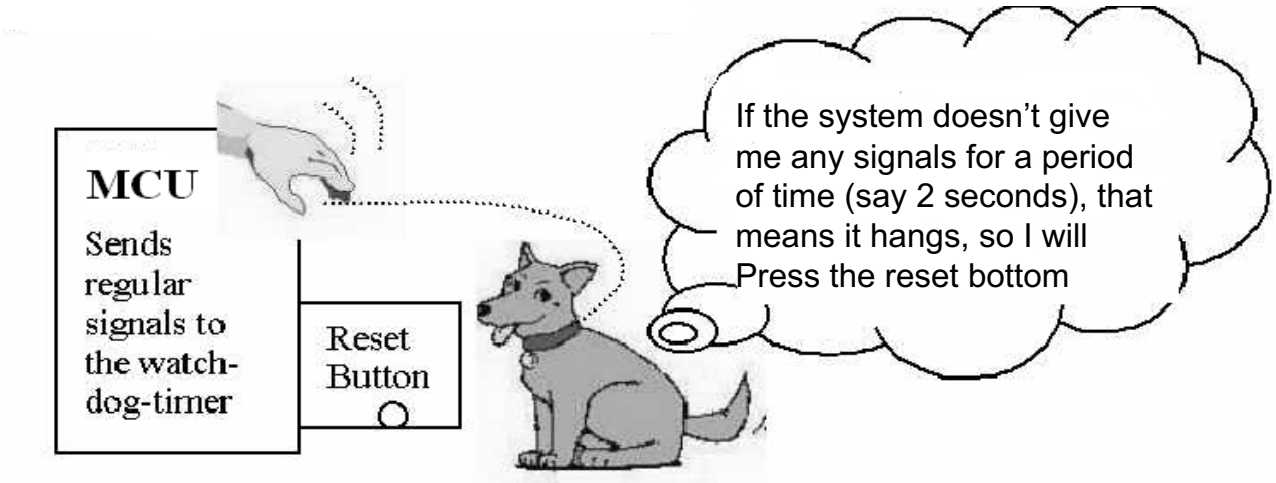
Example of a 1KHz=freq_out interrupt generator

- PCLK /freq_out= PCLK/1K=(11059200 x 5)/(4)=13.824 MHz/1K=13824
- When timer counter (TC)=match reg0 (T0MR0), an interrupt is generated



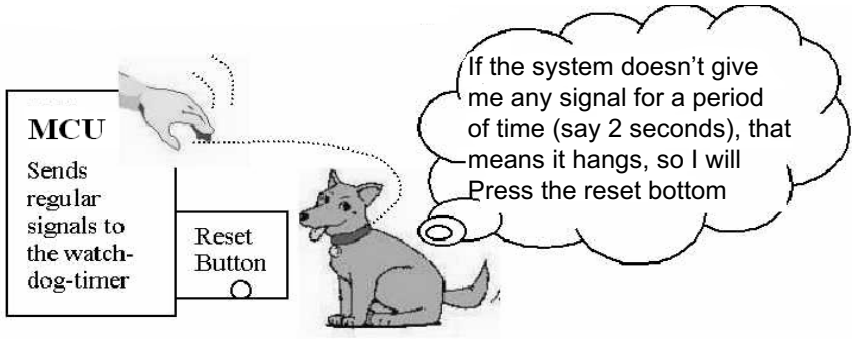
2) Watchdog timer

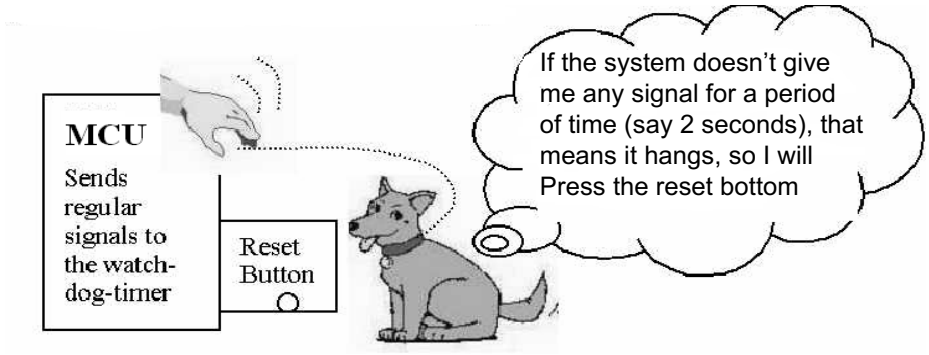
- For implementing fail safe systems



Example, solar power wireless telephone (register setting , see appendix)

- At remote area, maintenance is difficult
- If the software does not operate properly (hangs)
  - That means it sends no regular signals to the watch dog sensor
- Then
  - the watch-dog resets the system





- Main
- {
  - While(1)
  - { Do_the _neccessary();
  - Send_a_pulse_to_watch_dog();
  - }
- }

- If the software hangs, it will not Send_a_pulse_to_watch_dog();
- so the system is reset by the watch_dog_hardware

Example

http://www.keil.com/download/docs/317.asp

```
void feed_watchdog (void) {        /* Reload the watchdog timer      */
  WDFEED = 0xAA;
  WDFEED = 0x55;
}
void sendhex (int hex) {                /* Write Hex Digit to Serial Port  */
  if (hex > 9) sendchar('A' + (hex - 10));
  else        sendchar('0' +  hex);
}
void sendstr (char *p) {                /* Write string */
  while (*p) {
    sendchar (*p++);
  }
}
/* just waste time here          for demonstration */
void do_job (void)  {
  int i;
  for (i = 0; i < 10000; i++);
}
```


Demo to see how watchdog action

```
int main (void) {
  unsigned int i;
  init_serial();                    /* Initialize Serial Interface   */
  if( WDMOD & 0x04 ) {          /* Check for watchdog time out   */
    sendstr("Watchdog Reset Occurred\n");
    WDMOD &= ~0x04;            /* Clear time out flag          */
  }
  WDTC  = 0x2000;                /* Set watchdog time out value     */
  WDMOD = 0x03;                  /* Enable watchdog timer and reset */
  for(i = 0; i < 50; i++) {      /* for this 50 times do_job will run successfuly
    do_job ();                   /* the actual job of the CPU */
    feed_watchdog();             /*restart watchdog timer, for_loop will run until complete */
  }
  while (1) {                    /* Loop forever, but watch dog will rest the MCU */
    do_job ();                   /*so do_job( ) will not run for_ever, MCU will soon be reset*/
    /* no watchdog restart, watchdog reset will occur! */
  }
}
void feed_watchdog (void) { /* Reload the watchdog timer  */
    WDFEED = 0xAA;
    WDFEED = 0x55;
}
```

```
void feed_watchdog (void) { /*
Reload the watchdog timer  */
        WDFEED = 0xAA;
        WDFEED = 0x55;
}
```

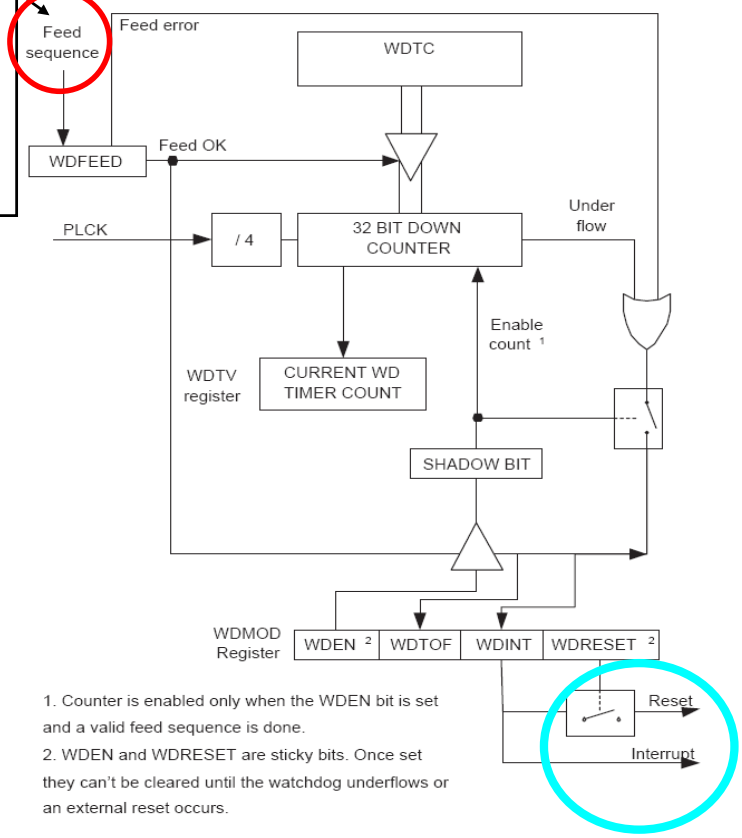


Fig 54. Watchdog block diagram

Watchdog
Block diagram

Applications of watchdog timers

Space robot
www.links9 99.net

-

Pay Telephone box
www.viewimages.com

-

-

Industrial machine
http://www.maxengineering.us/img/machine1.jpg

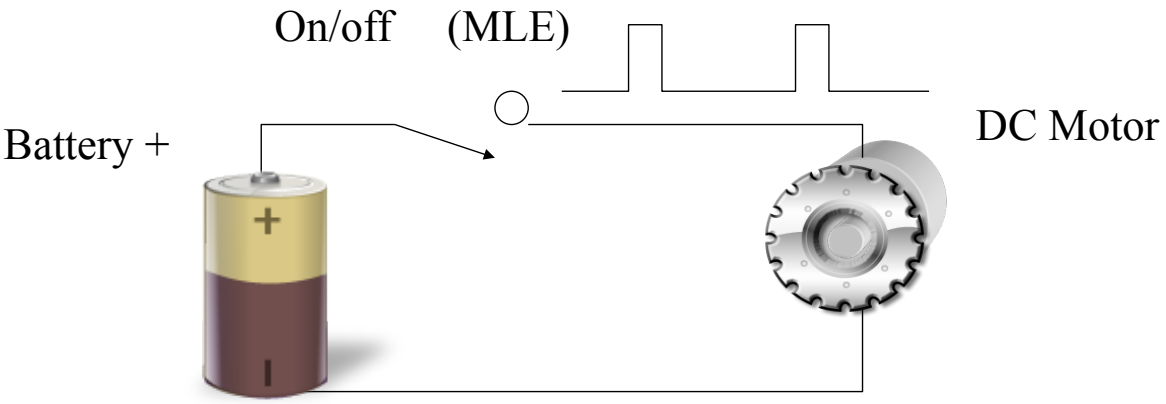
-

Solar power wireless emergency telephone
http://www.homepower.ca/

3) Pulse Width Modulation PWM unit

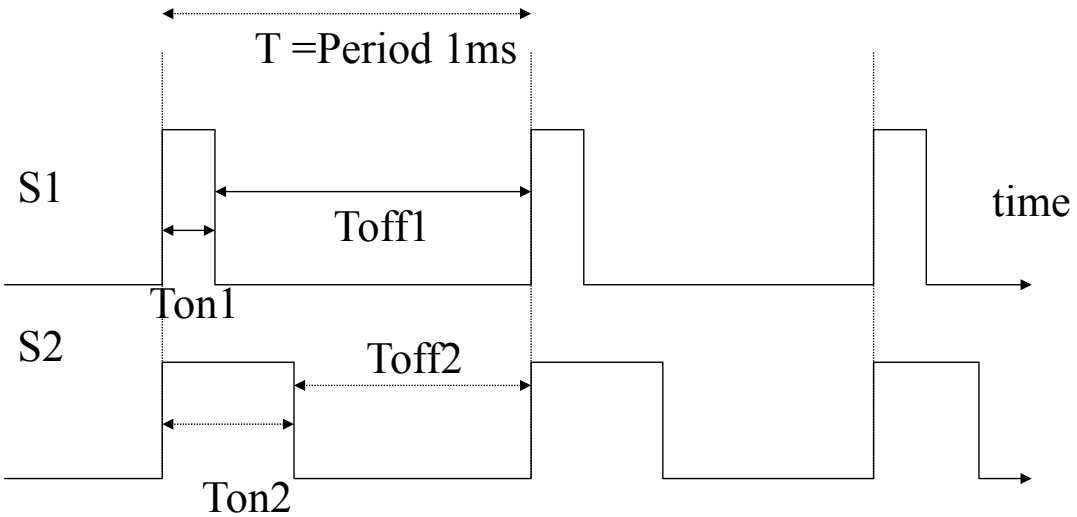
Use on-off time to control energy delivery

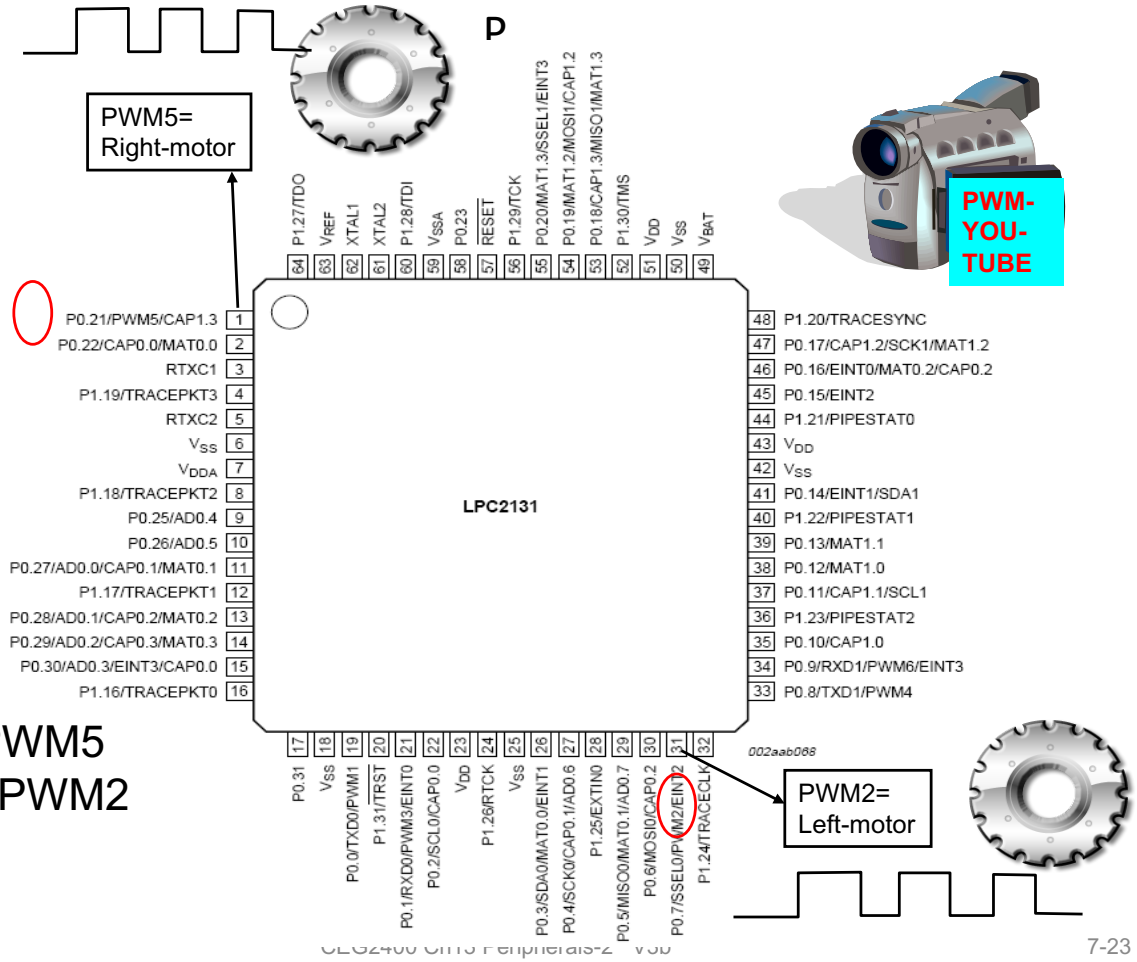
- The DC motor speed is determined by the on/off time of the motor enable signal MLE



Timing diagrams of pulse width modulation

- Comparing two pulse modulated signals S1,S2





Pin1=PWM5
Pin31=PWM2

Code for Pulse Width Modulation PWM
ARM06demo.c (with line numbers)

PCLK =13.824MHz (see previous slide)
The formula: will set PWM frequency = **PCLK/PWM_FREQ= 13.824MHz/** 276480=50Hz

- 17) #define PWM_FREQ 276480 //set PWM frequency to 50 Hz, since timer is 13824KHz
- //FREQ_of_PWM=13824000/276480=50

- 85) int main(void) {
- .....
- 89)long leftPWM,rightPWM;
- .....
- **// Initialize IO pin for PWM**
- **97) PINSEL1 |= 0x00000400; // set p0.21 to PWM5-right motor**
- **98) PINSEL0 |= 0x00008000; // set p0.7 to PWM2-left motor**
- .....

PINSEL1=0xE002 C004

11:10	P0.21	00	GPIO Port 0.21
		01	PWM5
		10	Reserved[1][2] or AD1.6[3]
		11	Capture 1.3 (Timer 1)

PINSEL0 =0xE002 C000

15:14	P0.7	00	GPIO Port 0.7
		01	SSEL0 (SPI0)
		10	PWM2
		11	EINT2

- 122) PWMPCR=0x2000; // enable pwm5
- 123) PWMPCR|=0x0400;// enable pwm2
- 124) PWMMCR=0x0002;
- 125) PWMMR0 = PWM_FREQ; //set PWM frequency to 50 Hz

See http://www.nxp.com/acrobat_download/usermanuals/UM10120_1.pdf

Code for Pulse Width Modulation PWM ARM06demo.c (refer to line numbers)

- 17) #define PWM_FREQ 276480 //set PWM frequency to 50 Hz,
- ……
- 122) PWMPCR=0x0000 2000; // enable pwm5;(bit 13 is set to 1)
- 123) PWMPCR|=0x0000 0400;// enable pwm2 ;(bit 10 is set to 1)
- 124) PWMMCR=0x0000 0002;
- 125) PWMMR0 = PWM_FREQ; //set PWM frequency to 50 Hz

Table 159: PWM Control Register (PWMPCR - address 0xE001 404C) bit description

Bit	Symbol	Value	Description	Reset value
13	PWMENA5	1	The PWM5 output enabled.	0
		0	The PWM5 output disabled.	
10	PWMENA2	1	The PWM2 output enabled.	0
		0	The PWM2 output disabled.	

Code for Pulse Width Modulation PWM ARM06demo.c (refer to line numbers)

- 17) #define PWM_FREQ 276480 //set PWM frequency to 50 Hz,
- ……
- 122) PWMPCR=0x0000 2000; // enable pwm5
- 123) PWMPCR|=0x0000 0400;// enable pwm2
- 124) PWMMCR=0x0000 0002; (BIT 1 IS SET TO 1)
- 125) PWMMR0 = PWM_FREQ; //set PWM frequency to 50 Hz

15.4.7 PWM Match Control Register (PWMMCR - 0xE001 4014)

The PWM Match Control Register is used to control what operations are performed when one of the PWM Match Registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 158.

Table 158: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.	0
		0	This feature is disabled.	

Code for Pulse Width Modulation PWM ARM06demo.c (refer to line numbers)

- 17) #define PWM_FREQ 276480 //set PWM frequency to 50 Hz,
- ……
- 122) PWMPCR=0x0000 2000; // enable pwm5
- 123) PWMPCR|=0x0000 0400;// enable pwm2
- 124) PWMMCR=0x0000 0002;
- 125) PWMMR0 = PWM_FREQ; //set PWM frequency to 50 Hz

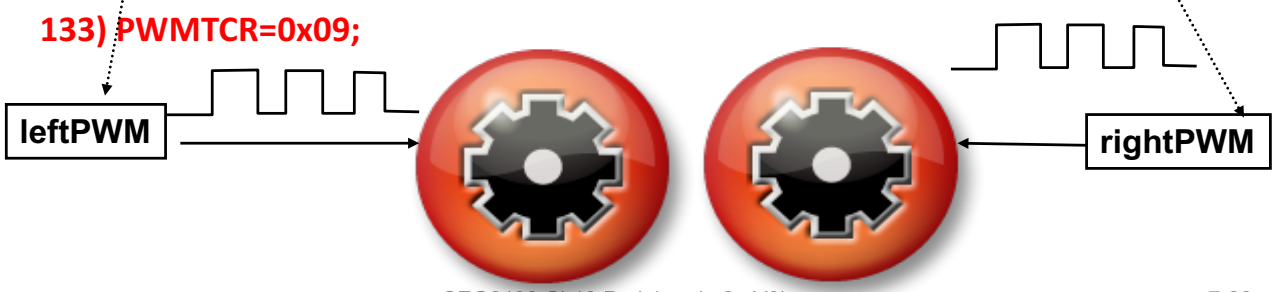
15.4.6 PWM Match Registers (PWMMR0 - PWMMR6)

The 32-bit PWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

PCLK =13.824MHz (see previous slide)
The formula: will set PWM frequency = **PCLK/PWM_FREQ= 13.824MHz/** 276480=50Hz

Code for Pulse Width Modulation PWM

-
- 17) #define PWM_FREQ 276480
- :
- **127) //set robot to full speed**
- **128) leftPWM=PWM_FREQ;//set a value you prefer**
- **129) rightPWM=PWM_FREQ; //a value you prefer**
- **130) PWMMR2 = leftPWM;// left motor PWM width to full speed**
- **131) PWMMR5 = rightPWM;//right motor PWM width to full**
- **132) PWMLER = 0x25; //enable match 0,2,5 latch to effective**
- **133) PWMTCR=0x09;**



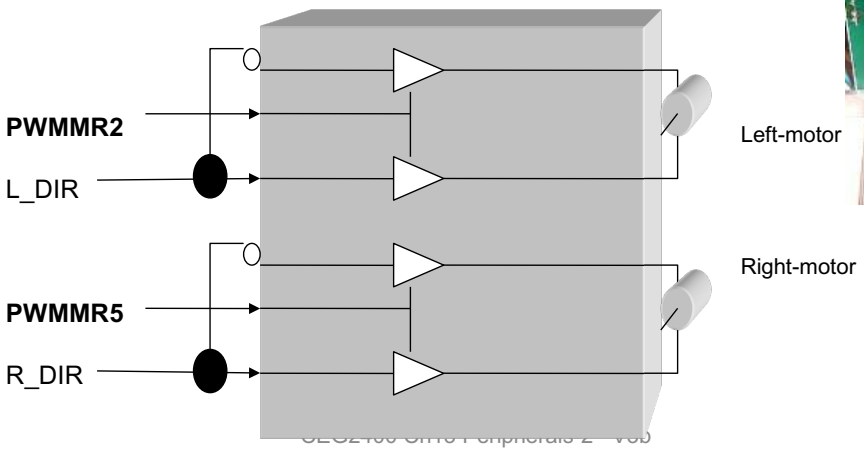
133) PWMTCR=0x09;//=0000 1001B, enable counter,PWM

Table 157: PWM Timer Control Register (PWMTCR - address 0xE001 4004) bit description

Bit	Symbol	Description	Reset value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match 0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0
7:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Use L293 H bright circuit

- A chip for generating enough current to drive 2 motors controlled by 4 signals



Exercise 13.2 Application– driving a robot Fill in "?__"

Left-motor forward
P0.16 =L_DIR =?___
P0.17=L_DIRinv=?__

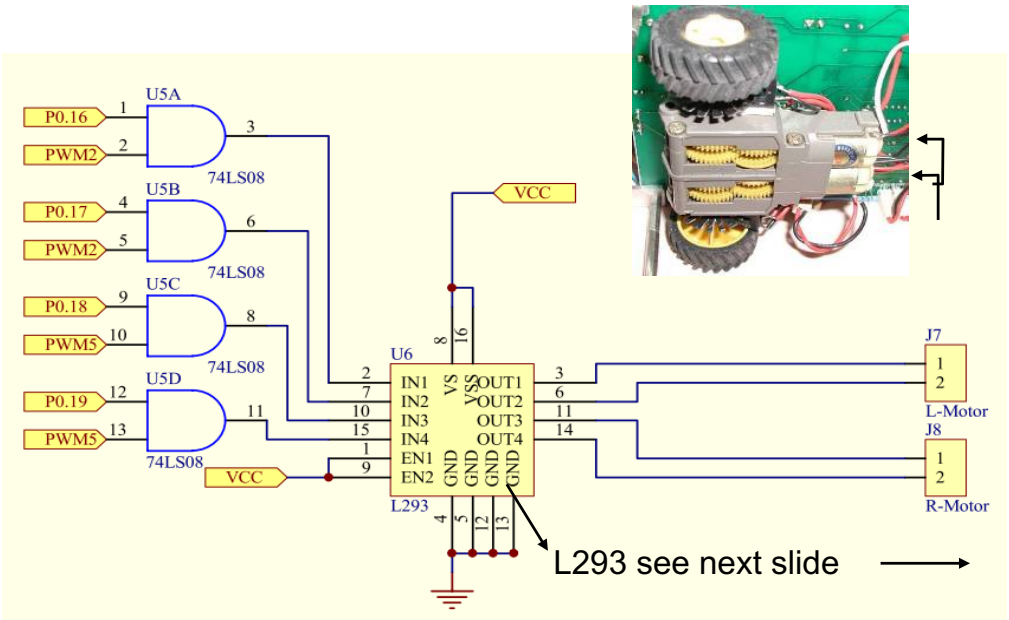
Left motor backward
P0.16 =L_DIR=?__
P0.17= L_DIRinv=?__

Left-motor speed
=**PWMMR2**

Right-motor forward
P0.18 = R_DIR =?__
P0.19= R_DIRinv=?__

Left motor backward
P0.18 = R_DIR=?__
P0.19= R_DIRinv=?__

Right-motor speed
=**PWMMR5**



Setting drive direction pins

- 18) #define L_DIR 0x00010000 //set p0.16 left motor dir.
- 19) #define L_DIRinv 0x00020000 //set p0.17 inverted left motor dir.
- 20) #define R_DIR 0x00040000 //set p0.18 right motor dir.
- 21) #define R_DIRinv 0x00080000 //p0.19 inverted right motor dir.
- 22) #define TEST_PIN 0x00010000 //set p1.16 as Test pin
- :
- 135) //set p0.16-p0.19 as output
- 136) IO0DIR|=L_DIR; //p0.16
- 137) IO0DIR|=L_DIRinv; //p0.17
- 138) IO0DIR|=R_DIR; //p0.18
- 139) IO0DIR|=R_DIRinv; //p0.19
- 140) IO1DIR|=TEST PIN;// p1.16 as Outputs

Set p0.16-19 as output pins

Table 70: GPIO Direction Register 0 (IO0DIR - address 0xE002 8008) bit description

Bit	Symbol	Value	Description	Reset value
31:0	P0xDIR		Direction control bits. Bit 0 in IO0DIR controls P0.0 ... Bit 30 in IO0DIR controls P0.30.	0x0000 0000
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Four line (170-173) to start the robot move forward

- 170) IO0SET|=L_DIR;
- 171) IO0CLR|=L_DIRinv;
- 172) IO0SET|=R_DIRinv;
- 173) IO0CLR|=R_DIR;

Table 66: GPIO Output Set register 0 (IO0SET - address 0xE002 8004 bit description

Bit	Symbol	Description	Reset value
31:0	P0xSET	Output value SET bits. Bit 0 in IO0SET corresponds to P0.0 ... Bit 31 in IO0SET corresponds to P0.31.	0x0000 0000

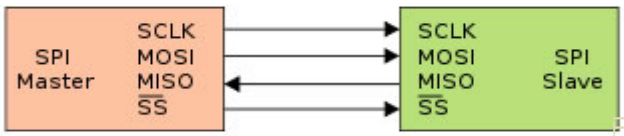
Table 68: GPIO Output Clear register 0 (IO0CLR - address 0xE002 800C) bit description

Bit	Symbol	Description	Reset value
31:0	P0xCLR	Output value CLEAR bits. Bit 0 in IO0CLR corresponds to P0.0 ... Bit 31 in IO0CLR corresponds to P0.31.	0x0000 0000

Serial Peripheral Interface (SPI)

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Introduction



Serial Peripheral Interface
http://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/SPI_single_slave.svg/350px-SPI_single_slave.svg.png

- What is it?
- Basic Serial Peripheral Interface (SPI)
- Capabilities
- Protocol
- Pro / Cons and Competitor
- Uses
- Conclusion

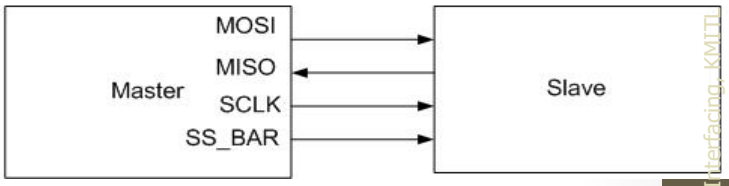
SPI Basics

- A communication protocol using 4 wires
  - Also known as a 4 wire bus
- Used to communicate across small distances
- Multiple Slaves, Single Master
- Synchronized

Capabilities of SPI

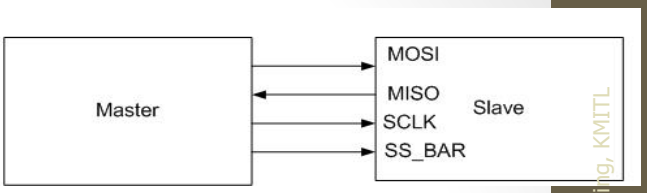
- Always Full Duplex
  - Communicating in two directions at the same time
- Multiple Mbps transmission speed
- Transfers data in 4 to 16 bit characters
- Multiple slaves
  - Daisy-chaining possible

Protocol



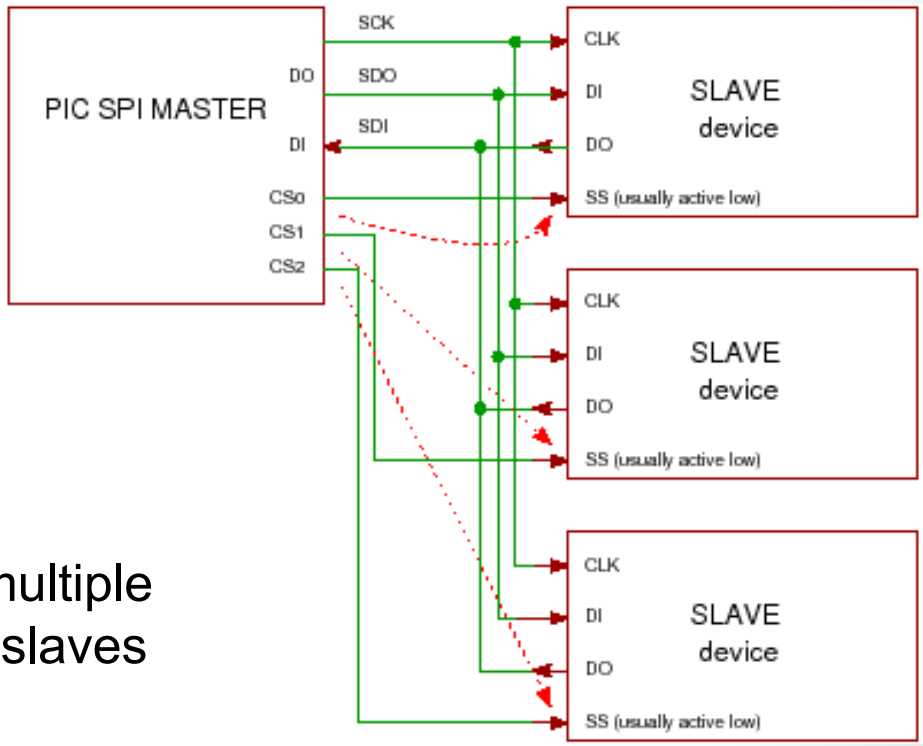
- Wires:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - System Clock (SCLK)
  - Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data

Wires in Detail



- MOSI – Carries data out of Master to Slave
- MISO – Carries data from Slave to Master
  - Both signals happen for every transmission
- SS_BAR – Unique line to select a slave
- SCLK – Master produced clock to synchronize data transfer

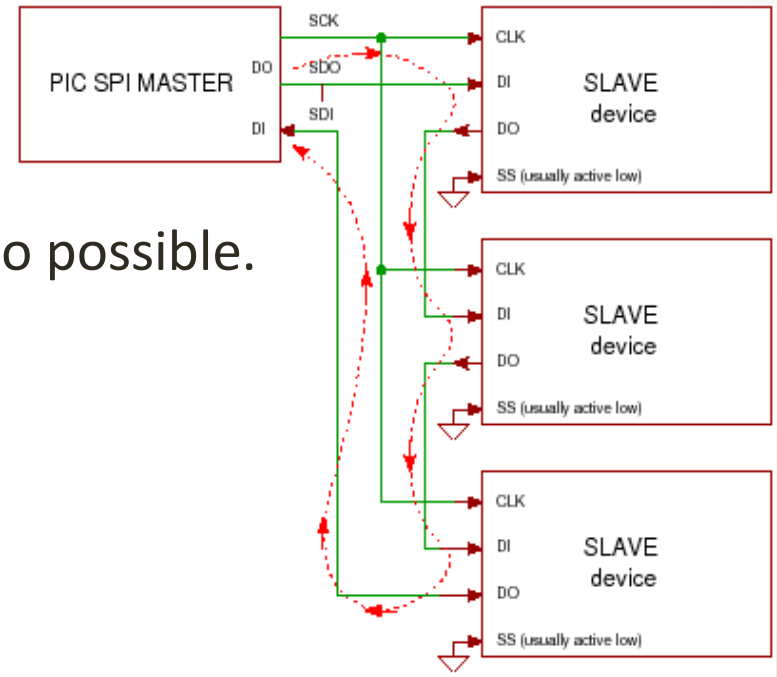
Master with Independent Slaves



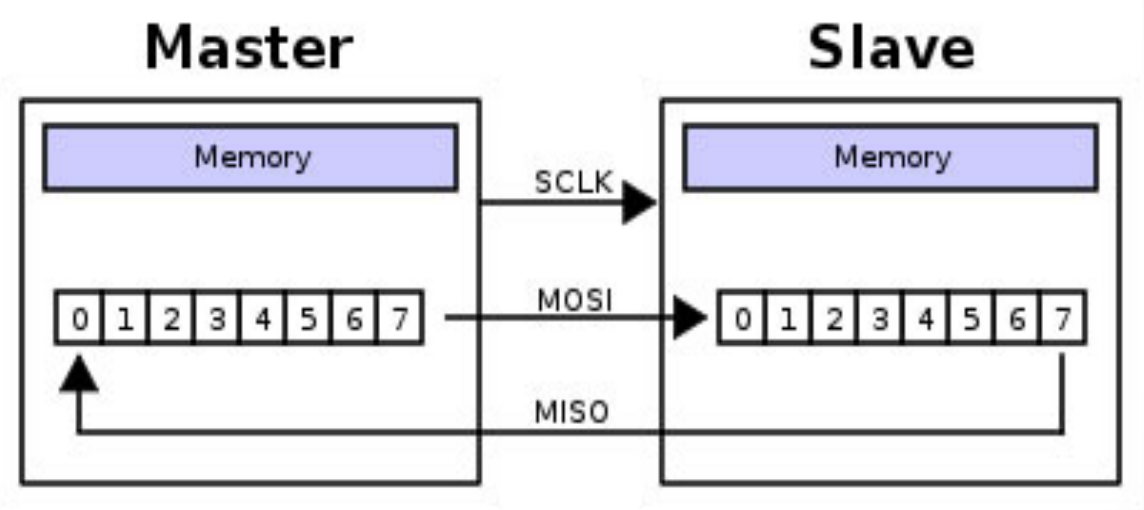
Master and multiple independent slaves

Master-Slaves/Daisy Chain

- Less wiring
- Slow data xFer
- Broadcast is also possible.



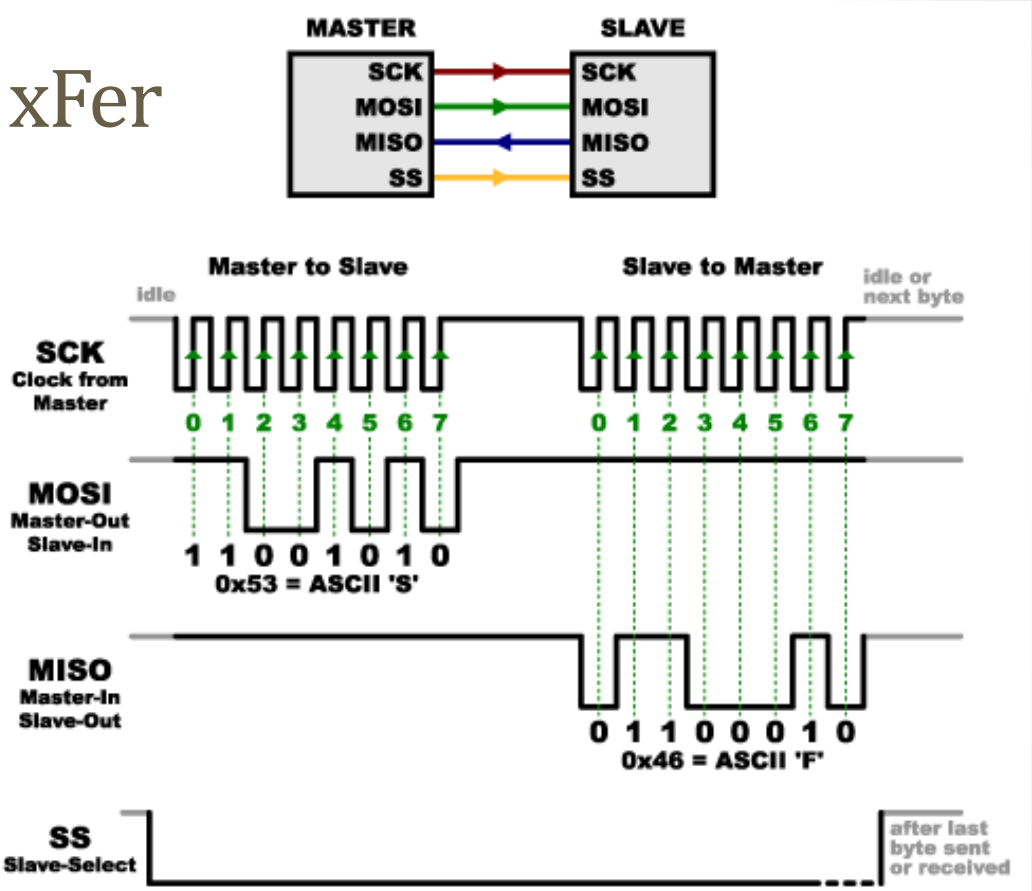
Shifting Protocol



Master shifts out data to Slave, and shift in data from Slave

http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

Data xFer



Clock Phase (Advanced)

SPI-mode	CPOL	CPHA	Sampling
0	0	0	Leading (Rising) Edge
1	0	1	Trailing (Falling) Edge
2	1	0	Leading (Falling) Edge
3	1	1	Trailing (Rising) Edge

- Two phases and two polarities of clock => Four modes
- Master and selected slave must be in the same mode
- Master must change polarity and phase to communicate with slaves of different numbers

Clock & Sampling

Figure 76. SPI Transfer Format with CPHA = 0

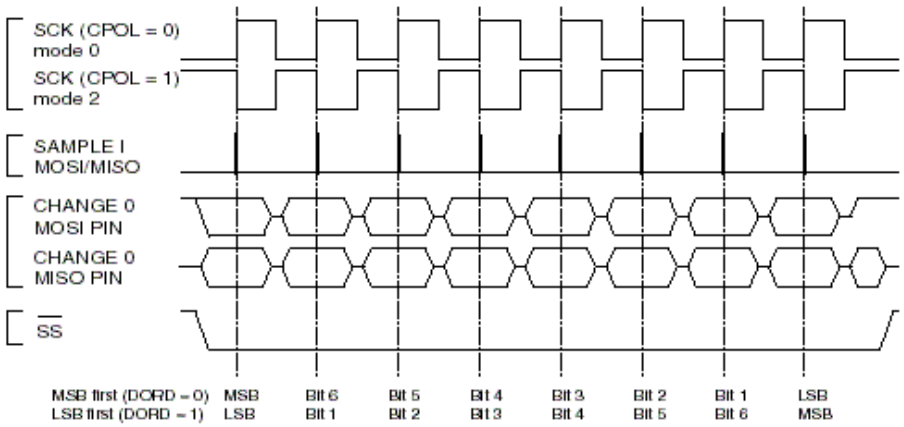
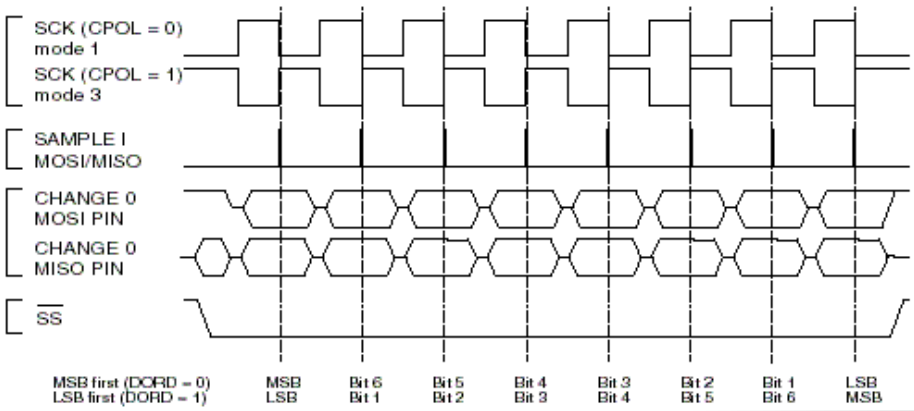


Figure 77. SPI Transfer Format with CPHA = 1



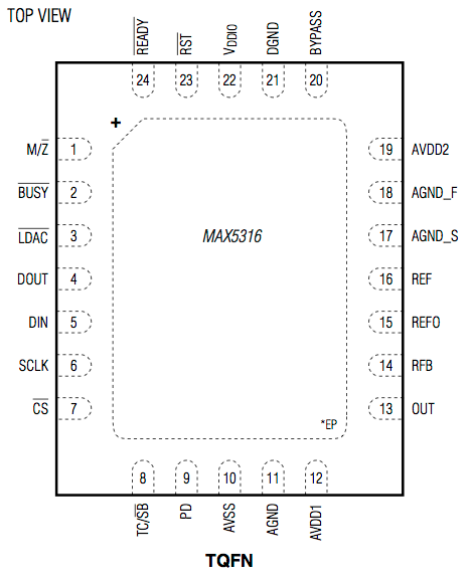
SPI Data Transfer Modes

- These four modes are the combinations of CPOL and CPHA.
- Modes 0 and 3 are the most common.
- With SPI modes 0 and 3, data is always latched in on the rising edge of SCK and always output on the falling edge of SCK.

Example SPI devices

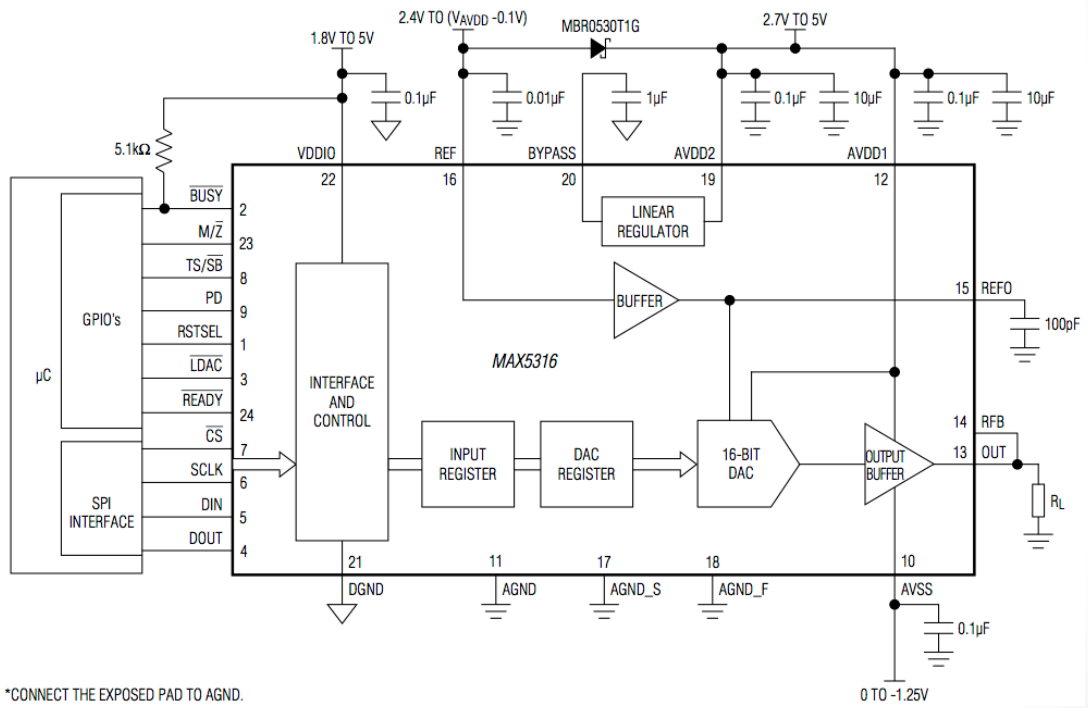
- 25LC020A - 2K SPI Bus Serial EEPROM
- TC77-5.0 - Thermal Sensor with SPI Interface
- MCP3201 - 2.7V 12-Bit A/D Converter with SPI Serial Interface
- MCP4822 - 12-Bit DAC with Internal VREF and SPI Interface
- MCP41010 - Single/Dual Digital Potentiometer with SPI Interface
- MCP6S92 - Single-Ended, Rail-to-Rail I/O, Low-Gain PGA
- MCP23S08 - 8-Bit I/O Expander with Serial Interface

Example: SPI-DAC



- MAXIM MAX5316
- 16-bit +/- 1 LSB Accuracy
- 2.7V to 5.5V supply voltage range
- 50MHz 3-wire SPI interface

MAX5316 Diagram



MAX5316 Timing Diagram

Table 5. SPI Command and Data Mapping with Clock Falling Edges

CLOCK EDGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DIN	R3	R2	R1	R0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X
DOUT	0	R3	R2	R1	R0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X

Note that 'X' is don't care.

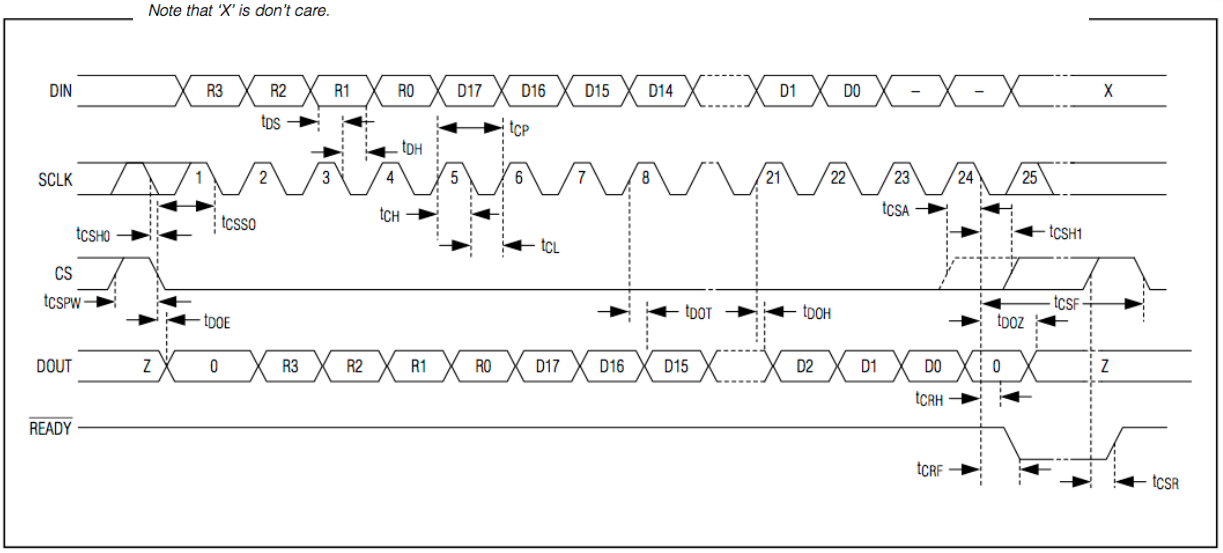
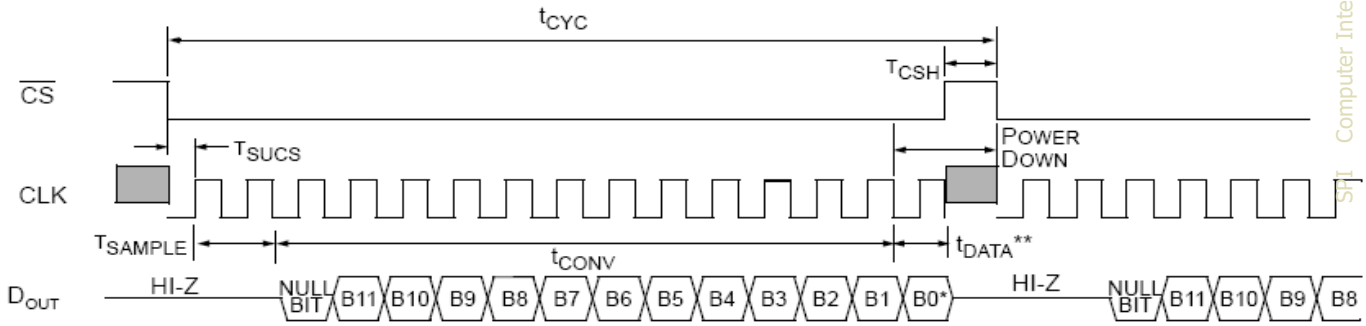


Figure 1. Serial Interface Timing Diagram, Stand-Alone Operation

Example - reading data from the MCP3201 (12 bit ADC)



Initialization

```
/*Initialise SSP in SPI mode */
void init_SSP1(void)
{

  IODIR0  = IODIR0|PIN_SSP1_SS;  //Enable SSEL pin as GPIO
  PINSEL0   |= 0x000A8000;       //Enable SSP1 pins with SSEL as GPIO
  //PINSEL0 |= 0x000AA000;       //Enable SSP1 pins with SSEL as SS

  /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR
    (serial clock Rate (presscaler) is 15 */

  SSP1CR0 = 0xFF07;
  SSP1CR1 = 0x00000002;      //Configure as SPI Master

  /* SSPCPSR clock prescale register, master mode, minimum divisor
    is 0x02 MAX 254 */
  SSP1CPSR = 15;
}
```


Sender

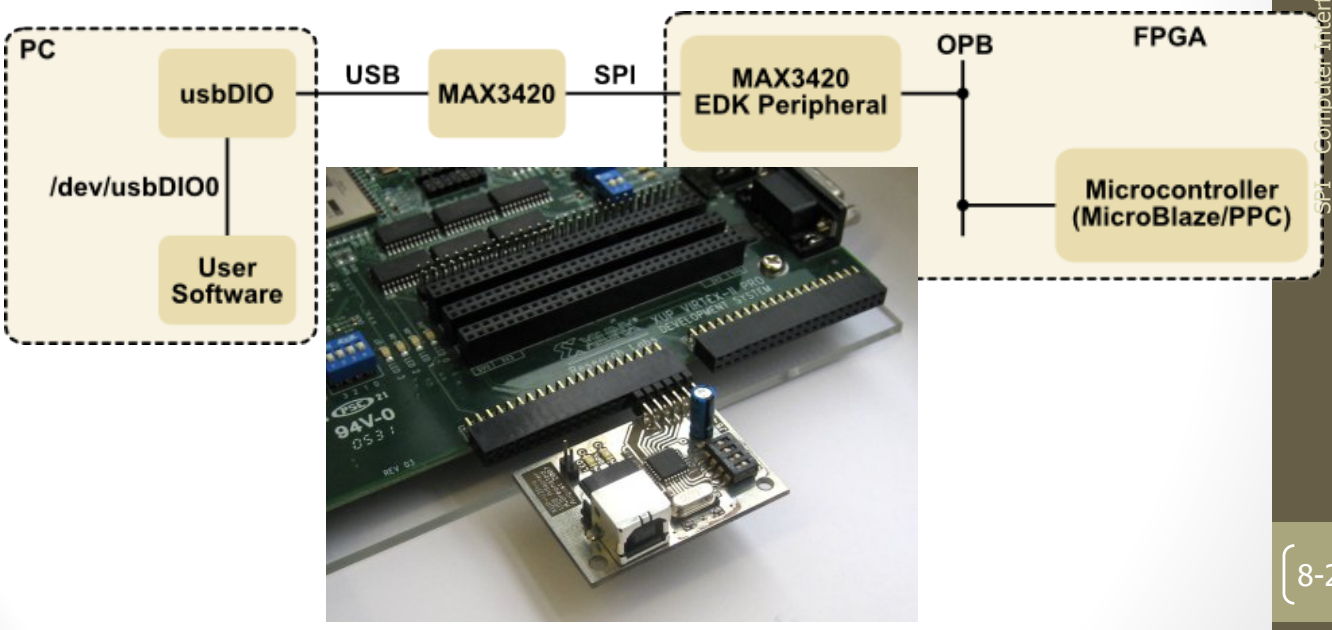
```
/*Send data to the SSP the start address and number of bytes (size of
array) must be given*/
void SSP1SendData(unsigned char *data_add,unsigned char arr_size)
{
  int i;
  SSP1_SS(0);         //Select slave active low
  for(i=0;i<arr_size;i++)
  {
    while ( (SSP1SR & SSPSR_BSY) ); // Wait until the Busy bit is
                                    // cleared
    SSP1DR = *data_add;
    while ((SSP1SR & SSPSR_BSY) );//Wait until the Busy bit is cleared
    data_add++;
   }
  SSP1_SS(1);             //de-select slave
}
```

Receiver

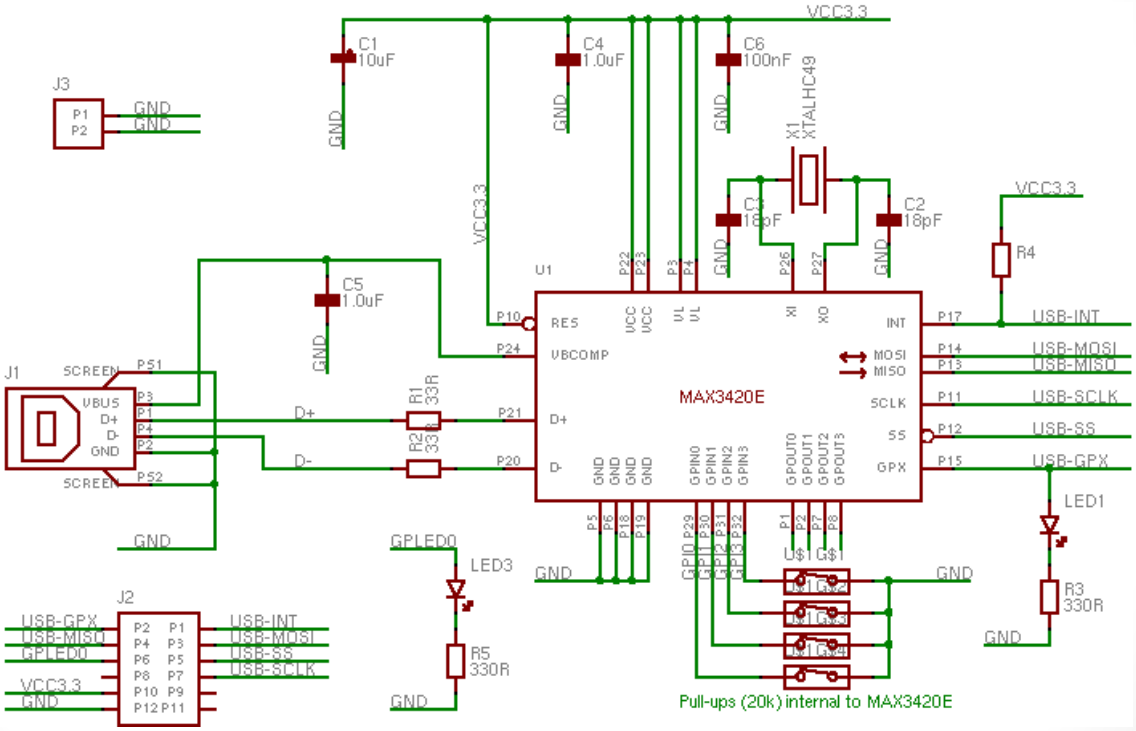
```
/*Receive data to the SSP the start address of where the data should
be stored, size of the data must be given */

void SSP1ReceiveData(unsigned char *data_add,unsigned char arr_size)
{
   int i;
   SSP1_SS(0);
   for(i=0;i<arr_size;i++)
   {
    SSP1DR = 0x00;                  //dummy write
    while((SSP1SR & SSPSR_BSY)); //Wait until the Busy bit is cleared
    *data_add= SSP1DR;
    while ((SSP1SR & SSPSR_BSY) ); //Wait until the Busy bit is
cleared
     data_add++;
   }
   SSP1_SS(1);
}
```


USB-SPI Board: MAX3420



USB-SPI: MAX3420



Conclusion

- SPI – 4 wire serial bus protocol
  - MOSI MISO SS SCLK wires
- Full duplex
- Multiple slaves, One master
- Best for point-to-point streaming data
- Easily Supported

Pros and Cons

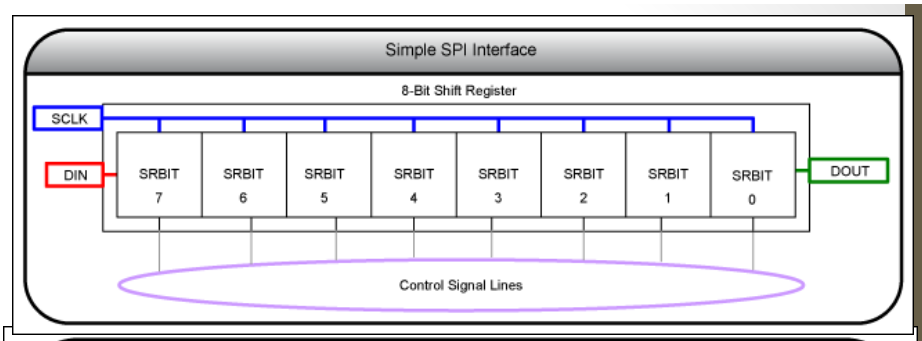
Pros:

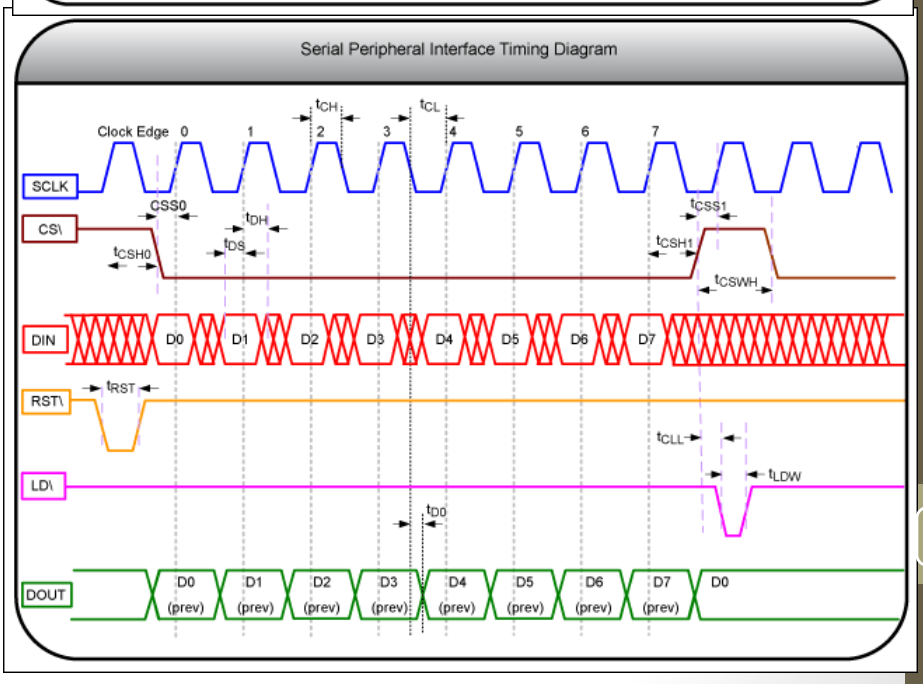
- Fast and easy
  - Fast for point-to-point connections
  - Easily allows streaming/Constant data inflow
  - No addressing/Simple to implement
- Everyone supports it

Cons:

- SS makes multiple slaves very complicated
- No acknowledgement ability
- No inherent arbitration
- No flow control

SPI Shift Register





LCD DISPLAY TECHNOLOGY

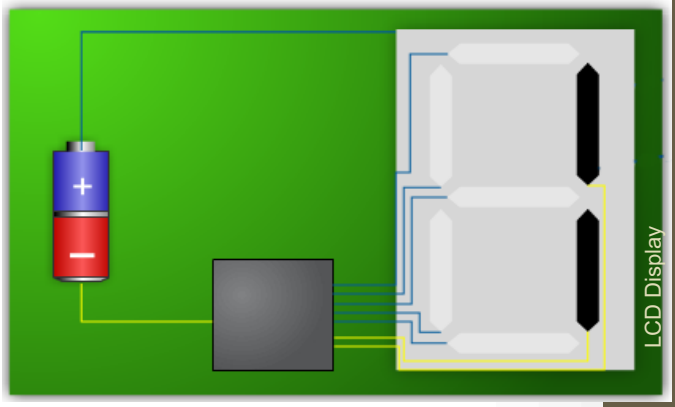
ผศ.ดร. สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Topics

- LED Direct Address Display
- Passive Matrix Display
- Active Matrix Display
- Twisted Nematic LCD Display
- STM32 Board LCD Display

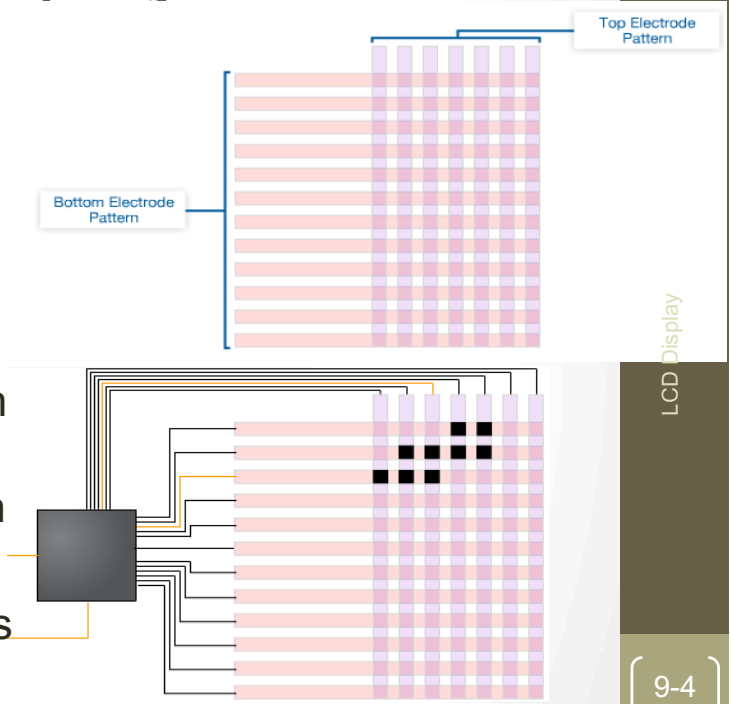
Direct Address Display

- When the display include limited variable components such as
  - Watches
  - Calculators
- Simple electronics is used to control the components



Passive Matrix Display

- Passive matrix display has
  - Rows of electrodes on one piece of glass.
  - Columns of electrodes on the opposing piece of glass.
  - Complex electrical waveform control the voltage differential at the intersection of the electrodes.
- The intersection of the columns and rows are the pixels

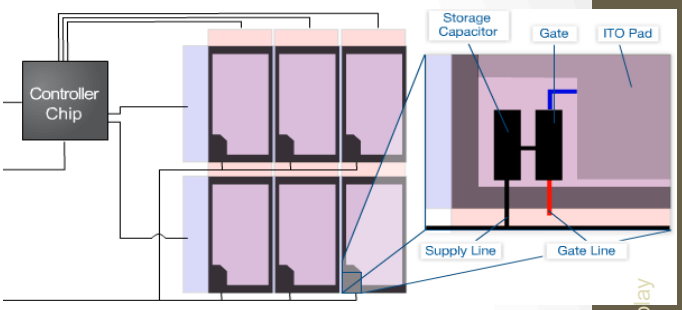


Passive Matrix Display
-disadvantages-

- As more rows and columns are added the range of the allowed voltage is reduced.
  - At high range adjacent channels interferes
  - Range limit reduces contrast
  - Limit the types of useful liquid crystal.
- It is usually limited to about 50 rows
- Twisted nematic (TN) Display work best with large voltage variation.
  - It can not be used in Passive Matrix Display

Active Matrix Display

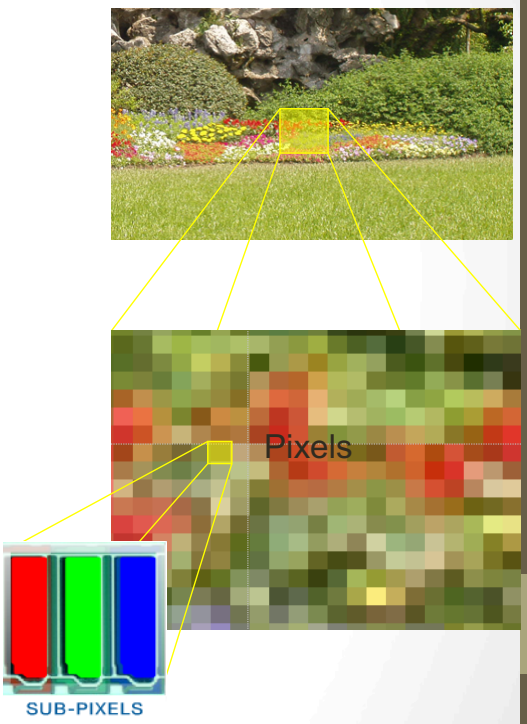
- Allow very high resolution
- Each sub-pixel is individually controlled by an isolated thin-film transistor (TFT).
- It allows the electrical signal for each sub-pixel to avoid influencing adjacent elements.
- The TFT is patterned into the glass layer



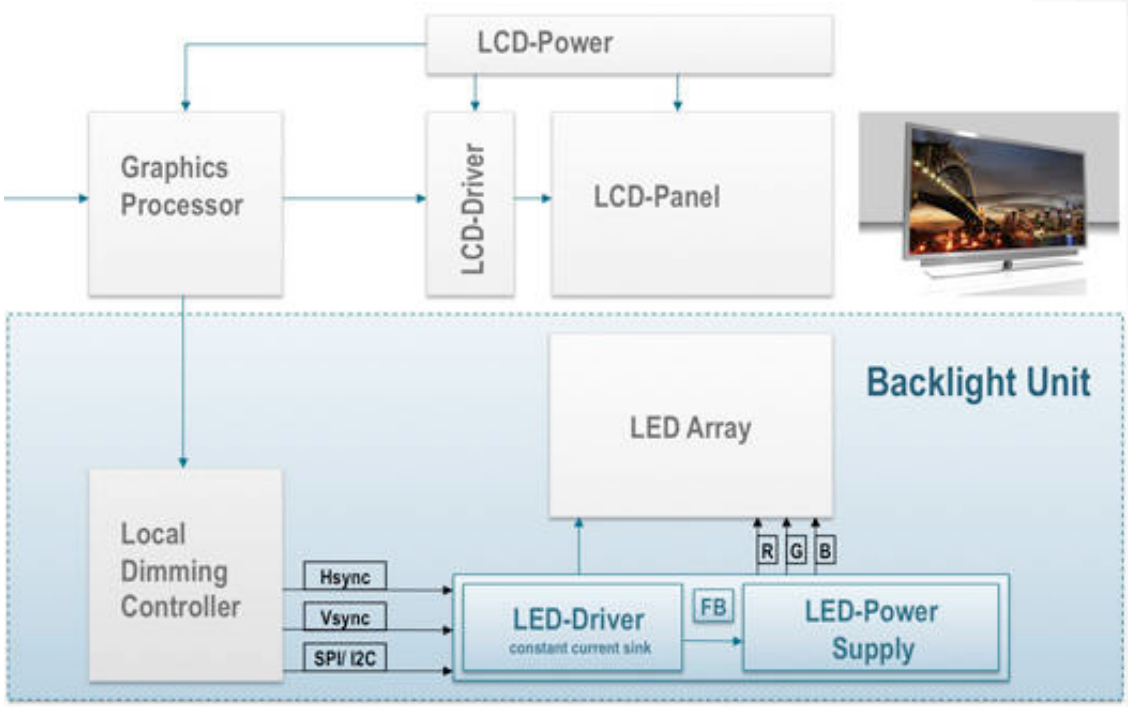
A display with 1024x768 resolution Include 1024x768x3 = 2,359,296 sub-pixels

Digital Images and Pixels

- A digital image is a binary (digital) representation of a two-dimensional pictorial data.
- Digital images may have a raster or vector representation.
- Raster Images defined over a 2D grid of picture elements, called pixels.
- A pixel is the basic item of a raster image and include intensity or color value.

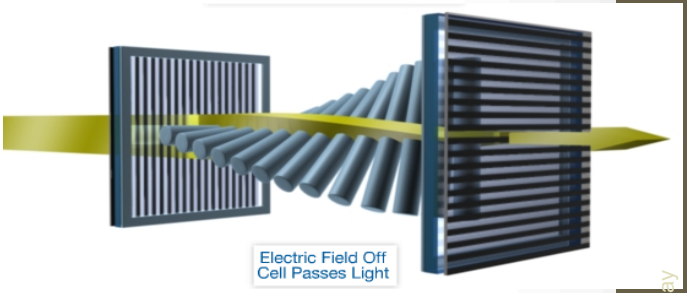


LCD Display Structure



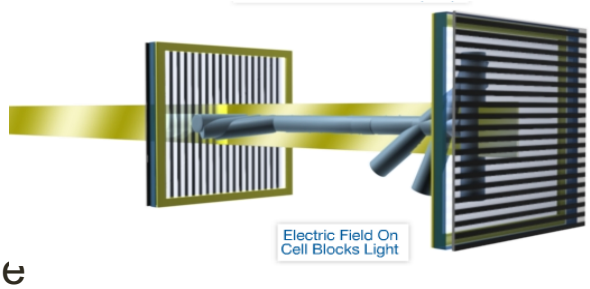
Twisted Nematic (TN) Display

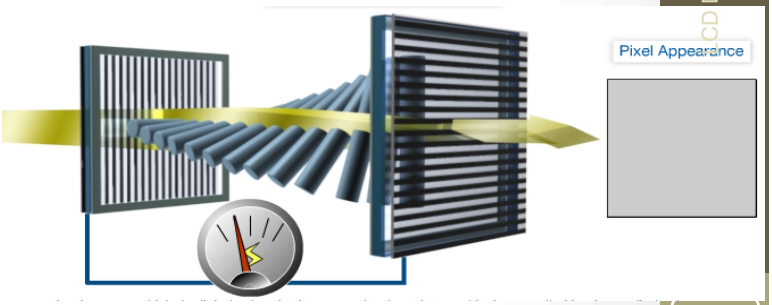
- Is the most common LCD Display.
- The two alignments layer for the liquid crystal material are orthogonal.
- The light entering the polarize panel rotates by the twist in the liquid crystal and allowing it to pass through the second polarize



Twisted Nematic (TN) Display

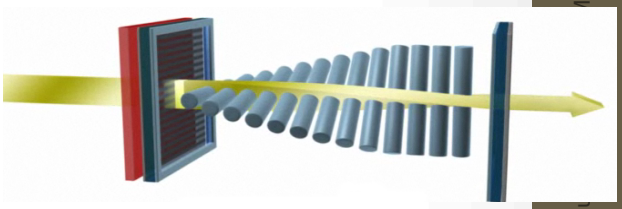
- The electric field is applied
  - The liquid crystal loses its twist.
  - A light to the electric field.
  - Prevents the rotation of the polarized light
  - The second polarizer absorbs the light.
- The applied voltage control the absorbed and transmitted light

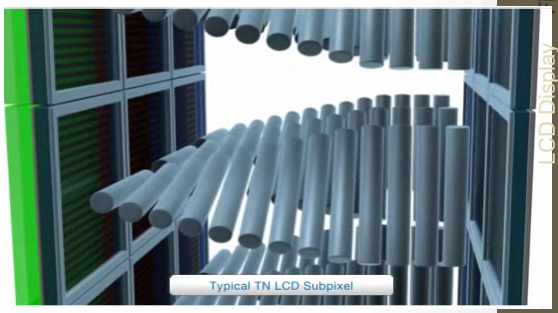




LCD (Liquid Crystal Display)

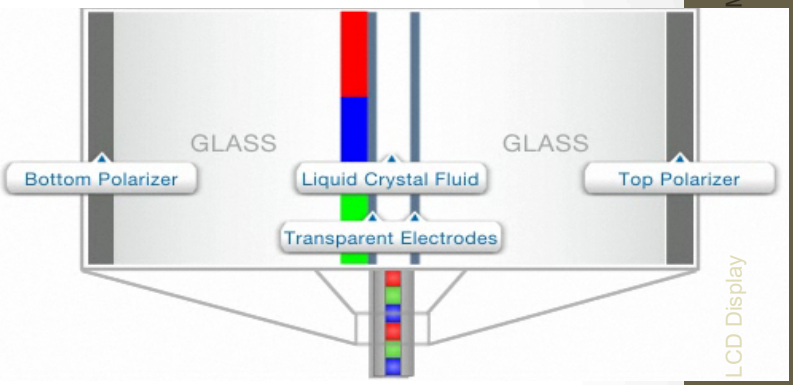
- LCD Panel is based on
  - A light valve for each pixel that turn the light on, off, or an intermediate level.
- Grid of such light valve for the LCD display panel.
- A back light and display enhancement films create the illumination.

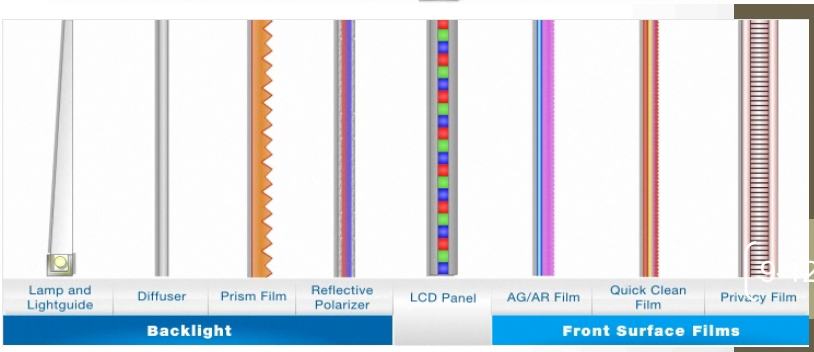




LCD-Display

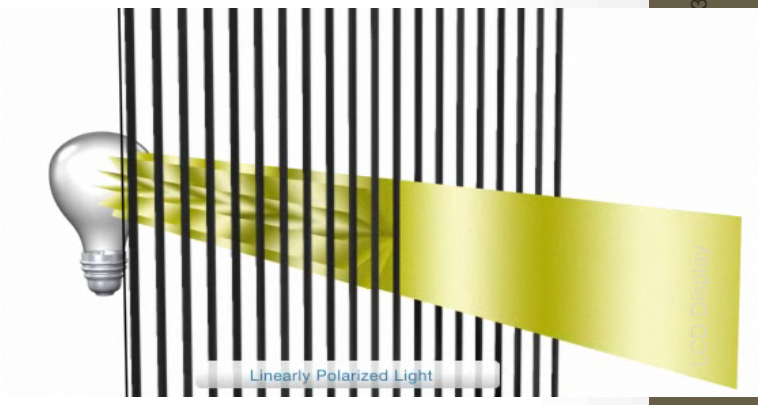
- Applying voltage to the electrodes changes the level of illumination in each sub-pixel
- The panel is sandwiched between
  - Front surface films to enhance display property
  - Backlight





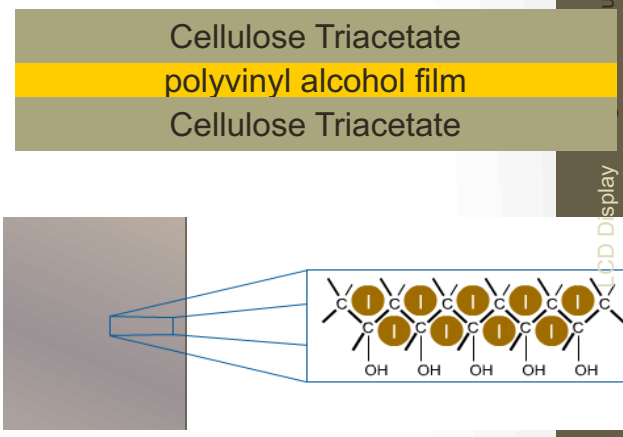
Linear Polarized Light

- Light usually vibrates in all direction
- A linear polarized light limit the vibration to one direction
- It absorbs the componen of light that vibrate in all other direction.
- LCD require light to vibrate in one direction



Iodine Based Polarizer

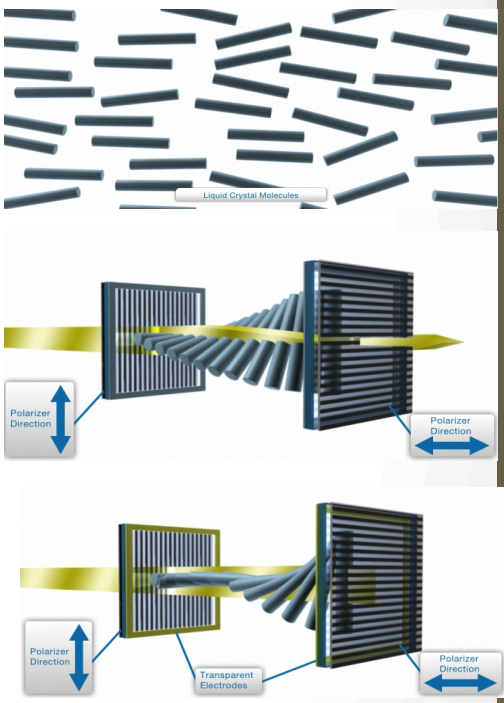
- Is the most common polarizer
- It is made by
  - Stretching a cast polyvinyl alcohol film (PVA) to align the iodine in turn.
  - Staining it with iodine
  - The stained PVA laminated between two slices of cellulose triacetate.
- The cellulose triacetate
  - Provide physical rigidity
  - Some degree of heat and humidity protection



polyvinyl alcohol film

About Liquid Crystal

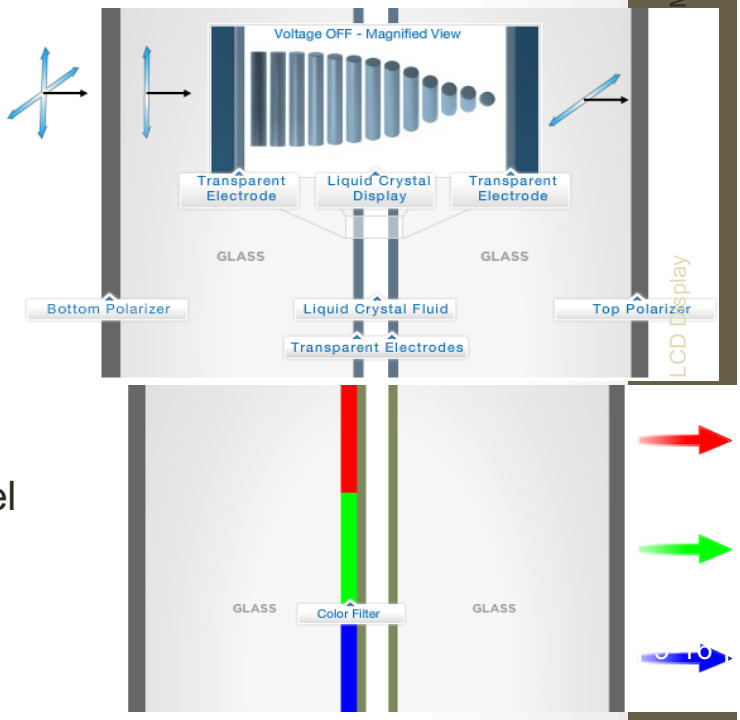
- Liquid crystal molecules can move freely while maintaining their orientation.
- It align itself to a polyimide film to the inside of a panel glass.
- When the two glass panels are not aligned the liquid crystal twists accordingly.
- The liquid crystal will also align to electric field.``



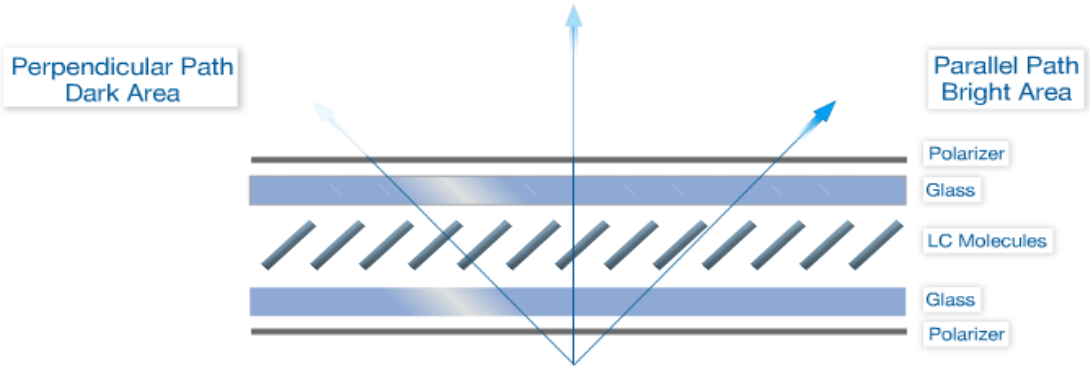
Figures are curtsy of 3M

Light Path

- The light passes through the polarizer.
- The voltage applied to the electrodes controls the liquid crystal orientation
- The liquid crystal orientation controls the rotation of the incoming polarized light.
- Color filters are used in color LCD, where each color sub-pixel is controlled individually



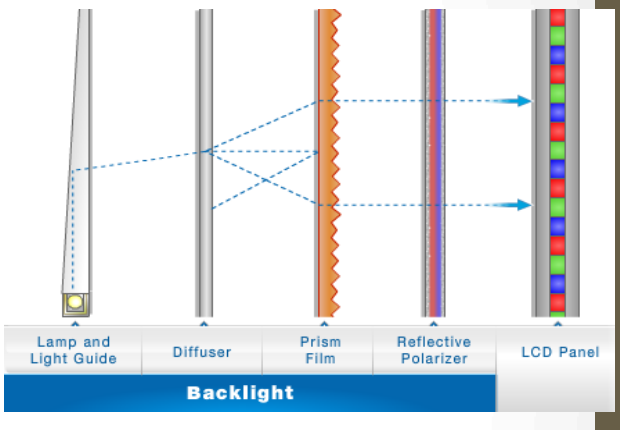
Twist Nematic Viewing Angle

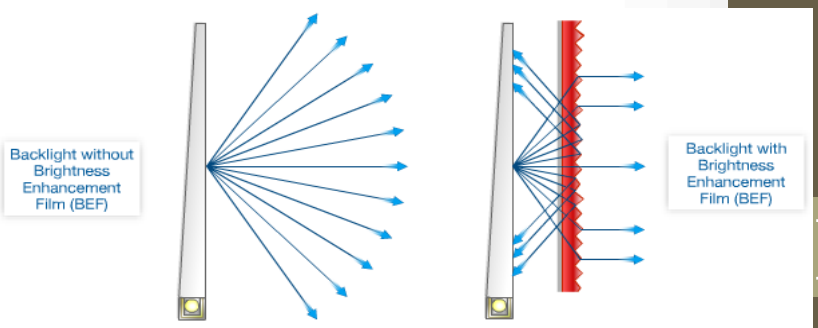


- Light passes through the TN display from one angle may be blocked in other angles by the twist of the liquid crystal.
- To overcome this limitation different approaches have use various orientations for the polarize layers.

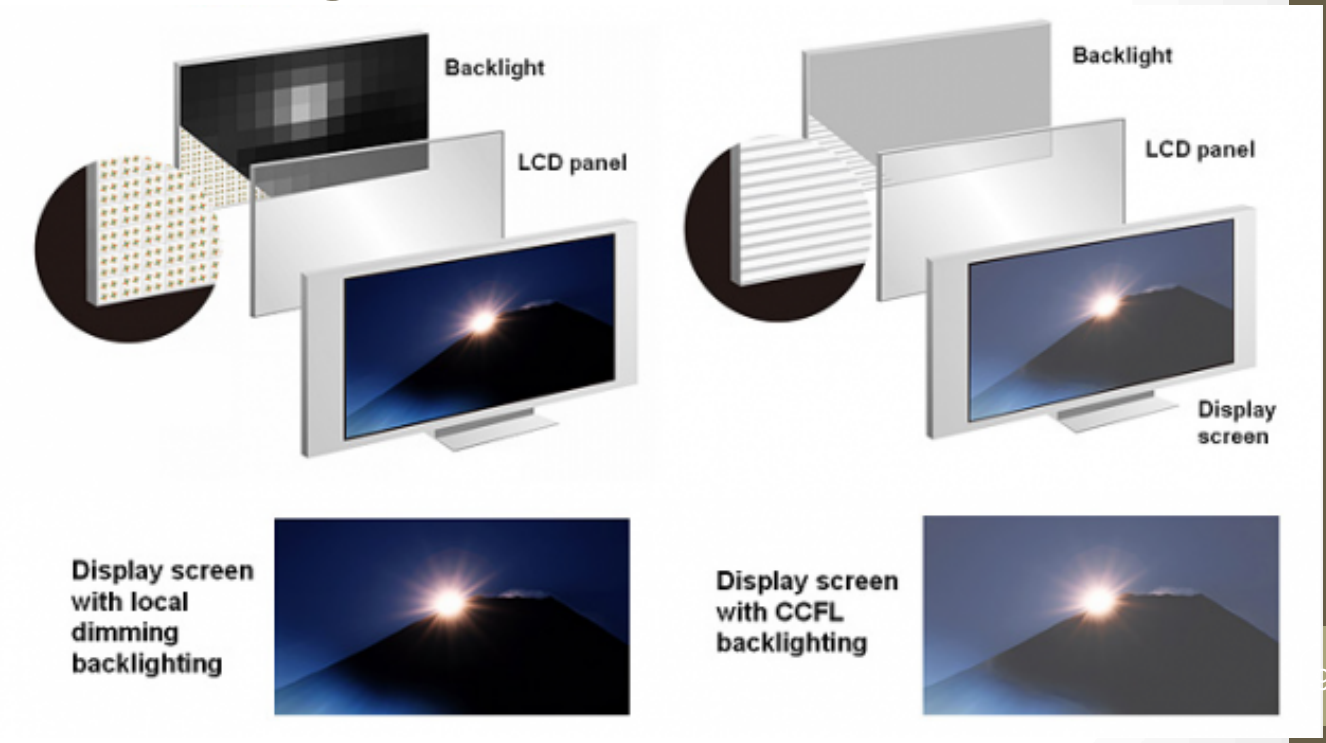
Back-Light

- The light generated by the backlight.
- The light is evenly distributed the light evenly over the LCD panel.
- Display enhancement films are placed between the light diffuser and the LCD panel. They aim to maximize the light reaching the observer.





Back-Light



Figures are curtsy of 3M

LCD Display

3-D LCD Displays

- This technology aims to generate stereoscopic 3D images with the need for glasses.
- The idea is based on combining
  - Active-matrix LCD with
  - Lenticular lens: an array of magnifying lenses, designed to magnify different images when viewed from slightly different angles.
- Applications
  - Industrial Visualization
  - Medical imaging
  - Entertainment

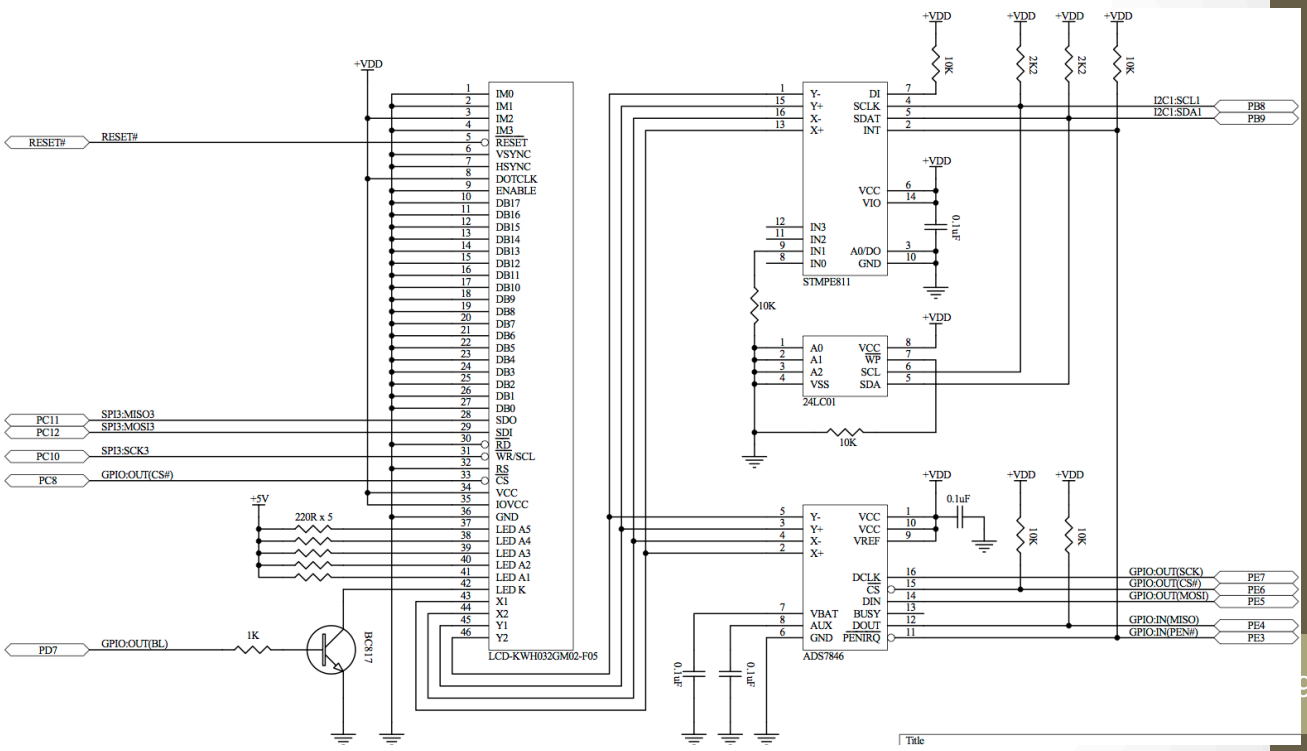
Figures are curtsy of 3M

LCD Display

STM32 Board LCD Display

- TFT LCD Model KWH032GM02-F05
- http://www.displayfuture.com/Display/datasheet/TFT/KWH032GM02-F05.pdf
- 3.2 Inches
- 320 x 240 Pixels
- SPI mode

Board Schematic



Pin Assignment

การเชื่อมต่อ TFT LCD รุ่น KWH032GM02-F05

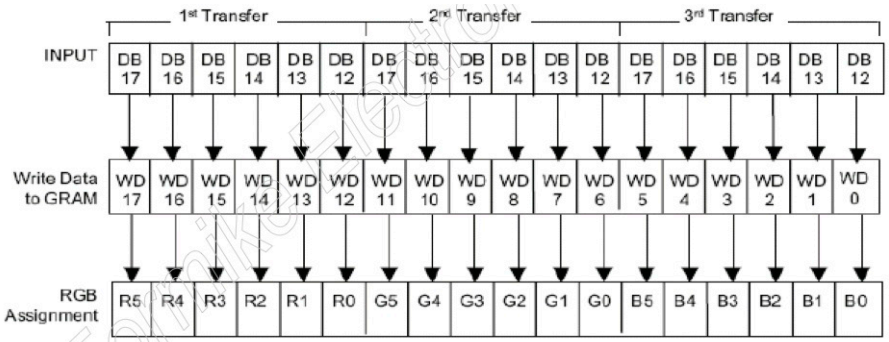
ส่วนของ TFT LCD รุ่น KWH032GM02-F05 จะใช้การเชื่อมต่อแบบ SPI Mode โดยจะใช้ SPI3 ของ MCU ในการติดต่อ โดยจะใช้สัญญาณการเชื่อมต่อดังนี้

- CS GLCD จะใช้ PC8 ในหน้าที่ GPIO Output
- SCL GLCD จะใช้ PC10 ในหน้าที่ SCK3 ของ SPI3
- SDO GLCD จะใช้ PC11 ในหน้าที่ MISO3 ของ SPI3
- SDI GLCD จะใช้ PC12 ในหน้าที่ MOSI3 ของ SPI3
- BL GLCD จะใช้ PD7 ในหน้าที่ GPIO Output

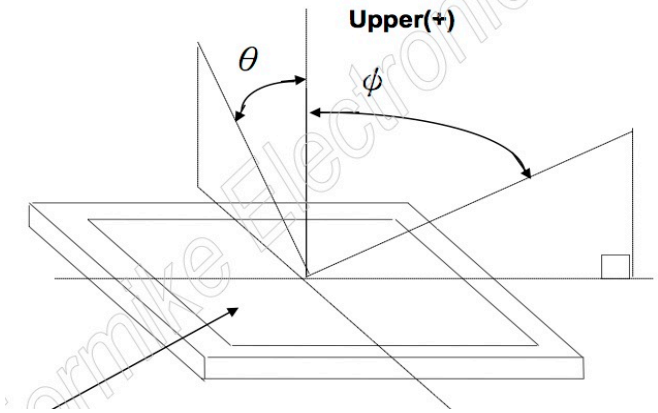
```
#define LCD_CS_PIN            GPIO_Pin_8              // PC8 = CS# GLCD
#define LCD_CS_GPIO_PORT      GPIOC
#define LCD_CS_GPIO_CLK       RCC_APB2Periph_GPIOC
#define LCD_BL_PIN            GPIO_Pin_7              // PD7 = BL GLCD
#define LCD_BL_GPIO_PORT      GPIOD
#define LCD_BL_GPIO_CLK       RCC_APB2Periph_GPIOD

#define LCD_SPI_SCK_PIN       GPIO_Pin_10              //SPI3
#define LCD_SPI_MISO_PIN      GPIO_Pin_11
#define LCD_SPI_MOSI_PIN      GPIO_Pin_12
#define LCD_SPI_GPIO_PORT     GPIOC
#define LCD_SPI_GPIO_CLK      RCC_APB2Periph_GPIOC
#define LCD_SPI               SPI3
#define LCD_SPI_CLK           RCC_APB1Periph_SPI3
```


Color Assignment



- Up to 256*1024 colors per pixel
- Red = R5..R0 (6bits), Green = G5..G0, Blue = B5..B0
- 2^18 = 2^8 * 2^10 = 256*1024

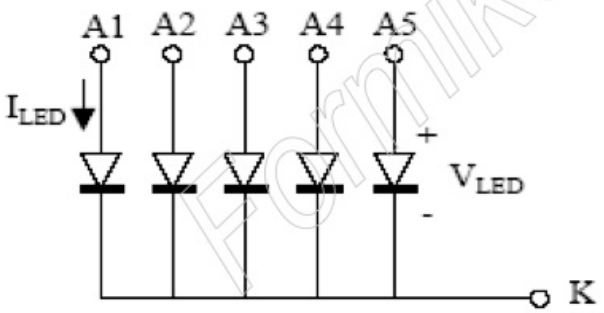


Optical Specification

Item		Symbol	Condition	Min	Typ	Max	Unit	Remark
Brightness		L	White pattern	160	200	-	cd/m^2	
Response time		T_R+T_F	$\Theta=\phi=0$	-	25	-	ms	Note 2
Contrast ratio		CR	At the center point of A.A.	300	350	-		
Color Chromaticity	White	W_x	$\Theta=\phi=0$	0.273	0.313	0.353		Note 2
		W_y		0.289	0.329	0.369		
Viewing Angle	Horizontal		CR≧10	-	110	-	Degree	
	Vertical			-	120	-		

Fernike Electronics Co.,Ltd.

Back-Light Characteristic



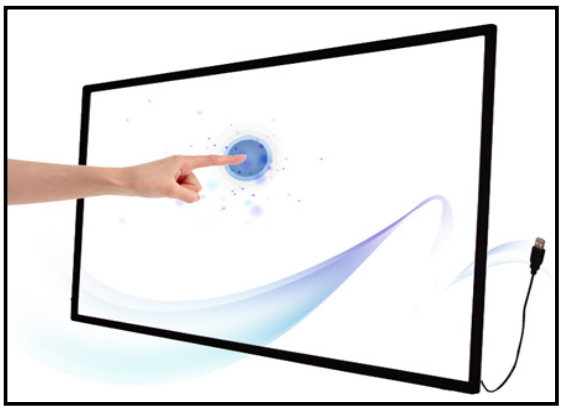
Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Power Consumption	P	288	330	450	mW	Backlight unit
LED Current	I_{LED}	18	20	25	mA	Single LED
LED Voltage	V_{LED}	3.2	3.3	3.6	V	Single LED

Fernike Electronics Co.,Ltd.

Touch Screen Sensor

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

What is a touch screen?



- An electronic visual display that locates the coordinates of a users touch within display area
- Works independently of what is being displayed on screen

How Does a Touchscreen Work?

A basic touchscreen has three main components:

1 Touch sensor;
2 Controller;
3 Software driver.

The touchscreen is an input device, so it needs to be combined with a display and a PC or other device to make a complete touch input system.

Touch Sensor

- A touch screen sensor is a clear glass panel with a touch responsive surface. The touch sensor/panel is placed over a display screen so that the responsive area of the panel covers the viewable area of the video screen.
- There are several different touch sensor technologies on the market today, each using a different method to detect touch input.
- The sensor generally has an electrical current or signal going through it and touching the screen can cause a voltage or signal change.
- This change is used to determine the location of the touch to the screen.

Controller

- The controller connects between the touch sensor and the PC. It takes information from the touch sensor and translates it into information that PC can understand.
- The controller determines what type of interface/connection you will need on the PC.
- Controllers are available that can connect to a Serial/COM port (PC) or to a USB port. Specialized controllers are also available that work with DVD players and other devices.

Software Driver

- The driver allows the touchscreen and computer to work together. It tells the computer's operating system how to interpret the touch event information that is sent from the controller.
- Most touch screen drivers today are a mouse-emulation type driver. This makes touching the screen as same as clicking your mouse at the same location on the screen.
- This allows the touchscreen to work with existing software and allows new applications to be developed without the need for touchscreen specific programming.

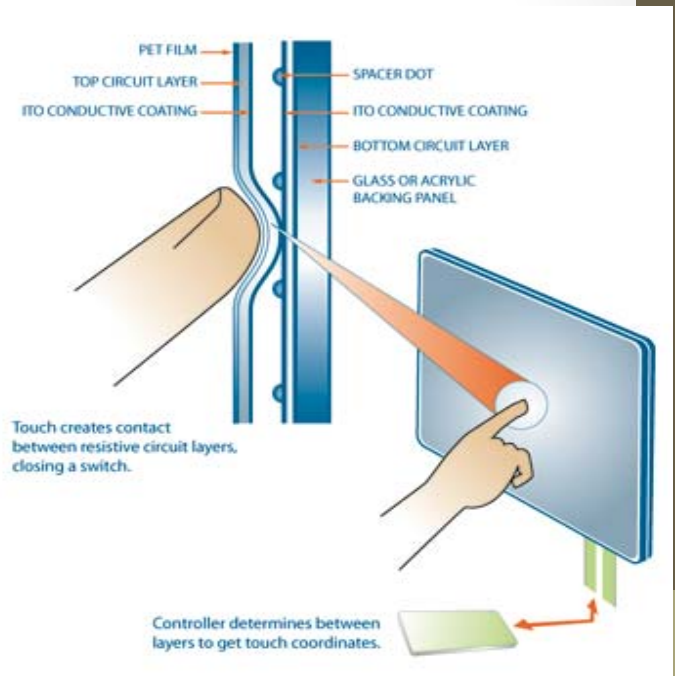
Touchscreen Technologies

- Resistive touchscreen
- Capacitive touchscreen
- Infrared touchscreen
- Surface acoustic wave (SAW) touchscreen

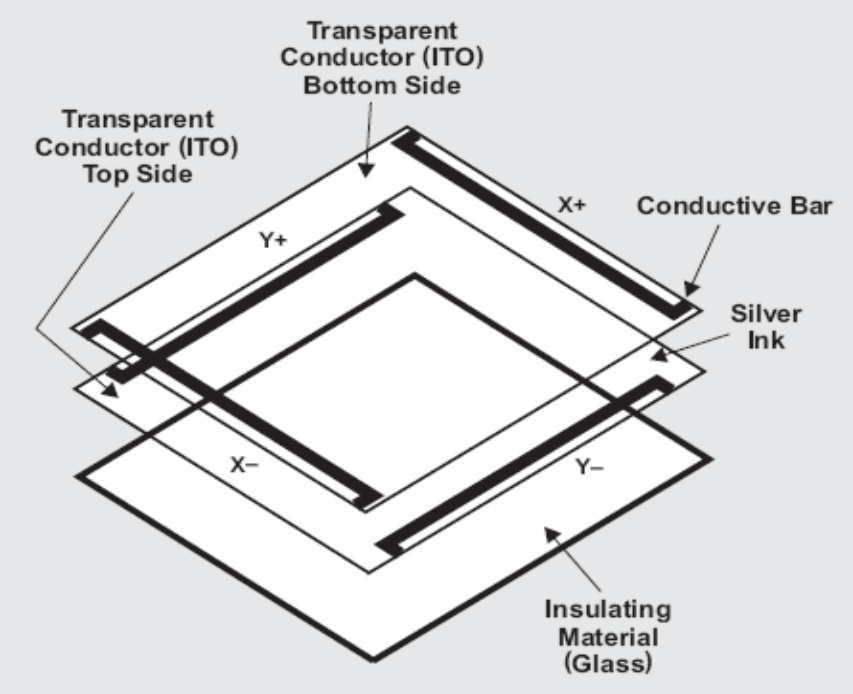
Resistive touchscreen

- Structure

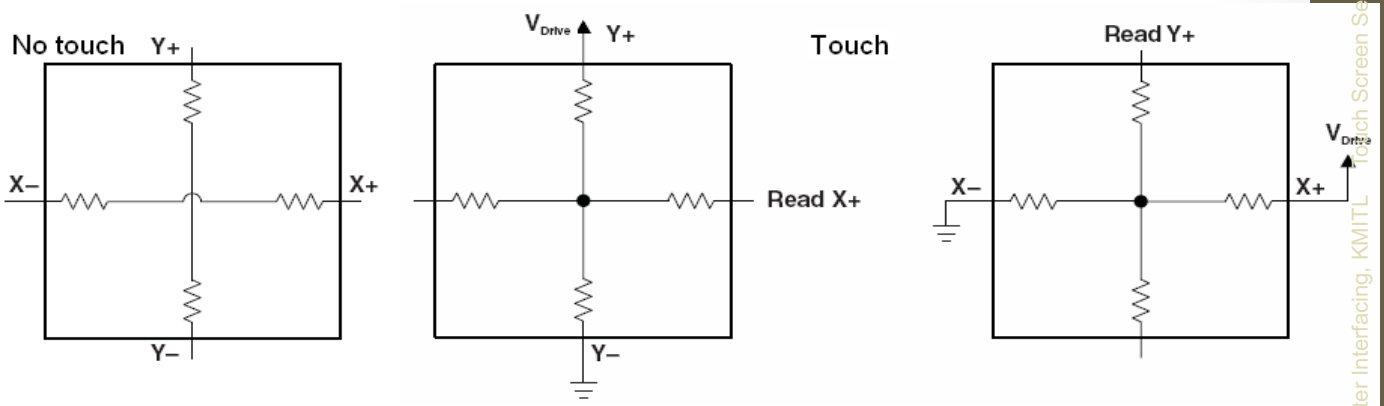
  Resistive touch screens consist of a glass or acrylic panel that is coated with electrically conductive and resistive layers made with indium tin oxide (ITO) .The thin layers are separated by invisible spacers.



4-wire resistive touchscreen



4-wire resistive touchscreen



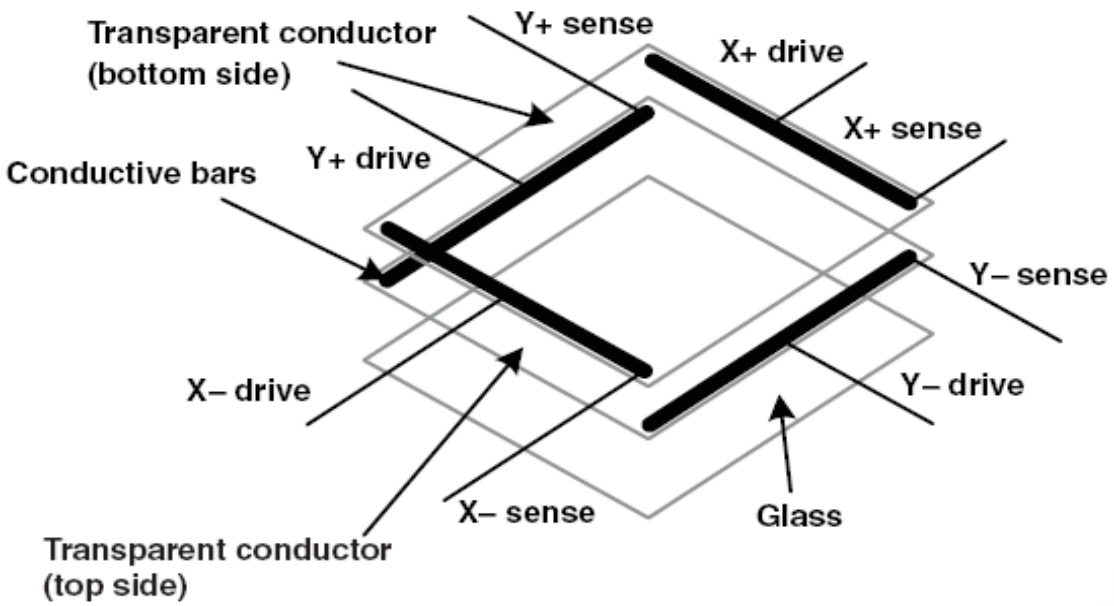
$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$

$$x = \frac{V_{Y+}}{V_{Drive}} \times width_{screen}$$

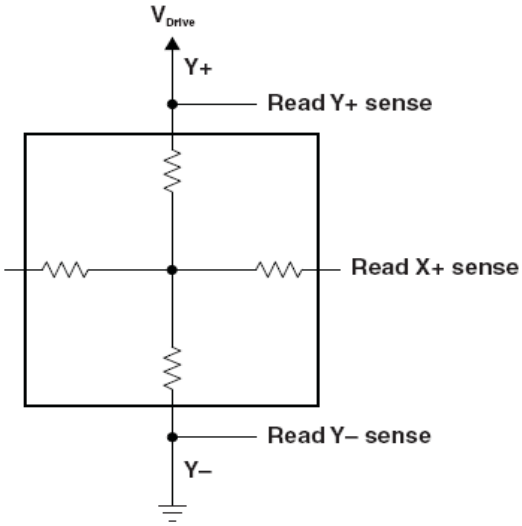
5-wire resistive touchscreen

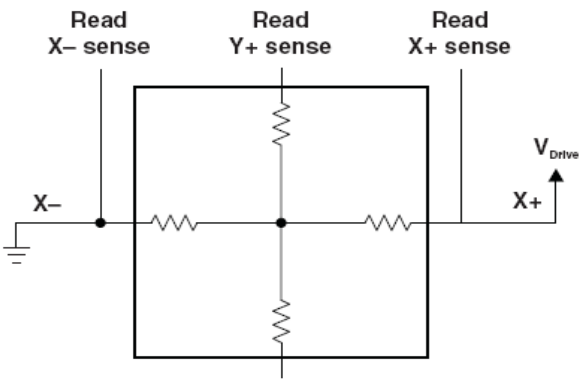
- The construction of the panels are similar with 4-wire technology, but for a 5-wire touch screen all four bus bars are connected to the lower, non-flexible layer of the screen.
- The flexible layer is always used as a sense layer to read the voltage connection point to the bottom layer.

8-wire resistive touchscreen



8-wire resistive touchscreen





$$y = \frac{\left(\frac{V_{X+} - V_{Ymin}}{V_{Ymax} - V_{Ymin}}\right)}{V_{Drive}} \times \text{height}_{screen}$$

$$x = \frac{\left(\frac{V_{Y+} - V_{Xmin}}{V_{Xmax} - V_{Xmin}}\right)}{V_{Drive}} \times \text{width}_{screen}$$

Resistive touchscreen

- Characters:

1. Cost effective solutions
2. Activated by a stylus, a finger or gloved hand
3. Not affected by dirt, dust, water, or light
4. 75%~85% clarify
5. resistive layers can be damaged by a very sharp object

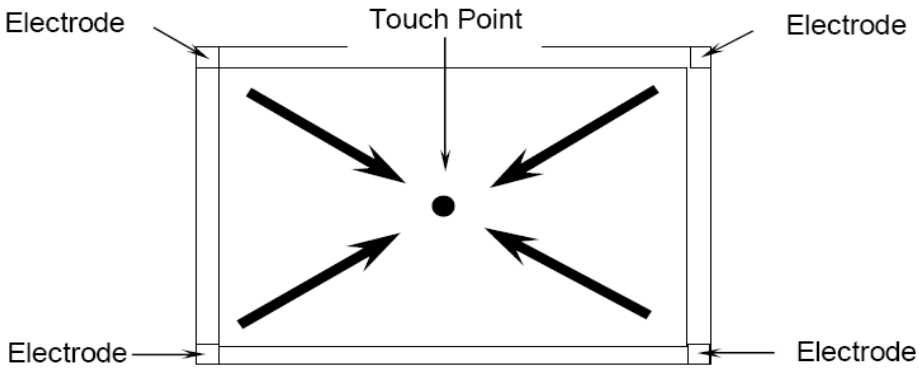
Capacitive Touch Technology

- Consists of:
  - Insulator (glass or Air)
  - Conductive coating (ITO)
- Two types:
  - Surface
  - Projected

Why Capacitive?

- Advantages:
- Durable surface material
- High endurance (~255 million touches)
- Very accurate
- Good optical quality
- Disadvantages:
- Triggered only by bare finger or active stylus

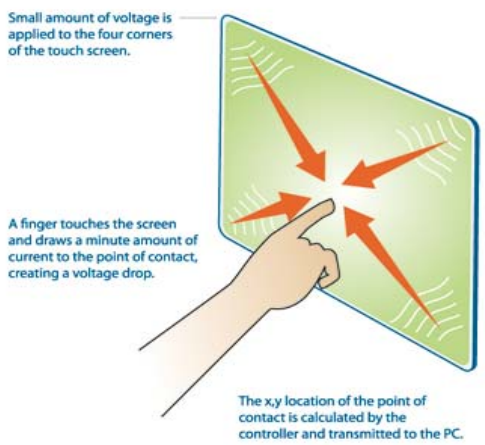
Surface-capacitive touchscreen

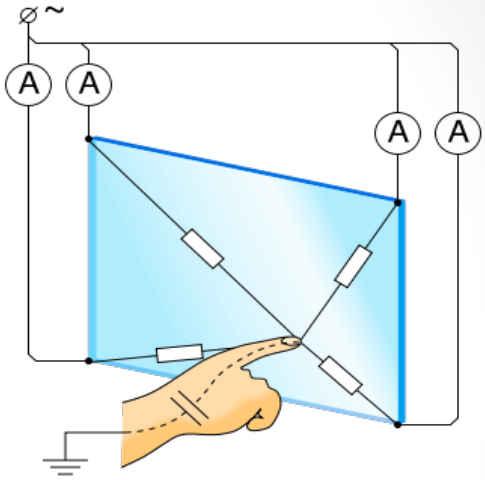


- Structure

  Surface capacitive technology consists of a uniform conductive coating on a glass panel.
- Electrodes around the panel' s edge evenly distribute a low voltage across the conductive layer, creating a uniform electric field.

Surface-capacitive touchscreen





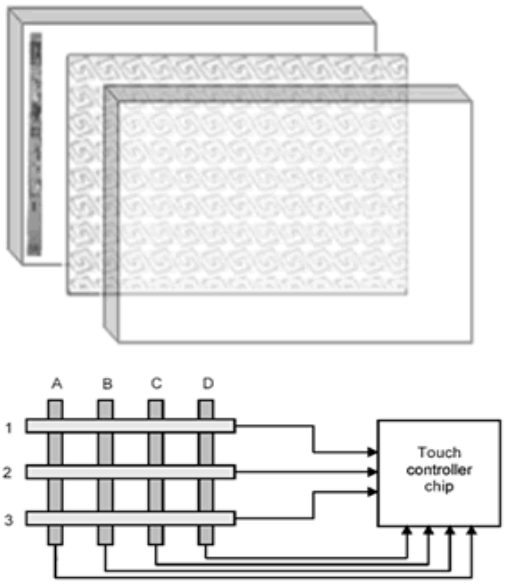
- Working principle
- A human body is an electric conductor, so when you touch the screen with a finger, a slight amount of current is drawn, creating a voltage drop.
- The current respectively drifts to the electrodes on the four corners.
- Theoretically, the amount of current that drifts through the four electrodes should be proportional to the distance from the touch point to the four corners.
- The controller precisely calculates the proportion of the current passed through the four electrodes and figures out the X/Y coordinate of a touch point.

Projected-capacitive touchscreen

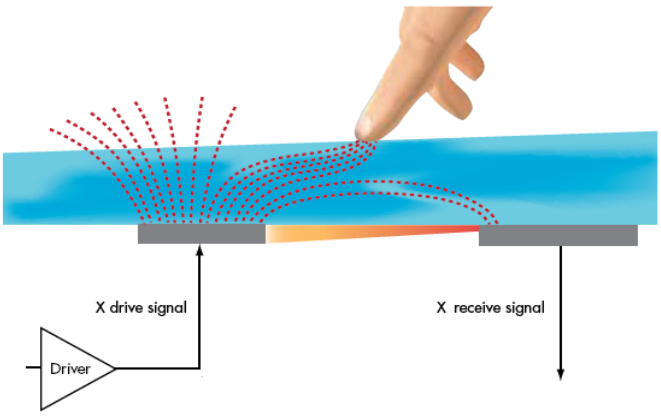
- Structure

  Projected capacitive touchscreens have front and back protective glass providing optical and strength enhancement options.

  Its middle layer consists of a laminated sensor grid of micro-fine wires, and optical enhancement options.

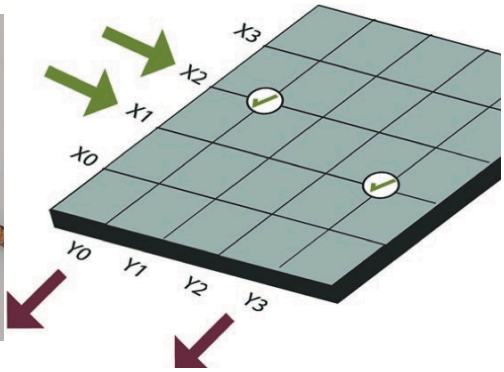


Projected-capacitive touchscreen



- During a touch, capacitance forms between the finger and the sensor grid.
- The embedded serial controller in the touchscreen calculates touch location coordinates and transmits them to the computer for processing.

Projected-capacitive Multi-touch

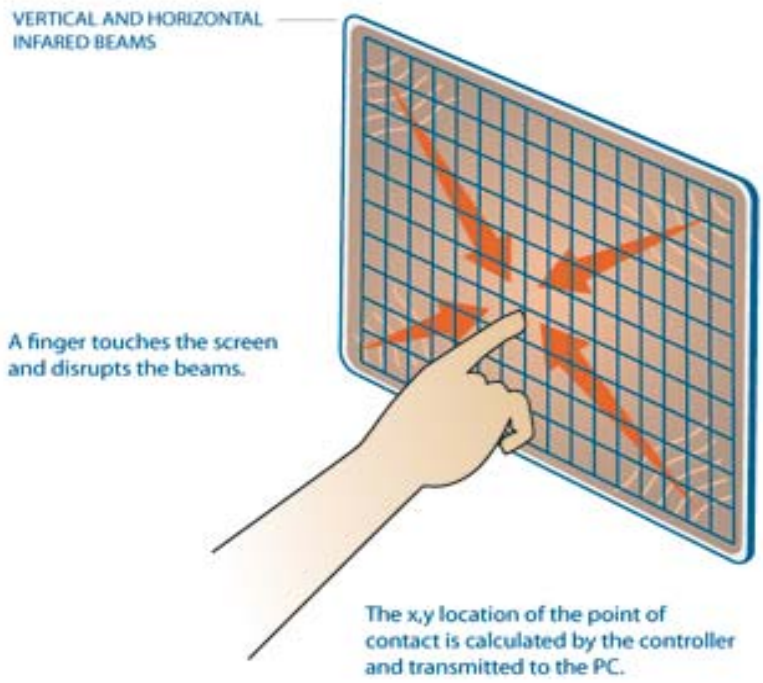


Surface vs. Projected

- Limited resolution
- Single touch
- Operation with direct contact

- High resolution
- Multi touch
- Operation with indirect contact

Infrared touchscreen



Infrared touchscreen

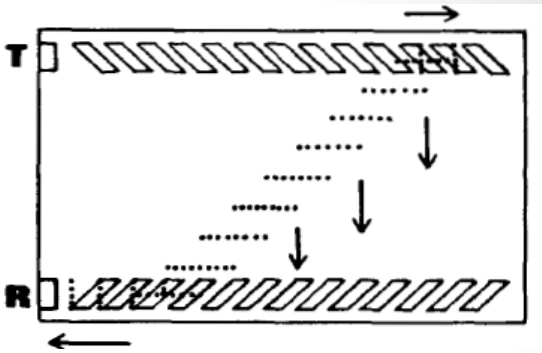
- Infrared (IR) technology relies on the interruption of an IR light grid in front of the display screen. The touch frame contains a row of IR-light emitting diode (LEDs) and photo transistors, each mounted on two opposite sides to create a grid of invisible infrared light.
  The IR controller sequentially pulses the LEDs to create a grid of IR light beams. When a stylus, such as a finger, enters the grid, it obstructs the beams. One or more photo transistors from each axis detect the absence of light and transmit signals that identifies the x and y coordinates.

Infrared touchscreen

- Characters:

1. Clear as glass, improves reading ability
2. Most durable surface

SAW touchscreen



- Principle

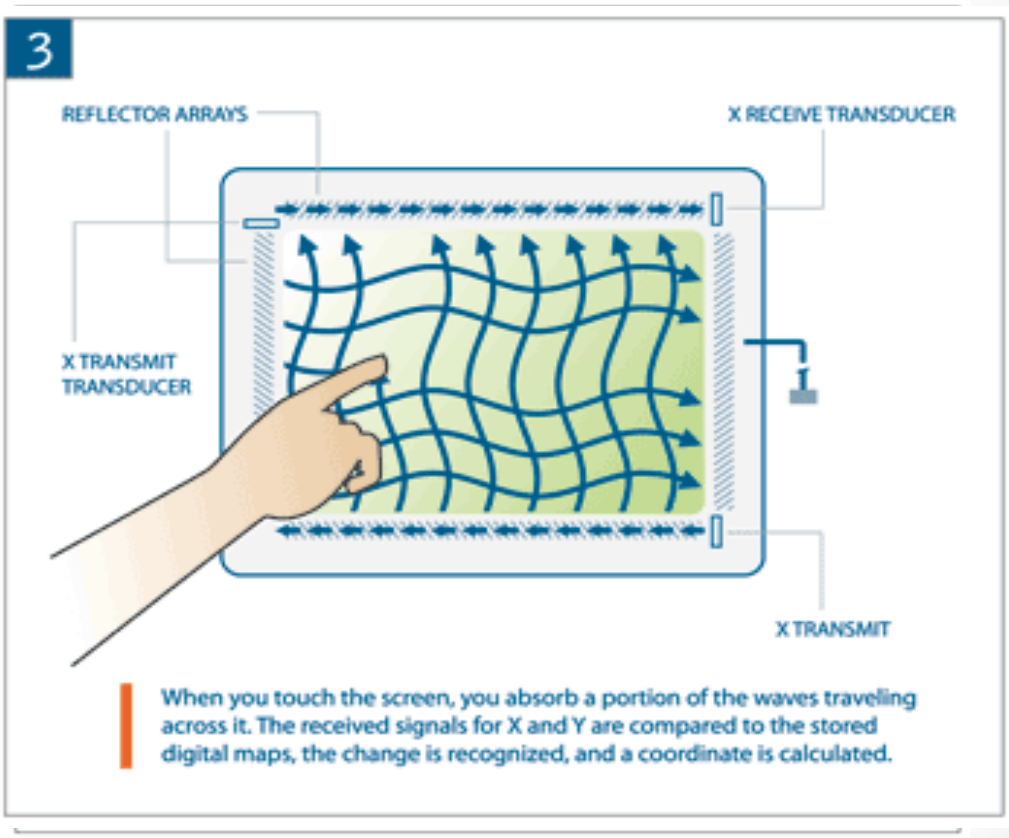
  Surface waves are readily absorbed when a soft object such as a fingertip touches the substrate.

  SAW Touch Screen use pure glass with transmitting and receiving piezoelectric transducers for both the X and Y axes.

  The touch screen controller sends an electrical signal to the transmitting transducer, which converts the signal into ultrasonic waves within the glass.

  When you touch the screen, you absorb a portion of the wave traveling across it. The received signal is then compared to the stored digital map, the change recognized, and a coordinate calculate.

SAW touchscreen



SAW touchscreen

- Characters:

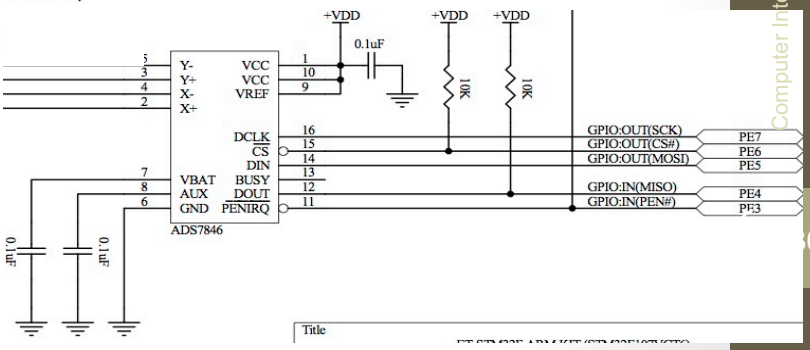
1. Durable glass construction
2. High optical clarity
3. Activated by a finger, gloved hand or soft tip
4. Not completely sealable, can be affected by large amounts of dirt, dust, and / or water in the environment

Technology	Capacitive	SAW	Infrared	Resistive
Transparence	Very good >92%	Very good >92%	Very good >92%	75%~85%
Resolution	Good	Good	Limited due to spacing of IR sensors	good
Surface Contaminants /durability	Resistant to moisture and other surface contaminants	Adversely affected by moisture or Surface contaminants	Potential for False activation or dead zones From Surface Contaminants	Unaffected by Surface contaminants. Polyester top sheet is easily scratched
Sensor substrate	Glass with ITO coating	Glass with ITO coating	Any substrate	Polyester top sheet, glass substrate with ITO coating
Display size	8.4"-21"	10.4"-30"	10.4"-60"	up to 19"
Touch method	Human touch	finger, gloved hand or soft tip	Can use any pointing device	Can use any pointing device

ETT STM32 Pin Assignment

```
#define TCS_GPIO_CLK    RCC_APB2Periph_GPIOE
#define TCS_GPIO_PORT GPIOE
#define TCS_PEN_PIN    GPIO_Pin_3       // PE3 = PEN# Touch Sensor
#define TCS_MISO_PIN   GPIO_Pin_4       // PE4 = MISO Touch Sensor
#define TCS_MOSI_PIN   GPIO_Pin_5       // PE5 = MOSI Touch Sensor
#define TCS_CS_PIN     GPIO_Pin_6       // PE6 = CS# Touch Sensor
#define TCS_SCK_PIN    GPIO_Pin_7       // PE7 = SCK Touch Sensor
```

- DCLK ADS7846 จะใช้ PE7 ในหน้าที่ GPIO Output (SPI:SCK)
- CS ADS7846 จะใช้ PE6 ในหน้าที่ GPIO Output (CS#)
- DOUT ADS7846 จะใช้ PE4 ในหน้าที่ GPIO Input (SPI:MISO)
- DIN ADS7846 จะใช้ PE5 ในหน้าที่ GPIO Output(SPI:MOSI)
- PENIRQ ADS7846 จะใช้ PE3 ในหน้าที่ GPIO Input



ETT STM32 Board
Resistive Touch Screen Sensor

- ประกอบด้วย
- แผ่นเซ็นเซอร์บนหน้าจอ LCD
- ชิพ Texas Instruments ADS 7845 และ Texas Instruments ADS 7846 สำหรับแปลงโวลเตจการสัมผัสไปเป็นตำแหน่ง X และ Y
- การเชื่อมต่อกับชิพ STM32 ผ่าน SPI

Our Touch Screen Controller

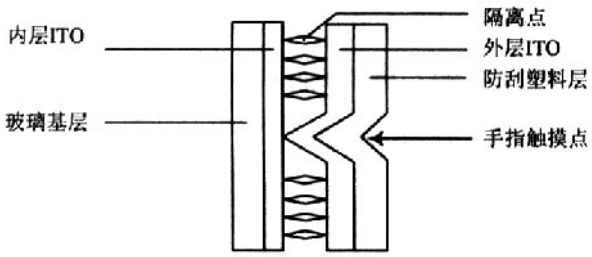


图 1 电阻式触摸屏结构

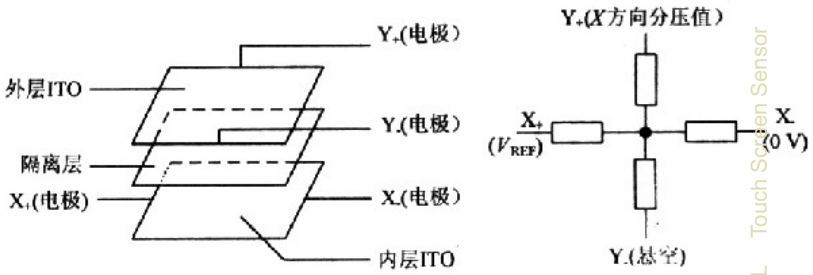
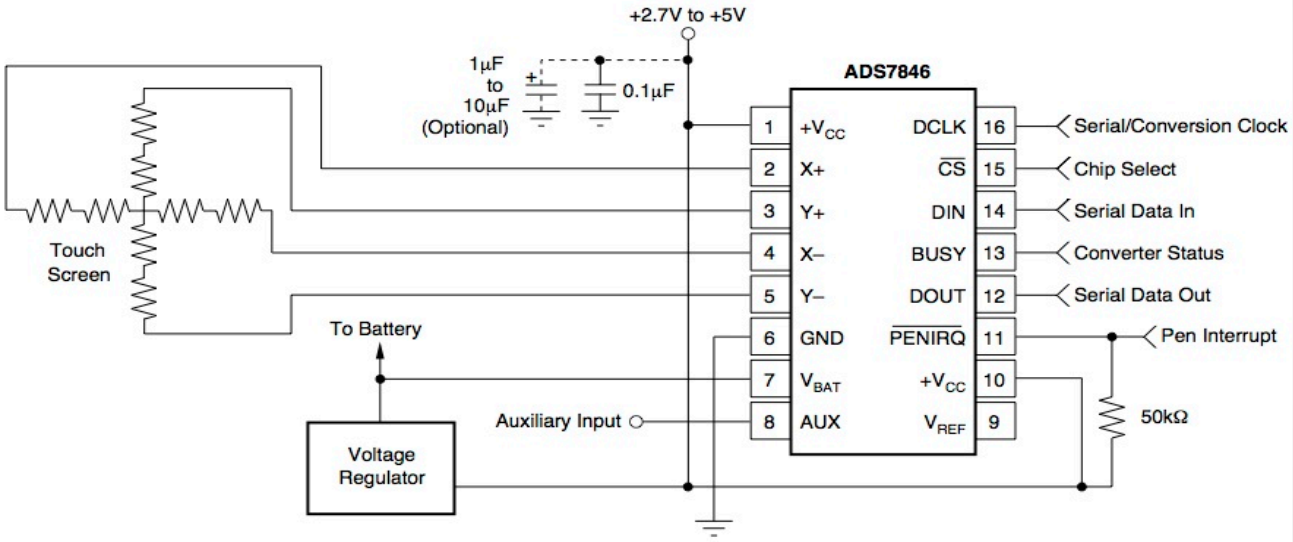


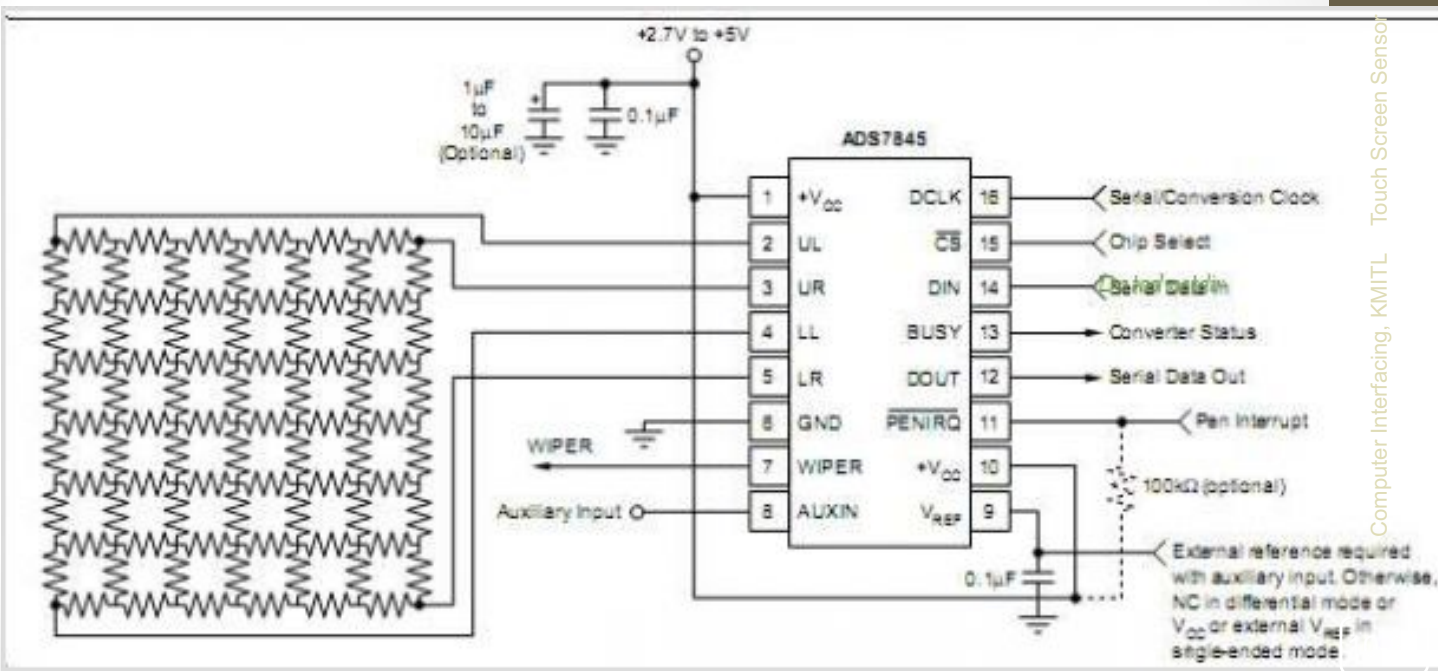
图 2 四线电阻触摸屏检测原理

- Texas Instruments ADS 7846
- Resistive Touch-Pressure Measurement
- SPI connection to Microcontroller

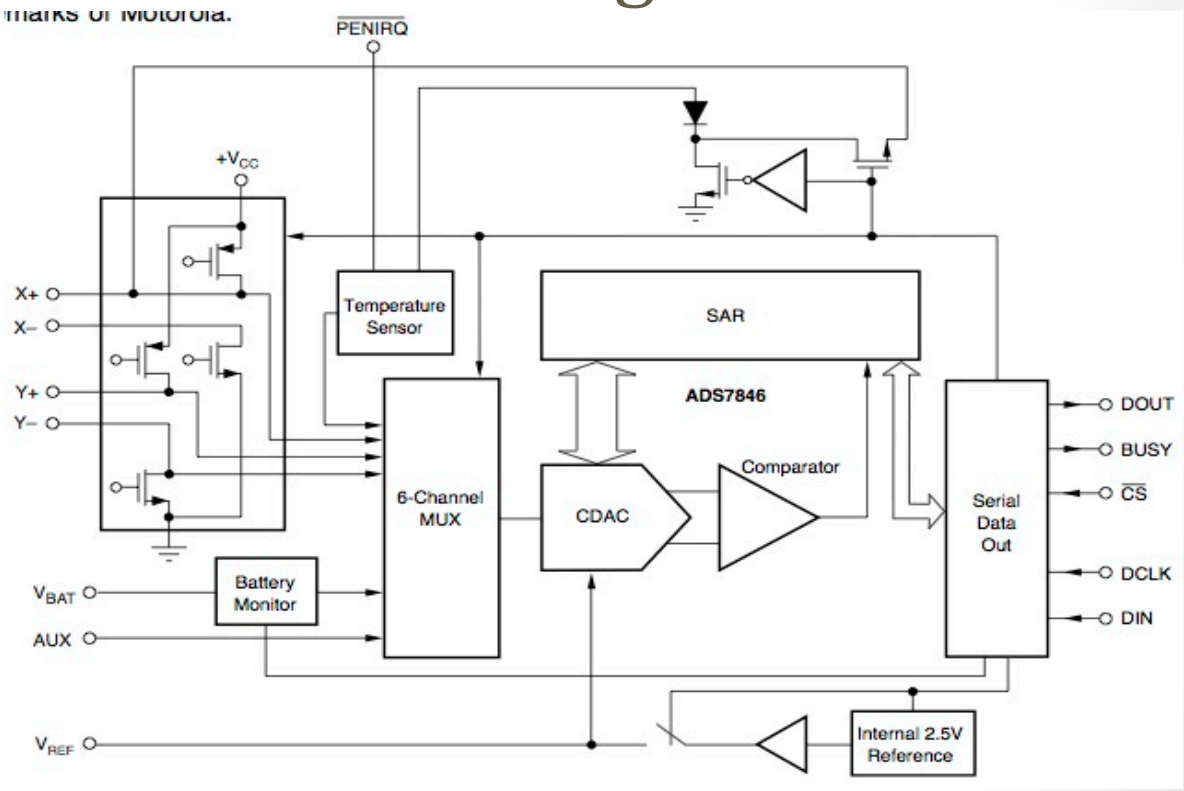
ADS7846 Circuit Diagram



ADS7845 Circuit Diagram



ADS7846 Block Diagram



Sensing Flow Chart

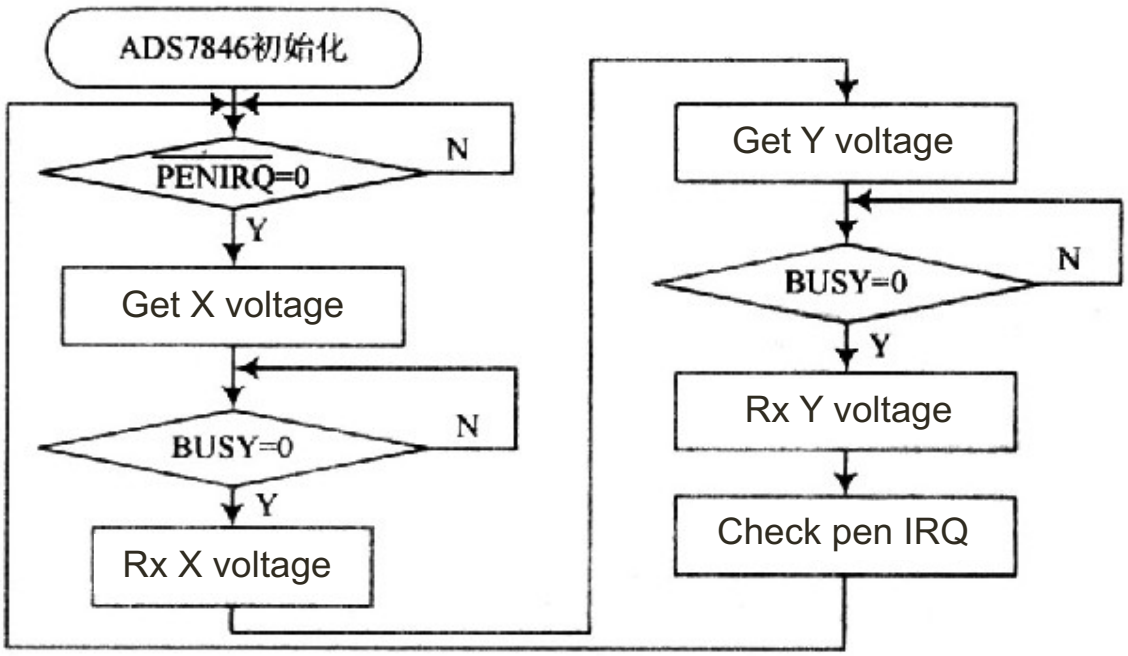
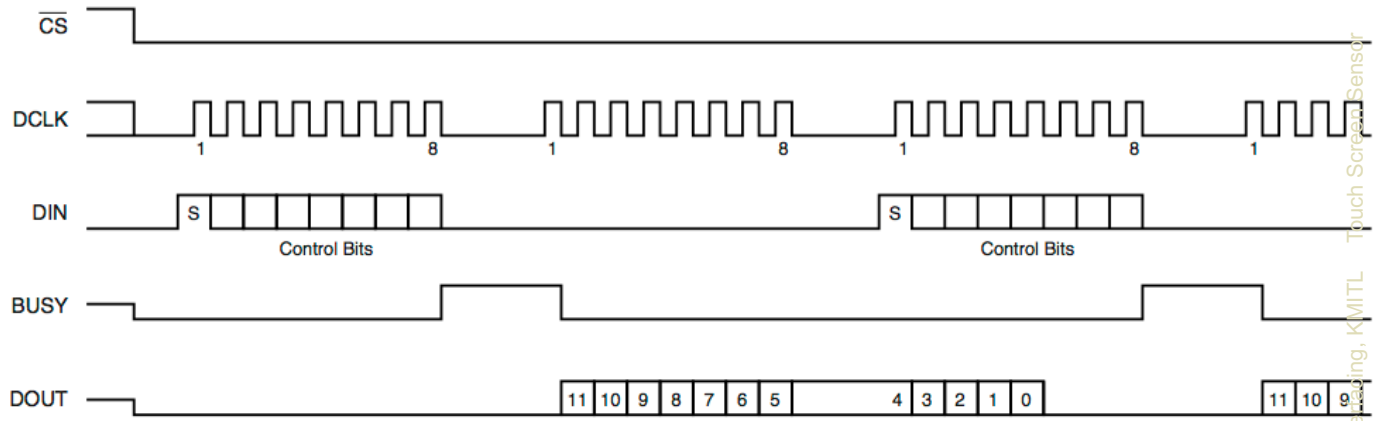
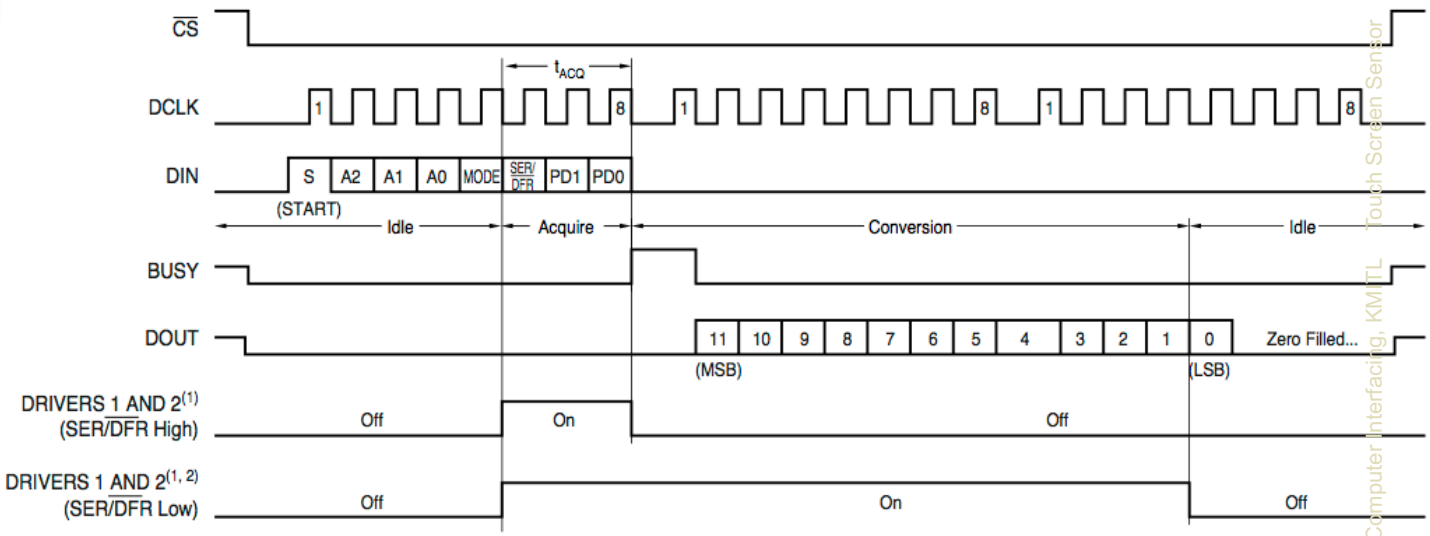


图 6 坐标读取流程图

ADS7846 Timing Diagram



ADS7846 Timing Diagram



- Mode=1, 12-bit resolution
- Mode=0, 8-bit resolution

Control Byte

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
S	A2	A1	A0	MODE	SER/$\overline{\text{DFR}}$	PD1	PD0

TABLE III. Order of the Control Bits in the Control Byte.

PD1	PD0	$\overline{\text{PENIRQ}}$	DESCRIPTION
0	0	Enabled	Power-Down Between Conversions. When each conversion is finished, the converter enters a low-power mode. At the start of the next conversion, the device instantly powers up to full power. There is no need for additional delays to assure full operation and the very first conversion is valid. The Y– switch is on when in power-down.
0	1	Disabled	Reference is off and ADC is on.
1	0	Enabled	Reference is on and ADC is off.
1	1	Disabled	Device is always powered. Reference is on and ADC is on.

TABLE V. Power-Down and Internal Reference Selection.

A2	A1	A0	V_{BAT}	AUX_{IN}	TEMP	Y–	X+	Y+	Y-POSITION	X-POSITION	Z_1-POSITION	Z_2-POSITION	X-DRIVERS	Y-DRIVERS
0	0	0			+IN (TEMP0)								Off	Off
0	0	1					+IN		Measure				Off	On
0	1	0	+IN										Off	Off
0	1	1					+IN				Measure		X–, On	Y+, On
1	0	0				+IN						Measure	X–, On	Y+, On
1	0	1						+IN		Measure			On	Off
1	1	0		+IN									Off	Off
1	1	1			+IN (TEMP1)								Off	Off

TABLE I. Input Configuration (DIN), Single-Ended Reference Mode (SER/$\overline{\text{DFR}}$ high).

A2	A1	A0	+REF	–REF	Y–	X+	Y+	Y-POSITION	X-POSITION	Z_1-POSITION	Z_2-POSITION	DRIVERS ON
0	0	1	Y+	Y–		+IN		Measure				Y+, Y–
0	1	1	Y+	X–		+IN				Measure		Y+, X–
1	0	0	Y+	X–	+IN						Measure	Y+, X–
1	0	1	X+	X–			+IN		Measure			X+, X–

TABLE II. Input Configuration (DIN), Differential Reference Mode (SER/$\overline{\text{DFR}}$ low).

Ethernet Basics

ผศ.ดร.สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

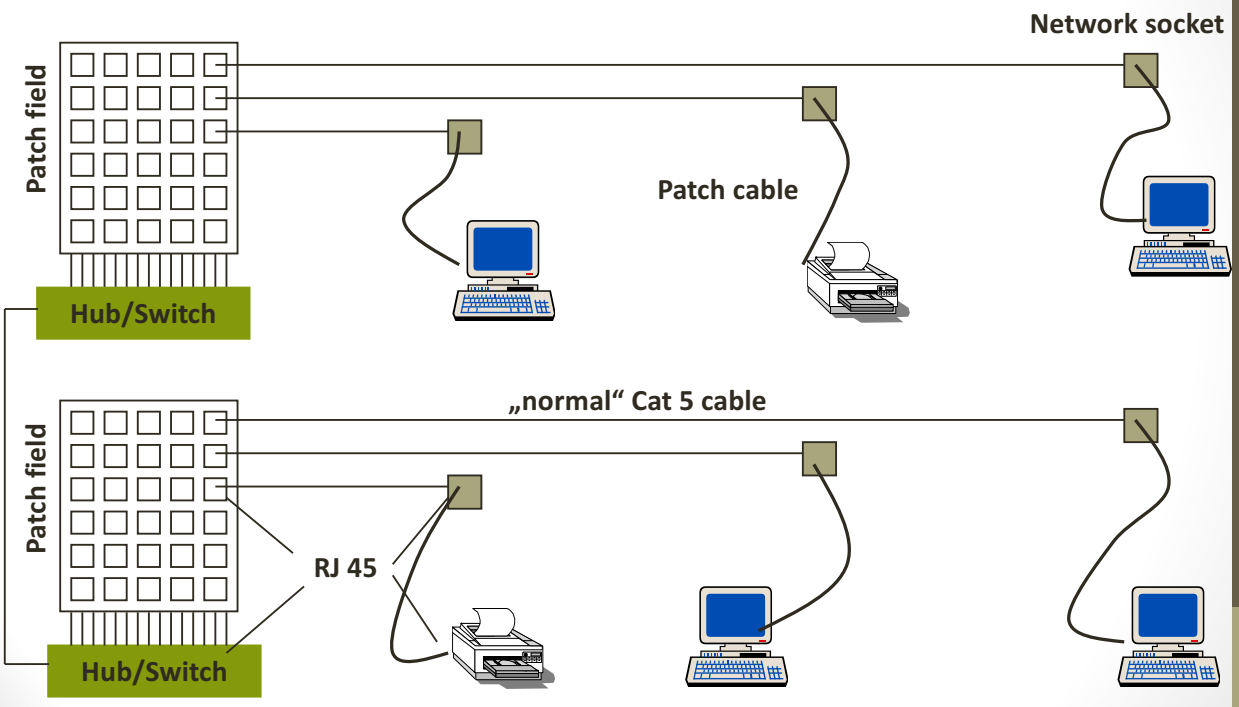
Topics

- History, Standards, Terminologies
- Transmission media
- Protocol: Access methods and Collision management
- ETT Board Ethernet Connection and Circuit Diagram
- STM32 Ethernet MAC
- Ethernet Phy: DP83848

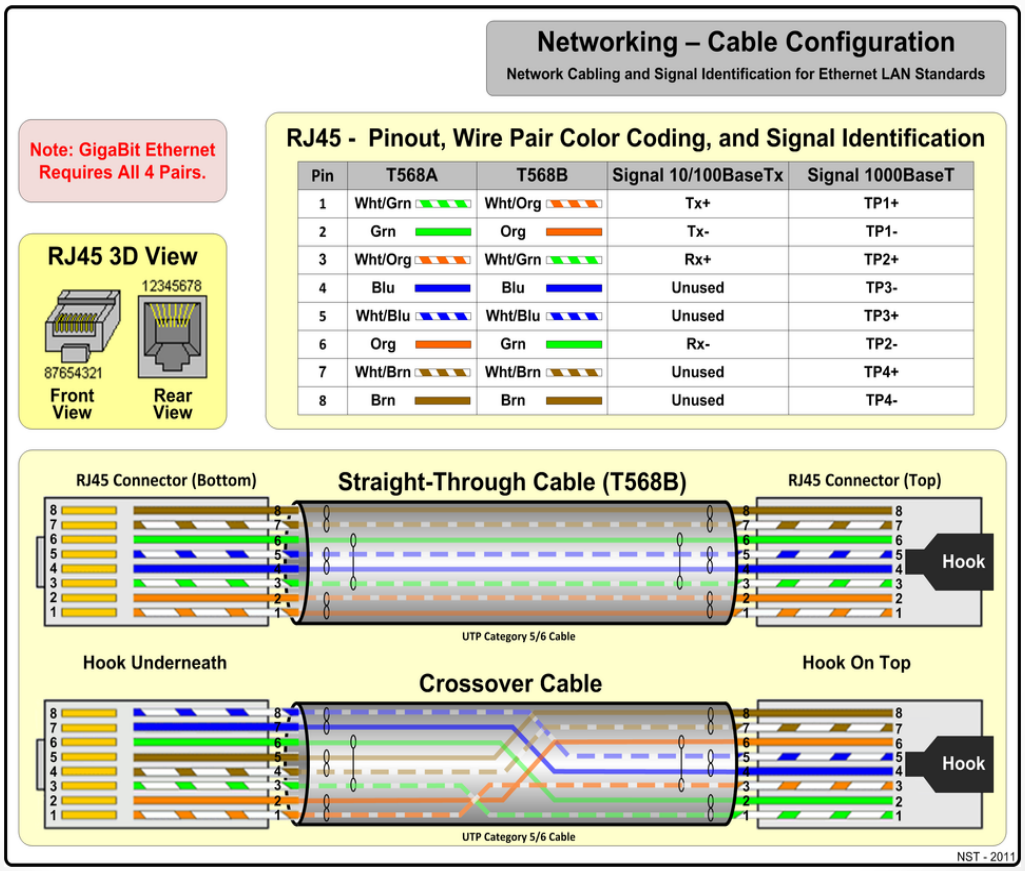
What is Ethernet ?

- Ethernet is a certain type of a local area network (LAN) which was developed in 1972 in the renowned PARC-research facility of Xerox in Palo Alto by Robert Metcalfe.
- In the meantime the companies Intel, DEC and Xerox have specified a common standard that has been established in the IEEE-standard 802.3.

Typical office wiring



Cabling: Pins, Names, RJ45



Naming of the cable types

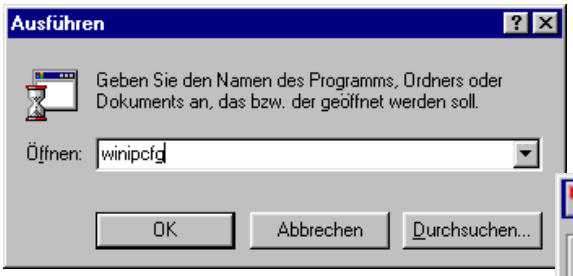
- Example: 10base5
  - 10 Transmission rate in Mbytes/s
  - base Base or Broadband
  - 5 Segment length in 100 meters
- UTP unshielded twisted pair
- STP shielded twisted pair
- S/STP screened shielded twisted pair

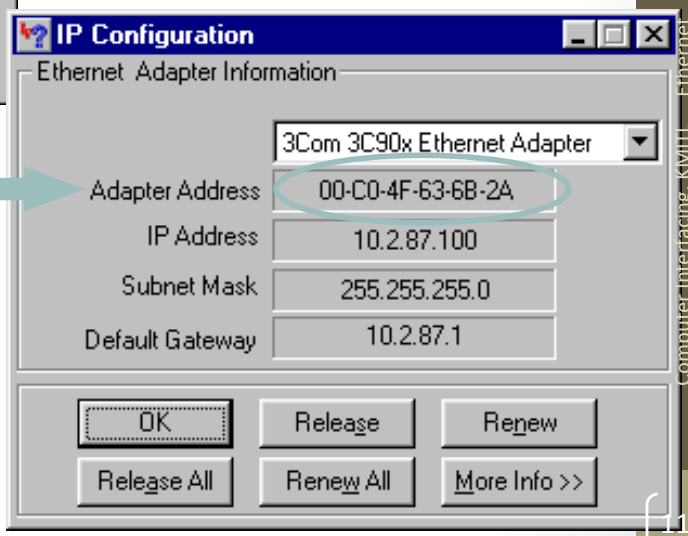
Extension

- The maximum extension depends on the medium and the transmission rate; here are some examples:

• 10base5	Segment: 500m Total: 2500m (with 4 repeaters)
• ***100baseTX UTP***	***Switch-Station: 100m***
• 100baseFX	Hub-Station: 400m 25km (with Mono mode fiber)
• 1000baseSX	Switch-Station: 550m

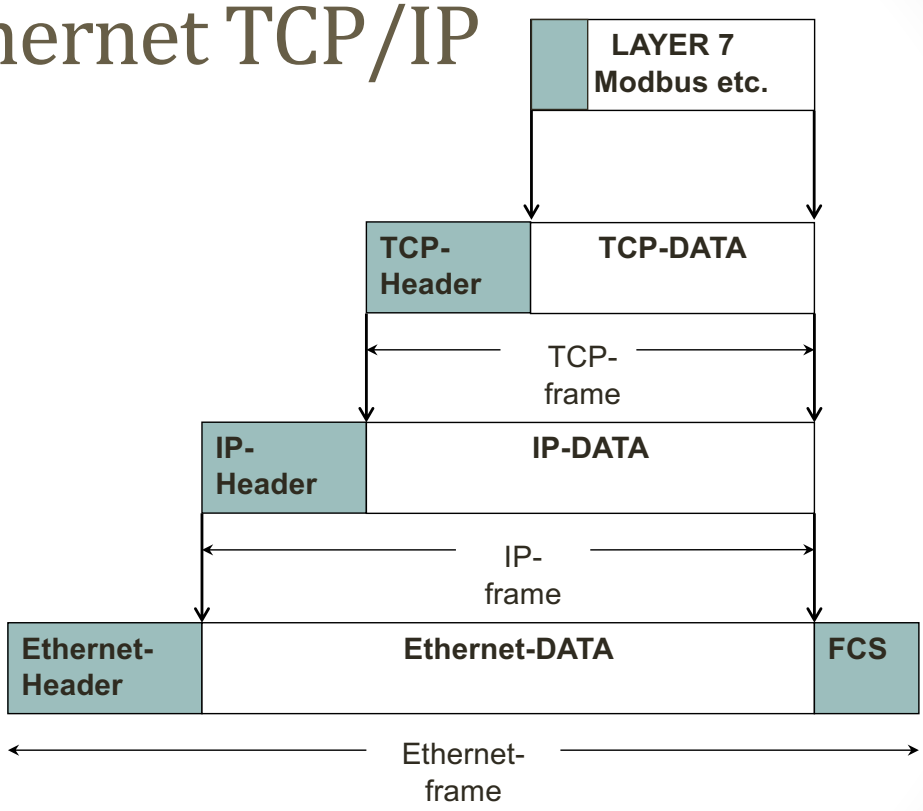
Ethernet Address





WINDOWS: ipconfig /all

Ethernet TCP/IP



Ethernet frame

Ethernet II DIX Frame:

Preamble	DA	SA	Type	Data	Pad	FCS
8	6	6	2	>=46		4

IEEE 802.3 Frame:

Preamble	SFD	DA	SA	LEN	Data	Pad	FCS
7	1	6	6	2	>=46		4

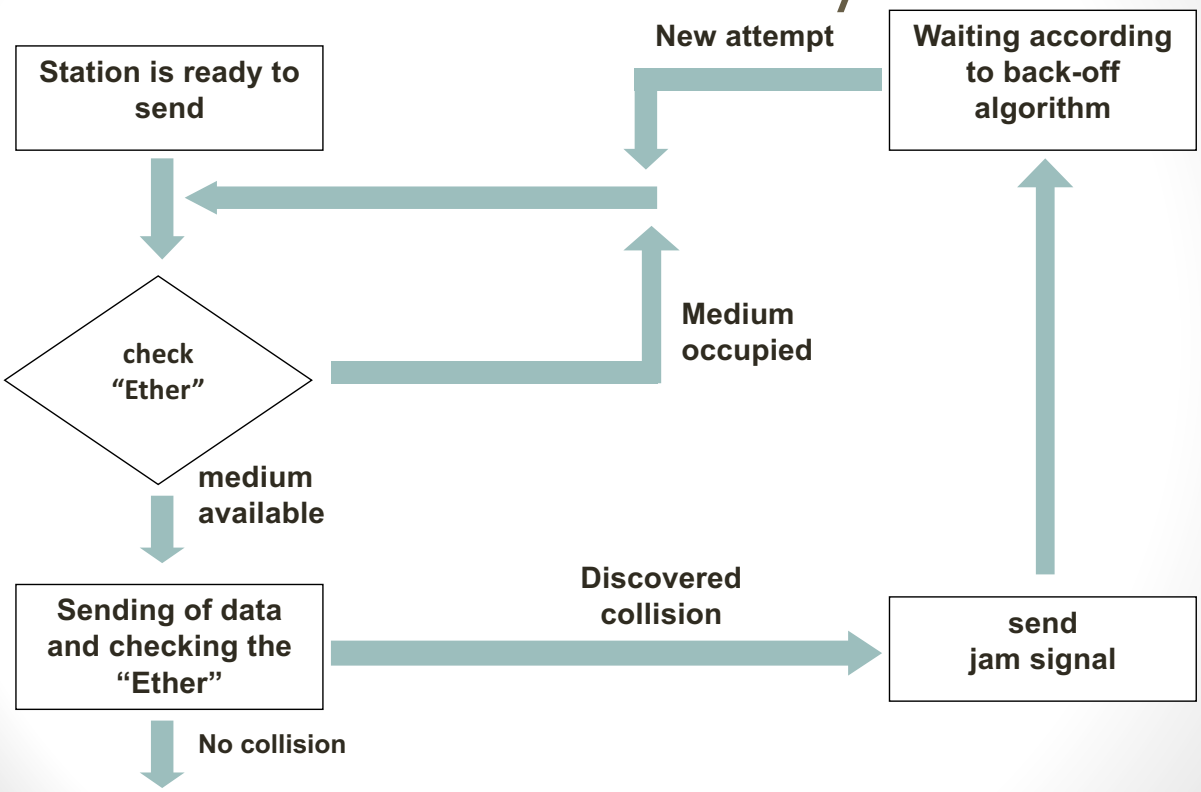
Ethernet frame

- **Preamble** Trailer consisting of the bit sequence "0101010101..." serving the bit synchronization of the receiver.
- **SFD** (Start Frame Delimiter) Start character consisting of the bit pattern "10101011" showing the recipient that the actual information will follow now.
- **DA** (Destination Address) Evaluated by the recipient's address filter; only data frames destined for this recipient will be passed on to the communication software.
- **SA** (Source Address) Sender's address
- **LEN** (Length) Indicates the length of the subsequent data field in Bytes according to IEEE 802.3.

Ethernet frame

- Data and Pad
  The data field may contain 46 to 1500 user data bytes. If user data length is less than 46 bytes, the Ethernet controller adds padding bytes, until the total amount (data + pad) is 46.
- FCS (Frame Check Sequence)
  A check character. It is obtained by taking the rest of the division operation from the formula representing the wide-spread cyclic- redundancy-check procedure.

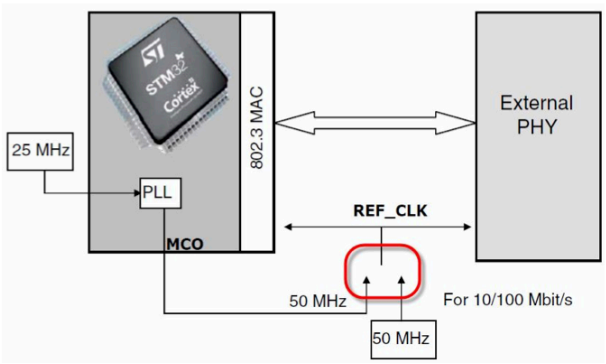
Access method: CSMA/CD

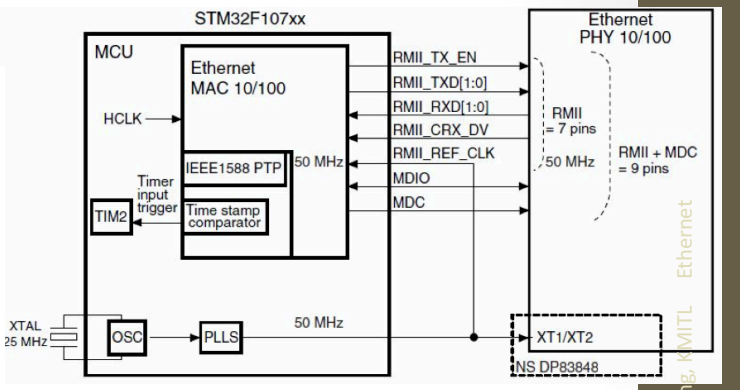


Back-Off Algorithm

- If a collision has occurred, the stations try to send again after a certain period of time.
- After the first collision there a two different back-off times available, from which one is chosen at random. Transmission probability is 50%
- After the second consecutive collision there are four different back-off times available, from which one is chosen at random.
- The transmission probability now is 75%

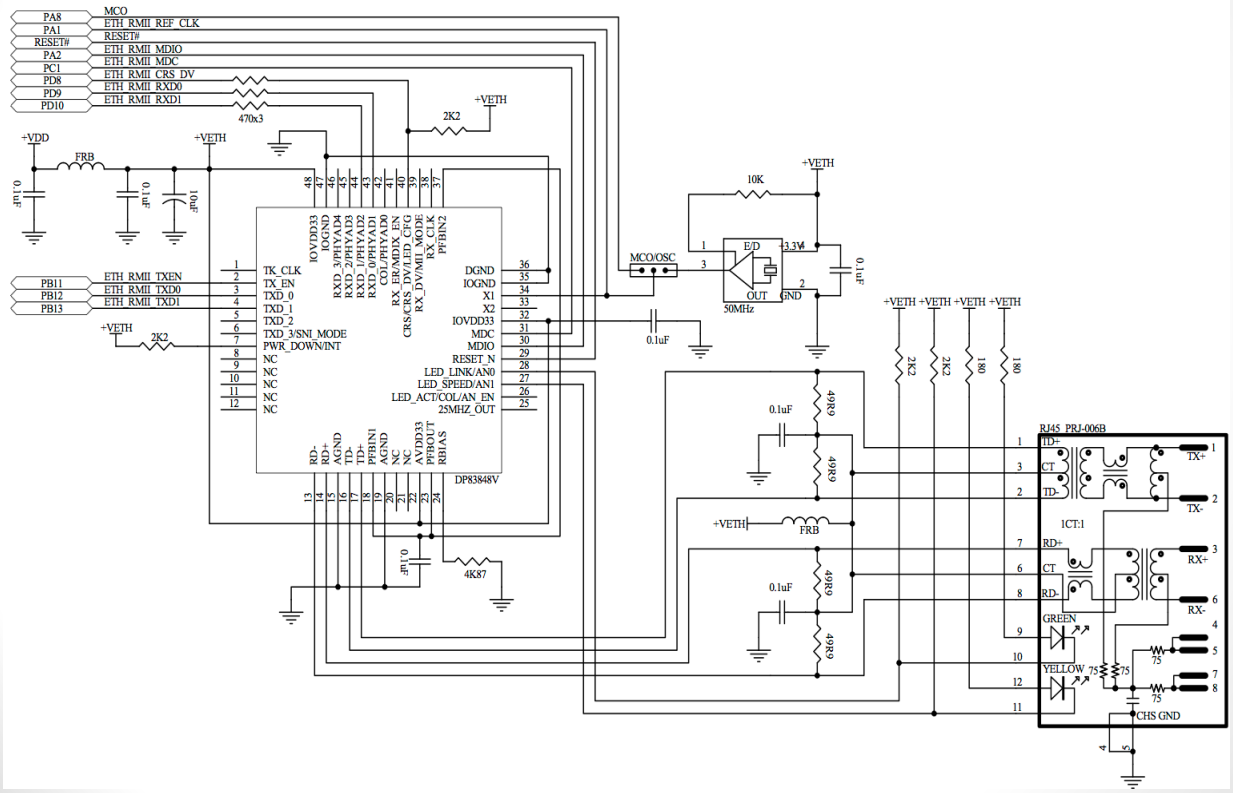
ETT Board: Ethernet (RMII)





- TX_EN: Transmit Enable)
- TXD: Transmit Data [1:0]
- RXD: Receive Data [1:0]
- CRX_DV (CRS_DV): Carrier Sense/Receive Data Valid
- REF_CLK: Reference Clock
- RXER: Receiver Error
- MDIO: Management Data Input/Output, MDC: Management Data CLK

ETT Board Circuit Diagram



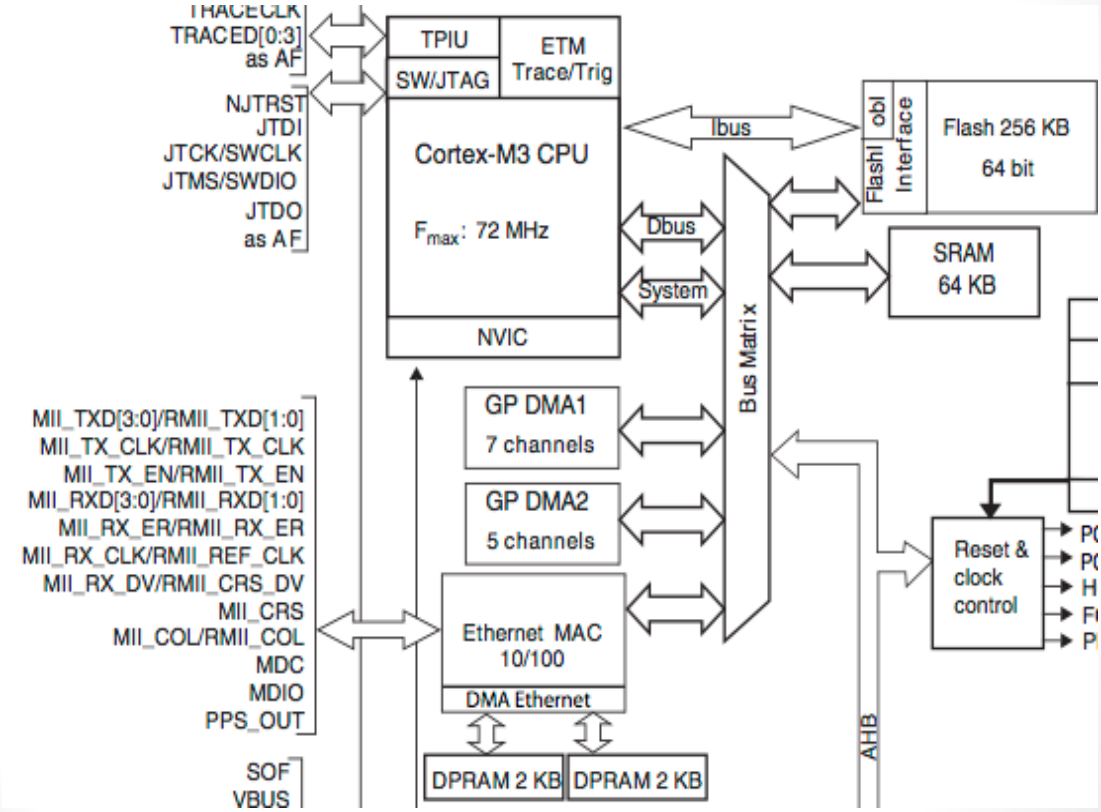
Pin Assignment & Initialization

```
/* Configure PA2 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;        //PA2 = ETH_RMII_MDIO
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Configure PC1 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;        //PC1 = ETH_RMII_MDC
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* Configure PB11, PB12 and PB13 as alternate function push-pull */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 |      //PB11 = ETH_RMII_TXEN
                              GPIO_Pin_12 |      //PB12 = ETH_RMII_TXD0
                              GPIO_Pin_13;       //PB13 = ETH_RMII_TXD1
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

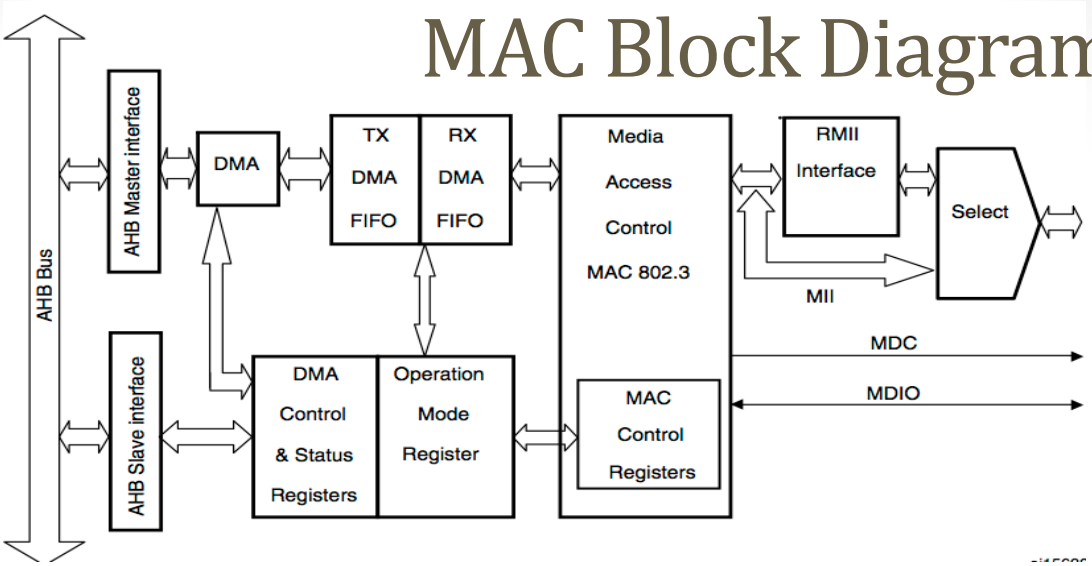

STM32 Ethernet MAC



STM32 Ethernet MAC

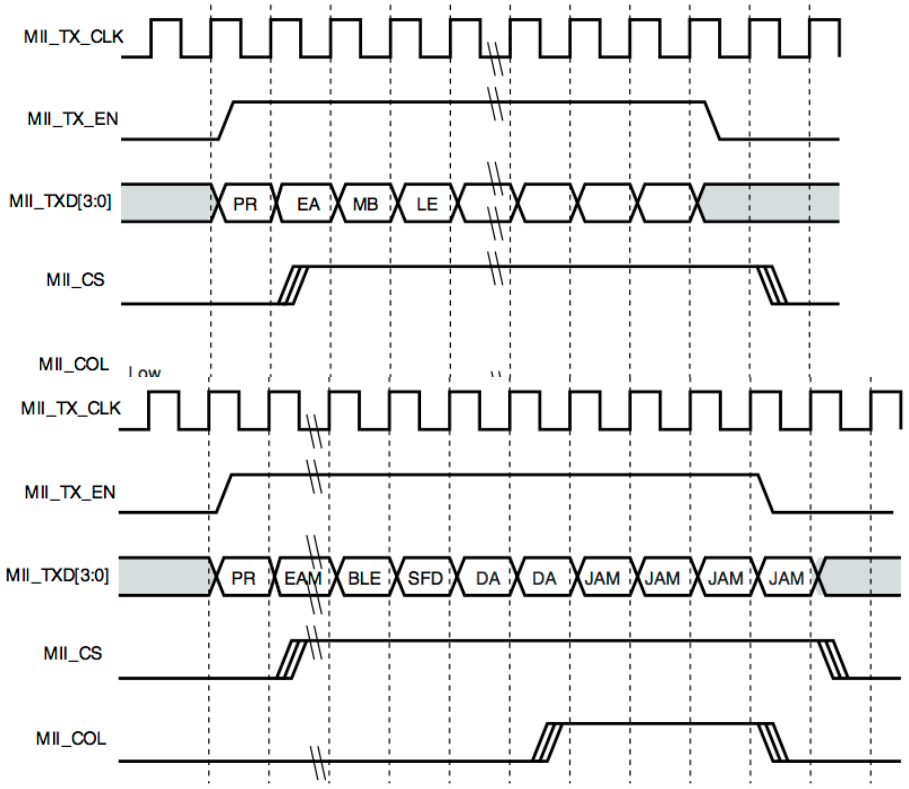
- Supports 10/100 Mbit/s data transfer rates with external PHY interfaces
- IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
- Supports IEEE 802.3x flow control for full-duplex operation
- Two sets of FIFOs: a 2-KB Transmit FIFO with programmable threshold capability, and a 2-KB Receive FIFO with a configurable threshold (default of 64 bytes)

MAC Block Diagram

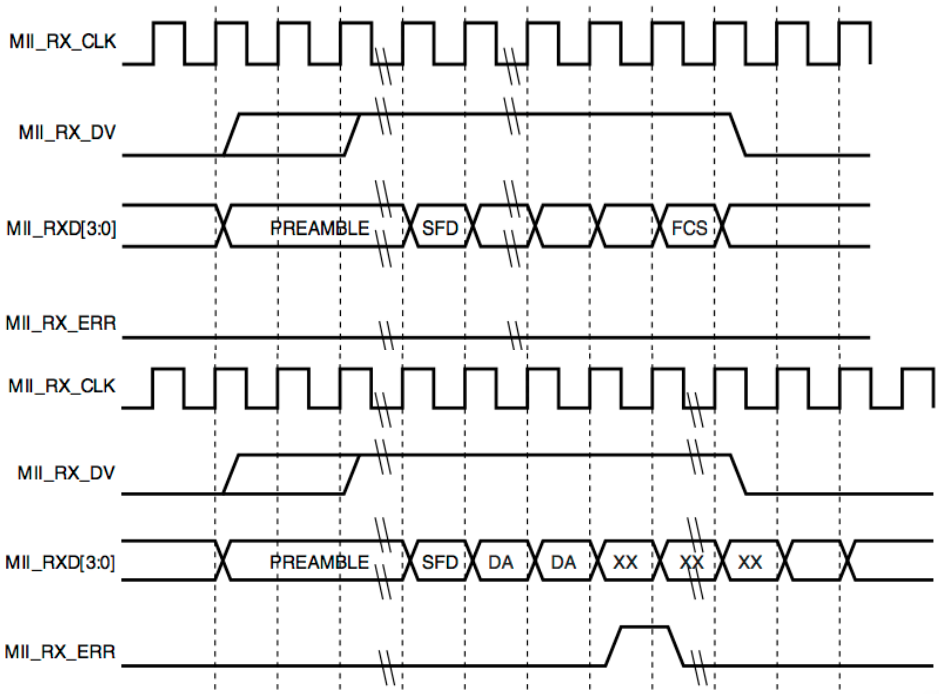


- DMA FIFO = DPRAM 2KB: TX, RX
- DMA = Direct Memory Access
- AHB = Advanced High-performance Bus (AHB)

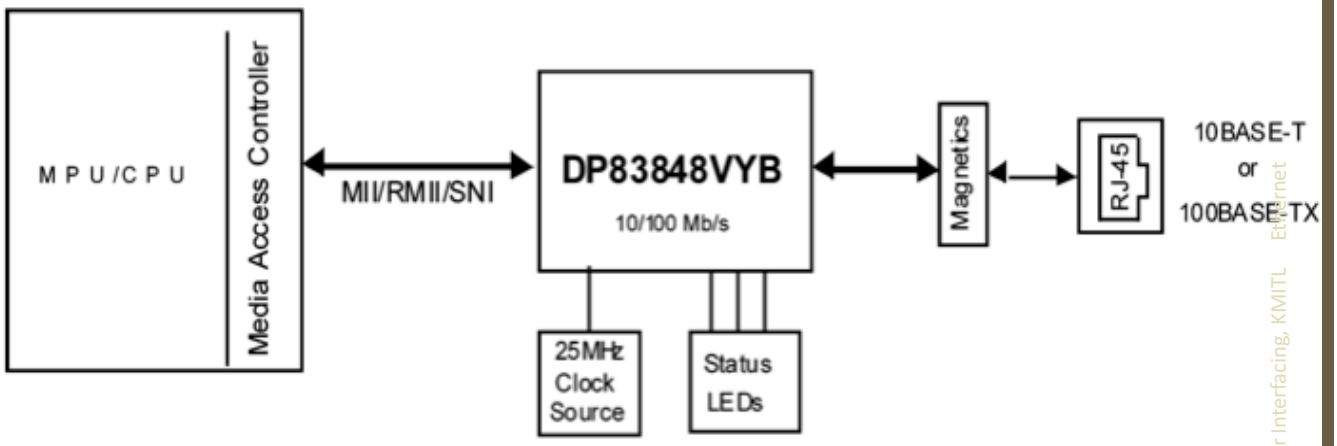
MII Timing Diagram: TX



MII Timing Diagram: RX

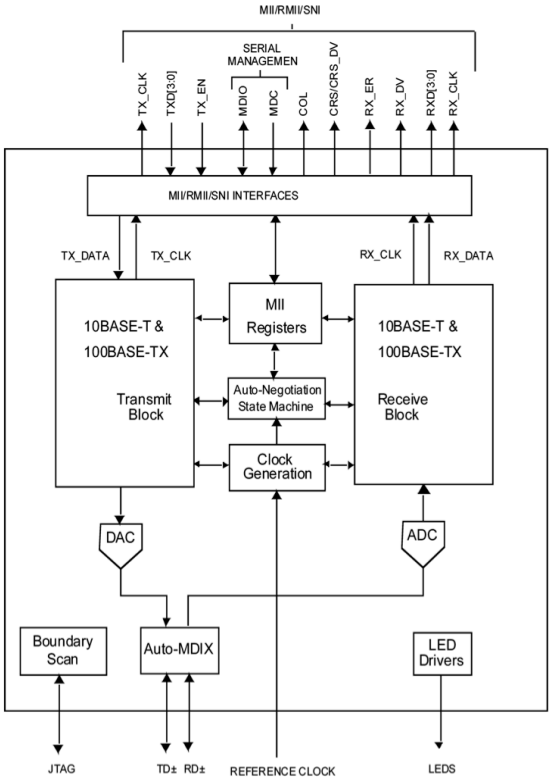


Ethernet Phy: DP83848

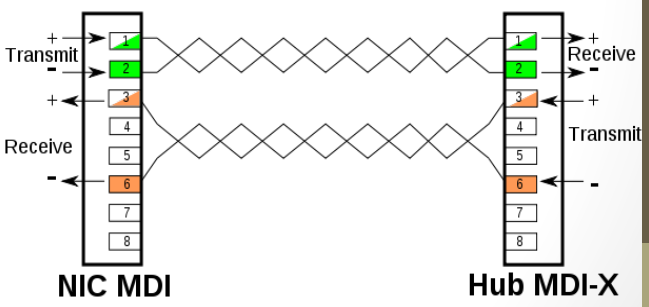


- Physical Layer Transceiver and Manufactured by Texas Instrument
- Supports MII/RMII/SNI (Media Independence Interface/Reduced Media Independence Interface/Serial Network Interface) with the MAC (Media Access Controller)

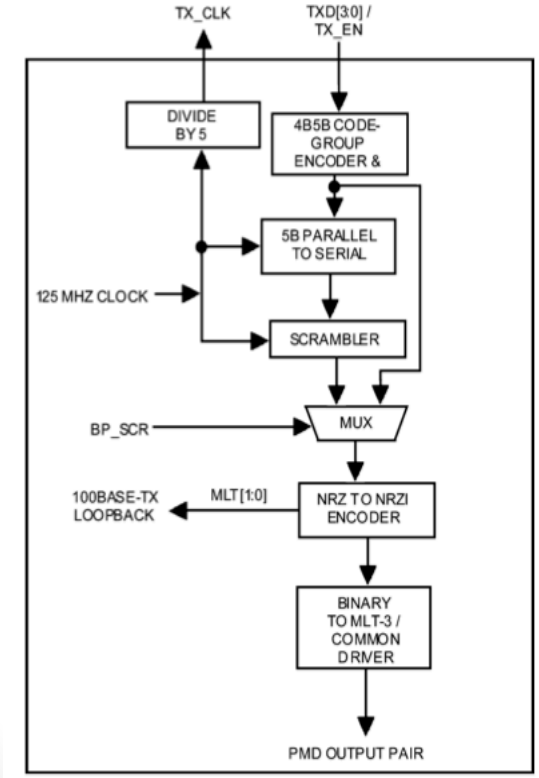
DP83848: Block Diagram



- TD (+/-) = Transmit Data (+/-)
- RD (+/-) = Receive Data (+/-)
- MDI = Media Dependent Interface
- MDIX = MDI Cross
- DAC = Digital to Analog Converter
- ADC = Analog to Digital Converter

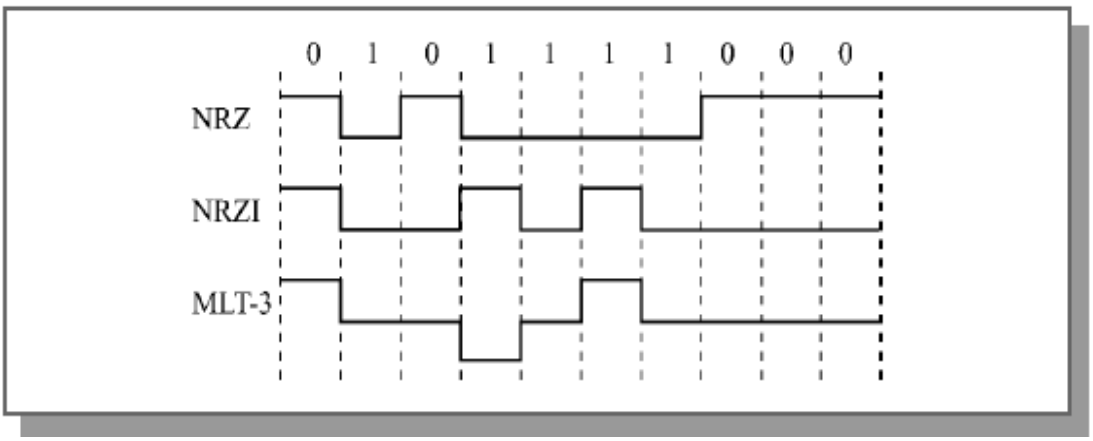


DP83848: Tx Block Diagram



- แปลงข้อมูลจาก 4 bit => 5 bit (4B5B)
- Scrambler ทำการสลับข้อมูล 5 bit
- แปลงข้อมูลเป็น NRZ (Non Return to Zero)
- แปลงต่อให้เป็นแบบ NRZI (Non Return to Zero Inverted)
- แปลงจาก NRZI ให้เป็น MLT-3 (Multi Level Transmit)-3
- สัญญาณ MLT-3 จะถูกส่งต่อให้ PMD (Physical Medium Dependent) แปลงเป็นไฟฟ้า หรือ แสง ขึ้นอยู่กับสายที่ใช้

NRZ, NRZI, MLT-3



- MLT-3 cycles sequentially through the voltage levels −1, 0, +1, 0. It moves to the next state to transmit a 1 bit, and stays in the same state to transmit a 0 bit.
- MLT-3 has a coding efficiency of 1 bit/baud, however it requires four transitions (bauds) to complete a full cycle.
- Thus, the maximum fundamental frequency is reduced to one fourth of the baud rate. This makes signal transmission more amenable to copper wires.

4B5B vs MLT-3

- 4B5B แปลงข้อมูล 4 bit ให้เป็น 5 bit
- 4B5B ทำหน้าที่เพิ่ม “1” เข้าไปในข้อมูล
- 4B5B เพิ่มสัญลักษณ์ ในการเชื่อมต่อ เช่น idle, start of stream..
- MLT-3 แปลงข้อมูล ให้เป็นไฟกระแสสลับ เพื่อให้ส่งสัญญาณไปได้ไกลขึ้น

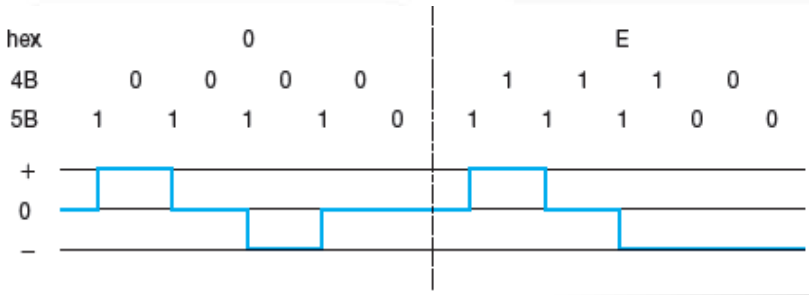
4B/5B Code Symbols

Data Codes

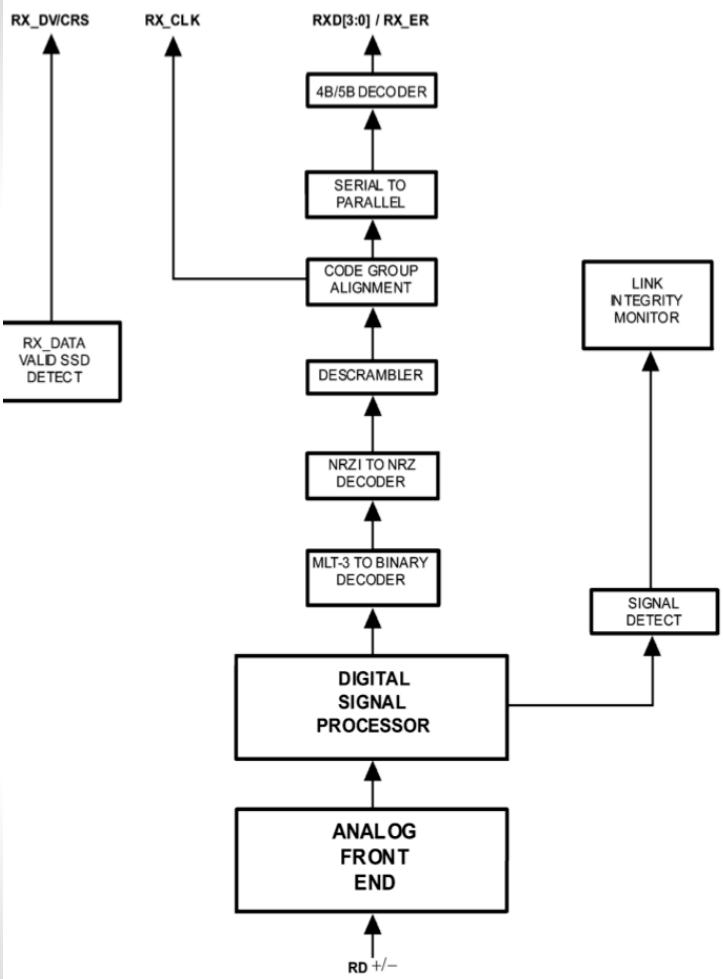
4B Code	5B Symbol
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Control and Invalid Codes

4B Code	5B Symbol
idle	11111
start of stream	11000
start of stream	10001
end of stream	01101
end of stream	00111
transmit error	00111
invalid	00000
invalid	00001
invalid	00010
invalid	00011
invalid	00100
invalid	00101
invalid	00110
invalid	01000
invalid	10000
invalid	11001



DP83848: Rx Block Diagram



- Digital Signal Processor filters and processes the received signal to MLT-3 signal.
- Descrambler <> Scrambler
- 4B5B Decoder decodes the 5 bits code to 4 bits data.
- Code Group Alignment extracts the RX_CLK.

USB 2.0 & 3.0

ผศ.ดร. สุรินทร์ กิตติธรกุล และ อ.สรยุทธ กลมกล่อม
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

USB Specifications

- USB 1.0 – The original spec, superseded quickly by USB 1.1
- USB 1.1 – Minor enhancements to the USB 1.0 spec, supported Low and Full speed devices
- USB 2.0 – The Current Standard, added "high speed" to USB 1.1
- USB OTG – USB On-the-Go, a supplement to the USB 2.0 spec that added a form of device to device communications.
- WUSB – Wireless USB, a work in progress -- not a standard yet

USB 2.0 in a Nutshell

- Runs 40X faster than USB 1.1
  - Low speed: 1.5Mb/s
  - Full speed: 12Mb/s
  - High speed: 480Mb/s
- Fully supports existing USB devices
  - Forward compatible—plug existing 1.1 devices into new 2.0 hosts
  - Backward compatible—plug new 2.0 devices into existing 1.1 hosts
- Uses the same cables as USB 1.1

USB On-the-Go (USB OTG)

- Connect two 'peripherals' together
  - PC is not required (but still supported)
- Allow peripherals to wake up hosts
- Allow two devices to exchange the host role
- New OTG devices can tap into the existing 900 million USB devices

USB OTG Example Applications

Source: Beeman: www.usbonthego.com

Host	Peripheral	Applications
PC	Cell Phone Camera MP3 player	Synchronize phone list, meetings Upload pictures Download songs
Cell Phone	Cell Phone Printer Camera	Exchange Contact info Print call info Upload pictures to web
Camera	Camera Cell Phone Printer	Exchange pictures Upload pictures to web Print pictures
MP3 player	MP3 player	Exchange songs

USB OTG Details

- Defines a new connector and cable
  - "Mini AB" receptacle
  - Mini-A to Mini-B cable
  - Cable establishes the default host
    - A-Device is the default host
- Dual-role USB devices:
  - Are sometimes a peripheral, sometimes a host
  - Must use the new AB connector
  - Provide limited host capability
    - Targeted peripheral list
  - Operate at full speed (high speed optional)

USB Basics

- USB is a Master/Slave Polled Bus (PC is the "Master", Devices are the "Slaves")
- USB has a "tiered star" architecture that can USB can support up to 126 devices
  - 7-bit address = 128 – root hub – reserved addr 0 for enumeration
- USB is "Hot Pluggable"
- USB connections can provide both data and power to the devices

Components of USB Systems

- USB Host Controllers
- USB Hubs
- USB Cables
- USB Peripheral Devices

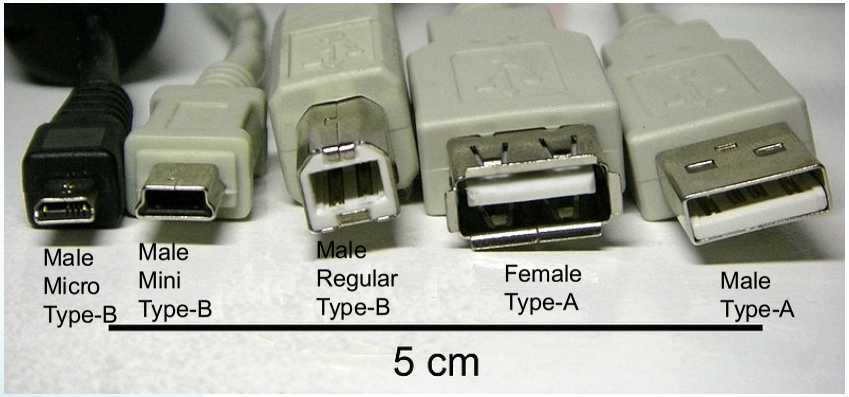
USB Host Controllers

- Reside in PC on motherboard or add-in card
- Are the "master" device on the USB bus
- Host Controller has integral "Root Hub"
- Host Controller Interfaces:
  - Universal Host Controller Interface (UHCI)
  - Open Host Controller Interface (OHCI)
  - Enhanced Host Controller Interface (EHCI)

USB Cables

- Transport both Data and Power
- Four wires: Vbus, GND, D+, D–
- Cables are 5 meters MAX
- Two connector types, "A" and "B"
  - Prevents illegal topologies
  - "A" connectors are the ones that goes to the PC or Hub, "B" connectors goes to the device) (OTG adds Miny A and Mini B connectors)
- Cables can be either "captive" (like mice) or detachable
- Cables can be unshielded (Low speed devices) or shielded (Full Speed and High Speed devices)
- USB extension cables are *ILLEGAL*

USB 2.0 cables and connectors



Standard-A USB 3.0 Receptacle and Plug

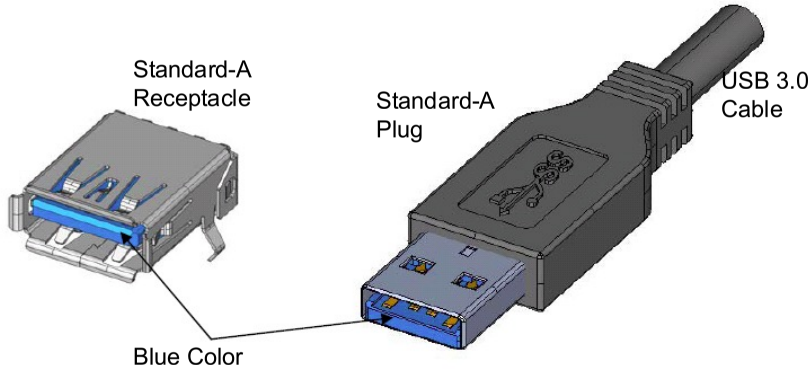
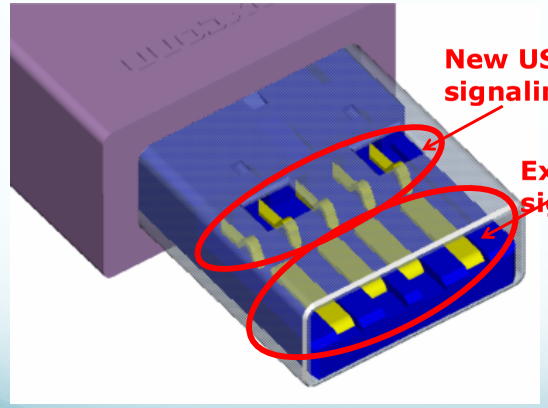


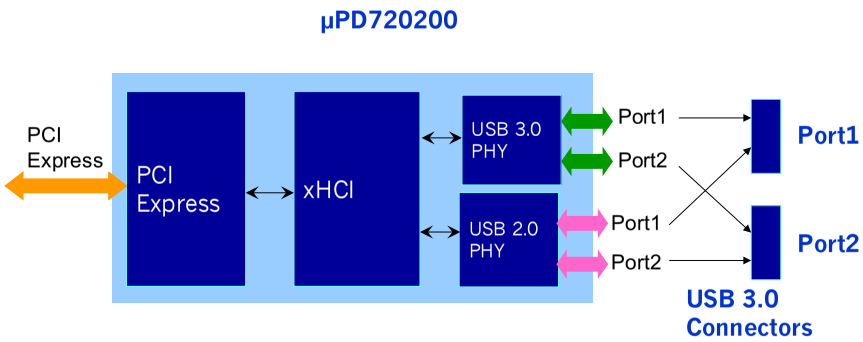
Figure 5-5. Illustration of Color Coding Recommendation for USB 3.0 Standard-A Connector

USB3.0 Connector Compatibility

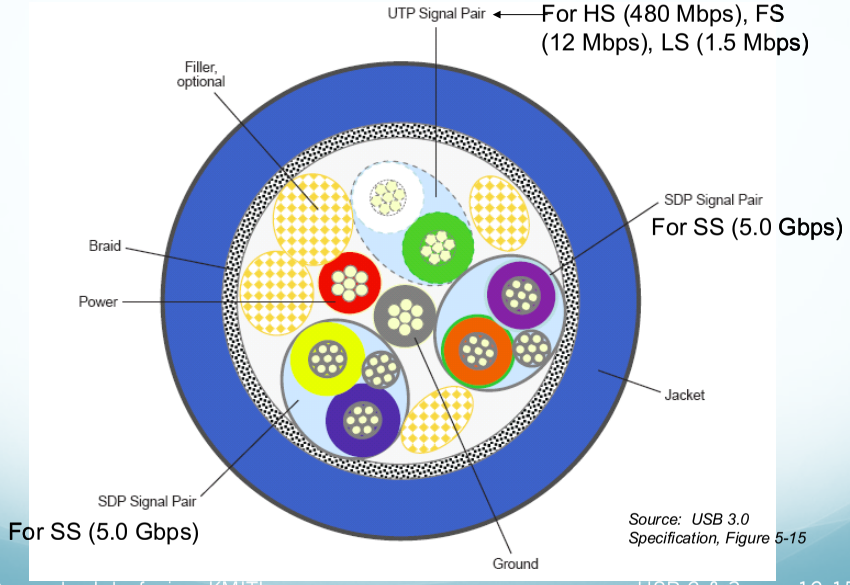
USB3.0 A connecter has same form factor as in USB2.0



USB 3.0 from Renesas

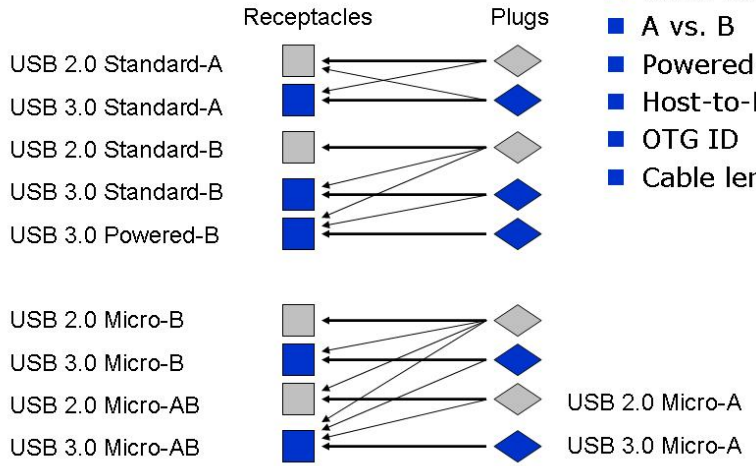


World's first



USB 3.0 changes to PCB and cables and connectors (examples)

USB 3.0 Supports USB 2.0

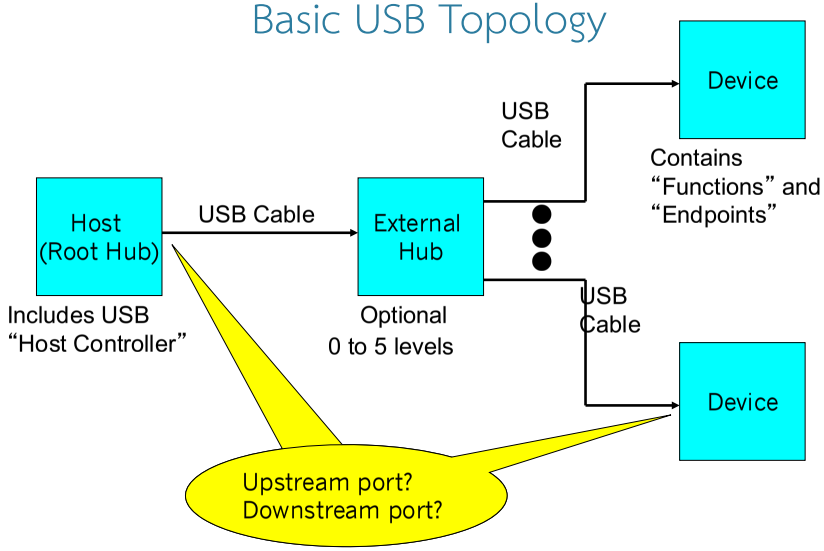


- USB3 vs. USB2
- A vs. B
- Powered-B
- Host-to-Host
- OTG ID
- Cable length

Reference: USB 3.0 Specification, Table 5-1

© 2010 Renesas Electronics America Inc. All rights reserved.

Basic USB Topology



USB Hubs

- A hub provides additional connection points (ports) for devices
- Hubs can be:
  - Self-Powered (<= 500ma to each device)
  - Bus-Powered (<= 100mA to each device)
- Hubs contain most of the "magic" and differences between USB 1.1 and USB 2.0

USB 3.0 Hubs

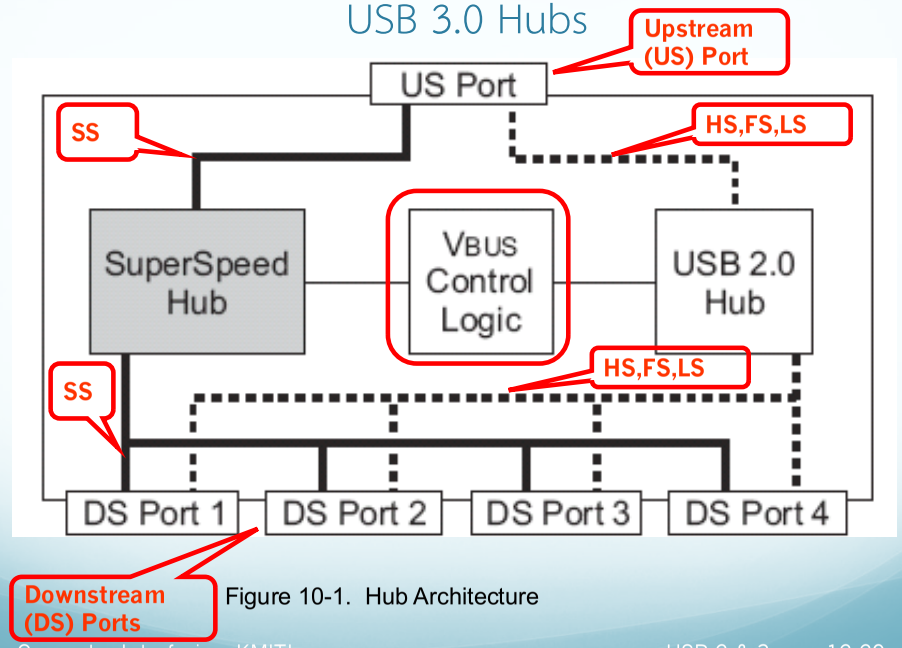


Figure 10-1. Hub Architecture

USB Devices

- Devices can be:
  - Self-Powered
    - External Power Source (i.e. wall wart)
    - Batteries
  - Bus-Powered
    - Low Power Bus Powered (<= 100mA)
    - High Power Bus Powered >100mA, <=500mA)
- Devices are "self describing"
  - Devices return data (USB descriptors) to host to indicate its capabilities, configurations, and how the device is able to communicate

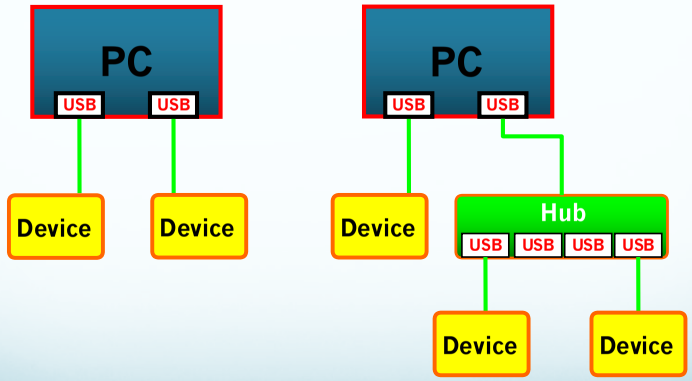
Compound and Composite Devices

- Composite Device
  - Multiple interfaces, independently controlled
    - Each interface can have a different driver
- Compound Device
  - Collection of separate functions, each with a USB address, connected to an internal hub
  - Example: Keyboard & Trackball in same package

USB is a Polled BUS

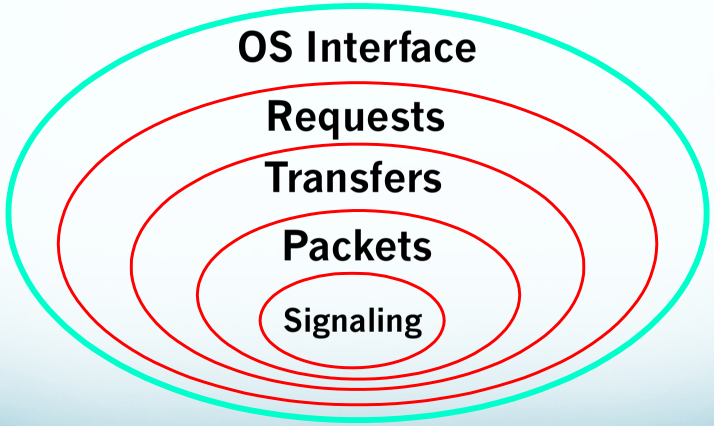
- The Host (PC) initiates all transfers
- Devices respond to host requests
- Direction: OUT is host-to-device
- Direction: IN is device-to-host
- USB is NOT peer-to-peer (not even in USB OTG)

USB Topology



"Tiered Star"

The USB Onion



Enumeration

1. The user plugs the device into a USB port.
2. The hub detects the device.
3. The host learns of the devices presence from the hub (Get Port Status Request)
4. The hub detects whether a device is Low speed or Full Speed.
5. The hub resets the device.
6. The host learns if a full speed device supports high speed.
7. The hub establishes a signal path between the device and the bus.
8. The host sends a Get Descriptor Request to learn the maximum packet size of the default pipe.
9. The host assigns an address (Set Address Request).
10. The host learn about a devices abilities (Get Descriptor Request).
11. The host assigns and loads a device driver.
12. The host device driver selects a configuration (Set Configuration Request).

USB Software

- USB software is a major task
  - Host Controller driver
  - Additional "class drivers"
  - Device-specific drivers
- USB 2.0 software today – PC & MCU
- USB 3.0 software still very new, PC only
  - Can MCUs support SS transfer rates?
  - Mass storage peripherals for PCs
  - LucidPort USB300 ASIC

USB Standards for Host Controller Software

- **USB 2.0 (Full/Low-speed)**
  - **UHCI** (Universal Host Controller Interface)
  - **OHCI** (Open Host Controller Interface)
- **USB 2.0 (for High-speed)**
  - **EHCI** (Enhanced Host Controller Interface) (Released from Intel)
- **USB 3.0 (Super/High/Full/Low-speed)**
  - **xHCI** (extensible Host Controller Interface (Released from Intel)

Endpoints

- USB Spec: "a source or sink of data"
  - A Control Transfer Type endpoint is bi-directional
  - Others are uni-directional
- Four address bits plus a direction bit selects between up to 32 buffers (FIFOs)
- Different USB chips support
  - Various numbers of endpoints
  - Various buffer sizes
- Each device must have 1 Control Type Endpoint (Commonly referred to as Endpoint 0 or the Default Endpoint)

Pipes

- An abstraction used by the USB spec used to indicate that 2 endpoints are joined (one in the PC Host and the other in the device).
  - CONTROL pipe is bi-directional
  - Others are uni-directional

USB Descriptors

- Device Descriptors
  - Contains the VID/PID/DID/Serial Number
- Configuration Descriptors
- Interface Descriptors
- Endpoint Descriptors
- String Descriptors
- Device Qualifier Descriptors (USB 2.0)
- Other Speed Configuration Descriptors (USB 2.0)
- USB "Classes" can add other class specific Descriptors

Host Controller Driver

- Fundamental component of Operating System support for USB
- USB host controllers are PCI devices
- What Does it Do?
  - Handles USB peripheral enumeration
  - Provides USB services for higher level drivers. All access to USB peripherals is via these services

USB Peripheral Drivers

- Class Drivers
  - Generic driver that supports a certain class of device
  - Human Interface Devices (HID), USB hubs, speakers, mass storage, modems
  - *Note: Not all USB Classes that are defined by the USB-IF are implemented!!! Check your target OS for support. (i.e. I know of no OS that has implemented the Firmware Update Class)*
- Custom Drivers
  - Operating Systems often include vendor specific drivers that have passed certification (e.g. WHQL)

Custom Device Driver Options

- Look for an Open Source Device Driver
  - http://libusb-win32.sourceforge.net (Windows)
  - http://libusb.sourceforge.net (Linux and others)
- Use a "General Purpose" Device Driver
- Write a "Proprietary Custom Device" Driver (or have it written for you by a consultant)

HID (Human Interface Device) Class

- Built into Windows 98 (or later)
- No need to write (or install) a driver!
- Used by mice and keyboards, but can be used for custom devices.
- Communicates with devices using reports
  - Set Report, Get Report
- Full Spec at http://www.usb.org
  - You need the spec, *and* the HID Usage Tables

Standard USB Requests

- Get/Set Status
- Get/Set Descriptor
- Get/Set Configuration
- Get/Set Interface
- Set/Clear Feature
- Sync Frame

Class Specific Requests

- Specific to a given class, see the class specification for details.
- Examples:
  - The Human Interface Device (HID) Class adds "Get Report and "Set Report" requests
  - The Hub Class adds a "Get Port Status" request

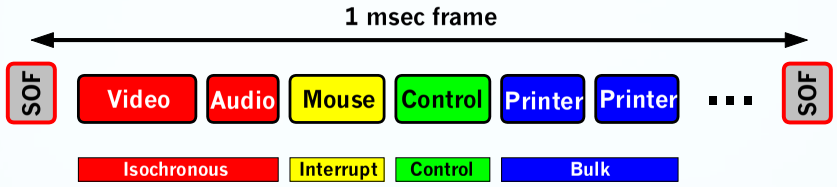
USB Transfer Types

- Bulk
  - Guaranteed accuracy, but delivery time is variable
  - Best for "bursty" data
- Isochronous
  - Guaranteed delivery time, but accuracy is not guaranteed
- Control
  - Enumeration and device control
- Interrupt
  - Predictable polling time

USB Transfers

- USB Transfers Occur in 1ms Frames
  (USB 2.0 adds 125 uS Microframes)
- Host sends "SOF" (Start Of Frame) token every 1ms
- Host schedules packets inside frames.

Anatomy of a USB Frame

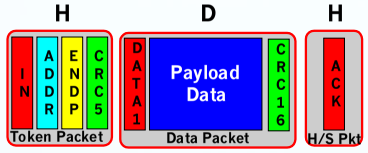


- 12MHz = 1.5MB/s or 1500 bytes/ms
- Isochronous/Interrupt traffic have guaranteed bandwidth
- Control traffic is "best-effort"
- Bulk uses what is left
- Actual scheduling order depends on host controller
  - UHCI, OHCI

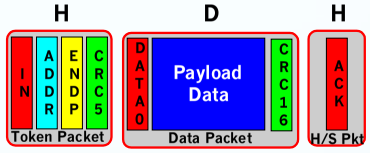
Packet Sizes

Transfer Type	*Packet Size*	
	USB 1.1	USB 2.0
Control	8, 16, 32, 64	64
Bulk	8, 16, 32, 64	512
Interrupt	1–64	1024
Isochronous	1023	1024

Bulk IN Transfer (a)

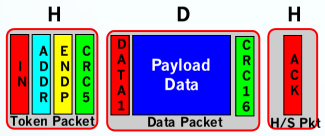


good

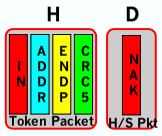


good—note the data toggle

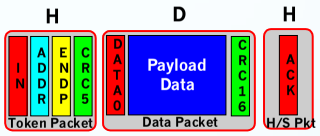
Bulk IN Transfer (b)



good

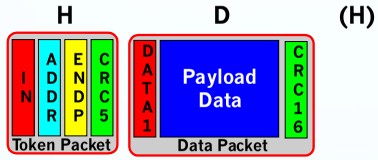


not ready

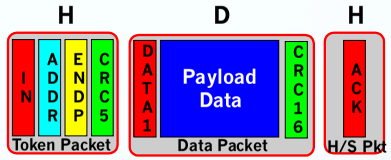


good

Bulk IN Transfer (c)

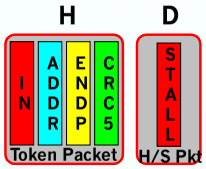


host sees error; no response

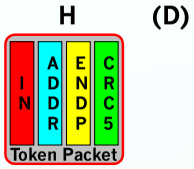


good—note same data and data toggle

Bulk IN Transfer (d)

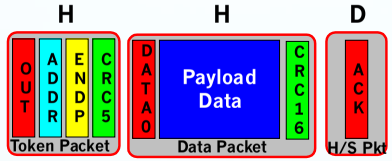


device has a problem

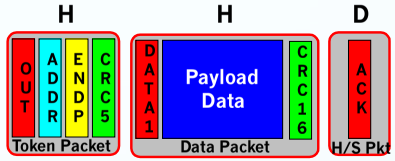


device detects token error
or does not respond

Bulk OUT Transfer (a)

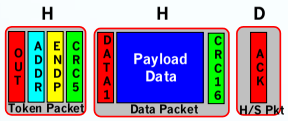


good

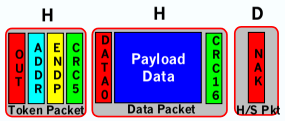


good—note the data toggle

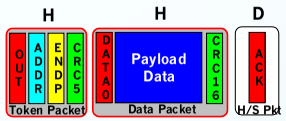
Bulk OUT Transfer (b)



good

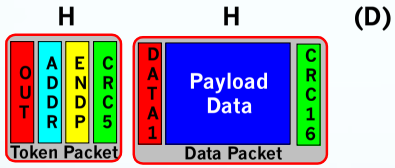


device not ready for data
host sends data anyway

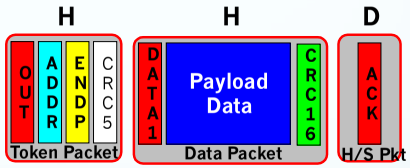


good

Bulk OUT Transfer (c)



device sees error; no response



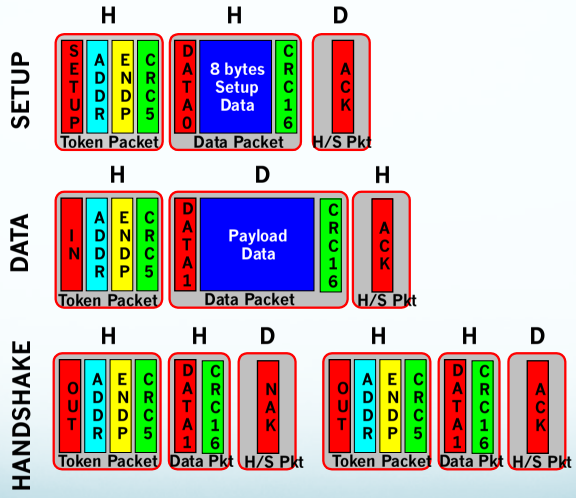
good

Bulk OUT Transfer (d)

device has problem

device detects token error

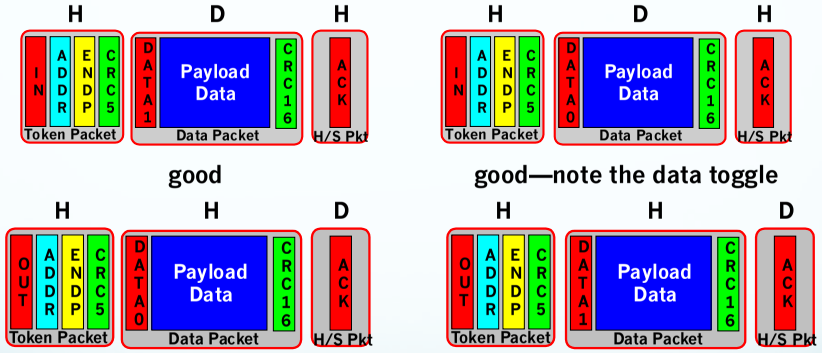
Control Transfer



Control operation not completed

Control operation completed

Interrupt IN & OUT



good

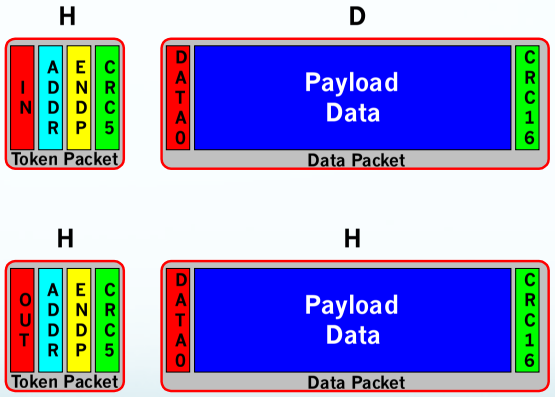
good—note the data toggle

good

good—note the data toggle

Interrupt transfers are indistinguishable from BULK transfers. They occur at most once per frame.

Isochronous Transfer

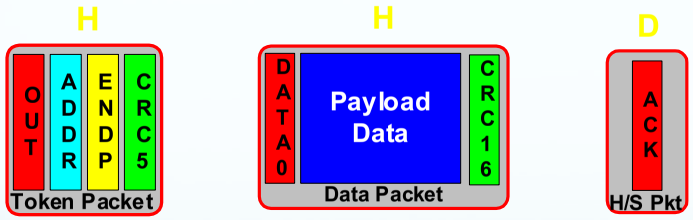


Note: Always DATA0 PID, no ACK

USB Packets

- USB data travels in packets
- Identified by "Packet ID" (PID)
- Token packet tells what's coming
- Data packets deliver bytes
- Handshake packets report success or otherwise

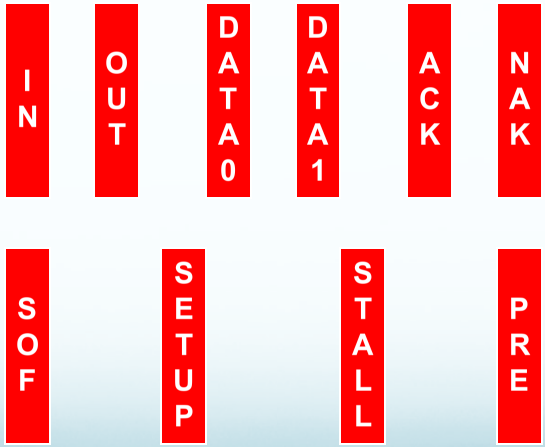
Three Packet Types



Packet IDs (PIDS)

- Token Packets
  - IN, OUT, SOF, SETUP
- Data Packets
  - DATA0, DATA1 (USB LS/FS)
  - DATA2, MDATA (USB HS)
- Handshake Packets
  - ACK, NAK, STALL (USB LS/FS)
  - NYET (USB HS)

Packets: Identified by PIDS

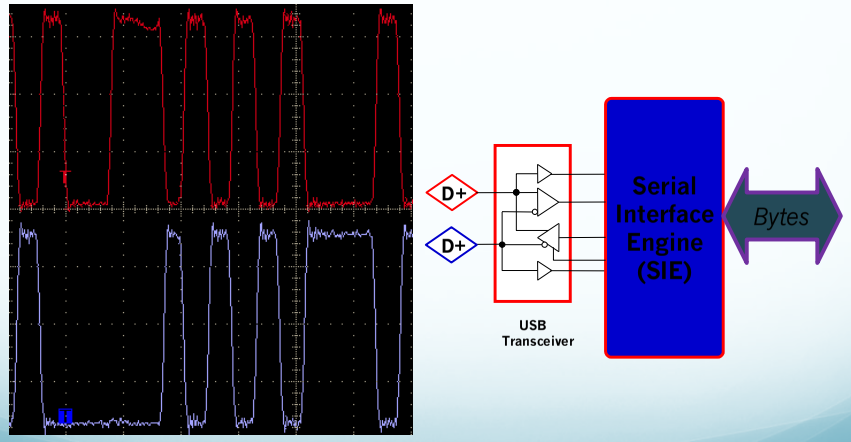


Handshaking Packets

- Used to ensure correct data delivery on Control, Bulk and Interrupt Transfers
- ACK – Received with out Error
- NAK – Device Busy, has no data
- Stall – Unsupported Request, Request Failed
- No Response – Request not received or corrupt, will be retried up to 3 times.

USB Signaling

- USB utilizes differential signaling on the D+ and D- lines.
- Data is encoded in "non-return to zero with bit stuffing"
  - Bit stuffing is used to ensure enough transitions for the clock recovery circuitry.
- Most designs use silicon that incorporates an integral Serial Interface Engine (SIE) so you don't have to worry about the decoding yourself.
- PCB layout of D= and D- requires care (especially on High Speed devices).

The SIE

(Serial Interface Engine)