Peripherals and I/O Devices

Shao-Yi Chien

Outline

Various Kinds of Peripherals!

- System
  - □ Timers
  - □ Counters
  - □ Watchdog timers
  - □ Real-time clocks
- Serial I/F
  - □ UART
  - □ I^{2}C
  - □ I^{2}S
  - □ SPI
  - □ FireWire
  - □ USB
  - □ Thunderbolt

- A/D and D/A converters
- GPIO

- Display I/F
- Image sensor I/F
- Keypad controllers
- Pulse width modulators (PWM)
- Stepper motor controllers
- Communication peripherals
- ……

- **MIPI**

Introduction

- Peripherals are often **single-purpose processors**
  - Performs specific computation task
  - Custom single-purpose processors
    - Designed by us for a unique task
  - *Standard* single-purpose processors
    - "Off-the-shelf" -- pre-designed for a common task
    - i.e. peripherals
    - serial transmission
    - analog/digital conversions
- I/O peripherals are the communication channels between the SoC and the real-world
- The functions of an SoC determines the requirements of peripherals

System Peripherals

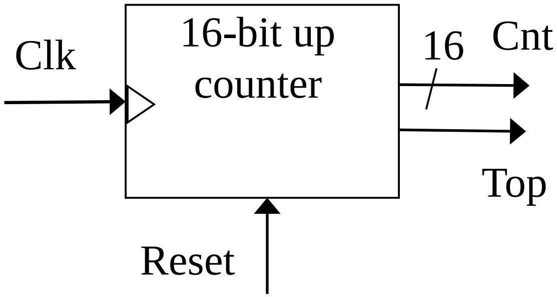
- Timers
- Counters
- Watchdog timers
- Real-time clocks

- Are often integrated in a microcontroller

Timers

- Measures time intervals based on counting clock pulses
    - □ To generate timed output events
        - e.g., hold traffic light green for 10 s
    - □ To measure input events
        - e.g., measure a car’s speed

Basic timer



Timers

- **Range** and **resolution** are important parameters
- Example
  - E.g., let Clk period be 10 ns
  - And we count 20,000 Clk pulses
  - Then 200 microseconds have passed
  - 16-bit counter would count up to 65,535*10 ns = 655.35 ms **(range)**, **resolution** = 10 ns

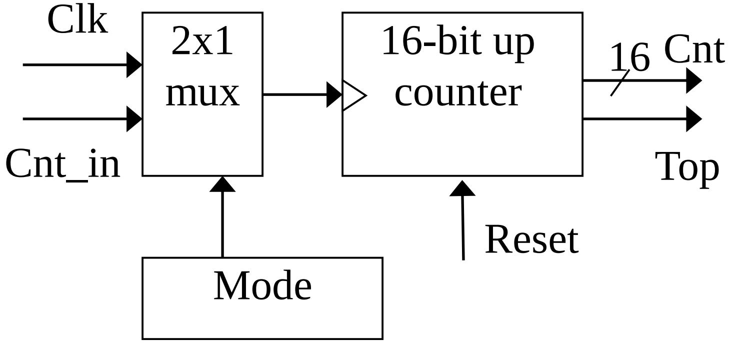
Timers

- Top: indicates top count reached, wrap-around, also known as an overflow occurring
  - The ping Top is often connected to the interrupt pin of the microprocessor
- How to extend the range?
  - Create interrupt service routine to count the number of times the overflow occurs

Counters

- Counter: like a timer, but counts pulses on a general input signal rather than clock
  - e.g., count cars passing over a sensor
  - Can often configure device as either a timer or counter

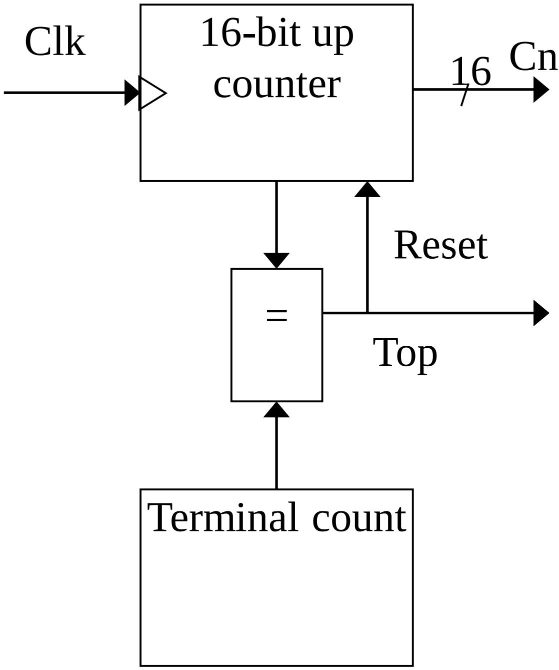
Timer/counter



Other Timer Structures

- Interval timer
  - □ Indicates when desired time interval has passed
  - □ We set terminal count to desired interval
    - *Number of clock cycles = Desired time interval / Clock period*
- An alternative form: count down from terminal count to 0
  - □ The hardware cost is lower because of the zero-checking circuits is simpler

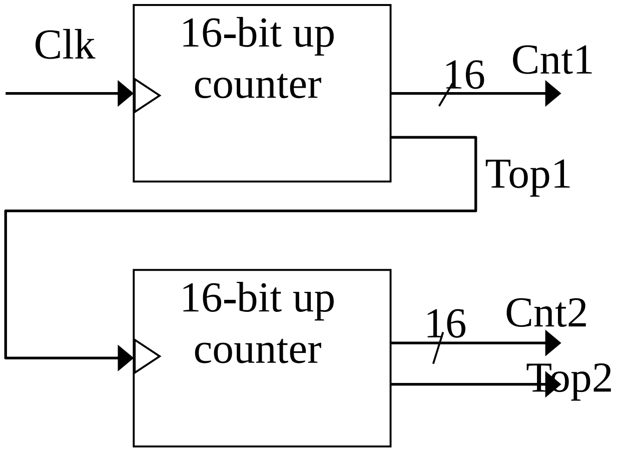
Timer with a terminal count



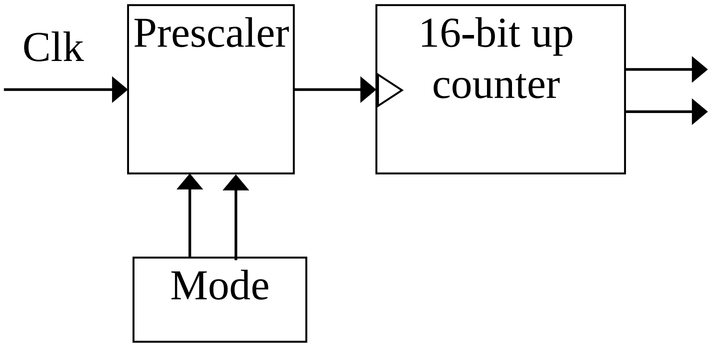
Other Timer Structures

- Cascaded counters
- Prescaler
  - □ Divides clock
  - □ Increases range, decreases resolution

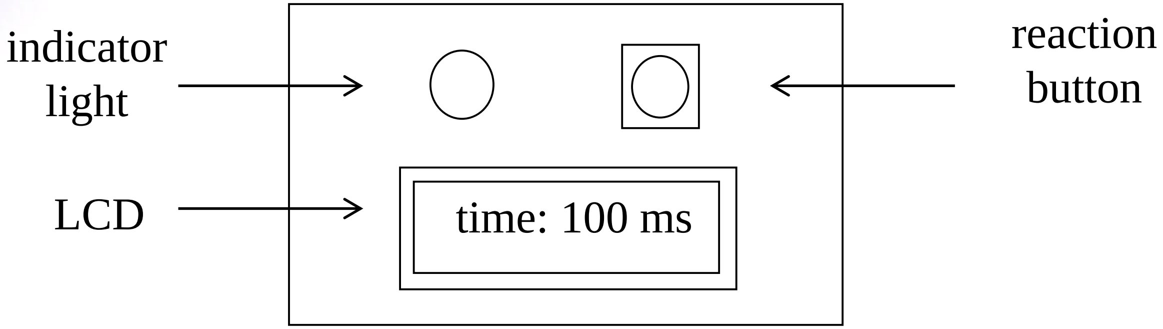
16/32-bit timer



Timer with prescaler



Example: Reaction Timer



- Measure time between turning light on and user pushing button
  - □ Precision requirement: millisecond
  - □ 16-bit timer, clk period is 83.33 ns, counter increments every 6 cycles (no prescaler is available)
  - □ Resolution = 6*83.33=0.5 microsec.
  - □ Range = 65535*0.5 microseconds = 32.77 milliseconds
  - □ Want program to count millisec., so initialize counter to 65535 – 1000/0.5 = 63535

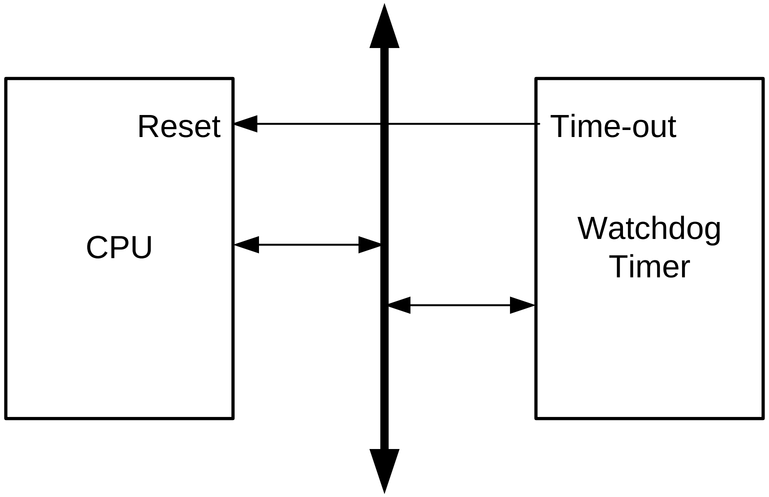
Example: Reaction Timer

```
/* main.c */

  #define MS_INIT       63535
  void main(void){
     int count_milliseconds = 0;

     configure timer mode
     set Cnt to MS_INIT

     wait a random amount of time
     turn on indicator light
     start timer
```

```
while (user has not pushed
   reaction button){
      if(Top) {
         stop timer
         set Cnt to MS_INIT
         start timer
         reset Top
         count_milliseconds++;
      }
   }
   turn light off
   printf(“time:  %i ms“,
   count_milliseconds);
}
```

Watchdog Timer (WDT)

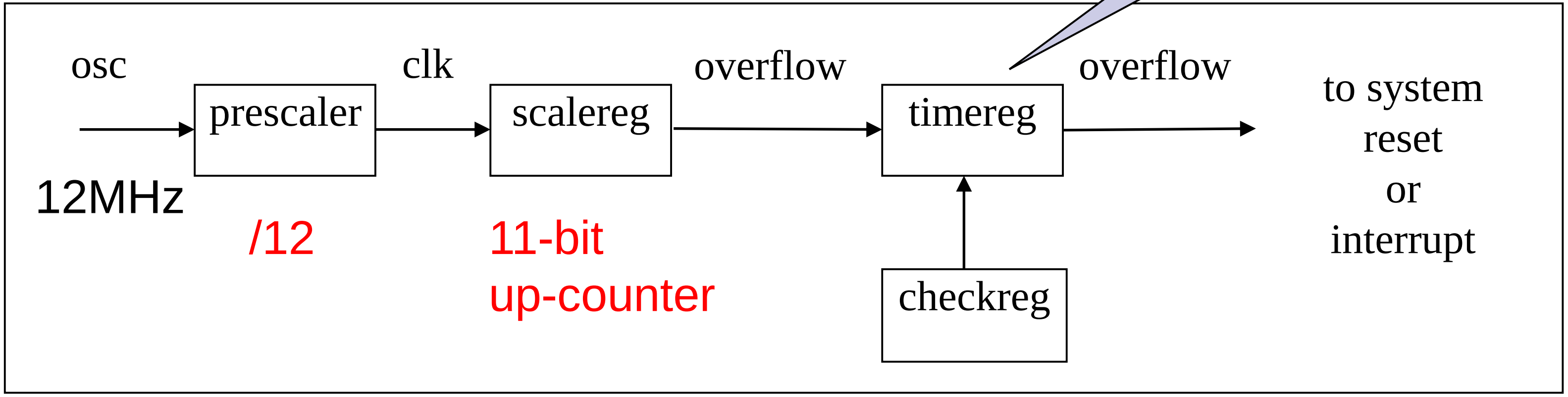
- Special type of timer
- Instead of the timer generating a signal every X time units, we must generate a signal for the timer every X time unit, or the timer will be **time out**
- Common use: detect failure, self-reset, detect time-out



Example: ATM Timeout Using a Watchdog Timer

- The longest waiting time is 2 min.
- 16-bit timer, 2ms resolution
- *timereg* value = $[2*(2^{16}-1)–X]/2 = [131070–X]/2$
- For 2 min., X = 120,000ms, timereg = 5535

16-bit up-counter Watchdog Timer



Example: ATM Timeout Using a Watchdog Timer

```
/* main.c */

main(){
   wait until card inserted
   call watchdog_reset_routine

   while(transaction in progress){
      if(button pressed){
         perform corresponding
   action
         call watchdog_reset_routine
      }

/* if watchdog_reset_routine not
   called every < 2 minutes,
   interrupt_service_routine is
   called */
}
```

```
watchdog_reset_routine(){
/* checkreg is set so we can load
   value into timereg.  Zero is
   loaded into scalereg and 11070 is
   loaded into timereg */

   checkreg = 1
   scalereg = 0
   timereg  = 5535
}

void interrupt_service_routine(){
   eject card
   reset screen
}
```

Real-Time Clocks

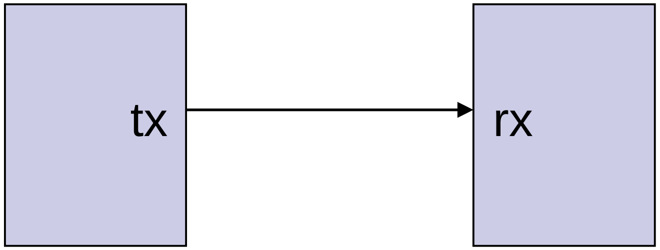
- Real-time clock (RTC) keeps the time and data in an embedded system
- Components
  - □ Crystal-controlled oscillator
  - □ Numerous cascaded counters
    - Clock cycle, second, minute, hour, date, month, year
  - □ Rechargeable back-up battery is used to keep the real-time clock running while the system is powered off
- Can be set to a desired value from microprocessor
- Can be controlled through a serial bus, such as I^2C

Serial I/F

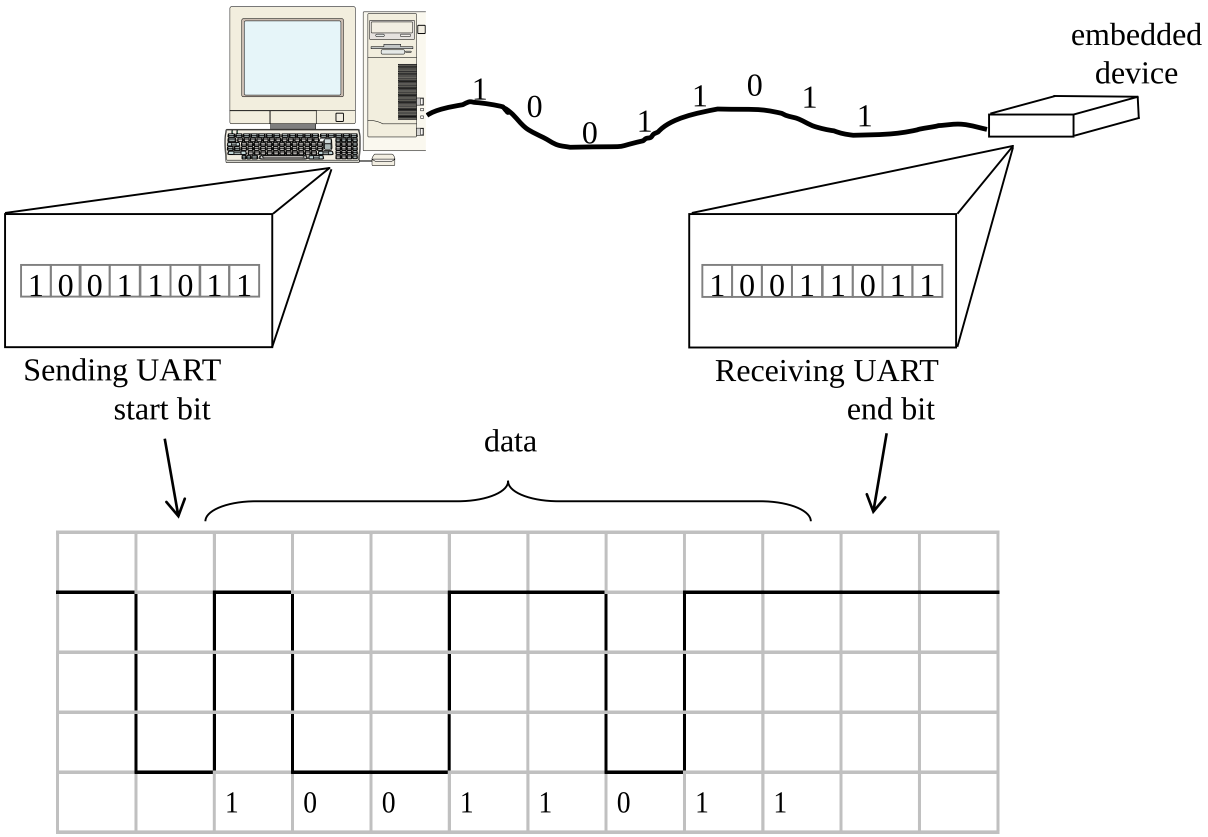
- UART
- I^{2}C
- I^{2}S
- FireWire
- USB
- Thunderbolt

UARTs

- UART: Universal Asynchronous Receiver Transmitter
  - □ Transmitter: Takes parallel data and transmits serially
  - □ Receiver: Receives serial data and converts to parallel
  - □ Data is transmitted bit by bit at predetermined intervals
- Parity: extra bit for simple error checking
- Start bit, stop bit
- Baud rate
  - □ Signal changes per second
  - □ Bit rate usually higher



UARTs

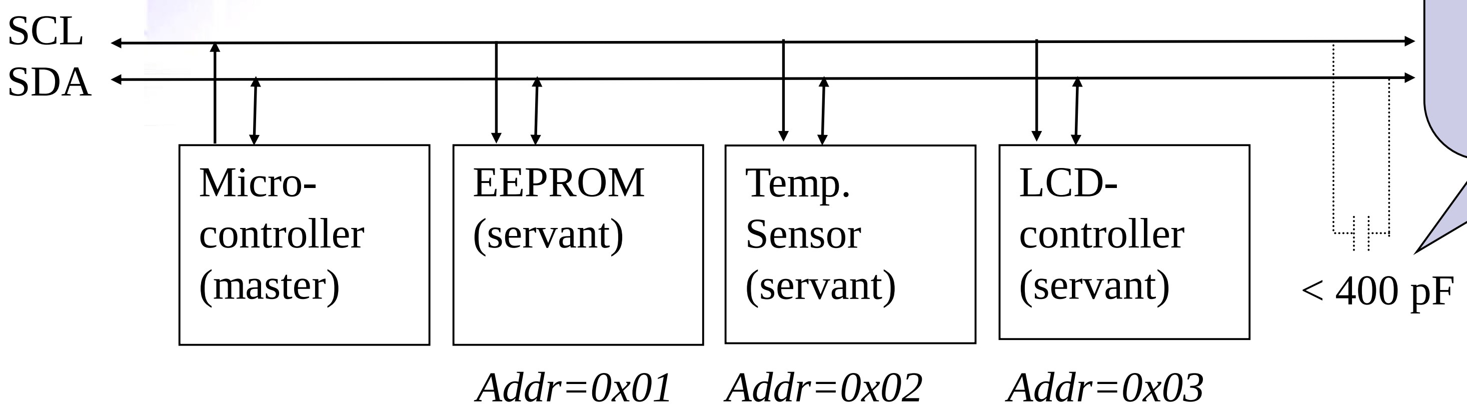

embedded
device
1 0 0 1 1 0 1 1
1 0 0 1 1 0 1 1
Sending UART
1 0 0 1 1 0 1 1
Receiving UART
start bit
data
end bit
1 0 0 1 1 0 1 1

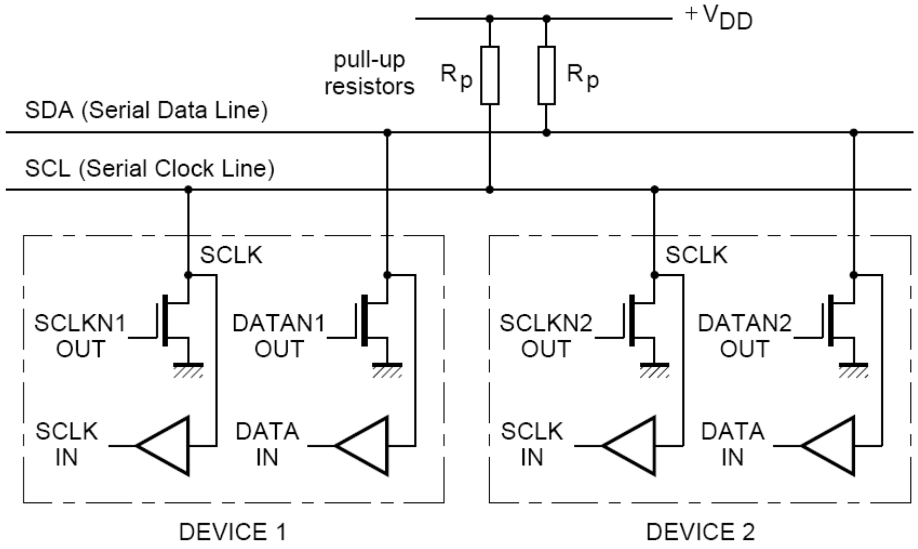
Serial Protocols: I^{2}C

- I^{2}C (Inter-IC)
  - Two-wire serial bus protocol developed by Philips Semiconductors nearly 30 years ago
  - Enables peripheral ICs to communicate using simple communication hardware
  - Data transfer rates up to 100 kbits/s and 7-bit addressing (128 devices in maximum) possible in normal mode
  - 3.4 Mbits/s and 10-bit addressing in fast-mode
  - Common devices capable of interfacing to I^{2}C bus:
    - EPROMS, Flash, and some RAM memory, real-time clocks, watchdog timers, and microcontrollers

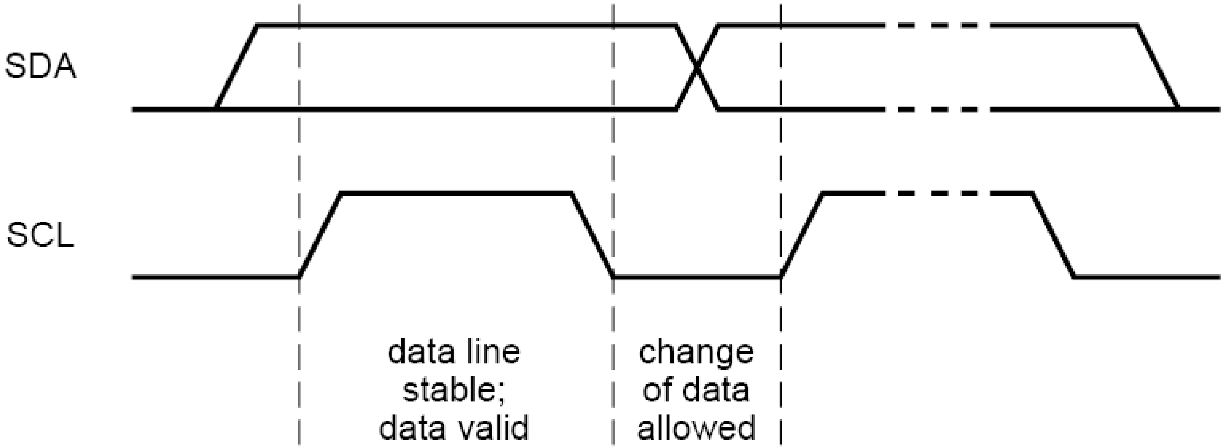
I^{2}C Bus Structure

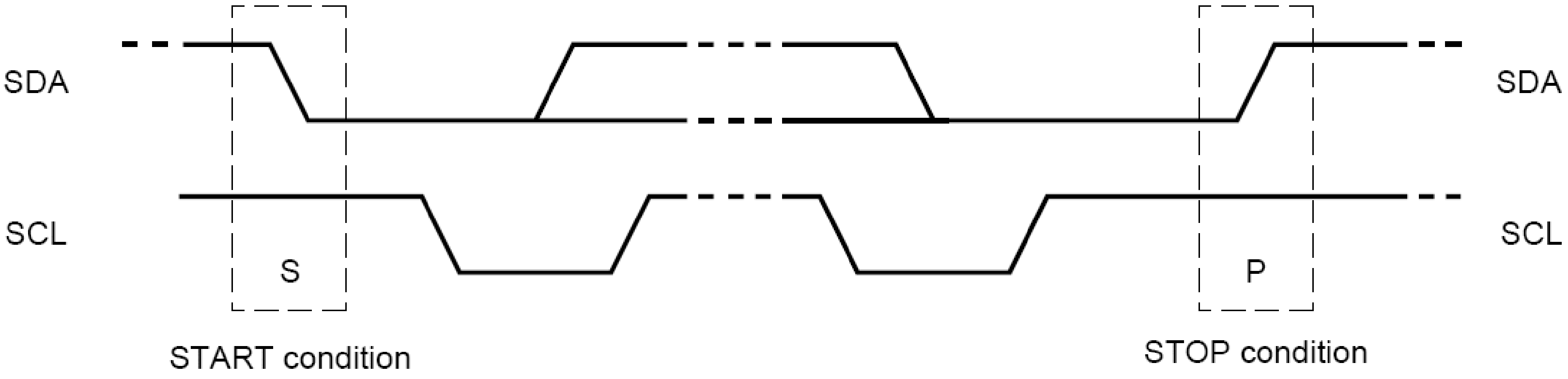
The total capacitance of the bus should remain under 400pF





I^2C Bus Signal



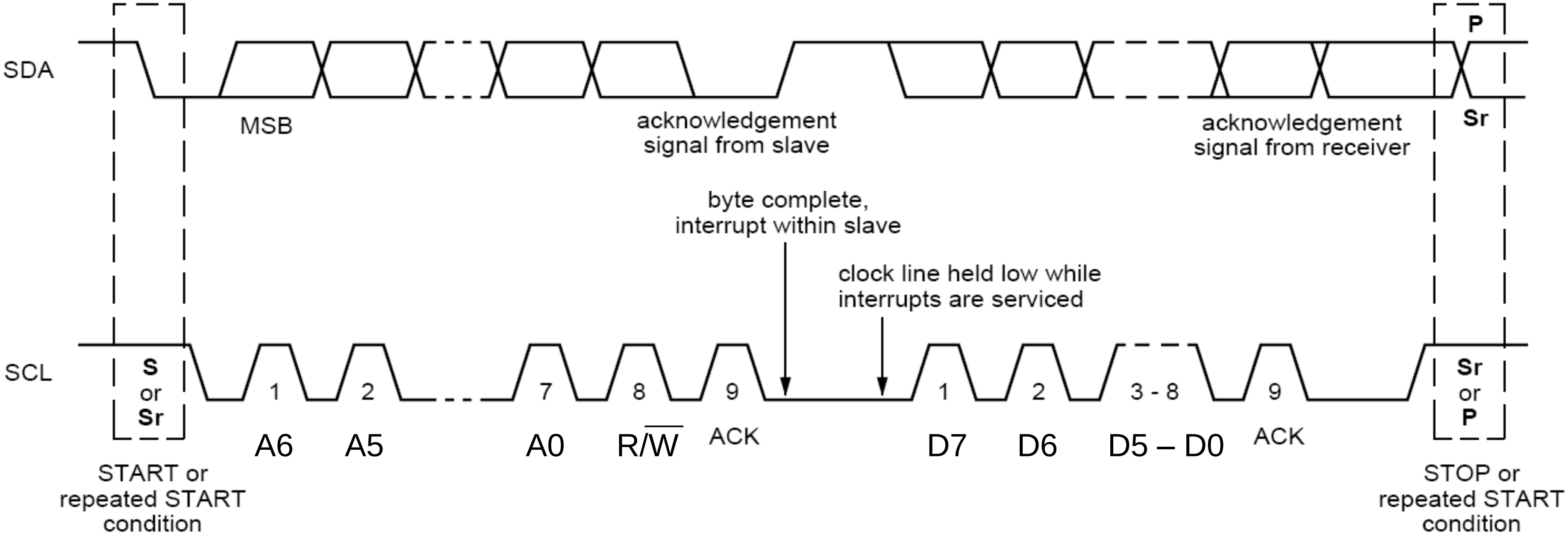


I²C Bus Data Transferring

- Standard write sequences
  - Start
  - Send address
  - Indicate write(0)
  - Wait ack
  - Send a byte from MSB to LSB
  - Wait ack
  - Stop

- Standard read sequences
  - Start
  - Send address
  - Indicate read(1)
  - Wait ack
  - Receive a byte from MSB to LSB
  - Send ack
  - Stop

I^{2}C Bus Data Transferring

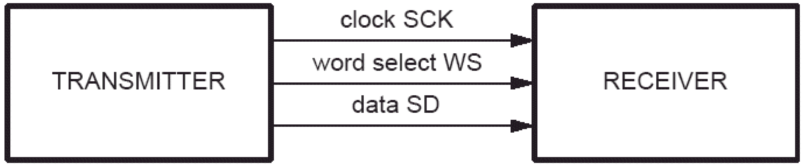

P
SDA
Sr
MSB
acknowledgement signal from slave
acknowledgement signal from receiver
byte complete, interrupt within slave
clock line held low while interrupts are serviced
SCL
S or Sr
1
2
7
8
9
1
2
3 - 8
9
Sr or P
A6
A5
A0
R/$\overline{W}$
ACK
D7
D6
D5 – D0
ACK
START or repeated START condition
STOP or repeated START condition

Serial Protocols: I^{2}S

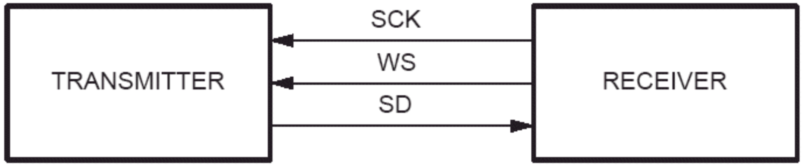
- ICs for digital audio systems (compact disc, digital audio tape, digital sound processors, and digital TV-sound)
  - A/D and D/A converters;
  - Digital signal processors;
  - Error correction for compact disc and digital recording;
  - Digital filters;
  - Digital input/output interfaces.
- Standardized communication structures are vital for both the equipment and the IC manufacturer
- Philips developed inter-IC sound (I2S) bus – a serial link especially for digital audio, in 1986

Serial Protocols: I^{2}S

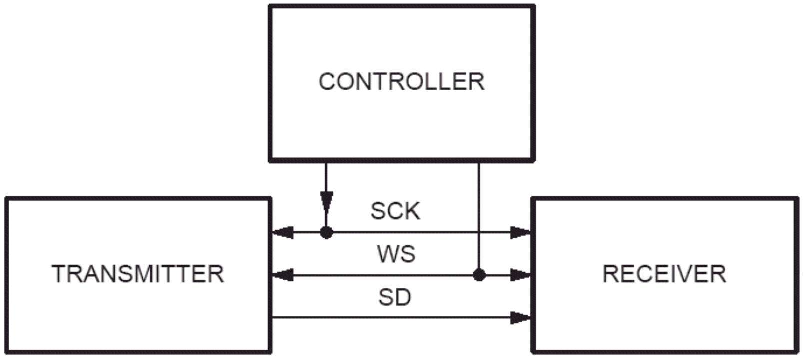
- 3-line serial bus is used consisting of a line for two time-multiplexed data channels, a word select line and a clock line.



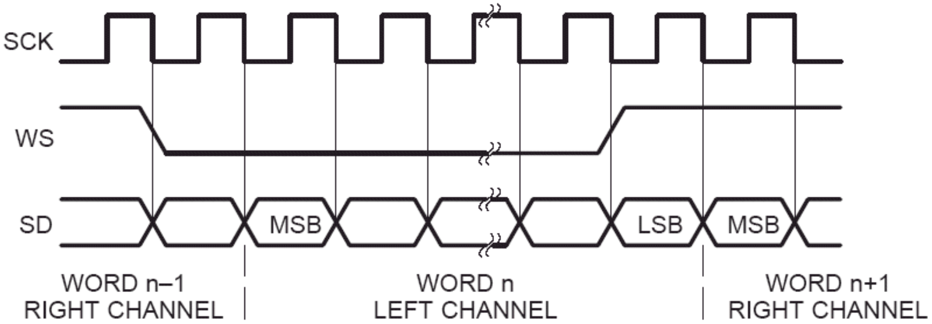
TRANSMITTER = MASTER



RECEIVER = MASTER

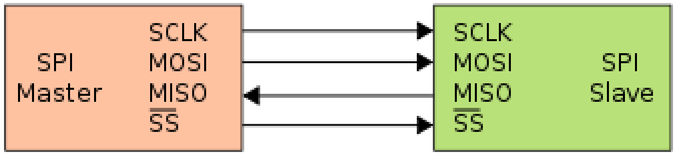


CONTROLLER = MASTER

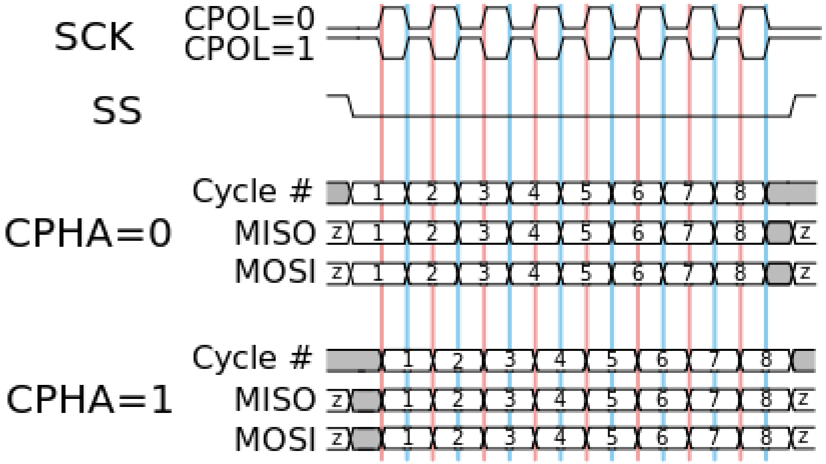


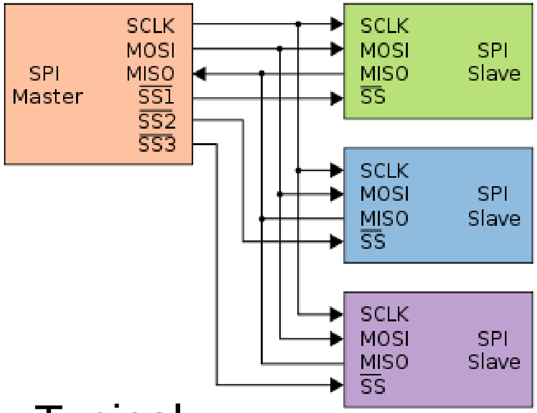
Serial Protocols: SPI

- SPI: Serial Peripheral Interface Bus
- Developed by Motorola

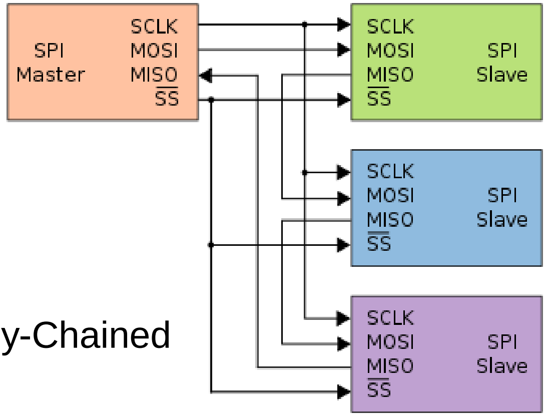


MOSI: Master output slave input
MISO: Master input slave output
SS: Slave selected (active-low)





Typical



Daicy-Chained

[wiki]

Serial Protocols: FireWire

- FireWire (a.k.a. i-Link, Lynx, IEEE 1394)
  - □ High-performance serial bus developed by Apple Computer Inc.
  - □ Designed for interfacing independent electronic components
    - e.g., Desktop, scanner
  - □ Data transfer rates from 12.5 to 400 Mbits/s, 64-bit addressing
  - □ Plug-and-play capabilities
  - □ Packet-based layered design structure

Serial Protocols: FireWire

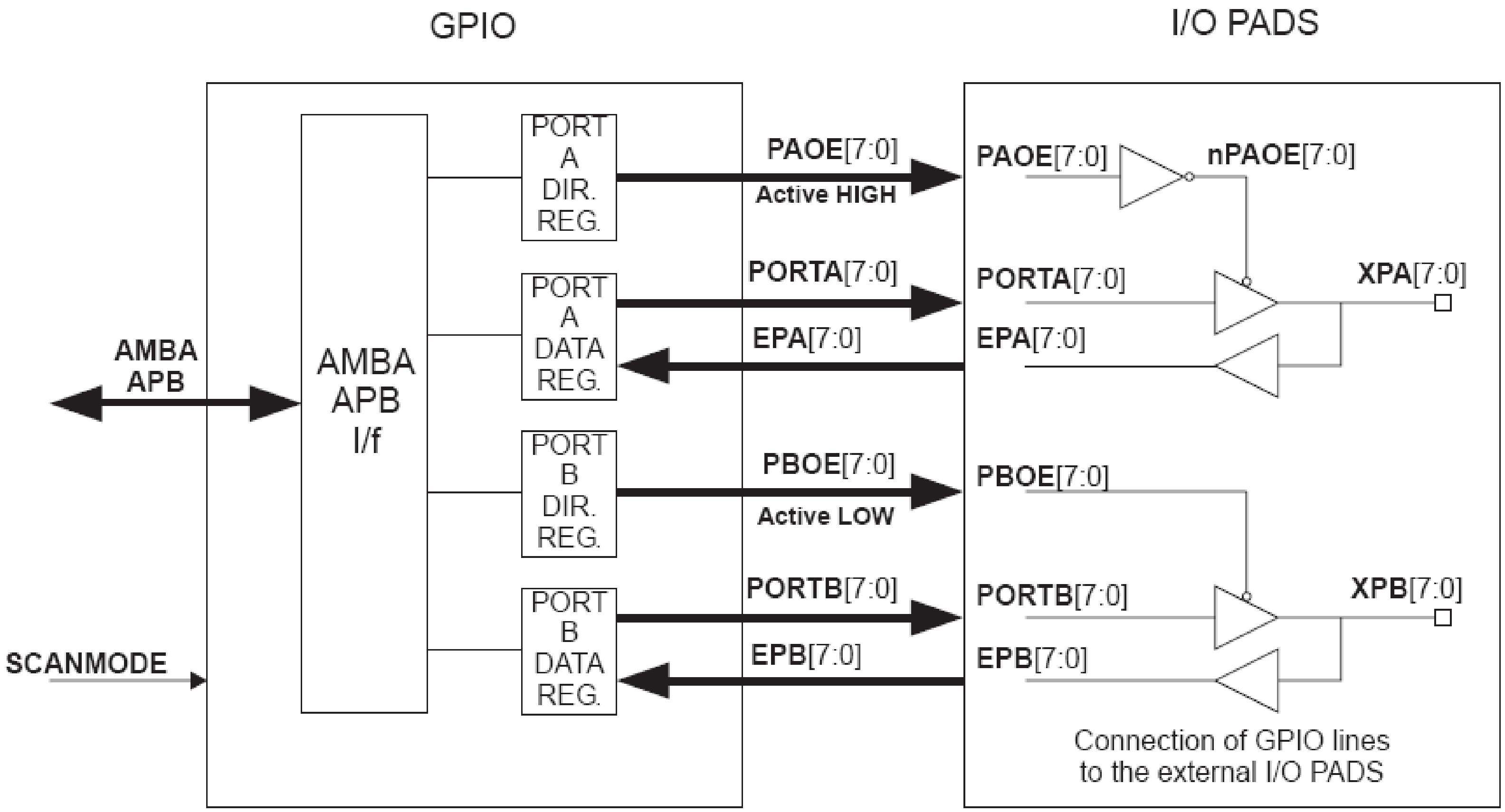
- Applications using FireWire include:
  - Disk drives, printers, scanners, cameras
- Capable of supporting a LAN similar to Ethernet
  - 64-bit address:
    - 10 bits for network ids, 1023 subnetworks
    - 6 bits for node ids, each subnetwork can have 63 nodes
    - 48 bits for memory address, each node can have 281 terabytes of distinct locations

GPIO

- GPIO: general-purpose input/output
- Very useful for
  - Debugging
  - Extend the I/O function of the system
  - Can be used for multiple purposes

- References
  - S&IP聯盟, “Lab 5: External I/O Control,” *SoC實驗教材*
  - *ARM PrimeCell General Purpose Input/Output (PL060) Technical Reference Manual (ARM DDI 0142B)*
  - D. Lampret and G. Djakovic, *GPIO IP Core Specification*, Opencores, http://www.opencores.org

The Block Diagram of GPIO

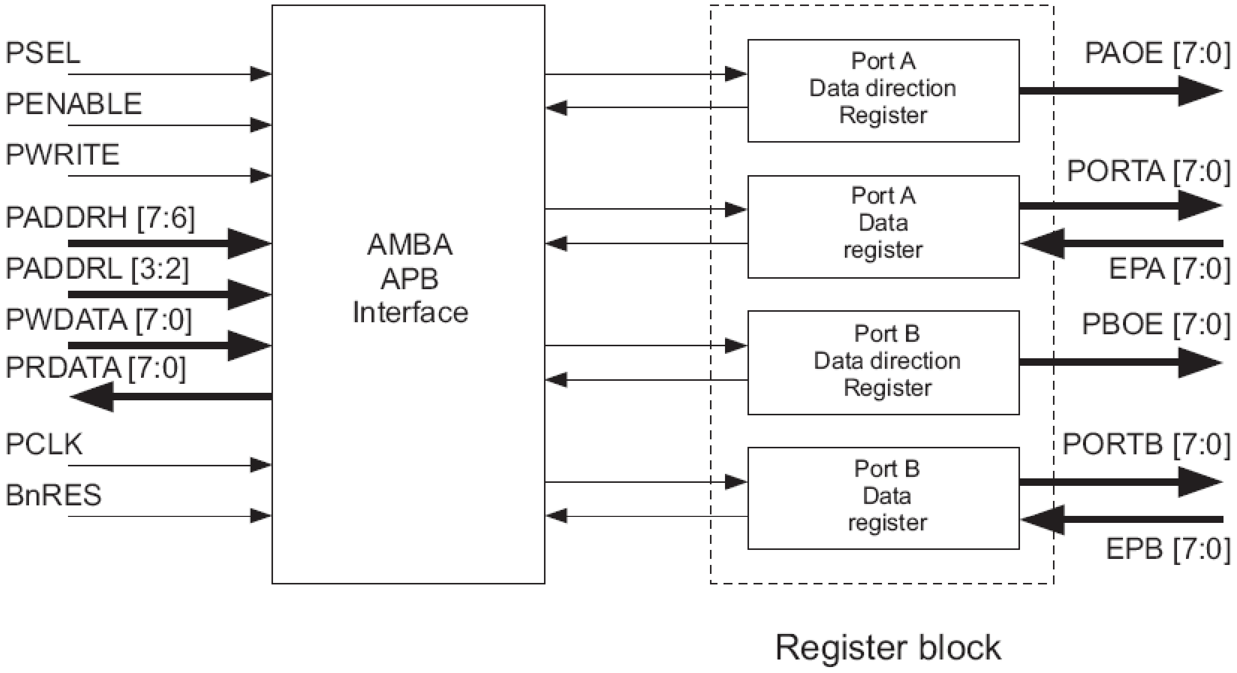


Functional Overview

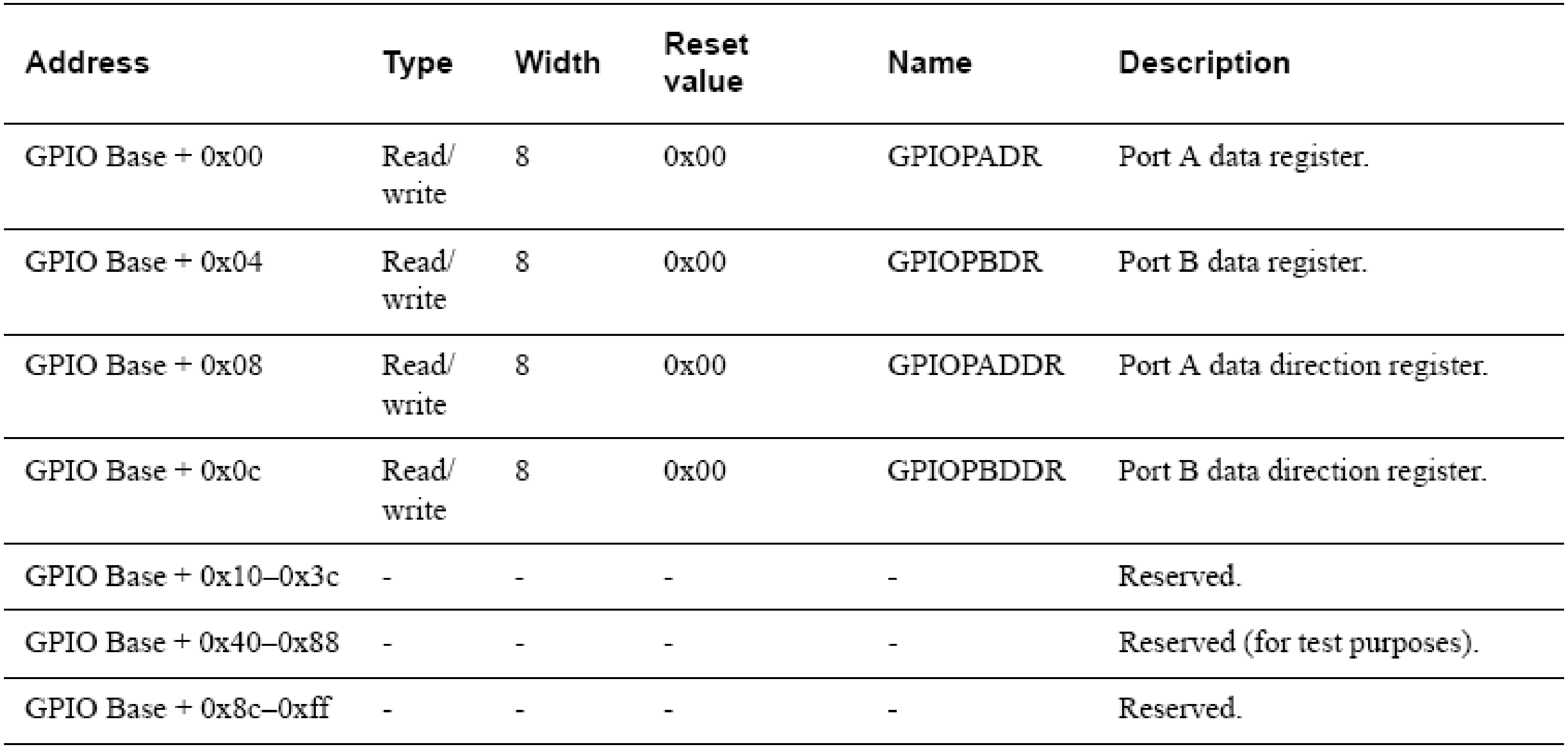
- Data direction register
  - □ 8 bits wide
  - □ Select whether each individual input/output pin is configured as an input or an output
- Data register
  - □ 8 bits wide
  - □ Used to read the value input on those pins are configured as input, and program the value output on the output pins

AMBA APB interface

- The AMBA APB group narrow-bus peripherals to avoid loading the system bus, and provides an interface using memory-mapped registers.



Memory Mapped Registers



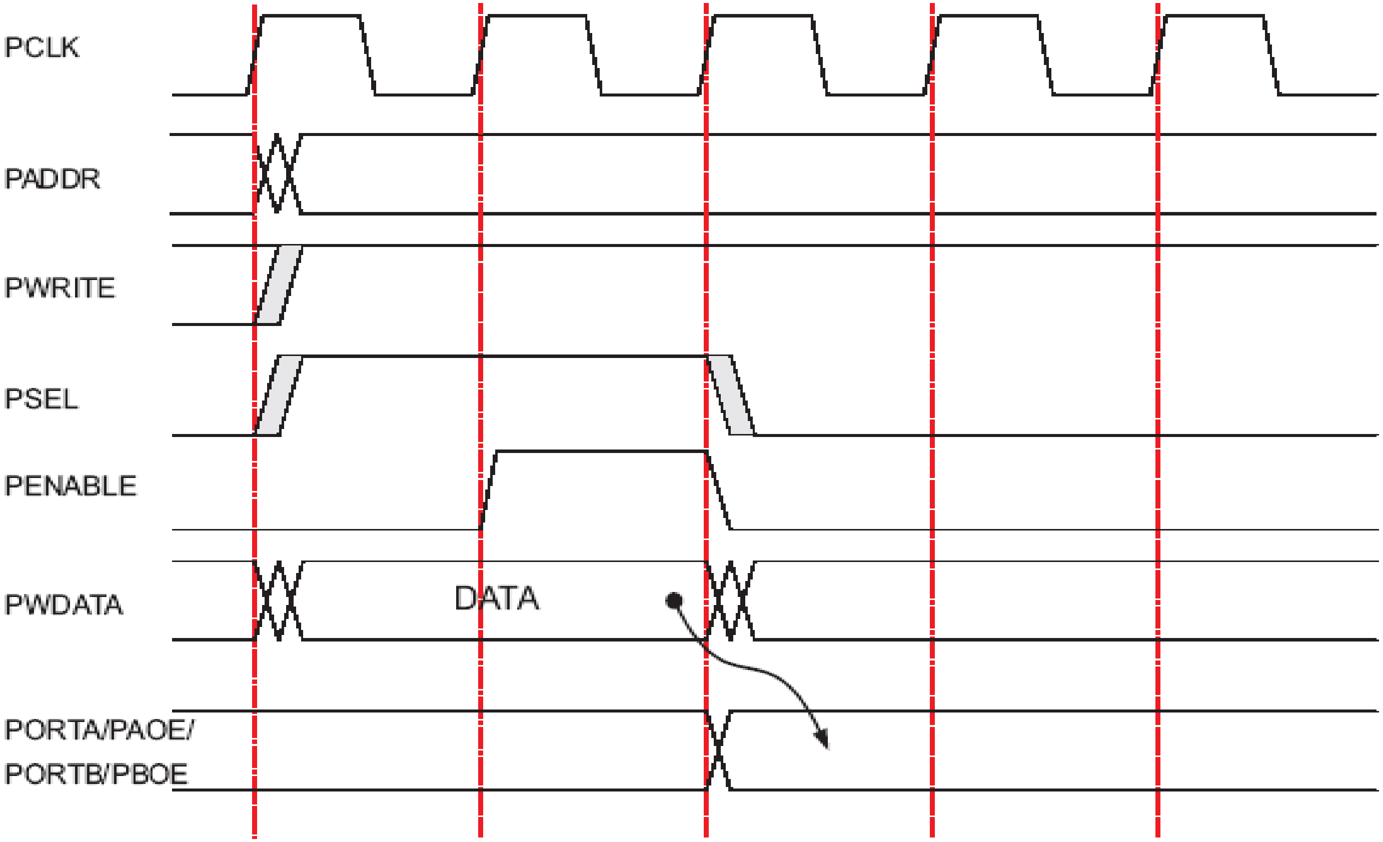
Address	Type	Width	Reset value	Name	Description
GPIO Base + 0x00	Read/ write	8	0x00	GPIOPADR	Port A data register.
GPIO Base + 0x04	Read/ write	8	0x00	GPIOPBDR	Port B data register.
GPIO Base + 0x08	Read/ write	8	0x00	GPIOPADDR	Port A data direction register.
GPIO Base + 0x0c	Read/ write	8	0x00	GPIOPBDDR	Port B data direction register.
GPIO Base + 0x10–0x3c	-	-	-	-	Reserved.
GPIO Base + 0x40–0x88	-	-	-	-	Reserved (for test purposes).
GPIO Base + 0x8c–0xff	-	-	-	-	Reserved.

Data Direction Register

- GPIOPADDR is the port A data direction register.
  - Bits set in GPIOPADDR will set the corresponding pin in PORT A to be an output.
  - Clearing a bit configures the pin to be an input.
- GPIOPBDDR is the port B data direction register.
  - Bits cleared in GPIOPBDDR will set the corresponding pin in PORT B to be an output.
  - Setting a bit configures the pin to be an input

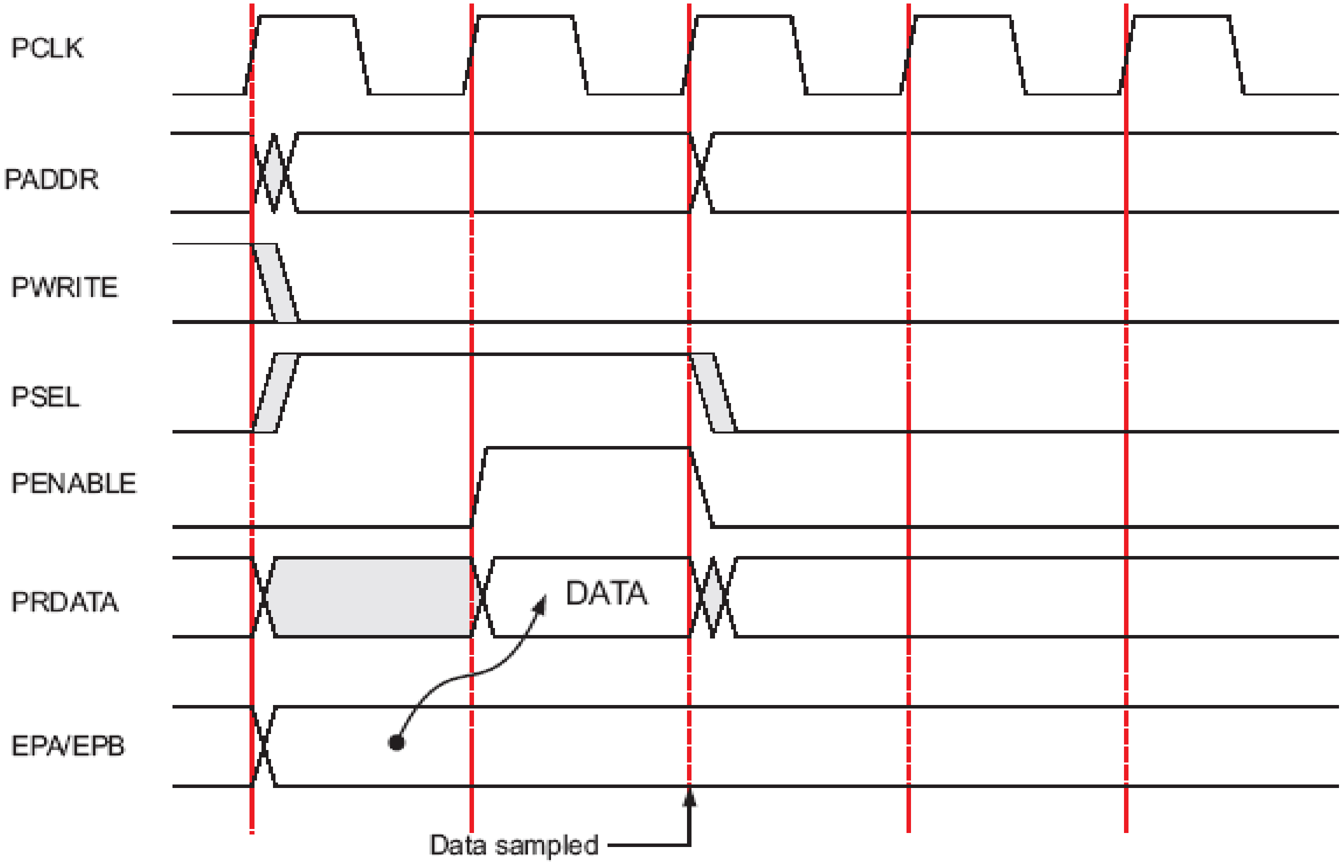
GPIO Operation

- Write Operation



GPIO Operation (cont.)

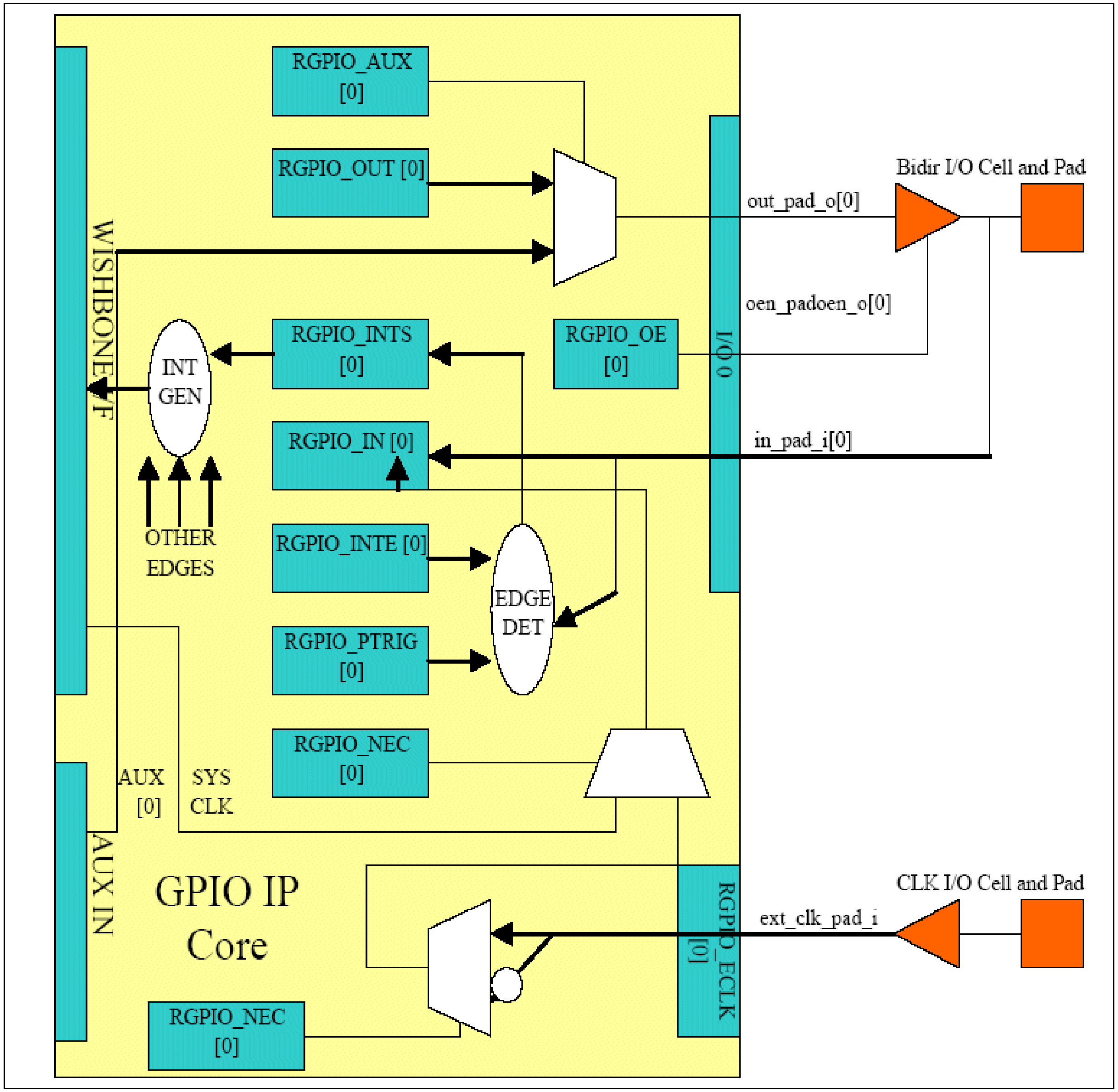
- Read Operation



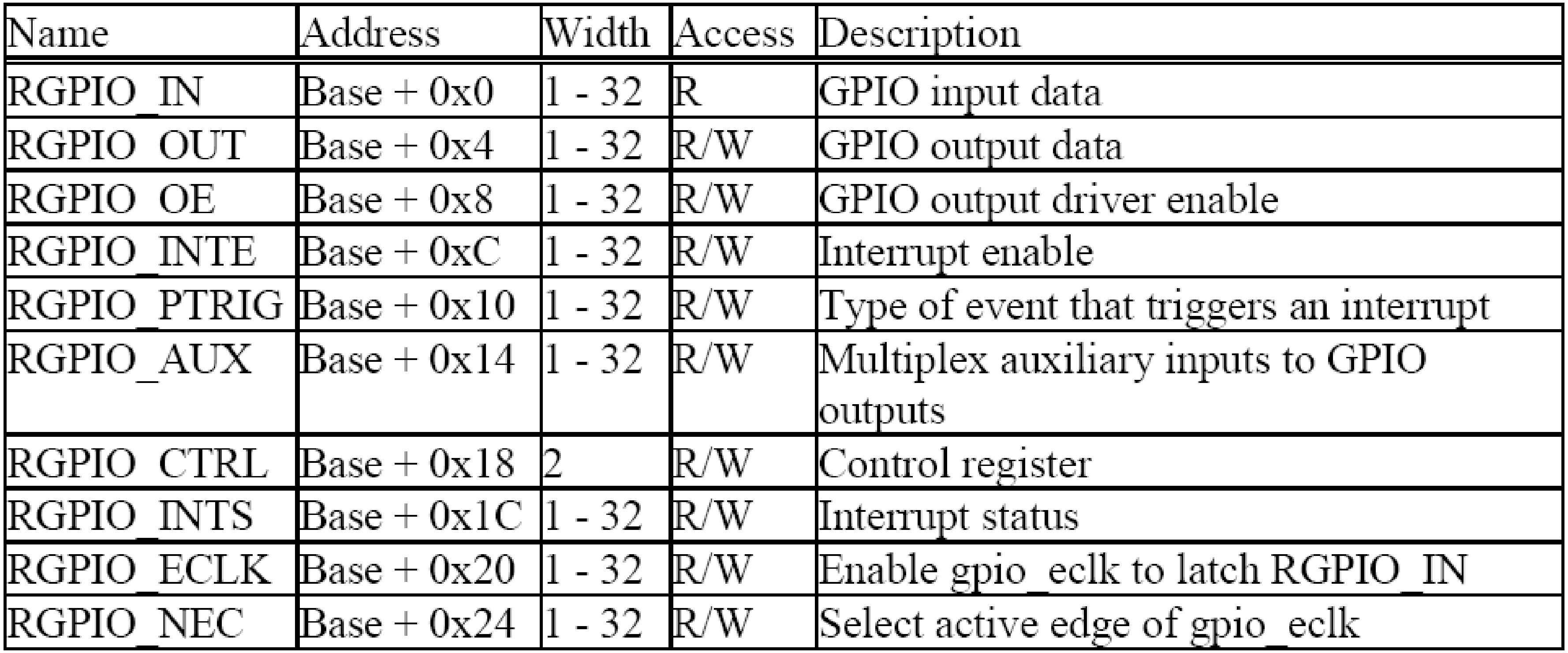
More Complex GPIO

- Inputs can cause an interrupt request to the CPU
- Inputs can be registered at raising edge of system clock or at user programmed edge of external clock
- Auxiliary inputs to GPIO core to bypass outputs from RGPIO_OUT register.
- Alternative input reference clock signal from external interface.
- …

More Complex GPIO



Memory Mapped Register



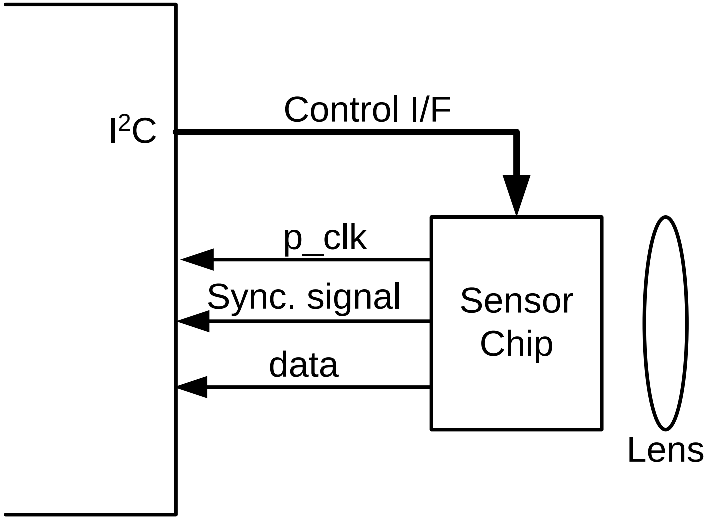
Name	Address	Width	Access	Description
RGPIO_IN	Base + 0x0	1 - 32	R	GPIO input data
RGPIO_OUT	Base + 0x4	1 - 32	R/W	GPIO output data
RGPIO_OE	Base + 0x8	1 - 32	R/W	GPIO output driver enable
RGPIO_INTE	Base + 0xC	1 - 32	R/W	Interrupt enable
RGPIO_PTRIG	Base + 0x10	1 - 32	R/W	Type of event that triggers an interrupt
RGPIO_AUX	Base + 0x14	1 - 32	R/W	Multiplex auxiliary inputs to GPIO outputs
RGPIO_CTRL	Base + 0x18	2	R/W	Control register
RGPIO_INTS	Base + 0x1C	1 - 32	R/W	Interrupt status
RGPIO_ECLK	Base + 0x20	1 - 32	R/W	Enable gpio_eclk to latch RGPIO_IN
RGPIO_NEC	Base + 0x24	1 - 32	R/W	Select active edge of gpio_eclk

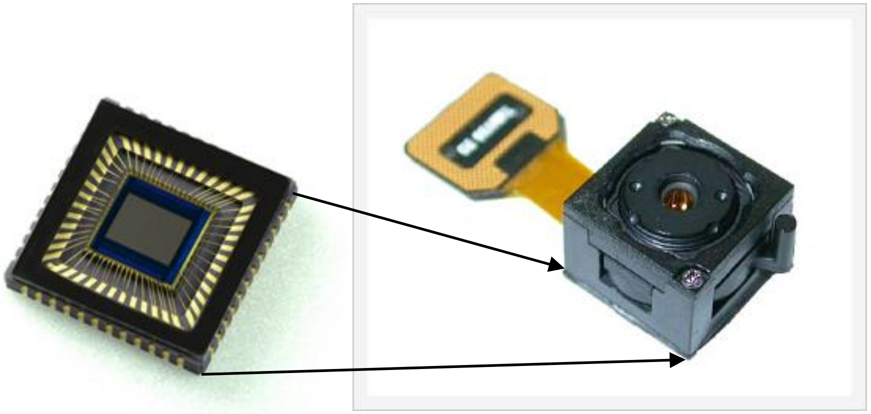
Important Peripherals in Multimedia SoC

- Image sensor I/F
- Display I/F
  - LCD I/F
  - NTSC I/F
- Audio I/F
  - I^2S
  - S/PDIF

Image Sensor I/F

- Image sensor is an important input device for multimedia systems
  - Capture image and video
- General model to control a sensor:





Functional Block Diagram

(Take OmniVision OV7640 as an example,
whose datasheet is from http://www.datasheet4u.com/)

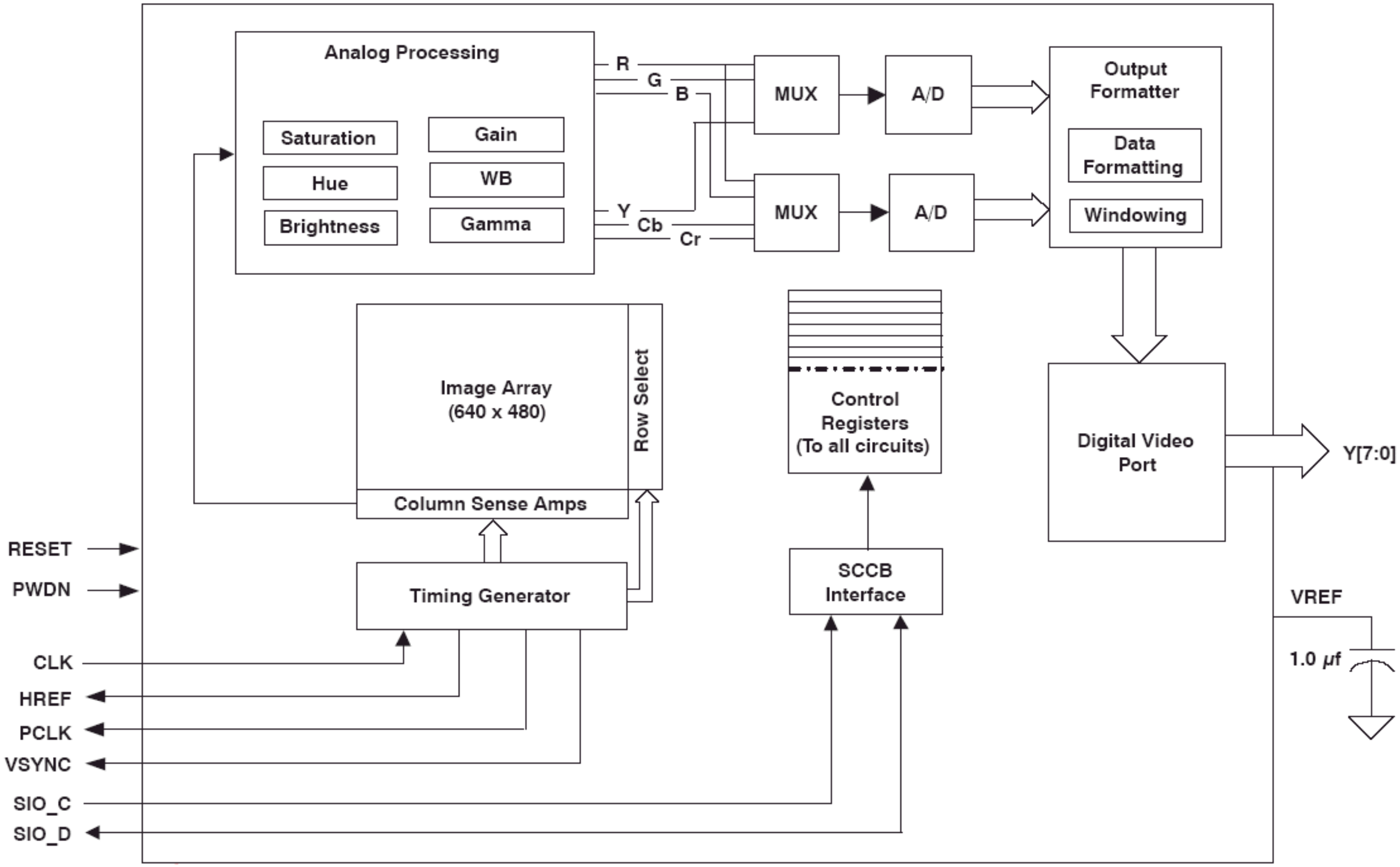
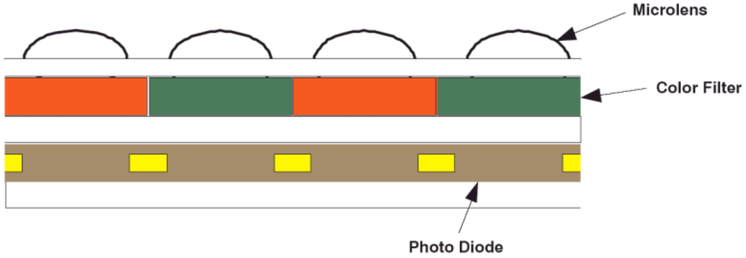
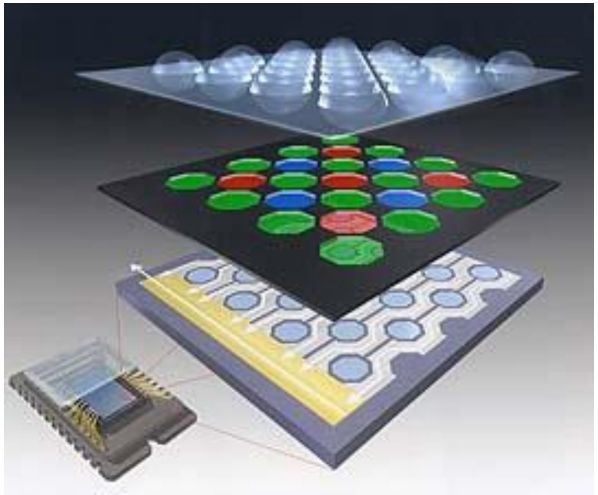
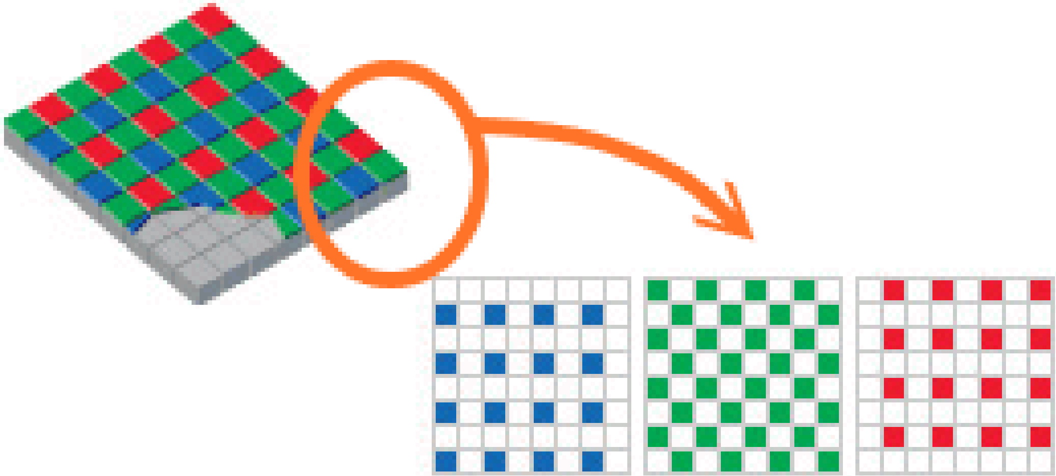


Image Sensor Array

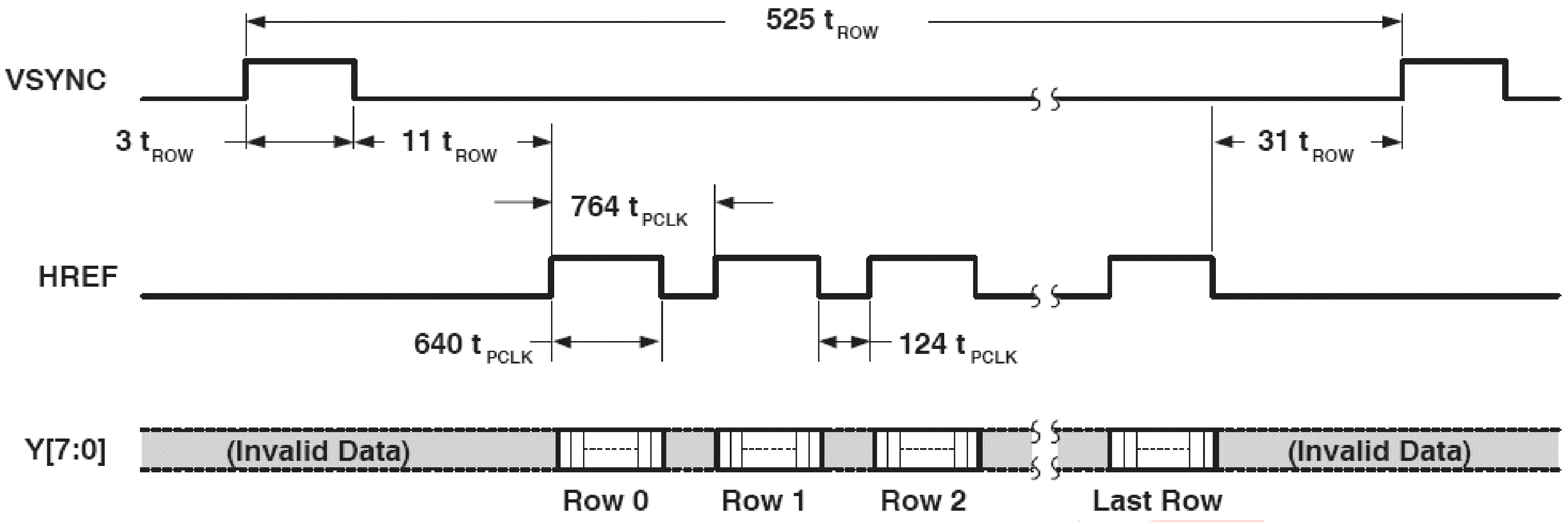
- Bayer pattern



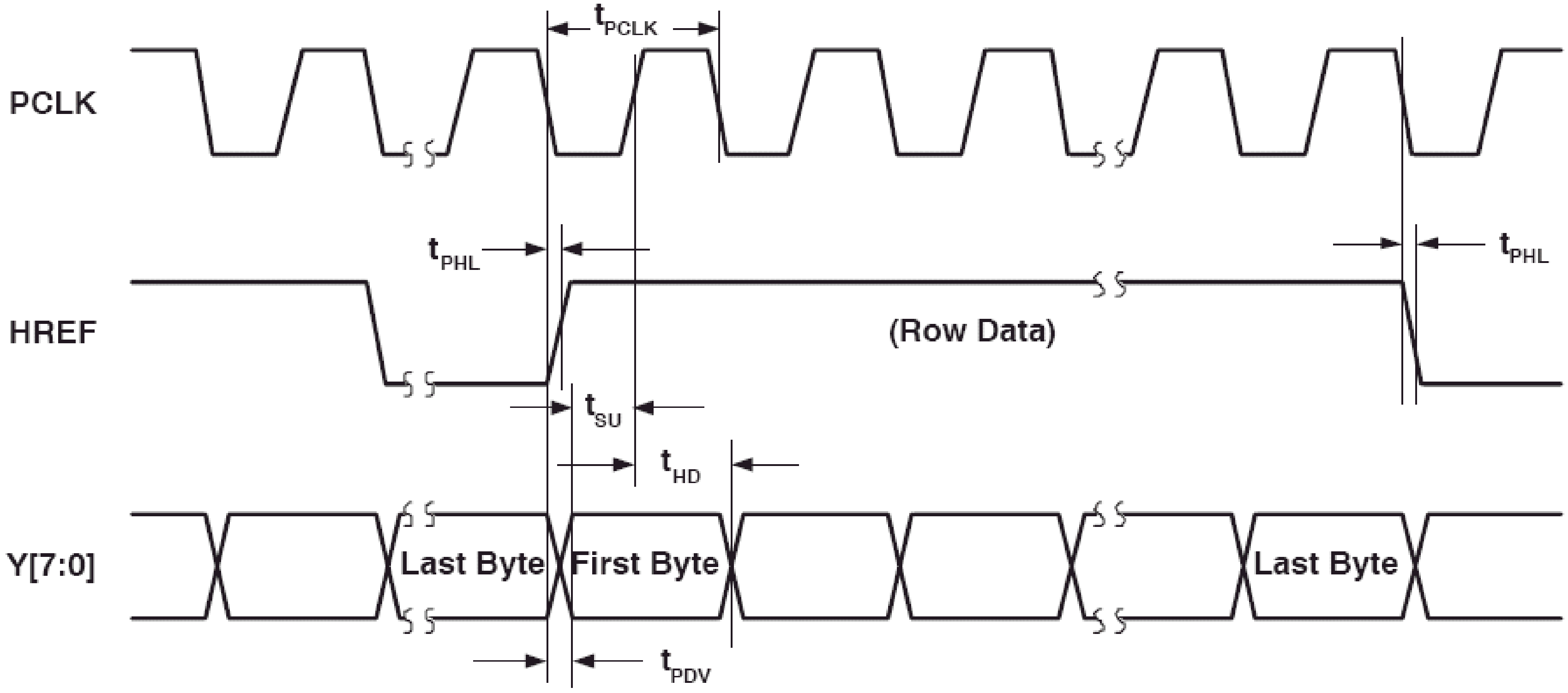




Typical Waveform -- Frame Timing



Typical Waveform -- Row Timing



Other I/F: MIPI (Mobile Industry Processor Interface) CSI (Camera Serial I/F)

LCD I/F

- LCD panel, LCD controller, and LCD module (LCM)

LCD Panel

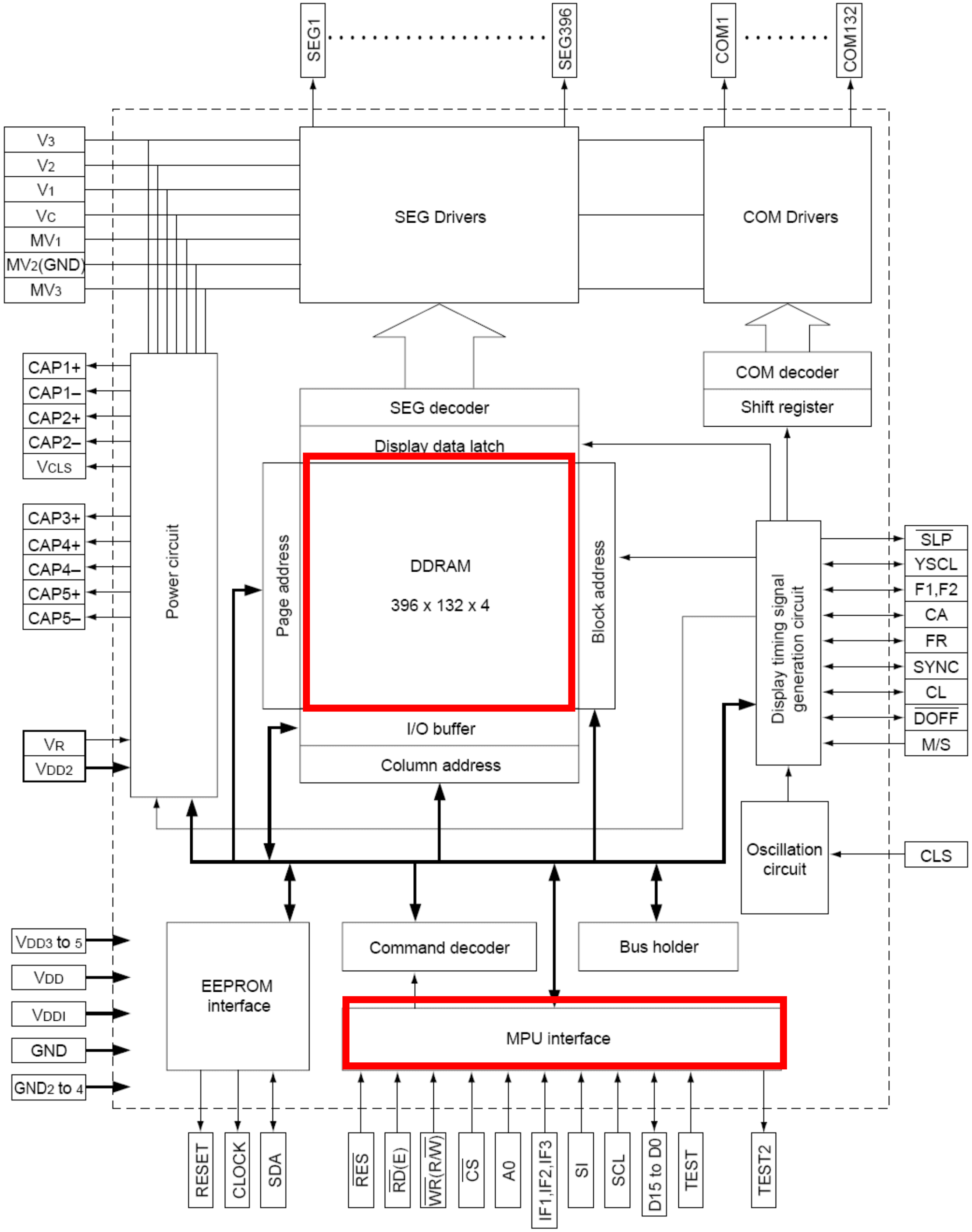
LCD Controller

LCD Module

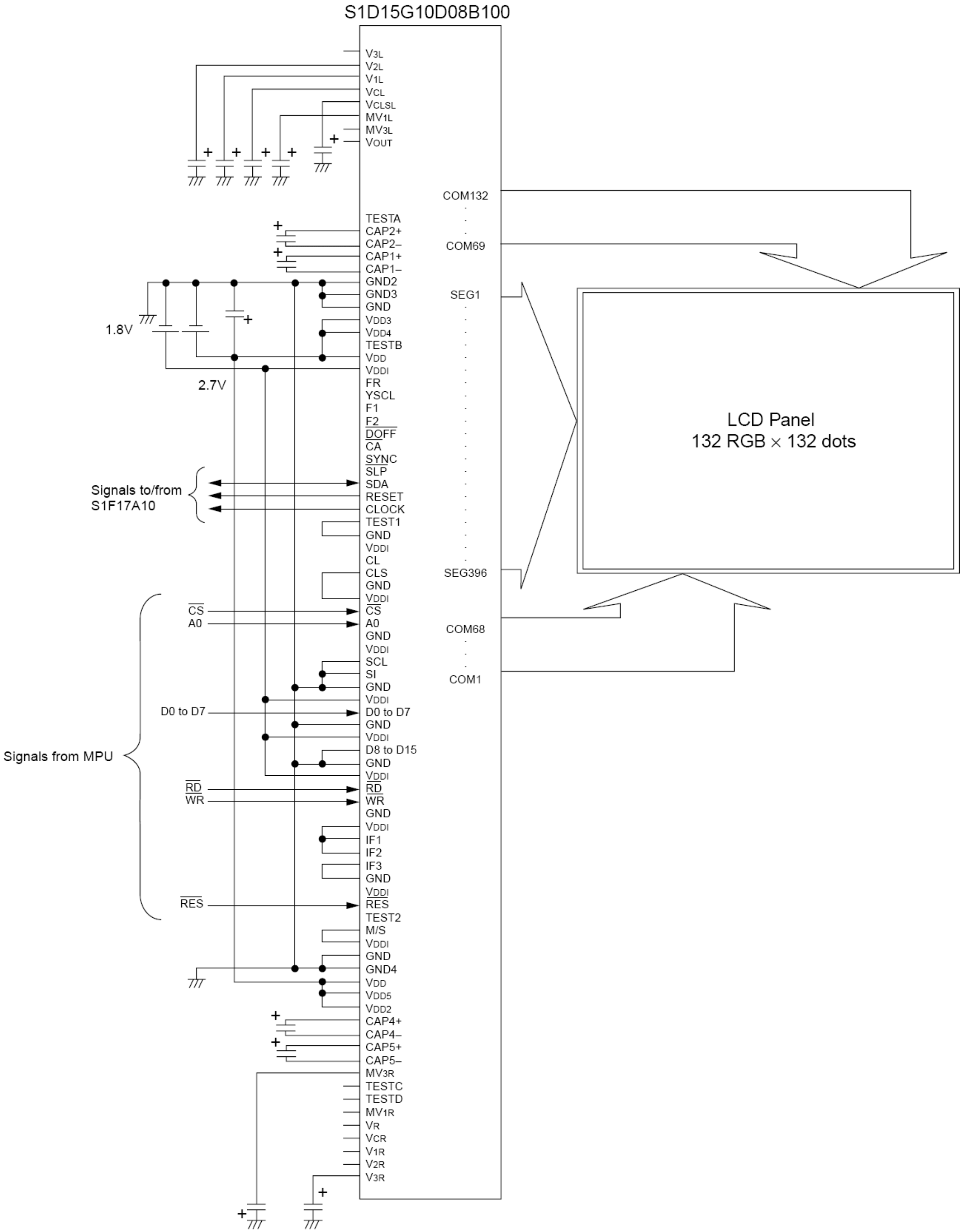
Functional Block Diagram

(Take EPSON S1D15G10 as an example, whose datasheet is from http://www.datasheet4u.com/)

- Support up to 132RGBx132 resolutions
- With display RAM and MPU I/F

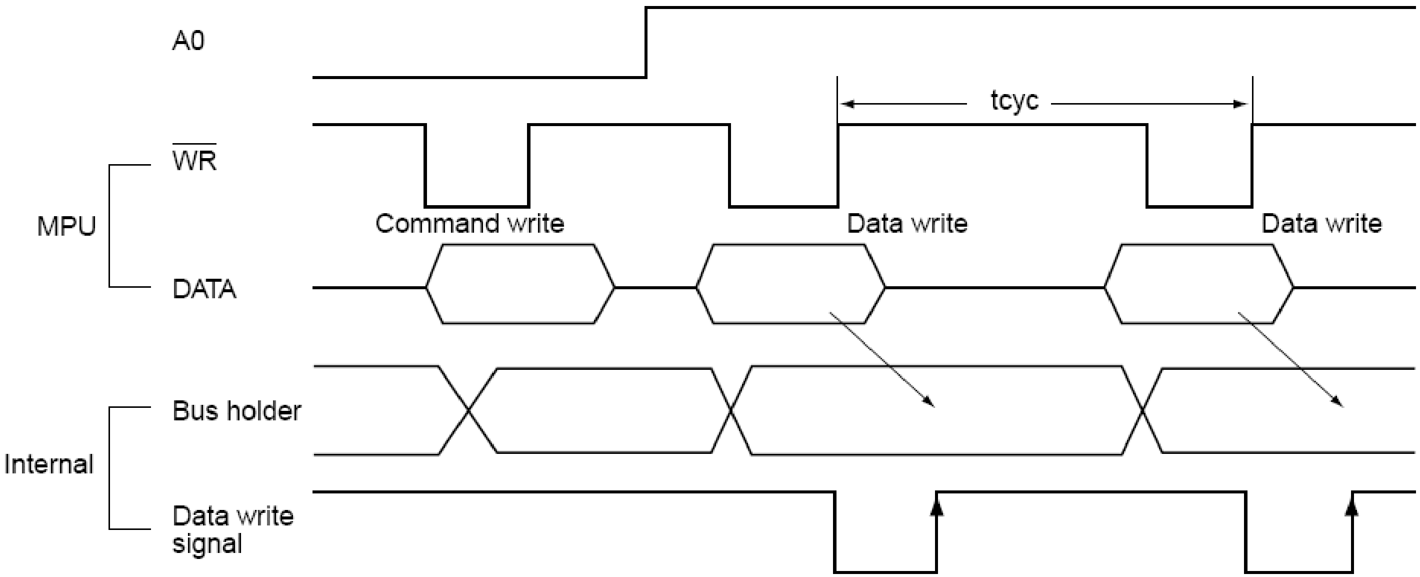


Connect to LCD Panel



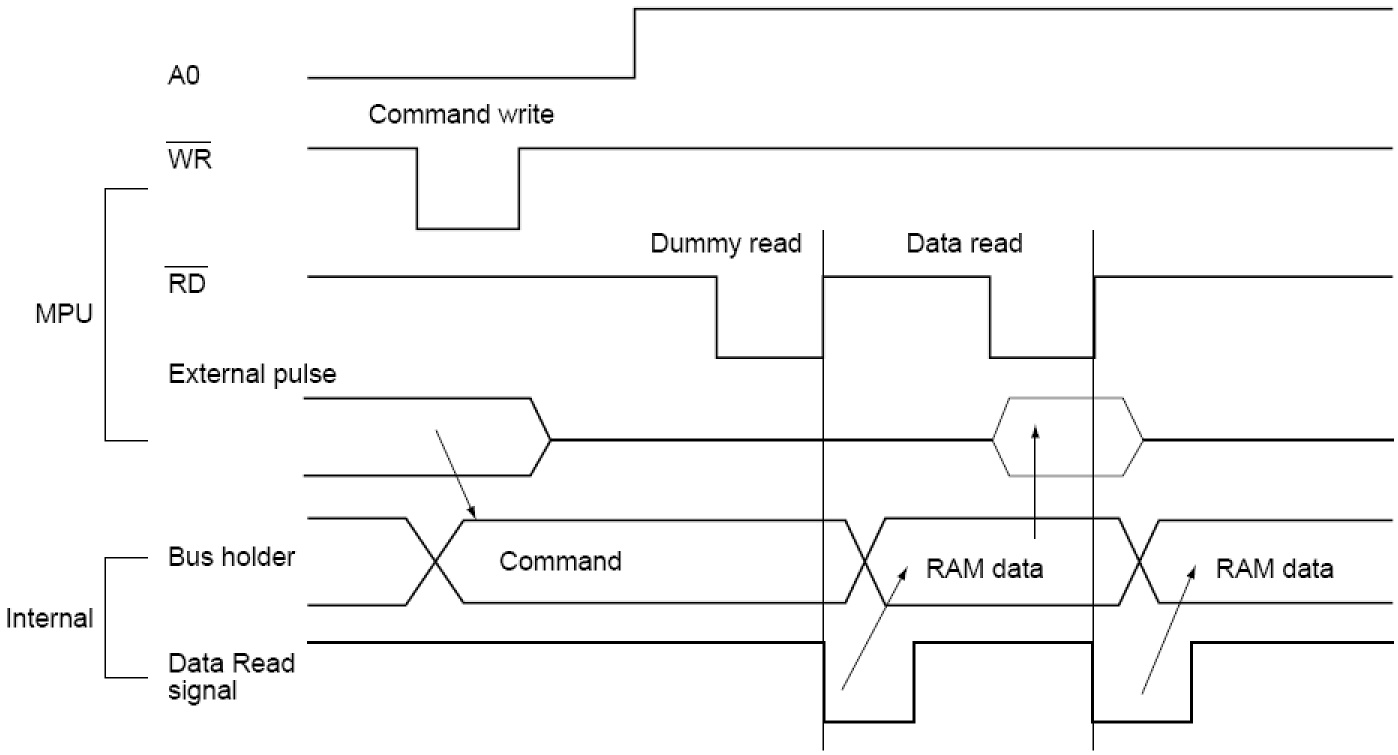
Timing Diagram

- I2C I/F can also be used
- Parallel I/F:
    - □ Write

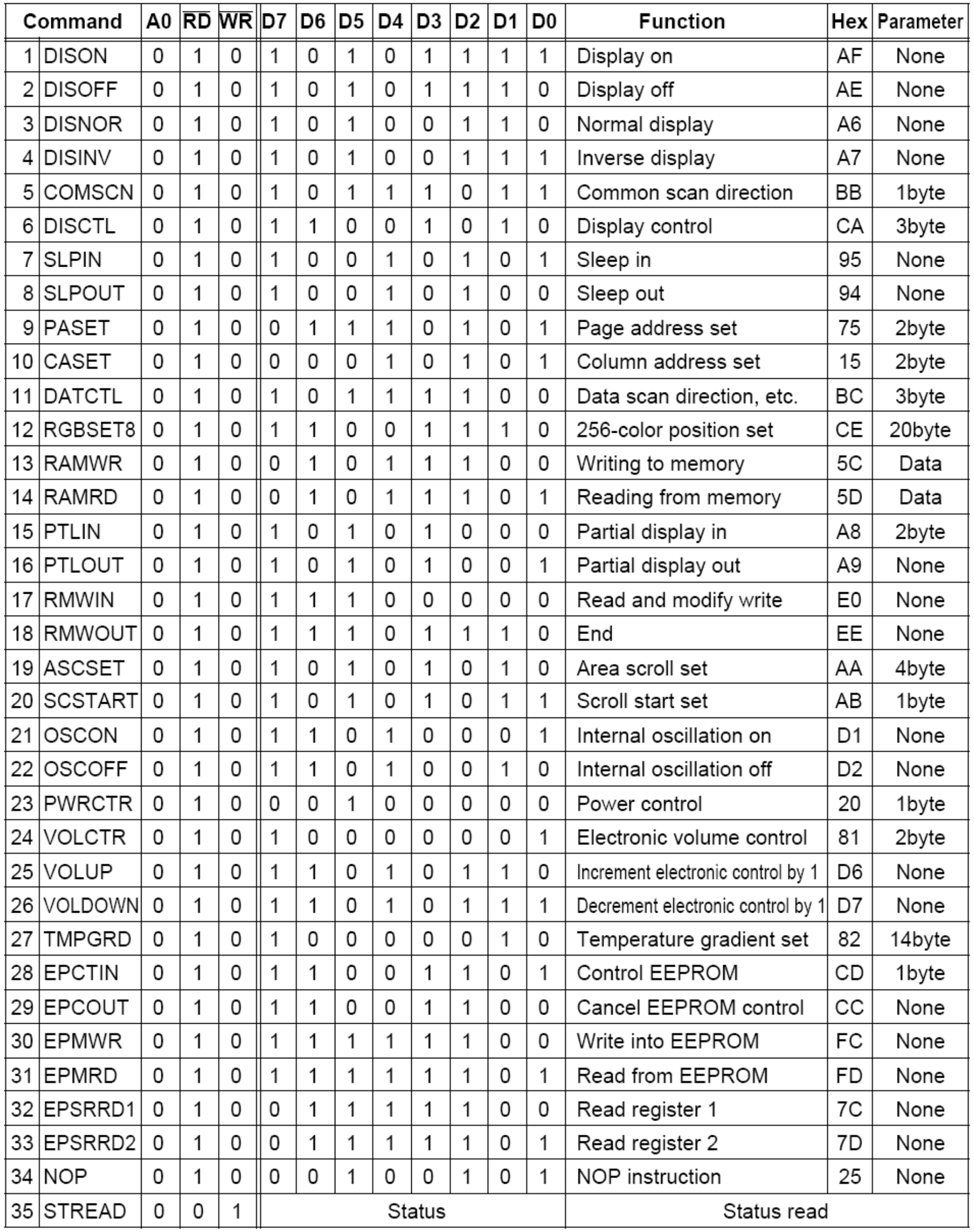


Timing Diagram

- Read

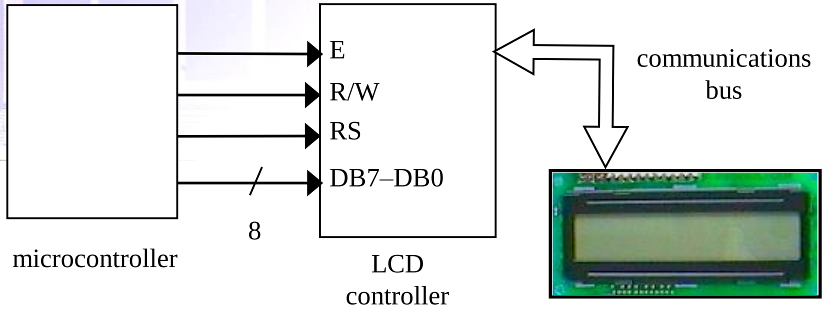


Commands



	Command	A0	$\overline{RD}$	$\overline{WR}$	D7	D6	D5	D4	D3	D2	D1	D0	Function	Hex	Parameter
1	DISON	0	1	0	1	0	1	0	1	1	1	1	Display on	AF	None
2	DISOFF	0	1	0	1	0	1	0	1	1	1	0	Display off	AE	None
3	DISNOR	0	1	0	1	0	1	0	0	1	1	0	Normal display	A6	None
4	DISINV	0	1	0	1	0	1	0	0	1	1	1	Inverse display	A7	None
5	COMSCN	0	1	0	1	0	1	1	1	0	1	1	Common scan direction	BB	1byte
6	DISCTL	0	1	0	1	1	0	0	1	0	1	0	Display control	CA	3byte
7	SLPIN	0	1	0	1	0	0	1	0	1	0	1	Sleep in	95	None
8	SLPOUT	0	1	0	1	0	0	1	0	1	0	0	Sleep out	94	None
9	PASET	0	1	0	0	1	1	1	0	1	0	1	Page address set	75	2byte
10	CASET	0	1	0	0	0	0	1	0	1	0	1	Column address set	15	2byte
11	DATCTL	0	1	0	1	0	1	1	1	1	0	0	Data scan direction, etc.	BC	3byte
12	RGBSET8	0	1	0	1	1	0	0	1	1	1	0	256-color position set	CE	20byte
13	RAMWR	0	1	0	0	1	0	1	1	1	0	0	Writing to memory	5C	Data
14	RAMRD	0	1	0	0	1	0	1	1	1	0	1	Reading from memory	5D	Data
15	PTLIN	0	1	0	1	0	1	0	1	0	0	0	Partial display in	A8	2byte
16	PTLOUT	0	1	0	1	0	1	0	1	0	0	1	Partial display out	A9	None
17	RMWIN	0	1	0	1	1	1	0	0	0	0	0	Read and modify write	E0	None
18	RMWOUT	0	1	0	1	1	1	0	1	1	1	0	End	EE	None
19	ASCSET	0	1	0	1	0	1	0	1	0	1	0	Area scroll set	AA	4byte
20	SCSTART	0	1	0	1	0	1	0	1	0	1	1	Scroll start set	AB	1byte
21	OSCON	0	1	0	1	1	0	1	0	0	0	1	Internal oscillation on	D1	None
22	OSCOFF	0	1	0	1	1	0	1	0	0	1	0	Internal oscillation off	D2	None
23	PWRCTR	0	1	0	0	0	1	0	0	0	0	0	Power control	20	1byte
24	VOLCTR	0	1	0	1	0	0	0	0	0	0	1	Electronic volume control	81	2byte
25	VOLUP	0	1	0	1	1	0	1	0	1	1	0	Increment electronic control by 1	D6	None
26	VOLDOWN	0	1	0	1	1	0	1	0	1	1	1	Decrement electronic control by 1	D7	None
27	TMPGRD	0	1	0	1	0	0	0	0	0	1	0	Temperature gradient set	82	14byte
28	EPCTIN	0	1	0	1	1	0	0	1	1	0	1	Control EEPROM	CD	1byte
29	EPCOUT	0	1	0	1	1	0	0	1	1	0	0	Cancel EEPROM control	CC	None
30	EPMWR	0	1	0	1	1	1	1	1	1	0	0	Write into EEPROM	FC	None
31	EPMRD	0	1	0	1	1	1	1	1	1	0	1	Read from EEPROM	FD	None
32	EPSRRD1	0	1	0	0	1	1	1	1	1	0	0	Read register 1	7C	None
33	EPSRRD2	0	1	0	0	1	1	1	1	1	0	1	Read register 2	7D	None
34	NOP	0	1	0	0	0	1	0	0	1	0	1	NOP instruction	25	None
35	STREAD	0	0	1	Status								Status read		

LCD Controller

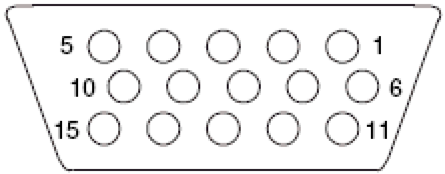


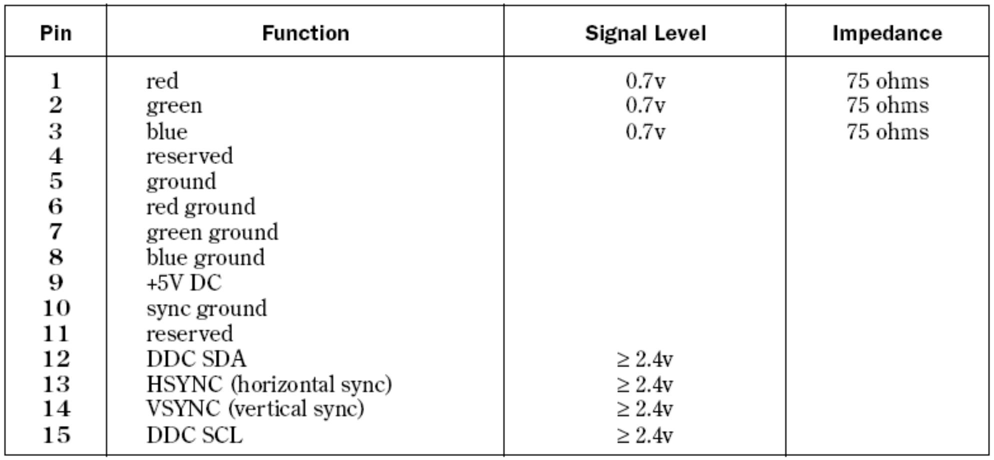
```
void WriteChar(char c){

  RS = 1;                          /* indicate data being sent */
  DATA_BUS = c;                    /* send data to LCD */
  EnableLCD(45);                   /* toggle the LCD with appropriate delay */
}
```

CODES	
I/D = 1 cursor moves left	DL = 1 8-bit
I/D = 0 cursor moves right	DL = 0 4-bit
S = 1 with display shift	N = 1 2 rows
S/C =1 display shift	N = 0 1 row
S/C = 0 cursor movement	F = 1 5x10 dots
R/L = 1 shift to right	F = 0 5x7 dots
R/L = 0 shift to left	

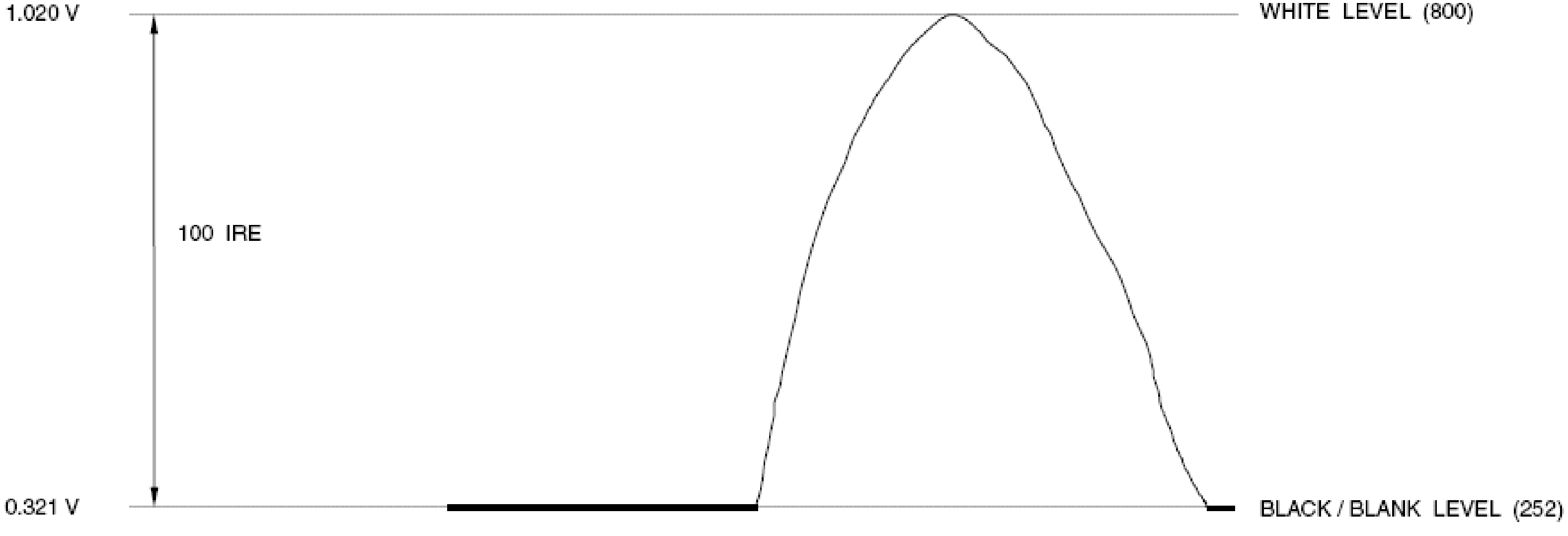
RS	R/W	DB$_7$	DB$_6$	DB$_5$	DB$_4$	DB$_3$	DB$_2$	DB$_1$	DB$_0$	Description
0	0	0	0	0	0	0	0	0	1	Clears all display, return cursor home
0	0	0	0	0	0	0	0	1	*	Returns cursor home
0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and/or specifies not to shift display
0	0	0	0	0	0	1	D	C	B	ON/OFF of all display(D), cursor ON/OFF (C), and blink position (B)
0	0	0	0	0	1	S/C	R/L	*	*	Move cursor and shifts display
0	0	0	0	1	DL	N	F	*	*	Sets interface data length, number of display lines, and character font
1	0	WRITE DATA								Writes Data

VGA I/F





Pin	Function	Signal Level	Impedance
1	red	0.7v	75 ohms
2	green	0.7v	75 ohms
3	blue	0.7v	75 ohms
4	reserved		
5	ground		
6	red ground		
7	green ground		
8	blue ground		
9	+5V DC		
10	sync ground		
11	reserved		
12	DDC SDA	≥ 2.4v	
13	HSYNC (horizontal sync)	≥ 2.4v	
14	VSYNC (vertical sync)	≥ 2.4v	
15	DDC SCL	≥ 2.4v	

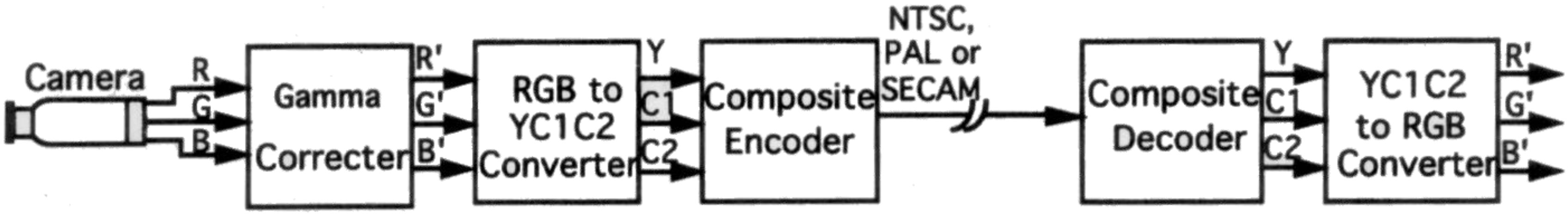


GREEN, BLUE, OR RED CHANNEL, NO SYNC PRESENT

Component Color Video System

Composite System

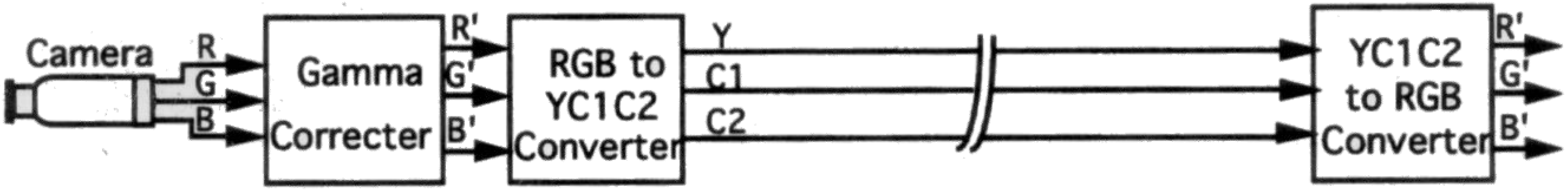
In order to transmit or broadcast the analog video over a signal channel, there is a need for composite color system that combine the three color components into a single signal



(a)

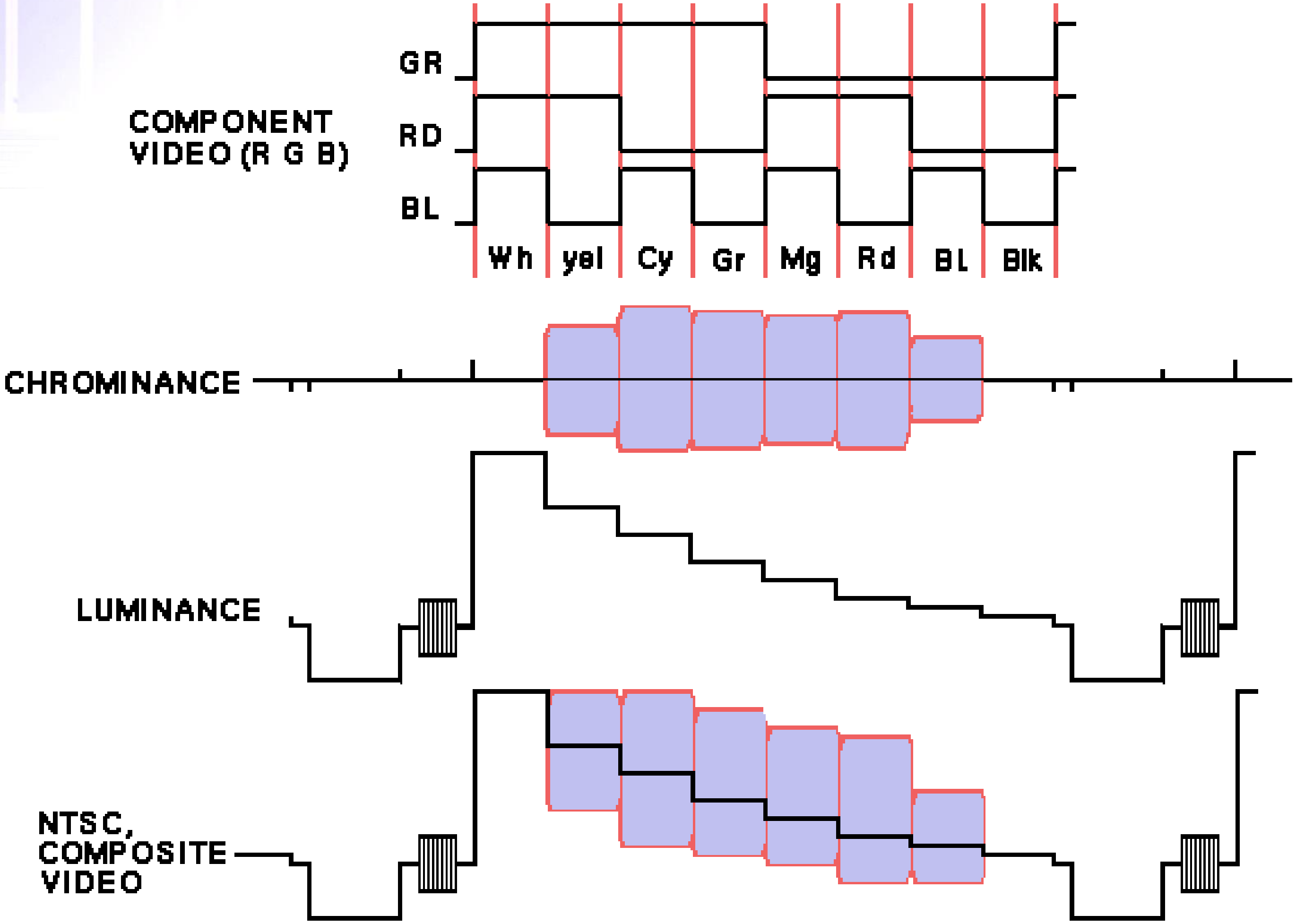
Component System

In color video system, the devices must deliver three components to control the light sources of display.



(b)

NTSC Composite System

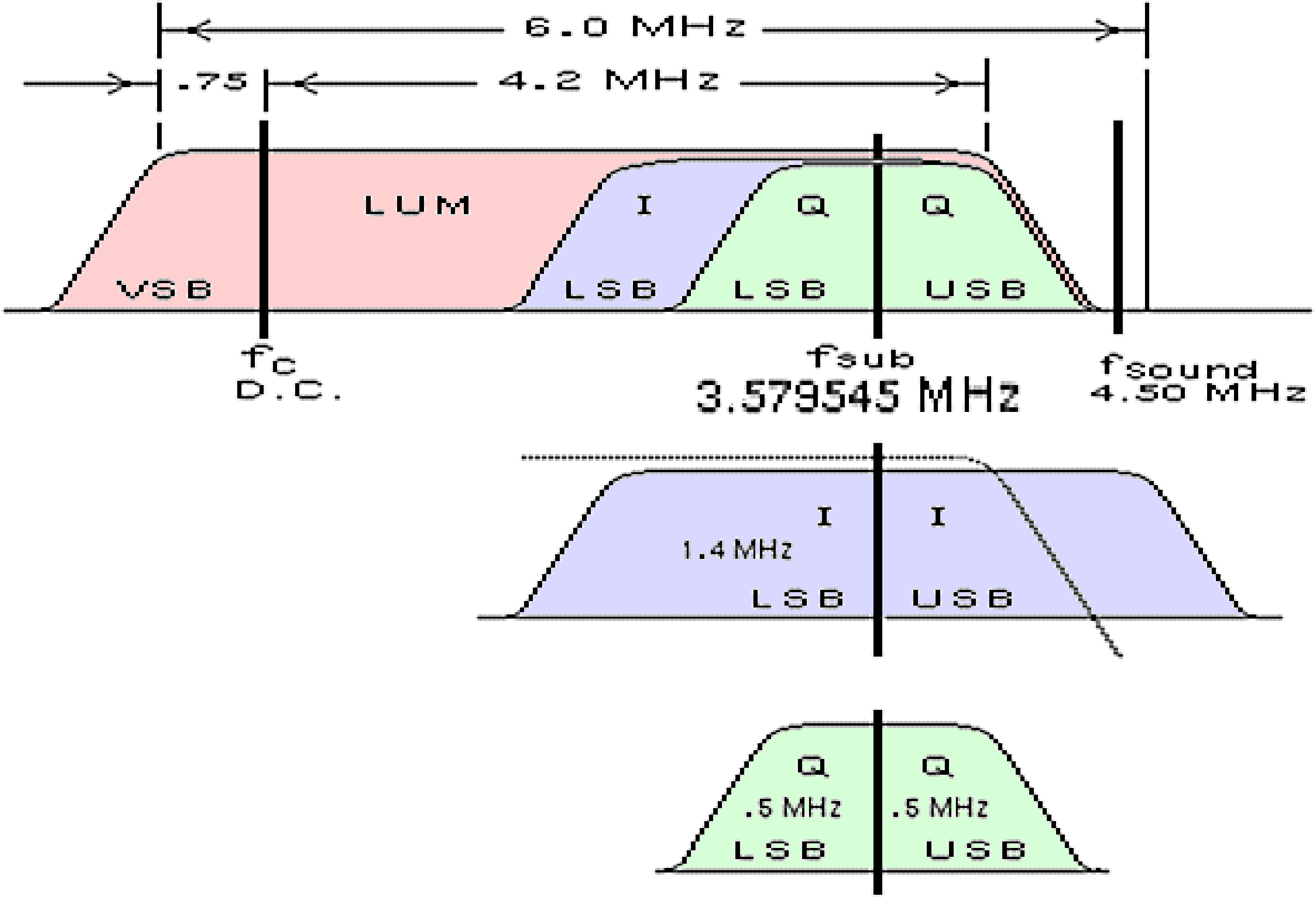


CCIR 601 4:2:2 FORMAT

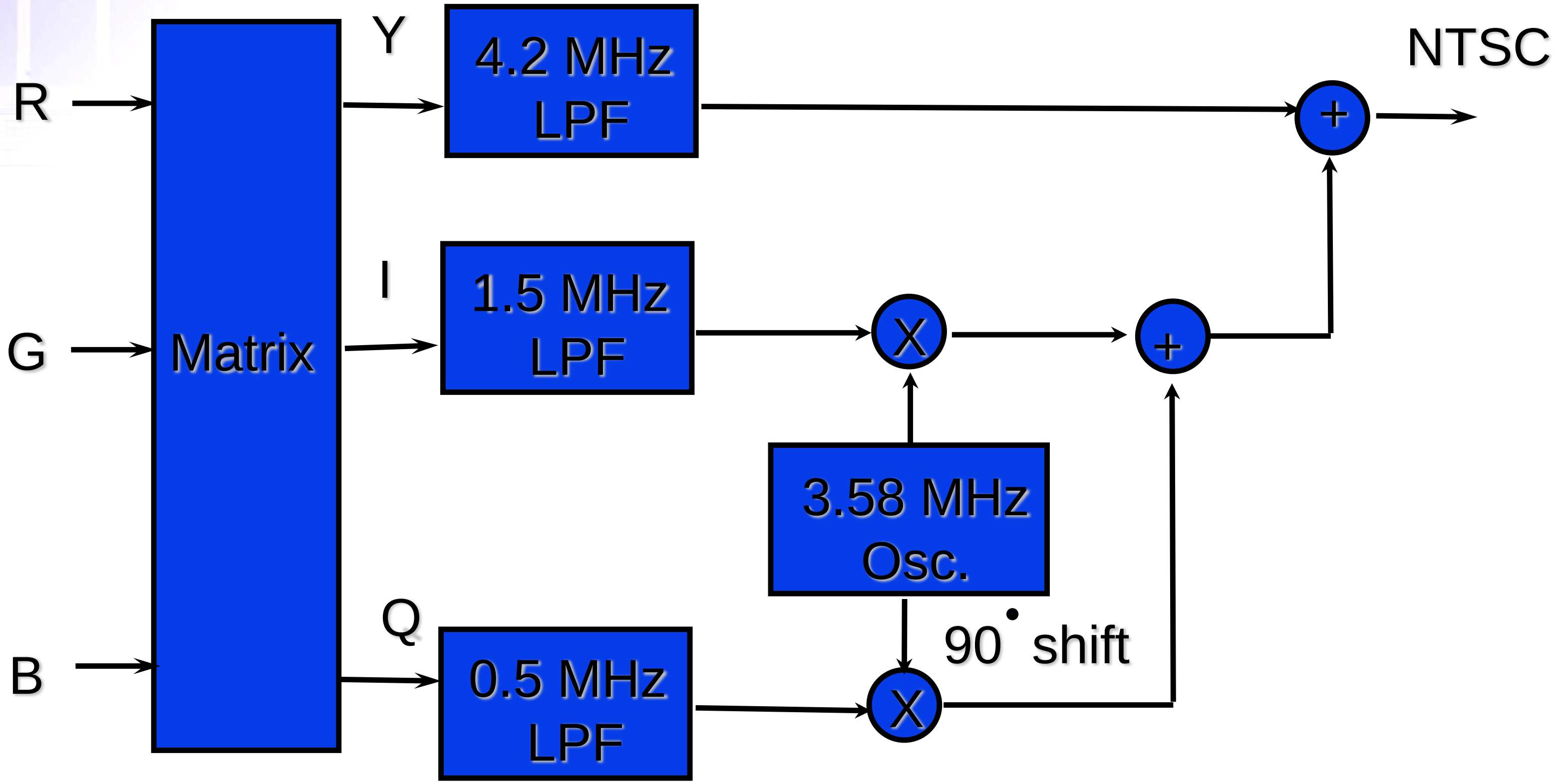
	NTSC	PAL
Luma sampling frequency	13.5 MHz	13.5 MHz
Chroma sampling frequency	6.75 MHz	6.75 MHz
Frames/sec	30	25
Luma #active samples/line	720	720
Chroma #active samples/line	360	360
Active #lines/frame	480	576
Sample resolution	8 bits	8 bits
Data rate	166 Mb/s	166 Mb/s
Scan line	525	625

30x720x480x8x2 = 25x720x576x8x2 = 166 Mb/s

NTSC Video Channel



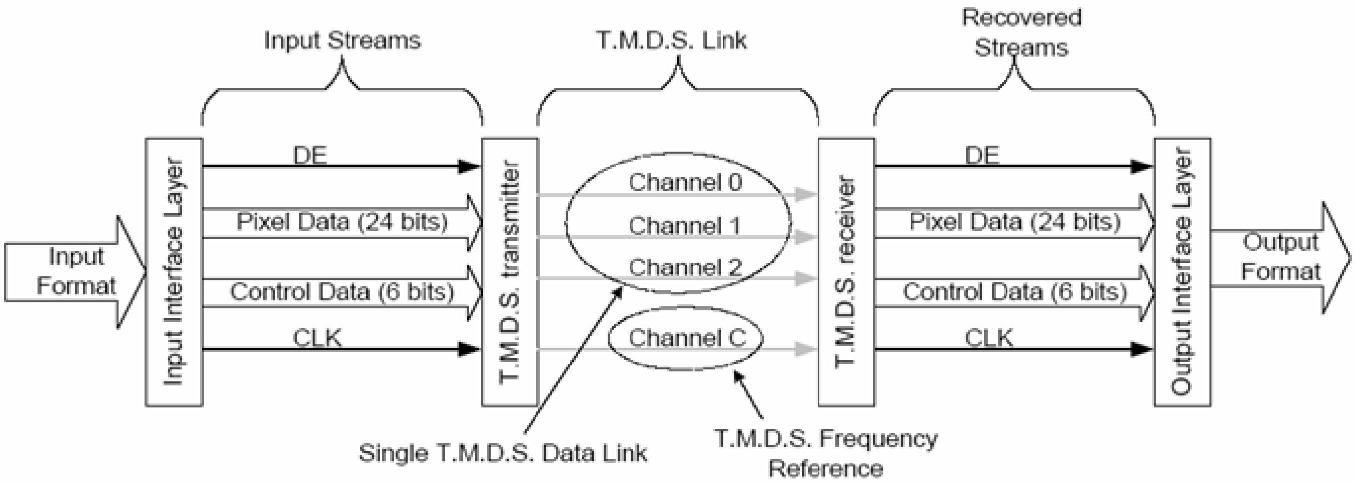
NTSC COMPOSITE VIDEO



- Artifact of the NTSC composite video due to imperfection of luminance-chrominance separation

Digital I/F

- Based on transition-minimized differential signaling (TMDS) protocol
  - □ 8 bits of video data are converted to a 10-bit transition-minimized, DC balanced value, which is then **serialized**
  - □ **It is a serial I/F!!**.

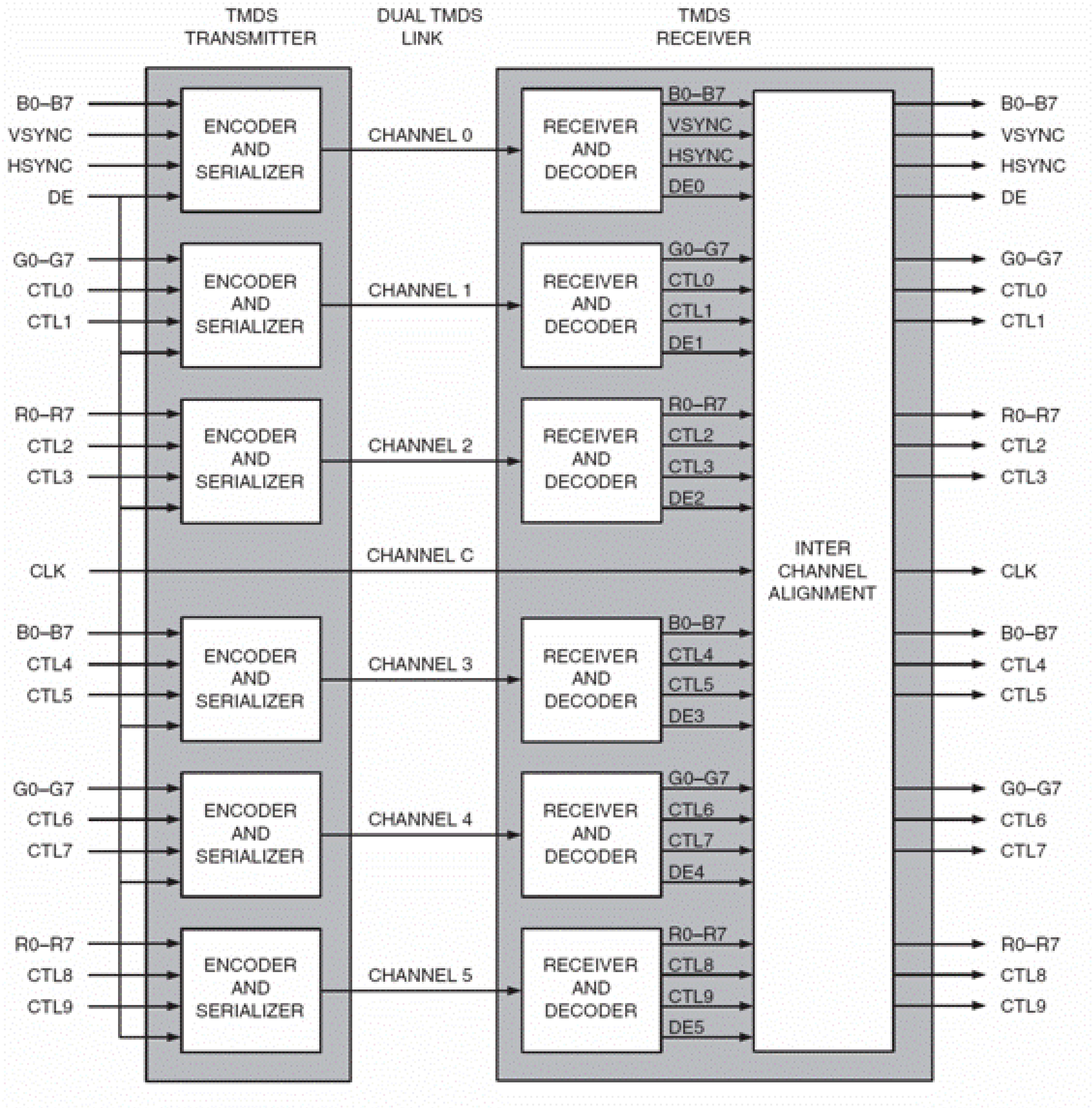


Clocking

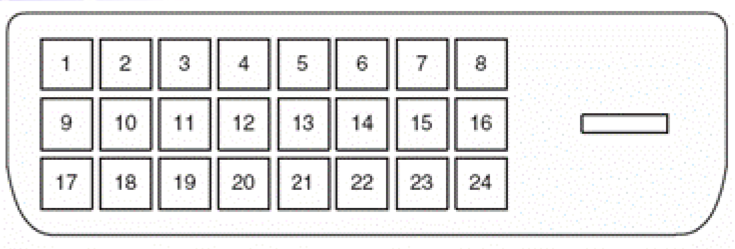
- The TMDS clock channel carries a **character-rate frequency reference** from which **the receiver produces a bit-rate sample clock** for the incoming serial stream.
- Due to the high pair-to-pair skew that must be tolerated, **the phase of the derived sample clock must be adjusted individually** for each data channel.

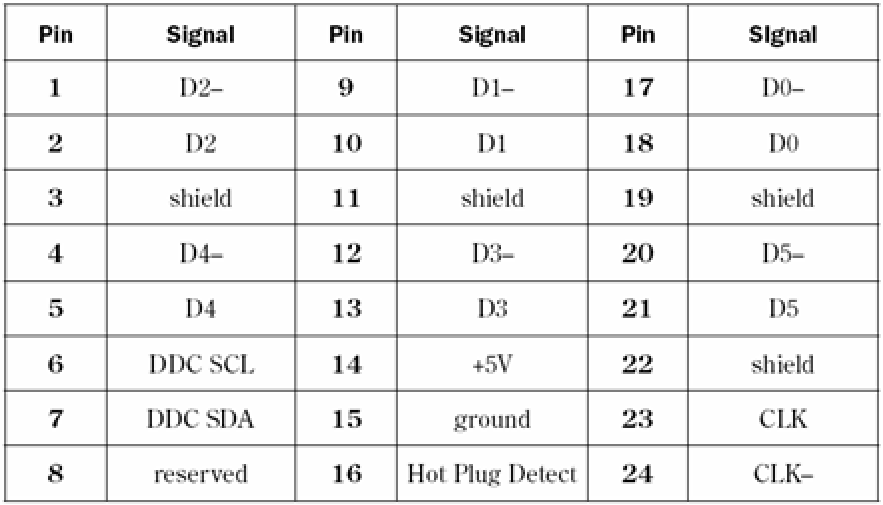
DVI I/F

- Dual channel: to double the bandwidth
- One is for odd pixels, and the other is for even pixels
- The first pixel is 1, odd pixel

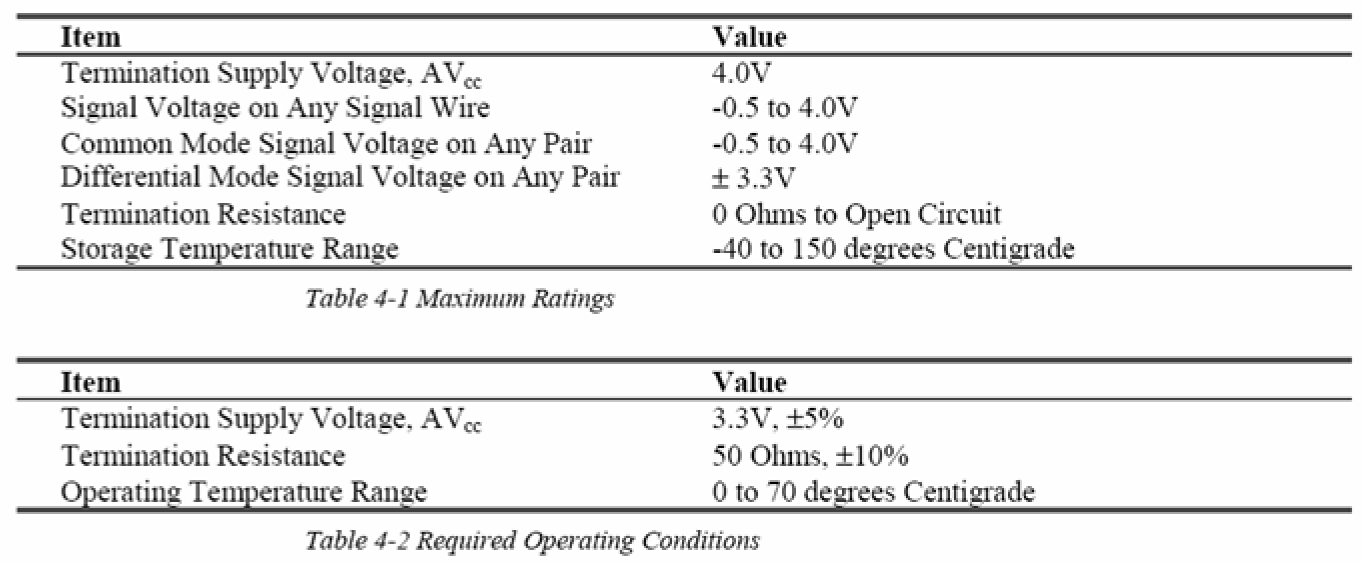


DVI I/F





Pin	Signal	Pin	Signal	Pin	Signal
1	D2–	9	D1–	17	D0–
2	D2	10	D1	18	D0
3	shield	11	shield	19	shield
4	D4–	12	D3–	20	D5–
5	D4	13	D3	21	D5
6	DDC SCL	14	+5V	22	shield
7	DDC SDA	15	ground	23	CLK
8	reserved	16	Hot Plug Detect	24	CLK–



Item	Value
Termination Supply Voltage, AV_{cc}	4.0V
Signal Voltage on Any Signal Wire	-0.5 to 4.0V
Common Mode Signal Voltage on Any Pair	-0.5 to 4.0V
Differential Mode Signal Voltage on Any Pair	± 3.3V
Termination Resistance	0 Ohms to Open Circuit
Storage Temperature Range	-40 to 150 degrees Centigrade

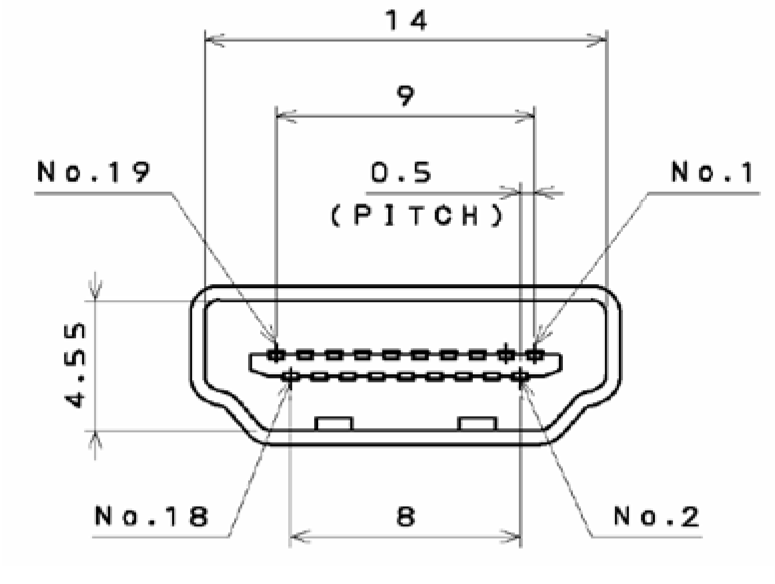
Table 4-1 Maximum Ratings

Item	Value
Termination Supply Voltage, AV_{cc}	3.3V, ±5%
Termination Resistance	50 Ohms, ±10%
Operating Temperature Range	0 to 70 degrees Centigrade

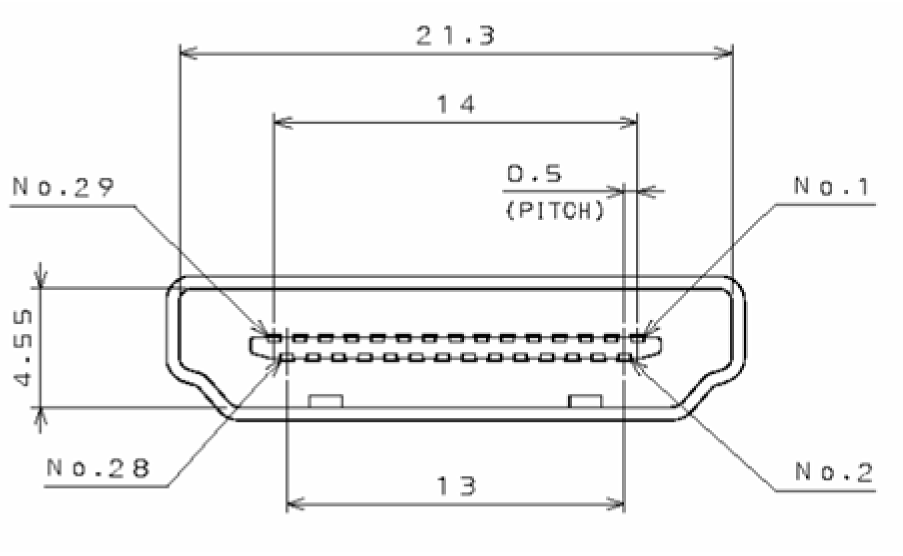
Table 4-2 Required Operating Conditions

HDMI I/F

- Compatible to DVI I/F
- Almost the same

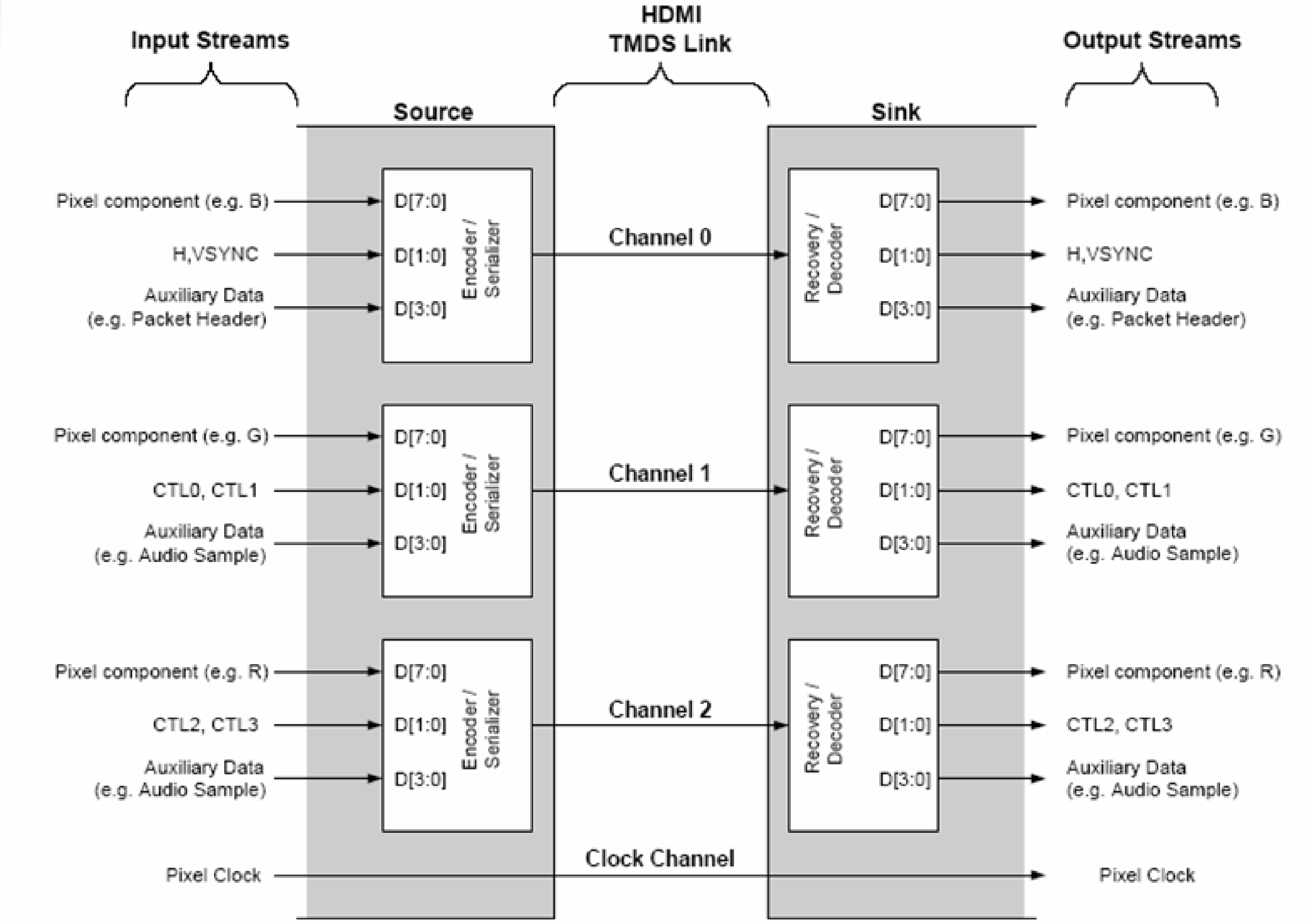


Type A: 19-pin single TMDS link



Type B: 29-pin dual TMDS link

HDMI I/F

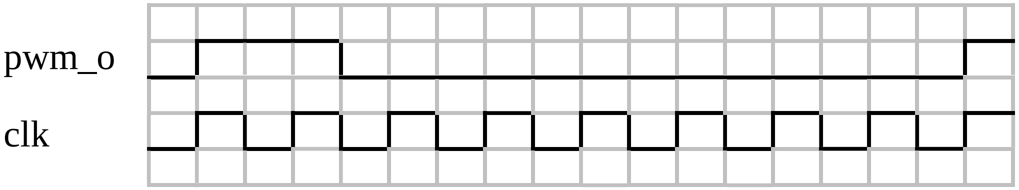

HDMI
TMDS Link
Input Streams
Output Streams
Source
Sink
Pixel component (e.g. B)
H,VSYNC
Auxiliary Data
(e.g. Packet Header)
D[7:0]
D[1:0]
D[3:0]
Encoder / Serializer
Channel 0
Recovery / Decoder
Pixel component (e.g. G)
CTL0, CTL1
Auxiliary Data
(e.g. Audio Sample)
Channel 1
Pixel component (e.g. R)
CTL2, CTL3
Auxiliary Data
(e.g. Audio Sample)
Channel 2
Clock Channel
Pixel Clock

Other Peripherals

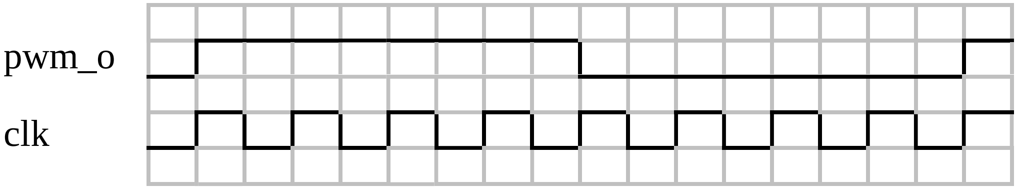
- Keypad controllers
- Pulse width modulators (PWM)
- Keypad controller

Pulse Width Modulator

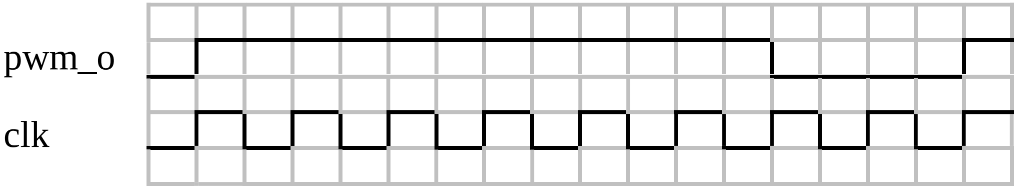
- Generates pulses with specific high/low times
- Duty cycle: % time high
  - Square wave: 50% duty cycle
- Common use: control average voltage to electric device
  - Simpler than DC-DC converter or digital-analog converter
  - DC motor speed, dimmer lights



25% duty cycle – average pwm_o is 1.25V

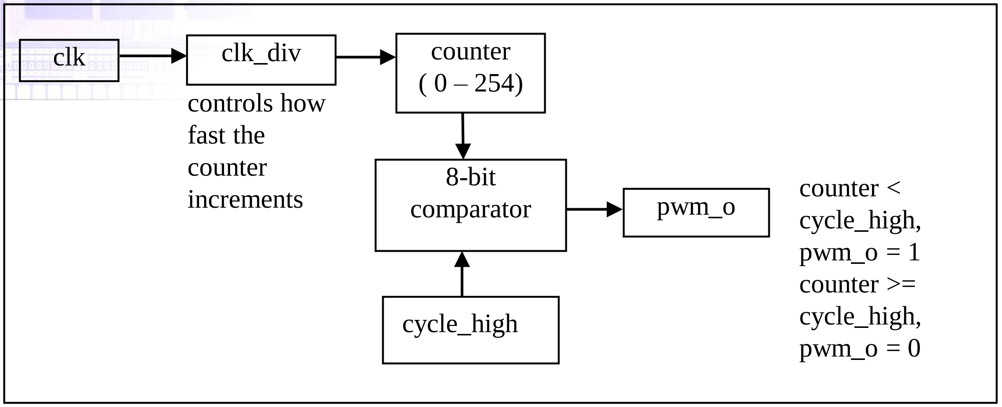


50% duty cycle – average pwm_o is 2.5V.



75% duty cycle – average pwm_o is 3.75V.

Controlling a DC Motor with a PWM



controls how fast the counter increments

counter < cycle_high, pwm_o = 1
counter >= cycle_high, pwm_o = 0

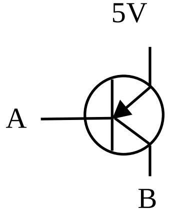
Internal Structure of PWM

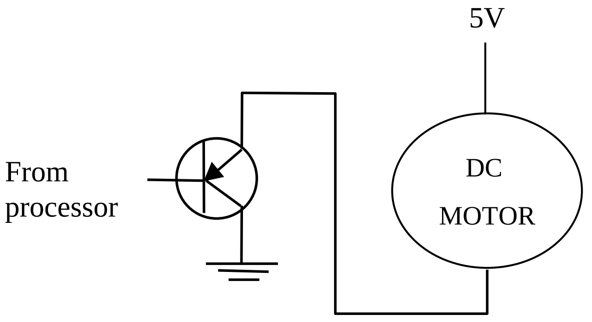
Input Voltage	% of Maximum Voltage Applied	RPM of DC Motor
0	0	0
2.5	50	1840
3.75	75	6900
5.0	100	9200

Relationship between applied voltage and speed of the DC Motor

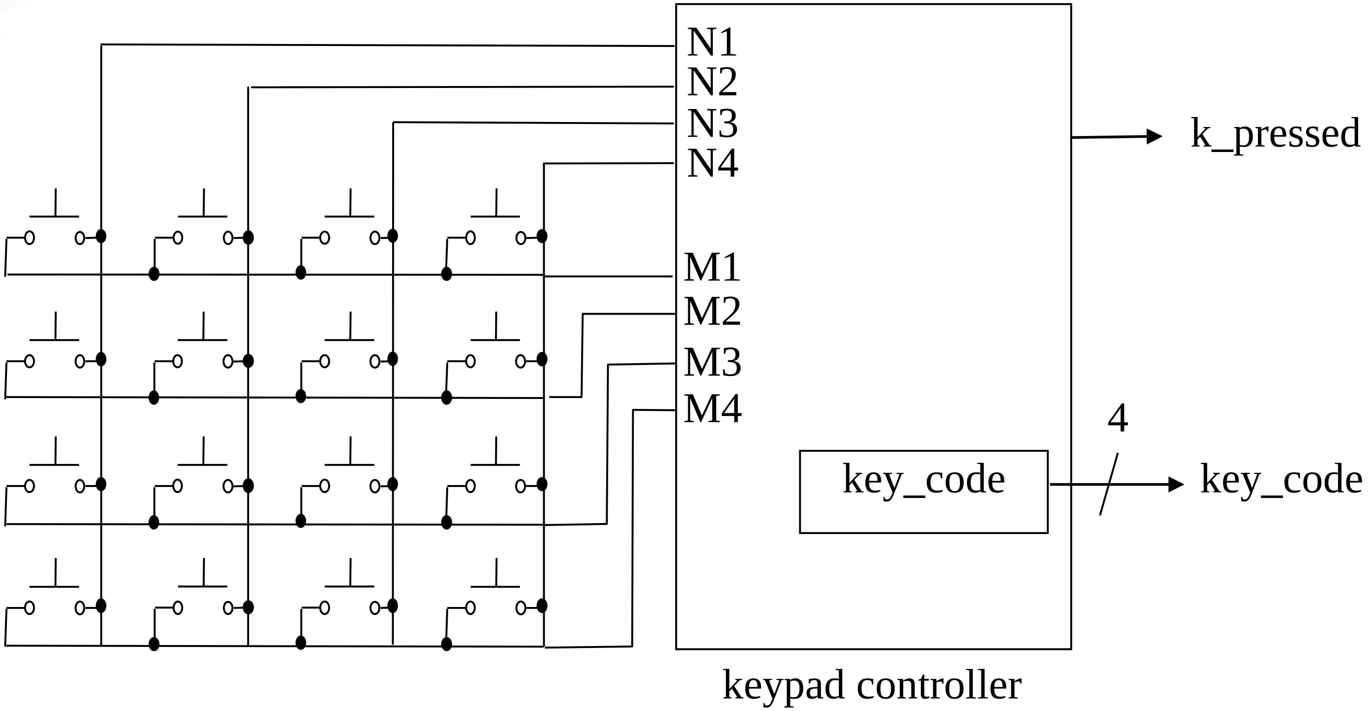
```
void main(void){

   /* controls period */
   PWMP = 0xff;
   /* controls duty cycle */
   PWM1 = 0x7f;

   while(1){};
 }
```

The PWM alone cannot drive the DC motor, a possible way to implement a driver is shown below using an MJE3055T NPN transistor.





Keypad Controller



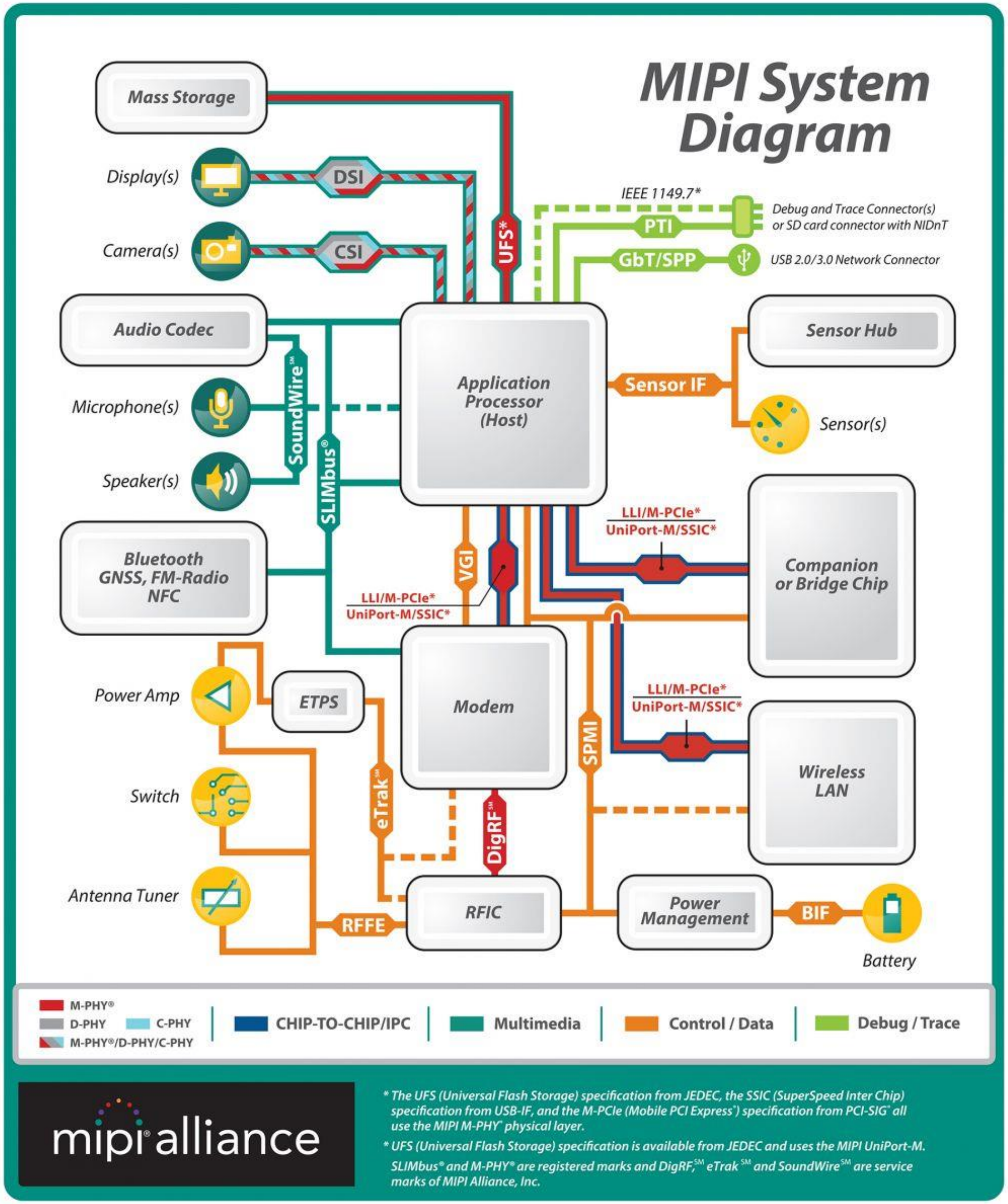
N=4, M=4

MIPI

- MIPI (Mobile Industry Processor Interface) Alliance
- Intel, Motorola, Nokia, Samsung, STMicron, TI, …
- MIPI specifications claimed a 20% penetration in smartphones in 2009 and will reach 100% by 2013.
- MIPI specification penetration in other phones will grow from 5% in 2009 to 90% in 2015.

PCI SIG
MEMS INDUSTRY GROUP
VESA
mipi alliance
JEDEC
SUPERSPEED
CERTIFIED USB

MIPI



* The UFS (Universal Flash Storage) specification from JEDEC, the SSIC (SuperSpeed Inter Chip) specification from USB-IF, and the M-PCIe (Mobile PCI Express®) specification from PCI-SIG® all use the MIPI M-PHY® physical layer.

* UFS (Universal Flash Storage) specification is available from JEDEC and uses the MIPI UniPort-M.

SLIMbus® and M-PHY® are registered marks and DigRF,℠ eTrak℠ and SoundWire℠ are service marks of MIPI Alliance, Inc.

CSI (Camera Serial Interface)

- CPI: camera parallel interface

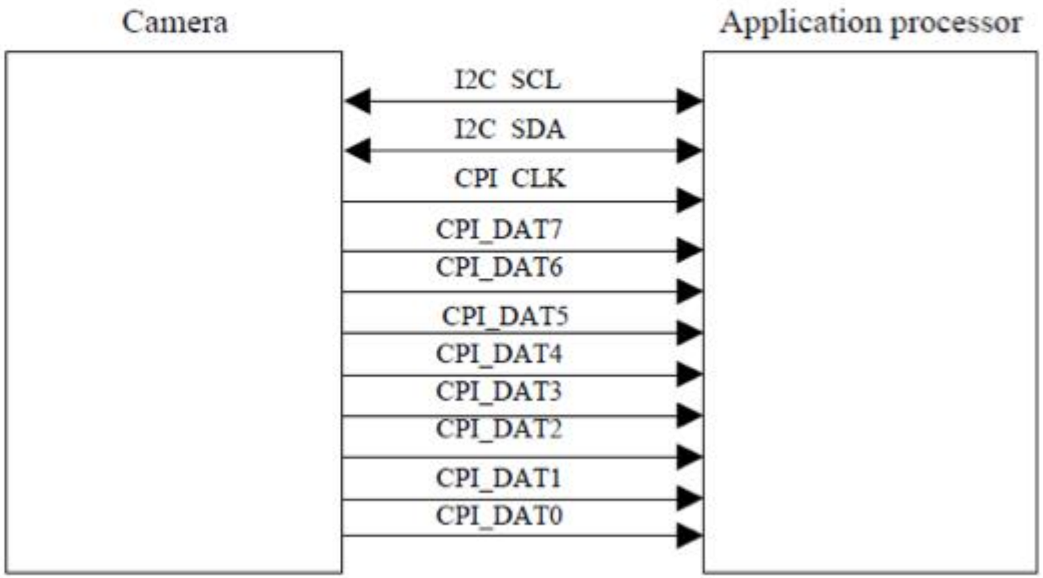


Figure 1. **CPI interface between camera and application processor**

CSI (Camera Serial Interface)

- CSI (Camera Serial Interface)

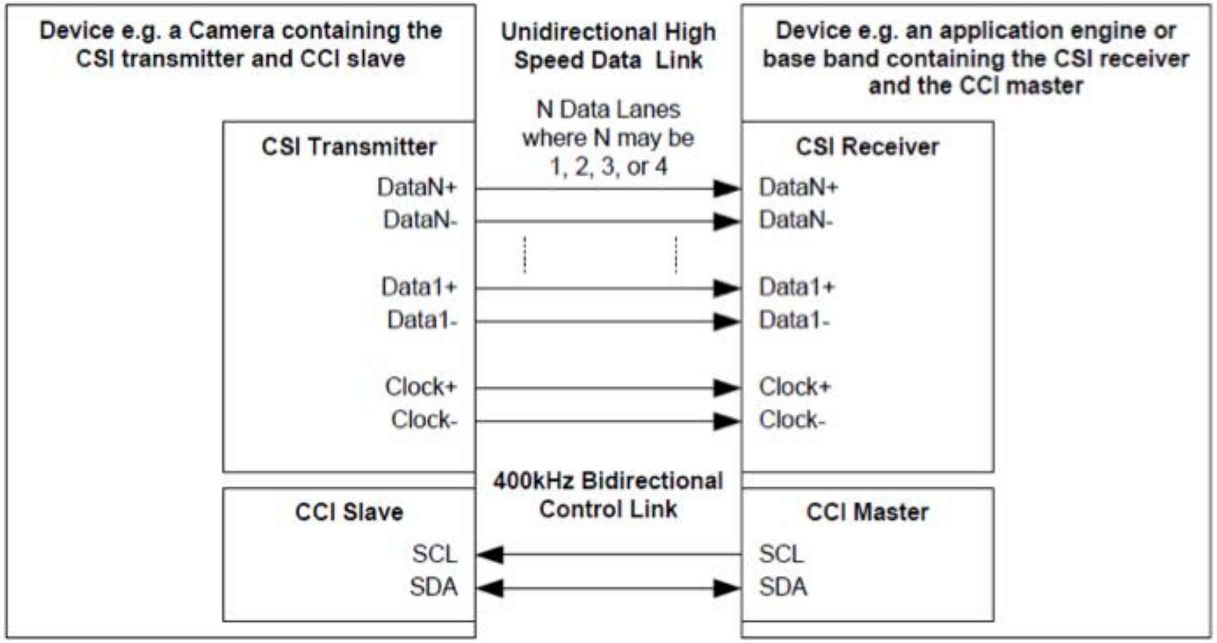

Device e.g. a Camera containing the CSI transmitter and CCI slave
Unidirectional High Speed Data Link
Device e.g. an application engine or base band containing the CSI receiver and the CCI master
N Data Lanes where N may be 1, 2, 3, or 4
CSI Transmitter
DataN+
DataN-
Data1+
Data1-
Clock+
Clock-
CSI Receiver
DataN+
DataN-
Data1+
Data1-
Clock+
Clock-
400kHz Bidirectional Control Link
CCI Slave
SCL
SDA
CCI Master
SCL
SDA

Figure 1 CSI-2 and CCI Transmitter and Receiver Interface

CSI (Camera Serial Interface)

- CSI-2 and CSI-3 define different generations of an interface between a peripheral device (camera, Image Signal Processor) and a host processor (baseband, application engine)
- **CSI-2** transfers image data over a unidirectional D-PHY physical layer, using Camera Control Interface ("CCI," compatible with I2C) for control
- **CSI-3** uses UniPort-M (UniPro v1.4 and M-PHY v2.0) as a bidirectional transport for data and control.
- CSI-2: v1.1
  - Speed CSI-2: 80 Mbps – 1.5 Gbps per D-PHY lane (up to 4 lanes)
- CSI-3: v1.0 Speed CSI-3: 1 Mbps - 3 Gbps per M-PHY lane (up to 4 lanes)