Module 3

INPUT/OUTPUT (IO) PROGRAMMING

INPUT/OUTPUT (IO) PROGRAMMING:

Pin connect block, Pin function select registers, General Purpose Input and Output (GPIO) registers, GPIO configuration, GPIO programming using ARM C language, Interfacing: LEDs, Seven segment, LCD, keyboard, DC motor, Stepper motor.

Text 2: Refer Ch 4, Ch 5
Ref: UM10360, LPC 176x/5x User Manual

On most embedded microcontrollers, the I/O ports are memory mapped.

The software can access an input/output port simply by reading from or writing to the appropriate address.

It is important to realize that even though I/O operations “look” like reads and writes to memory variables, the I/O ports often DO NOT act like memory.

5.12.5 Parallel Interface

- The simplest I/O port on a microcontroller is the parallel port. A parallel I/O port is a simple mechanism that allows the software to interact with external devices. It is called parallel because multiple signals can be accessed all at once.
- An input port, which allows the software to read external digital signals, is read only. That means a read cycle access from the port address returns the values existing on the inputs at that time. In particular, the tri-state driver will drive the input signals onto the data bus during a read cycle from the port address.
- A write cycle access to an input port usually produces no effect. The digital values existing on the input pins are copied into the microcontroller when the software executes a read from the port address. Typical examples of I/O port include output ports that drive LEDs, ports to scan a keypad, ports to control machinery, etc. Some ports are used to obtain status information about the interface through

Parallel Interface

- The simplest **I/O port** on a microcontroller is the **parallel port.**

- A parallel I/O port is a simple mechanism that allows the software to **interact with external devices**.

- It is called **parallel** because **multiple signals can be accessed all at once.**

Parallel Interface

- An **input port**, which **allows the software to read external digital signals**, is **read-only**.

- That means a **read cycle access from the port address returns the values existing on the inputs at that time.**

- In particular, the **tri-state driver** will drive the **input signals** onto **the data bus** during a **read cycle from the port address**.

Parallel Interface

- A **write cycle** access to an **input port** usually produces **no effect**.

- The digital values existing on the input pins are copied into the microcontroller when the software executes a **read operation** from the port address.

examples of I/O ports

- **Typical examples of I/O ports** include:
    - Output ports that drive **LEDs**.
    - Ports to **scan a keypad**.
    - Ports to **control machinery**.

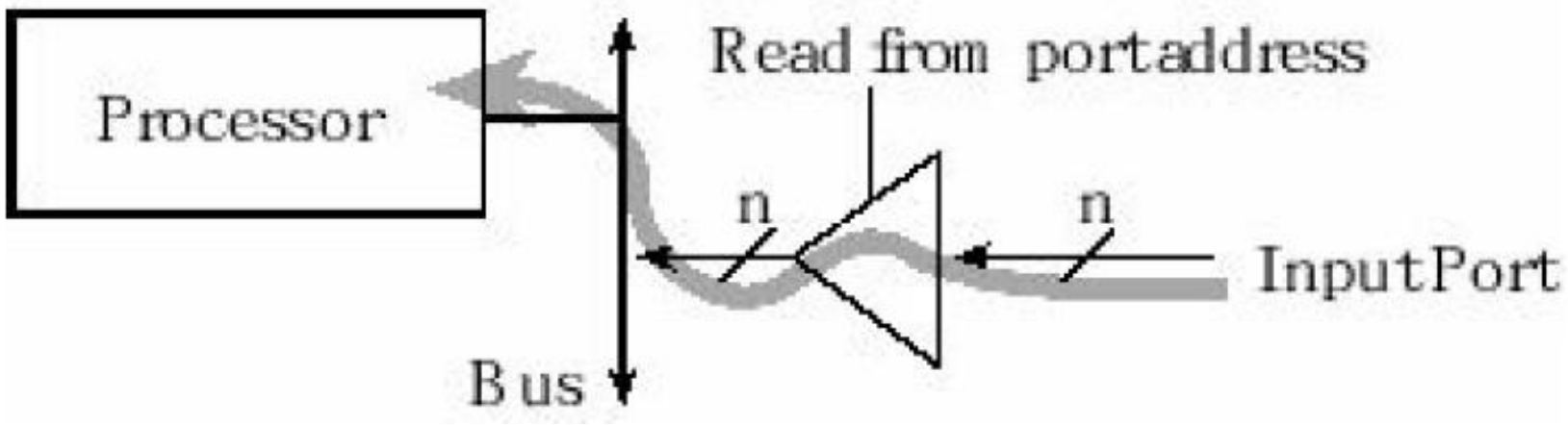
- The simplest I/O port on a microcontroller is the **parallel port**.

- A **parallel I/O port** is a simple mechanism that allows the software to **interact with external devices**.

- It is called parallel because **multiple signals** can be accessed all at once.

An **input port**, which is read only, allows the software to read external digital signals.

A read cycle access from the port address returns the values existing on the inputs at that time.



Input Port

- Input ports can also be implemented with a minimum of hardware. A tri-state buffer is used to connect the external digital input to the CPU's data bus during a read cycle if the CPU is addressing the memory or IO address assigned to the input port.
- The read strobe (RD) is used to enable the buffer so that it connects the input to the CPU data bus. Fig. 5.12.8 shows parallel input port.
- The CPU's inverted RD* strobe (RD) is used to enable the output of a tri-state buffer when RD is active and the address corresponds to the address of the input port.

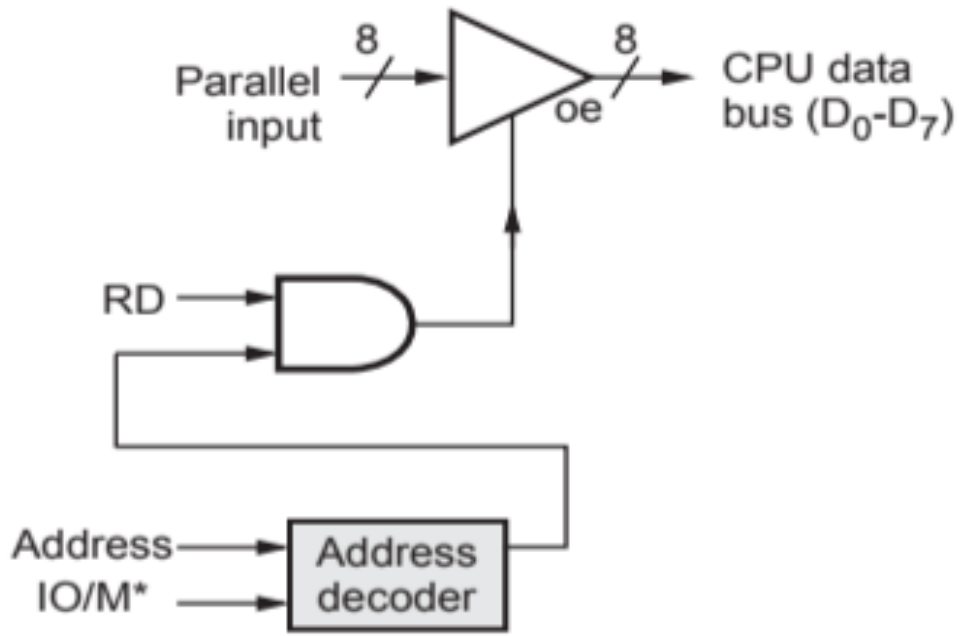


Fig. 5.12.8 Parallel input port

Input Port

- Input ports can also be implemented with a minimum of hardware.

- A **tri-state buffer** is used to connect the external digital input to the CPU's data bus during a read cycle if the CPU is addressing the memory or I/O address assigned to the input port.

- The read strobe (RD) is used to enable the buffer so that it connects the input to the CPU data bus.

Input Port

Fig. 5.12.8 shows a parallel input port.

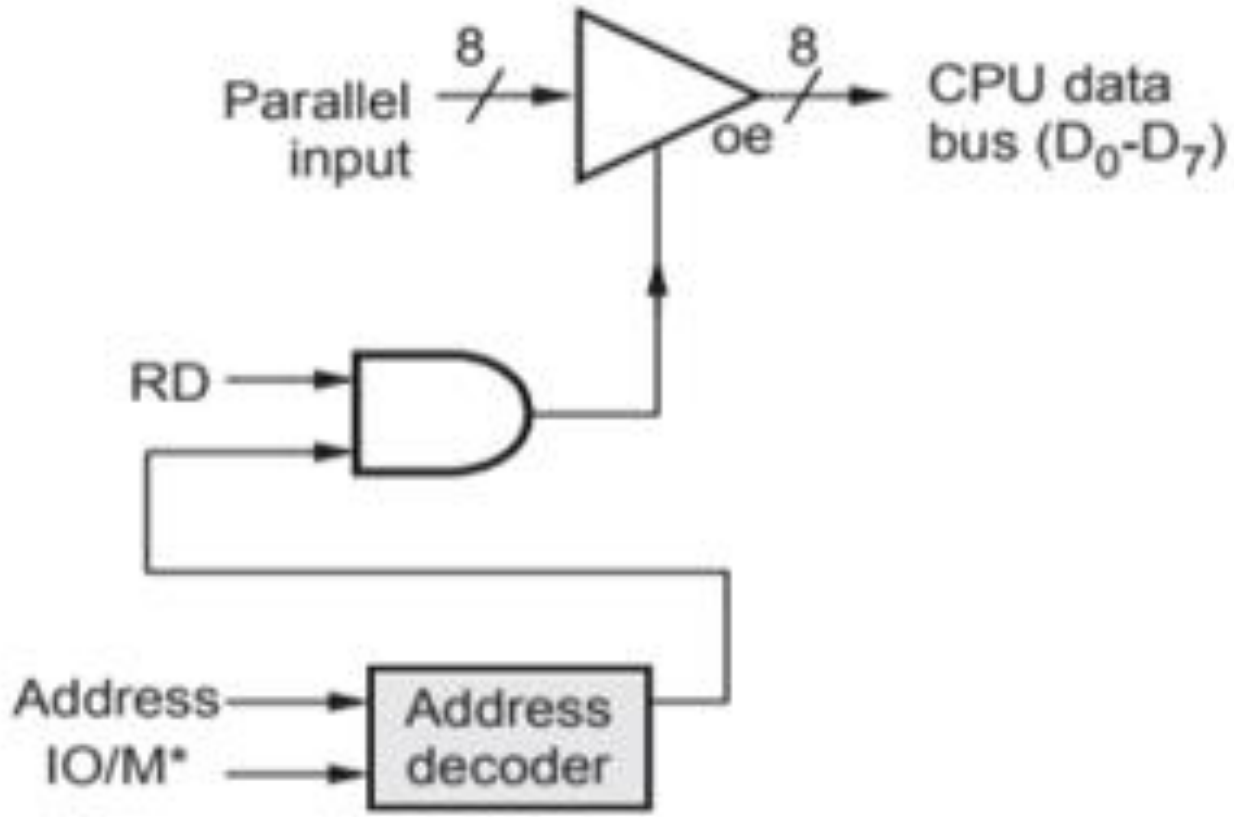


Fig. 5.12.8 Parallel input port

- The *CPU's inverted RD strobe (RD)** is used to enable the output of a **tri-state buffer** when RD is active and the address corresponds to the address of the input port.

What is an Input Port?

- An **input port** allows external digital signals to be transferred to the CPU.

- It acts as an interface between external devices (such as sensors, keyboards, or switches) and the processor.

Role of the Tri-State Buffer

- A **tri-state buffer** is a special type of electronic component that can be in one of three states: **high (1), low (0), or high impedance (Z)**.

- The buffer connects the external digital input to the **CPU's data bus (D_0 - D_7)** only when the CPU is reading data from the input port.

- When the buffer is in a high-impedance (Z) state, it effectively disconnects from the data bus, preventing interference.

How Does the Read Strobe (RD) Work?

- The **Read (RD) signal** is a control signal generated by the CPU when it wants to read data from an input port.

- When RD is **active (low)**, it enables the tri-state buffer, allowing data from the external input to be sent to the **CPU data bus**.

Role of the Address Decoder

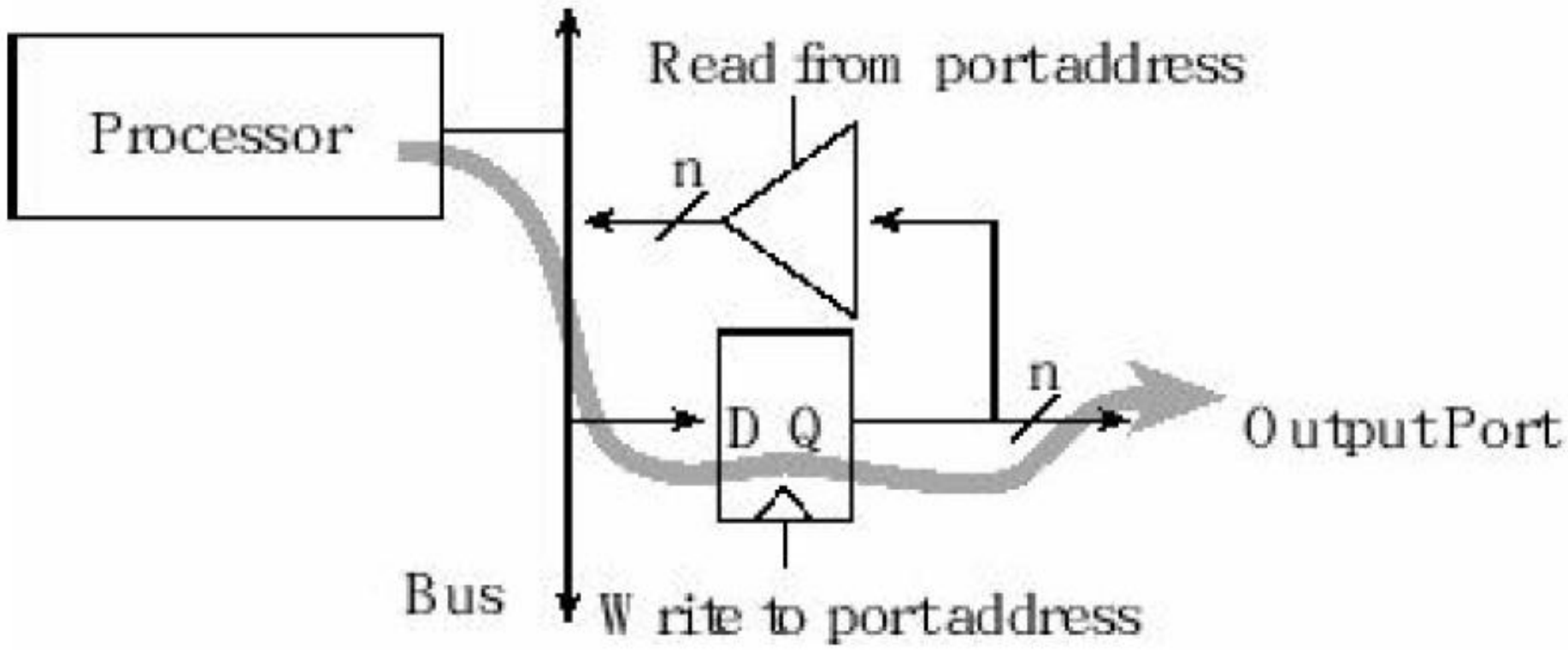
- The **address decoder** ensures that the buffer is enabled only when the CPU is addressing the specific input port.

- The decoder checks if the address matches the assigned input port and generates an enabling signal.

Working of the Circuit (Fig. 5.12.8)

- The **AND gate** takes the **RD signal** and an address decoding signal as inputs.
- When both the **RD signal is active** and the **correct address is detected**, the AND gate enables the **tri-state buffer**.
- The buffer then connects the **parallel input data** to the **CPU data bus**, allowing the CPU to read the external input.

What happens if the software writes to an input port?



Working of the Circuit

Writing to the Output Port

- The processor places data on the bus and activates the Write to Port Address signal.
- This stores the data in the D flip-flop, which holds the value until new data is written.
- The stored value is continuously sent to the output device.

Reading from the Output Port

Reading from the Output Port

- When the processor needs to read the data currently stored in the output port, it activates the **Read from Port Address** signal.

- This enables the **tri-state buffer**, allowing the processor to read the stored value.

"status registers" and other ports can control the interface's operation through "control registers.

Output Port

- Output ports are implemented using registers, multi-bit flip-flops with a common clock. The registers data inputs (D) are connected to the CPU data bus and the registers clock input is driven by the CPU write strobe (WR).
- In addition, an address decoder is used to make sure the clock is only asserted when the CPU is addressing the desired IO or memory address.
- The rising edge of the write strobe loads the data into the register output (Q) and this output stays fixed until the register is written again. Fig. 5.12.7 shows how a register could be connected to operate as an output port.
- The CPU's write strobe (WR) is used to clock the data into the register, but only if the address on the CPU bus corresponds to the address of the output port.

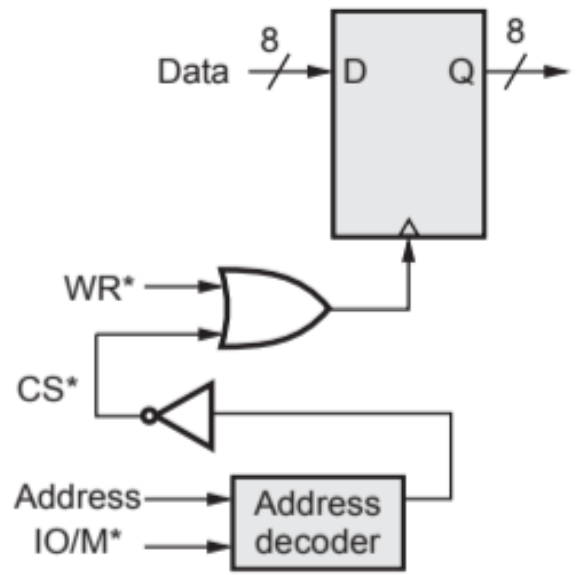


Fig. 5.12.7 Output port operation

Output Port

- Output ports are implemented using registers, multi-bit flip-flops with a common clock.

- The register’s data inputs (D) are connected to the CPU data bus, and the register's clock input is driven by the CPU write strobe (WR).

- In addition, an **address decoder** is used to make sure the clock is only asserted when the CPU is addressing the desired **I/O** or **memory address**.

- The **rising edge of the write strobe** loads the data into the register output (Q), and this output stays **fixed** until the register is written again.
- **Fig. 5.12.7** shows how a register could be connected to operate as an **output port**.
- The **CPU's write strobe (WR)** is used to clock the data into the register, but only if the address on the CPU bus **corresponds** to the address of the output port.

What is IO programming?

- Through IO programming microcontroller can be used to control other devices such as sensors, displays, On-chip modules etc.
- It is done through GPIO (General purpose Input Output).
- GPIO is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time. It's a standard interface used to connect microcontrollers to other electronic devices.

- In LPC1768, there are 5 ports, P0 to P4 each 32 bit (i.e., 32 pins). Few port pins are not available. (total pins = 100)
- In GPIO operation we perform general purpose operation to read from the port and to write to the port.

LPC 1768

- The LPC1768 is an ARM Cortex-M3 based microcontroller for embedded applications requiring a high level of integration and low power dissipation.
- The ARM Cortex-M3 CPU incorporates a 3-stage pipeline and uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals.
- The ARM Cortex-M3 is a general purpose 32-bit microprocessor, which offers high
- performance and very low power consumption. The Cortex-M3 offers many new features, including a Thumb instruction set, low interrupt latency etc.

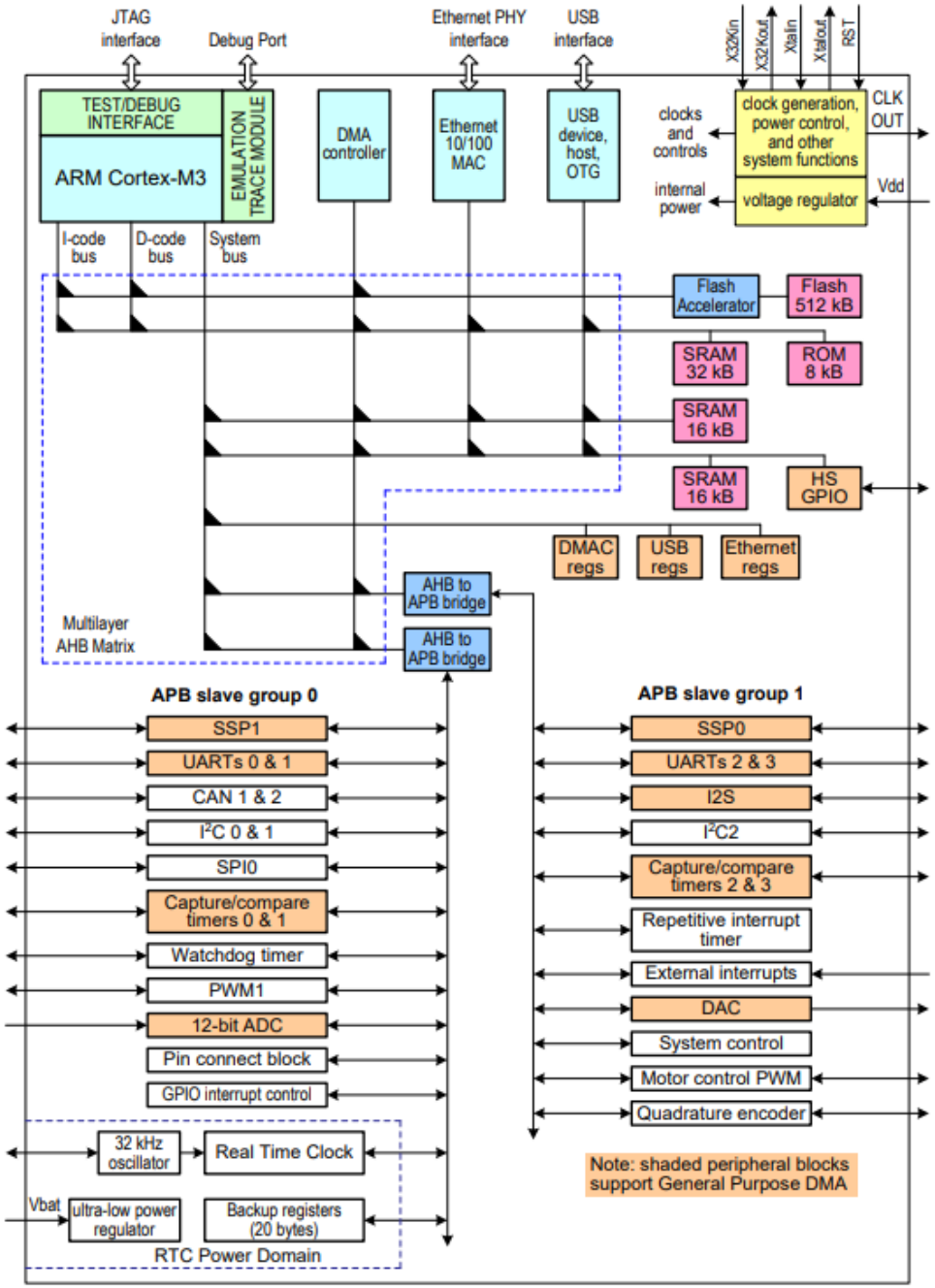
- Applications
  - e-Metering
  - Lighting
  - Industrial networking
  - Alarm systems
  - Motor control

The LPC1768 is a microcontroller based on the ARM Cortex-M3 architecture.

It has a variety of pins that can be configured for different purposes.

Basic overview of the pin configuration for the LPC1768:

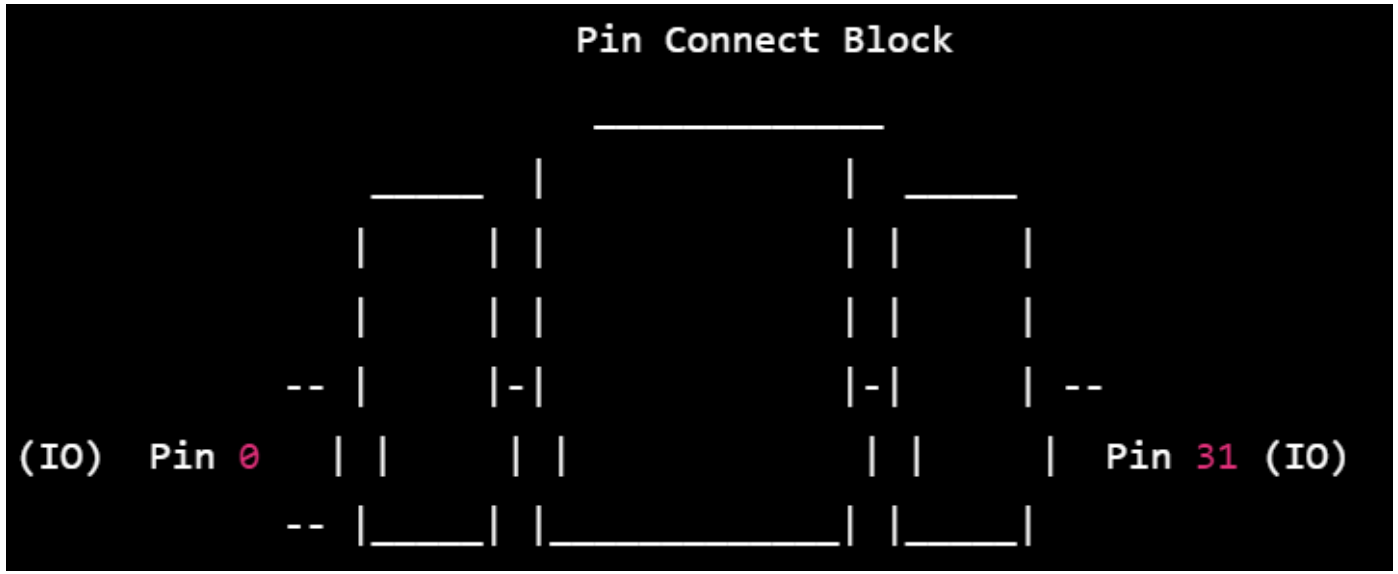
- **Pin Connect Block (PCB)**
- **Pin Function Select Registers (PFS)**
- **General Purpose Input and Output (GPIO) Registers**
- **GPIO Configuration**
- **GPIO Programming using ARM C Language**



LPC1768 block diagram, CPU and buses

Pin Connect Block (PCB)

- Responsible for configuring the pinout and pin functions of the device.
- Allows to assign various functions to the pins on the microcontroller, such as input, output, interrupts, and analog inputs.



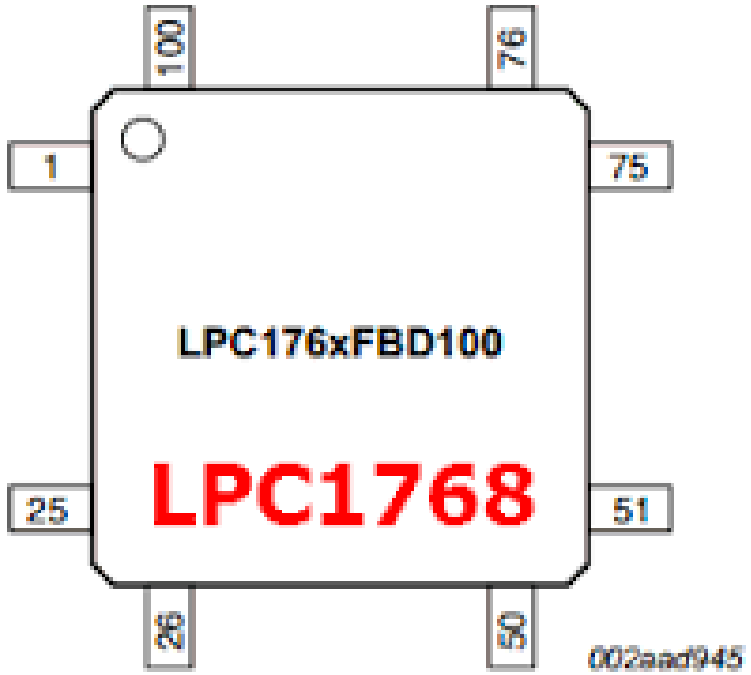
- The Pin Connect Block is depicted as a rectangular box with pins labeled from 0 to 31. This box represents the hardware on the microcontroller that is responsible for configuring the pinout and pin functions.
- The pins are arranged in a grid of rows and columns. Each pin can be assigned a specific function, such as input, output, or alternate function, using the Pin Connect Block.
- The specific pin functions and assignments will depend on the particular microcontroller being used.

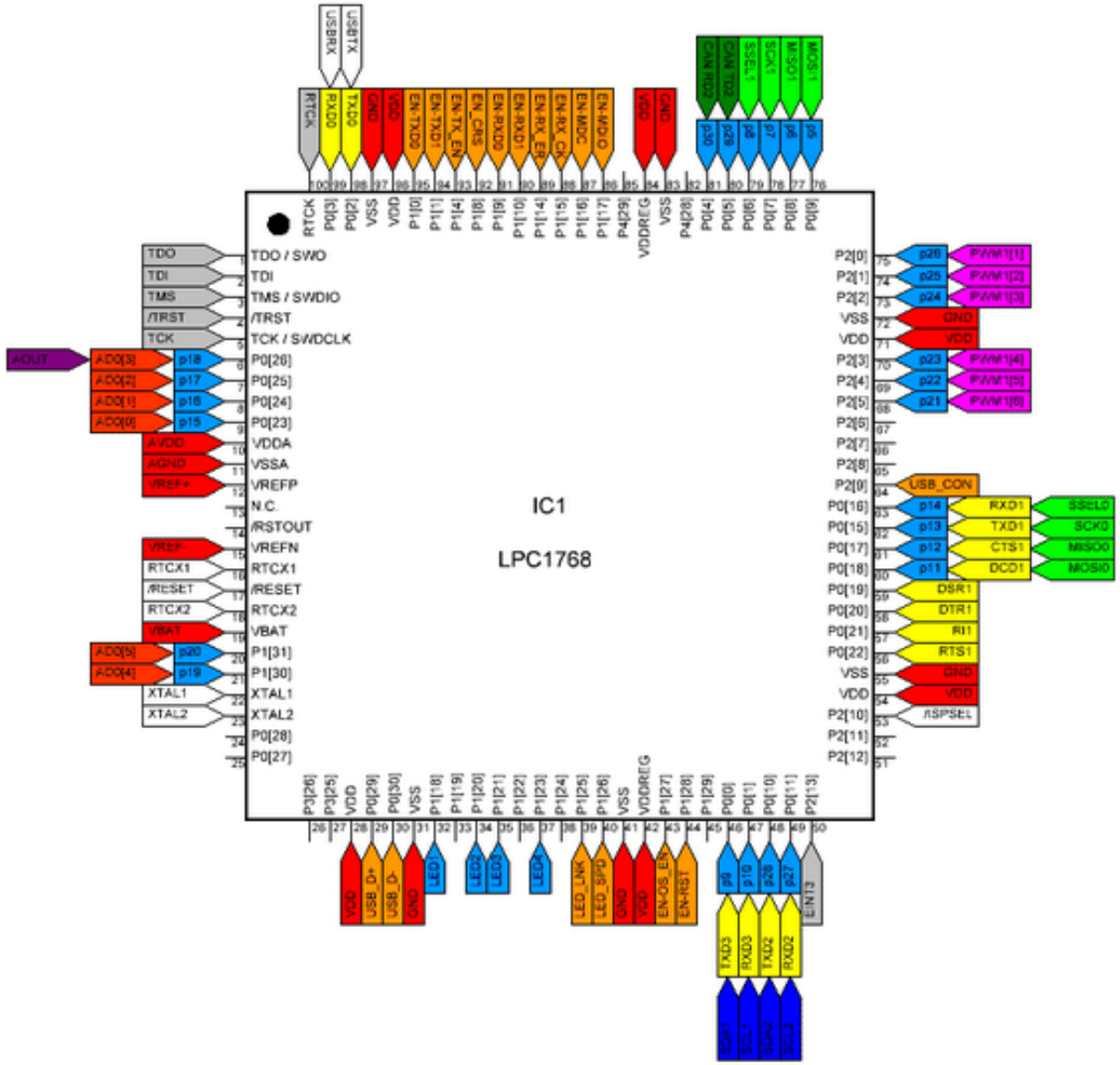
- The pins are labeled from 0 to 31, indicates that this particular microcontroller has 32 pins that can be configured using the Pin Connect Block.
- The pins labeled as "IO" indicate that they can be used as either inputs or outputs, depending on how they are configured.
- The horizontal lines at the top and bottom of the box represent the power and ground pins of the microcontroller. These pins provide the necessary power and ground connections for the device to operate.
- The Pin Connect Block is a key component of the ARM Cortex-M3 microcontroller, as it allows developers to configure the pin functions and assignments to meet the specific requirements of their applications.

INPUT/OUTPUT (IO) PROGRAMMING

LPC1768 pin configuration

512 K Flash memory
64K SRAM
8K ROM
100 pin IC





GPIO in LPC1768

- **5 general purpose bidirectional digital IO ports**, Port 0, Port 1, Port 2, Port 3 and Port 4.
  - ❑ Fast GPIO-enhanced /Accelerated Features
  - ❑ Additionally, Port 0 and Port 2 pins can provide a single interrupt
- **Application**
  - ❑ General purpose I/O, Driving LEDs or other indicators, Controlling off-chip devices, Sensing digital inputs, detecting edges

- Port 0 Pins 12, 13, 14 & 31 are not available.
- Port 1 Pins 2, 3, 7, 6, 5, 11, 12, & 13 are not available.
- Port 2 only pins 0 to 13 are available and rest are reserved.
- Port 3 only pins 25,26 are available and rest are reserved.
- Port 4 only 28,29 are available and rest are reserved.

- In total 70 GPIO pins are available for the user.

Pin connect block

LPC1768 microcontroller
- 100 pin IC provided by NXP semiconductor
- Five ports, Port 0, Port 1, Port 2, Port3 and Port 4

- Each pin can have maximum four functions

- The naming convention for port pins is '**Px.y**' where '**x**' is the port number and '**y**' is simply the pin number in port '**x**'.

For example:

P0.7 refers to Pin number **7** of Port **0** , **P2.11** refers to Pin number **11** in Port **2**.

Pin Connect Block Registers

- Pin function select registers

Note: all registers are 32 bit wide

Register	Controls
PINSEL0	P0[15:0]
PINSEL1	P0 [31:16]
PINSEL2	P1 [15:0] (Ethernet)
PINSEL3	P1 [31:16]
PINSEL4	P2 [15:0]
PINSEL5	P2 [31:16]
PINSEL6	P3 [15:0]
PINSEL7	P3 [31:16]
PINSEL8	P4 [15:0]
PINSEL9	P4 [31:16]
PINSEL10	Trace port enable

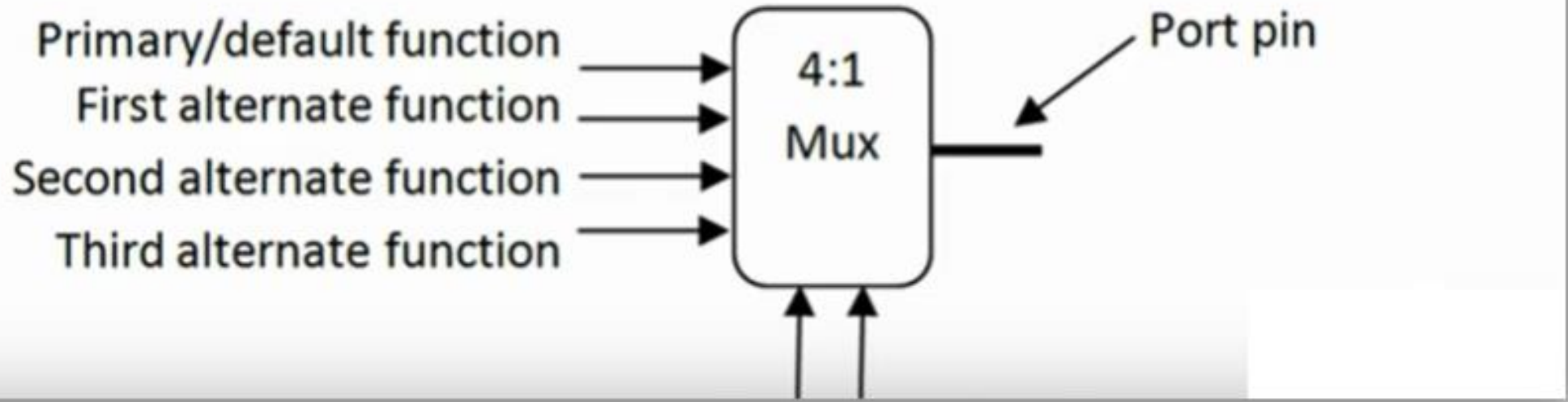
Pin function select register

- Configure the microcontroller pins to the desired functions
- Memory mapped registers
  - control the multiplexers
  - allows connection between on-chip peripherals and the pins
- Single function selection of any port pin completely excludes all other functions

Pin function select register

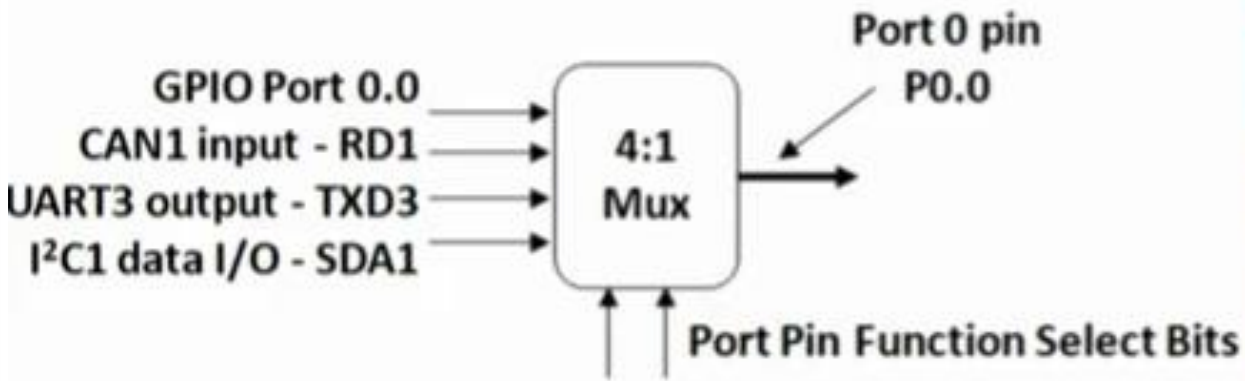
- Pin function select register
  - Each port pin can be used for either of maximum four functions with the help of multiplexer like structure

- Number of bits required to configure function of a port pin
  - 4=2^2, so 2 bits are required



Port Pin Function Select Bits

- Bit combinations to configure function of a port pin



Port Pin Function Select Bits	Function	Port0 pin P0.1 Function
00	Primary (default)	GPIO Port
01	First alternate	CAN1 – RD1
10	Second alternate	UART3 – TXD3
11	Third alternate	I²C1 - SDA1

5.12 On Board Communication Interface

5.12.1 I^2C

- I^2C bus means **Inter - Integrated Circuit Bus.** A Small Area Network connecting ICs and other electronic systems
- The I^2C Bus is a well-known bus commonly used to link microcontrollers into systems. It has even been used for the command interface in an MPEG-2 video chip. It is designed for low-cost, medium data rate applications.
- Today, a variety of devices are available with I^2C Interfaces like Microcontroller, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter.

Characteristics

1. It includes electrical and timing specifications, and an associated bus protocol.
2. Two wire serial data & control bus implemented with the serial data (SDA) and clock (SCL) lines
3. For reliable operation, a third line is required : Common ground
4. It uses unique start and stop condition
5. Slave selection protocol uses a 7-Bit slave address
6. The bus specification allows an extension to 10 bits
7. It support Bi-directional data transfer
8. It gives acknowledgement after each transferred byte

9. No fixed length of transfer
10. Transmission speeds up to 100 kHz
11. It is Compatible with different IC technologies
12. It uses an open-drain/open-collector with an input buffer on the same line, which allows a single data line to be used for bidirectional data flow.

- The I^2C bus is a standard bidirectional interface that uses a controller, known as the master, to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master.
- Each device on the I^2C bus has a specific device address to differentiate between other devices that are on the same I^2C bus. Many slave devices will require configuration upon startup to set the behavior of the device. This is typically done when the master accesses the slave's internal register maps, which have unique register addresses. A device can have one or multiple registers where data is stored, written, or read.
- Fig. 5.12.1 shows physical layer of I^2C bus.

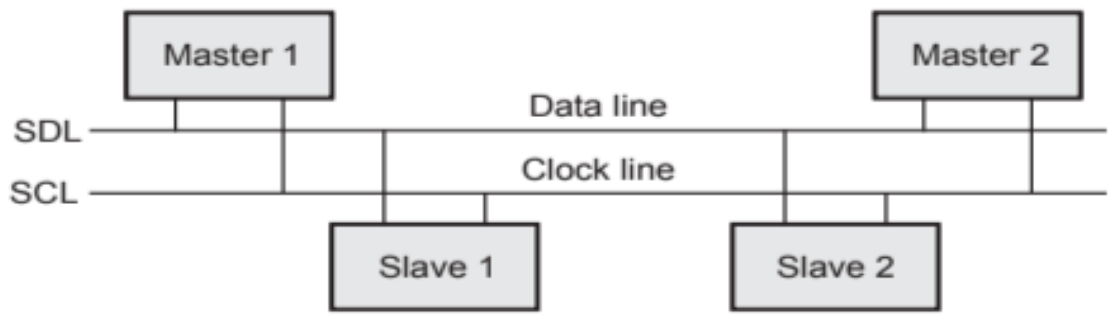


Fig. 5.12.1 Physical layer of I^2C bus

5.12.3 UART

- The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. Fig. 5.12.5 shows UART.

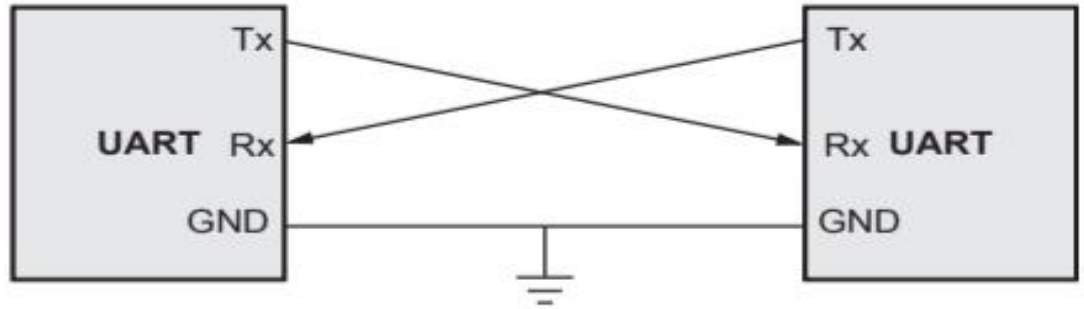
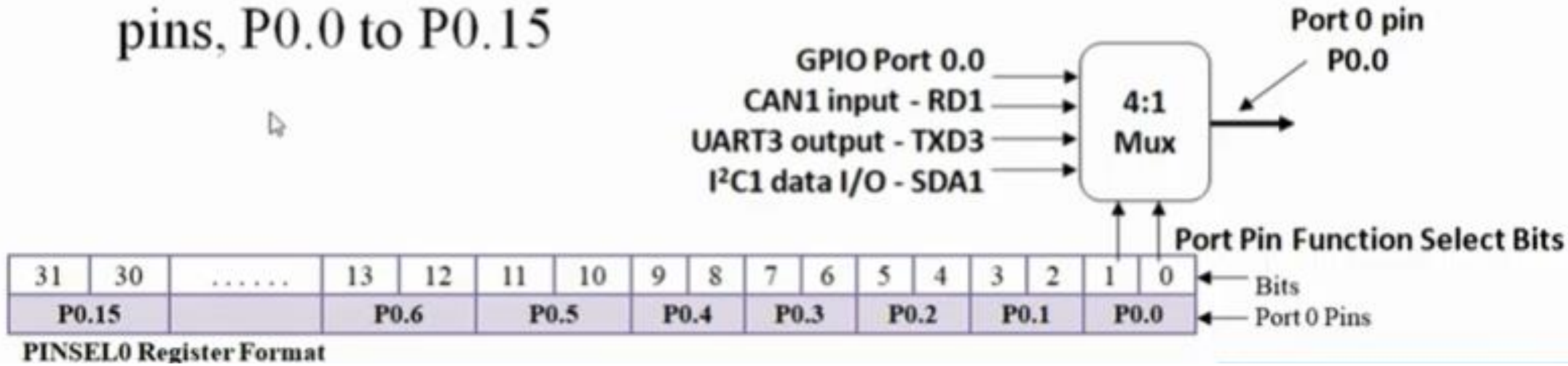


Fig. 5.12.5 UART

- The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.
- Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.
- It also convert data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media, and to regulate the flow of data in the event that the remote device is not prepared to accept more data.
- For example, when the device connected to the UART is a modem, the modem may report the presence of a carrier on the phone line while the computer may be able to instruct the modem to reset itself or to not take calls by raising or lowering one more of these extra signals.

- PINSEL0
  - 32 bit wide register controls the functions of the Port 0 pins, P0.0 to P0.15



PINSEL0 Register Format

- PINSEL1 - controls the functions of the Port 0 pins, P0.16 to P0.30
- PINSEL2 - controls the functions of the Port 1 pins, P1.0 to P1.15
- PINSEL3 - controls the functions of the Port 1 pins, P1.16 to P1.31

- PINSEL4 - controls the functions of the Port 2 pins, P2.0 to P2.13
- PINSEL7 - controls the functions of the Port 3 pins, P3.25 & P3.26
- PINSEL9 - controls the functions of the Port 4 pins, P4.28 & P4.29

- PINSEL10 - controls the Trace function on the Port 2 pins, P2.2 to P2.6

Pin function select register

- PINSEL0 bit description

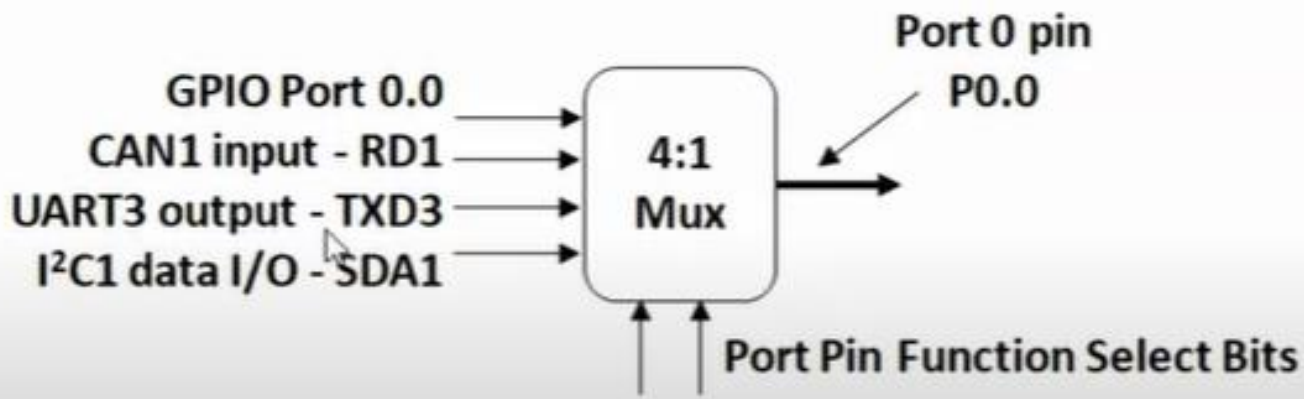
Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

Source : LPC1768 Data sheet UM10360

Pin function select register

- PINSEL0 - Selecting Port0 pin P0.0 as transmitter output for UART3



Pin function select register

- PINSEL0 - Selecting Port0 pin P0.0 as transmitter output for UART3

P0.15		Reserved						P0.11		P0.10		P0.9		P0.8		P0.7		P0.6		P0.5		P0.4		P0.3		P0.2		P0.1		P0.0	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	1	1	0

PINSEL0 Register

Predefined Values

Mask (**AND**)

Desired Value (**OR**)

Final value

Pin function select register

- PINSEL0 - Selecting Port0 pin P0.0 as transmitter output for UART3
  - PINSEL0 &= 0xFFFFFFFC; OR
    PINSEL0 = PINSEL0 & 0xFFFFFFFC;
  - PINSEL0 |= 0x00000002; OR
    PINSEL0 = PINSEL0 || 0x00000002;

Table 102. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if an FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be read as 0 regardless of the physical pin state.	R/W	0	FIO0PIN - 0x2009 C014 FIO1PIN - 0x2009 C034 FIO2PIN - 0x2009 C054 FIO3PIN - 0x2009 C074 FIO4PIN - 0x2009 C094
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.	R/W	0	FIO0SET - 0x2009 C018 FIO1SET - 0x2009 C038 FIO2SET - 0x2009 C058 FIO3SET - 0x2009 C078 FIO4SET - 0x2009 C098
FIOCLR	Fast Port Output Clear register using FIOMASK. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK can be altered.	WO	0	FIO0CLR - 0x2009 C01C FIO1CLR - 0x2009 C03C FIO2CLR - 0x2009 C05C FIO3CLR - 0x2009 C07C FIO4CLR - 0x2009 C09C

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

FIODIR: Fast GPIO Direction Control Register.
This register individually controls the direction of each port pin.

Values	Direction
0	Input
1	Output

FIODIR: Fast GPIO Direction Control Register.
This register individually controls the direction of each port pin.

Values	Direction
0	Input
1	Output

FIOSET: Fast Port Output Set Register.
This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register not the physical port value.

Values	FIOSET
0	No Effect
1	Sets High on Pin

FIOCLR: Fast Port Output Clear Register.
This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect.

Values	FIOCLR
0	No Effect
1	Sets Low on Pin

FIOPIN: Fast Port Pin Value Register.
This register is used for both reading and writing data from/to the PORT.
Output: Writing to this register places corresponding values in all bits of the particular PORT pins.
Input: The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC).
Note: It is recommended to configure the PORT direction and pin function before using it.

Table 104. Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIO4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

GPIO Port direction control registers

- Word accessible FIOxDIR register (32 bit wide) controls the direction of each pin of Port x

Name	Description	Access	Reset value	Memory Address
FIO0DIR	Port 0 Direction control register	Read/Write	0x0000 0000	0x2009 C000
FIO1DIR	Port 1 Direction control register	Read/Write	0x0000 0000	0x2009 C020
FIO2DIR	Port 2 Direction control register	Read/Write	0x0000 0000	0x2009 C040
FIO3DIR	Port 3 Direction control register	Read/Write	0x0000 0000	0x2009 C060
FIO4DIR	Port 4 Direction control register	Read/Write	0x0000 0000	0x2009 C080

GPIO port output Set register FIOxSET (FIO0SET to FIO4SET)

- This register is used to produce a High-level output at the port pins configured as GPIO in an OUTPUT mode.
  - ❑ **Writing 1 produces a HIGH level at the corresponding port pins**
  - ❑ **Writing 0 has no effect**
- If any **pin is configured as an input or a secondary function**, writing 1 to the corresponding bit in the FIOxSET has no effect.
- Reading the FIOxSET register returns the value of this register, as determined by previous writes to FIOxSET and FIOxCLR (or FIOxPIN).
- Access to a port pin via the FIOxSET register is conditioned by the corresponding bit of the FIOxMASK register

Table 106. Fast GPIO port output Set register (FIO0SET to FIO4SET - addresses 0x2009 C018 to 0x2009 C098) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0SET FIO1SET FIO2SET FIO3SET FIO4SET		Fast GPIO output value Set bits. Bit 0 in FIOxSET controls pin Px.0, bit 31 in FIOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

GPIO port output Clear register FIOxCLR (FIO0CLR to FIO4CLR)

- This register is used to produce a LOW-level output at port pins configured as GPIO in an OUTPUT mode.
- Writing 1 produces a LOW level at the corresponding port pin and clears the corresponding bit in the FIOxSET register.
- Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to FIOxCLR has no effect.
- Access to a port pin via the FIOxCLR register is conditioned by the corresponding bit of the FIOxMASK register

Table 108. Fast GPIO port output Clear register (FIO0CLR to FIO4CLR- addresses 0x2009 C01C to 0x2009 C09C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0CLR FIO1CLR FIO2CLR FIO3CLR FIO4CLR		Fast GPIO output value Clear bits. Bit 0 in FIOxCLR controls pin Px.0, bit 31 controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

GPIO Port direction control registers

- FIOxDIR controls the direction of each pin of Port x
  - e.g., in FIO0DIR - bit 0 controls P0.0 and bit 31 controls P0.31
  - direction control bit is effective only when the GPIO function is selected for a pin
- Direction values
  - 0 - Controlled pin is input
  - 1 - Controlled pin is output
  - All GPIO port pins default to inputs after reset

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin
  - logically AND default contents of FIO0DIR register with mask (*preserve the direction of other port pins*)
    - 0's in bits corresponding to pins that will be changed, and 1's for all others
  - logically OR the result with the desired direction value

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin P0.4 as input

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	**1**	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	**0**	0	1	0	0

FIO0DIR Register

Predefined Directions

Mask (**AND**)

Desired Value (**OR**)

Final value

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin P0.4 as input
  - FIO0DIR &= 0xFFFFFFEF; [4th bit is 0]
  - FIO0DIR |= 0x00000000; //Can skip

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin P0.5 as output

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	**0**	1	0	1	0	0	Predefined Directions
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1	1	1	1	Mask (**AND**)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	Desired Value (**OR**)
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	**1**	0	0	1	0	0	Final value

FIO0DIR Register

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin P0.5 as output
  - FIO0DIR &= 0xFFFFFFDF; //Can skip
  - FIO0DIR |= 0x00000020;

Ref:

- https://www.youtube.com/watch?v=5ipqQsguqdE -- FIOxDIR
- https://www.youtube.com/watch?v=WmZFfokztXg Pin Connect block
- **Chapter 8 and 9 in the user manual**

UM10360 pdf, UM10360 Description, UM10360 Datasheet, UM10360 view ::: ALLDATASHEET :::

GPIO port Pin value register FIOxPIN (FIO0PIN to FIO4PIN)

- This register provides the value of port pins that are configured to perform only digital functions.
- The register will give the logic value of the pin regardless of whether the pin is configured for input or output, or as GPIO or an alternate digital function.
- Any configuration of that pin will allow its current logic state to be read from the corresponding FIOxPIN register.
- Access to a port pin via the FIOxPIN register is conditioned by the corresponding bit of the FIOxMASK register
- Only pins masked with zeros in the Mask register will be correlated to the current content of the Fast GPIO port pin value register.

Fast GPIO port Mask register FIOxMASK (FIO0MASK to FIO4MASK)

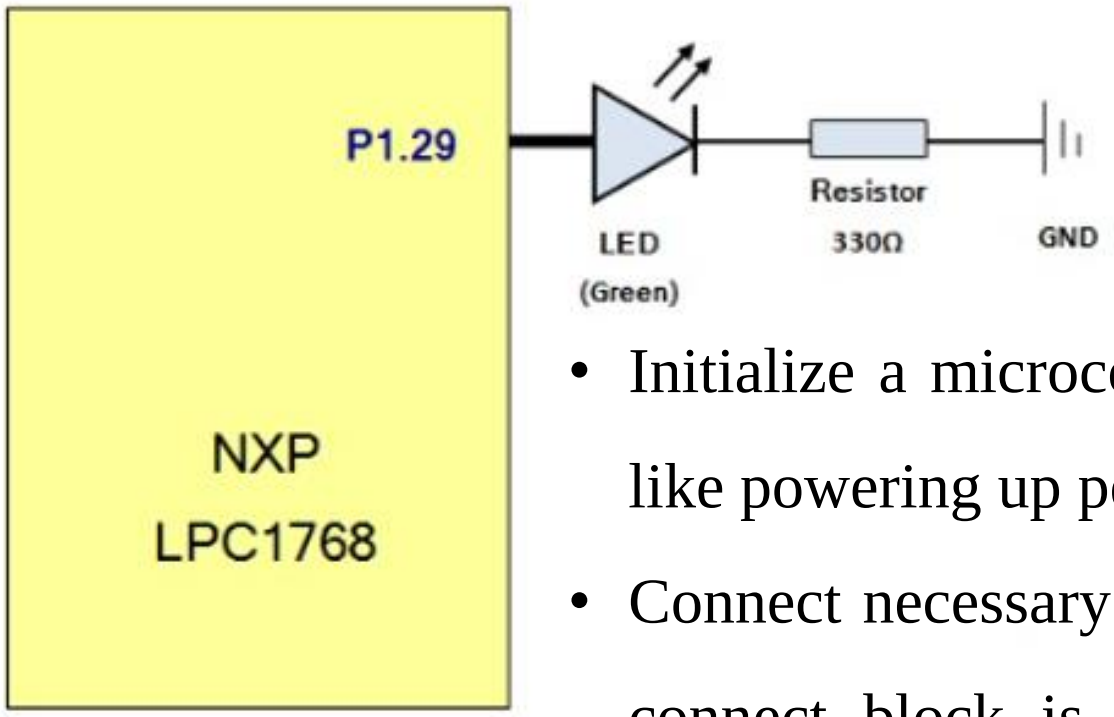
- This register is used to select port pins that will and will not be affected by write accesses to the FIOxPIN, FIOxSET or FIOxCLR register. Mask register also filters out port's content when the FIOxPIN register is read.
- A zero in this register's bit enables an access to the corresponding physical pin via a read or write access. If a bit in this register is one, corresponding pin will not be changed with write access and if read, will not be reflected in the updated FIOxPIN register.

Table 112. Fast GPIO port Mask register (FIO0MASK to FIO4MASK - addresses 0x2009 C010 to 0x2009 C090) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0MASK FIO1MASK FIO2MASK FIO3MASK FIO4MASK		Fast GPIO physical pin access control.	0x0
		0	Controlled pin is affected by writes to the port's FIOxSET, FIOxCLR, and FIOxPIN register(s). Current state of the pin can be read from the FIOxPIN register.	
		1	Controlled pin is not affected by writes into the port's FIOxSET, FIOxCLR and FIOxPIN register(s). When the FIOxPIN register is read, this bit will not be updated with the state of the physical pin.	

INTERFACING LED TO ARM MICROCONTROLLER

LED to pin P1.29 of LPC1768 Microcontroller as shown in circuit diagram.



General steps for an application to blink LED using LPC1768 Micro Controller

- Initialize a microcontroller system, which take care setup procedure like powering up peripherals, set clock rate etc.
- Connect necessary pins using pin connect block. The purpose of pin connect block is to configure the microcontroller pin to desired function.
- And then set or clear the bit of respective pin to turn ON & OFF LED.

```c
#include <lpc17xx.h>

void delay_ms(unsigned int ms)
{
    unsigned int i,j;

    for(i=0;i<ms;i++)
        for(j=0;j<20000;j++);
}

/* start the main program */
int main()
{
    SystemInit();                          //Clock and PLL configuration
    LPC_PINCON->PINSEL4 = 0x000000;  //Configure the PORT2 Pins as GPIO;
    LPC_GPIO2->FIODIR = 0xffffffff;  //Configure the PORT2 pins as OUTPUT;

    while(1)
    {
        LPC_GPIO2->FIOSET = 0xffffffff;      // Make all the Port pins as high
        delay_ms(100);

        LPC_GPIO2->FIOCLR = 0xffffffff;      // Make all the Port pins as low
        delay_ms(100);
    }
}
```

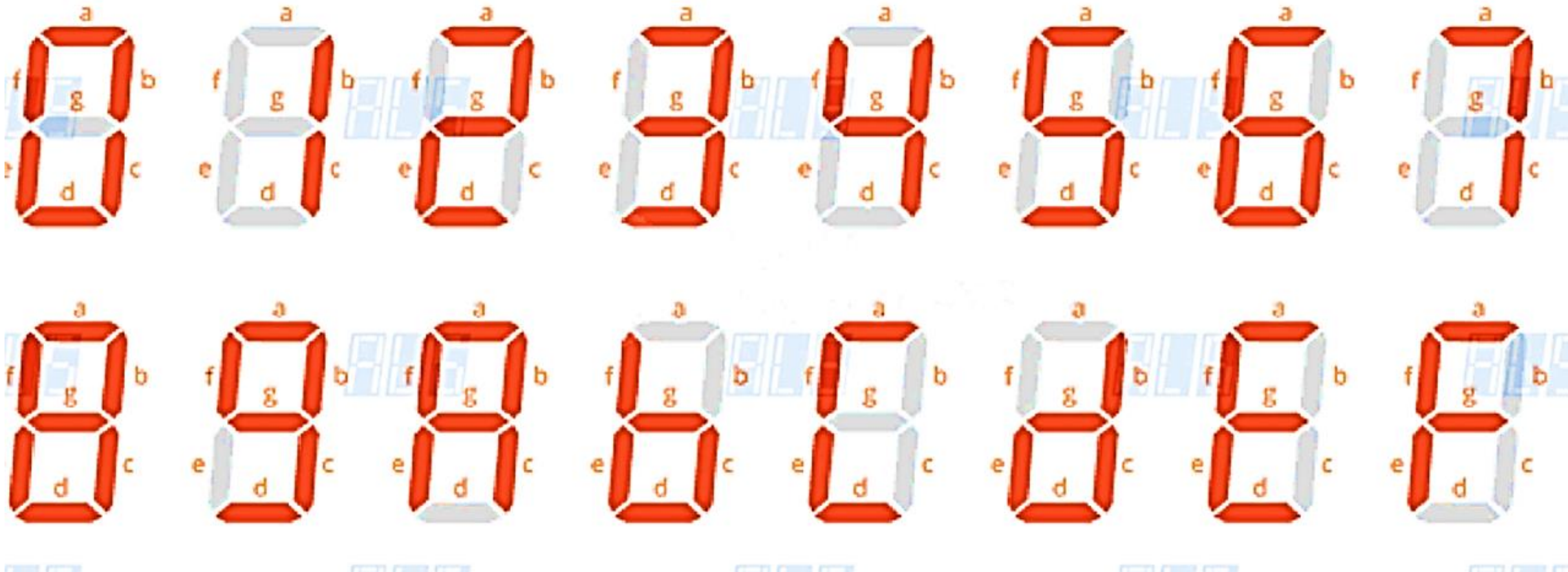
INTERFACING SEVEN SEGMENT TO ARM MICROCONTROLLER

Working of Seven Segment Displays

- The number 8 is displayed when the power is given to all the segments and if you disconnect the power for ‘g’, then it displays the number 0.
- In a seven-segment display, power (or voltage) at different pins can be applied at the same time, so we can form combinations of display numerical from 0 to 9.
- Each display unit has a Dot/Decimal point (DP).

Since seven-segment displays cannot form alphabets like X and Z, so it cannot be used for the alphabet, and they can be used only for displaying decimal numerical magnitudes.

However, seven-segment displays can form alphabets A, B, C, D, E, and F.


a
f
b
g
e
c
d

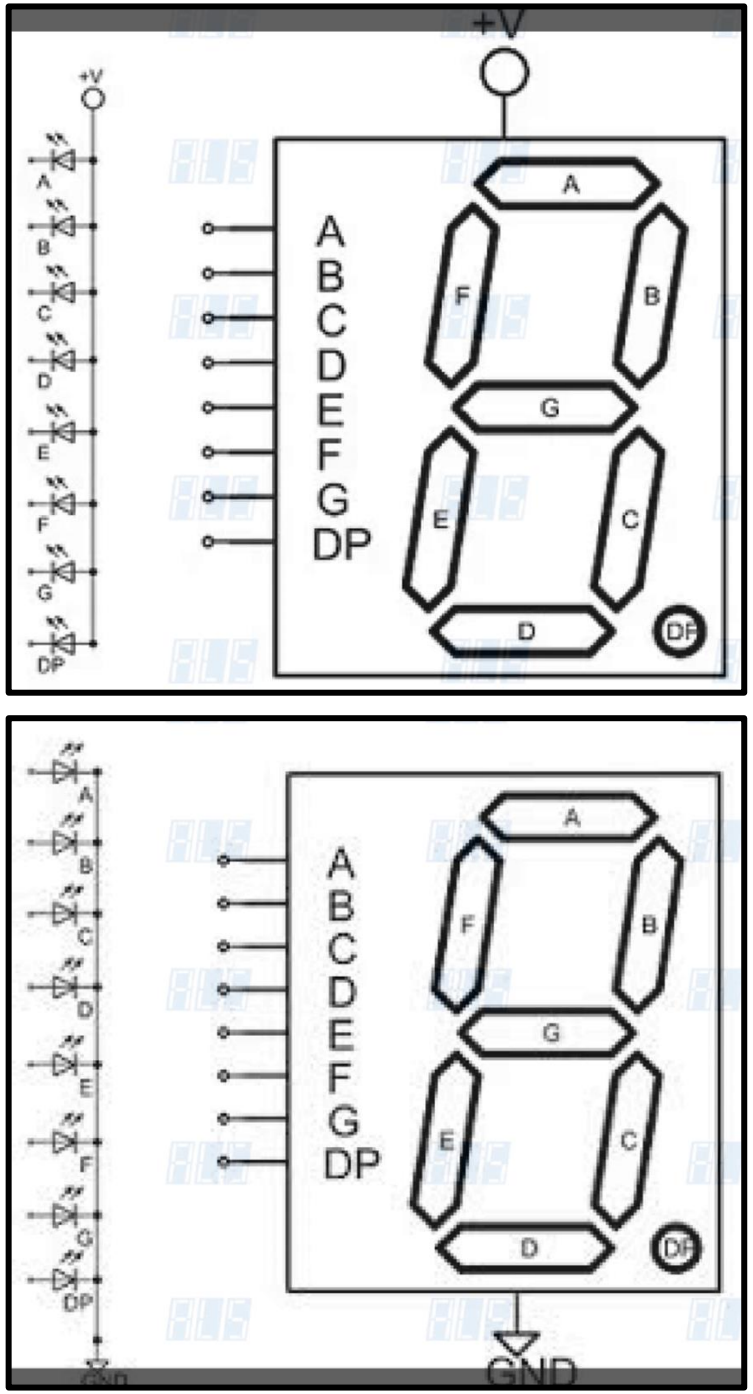
Types of Seven Segment Displays

Two types of configurations of seven-segment displays

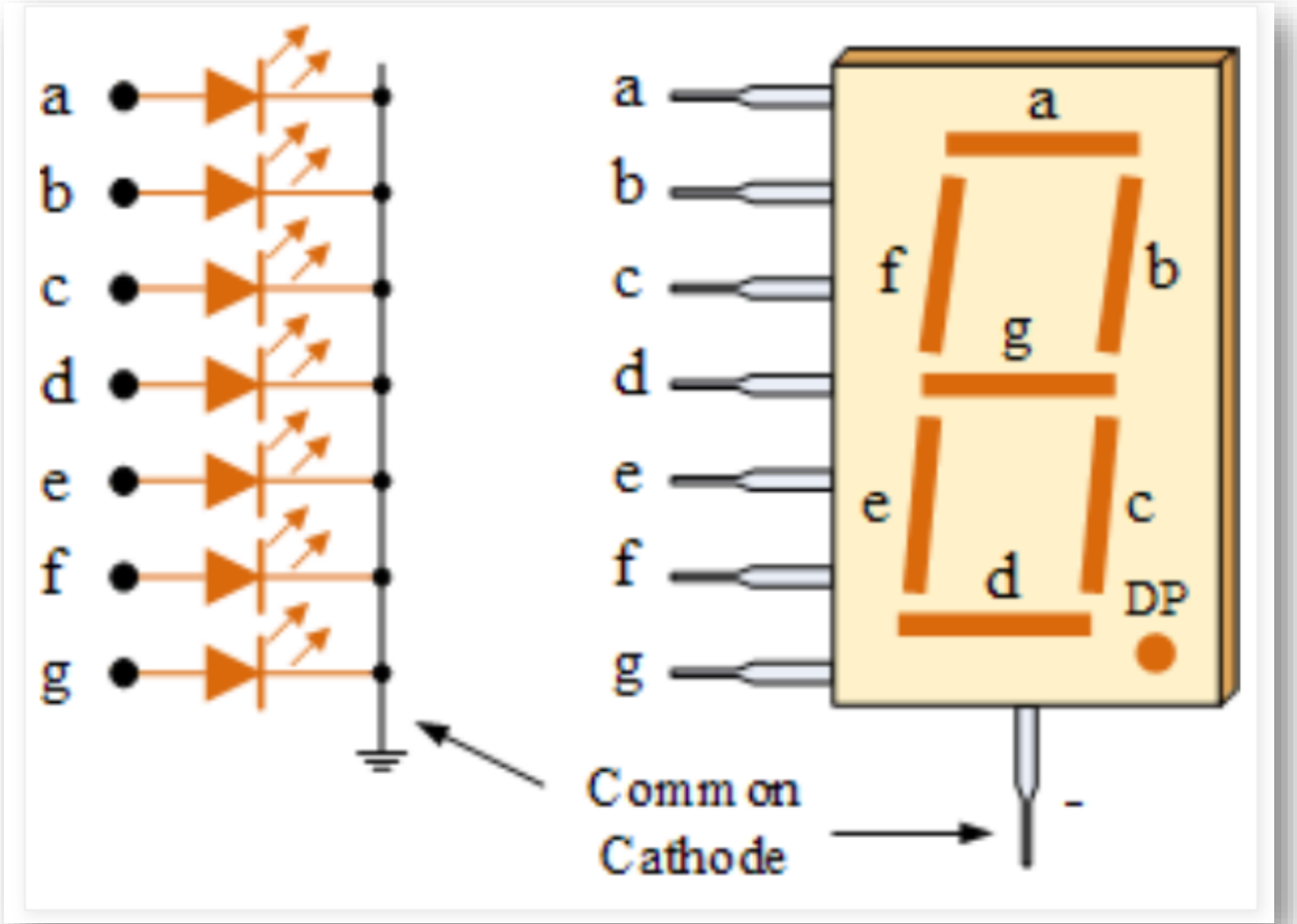
- Common anode display
- Common cathode display

Common Cathode Seven Segment Displays, all the cathode connections of LED segments are connected to logic 0 or ground.
We use logic 1 through a current limiting resistor to forward bias the individual anode terminals a to g.

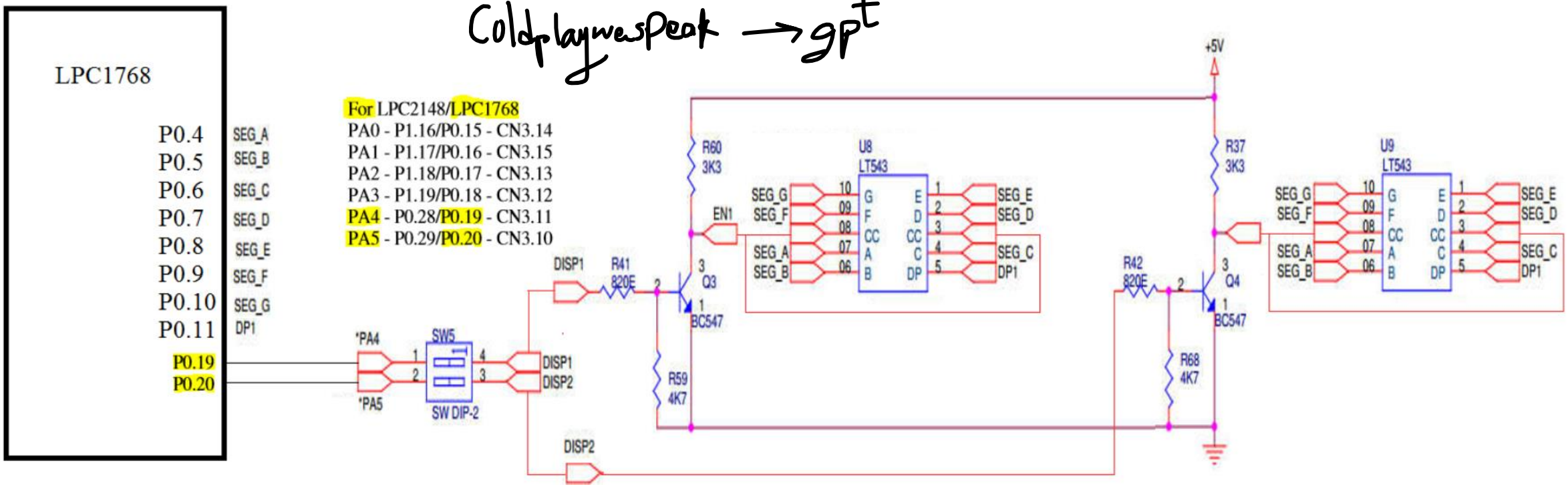
Anode connections of the LED segments are connected to logic 1 in a common anode seven segment display. We use logic 0 through a current limiting resistor to the cathode of a particular segment a to g.

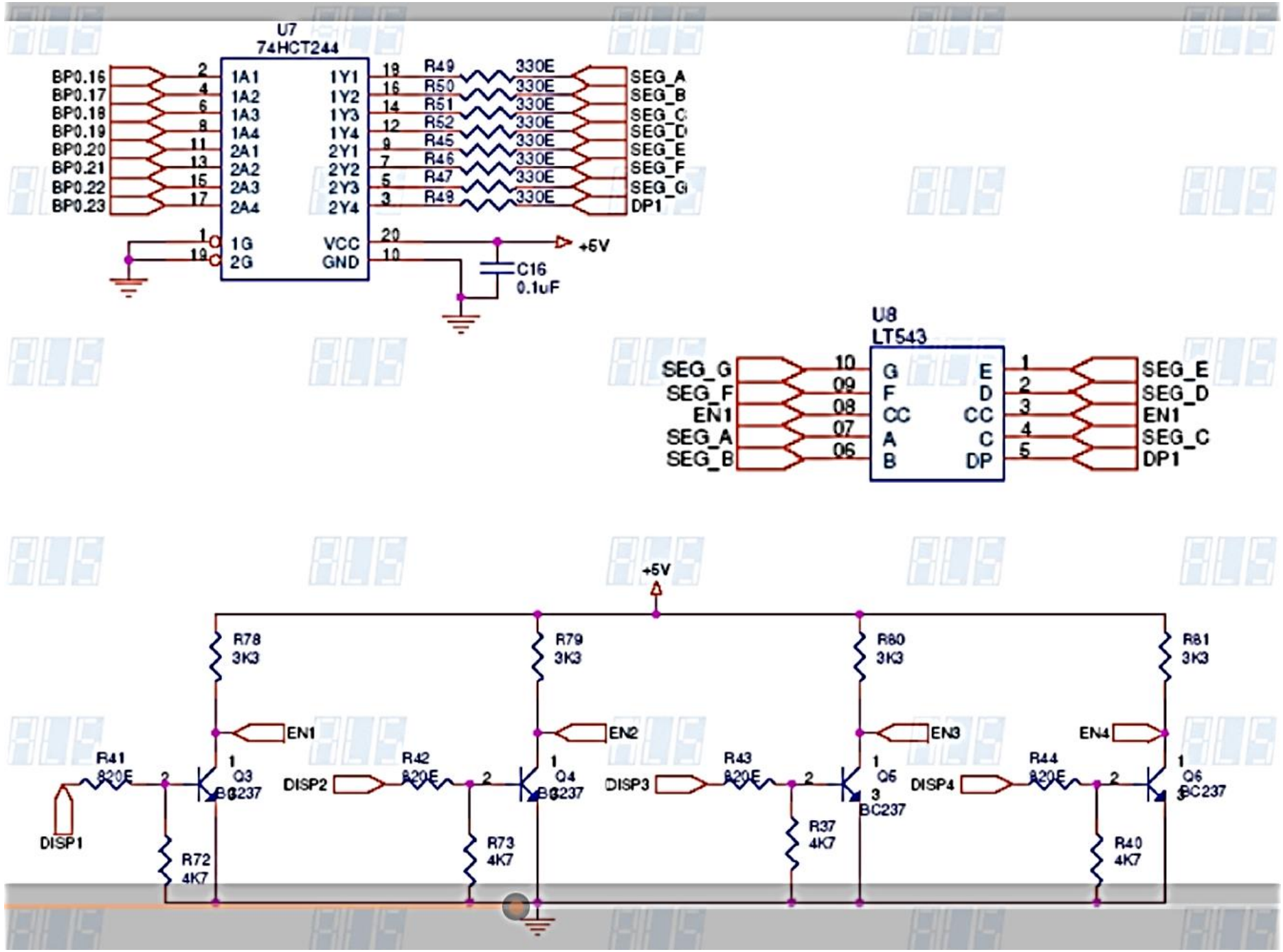


DP	G	F	E	D	C	B	A	DISPLAY VALUE	HEX VALUE
0	0	1	1	1	1	1	1	0	0X3F
0	0	0	0	0	1	1	0	1	0X06
0	1	0	1	1	0	1	1	2	0X5B
0	1	0	0	1	1	1	1	3	0X4F
0	1	1	0	0	1	1	0	4	0X66
0	1	1	0	1	1	0	1	5	0X6D
0	1	1	1	1	1	0	1	6	0X7D
0	0	0	0	0	1	1	1	7	0X07
0	1	1	1	1	1	1	1	8	0X7F
0	1	1	0	1	1	1	1	9	0X6F
0	1	1	1	0	1	1	1	A	0X77
0	1	1	1	1	1	0	0	B	0X7C
0	0	1	1	1	0	0	1	C	0X39
0	1	0	1	1	1	1	0	D	0X5E
0	1	1	1	1	0	0	1	E	0X79
0	1	1	1	0	0	0	1	F	0X71


a
b
c
d
e
f
g
a
b
c
d
e
f
g
a
f
b
g
e
c
d
DP
Common
Cathode
-

Coldplay vs Peak → gpt



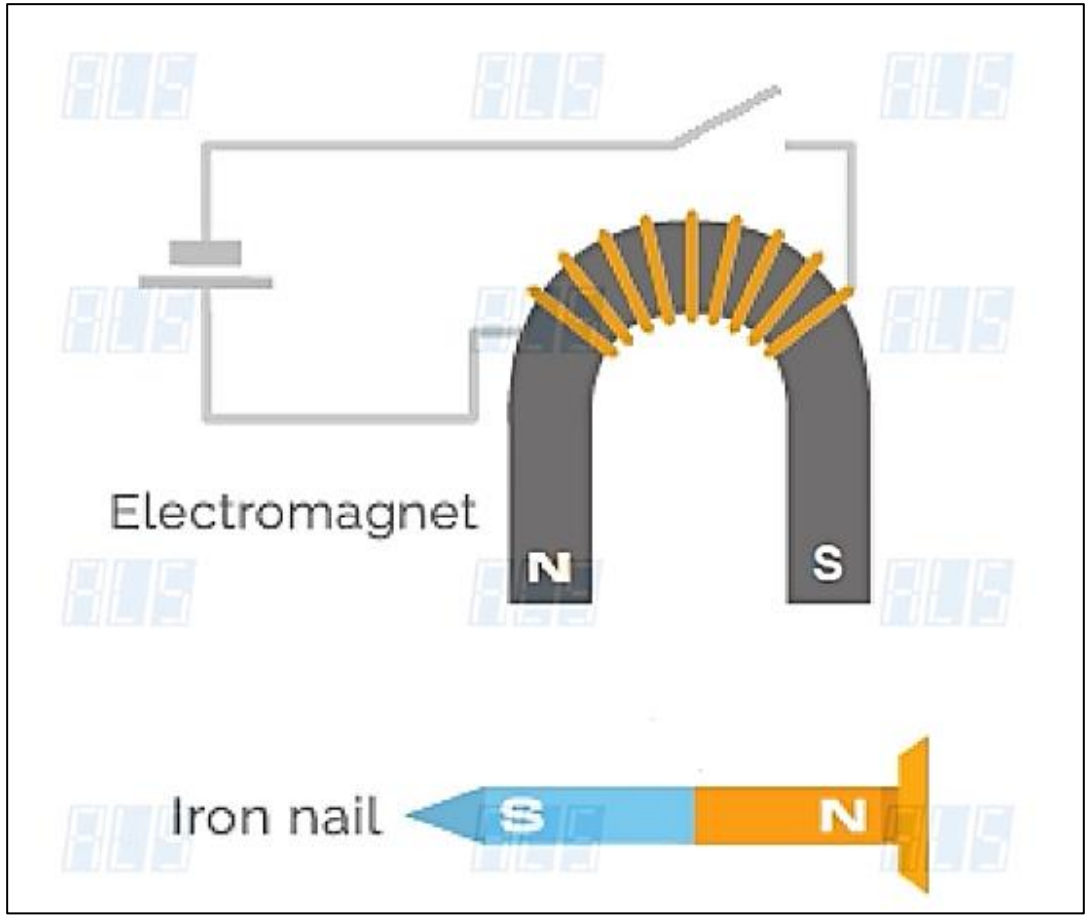

U7
74HCT244
BP0.16
BP0.17
BP0.18
BP0.19
BP0.20
BP0.21
BP0.22
BP0.23
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1G
2G
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
VCC
GND
R49 330E
R50 330E
R51 330E
R52 330E
R45 330E
R46 330E
R47 330E
R48 330E
SEG_A
SEG_B
SEG_C
SEG_D
SEG_E
SEG_F
SEG_G
DP1
+5V
C16
0.1uF
U8
LT543
SEG_G
SEG_F
EN1
SEG_A
SEG_B
G
F
CC
A
B
E
D
CC
C
DP
SEG_E
SEG_D
EN1
SEG_C
DP1
+5V
R78 3K3
R79 3K3
R80 3K3
R81 3K3
EN1
EN2
EN3
EN4
R41 820E
R42 820E
R43 820E
R44 820E
Q3 BC237
Q4 BC237
Q5 BC237
Q6 BC237
DISP1
DISP2
DISP3
DISP4
R72 4K7
R73 4K7
R37 4K7
R40 4K7

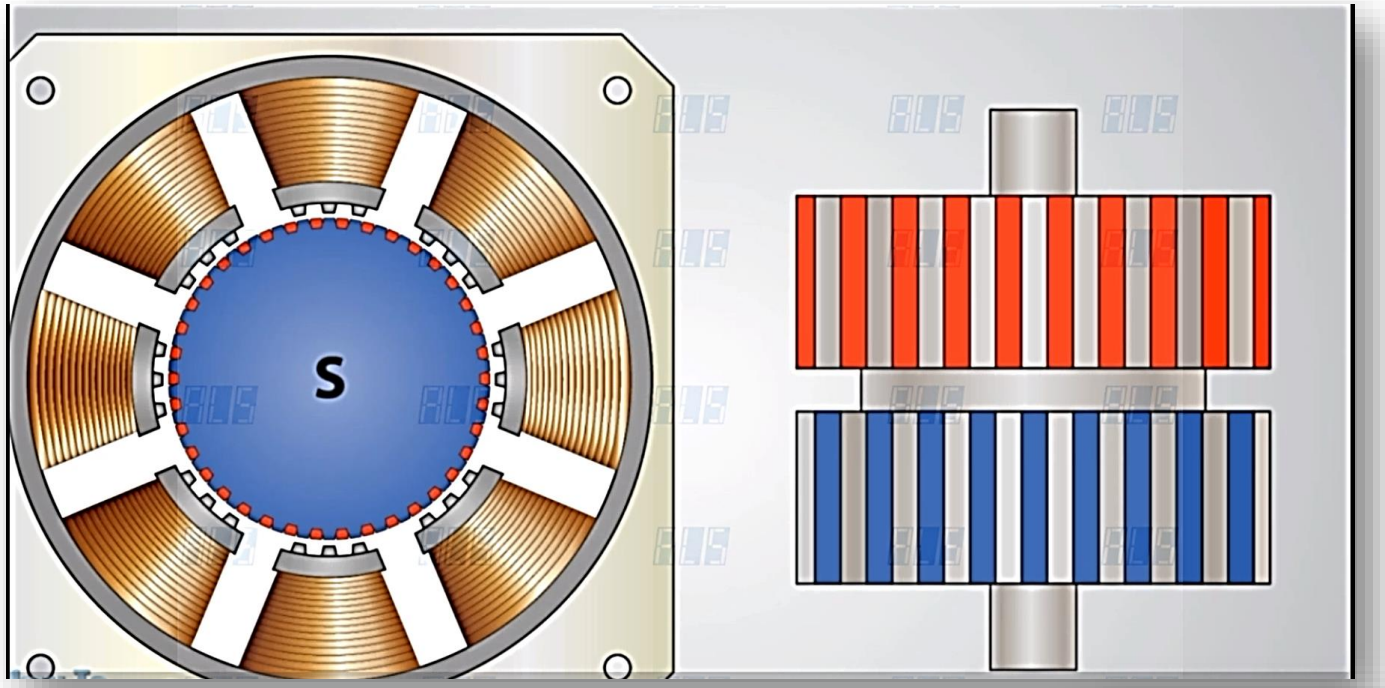
```
/* Description  :\\\\\\\\\\\\\\\DISPLAY ARE CONNECTED IN COMMON CATHODE MODE\\\\\\\\\\\\\\\\\\\\\
  Port0 Connected to data lines of all 7 segement displays
     a
    ----
  f|  g |b
   |----|
  e|    |c
    ----  . dot
     d
  a = P0.04
  b = P0.05
  c = P0.06
  d = P0.07
  e = P0.08
  f = P0.09
  g = P0.10
  dot = P0.11
  */
```

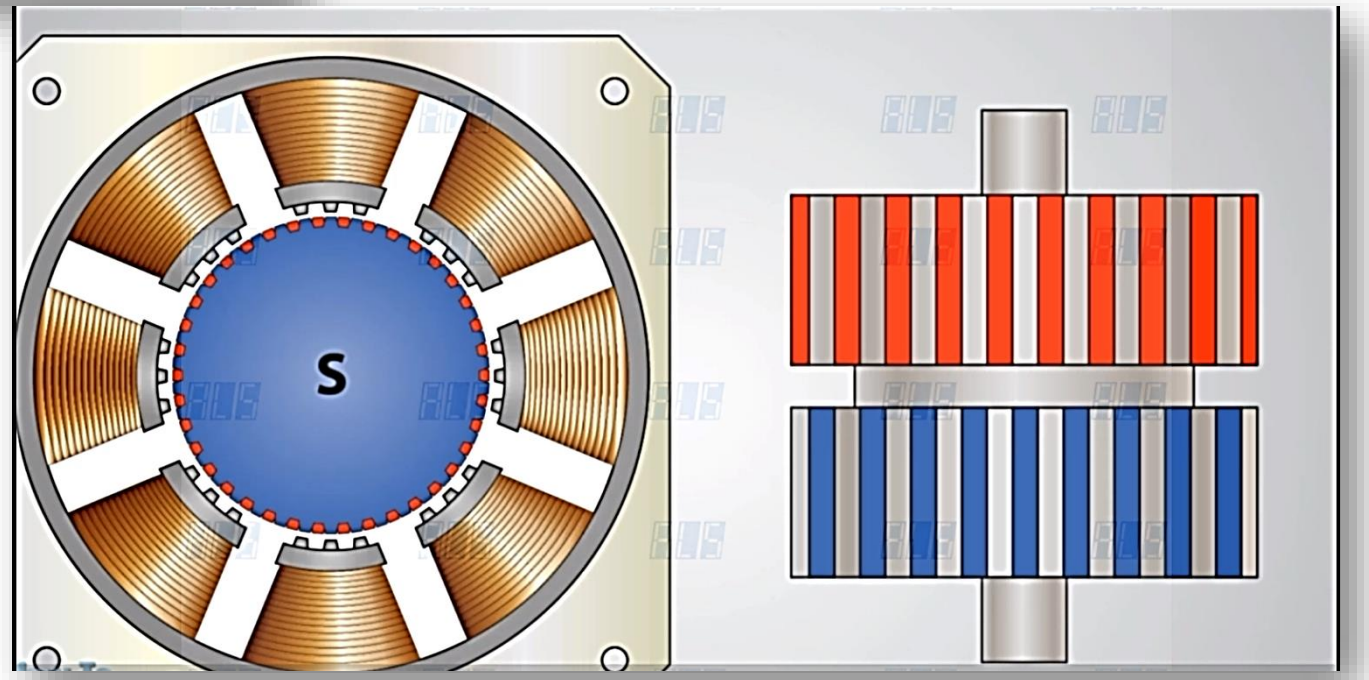
```c
#include <LPC17xx.h>
unsigned int delay, count=0, Switchcount=0,j;

unsigned int Disp[16]={0x000003f0, 0x00000060, 0x000005b0, 0x000004f0, 0x00000660,0x000006d0,
             0x000007d0, 0x00000070, 0x000007f0, 0x000006f0, 0x00000770,0x000007c0,
             0x00000390, 0x000005e0, 0x00000790, 0x00000710 };

#define ALLDISP  0x00180000                          //Select all display
#define DATAPORT 0x00000ff0                          //P0.4 to P0.11 : Data lines connected to drive Seven Segments
int main (void)
{
  LPC_PINCON->PINSEL0 = 0x00000000;
  LPC_PINCON->PINSEL1 = 0x00000000;
  LPC_GPIO0->FIODIR = 0x00180ff0;

  while(1)
  {
    LPC_GPIO0->FIOSET |= ALLDISP;
    LPC_GPIO0->FIOCLR = 0x00000ff0;                  // clear the data lines to 7-segment displays
    LPC_GPIO0->FIOSET = Disp[Switchcount];           // get the 7-segment display value from the array

       for(j=0;j<3;j++)
      for(delay=0;delay<30000;delay++);              // delay

        Switchcount++;
        if(Switchcount == 0x10)                      // 0 to F has been displayed ? go back to 0
          {
           Switchcount = 0;
           LPC_GPIO0->FIOCLR  = 0x00180ff0;
          }
  }
```

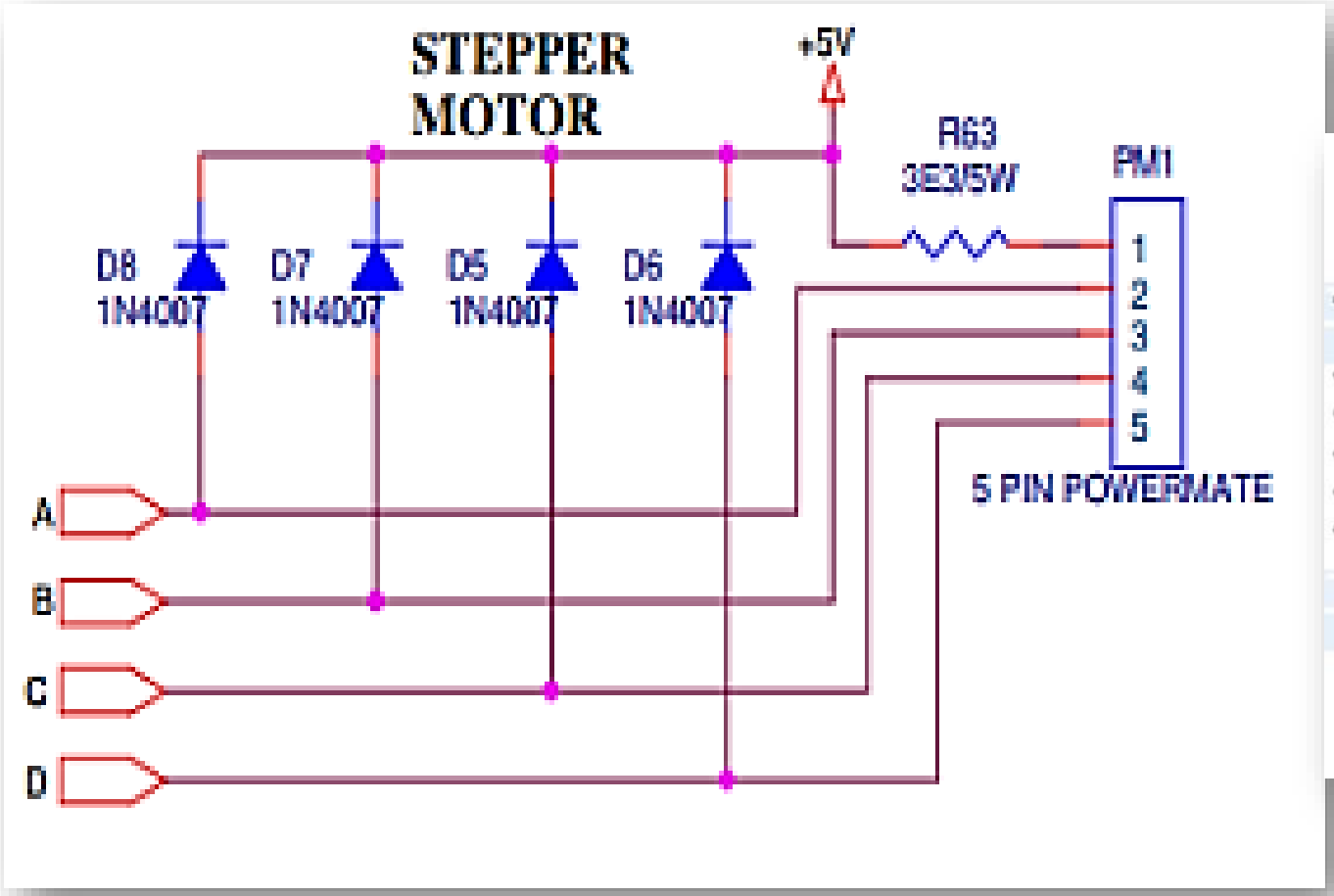
INTERFACING STEPPER MOTOR TO ARM MICROCONTROLLER

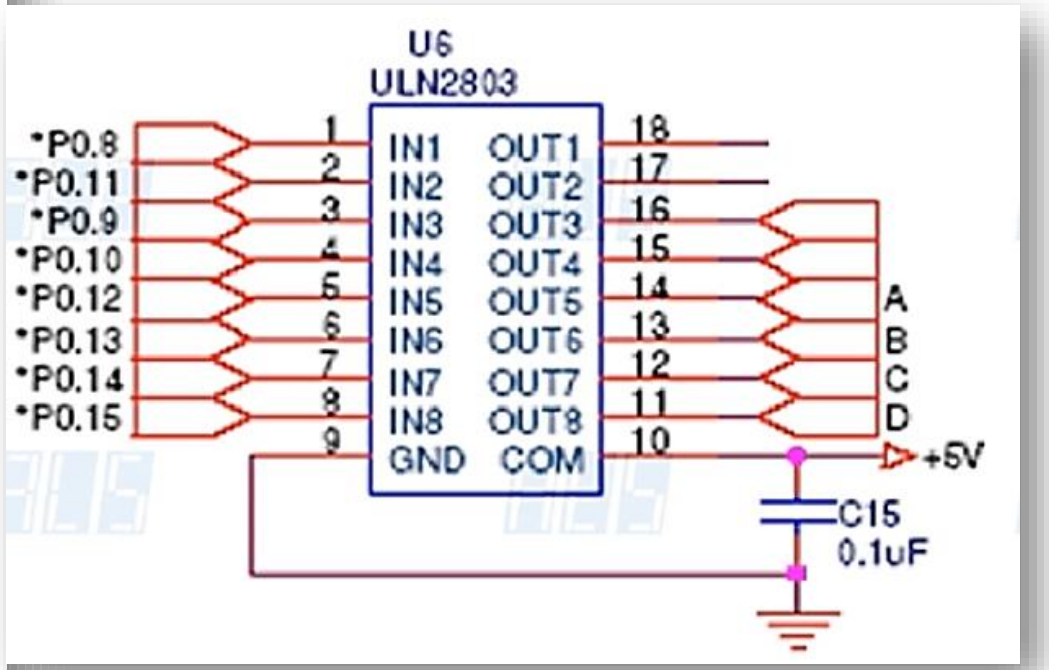
- Stepper motors are the motors that move in discrete steps or convert electrical pulses into rotatory motion.
- They have multiple coils (4coils) that are organized in groups called "phases"(stators named as A,B,C and D).
- By energizing each phase in sequence, the motor will rotate, one step at a time.




S

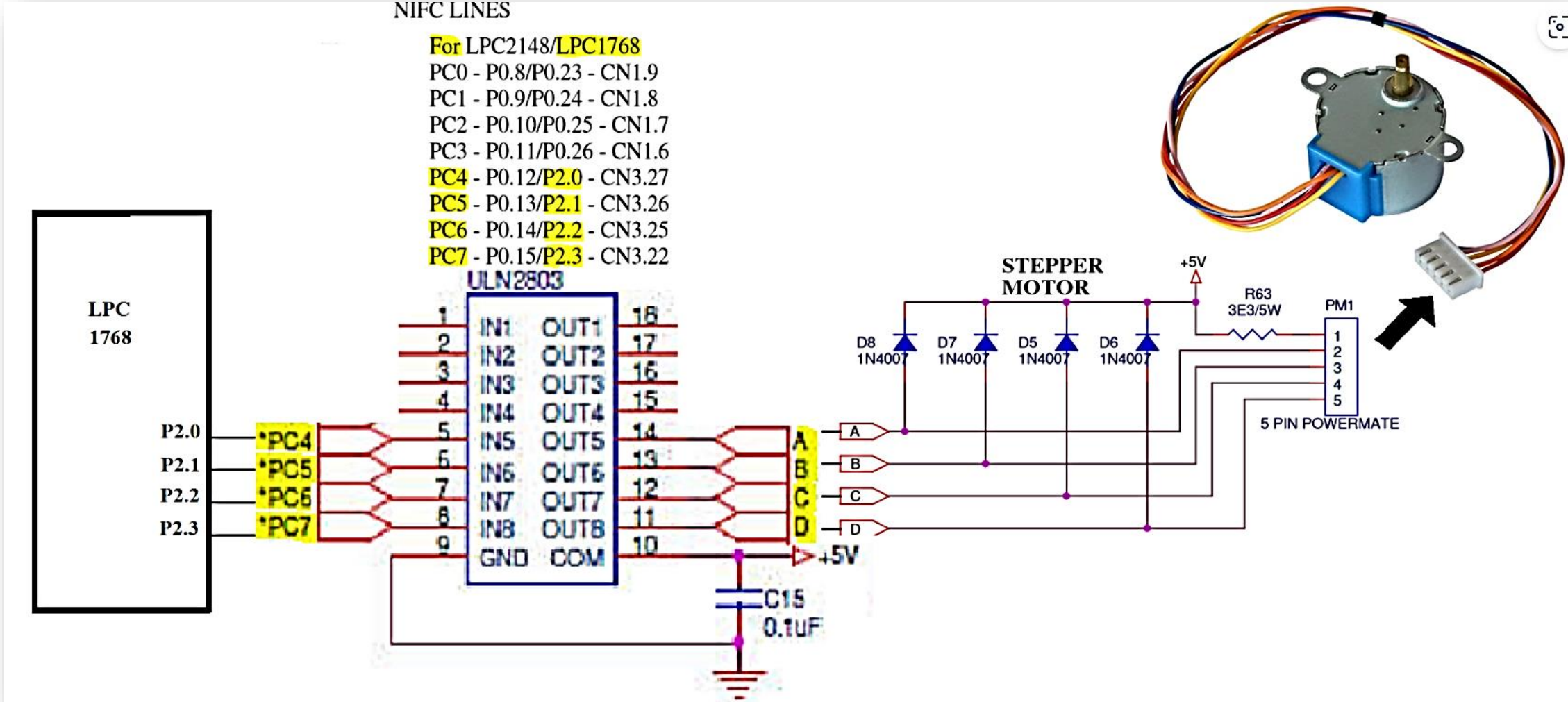

S

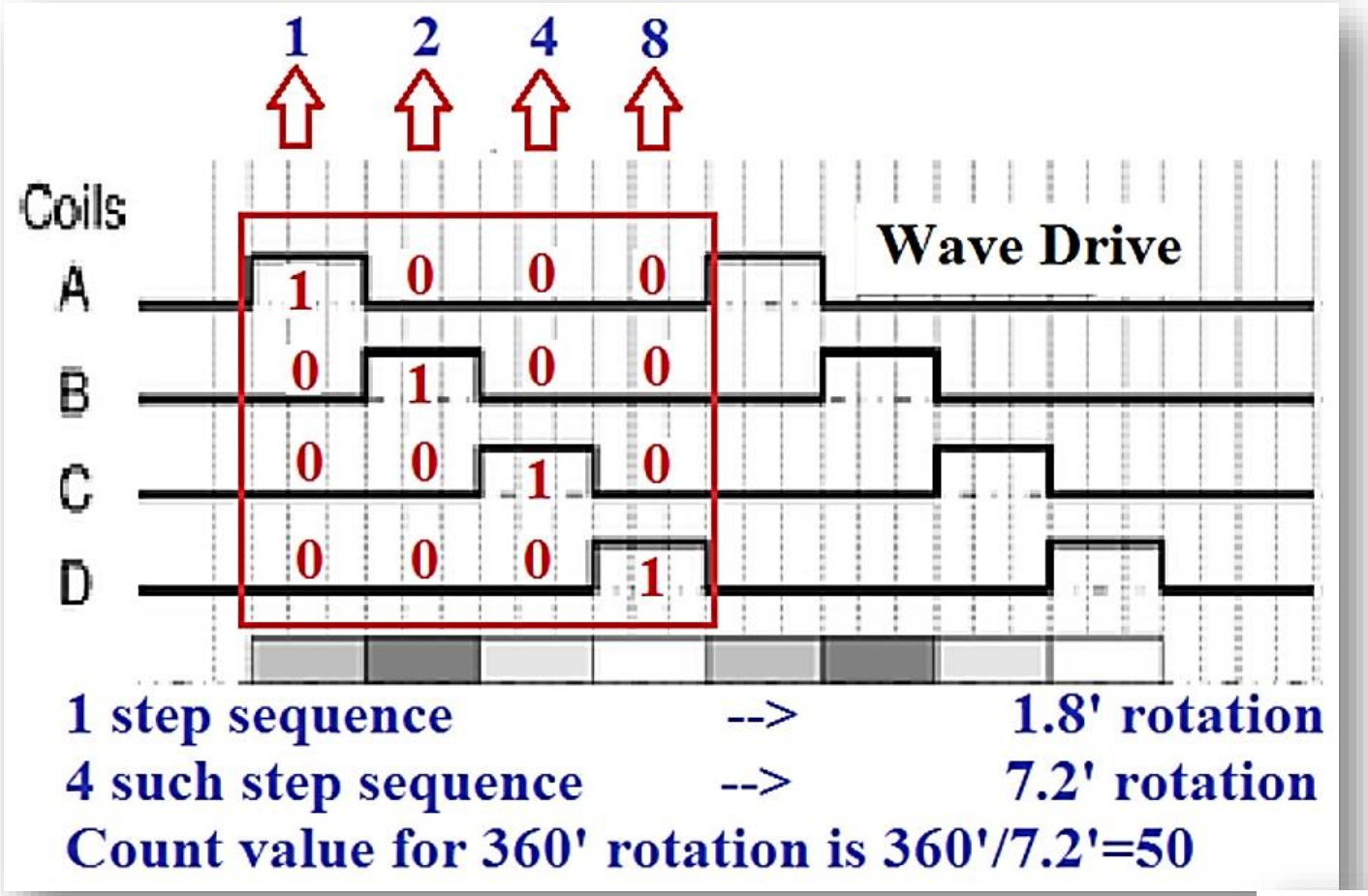

STEPPER
MOTOR
+5V
R63
3E3/5W
PM1
D8
1N4007
D7
1N4007
D5
1N4007
D6
1N4007
1
2
3
4
5
5 PIN POWERMATE
A
B
C
D


U6
ULN2803
*P0.8
*P0.11
*P0.9
*P0.10
*P0.12
*P0.13
*P0.14
*P0.15
1
2
3
4
5
6
7
8
9
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
GND
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
COM
18
17
16
15
14
13
12
11
10
A
B
C
D
+5V
C15
0.1uF

NIFC LINES

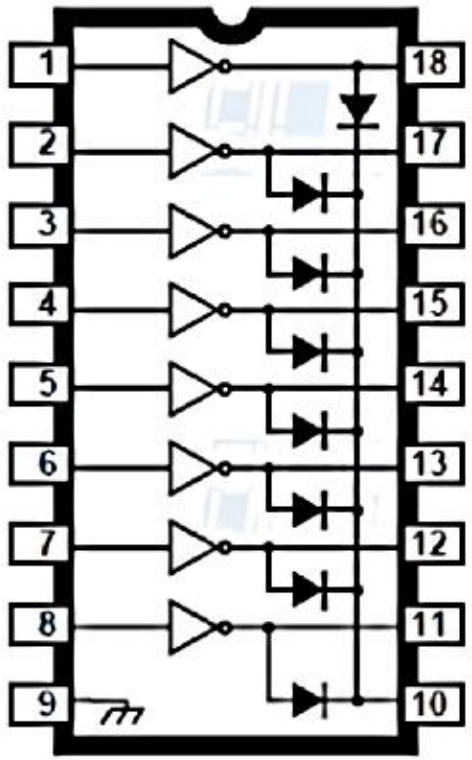
For LPC2148/LPC1768
PC0 - P0.8/P0.23 - CN1.9
PC1 - P0.9/P0.24 - CN1.8
PC2 - P0.10/P0.25 - CN1.7
PC3 - P0.11/P0.26 - CN1.6
PC4 - P0.12/P2.0 - CN3.27
PC5 - P0.13/P2.1 - CN3.26
PC6 - P0.14/P2.2 - CN3.25
PC7 - P0.15/P2.3 - CN3.22





Pin no	Description
1	+5v supply
2	Phase A
3	Phase B
4	Phase C
5	Phase D

P0.15	P0.14	P0.13	P0.12
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0



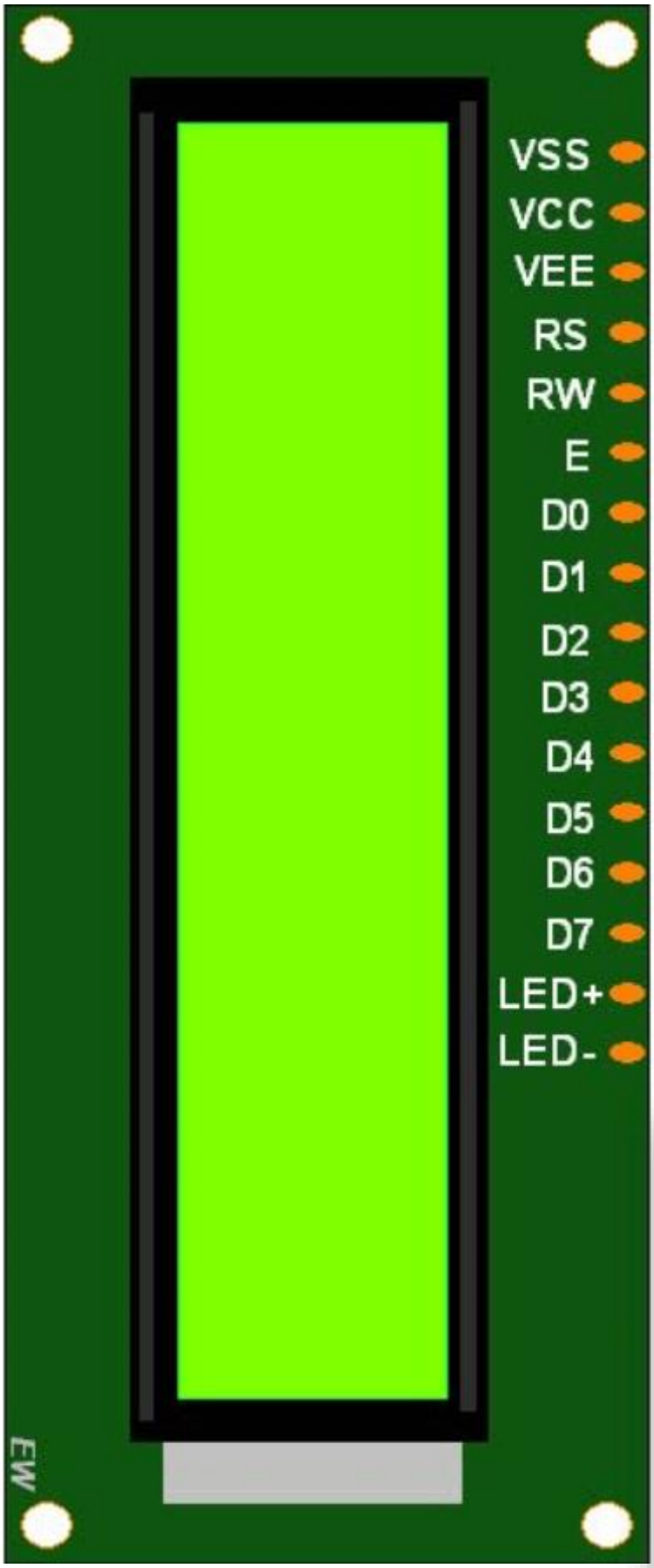
D	C	B	A
1	1	1	0
1	1	0	1
1	0	1	1
0	1	1	1

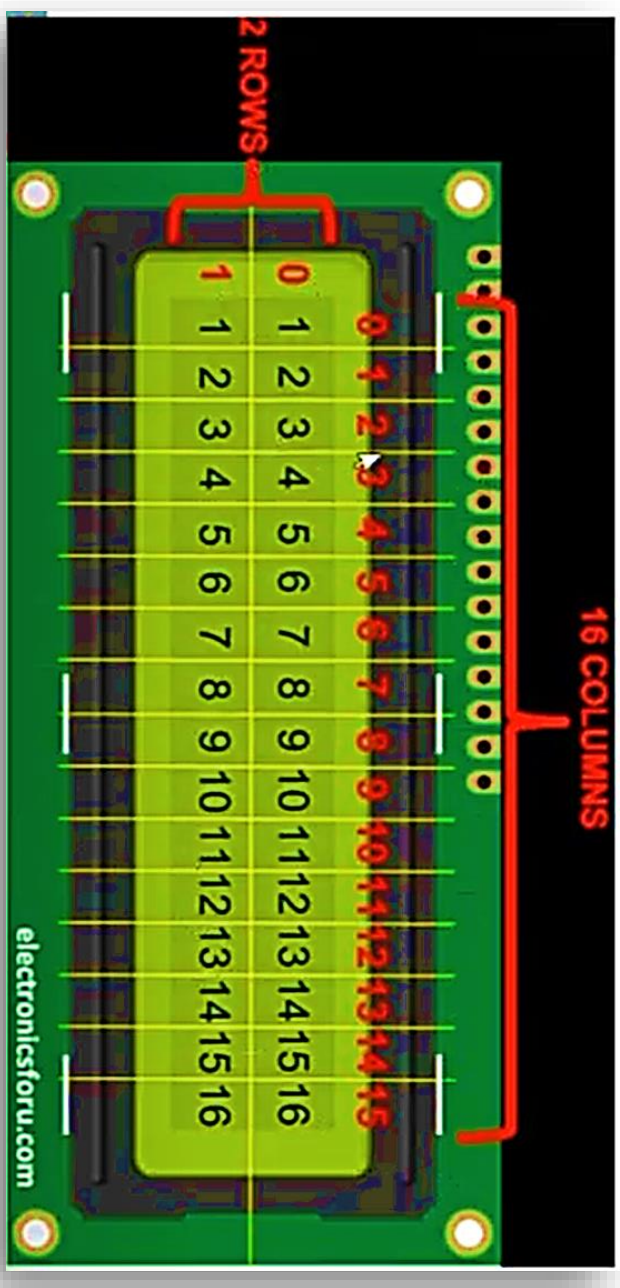
```c
#include <LPC17xx.H>
void clock_wise(void);
void anti_clock_wise(void);
unsigned long int var1,var2;
unsigned int i=0,j=0,k=0;

int main(void)
{
  LPC_PINCON->PINSEL4 = 0x00000000;    //P2.0 to P2.3 GPIO
  LPC_GPIO2->FIODIR = 0x0000000F;      //P2.0 to P2.3 output
  while(1)
  {
    for(j=0;j<30;j++)                  //50 times in Clock wise Rotation
      clock_wise();

    for(k=0;k<50000;k++);              //Delay to show  anti_clock Rotation

    for(j=0;j<30;j++)                  //50 times in  Anti Clock wise Rotation
      anti_clock_wise();

    for(k=0;k<50000;k++) ;             //Delay to show clock Rotation
  }                                    //End of while(1)
}                                      //End of main
```

```
void clock_wise(void)
{
    var1 = 0x00000001;                  //For Clockwise
    for(i=0;i<=3;i++)                   //for A B C D Stepping
  {
     LPC_GPIO2->FIOCLR =  0X0000000F;
     LPC_GPIO2->FIOSET =  var1;
     var1 = var1<<1;                    //For Clockwise
     for(k=0;k<15000;k++) ;             //for step speed variation
  }
}
```

```
void anti_clock_wise(void)
{
    var1 = 0x0000008;                   //For Anticlockwise
    for(i=0;i<=3;i++)                   //for A B C D Stepping
    {
      LPC_GPIO2->FIOCLR =  0X0000000F;
      LPC_GPIO2->FIOSET =  var1;
      var1 = var1>>1;                   //For Anticlockwise
      for(k=0;k<15000;k++);             //for step speed variation
    }
}
```

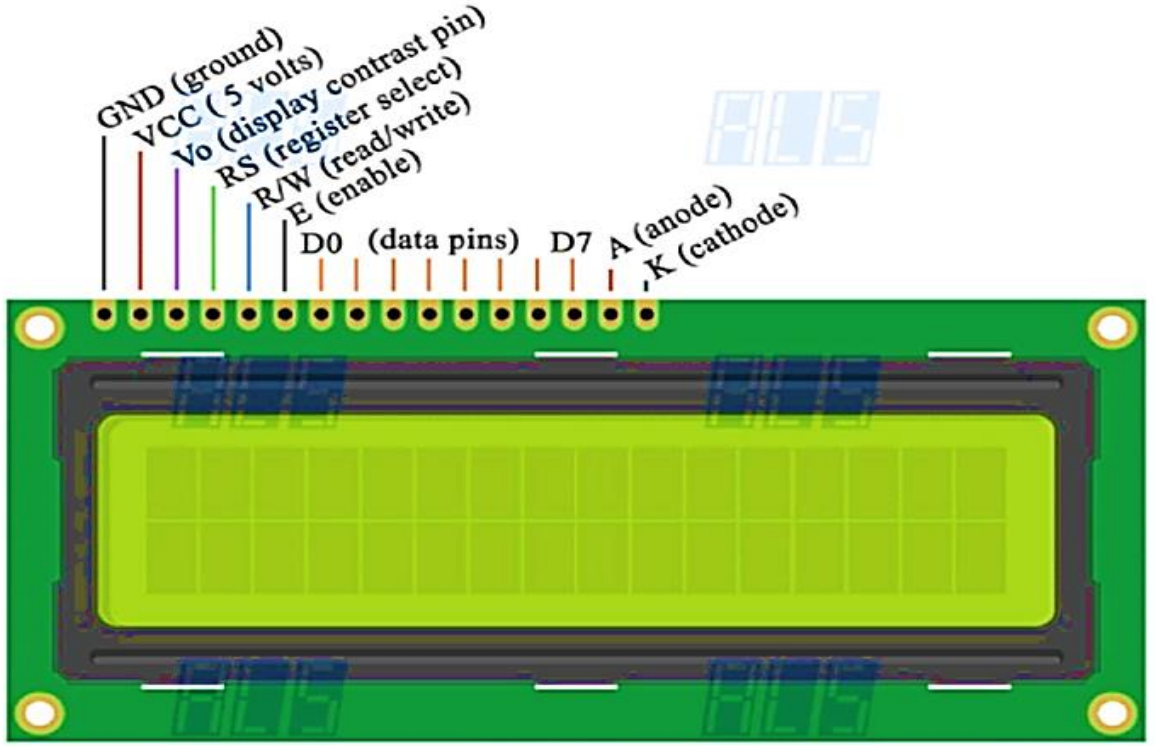
INTERFACING LCD
TO ARM MICROCONTROLLER


VSS
Ground
VCC
+5V
VEE
Contrast control
RS
Register select
RW
Read/write
E
Enable
D0
Data Pin 0
D1
Data Pin 1
D2
Data Pin 2
D3
Data Pin 3
D4
Data Pin 4
D5
Data Pin 5
D6
Data Pin 6
D7
Data Pin 7
LED+
LED+ 5V
LED-
LED- Ground
EW


2 ROWS
16 COLUMNS
electronicsforu.com

Pin Number	Name	Use
1	Vss	Ground
2	Vdd	Power
3	Vee	To adjust the contrast
4	RS	1=Data input 0=Instruction input
5	R/W	1=Read from LCD 0=Write to LCD
6	Enable (EN)	From 1 to 0 = Data is written to the LCD
7	DB0	Data Bus Lines
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+	Backlight
16	LED-	

Pin Number	Symbol	Pin Function
1	VSS	Ground
2	VCC	+5v
3	VEE	Contrast adjustment (VO)
4	RS	Register Select. 0:Command, 1: Data
5	R/W	Read/Write, R/W=0: Write & R/W=1: Read
6	EN	Enable. Falling edge triggered
7	D0	Data Bit 0 (Not used in 4-bit operation)
8	D1	Data Bit 1 (Not used in 4-bit operation)
9	D2	Data Bit 2 (Not used in 4-bit operation)
10	D3	Data Bit 3 (Not used in 4-bit operation)
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7/Busy Flag
15	A/LED+	Back-light Anode(+)
16	K/LED-	Back-Light Cathode(-)



Apart from the voltage supply connections the important pins from the programming perspective are:

- **The data lines(8-bit Data bus)**
- **Register select**
- **Read/Write**
- **Enable pin**

Data Bus

- An LCD has **8-bit data bus** referenced as **D0-D7**.
- As it is an 8-bit data bus, we can send the data/cmd to LCD in bytes.
- It also provides the provision to send the the data/cmd in chunks of 4-bit, which is used when there are limited number of GPIO lines on the microcontroller.

Register Select(RS)

The LCD has **two register** namely a **Data register and Command register**.

- **Any data that needs to be displayed on the LCD must be written to the data register of LCD.**
- **Command can be issued to LCD by writing it to Command register of LCD. This signal is used to differentiate the data/cmd received by the LCD.**

❑ If the RS signal is **LOW,** then the LCD interprets the 8-bit info as **Command** and writes it **Command** performs the action as per the command.

❑ If the RS signal is **HIGH,** then the LCD interprets the 8-bit info as **data** and copies it to **data register**. After LCD decodes the data for generating the 5x7 pattern and finally displays on the LCD.

NOTE:

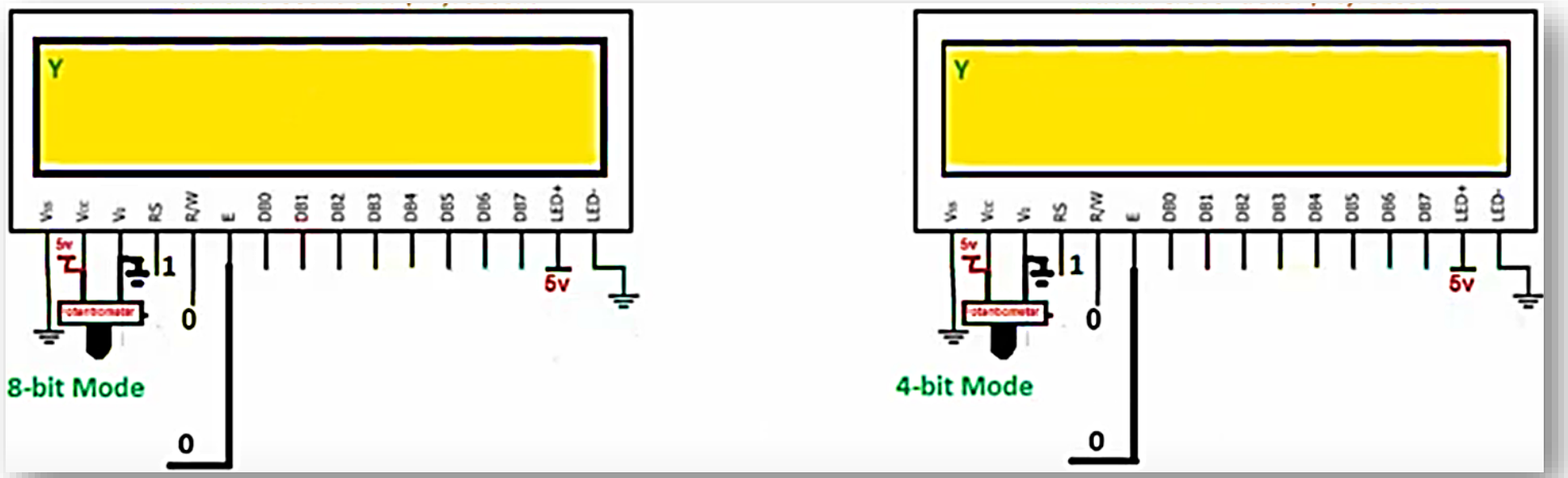
- A character in lcd is generated in a matrix of **5×8 or 5×7**.
- Where 5 represents number of columns and 7/8 represent number of rows.
- Maximum size of the matrix is 5×8.
- You can not display character greater than 5×8-dimension matrix.
- Normally we display a character in 5×7 matrix and left the 8th row for the cursor.
- If we use the 8th row of the matrix for the character display, then there will be no room for cursor.

Read/Write(RW)

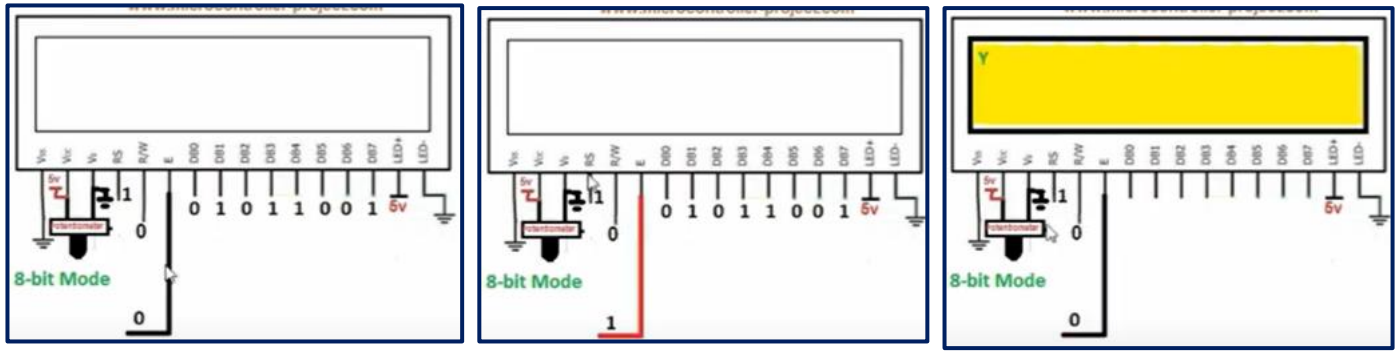
- This signal is used to write the data/cmd to LCD and reads the busy flag of LCD.
- For write operation the RW should be **LOW** and for read operation the R/W should be **HIGH**.

Enable(EN)

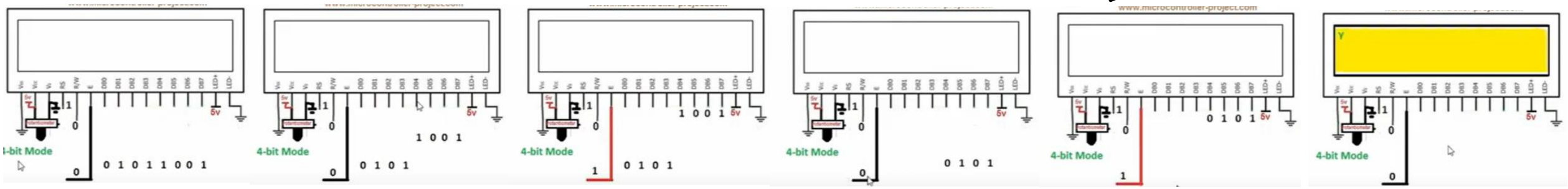
- This pin is used to enable and disable LCD.
- When this pin is active low, LCD controller will be disabled.
- That means control pins and data pins will not have any effect on the display.
- On the other hand, when the enable pin is set to active high, the LCD will work normally and process all data and control instructions. .



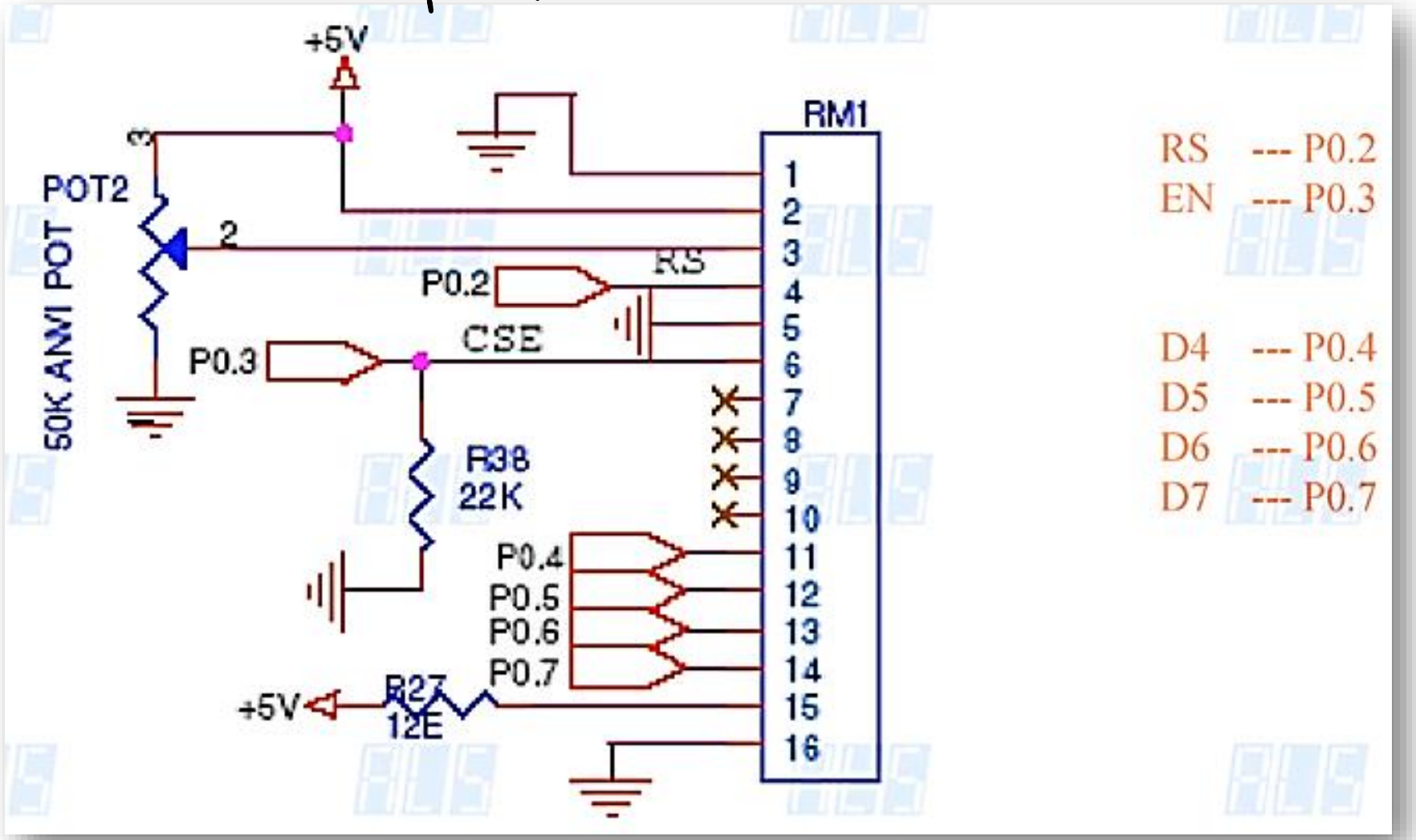
8-bit Mode



4-bit Mode



deepseek

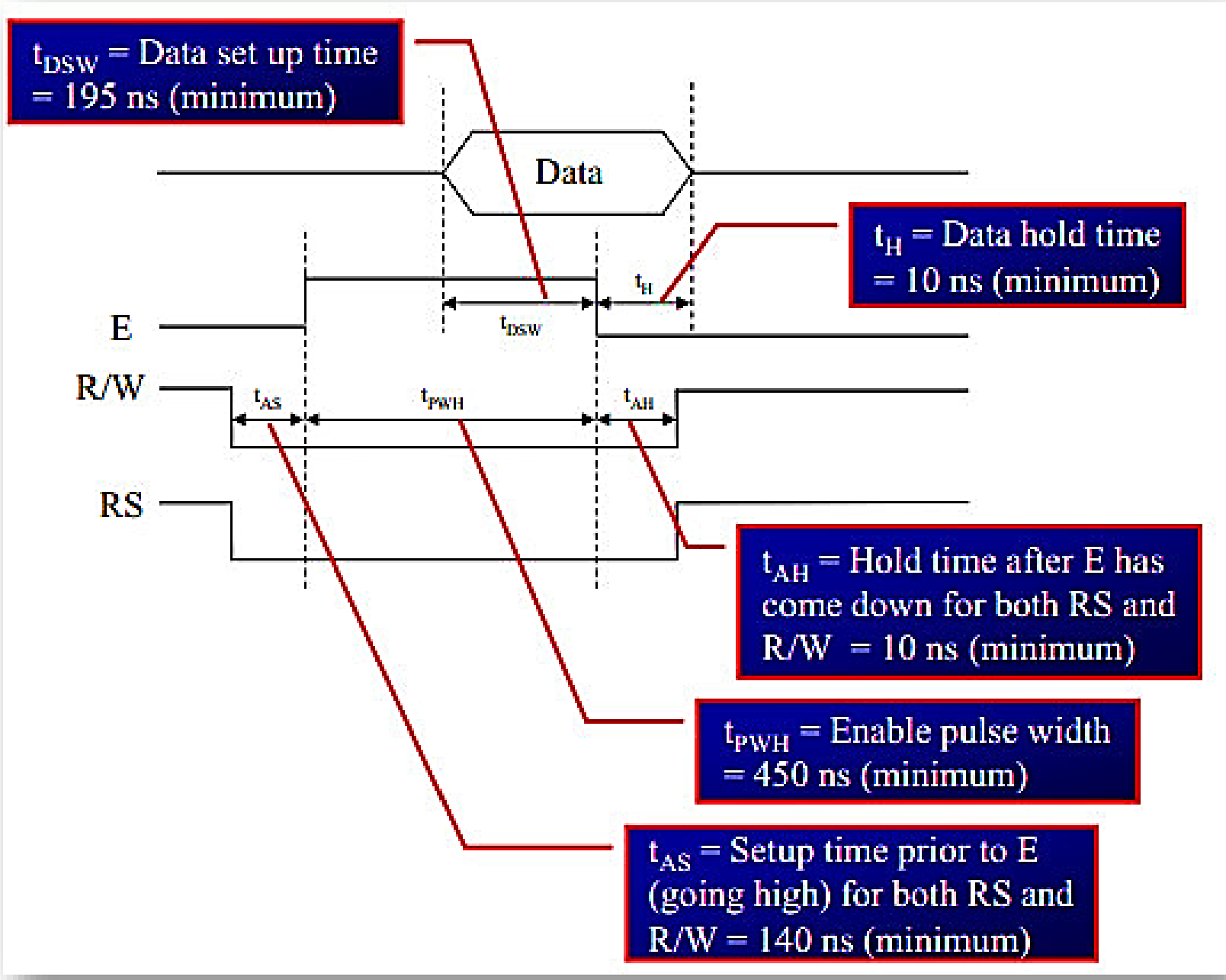

+5V
POT2
50K ANVI POT
2
P0.2
RS
P0.3
CSE
R38
22K
P0.4
P0.5
P0.6
P0.7
+5V
R27
12E
RM1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
RS --- P0.2
EN --- P0.3
D4 --- P0.4
D5 --- P0.5
D6 --- P0.6
D7 --- P0.7

Steps for Sending Command

- Send the I/P command to LCD.
- Select the Control Register by making RS low.
- Select Write operation making RW low.
- Send a High-to-Low pulse on Enable PIN with some delay_us.

Steps for Sending Data

- Send the character to LCD.
- Select the Data Register by making RS high.
- Select Write operation making RW low.
- Send a High-to-Low pulse on Enable PIN with some delay_us.



- LCD display takes a time of 39-43µS to place a character or execute a command, except for clearing display and to seek cursor to home position which takes 1.53ms to 1.64ms.

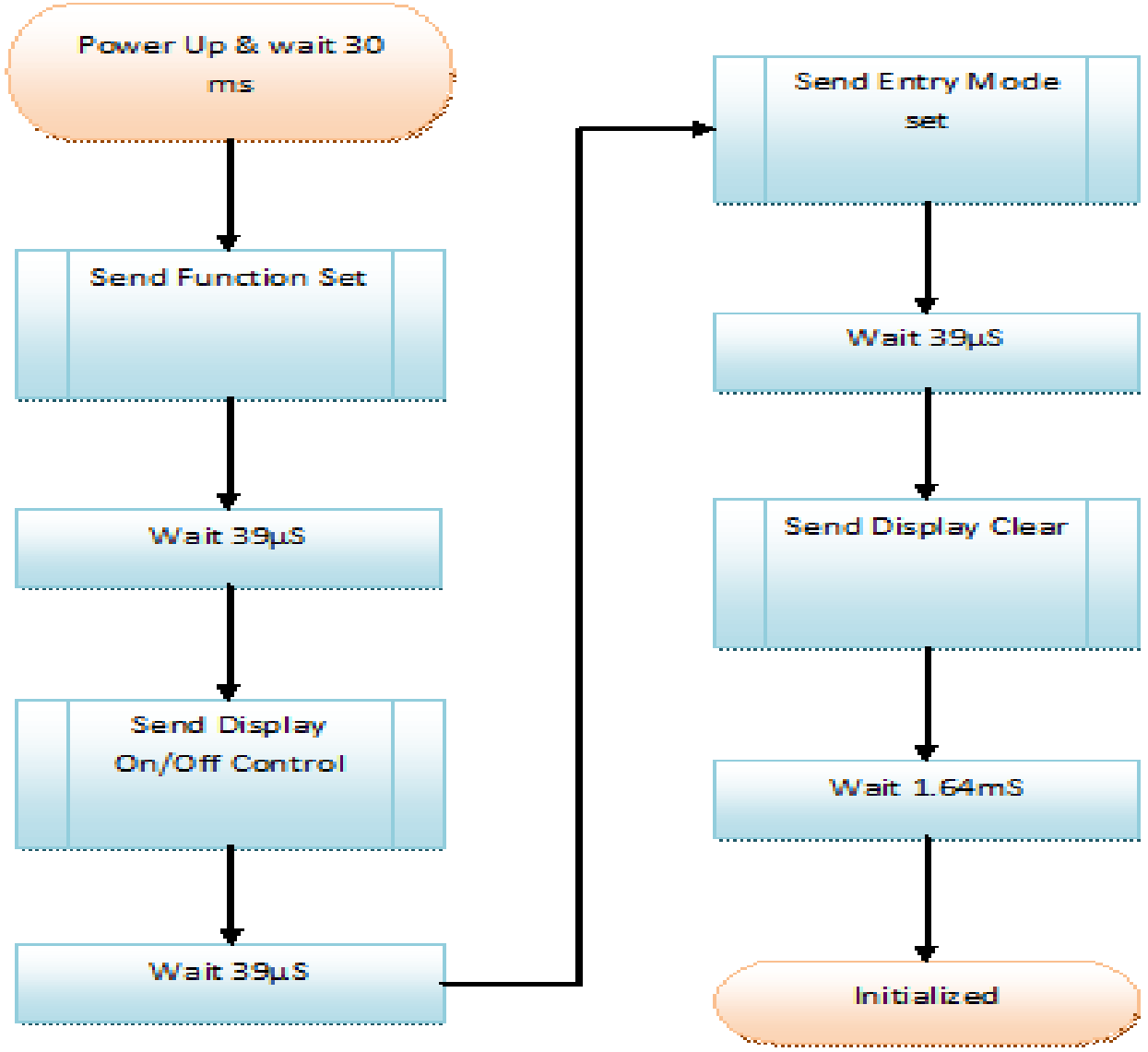
- Any attempt to send any data before this interval may lead to failure to read data or execution of the current data. So we have to give appropriate delay after each command/data writing.

LCD Initialization

Before displaying characters on the LCD display, it must be configured first. To configure an LCD display, four command words must be sent to LCD.

The commands are:

1. Function set
2. Display On/Off control
3. Entry mode set
4. Display and Clear



LCD Initialization Flow Chart

Before function set, we need to configure in 4 bit mode

PIN LOGIC VALUE WHEN COMMANDS ARE PASSED IN 4 BIT MODE

COMMAND EXAMPLE 0x28 AND 0x0C

deepseek

RS	R/W	EN	D0	D1	D2	D3	D4	D5	D6	D7
0	0	1	NC	NC	NC	NC	0	1	0	0
DELAY OF 10 uSEC										
0	0	0	NC	NC	NC	NC	0	1	0	0
0	0	1	NC	NC	NC	NC	0	0	0	1
DELAY OF 10 uSEC										
0	0	0	NC	NC	NC	NC	0	0	0	1
DELAY OF 10 mSEC before giving the next COMMAND										
Note :- COMMAND EXAMPLE 0x28 – 8 IS MASKED AND ONLY 2 IS SENT										
Note :- COMMAND EXAMPLE 0x28 MASK THE UPPER NIBBEL 8 IS SENT										
Note :- This whole proces sends the COMMAND 0x28 to LCD										

Control and display commands

Sl.No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 Lines and 5x7 Matrix

INTERFACING 4X4 Keyboard TO ARM MICROCONTROLLER

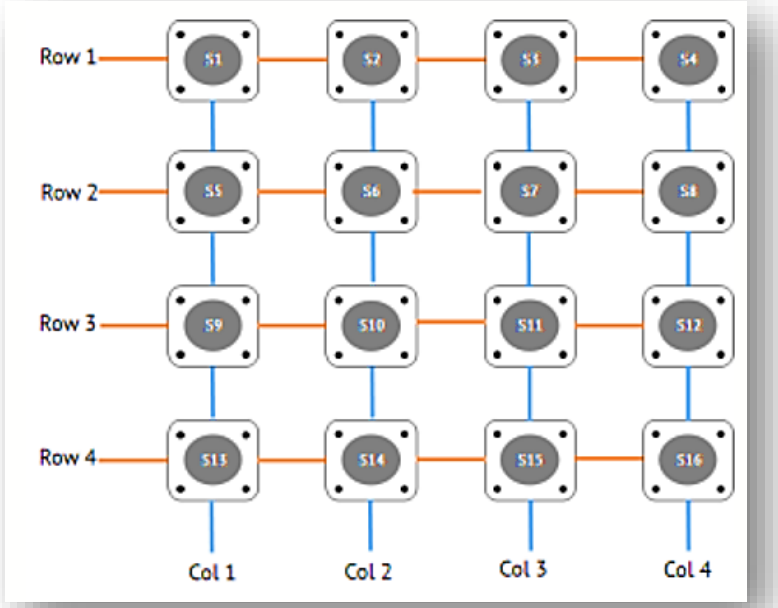
4×4 matrix keypad

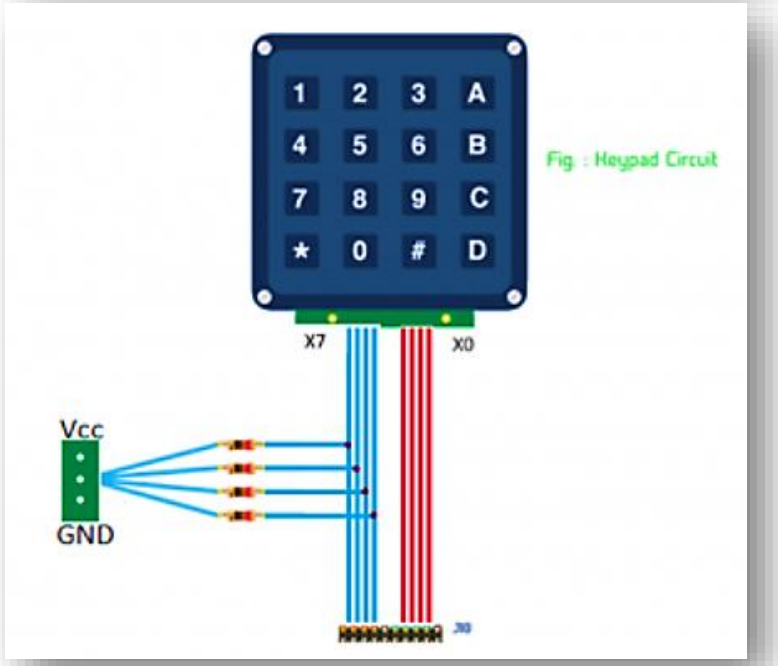
A total of 8 input-output pins of the micro-controller will be required for interfacing.

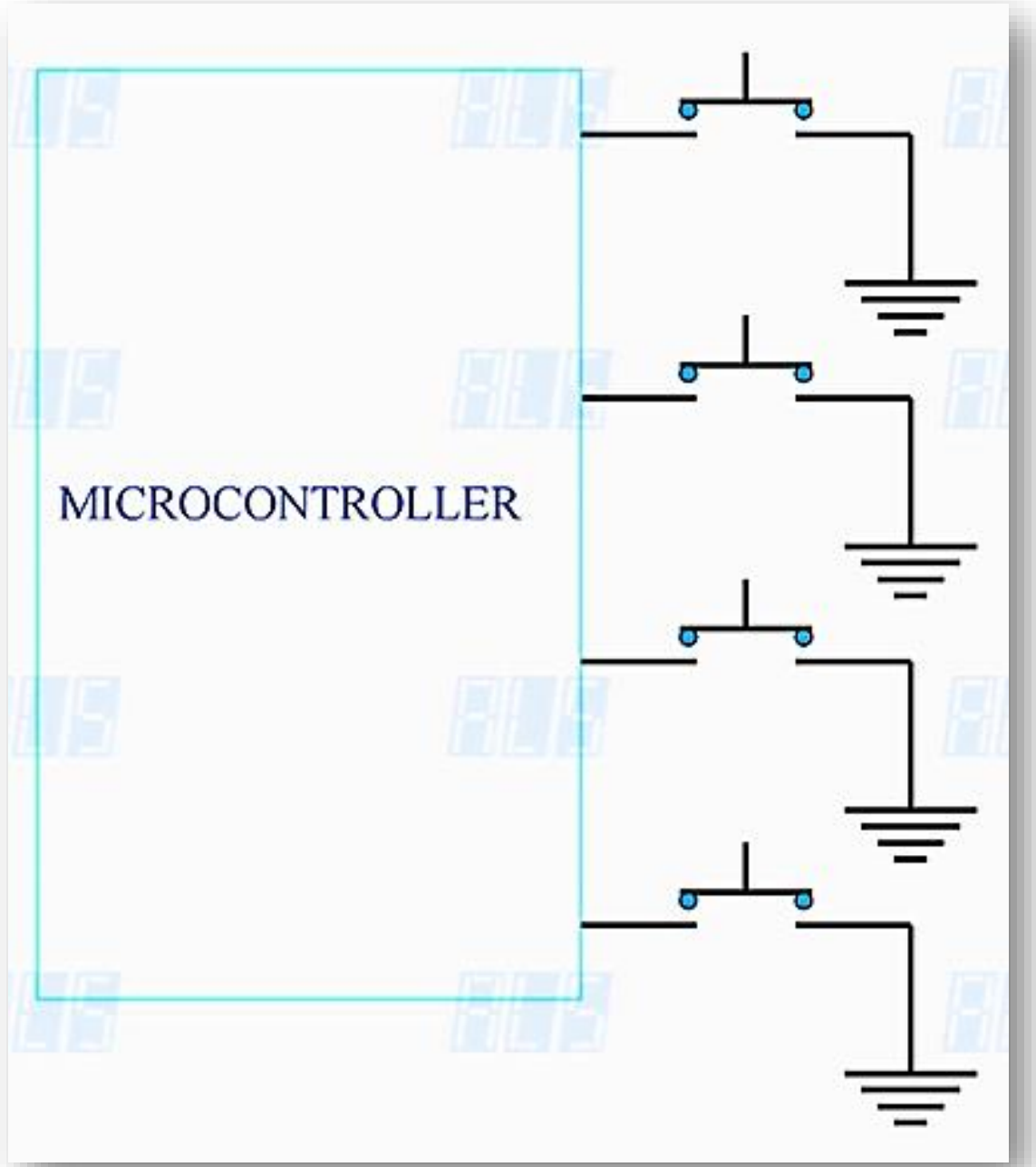
One-half of the 8 pins will be hardware controlled and the other half will be software controlled.

The blue lines are the columns, and the red lines are the rows.

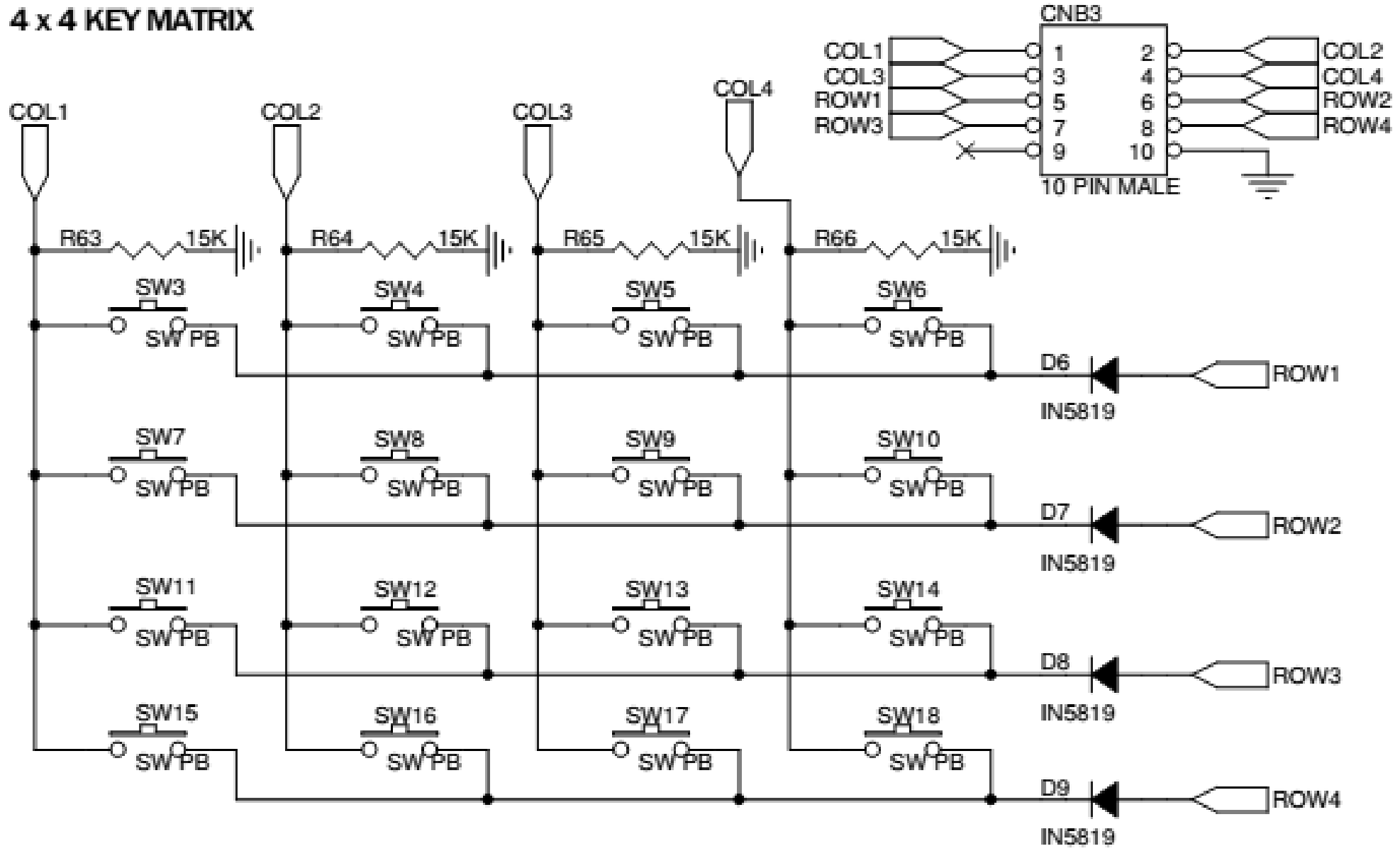
The rows are the four LSBs of the 8-bit pins and the columns are the MSBs of the 8-bit pins.






MICROCONTROLLER

4 x 4 KEY MATRIX



C1 to C4 -> P1.23 to P1.26. R1 to R4 -> P2.10 to P2.13

- The user provides input through the keyboard. The basic unit of exchange is the character. Push button switch is also used as keyboard if required keys are less. In this case, push button switch is directly interfaced to the port pins for reading.
- Keyboard controller has a counter which continuously increments at a certain rate and scans each key whether it is in pressed or released state.
- Keys are arranged in matrix format. For example, 12 keys are arranged in 3 column x 4 row matrix.
- A keypad consists of a set of buttons that may be pressed to provide input to an embedded system.

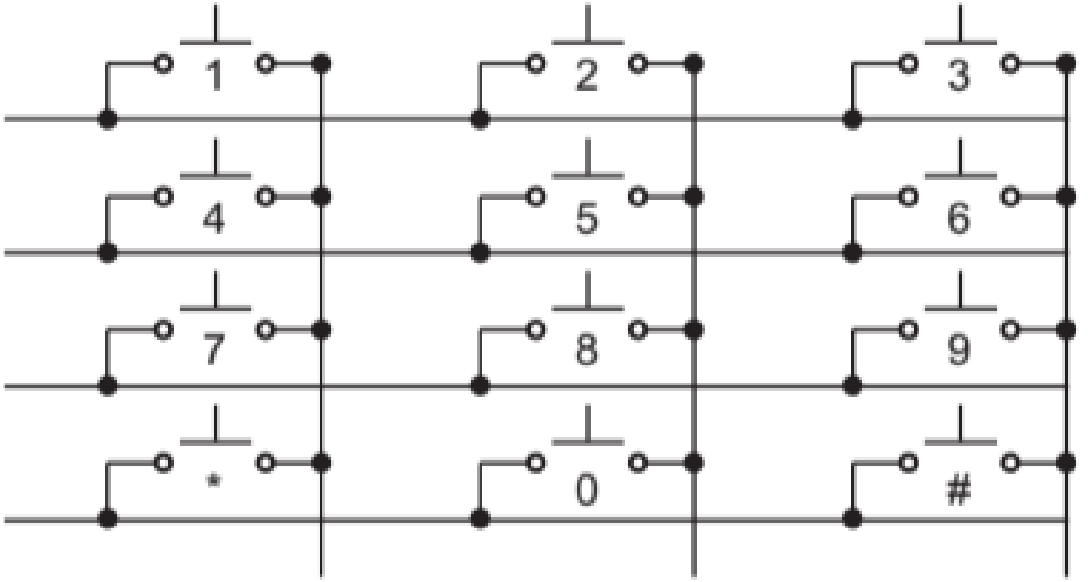
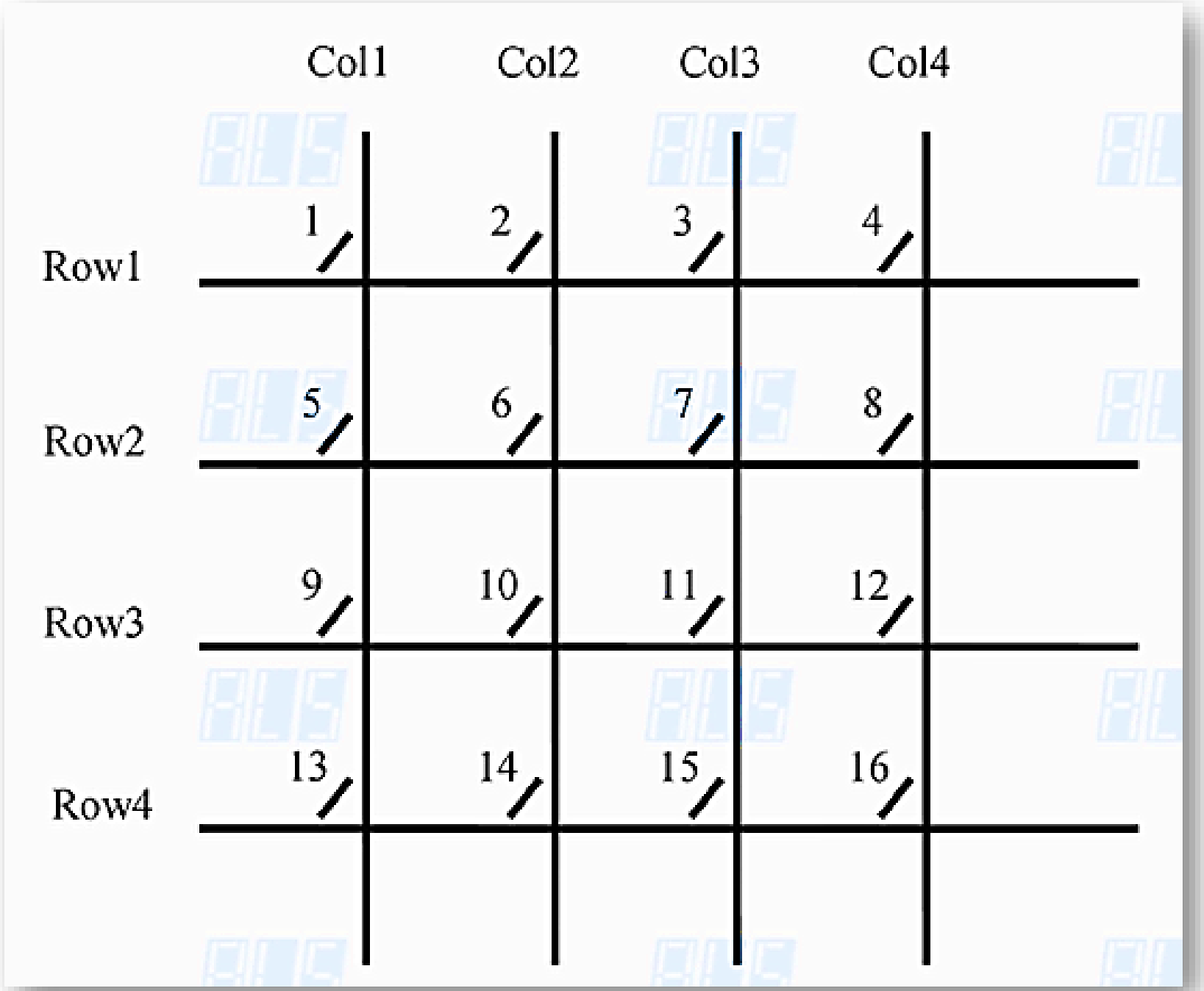


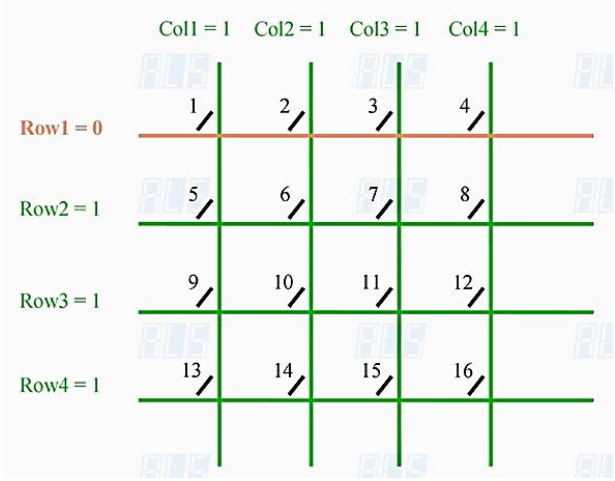
Fig. 5.11.7 Matrix keyboard interfacing

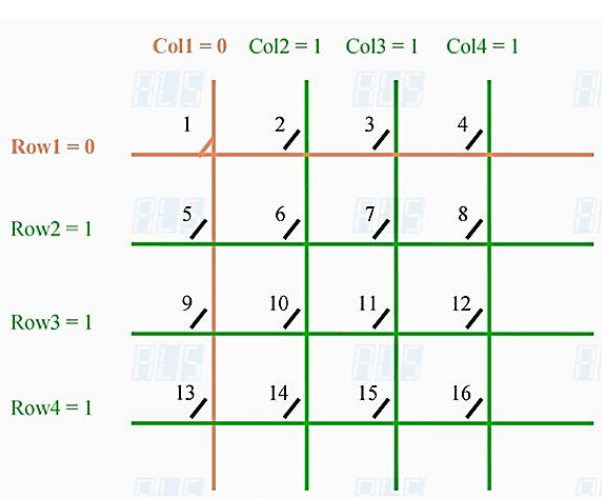
- A simple keypad has buttons arranged in an N-column by M-row grid. The device has N outputs, each output corresponding to a column, and another M outputs, each output corresponding to a row. When we press a

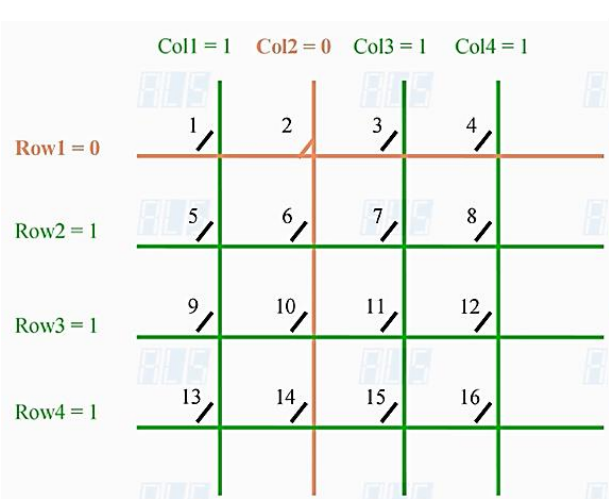
button, one column output and one row output go high, uniquely identifying the pressed button. To read such a keypad from software, we must scan the column and row outputs.

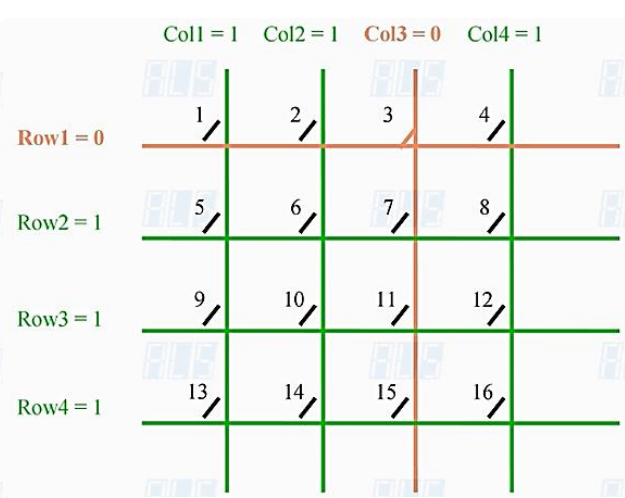
- The scanning may instead be performed by a keypad controller.
- Each key is a simple mechanical switch located at the crossing between the matrix rows and columns. When a key is pressed, its row and column form an electrical contact. The rows and columns can be connected to the pins of microcontroller ports.
- Keyboard controller has counter driven by a scan clock, which continuously increments at certain rate and scans each key whether that is in pressed or released state.
- Bounces create on pressing the key. Each bounce creates a false pulse. Keyboard controller has hardware debouncer to the care of bouncing of a key.

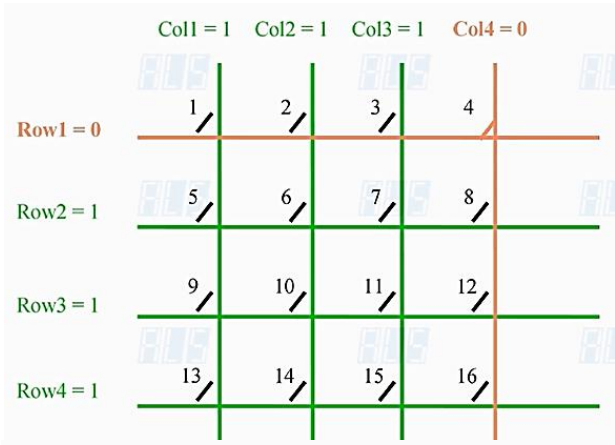

Col1
Col2
Col3
Col4
Row1
Row2
Row3
Row4
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

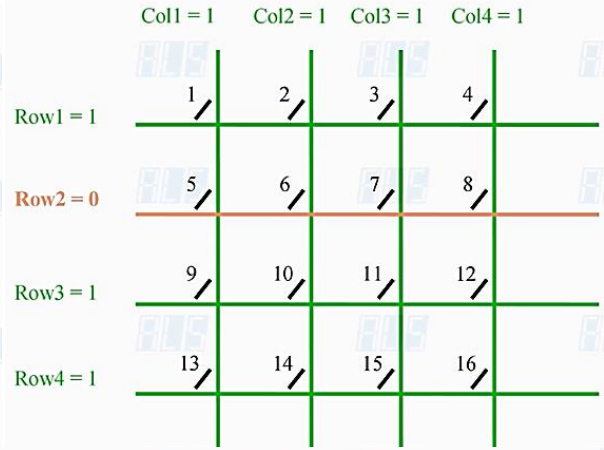

Col1 = 1
Col2 = 1
Col3 = 1
Col4 = 1
Row1 = 0
Row2 = 1
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

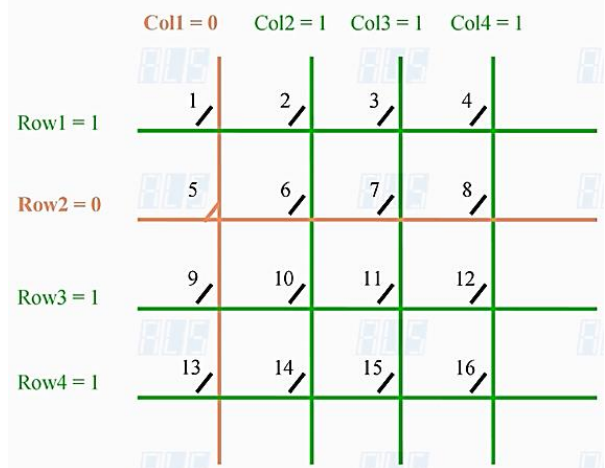

Col1 = 0
Col2 = 1
Col3 = 1
Col4 = 1
Row1 = 0
Row2 = 1
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

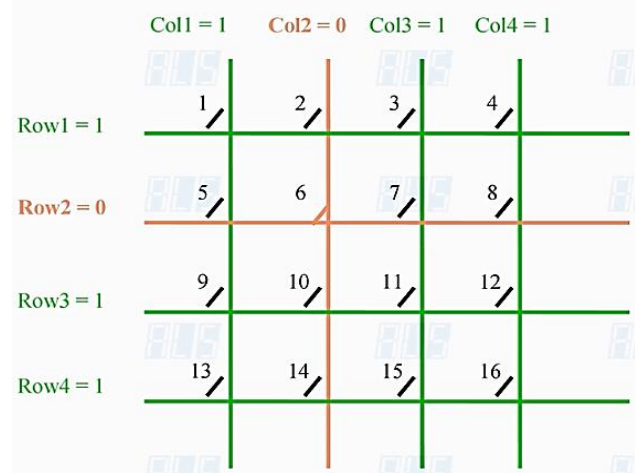

Col1 = 1
Col2 = 0
Col3 = 1
Col4 = 1
Row1 = 0
Row2 = 1
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

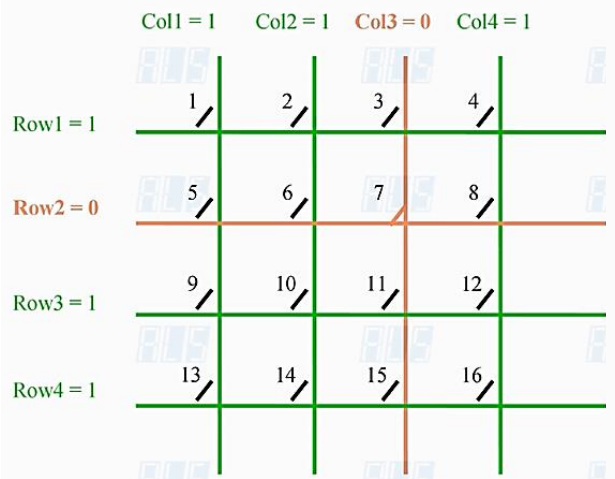

Col1 = 1
Col2 = 1
Col3 = 0
Col4 = 1
Row1 = 0
Row2 = 1
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

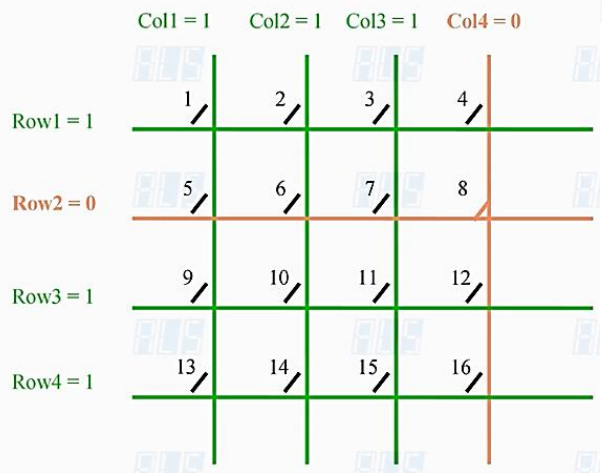

Col1 = 1
Col2 = 1
Col3 = 1
Col4 = 0
Row1 = 0
Row2 = 1
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

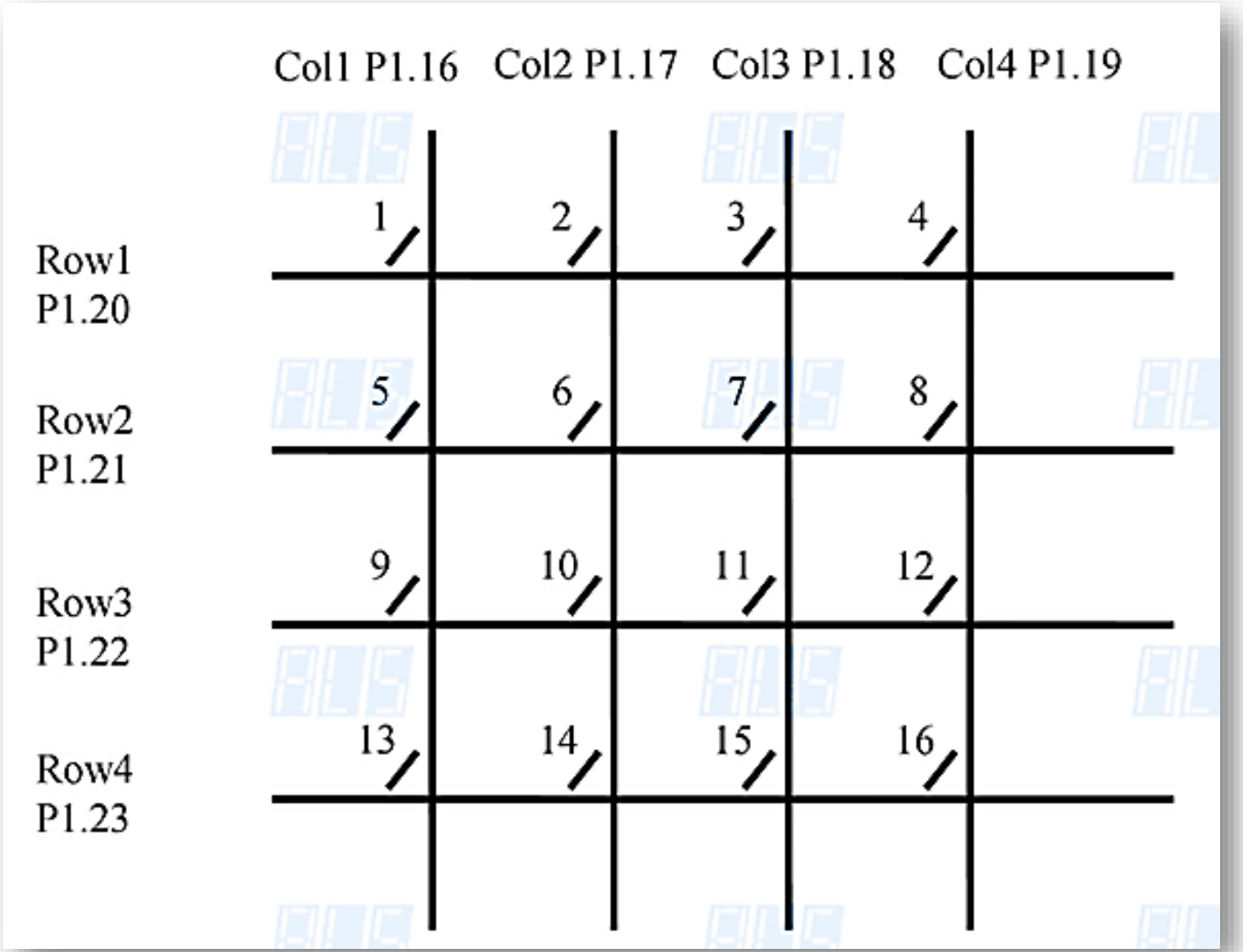

Col1 = 1
Col2 = 1
Col3 = 1
Col4 = 1
Row1 = 1
Row2 = 0
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

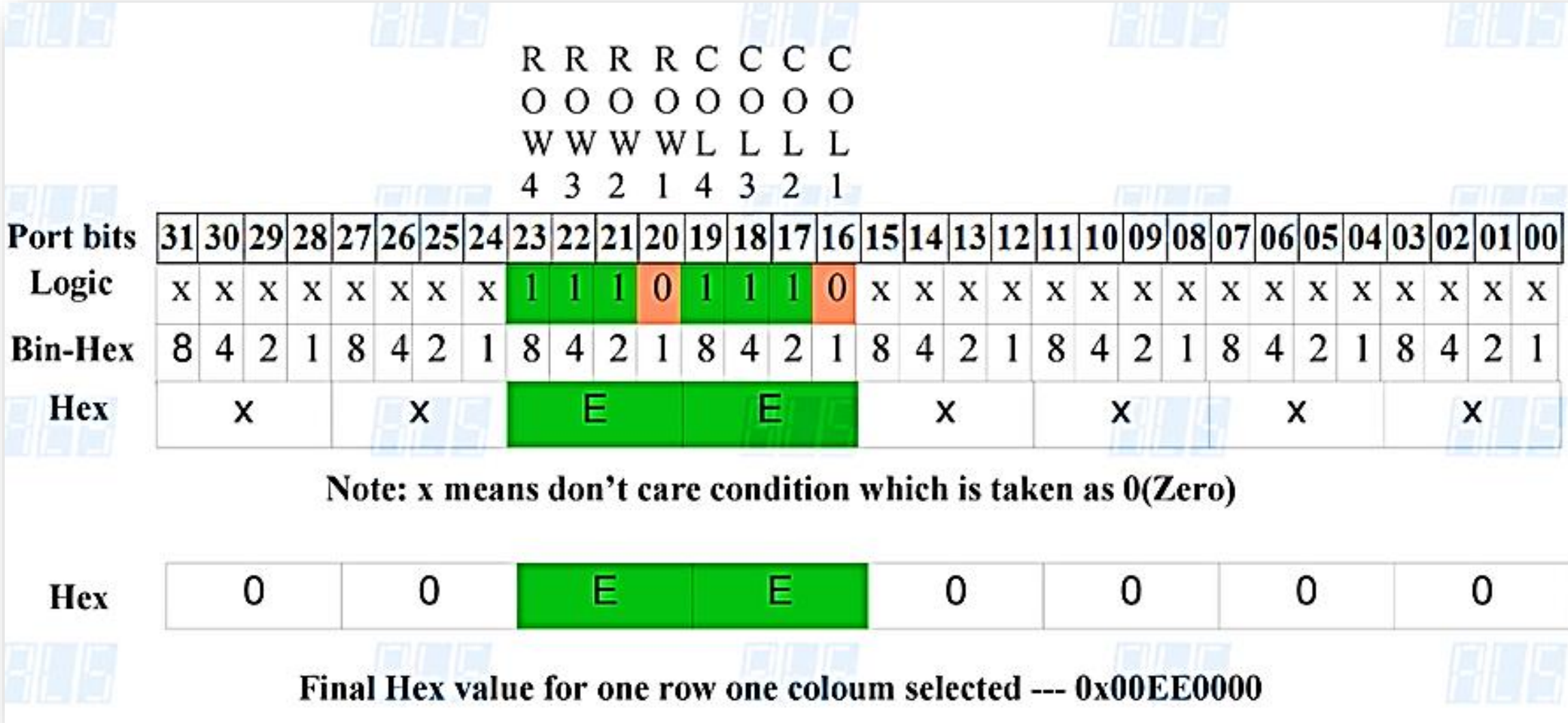

Col1 = 0
Col2 = 1
Col3 = 1
Col4 = 1
Row1 = 1
Row2 = 0
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


Col1 = 1
Col2 = 0
Col3 = 1
Col4 = 1
Row1 = 1
Row2 = 0
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


Col1 = 1
Col2 = 1
Col3 = 0
Col4 = 1
Row1 = 1
Row2 = 0
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


Col1 = 1
Col2 = 1
Col3 = 1
Col4 = 0
Row1 = 1
Row2 = 0
Row3 = 1
Row4 = 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


Col1 P1.16
Col2 P1.17
Col3 P1.18
Col4 P1.19
Row1
P1.20
Row2
P1.21
Row3
P1.22
Row4
P1.23
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


R R R R C C C C
O O O O O O O O
W W W W L L L L
4 3 2 1 4 3 2 1
Port bits 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Logic x x x x x x x x 1 1 1 0 1 1 1 0 x x x x x x x x x x x x x x x x
Bin-Hex 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1
Hex x x E E x x x x
Note: x means don't care condition which is taken as 0(Zero)
Hex 0 0 E E 0 0 0 0
Final Hex value for one row one coloum selected --- 0x00EE0000

								R O W 4	R O W 3	R O W 2	R O W 1	C O L 4	C O L 3	C O L 2	C O L 1														
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
0				0				E				E				0				0				0				0			
0				0				E				D				0				0				0				0			
0				0				E				B				0				0				0				0			
0				0				E				7				0				0				0				0			

								ROW4	ROW3	ROW2	ROW1	COL4	COL3	COL2	COL1														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
0				0				D				E				0				0				0				0			
0				0				D				D				0				0				0				0			
0				0				D				B				0				0				0				0			
0				0				D				7				0				0				0				0			

								R O W 4	R O W 3	R O W 2	R O W 1	C O L 4	C O L 3	C O L 2	C O L 1														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

0		0		7		E	
0		0		7		D	
0		0		7		B	
0		0		7		7	