

---

# **DIGITAL DESIGN**

## **Through Embedded Programming**

---

**G. V. V. Sharma**



Copyright ©2022 by G. V. V. Sharma.

<https://creativecommons.org/licenses/by-sa/3.0/>

and

<https://www.gnu.org/licenses/fdl-1.3.en.html>

# Contents

|   |           |
|---|-----------|
| Introduction  | iii       |
| <b>1 Installation</b>                                 | <b>1</b>  |
| <b>1.1 Termux</b> . . . . .                           | 1         |
| <b>1.2 Platformio</b> . . . . .                       | 2         |
| <b>1.3 Arduino Droid</b> . . . . .                    | 3         |
| <b>2 Seven Segment Display</b>                        | <b>5</b>  |
| <b>2.1 Components</b> . . . . .                       | 5         |
| <b>2.1.1 Breadboard</b> . . . . .                     | 5         |
| <b>2.1.2 Seven Segment Display</b> . . . . .          | 6         |
| <b>2.1.3 Arduino</b> . . . . .                        | 6         |
| <b>2.2 Display Control through Hardware</b> . . . . . | 6         |
| <b>2.2.1 Powering the Display</b> . . . . .           | 6         |
| <b>2.2.2 Controlling the Display</b> . . . . .        | 7         |
| <b>2.3 Display Control through Software</b> . . . . . | 8         |
| <b>3 7447</b>   | <b>11</b> |
| <b>3.1 Components</b> . . . . .                       | 11        |
| <b>3.2 Hardware</b> . . . . .                         | 11        |

|                                  |           |
|----------------------------------|-----------|
| <b>3.3 Software</b>              | 12        |
| <b>3.4 Problems</b>              | 14        |
| <b>4 Karnaugh Map</b>            | <b>25</b> |
| <b>4.1 Introduction</b>          | 25        |
| <b>4.2 Incrementing Decoder</b>  | 25        |
| <b>4.3 Karnaugh Map</b>          | 25        |
| <b>4.4 Dont Care</b>             | 31        |
| <b>4.5 Don't Care Conditions</b> | 32        |
| <b>4.6 Problems</b>              | 37        |
| <b>5 7474</b>                    | <b>47</b> |
| <b>5.1 Components</b>            | 47        |
| <b>5.2 Decade Counter</b>        | 47        |
| <b>6 Finite State Machine</b>    | <b>51</b> |
| <b>6.1 The Decade Counter</b>    | 51        |
| <b>6.2 Finite State Machine</b>  | 51        |
| <b>6.3 Problems</b>              | 52        |
| <b>7 Assembly Programming</b>    | <b>57</b> |
| <b>7.1 Software Installation</b> | 57        |
| <b>7.2 Seven Segment Display</b> | 59        |
| <b>7.3 7447</b>                  | 61        |
| <b>7.3.1 Components</b>          | 61        |

|   |    |
|---|----|
| <b>7.3.2 Boolean Operations</b>         | 62 |
| <b>7.3.3 Controlling the Display</b>    | 63 |
| <b>7.4 Timer</b>                        | 65 |
| <b>7.4.1 Components</b>                 | 65 |
| <b>7.4.2 Blink through TIMER</b>        | 65 |
| <b>7.4.3 Blink through Cycle Delays</b> | 67 |
| <b>7.5 Memory</b>                       | 68 |
| <br>                                    |    |
| <b>8 Embedded C</b>                     | 71 |
| <b>8.1 Blink</b>                        | 71 |
| <b>8.1.1 Components</b>                 | 71 |
| <b>8.1.2 Blink</b>                      | 71 |
| <b>8.2 Display Control</b>              | 72 |
| <b>8.3 Input</b>                        | 72 |
| <b>8.4 GCC-Assembly</b>                 | 73 |
| <b>8.4.1 Components</b>                 | 73 |
| <b>8.4.2 GCC with Assembly</b>          | 73 |
| <b>8.5 LCD</b>                          | 74 |
| <b>8.5.1 Components</b>                 | 75 |
| <b>8.5.2 Display Number on LCD</b>      | 75 |
| <br>                                    |    |
| <b>9 Vaman-FPGA</b>                     | 77 |
| <b>9.1 Setup</b>                        | 77 |
| <b>9.1.1 Software</b>                   | 77 |

|                                   |     |
|-----------------------------------|-----|
| <b>9.1.2 Setup</b>                | 77  |
| <b>9.1.3 Frequency</b>            | 78  |
| <b>9.2 Seven Segment Display</b>  | 81  |
| <b>9.2.1 Software</b>             | 81  |
| <b>9.2.2 Setup</b>                | 81  |
| <b>9.2.3 Examples</b>             | 83  |
| <b>9.3 Boolean Logic</b>          | 83  |
| <b>9.3.1 Software</b>             | 84  |
| <b>9.3.2 Setup</b>                | 84  |
| <b>9.3.3 Decade Counter</b>       | 87  |
| <b>10 Vaman-ARM</b>               | 95  |
| <b>10.1 Setup</b>                 | 95  |
| <b>10.1.1 Software</b>            | 95  |
| <b>10.1.2 Setup</b>               | 95  |
| <b>10.1.3 Delay</b>               | 95  |
| <b>10.2 Seven Segment Display</b> | 97  |
| <b>10.2.1 Software</b>            | 97  |
| <b>10.2.2 Setup</b>               | 97  |
| <b>10.2.3 Examples</b>            | 100 |
| <b>10.3 FSM</b>                   | 101 |
| <b>10.3.1 Software</b>            | 101 |
| <b>10.3.2 Setup</b>               | 101 |

|   |     |
|---|-----|
| <b>10.3.3 Decade Counter</b>                | 104 |
| <b>11 Vaman-ESP32</b>                       | 113 |
| <b>11.1 Software</b>                        | 113 |
| <b>11.2 Flash Vaman-ESP32 using Arduino</b> | 113 |
| <b>11.3 OTA</b>                             | 115 |
| <b>11.4 Onboard LED</b>                     | 116 |



# **Introduction**

This book introduces digital design through using the arduino framework.



# Chapter 1

## Installation

### 1.1. Termux

1. On your android device, install fdroid apk from

```
https://www.f-droid.org/
```

2. Install Termux from apkpure

3. Install basic packages on termux

```
#Give termux access to your user directory in android  
termux-setup-storage  
  
#Upgrade packages  
apt update && apt upgrade  
apt install build-essential openssh  
  
#Mandatory packages  
apt install curl git wget subversion proot proot-distro python nmap neovim ranger  
#-----End Install Termux
```

4. Install Ubuntu on termux

```
proot-distro install ubuntu  
proot-distro login ubuntu
```

## 1.2. Platformio

1. Install Packages

```
apt update && apt upgrade  
apt install apt-utils build-essential cmake neovim  
apt install git wget subversion imagemagick nano  
apt install avra avrdude gcc-avr avr-libc  
#-----End Installing ubuntu on termux  
  
#----- Installing python3 on termuxubuntu  
  
apt install python3-pip python3-numpy python3-scipy python3-matplotlib  
    python3-mpmath python3-sympy python3-cvxopt  
#----- End installing python3 on termuxubuntu  
  
#----- Installing platformio on termuxubuntu
```

```
-----  
pip3 install platformio  
#----- End installing python3 on termuxubuntu  
-----
```

2. Execute the following on ubuntu

```
cd ide/piosetup/codes  
pio run
```

3. Connect your arduino to the laptop/rpi and type

```
pio run -t nobuild -t upload
```

4. The LED beside pin 13 will start blinking

### 1.3. Arduino Droid

1. Install ArduinoDroid from apkpure
2. Open ArduinoDroid and grant all permissions
3. Connect the Arduino to your phone via USB-OTG
4. For flashing the bin files, in ArduinoDroid,

```
Actions->Upload->Upload Precompiled
```

then go to your working directory and select

```
pio/build/uno/firmwarehex
```

for uploading hex file to the Arduino Uno

5. The LED beside pin 13 will start blinking

# Chapter 2

## Seven Segment Display

We show how to control a seven segment display.

### 2.1. Components

| Component                | Value            | Quantity |
|--------------------------|------------------|----------|
| Breadboard               |                  | 1        |
| Resistor                 | $\geq 220\Omega$ | 1        |
| Arduino                  | Uno              | 1        |
| Seven Segment<br>Display | Common<br>Anode  | 1        |
| Jumper Wires             |                  | 20       |

Table 2.1:

#### 2.1.1. Breadboard

The breadboard can be divided into 5 segments. In each of the green segments, the pins are internally connected so as to have the same voltage. Similarly, in the central segments, the pins in each column are internally connected in the same fashion as the blue columns.

## 2.1.2. Seven Segment Display

The seven segment display in Fig. 2.2 has eight pins, *a*, *b*, *c*, *d*, *e*, *f*, *g* and *dot* that take an active LOW input, i.e. the LED will glow only if the input is connected to ground. Each of these pins is connected to an LED segment. The *dot* pin is reserved for the · LED.

## 2.1.3. Arduino

The Arduino Uno has some ground pins, analog input pins A0-A3 and digital pins D1-D13 that can be used for both input as well as output. It also has two power pins that can generate 3.3V and 5V. In the following exercises, only the GND, 5V and digital pins will be used.

# 2.2. Display Control through Hardware

## 2.2.1. Powering the Display

1. Plug the display to the breadboard in Fig. 2.1 and make the connections in Table 2.2.

Henceforth, all 5V and GND connections will be made from the breadboard.

| Arduino | Breadboard   |
|---------|--------------|
| 5V      | Top Green    |
| GND     | Bottom Green |

Table 2.2:

2. Make the connections in Table 2.3.
3. Connect the Arduino to the computer. The DOT led should glow.

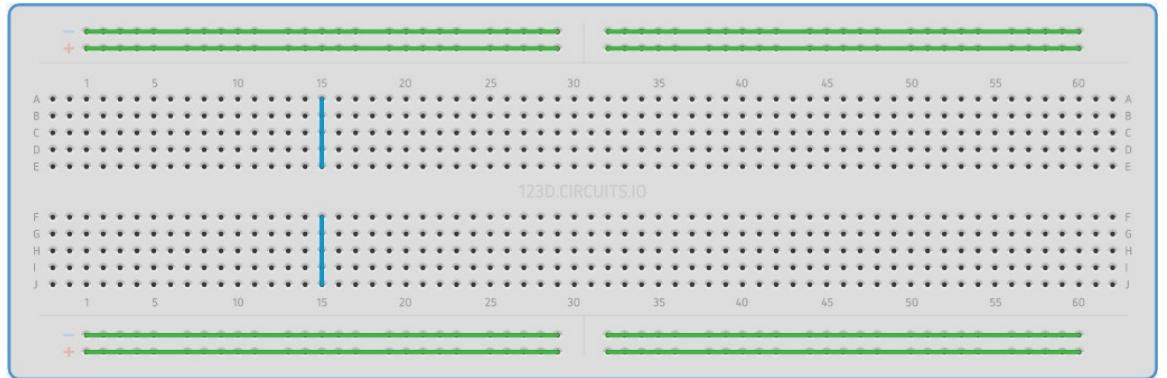


Figure 2.1:

| Breadboard |          | Display |
|------------|----------|---------|
| 5V         | Resistor | COM     |
| GND        |          | DOT     |

Table 2.3:

## 2.2.2. Controlling the Display

Fig. 2.3 explains how to get decimal digits using the seven segment display. GND=0.

1. Generate the number 1 on the display by connecting only the pins *b* and *c* to GND (=0). This corresponds to the first row of 2.4. 1 means not connecting to GND.
2. Repeat the above exercise to generate the number 2 on the display.
3. Draw the numbers 0-9 as in Fig. 2.3 and complete Table 2.4

| a | b | c | d | e | f | g | decimal |
|---|---|---|---|---|---|---|---------|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0       |

Table 2.4:

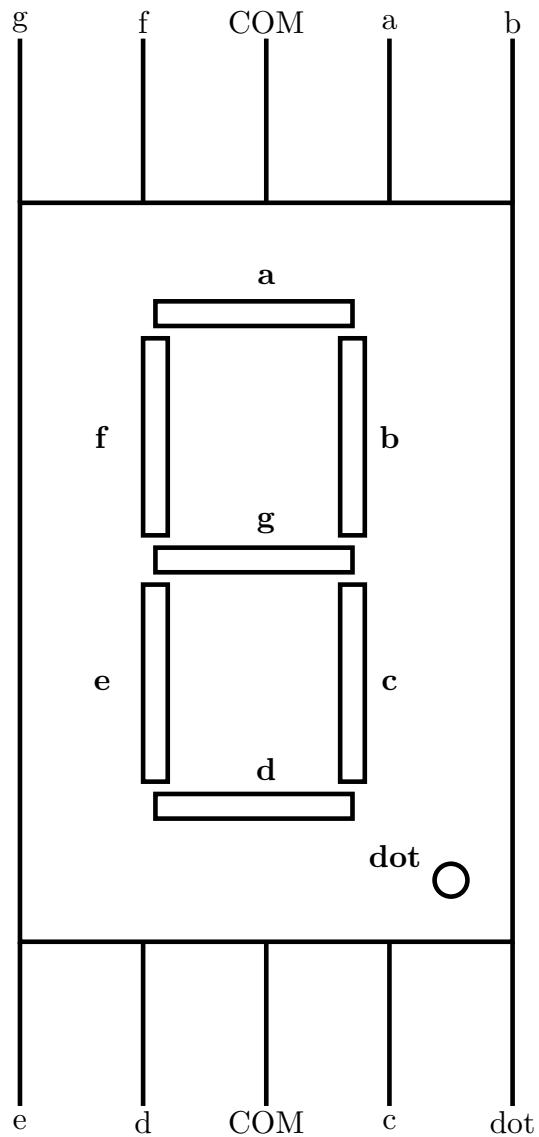


Figure 2.2:

## 2.3. Display Control through Software

1. Make connections according to Table 2.5

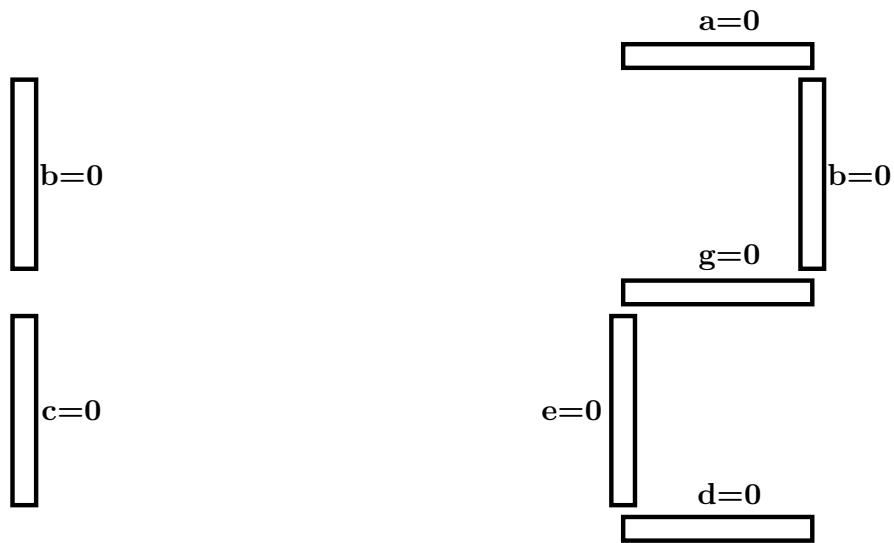


Figure 2.3:

| Arduino | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------|---|---|---|---|---|---|---|
| Display | a | b | c | d | e | f | g |

Table 2.5:

2. Download the following code using the arduino IDE and execute

```
ide/sevenseg/codes/sevenseg/sevenseg.ino
```

3. Now generate the numbers 0-9 by modifying the above program.



# Chapter 3

7447

Here we show how to use the 7447 BCD-Seven Segment Display decoder to learn Boolean logic.

## 3.1. Components

| Component             | Value   | Quantity |
|-----------------------|---------|----------|
| Resistor              | 220 Ohm | 1        |
| Arduino               | UNO     | 1        |
| Seven Segment Display |         | 1        |
| Decoder               | 7447    | 1        |
| Jumper Wires          | M-M     | 20       |
| Breadboard            |         | 1        |

Table 3.1:

## 3.2. Hardware

1. Make connections between the seven segment display in Fig. 2.2 and the 7447 IC in Fig. 3.1 as shown in Table 3.2

|                |           |           |           |           |           |           |           |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>7447</b>    | $\bar{a}$ | $\bar{b}$ | $\bar{c}$ | $\bar{d}$ | $\bar{e}$ | $\bar{f}$ | $\bar{g}$ |
| <b>Display</b> | a         | b         | c         | d         | e         | f         | g         |

Table 3.2:

2. Make connections to the lower pins of the 7447 according to Table 3.3 and connect  $V_{CC} = 5V$ . You should see the number 0 displayed for 0000 and 1 for 0001.

| D | C | B | A | Decimal |
|---|---|---|---|---------|
| 0 | 0 | 0 | 0 | 0       |
| 0 | 0 | 0 | 1 | 1       |

Table 3.3:

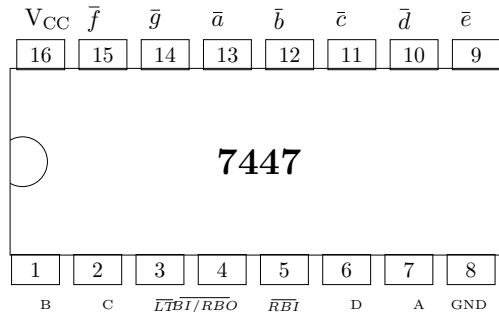


Figure 3.1:

3. Complete Table 3.3 by generating all numbers between 0-9.

### 3.3. Software

1. Now make the connections as per Table 3.4 and execute the following program

```
ide/7447/codes/gvv_ard_7447/gvv_ard_7447.cpp
```

|                |   |   |   |   |
|----------------|---|---|---|---|
| <b>7447</b>    | D | C | B | A |
| <b>Arduino</b> | 5 | 4 | 3 | 2 |

Table 3.4:

In the truth table in Table 3.5,  $W, X, Y, Z$  are the inputs and  $A, B, C, D$  are the outputs. This table represents the system that increments the numbers 0-8 by 1 and resets the number 9 to 0. Note that  $D = 1$  for the inputs 0111 and 1000. Using boolean logic,

$$D = WXYZ' + W'X'Y'Z \quad (3.1)$$

Note that 0111 results in the expression  $WXYZ'$  and 1000 yields  $W'X'Y'Z$ .

2. The code below realizes the Boolean logic for B, C and D in Table 3.5. Write the logic for A and verify.

ide/7447/codes/inc\_dec/inc\_dec.ino

| Z | Y | X | W | D | C | B | A |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 3.5:

3. Now make additional connections as shown in Table 3.6 and execute the following code. Comment.

```
ide/7447/codes/ip_inc_dec/ip_inc_dec.cpp
```

**Solution:** In this exercise, we are taking the number 5 as input to the arduino and displaying it on the seven segment display using the 7447 IC.

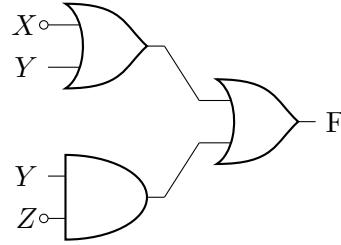
|         | Z | Y | X | W |
|---------|---|---|---|---|
| Input   | 0 | 1 | 0 | 1 |
| Arduino | 9 | 8 | 7 | 6 |

Table 3.6:

4. Verify the above code for all inputs from 0-9.
5. Now write a program where
  - (a) the binary inputs are given by connecting to 0 and 1 on the breadboard
  - (b) incremented by 1 using Table 3.5 and
  - (c) the incremented value is displayed on the seven segment display.
6. Write the truth table for the 7447 IC and obtain the corresponding boolean logic equations.
7. Implement the 7447 logic in the arudino. Verify that your arduino now behaves like the 7447 IC.

## 3.4. Problems

1. Obtain the Boolean Expression for the Logic circuit shown below (CBSE 2013)



2. Verify the Boolean Expression

(CBSE 2013)

$$A + C = A + A'C + BC \quad (3.2)$$

3. Draw the Logic Circuit for the following Boolean Expression

(CBSE 2015)

$$f(x, y, z, w) = (x' + y)z + w' \quad (3.3)$$

4. Verify the following

(CBSE 2015)

$$U' + V = U'V' + U'V + UV \quad (3.4)$$

5. Draw the Logic Circuit for the given Boolean Expression

(CBSE 2015)

$$(U + V')W' + Z \quad (3.5)$$

6. Verify the following using Boolean Laws

(CBSE 2015)

$$X + Y' = XY + XY' + X'Y' \quad (3.6)$$

7. Write the Boolean Expression for the result of the Logic Circuit as shown in Fig. 3.2

(CBSE 2016)

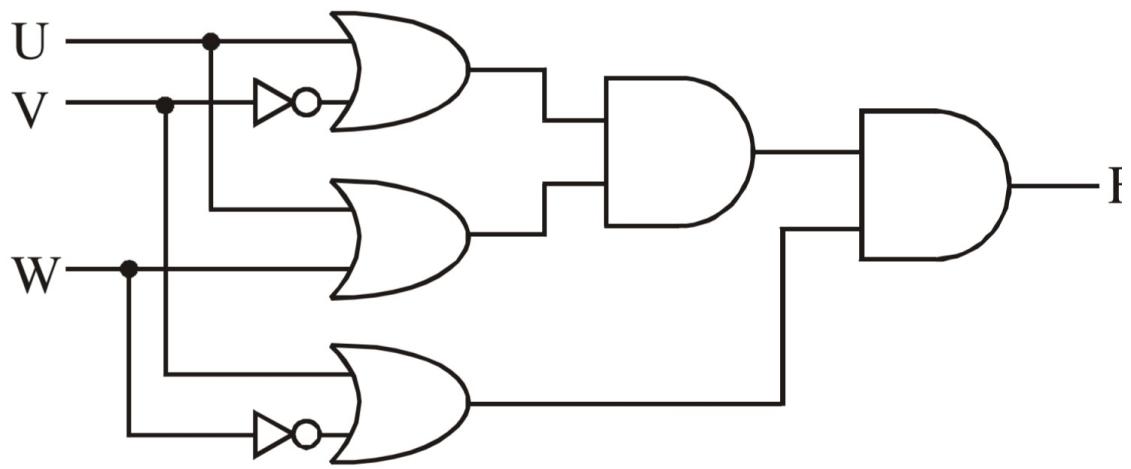


Figure 3.2:

8. Draw the logic circuit of the following Boolean Expression using only NAND Gates.

(CBSE 2017)

$$XY + YZ \quad (3.7)$$

9. Draw the Logic Circuit of the following Boolean Expression using only NOR Gates

(CBSE 2017)

$$(A + B)(C + D) \quad (3.8)$$

10. Draw the Logic Circuit of the following Boolean Expression

(CBSE 2018)

$$(U' + V)(V' + W') \quad (3.9)$$

11. Derive a Canonical POS expression for a Boolean function F, represented by Table 3.7 (CBSE 2019)

| X | Y | Z | F(X,Y,Z) |
|---|---|---|----------|
| 0 | 0 | 0 | 1        |
| 0 | 0 | 1 | 0        |
| 0 | 1 | 0 | 1        |
| 0 | 1 | 1 | 0        |
| 1 | 0 | 0 | 1        |
| 1 | 0 | 1 | 1        |
| 1 | 1 | 0 | 0        |
| 1 | 1 | 1 | 0        |

Table 3.7:

12. For the logic circuit shown in Fig.3.3, find the simplified Boolean expression for the output. (GATE EC 2000)

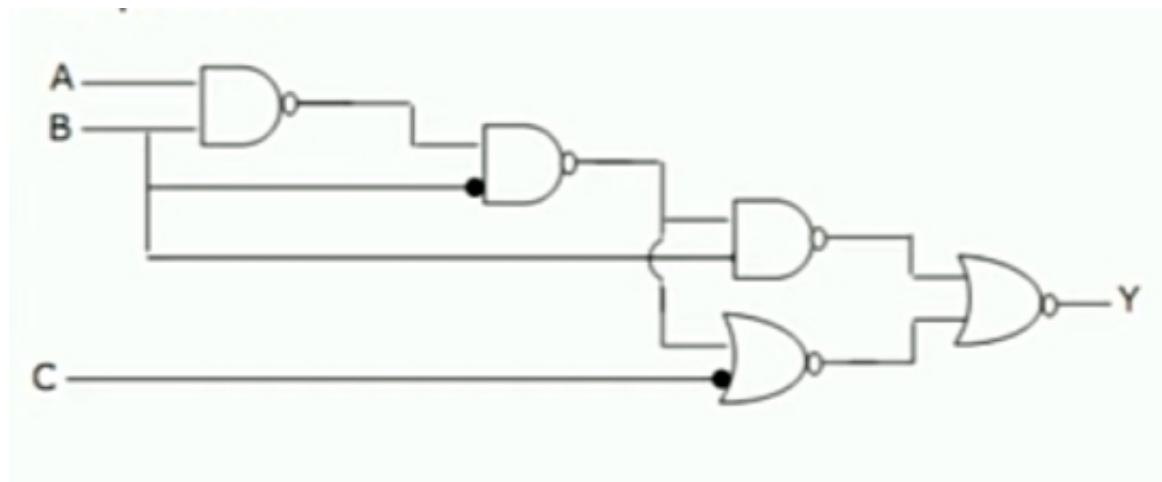
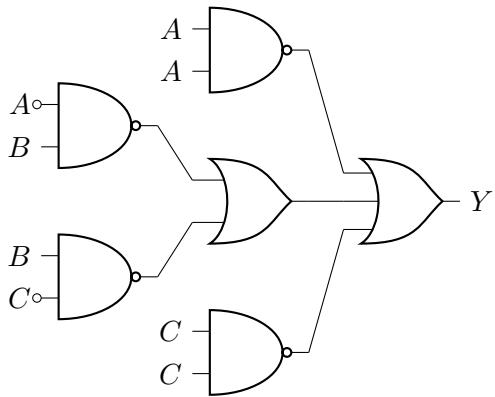


Figure 3.3:

13. Obtain the Boolean Expression for the Logic circuit shown below (GATE EC 1993)



14. Implement Table 3.8 using XNOR logic. (GATE EC 1993)

| <b>A</b> | <b>B</b> | <b>Y</b> |
|----------|----------|----------|
| 0        | 0        | 1        |
| 0        | 1        | 0        |
| 1        | 0        | 0        |
| 1        | 1        | 1        |

Table 3.8:

15. For a binary half-sub-tractor having two inputs A and B, find the correct set of logical expressions for the outputs D ( $=A$  minus  $B$ ) and X (=borrow). (GATE EC 1999)

16. Find  $X$  in the following circuit in Fig. 3.4 (GATE EC 2007)

17. A logic circuit implements the boolean function  $F=X' \cdot Y + X \cdot Y' \cdot Z'$ . It is found that the input combination  $X=Y=1$  can never occur. Taking this into account, find a simplified expression for  $F$ . (GATE IN 2007)

18. Find the Boolean logic realised by the following circuit in Fig. 3.5 (GATE EC 2010)

19. Find the logic function implemented by the circuit given below in Fig. 3.6 (GATE EC 2011)

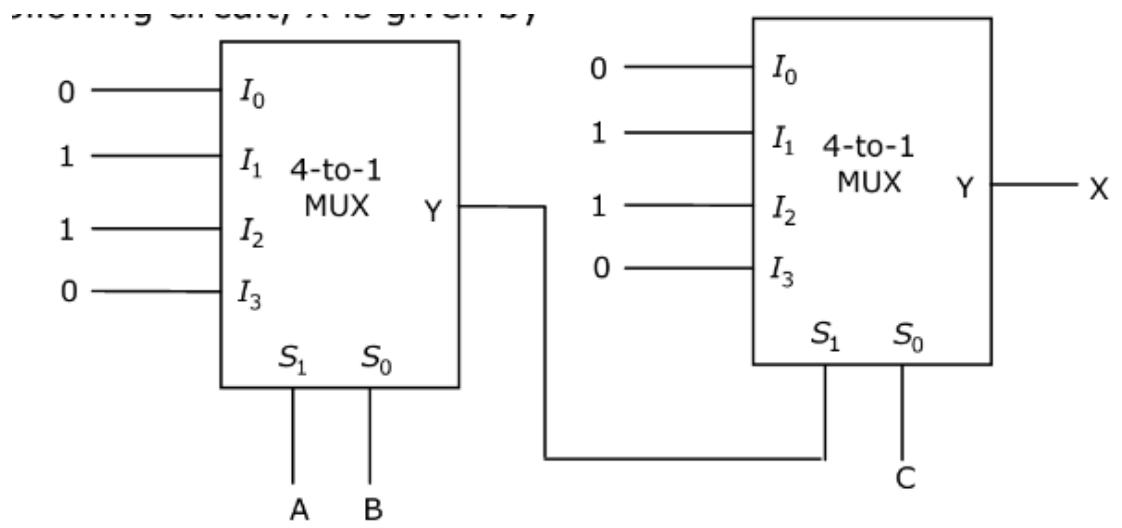


Figure 3.4:

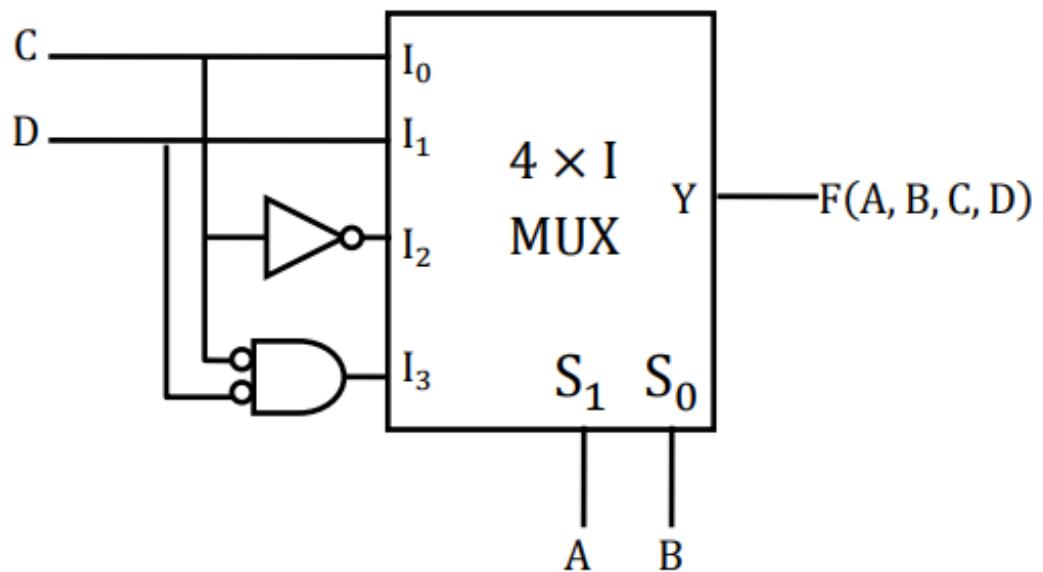


Figure 3.5:

20. Find  $F$  in the Digital Circuit given in the figure below in Fig. 3.7. (GATE IN 2016)

21. Find the logic function implemented by the circuit given below in Fig. 3.8 (GATE

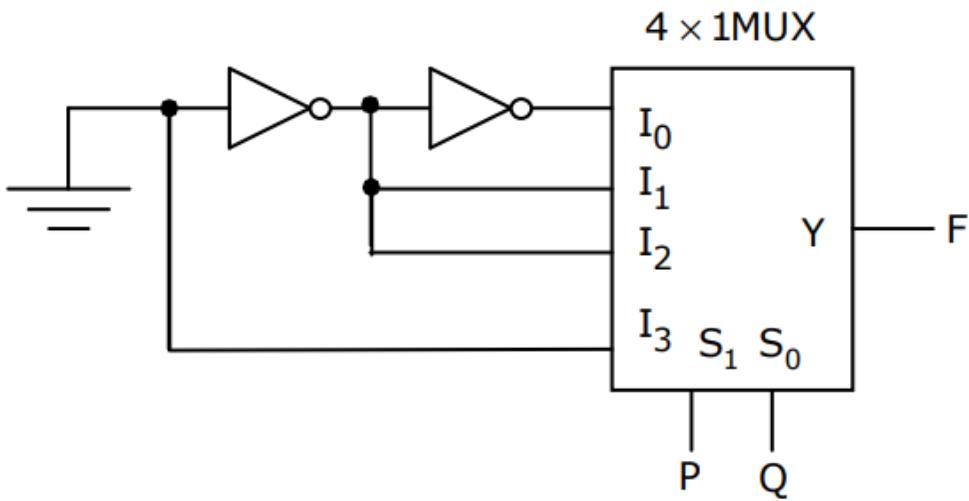


Figure 3.6:

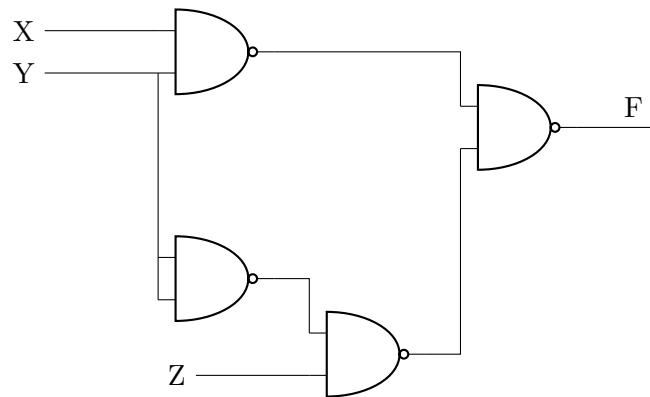


Figure 3.7:

EC 2017)

22. Find the logic function implemented by the circuit given below in Fig. 3.9 (GATE EC 2018)
23. Find the logic function implemented by the circuit given below in Fig. 3.10 (GATE EE 2018)

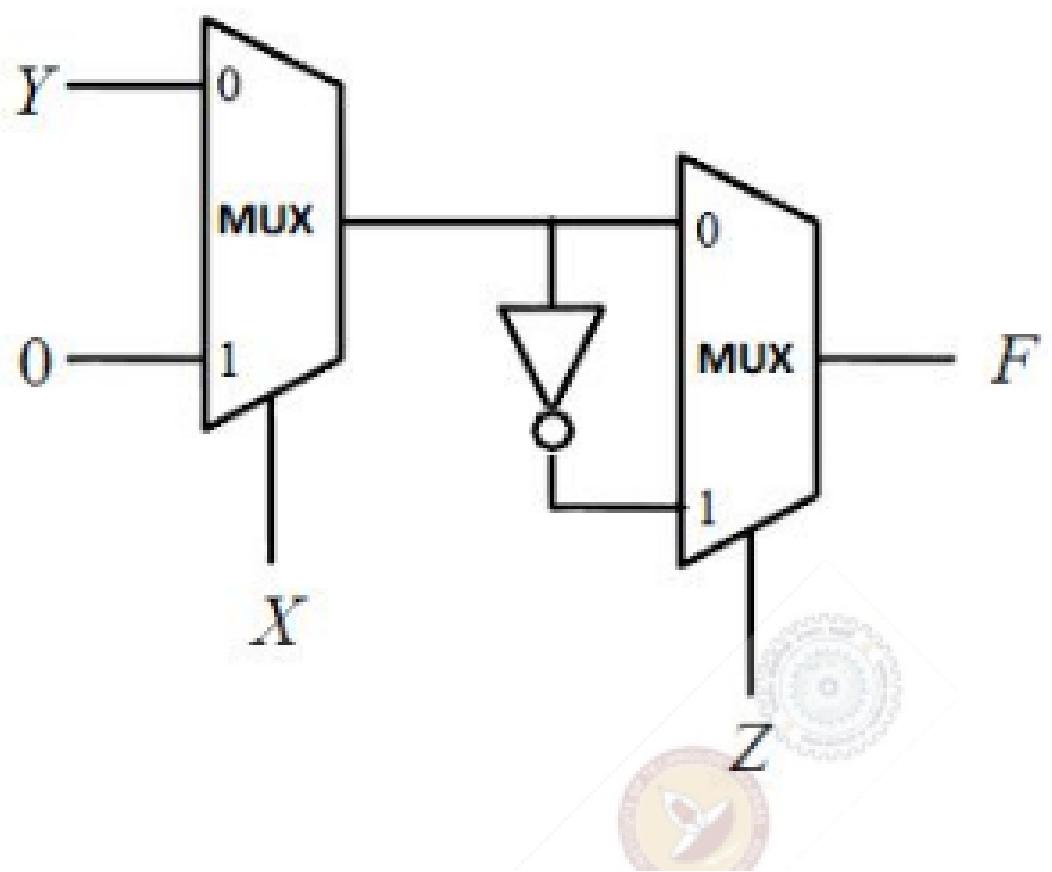


Figure 3.8:

24. Find the logic function implemented by the circuit given below in Fig. 3.11 (GATE EE 2019)

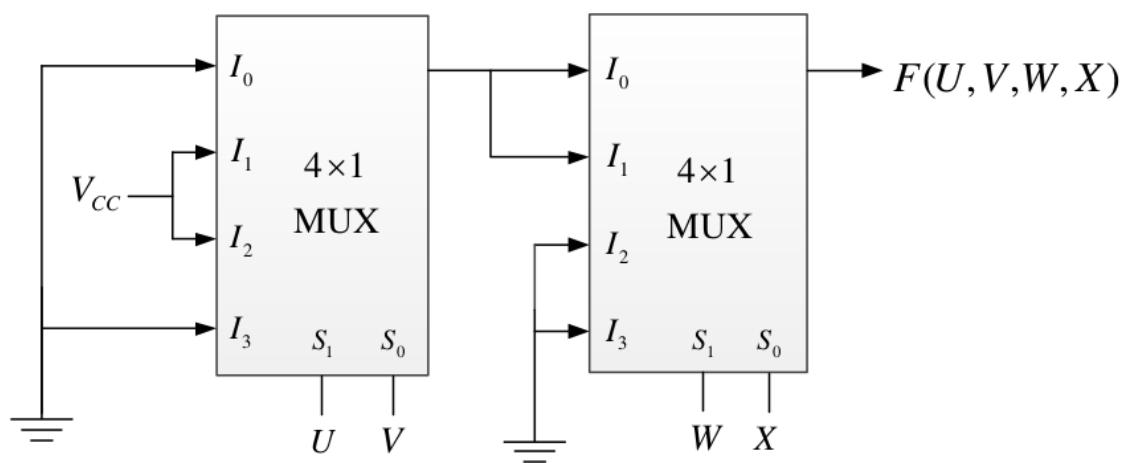


Figure 3.9:

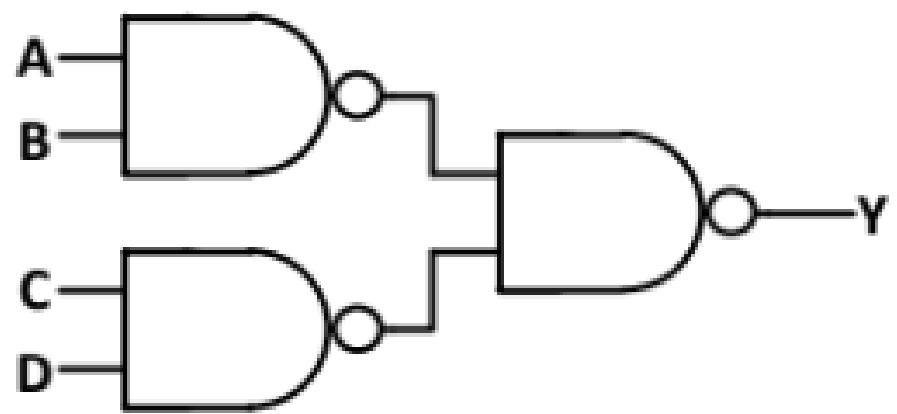


Figure 3.10:

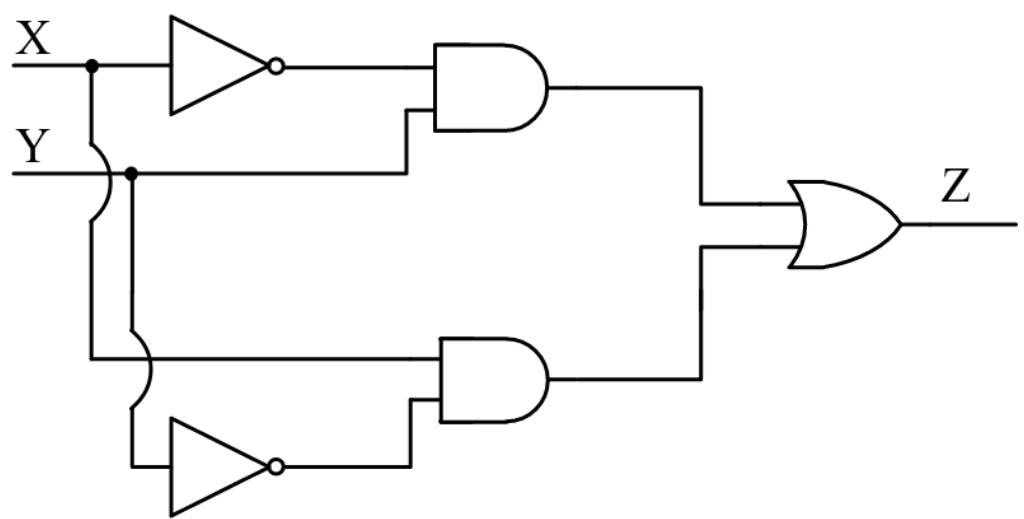


Figure 3.11:



# Chapter 4

## Karnaugh Map

### 4.1. Introduction

We explain Karnaugh maps (K-map) by finding the logic functions for the incrementing decoder

### 4.2. Incrementing Decoder

The incrementing decoder takes the numbers  $0, 1, \dots, 9$  in binary as inputs and generates the consecutive number as output. The corresponding truth table is available in Table 4.1.

### 4.3. Karnaugh Map

Using Boolean logic, output  $A$  in Table 4.1 can be expressed in terms of the inputs  $W, X, Y, Z$  as

$$\begin{aligned} A = & W'X'Y'Z' + W'XY'Z' + W'X'YZ' \\ & + W'XYZ' + W'X'Y'Z \quad (4.1) \end{aligned}$$

| Z | Y | X | W | D | C | B | A |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 4.1:

1. K-Map for  $A$ : The expression in (4.1) can be minimized using the K-map in Fig 4.1

In Fig 4.1, the implicants in boxes 0,2,4,6 result in  $W'Z'$  The implicants in boxes 0,8 result in  $W'X'Y'$  Thus, after minimization using Fig 4.2, (4.1) can be expressed as

$$A = W'Z' + W'X'Y' \quad (4.2)$$

Using the fact that

$$\begin{aligned} X + X' &= \\ XX' &= 0, \end{aligned} \quad (4.3)$$

derive (4.2) from (4.1) algebraically

2. K-Map for  $B$ : From Table 4.1, using boolean logic,

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \quad (4.4)$$

Show that (4.4) can be reduced to

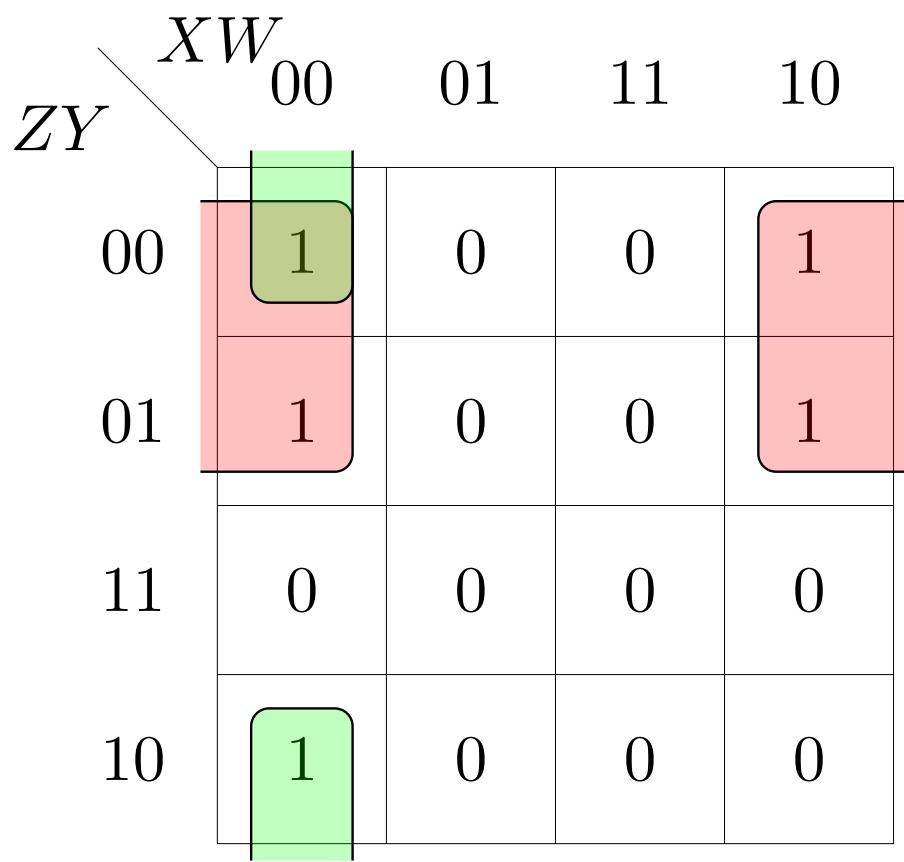


Figure 4.1: K-map for  $A$

$$B = WX'Z' + W'XZ' \quad (4.5)$$

using Fig 4.2

- Derive (4.5) from (4.4) algebraically using (4.3)

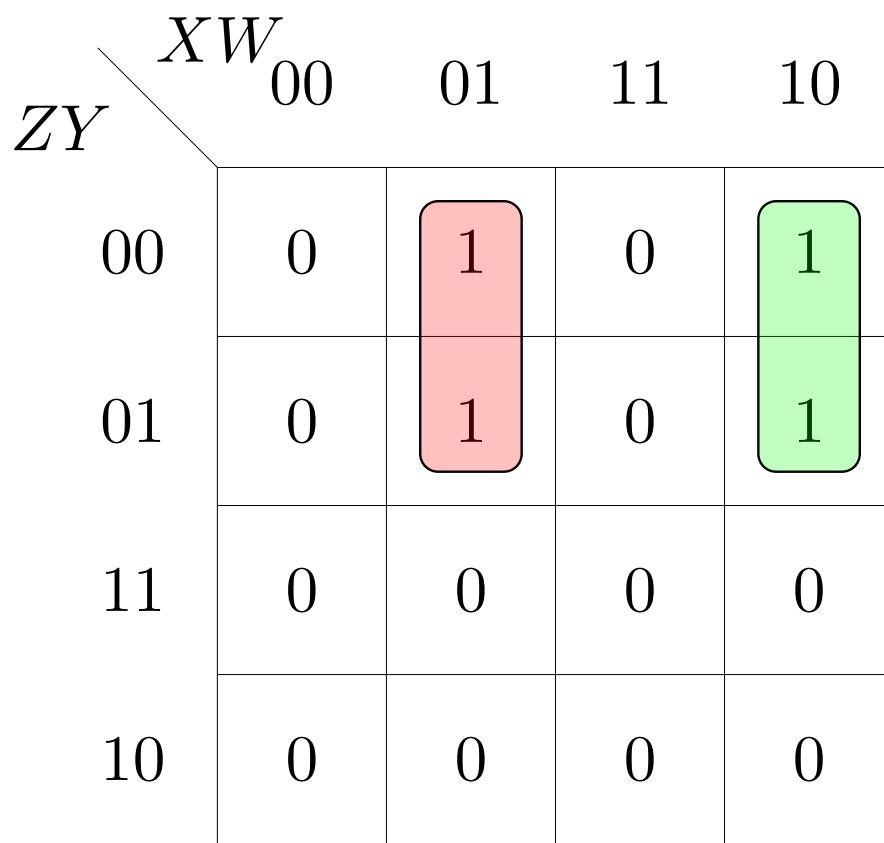


Figure 4.2: K-map for  $B$

4. K-Map for  $C$ : From Table 4.1, using boolean logic,

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \quad (4.6)$$

Show that (4.6) can be reduced to

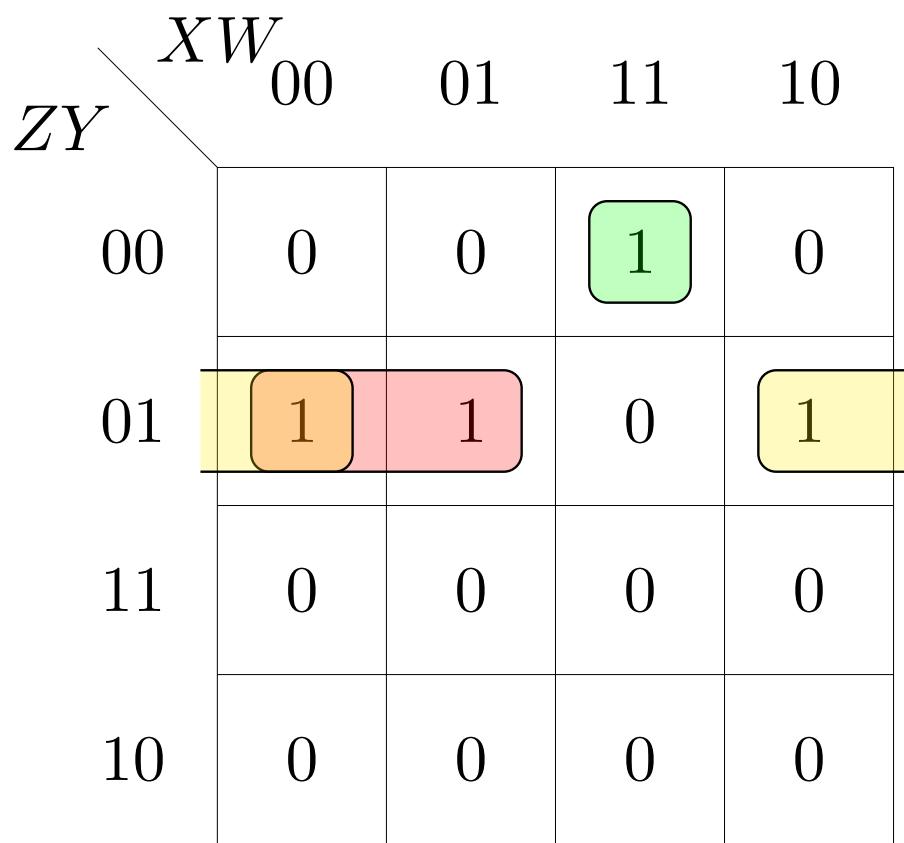


Figure 4.3: K-map for  $C$

$$C = WXY'Z' + X'YZ' + W'YZ' \quad (4.7)$$

using Fig 4.3

5. Derive (4.7) from (4.6) algebraically using (4.3)

6. K-Map for  $D$ : From Table 4.1, using boolean logic,

$$D = WXYZ' + W'X'Y'Z \quad (4.8)$$

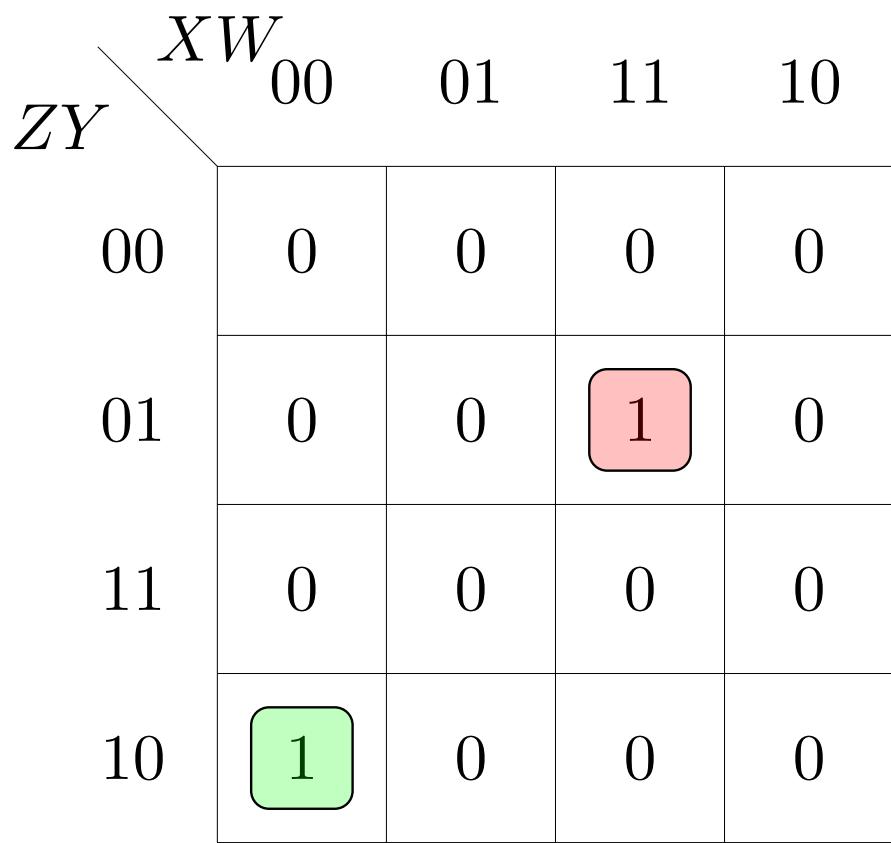


Figure 4.4: K-map for  $D$

7. Minimize (4.8) using Fig 4.4

| D | C | B | A | a | b | c | d | e | f | g | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0       |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1       |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2       |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3       |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 4       |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5       |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6       |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 7       |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8       |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 9       |

Table 4.2: Truth table for display decoder.

8. Execute the code in

```
ide/7447/codes/inc_dec/inc_dec.cpp
```

and modify it using the K-Map equations for A,B,C and D Execute and verify

9. Display Decoder: Table 4.2 is the truth table for the display decoder in Fig. 3.1. Use K-maps to obtain the minimized expressions for  $a, b, c, d, e, f, g$  in terms of  $A, B, C, D$  with and without don't care conditions

## 4.4. Dont Care

We explain Karnaugh maps (K-map) using don't care conditions

## 4.5. Don't Care Conditions

1. Don't Care Conditions: 4 binary digits are used in the incrementing decoder in Table 4.1. However, only the numbers from 0-9 are used as input/output in the decoder and we don't care about the numbers from 0-5. This phenomenon can be addressed by revising the truth table in Table 4.1 to obtain Table 4.3

| Z | Y | X | W | D | C | B | A |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | - | - | - | - |
| 1 | 0 | 1 | 1 | - | - | - | - |
| 1 | 1 | 0 | 0 | - | - | - | - |
| 1 | 1 | 0 | 1 | - | - | - | - |
| 1 | 1 | 1 | 0 | - | - | - | - |
| 1 | 1 | 1 | 1 | - | - | - | - |

Table 4.3:

2. The revised K-map for A is available in Fig 4.5. Show that

$$A = W' \quad (4.9)$$

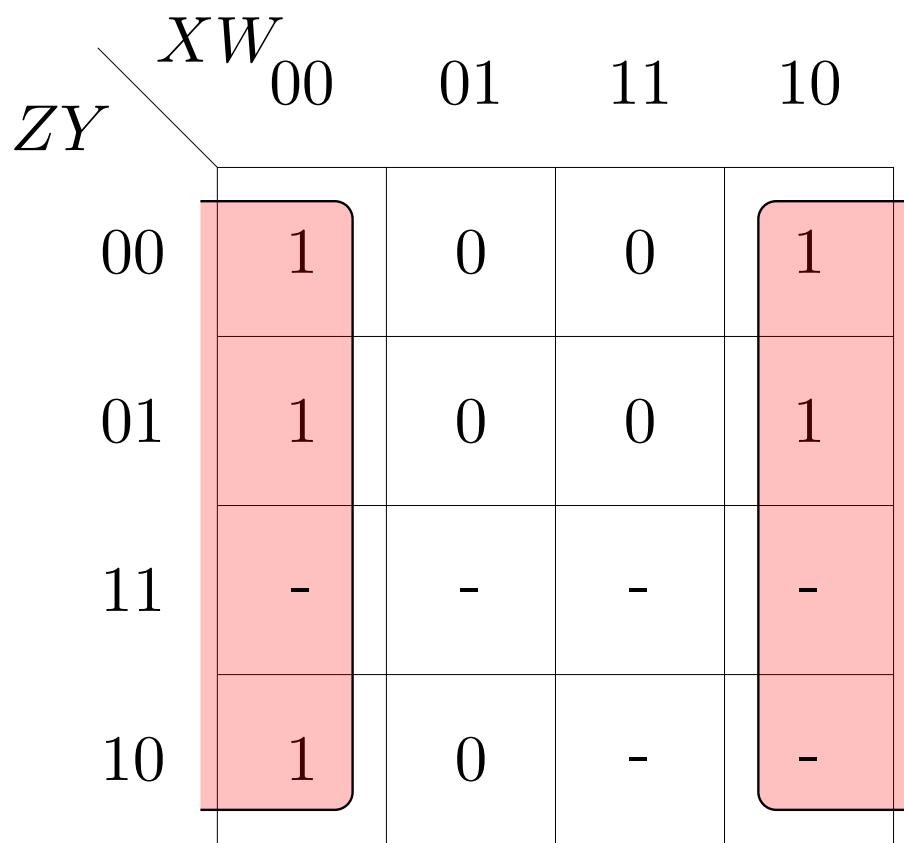


Figure 4.5: K-map for  $A$  with don't cares

3. The revised K-map for B is available in Fig 4.6 Show that

$$B = WX'Z' + W'X \quad (4.10)$$

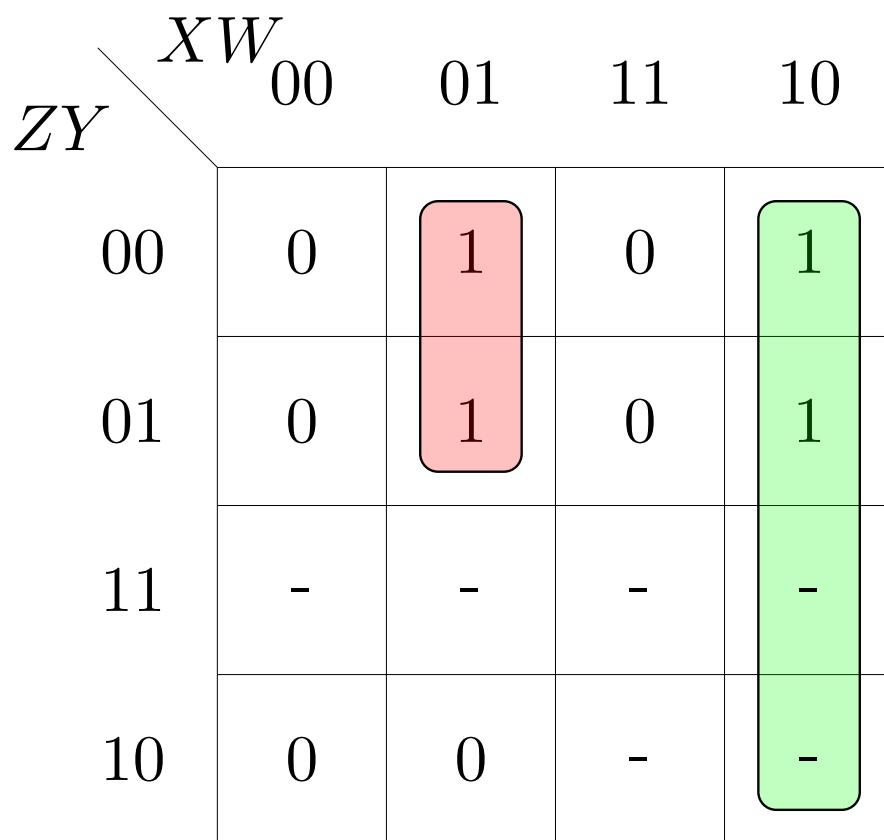


Figure 4.6: K-map for  $B$  with don't cares

4. The revised K-map for C is available in Fig 4.7 Show that

$$C = X'Y + W'Y + WXY' \quad (4.11)$$

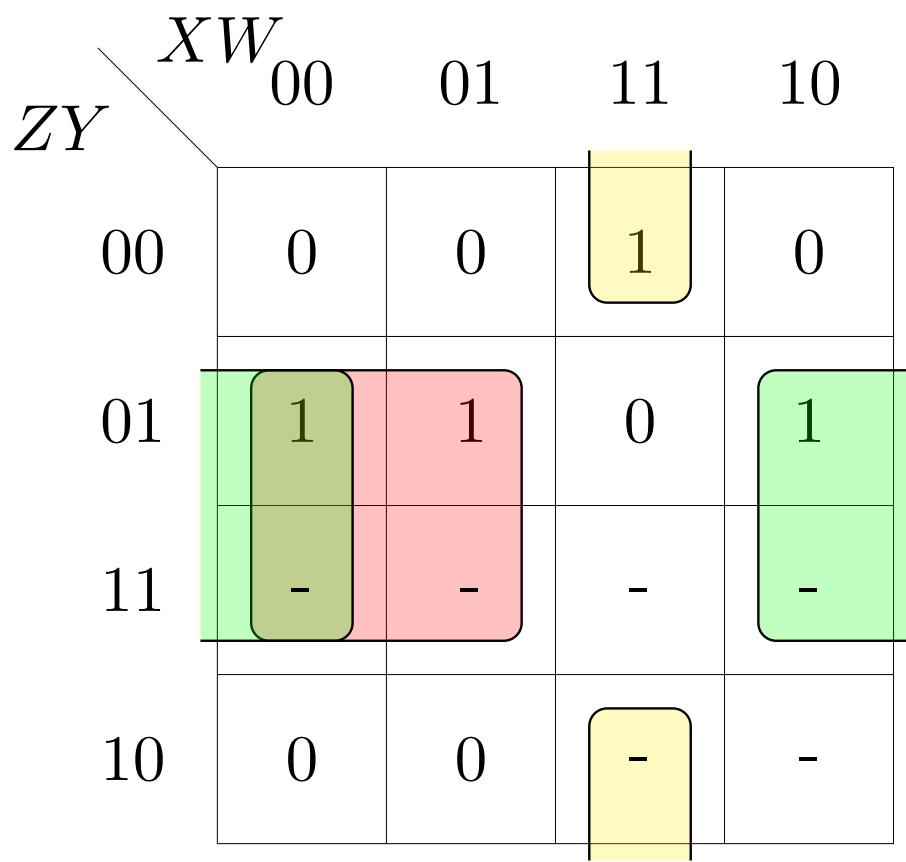


Figure 4.7: K-map for  $C$  with don't care

5. The revised K-map for D is available in Fig 4.8 Show that

$$D = W'Z + WXY \quad (4.12)$$

6. Verify the incrementing decoder with don't care conditions using the arduino

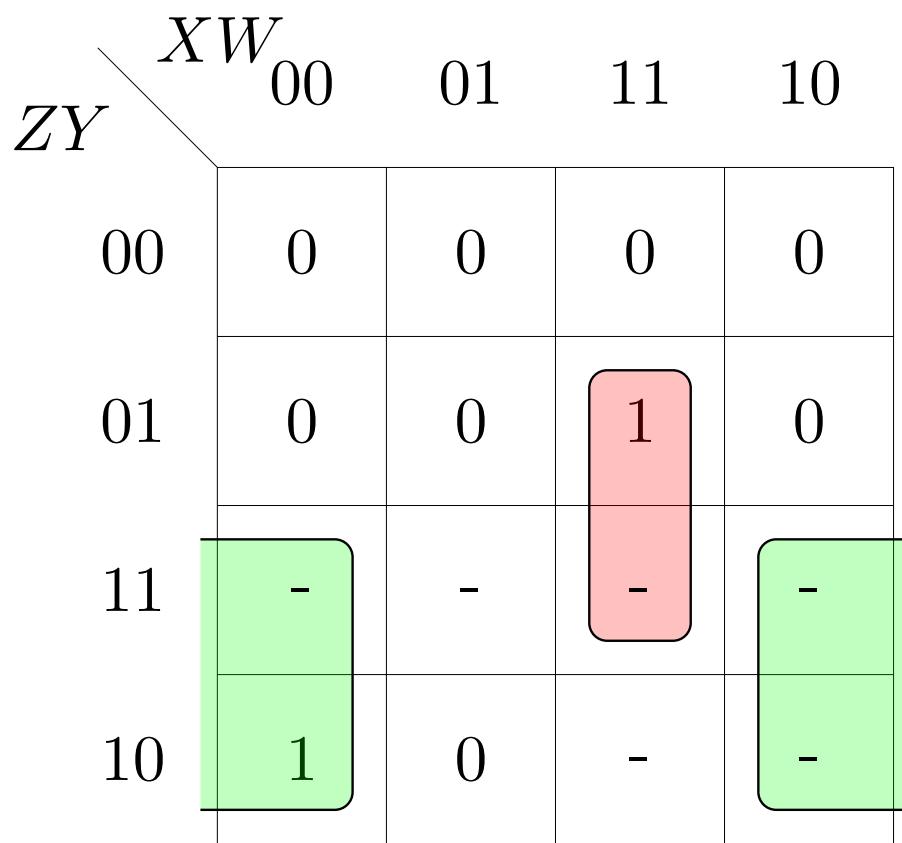


Figure 4.8: K-map for  $D$  with don't cares

7. Display Decoder: Use K-maps to obtain the minimized expressions for  $a, b, c, d, e, f, g$  in terms of  $A, B, C, D$  with don't care conditions
  
  
  
  
8. Verify the display decoder with don't care conditions using arduino

## 4.6. Problems

1. Obtain the Minimal Form for the Boolean Expression

(CBSE 2013)

$$H(P, Q, R, S) = \sum(0, 1, 2, 3, 5, 7, 8, 9, 10, 14, 15) \quad (4.13)$$

2. Write the POS form for the function G shown in Table 4.4.

(CBSE 2013)

| U | V | W | G |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Table 4.4:

3. Reduce the following Boolean Expression to its simplest form using K-Map (CBSE 2015)

$$F(X, Y, Z, W) = (0, 1, 4, 5, 6, 7, 8, 9, 11, 15) \quad (4.14)$$

4. Derive a Canonical POS expression for a Boolean function F, represented by the following truth table (CBSE 2015)
5. (CBSE 2015) Reduce the following Boolean Expression to its simplest form using

| X | Y | Z | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Table 4.5:

K-map

$$F(X, Y, Z, W) = \sum(0, 1, 6, 8, 9, 10, 11, 12, 15) \quad (4.15)$$

6. Reduce the following Boolean Expression to its simplest form using K-map. (CBSE 2016)

$$F(X, Y, Z, W) = \sum(2, 6, 7, 8, 9, 10, 11, 13, 14, 15) \quad (4.16)$$

7. Derive a Canonical POS expression for a Boolean function F, represented in Table 4.6 (CBSE 2016)

8. Verify the following (CBSE 2016)

$$A' + B'C = A'B'C' + A'BC' + A'BC + A'B'C + AB'C \quad (4.17)$$

9. Reduce the following boolean expression to it's simplest form using K-Map (CBSE

| P | Q | R | F(P, Q, R) |
|---|---|---|------------|
| 0 | 0 | 0 | 0          |
| 0 | 0 | 1 | 1          |
| 0 | 1 | 0 | 1          |
| 0 | 1 | 1 | 0          |
| 1 | 0 | 0 | 0          |
| 1 | 0 | 1 | 0          |
| 1 | 1 | 0 | 1          |
| 1 | 1 | 1 | 1          |

Table 4.6:

2017)

$$F(X, Y, Z, W) = \sum(0, 1, 2, 3, 4, 5, 10, 11, 14) \quad (4.18)$$

10. Reduce the following Boolean Expression to its simplest form using K-Map. (CBSE 2017)

$$E(U, V, Z, W) = (2, 3, 6, 8, 9, 10, 11, 12, 13) \quad (4.19)$$

11. Derive a canonical POS expression for a Boolean function  $G$ , represented by Table 4.7 (CBSE 2017)

12. Derive a canonical POS expression for a Boolean function  $FN$ , represented by Table 4.8. (CBSE 2018)

13. Reduce the following Boolean expression in the simplest form using K-Map.

$$F(P, Q, R, S) = \sum(0, 1, 2, 3, 5, 6, 7, 10, 14, 15) \quad (4.20)$$

| <b>X</b> | <b>Y</b> | <b>Z</b> | <b>G(X,Y,Z)</b> |
|----------|----------|----------|-----------------|
| 0        | 0        | 0        | 0               |
| 0        | 0        | 1        | 0               |
| 0        | 1        | 0        | 1               |
| 0        | 1        | 1        | 0               |
| 1        | 0        | 0        | 1               |
| 1        | 0        | 1        | 1               |
| 1        | 1        | 0        | 0               |
| 1        | 1        | 1        | 1               |

Table 4.7:

| <b>X</b> | <b>Y</b> | <b>Z</b> | <b>FN(X,Y,Z)</b> |
|----------|----------|----------|------------------|
| 0        | 0        | 0        | 1                |
| 0        | 0        | 1        | 1                |
| 0        | 1        | 0        | 0                |
| 0        | 1        | 1        | 0                |
| 1        | 0        | 0        | 1                |
| 1        | 0        | 1        | 0                |
| 1        | 1        | 0        | 0                |
| 1        | 1        | 1        | 1                |

Table 4.8:

(CBSE 2019)

14. Fig. 4.9 below shows a multiplexer where S0 and S1 are the select lines, I0 to I3 are the input lines, EN is the enable line and F(P,Q,R) is the output. Find the boolean expression for output F as function of inputs P,Q,R using K-map. (GATE EC 2020)

15. The four variable function  $f$  is given in terms of min-terms as

$$f(A, B, C, D) = \sum m(2, 3, 8, 10, 11, 12, 14, 15) \quad (4.21)$$

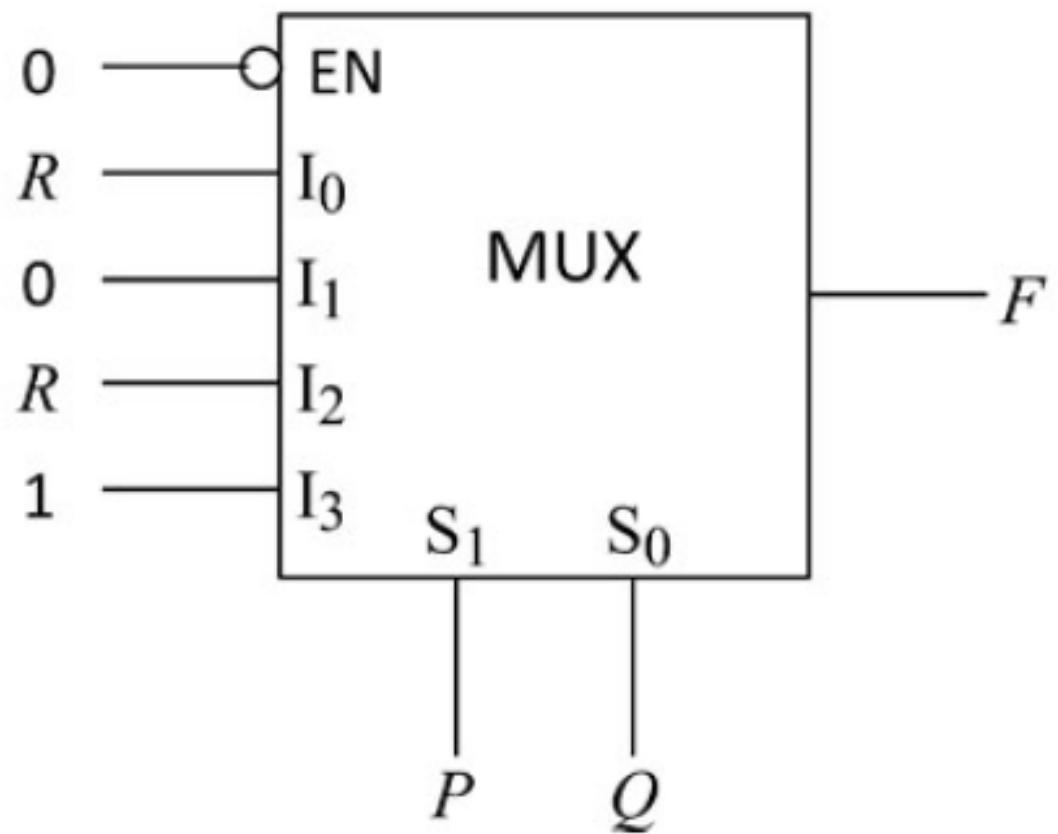


Figure 4.9:

Using the K-map minimize the function in the sum of products form. (GATE EC 1991)

16. Find the logic realized by the circuit in Fig. 4.10. (GATE EC 1992)

17. A combinational circuit has three inputs  $A$ ,  $B$  and  $C$  and an output  $F$ .  $F$  is true only for the following input combinations. (GATE EC 1992)

- (a)  $A$  is false and  $B$  is true

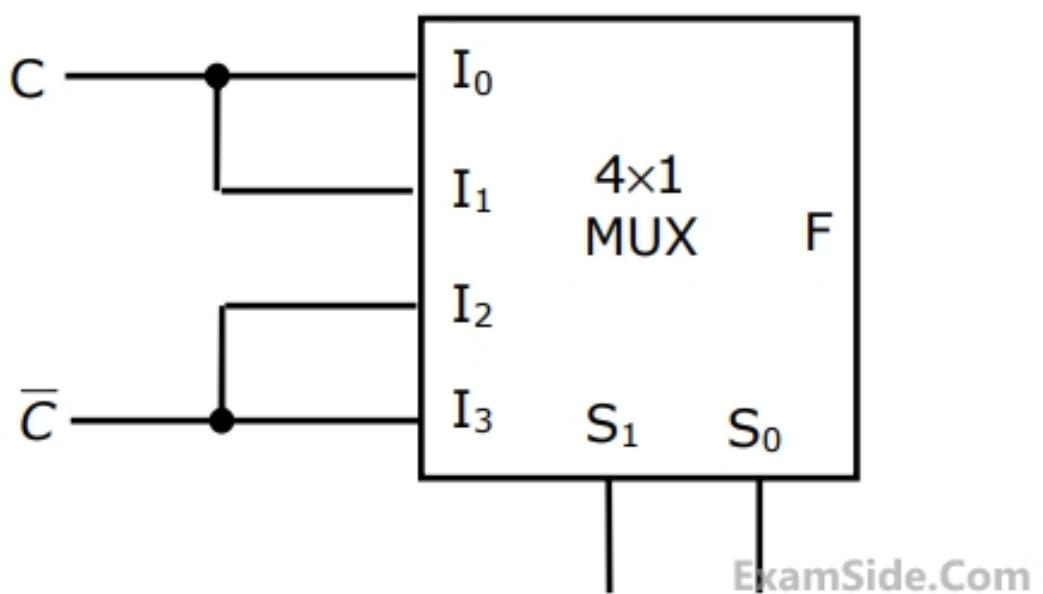


Figure 4.10:

- (b) A is false and C is true
  - (c) A, B and C are all false
  - (d) A, B and C are all true
- 
- (a) Write the truth table for F. use the convention, true = 1 and false = 0.
  - (b) Write the simplified expression for F as a Sum of Products.
  - (c) Write the simplified expression for F as a product of Sums.
18. Draw the logic circuit for Table 4.9 using only NOR gates. (GATE EC 1993)

| <b>C</b> | <b>B</b> | <b>A</b> | <b>Y</b> |
|----------|----------|----------|----------|
| 0        | 0        | 0        | 1        |
| 0        | 0        | 1        | 1        |
| 0        | 1        | 0        | 1        |
| 0        | 1        | 1        | 0        |
| 1        | 0        | 0        | 1        |
| 1        | 0        | 1        | 0        |
| 1        | 1        | 0        | 0        |
| 1        | 1        | 1        | 0        |

Table 4.9:

19. Implement the following Boolean function in a 8x1 multiplexer. (GATE EC 1993)

$$Q = BC + ABD' + A'C'D \quad (4.22)$$

20. Minimize the following Boolean function in 4.23.

$$F = A'B'C' + A'BC' + A'BC + ABC' \quad (4.23)$$

21. Find the Boolean expression for Table 4.10. (GATE EC 2005)

| <b>A</b> | <b>B</b> | <b>C</b> | <b>X</b> |
|----------|----------|----------|----------|
| 0        | 0        | 0        | 0        |
| 0        | 0        | 1        | 0        |
| 0        | 1        | 0        | 0        |
| 0        | 1        | 1        | 1        |
| 1        | 0        | 0        | 0        |
| 1        | 0        | 1        | 0        |
| 1        | 1        | 0        | 1        |
| 1        | 1        | 1        | 0        |

Table 4.10:

22. Minimize the logic function represented by the following Karnaugh map. (CBSE

|   |  | YZ |    |    |    |   |
|---|--|----|----|----|----|---|
|   |  | 00 | 01 | 11 | 10 |   |
| X |  | 0  | 1  | 1  | 1  | 0 |
|   |  | 1  | 0  | 0  | 1  | 0 |

2021)

23. Find the output for the Karnaugh map shown below (GATE EE 2019)

|    |  | PQ |    |    |    |   |
|----|--|----|----|----|----|---|
|    |  | 00 | 01 | 11 | 10 |   |
| RS |  | 00 | 0  | 1  | 1  | 0 |
|    |  | 01 | 1  | 1  | 1  | 1 |
|    |  | 11 | 1  | 1  | 1  | 1 |
|    |  | 10 | 0  | 0  | 0  | 0 |

24. The propagation delays of the XOR gate, AND gate and multiplexer (MUX) in the circuit shown in the Fig. 4.11 are 4 ns, 2 ns and 1 ns, respectively. If all the inputs P, Q, R, S and T are applied simultaneously and held constant, the maximum propagation delay of the circuit is (Gate EC-2021)

(a) 3 ns

(b) 5 ns

(c) 6 ns

(d) 7 ns

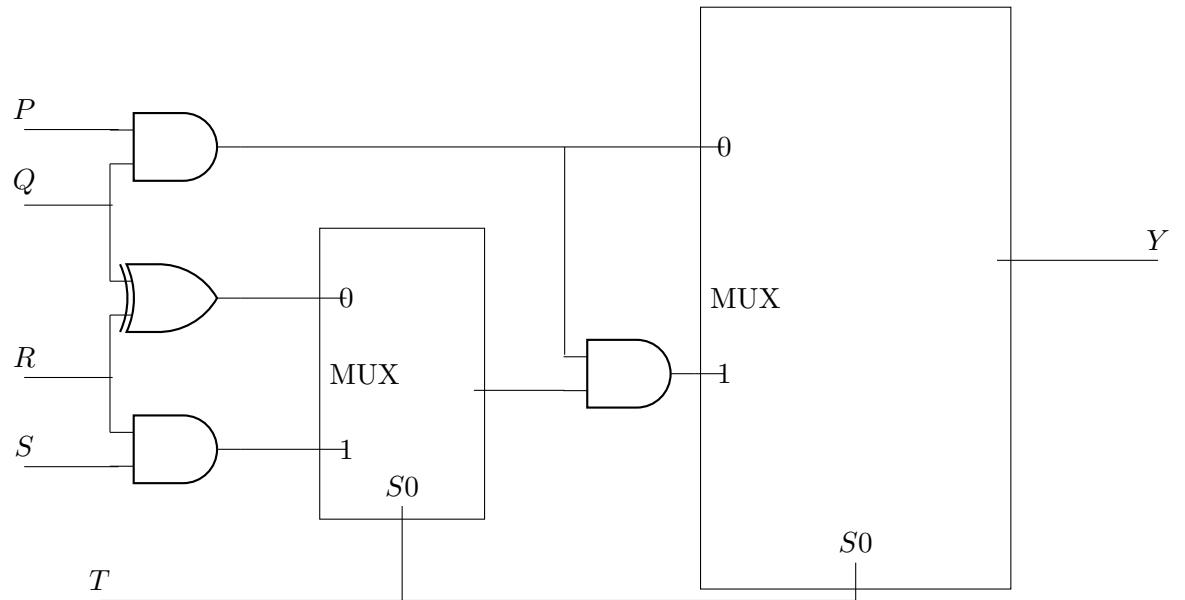


Figure 4.11:

25. Consider the 2-bit multiplexer(MUX) shown in the figure. For output to be the XOR of R and S, the values for  $W, X, Y$  and  $Z$  are ? (GATE EC-2022)

- (a)  $W = 0, X = 0, Y = 1, Z = 1$
- (b)  $W = 1, X = 0, Y = 1, Z = 0$
- (c)  $W = 0, X = 1, Y = 1, Z = 0$
- (d)  $W = 1, X = 1, Y = 0, Z = 0$

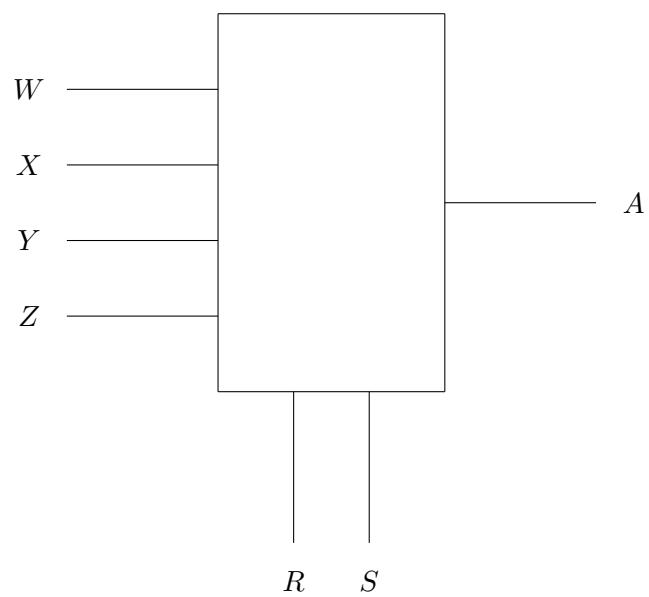


Figure 4.12:

# Chapter 5

7474

We show how to use the 7474 D-Flip Flop ICs in a sequential circuit to realize a decade counter.

## 5.1. Components

| Component             | Value            | Quantity |
|-----------------------|------------------|----------|
| Breadboard            |                  | 1        |
| Resistor              | $\geq 220\Omega$ | 1        |
| Arduino               | Uno              | 1        |
| Seven Segment Display | Common Anode     | 1        |
| Decoder               | 7447             | 1        |
| Flip Flop             | 7474             | 2        |
| Jumper Wires          |                  | 20       |

Table 5.1:

## 5.2. Decade Counter

1. Generate the CLOCK signal using the **blink** program.

|         | INPUT |    |    |    | OUTPUT |    |    |    | CLOCK | 5V   |      |    |   |    |    |
|---------|-------|----|----|----|--------|----|----|----|-------|------|------|----|---|----|----|
|         | W     | X  | Y  | Z  | A      | B  | C  | D  |       |      |      |    |   |    |    |
| Arduino | D6    | D7 | D8 | D9 | D2     | D3 | D4 | D5 | D13   |      |      |    |   |    |    |
| 7474    | 5     | 9  |    |    | 2      | 12 |    |    |       | CLK1 | CLK2 | 1  | 4 | 10 | 13 |
| 7474    |       |    | 5  | 9  |        |    | 2  | 12 |       | CLK1 | CLK2 | 1  | 4 | 10 | 13 |
| 7447    |       |    |    |    | 7      | 1  | 2  | 6  |       |      |      | 16 |   |    |    |

Table 5.2:

2. Connect the Arduino, 7447 and the two 7474 ICs according to Table 5.2 and Fig. 5.2.

The pin diagram for 7474 is available in Fig. 5.1

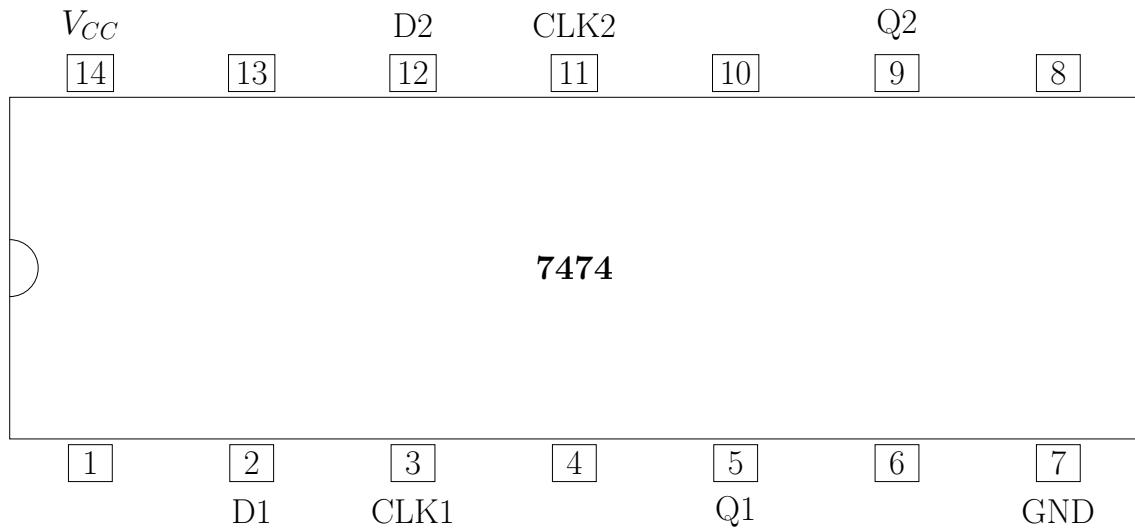


Figure 5.1:

3. Realize the decade counter in Fig. 5.2.

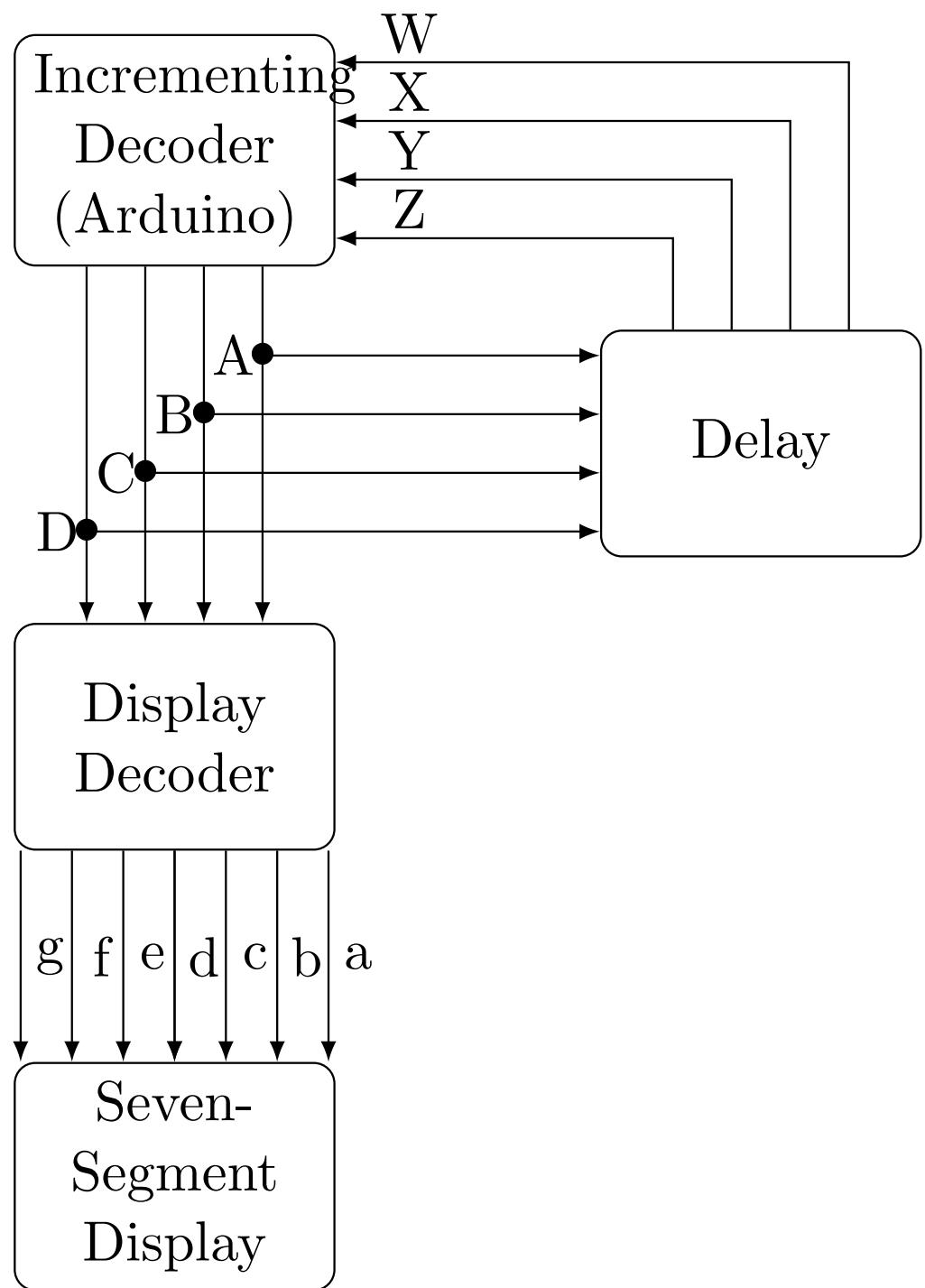


Figure 5.2:



# Chapter 6

## Finite State Machine

We explain a state machine by deconstructing the decade counter

### 6.1. The Decade Counter

The block diagram of a decade counter (repeatedly counts up from 0 to 9) is available in Fig 5.2 The incrementing decoder and display decoder are part of combinational logic, while the delay is part of sequential logic

### 6.2. Finite State Machine

1. Fig 6.1 shows a finite state machine (FSM) diagram for the decade counter in Fig 5.2  
 $s_0$  is the state when the input to the incrementing decoder is 0 The state transition table for the FSM is Table 4.1, where the present state is denoted by the variables  $W, X, Y, Z$  and the next state by  $A, B, C, D$ .
2. The FSM implementation is available in Fig 6.2 The flip-flops hold the input for the time that is given by the clock This is nothing but the implementation of the Delay block in Fig 5.2

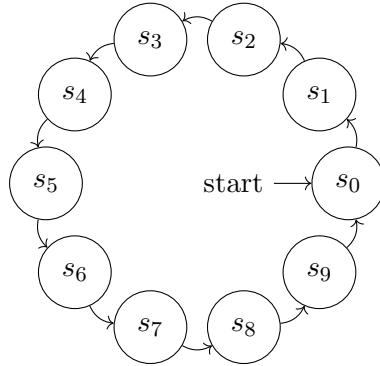


Figure 6.1: FSM for the decade counter

3. The hardware cost of the system is given by

$$\text{No of D Flip-Flops} = \lceil \log_2 (\text{No of States}) \rceil \quad (6.1)$$

For the FSM in Fig 6.1, the number of states is 9, hence the number flipflops required  
 $= 4$

4. Draw the state transition diagram for a decade down counter (counts from 9 to 0 repeatedly) using an FSM
5. Write the state transition table for the down counter
6. Obtain the state transition equations with and without don't cares
7. Verify your design using an arduino

## 6.3. Problems

1. The digital circuit shown in Fig. 6.3 generates a modified clockpulse at the output.  
Sketch the output waveform. (GATE EE 2004)

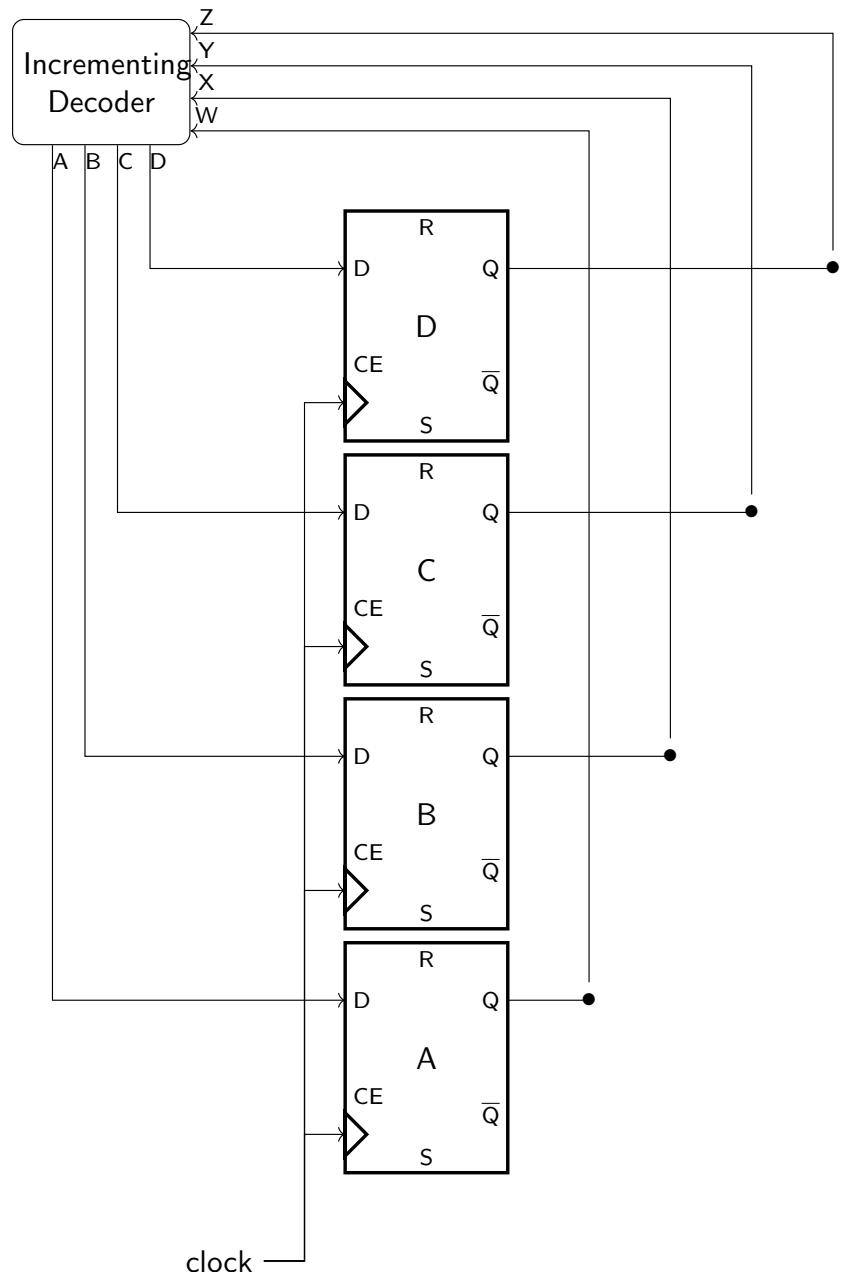


Figure 6.2: Decade counter FSM implementation using D-Flip Flops

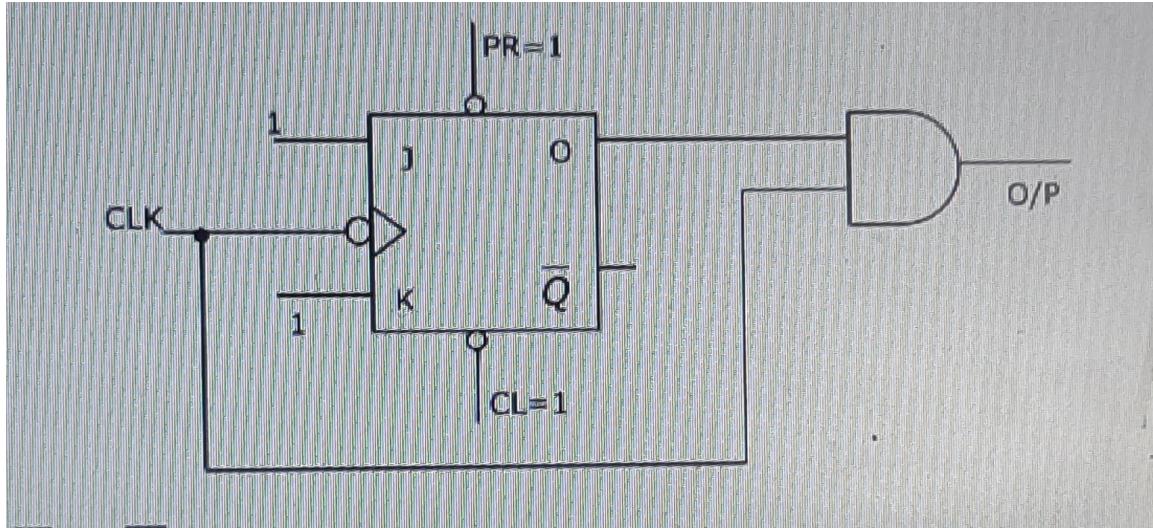


Figure 6.3:

2. The circuit shown in the figure below uses ideal positive edge-triggered synchronous J-K flip flops with outputs X and Y. If the initial state of the output is  $X=0$  and  $Y=0$ , just before the arrival of the first clock pulse, the state of the output just before the arrival of the second clock pulse is  
(GATE IN 2019)

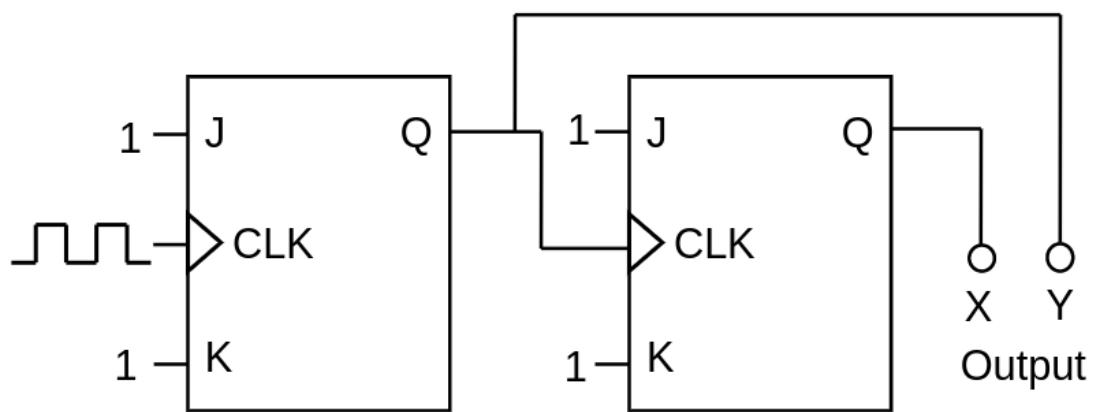


Figure 6.4:

3. The state diagram of a sequence detector is shown in Fig. 6.5 . State  $S_0$  is the initial state of the sequence detector. If the output is 1, then (GATE EC 2020)

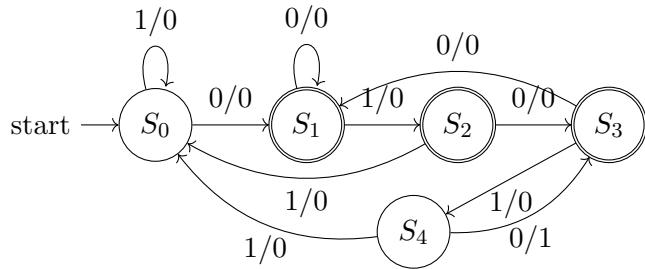


Figure 6.5: State diagram

- (a) the sequence 01010 is detected
  - (b) the sequence 01011 is detected
  - (c) the sequence 01110 is detected
  - (d) the sequence 01001 is detected
4. A counter is constructed with three D flip-flops. The input-output pairs are named (D0, Q0), (D1, Q1), and (D2, Q2), where the subscript 0 denotes the least significant bit. The output sequence is desired to be the Gray-code sequence 000, 001, 011, 010, 110, 111, 101, and 100, repeating periodically. Note that the bits are listed in the Q2 Q1 Q0 format. Find the combinational logic expression for D1. (GATE EE 2021)

5. For the circuit shown in Fig. 6.6, the clock frequency is  $f_0$  and the duty cycle is 25%. For the signal at the  $Q$  output of the Flip-Flop,

- (a) frequency of  $\frac{f_0}{4}$  and duty cycle is 50%
- (b) frequency of  $\frac{f_0}{4}$  and duty cycle is 25%
- (c) frequency of  $\frac{f_0}{2}$  and duty cycle is 50%

(d) frequency of  $f_0$  and duty cycle is 25%

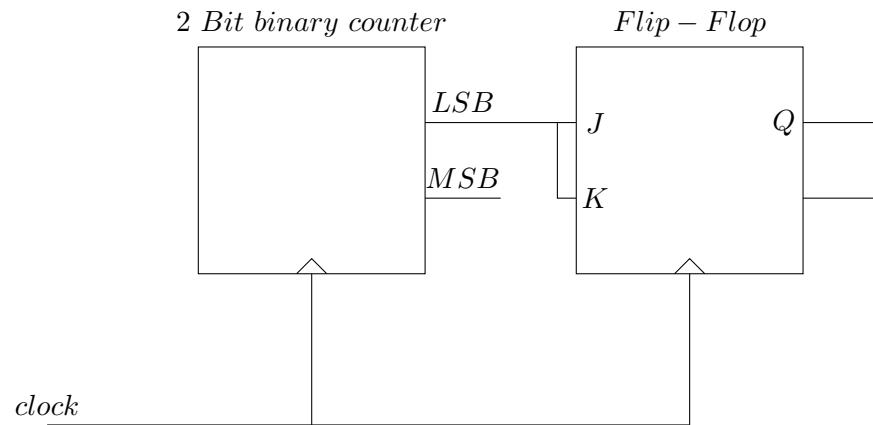


Figure 6.6:

(GATE EC-2022)

# Chapter 7

# Assembly Programming

This manual shows how to setup the assembly programming environment for the arduino.

## 7.1. Software Installation

1. Find the USB port to which arduino is connected.

```
%Finding the port  
  
sudo dmesg | grep tty  
  
%The output will be something like  
[ 6.153362] cdc_acm 1-1.2:1.0: ttyACM0: USB ACM device  
  
%and your port number is ttyACM0
```

2. Copy the .inc file to your home directory

```
cp assembly/setup/m328Pdef/m328Pdef.inc ~/
```

3. Execute

```
avr assemly/setup/codes/hello.asm
```

as

4. Then flash the .hex file

```
hello.hex
```

5. You should see the led beside pin 13 light up.

6. Now edit **hello.asm** by modifying the line to

```
ldi r17,0b00000000
```

Save and execute. The led should turn off.

7. What do the following instructions do?

```
ldi r16,0b00100000
```

```
out DDRB,r16
```

**Solution:** The Atmega328p microcontroller for the arduino board has 32 internal 8-bit registers, R0-R31. R16-R31 can be used directly for i/o. The first instruction loads an 8-bit binary number into R16. The second instruction loads the value in R16 to the DDRB register. Each bit of the DDRB register corresponds to a pin on the arduino. The second instruction declares pin 13 to be an output port. Both the instructions are equivalent to pinMode(13, OUTPUT).

8. What do the following instructions do?

```
ldi r17,0b00100000
```

```
out PortB,r17
```

**Solution:** The instructions are equivalent to digitalWrite(13).

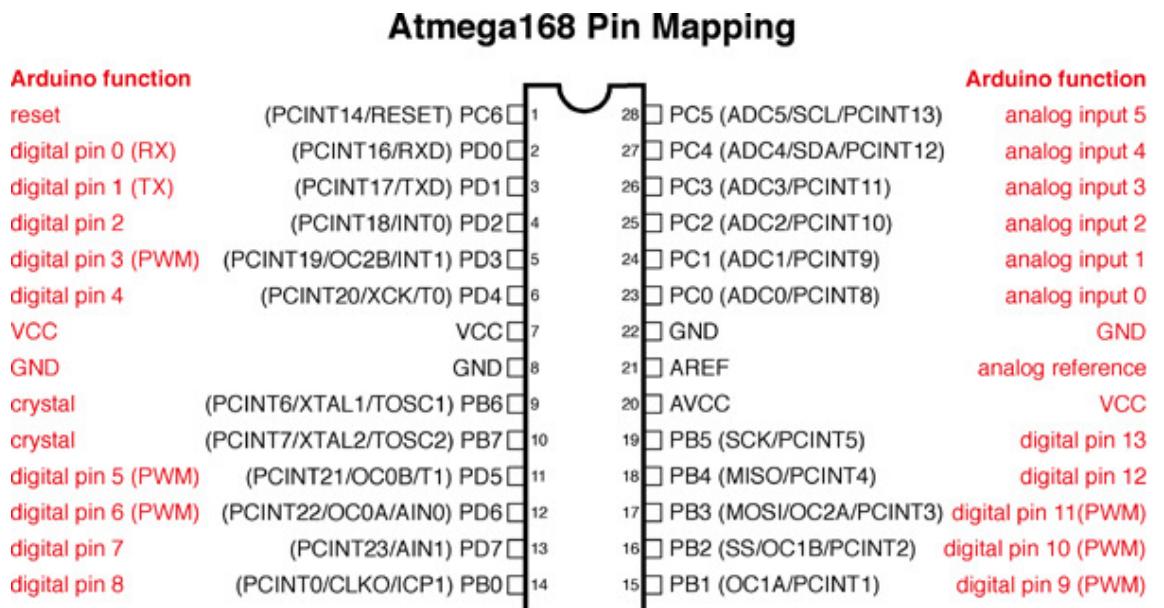
The objective of this manual is to show how to control a seven segment display through the AVR-Assembly.

## 7.2. Seven Segment Display

1. See Table 2.1 for components.
2. Complete Table 2 for all the digital pins using Fig. 2.

| Port Pin | Digital Pin |
|----------|-------------|
| PD2      | 2           |
| PB5      | 13          |

Table 2:



Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Figure 2:

3. Make connections according to Table 3.

| <b>Arduino</b> | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
|----------------|-----|-----|-----|-----|-----|-----|-----|
|                | PD2 | PD3 | PD4 | PD5 | PD6 | PD7 | PB0 |
| <b>Display</b> | a   | b   | c   | d   | e   | f   | g   |
| <b>2</b>       | 0   | 0   | 1   | 0   | 0   | 1   | 0   |

Table 3:

4. Execute the following code. The number 2 should be displayed.

```
;using assembly language for
;displaying number on
;seven segment display

.include "/home/gadepall/m328Pdef.inc"

;Configuring pins 2–7 (PD2–PD7) of Arduino
;as output
ldi r16,0b11111100
out DDRD,r16

;Configuring pin 8 (PB0) of Arduino
;as output
ldi r16,0b00000001
out DDRB,r16

;Writing the number 2 on the
;seven segment display
ldi r17,0b10010000
out PortD,r17
```

```

ldi r17,0b00000000
out PortB,r17
Start:
rjmp Start

```

- Now generate the numbers 0-9 by modifying the above program.

## 7.3. 7447

This manual shows how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

### 7.3.1. Components

| Component             | Value   | Quantity |
|-----------------------|---------|----------|
| Resistor              | 220 Ohm | 1        |
| Arduino               | UNO     | 1        |
| Seven Segment Display |         | 1        |
| Decoder               | 7447    | 1        |
| Jumper Wires          | M-M     | 20       |
| Breadboard            |         | 1        |

### 7.3.2. Boolean Operations

1. Verify the AND,OR and XOR operations in assembly using the following code and making pin connections according to Table 1.

```
wget https://raw.githubusercontent.com/gadepall/arduino/master/assembly/7447/
      count/codes/and_or_xor.asm
```

|                |   |   |   |   |
|----------------|---|---|---|---|
| <b>7447</b>    | D | C | B | A |
| <b>Arduino</b> | 5 | 4 | 3 | 2 |

Table 1:

2. Suppose R20=0b00000010, R16=0b00000001. Explain the following routine

```
loopw: lsl r16 ;left shift
        dec r20 ;counter --
        brne loopw ;if counter != 0
        ret
```

**Solution:** The routine shifts R16 by 2 bits to the left (the count in R20=2). At the end of the routine, R16=0b00000100.

3. What do the following instructions do?

```
rcall loopw
out PORTD,r16 ;writing output to pins 2,3,4,5
```

**Solution:** **rcall** calls for execution of the **loopw** routine, which shifts R16 by 2 bits to the left and writes R16 to the display through PORTD.

4. Use the following routine for finding the complement of a number.

```
wget https://raw.githubusercontent.com/gadepall/arduino/master/assembly/7447/
      count/codes/complement.asm
```

5. Write an assembly program for implementing the following equations. Note that ZYXW is the input nibble and DCBA is the output nibble. Display DCBA on the seven segment display for each input ZYXW from 0-9.

$$A = W' \quad (7.1)$$

$$B = WX'Z' + W'X \quad (7.2)$$

$$C = WXY' + X'Y + W'Y \quad (7.3)$$

$$D = WXY + W'Z \quad (7.4)$$

6. Repeat the above exercise by getting ZYXW as manual inputs to the arduino from the GND and 5V pins on the breadboard.

This manual shows how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

### 7.3.3. Controlling the Display

1. Connect the 7447 IC to the seven segment display.
2. Make connections between the 7447 and the arduino according to Table 2

| 7447    | D | C | B | A |
|---------|---|---|---|---|
| Arduino | 5 | 4 | 3 | 2 |

Table 2:

3. Execute the following program. The number 5 will be displayed.

```
assembly/7447/io/codes/op_7447.asm
```

4. Now generate the numbers 0-9 by modifying the above program.

5. Execute the following program after making the connections in Table 5. The number 3 will be displayed. What does the program do?

```
assembly/7447/io/codes/ip_7447.asm
```

|         | Z  | Y  | X  | W  |
|---------|----|----|----|----|
| Input   | 0  | 0  | 1  | 1  |
| Arduino | 13 | 12 | 11 | 10 |

Table 5:

**Solution:** The program reads from pins 10-13 and displays the equivalent decimal value on the display by writing to pins 2-5 of the arduino.

6. Explain the following instructions

```
ldi r17, 0b11000011 ; identifying input pins 10,11,12,13  
ldi r17, 0b11111111 ;  
out PORTB,r17 ;  
in r17,PINB
```

**Solution:** First define pins 10,11,12 and 13 as input pins. Then ensure that these pins have the input 1 by default. Load the inputs from the pins in port B (which includes pins 10-13) into R17.

## 7.4. Timer

This manual shows how to use the Atmega328p timer to blink the builtin led with a delay.

### 7.4.1. Components

| Component | Value | Quantity |
|-----------|-------|----------|
| Arduino   | UNO   | 1        |

### 7.4.2. Blink through TIMER

1. Connect the Arduino to the computer and execute the following code

```
assembly/timer/codes/timer.asm
```

2. Explain the following instruction

```
sbi DDRB, 5
```

3. What do the following instructions do?

```
ldi r16, 0b00000101  
out TCCR0B, r16
```

**Solution:** The system clock (SYSCLK) frequency of the Atmega328p is 16 MHz.

TCCR0B is the Timer Counter Control Register. When

$$TCCR0B = 0b101 \quad (7.5)$$

$$\Rightarrow CLK = \frac{SYSCLK}{1024} \quad (7.6)$$

$$= \frac{16M}{1K} = 16kHz. \quad (7.7)$$

4. Explain the PAUSE routine.

```
ldi r19, 0b01000000 ;times to run the loop = 64 for 1 second delay
PAUSE: ;this is delay (function)
lp2: ;loop runs 64 times
      IN r16, TIFR0 ;tifr is timer interrupt flag (8 bit timer runs 256
                     times)
      ldi r17, 0b00000010
      AND r16, r17 ;need second bit
      BREQ PAUSE
      OUT TIFR0, r17 ;set tifr flag high
      dec r19
      brne lp2
      ret
```

**Solution:** TIFR0 is the timer interrupt flag and TIFR0=0bxxxxxx10 after every 256 cycles. PAUSE routine waits till TIFR0=0bxxxxxx10, this checking is done by the AND and BREQ instructions above.

5. Explain the lp2 routine.

**Solution:** R19 = 64 and is used as a count for lp2. The lp2 routine returns after 64 PAUSE routines.

6. What is the blinking delay?

**Solution:** The blinking delay is given by

$$delay = \frac{CLK}{lp2 \times PAUSE} \text{seconds} \quad (7.8)$$

$$= \frac{16 \times 1024}{64 \times 256} \text{seconds} = 1 \text{second} \quad (7.9)$$

### 7.4.3. Blink through Cycle Delays

1. Connect pin 8 of the Arduino to an led and execute the following code

```
assembly/timer/codes/cycle_delay.asm
```

2. Explain how the delay is obtained

```
ldi r16,0x50  
ldi r17,0x00  
ldi r18,0x00
```

```
w0:
```

```
dec r18  
brne w0  
dec r17  
brne w0  
dec r16  
brne w0  
pop r18  
pop r17  
pop r16
```

```
ret
```

**Solution:** The w0 loop is executed using the counts in  $R16=2^6+2^4 = 80$ ,  $R17=R18=2^8 = 256$ . Thus

$$delay \approx 80 \times 256 \times 256 \text{cycles} \quad (7.10)$$

$$= \frac{80 \times 256 \times 256}{2^4 \times 2^{20}} \text{seconds} \quad (7.11)$$

$$= 0.3125 \text{seconds} \quad (7.12)$$

The actual time is slightly more since each instruction takes a few cycles to execute.

3. Should you use timer delay or cycle delay?

**Solution:** Timer delay is an accurate method for giving delays. Cycle delay is a crude method and should be avoided.

## 7.5. Memory

This manual shows how to use the Atmega328p internal memory for a decade counter through a loop.

1. Execute the following code by connecting the Arduino to 7447 through pins 2,3,4,5.

The seven segment display should be connected to 7447.

```
assembly/memory/codes/mem.asm
```

2. Explain the following instructions

```
ldi xl,0x00
```

```
ldi xh,0x01  
ldi r16,0b00000000  
st x,r16
```

**Solution:** X=R27:R26, Y=R29:R28, and Z=R31:R30 where R27:R26 represents XH:XL.

The above instructions load 0b00000000 into the memory location X=0x0100.

3. What does the **loop\_cnt** routine do?

```
ldi r16,0b00000000  
ldi r17,0x09  
loop_cnt:  
inc r16  
inc xl  
st x,r16  
dec r17  
brne loop_cnt
```

**Solution:** The routine loads the numbers 1-9 in memory locations 0x0101 - 0x0109.

4. Revise your code by using a timer for giving the delay.



# Chapter 8

## Embedded C

### 8.1. Blink

This manual shows how to control an led using AVR-GCC. AVR-GCC is a C compiler for the Atmega328p.

#### 8.1.1. Components

| Component | Value | Quantity |
|-----------|-------|----------|
| Arduino   | UNO   | 1        |

#### 8.1.2. Blink

1. Execute the following

```
cd avr-gcc/setup/codes
```

```
make
```

2. Now open **main.c**. Explain the following lines.

```
PORTB = ((0 << PB5));
```

```
_delay_ms(500);  
//turn led on  
PORTB = ((1 << PB5));  
_delay_ms(500);
```

**Solution:**  $((0 << PB5))$  writes 0 to pin 13 (PB5). `_delay_ms(500)` introduces a delay of 500 ms.

3. Modify the above code to keep the led on.
4. Repeat the above exercise to keep the led off.

This manual shows how to control a seven segment display using AVR-GCC with arduino

## 8.2. Display Control

1. Connect the arduino to the seven segment display
2. Execute the following code

```
avr-gcc/sevenseg/codes/main.c
```

3. Modify the above code to generate numbers between 0-9.

This manual shows how to control a seven segment display using AVR-GCC with arduino

## 8.3. Input

1. Connect the arduino to the seven segment display through 7447.

2. Execute the following code

```
avr-gcc/input/codes/main.c
```

3. Modify the above code to work without the 7447.

## 8.4. GCC-Assembly

This manual shows how to write a function in assembly and call it in a C program while programming the ATMega328P microcontroller in the Arduino. This is done by controlling an LED.

### 8.4.1. Components

| Component             | Value            | Quantity |
|-----------------------|------------------|----------|
| Breadboard            |                  | 1        |
| Resistor              | $\geq 220\Omega$ | 1        |
| Arduino               | Uno              | 1        |
| Seven Segment Display | Common Anode     | 1        |
| Jumper Wires          |                  | 10       |

Table 3:

### 8.4.2. GCC with Assembly

1. Execute

```
cd avr-gcc/gcc-assembly/codes  
make
```

2. Modify **main.c** and **Makefile** to turn the builtin led on.
3. Repeat the above exercise to turn the LED off.
4. Explain how the **disp\_led(0)** function is related to **Register R24** in **disp\_led** routine in **displedasm.S**. **Solution:** The function argument 0 in **disp\_led(0)** is passed on to R24 in the assembly routine for further operations. Also, the registers R18-R24 are available for storing more function arguments according to the Table 4. More details are available in official ATMEL AT1886 reference.

| Register          | r19 | r18 | r21 | r20 | r23 | r22 | r25 | r24 |
|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Function Argument | b7  | b6  | b5  | b4  | b3  | b2  | b1  | b0  |

Table 4: Relationship between Register in assembly and function argument in C

5. Write an assembly routine for controlling the seven segment display and call it in a C program.
6. Build a decade counter with **main.c** calling all functions from assembly routines.

## 8.5. LCD

This manual shows how to interface an Arduino to a  $16 \times 2$  LCD display using AVR-GCC.

This framework provides a useful platform for displaying the output of AVR-Assembly programs.

| Component    | Value         | Quantity |
|--------------|---------------|----------|
| Breadboard   |               | 1        |
| Arduino      | Uno           | 1        |
| LCD          | $16 \times 2$ | 1        |
| Jumper Wires |               | 20       |

Table 6:

### 8.5.1. Components

### 8.5.2. Display Number on LCD

1. Plug the LCD in Fig. 2 to the breadboard.
2. Connect the Arduino pins to LCD pins as per Table 2.

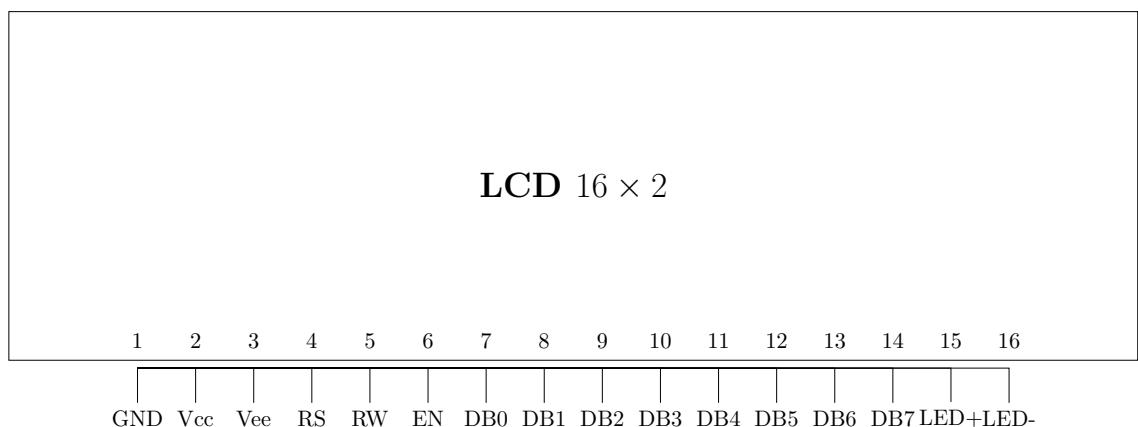


Figure 2: LCD

3. Execute

```
cd avr-gcc/lcd/codes
make
```

Table 2: Arduino to LCD Pin Connection.

| Arduino Pins | LCD Pins | LCD Pin Label | LCD Pin Description |
|--------------|----------|---------------|---------------------|
| GND          | 1        | GND           |                     |
| 5V           | 2        | Vcc           |                     |
| GND          | 3        | Vee           | Contrast            |
| D8           | 4        | RS            | Register Select     |
| GND          | 5        | R/W           | Read/Write          |
| D9           | 6        | EN            | Enable              |
| D10          | 11       | DB4           | Serial Connection   |
| D11          | 12       | DB5           | Serial Connection   |
| D12          | 13       | DB6           | Serial Connection   |
| D13          | 14       | DB7           | Serial Connection   |
| 5V           | 15       | LED+          | Backlight           |
| GND          | 16       | LED-          | Backlight           |

4. Modify the above code to display a string.
5. Modify the above code to obtain a decade counter so that the numbers from 0 to 9 are displayed on the lcd repeatedly.
6. Repeat the above exercises to display a string on the first line and a number on the second line of the lcd.
7. Write assembly routines for driving the lcd.

# Chapter 9

## Vaman-FPGA

### 9.1. Setup

This document provides a simple introduction to software and hardware using the Vaman FPGA/microcontroller board. The exercises provided here are suitable for students from primary school till college.

#### 9.1.1. Software

The codes are available at

```
vaman/fpga/setup/codes/
```

#### 9.1.2. Setup

9.1.2.1. Follow the instructions in

```
vaman/installation/termuxdebian/termux_debian_fpga.txt
```

### 9.1.3. Frequency

9.1.3.1. In the following verilog program,

```
codes/blink/helloworldfpga.v
```

pay attention to the following lines

```
delay = delay+1;  
if(delay > 20000000)  
begin  
delay=27'b0;  
led=!led;  
end
```

It may be deduced from the above that the blink frequency is 20 MHz.

9.1.3.2. In instruction 9.1.3.1, replace

```
if(delay > 20000000)
```

with

```
if(delay==27'b100110001001011010000000)
```

and execute the verilog code.

9.1.3.3. Since the delay is 20 MHz, the blink period is 1 second. Modify the verilog code so that the blink period becomes 0.5s.

9.1.3.4. Find the bit length of 20 MHz.

| Type   | Vaman Pin | Connection |
|--------|-----------|------------|
| Input  | IO_28     | GND        |
| Output | IO_11     | LED        |

Table 9.1.3.8.1: Vaman Input/Output.

**Solution:**

$$\log_2(20000000) \approx 27 \quad (9.1.3.4.1)$$

9.1.3.5. Obtain the above answer using a Python code.

**Solution:** Execute the following code and compare with instruction 9.1.3.2.

```
codes/blink/freq_count.py
```

9.1.3.6. Replace the following line in the code in instruction 9.1.3.1

```
assign redled = led; //If you want to change led colour to red,
```

with

```
assign blueled = led;
```

and execute the code.

9.1.3.7. Ensure that the LED stays on in green colour.

**Solution:** Execute the following code

```
vaman/setup/codes/blink/onoff.v
```

9.1.3.8. Using Table 9.1.3.8.1 and Fig. 9.1.3.8.1, control the onboard LED through an external input. Connect an external LED and control it using an output pin as well.

## PYGMY BB v1 PINOUT

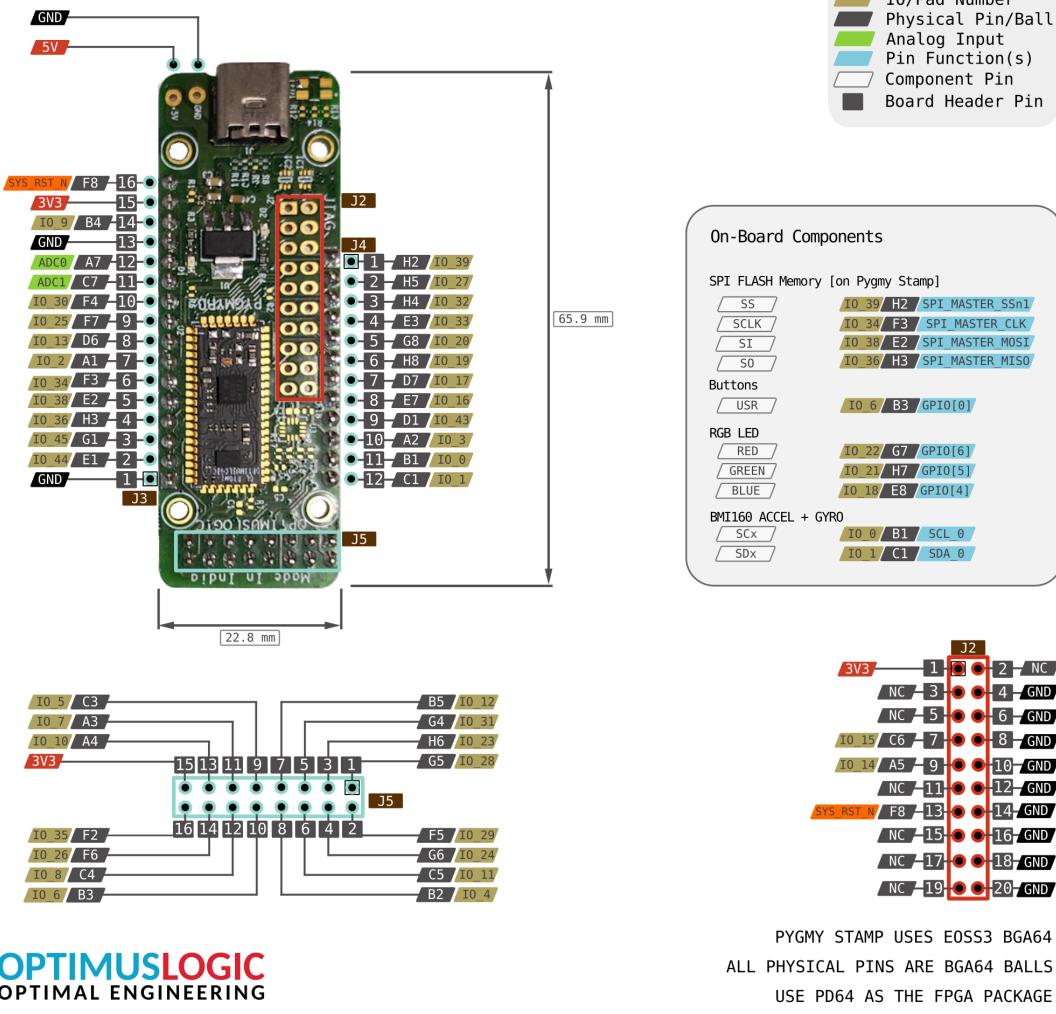


Figure 9.1.3.8.1: Pin diagram

**Solution:** Execute the following code and take out the input pin connect to GND.

Plug it again. Do this repeatedly.

vaman/setup/codes/input/blink\_ip.v

vaman/setup/codes/input/pygmy.pcf

## 9.2. Seven Segment Display

We show how to use Vaman as a decade counter.

### 9.2.1. Software

All codes used in this manual are available at the following link.

vaman/fpga/sevenseg/codes

### 9.2.2. Setup

9.2.2.1. The pin sheet for the Vaman is available in Fig. 9.2.2.1.3. Connect the pins in the bank J5 of the Vaman with the seven segment display shown in Fig. 9.2.2.1.1 according to Table 9.2.2.1.1. Ensure that the COM pin is connected to 3.3V through a resistor.

9.2.2.2. Now execute the following code.

codes/static/sevenseg.v

Flash the helloworldfpga.bin file to Vaman. You should see the number 5 displayed.

The following lines are used for generating numbers on the display.

```
assign a=0;  
assign b=1;  
assign c=0;  
assign d=0;  
assign e=1;  
assign f=0;
```

| Display | Pygmy |
|---------|-------|
| a       | IO_4  |
| b       | IO_5  |
| c       | IO_6  |
| d       | IO_7  |
| e       | IO_8  |
| f       | IO_10 |
| g       | IO_11 |
| COM     | 3.3 V |

Table 9.2.2.1.1: Seven segment display - Vaman connection.

|             |
|-------------|
| assign g=0; |
|-------------|

9.2.2.3. Modify the above code appropriately with the help of Table 9.2.2.3.1 and Fig. 9.2.2.1.2 to generate the numbers from 0-9 on the display.

| a | b | c | d | e | f | g | decimal |
|---|---|---|---|---|---|---|---------|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1       |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2       |

Table 9.2.2.3.1: Pin values used for generating decimal numbers on the seven segment display.

### 9.2.3. Examples

9.2.3.1. Table 9.2.2.1.1 and the PU 64 table in Fig. 9.2.3.2.1 explain the pin numbering in the following file.

```
codes/static/Vaman.pcf
```

9.2.3.2. Execute the code below. All the pins in the display are controlled using a 7 bit word.

```
codes/static/sevenseg_word.v
```

The above file is used for generating the number 4 on the display. The process is explained by the completion of Table 9.2.2.3.1.

```
gpio_out=7'b0100100;
```

9.2.3.3. Use a verilog function that takes a decimal input and display it on the seven segment display.

**Solution:** Execute the following code.

```
codes/static/sevenseg_dec.v
```

9.2.3.4. Use the Vaman as a decade counter.

**Solution:** Execute the following code.

```
codes/loop/decade_counter.v
```

## 9.3. Boolean Logic

In this document we show how to design a decade counter using Vaman and boolean logic.

### 9.3.1. Software

All codes used in this manual are available at the following link.

vaman/fpga/boolean/codes

### 9.3.2. Setup

9.3.2.1. Fig. 9.3.2.1.2 shows the pin diagram for the Vaman. Using the bank J5, connect the pins of the seven segment display in Fig. 9.3.2.1.1 to the Vaman according to Table 9.3.2.1.1. Make sure that the COM pin is connected to 3.3V through a resistor.

| Display | Pygmy |
|---------|-------|
| a       | IO_4  |
| b       | IO_5  |
| c       | IO_6  |
| d       | IO_7  |
| e       | IO_8  |
| f       | IO_10 |
| g       | IO_11 |
| COM     | 3.3 V |

| Input Variable | Pin   |
|----------------|-------|
| W              | IO_28 |
| X              | IO_23 |
| Y              | IO_31 |
| Z              | IO_12 |

Table 9.3.2.1.1: Pin connections between Vaman and the display.

9.3.2.2. Implement Table 9.3.2.2.1 using the Vaman and the display.

**Solution:** In Table 9.3.2.2.1, the output variables  $a, b, c, d, e, f, g$  can be expressed in

| Z | Y | X | W | a | b | c | d | e | f | g | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0       |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1       |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2       |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3       |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 4       |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5       |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6       |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 7       |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8       |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 9       |

Table 9.3.2.2.1: Truth table for the display decoder.

terms of the input variables  $W, X, Y, Z$  as

$$a = WX'Y'Z' + W'X'YZ' \quad (9.3.2.2.1)$$

$$b = WX'YZ' + W'XYZ' \quad (9.3.2.2.2)$$

$$c = Z'Y'XW' \quad (9.3.2.2.3)$$

$$d = WX'Y'Z' + W'X'YZ' + WXYZ' + WX'YZ \quad (9.3.2.2.4)$$

$$e = WX'Y'Z' + WXY'Z' + W'X'YZ' + WX'YZ' + WXYZ' + WX'Y'Z \quad (9.3.2.2.5)$$

$$f = WX'Y'Z' + W'XY'Z' + WXY'Z' + WXYZ' \quad (9.3.2.2.6)$$

$$g = W'X'Y'Z' + WX'Y'Z' + WXYZ' \quad (9.3.2.2.7)$$

Execute the following program.

```
vaman/fpga/boolean/codes/decoders/dispdec.v
```

```
vaman/fpga/boolean/codes/decoders/Vaman.pcf
```

| Z | Y | X | W | D | C | B | A |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 9.3.2.3.1: Truth table for the incrementing decoder.

Connect  $W, X, Y, Z$  to GND. For different values of the input variables, verify the output in on the display using Table 9.3.2.2.1.

9.3.2.3. Table 9.3.2.3.1 describes the properties of the incrementing decoder. Using Boolean logic, express  $A, B, C, D$  in terms of  $W, X, Y, Z$ . Subsequently, implement this decoder by implementing the the expressions so obtained in the Vaman using verilog.

**Solution:** The following equations contain the desired expressions.

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'X'Y'Z \quad (9.3.2.3.1)$$

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \quad (9.3.2.3.2)$$

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \quad (9.3.2.3.3)$$

$$D = WXYZ' + W'X'Y'Z \quad (9.3.2.3.4)$$

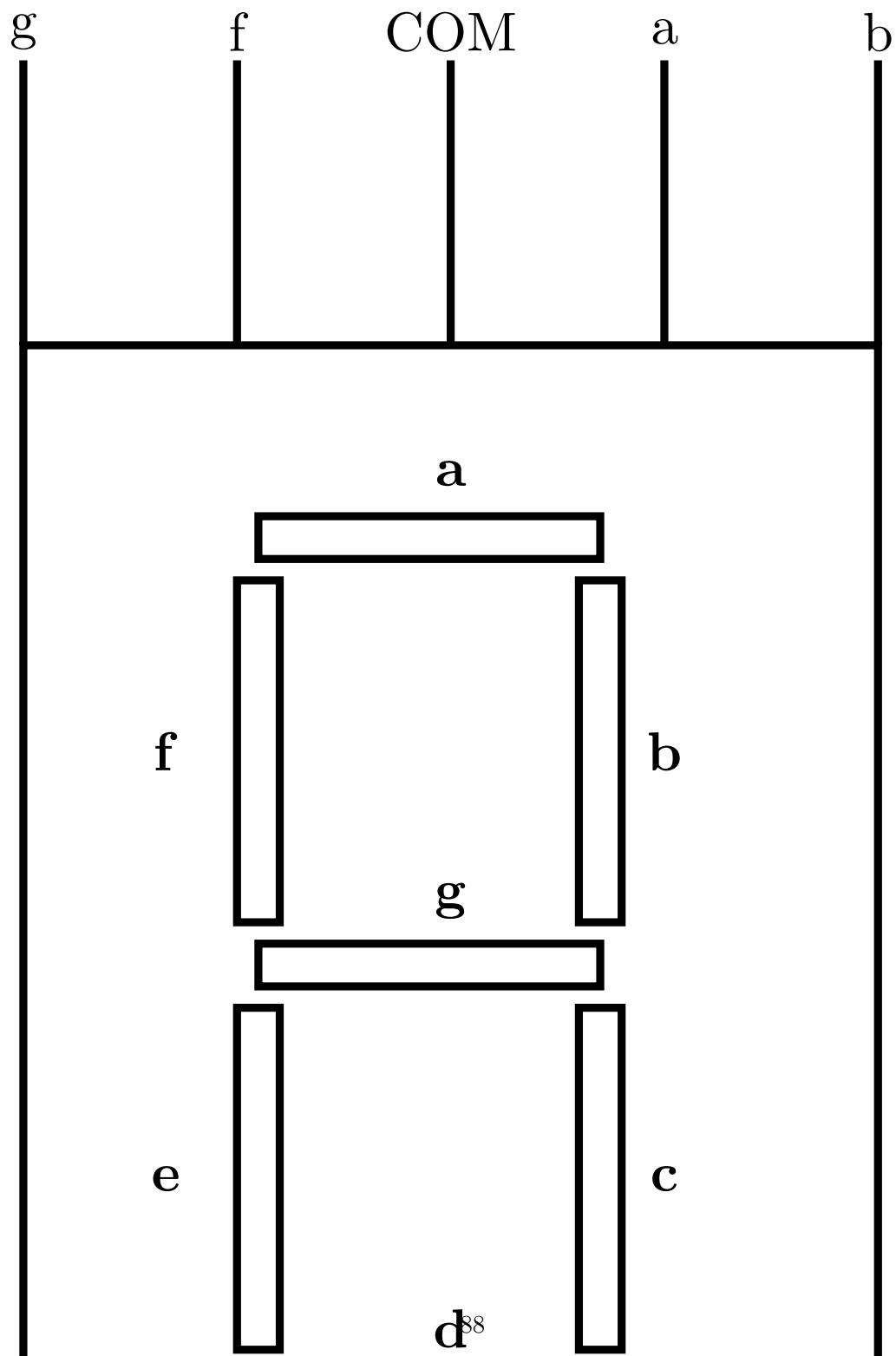
Execute the following code. The next number should be displayed.

```
vaman/fpga/boolean/codes/decoders/incdec.v
```

### **9.3.3. Decade Counter**

9.3.3.1. Using Fig. 9.3.3.1.1 and modifying the code in Problem 9.3.2.3, design the decade counter.

9.3.3.2. Design and implement the down counter.



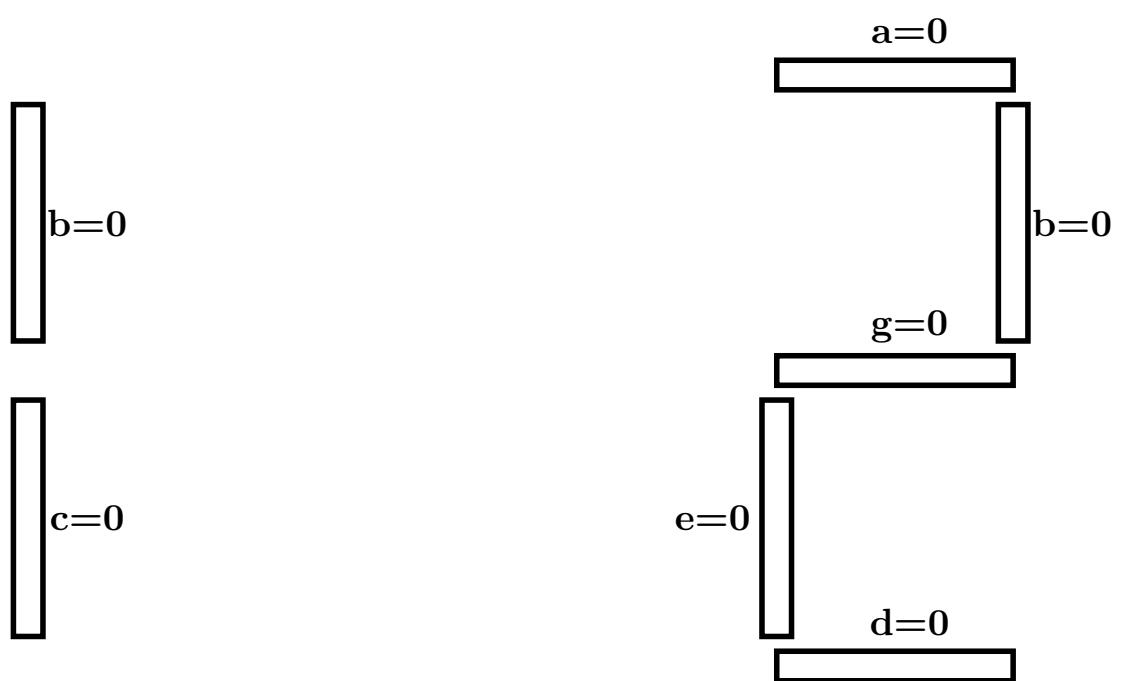


Figure 9.2.2.1.2: Companion figure for Table 9.2.2.3.1.

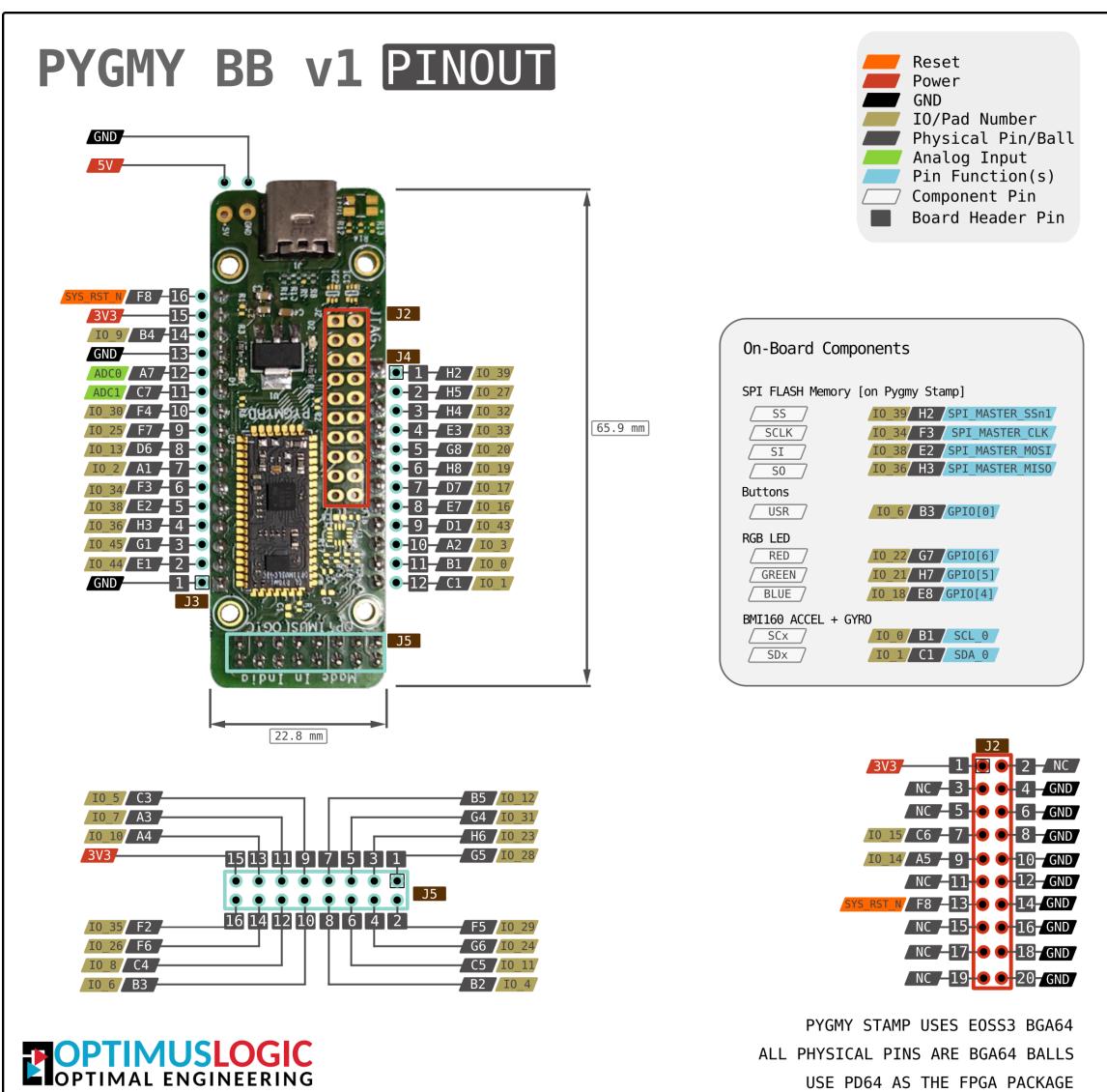
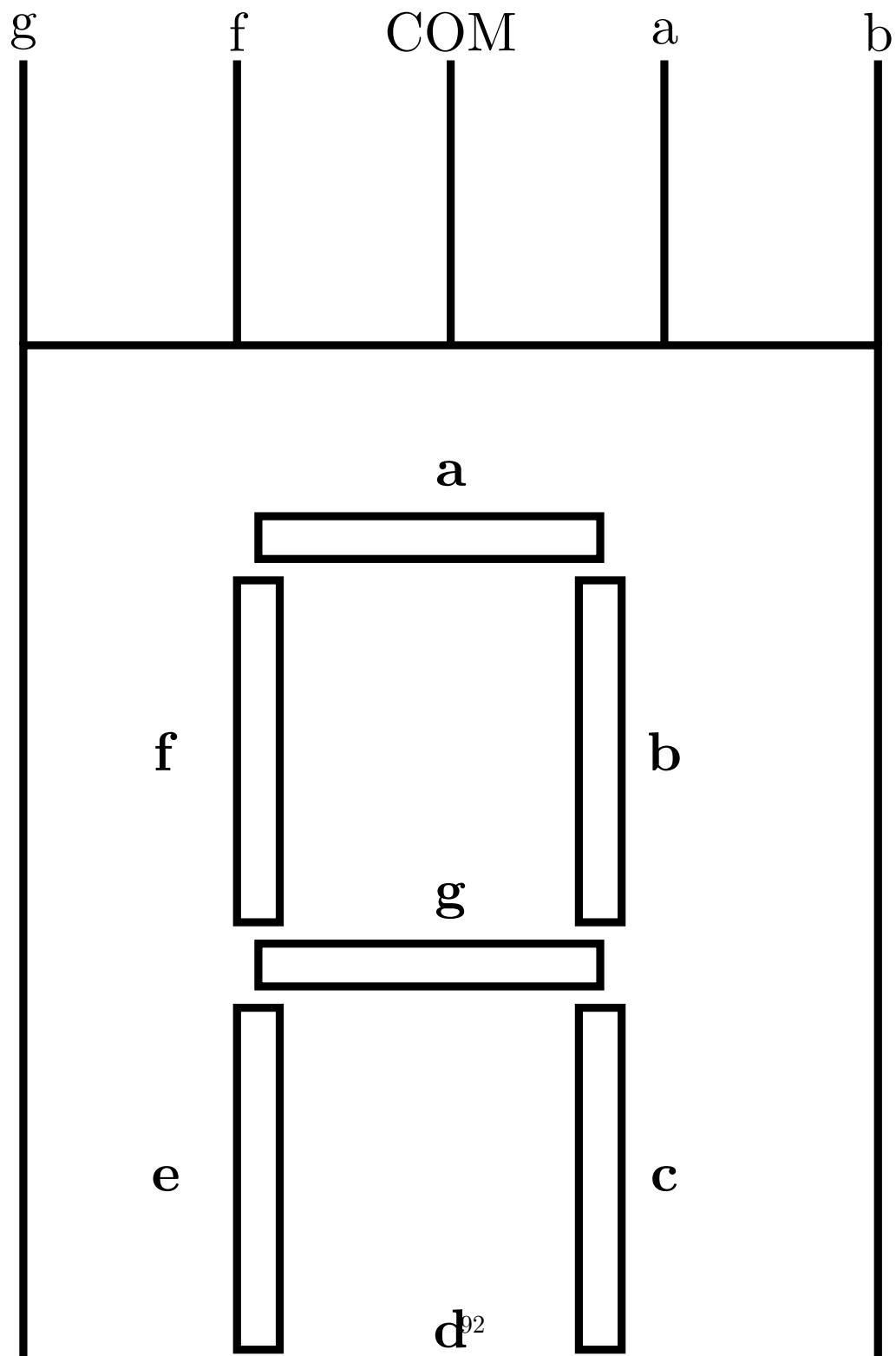


Figure 9.2.2.1.3: Pin Diagram

| PD64        |       |             | PU64        |       |             | WR42        |       |             |
|-------------|-------|-------------|-------------|-------|-------------|-------------|-------|-------------|
| IO Location | Alias | IO Type     | IO Location | Alias | IO type     | IO Location | Alias | IO Type     |
| B1          | IO_0  | BIDIR       | 4           | IO_0  | BIDIR       | A7          | IO_0  | BIDIR       |
| C1          | IO_1  | BIDIR       | 5           | IO_1  | BIDIR       | B7          | IO_1  | BIDIR       |
| A1          | IO_2  | BIDIR       | 6           | IO_2  | BIDIR       | C7          | IO_3  | BIDIR       |
| A2          | IO_3  | BIDIR       | 2           | IO_3  | BIDIR       | A6          | IO_6  | BIDIR       |
| B2          | IO_4  | BIDIR       | 3           | IO_4  | BIDIR       | B6          | IO_8  | BIDIR/CLOCK |
| C3          | IO_5  | BIDIR       | 64          | IO_5  | BIDIR       | A5          | IO_9  | BIDIR       |
| B3          | IO_6  | BIDIR       | 62          | IO_6  | BIDIR       | B5          | IO_10 | BIDIR       |
| A3          | IO_7  | BIDIR/CLOCK | 63          | IO_7  | BIDIR/CLOCK | A4          | IO_14 | BIDIR       |
| C4          | IO_8  | BIDIR/CLOCK | 61          | IO_8  | BIDIR/CLOCK | B4          | IO_15 | BIDIR       |
| B4          | IO_9  | BIDIR       | 60          | IO_9  | BIDIR       | E1          | IO_16 | BIDIR       |
| A4          | IO_10 | BIDIR       | 59          | IO_10 | BIDIR       | D1          | IO_17 | BIDIR       |
| C5          | IO_11 | BIDIR       | 57          | IO_11 | BIDIR       | C1          | IO_19 | BIDIR       |
| B5          | IO_12 | BIDIR       | 56          | IO_12 | BIDIR       | F2          | IO_20 | BIDIR       |
| D6          | IO_13 | BIDIR       | 55          | IO_13 | BIDIR       | E2          | IO_23 | BIDIR/CLOCK |
| A5          | IO_14 | BIDIR       | 54          | IO_14 | BIDIR       | D2          | IO_24 | BIDIR/CLOCK |
| C6          | IO_15 | BIDIR       | 53          | IO_15 | BIDIR       | D3          | IO_25 | BIDIR       |
| E7          | IO_16 | BIDIR       | 40          | IO_16 | BIDIR       | F3          | IO_28 | BIDIR       |
| D7          | IO_17 | BIDIR       | 42          | IO_17 | BIDIR       | E3          | IO_29 | BIDIR       |
| E8          | IO_18 | BIDIR       | 38          | IO_18 | BIDIR       | F4          | IO_30 | BIDIR       |
| H8          | IO_19 | BIDIR       | 36          | IO_19 | BIDIR       | E4          | IO_31 | BIDIR       |
| G8          | IO_20 | BIDIR       | 37          | IO_20 | BIDIR       | D5          | IO_34 | SDIOMUX     |
| H7          | IO_21 | BIDIR       | 39          | IO_21 | BIDIR       | F5          | IO_36 | SDIOMUX     |
| G7          | IO_22 | BIDIR/CLOCK | 34          | IO_22 | BIDIR/CLOCK | E6          | IO_38 | SDIOMUX     |
| H6          | IO_23 | BIDIR/CLOCK | 33          | IO_23 | BIDIR/CLOCK | F6          | IO_39 | SDIOMUX     |
| G6          | IO_24 | BIDIR/CLOCK | 32          | IO_24 | BIDIR/CLOCK | D7          | IO_43 | SDIOMUX     |
| F7          | IO_25 | BIDIR       | 31          | IO_25 | BIDIR       | E7          | IO_44 | SDIOMUX     |
| F6          | IO_26 | BIDIR       | 30          | IO_26 | BIDIR       | F7          | IO_45 | SDIOMUX     |
| H5          | IO_27 | BIDIR       | 28          | IO_27 | BIDIR       |             |       |             |
| G5          | IO_28 | BIDIR       | 27          | IO_28 | BIDIR       |             |       |             |
| F5          | IO_29 | BIDIR       | 26          | IO_29 | BIDIR       |             |       |             |
| F4          | IO_30 | BIDIR       | 25          | IO_30 | BIDIR       |             |       |             |
| G4          | IO_31 | BIDIR       | 23          | IO_31 | BIDIR       |             |       |             |
| H4          | IO_32 | SDIOMUX     | 22          | IO_32 | SDIOMUX     |             |       |             |
| E3          | IO_33 | SDIOMUX     | 21          | IO_33 | SDIOMUX     |             |       |             |
| F3          | IO_34 | SDIOMUX     | 20          | IO_34 | SDIOMUX     |             |       |             |
| F2          | IO_35 | SDIOMUX     | 18          | IO_35 | SDIOMUX     |             |       |             |
| H3          | IO_36 | SDIOMUX     | 17          | IO_36 | SDIOMUX     |             |       |             |
| G2          | IO_37 | SDIOMUX     | 15          | IO_37 | SDIOMUX     |             |       |             |
| E2          | IO_38 | SDIOMUX     | 16          | IO_38 | SDIOMUX     |             |       |             |
| H2          | IO_39 | SDIOMUX     | 11          | IO_39 | SDIOMUX     |             |       |             |
| D2          | IO_40 | SDIOMUX     | 13          | IO_40 | SDIOMUX     |             |       |             |
| F1          | IO_41 | SDIOMUX     | 14          | IO_41 | SDIOMUX     |             |       |             |
| H1          | IO_42 | SDIOMUX     | 10          | IO_42 | SDIOMUX     |             |       |             |
| D1          | IO_43 | SDIOMUX     | 7           | IO_43 | SDIOMUX     |             |       |             |
| E1          | IO_44 | SDIOMUX     | 8           | IO_44 | SDIOMUX     |             |       |             |
| G1          | IO_45 | SDIOMUX     | 9           | IO_45 | SDIOMUX     |             |       |             |

Figure 9.2.3.2.1: Pin Definitions



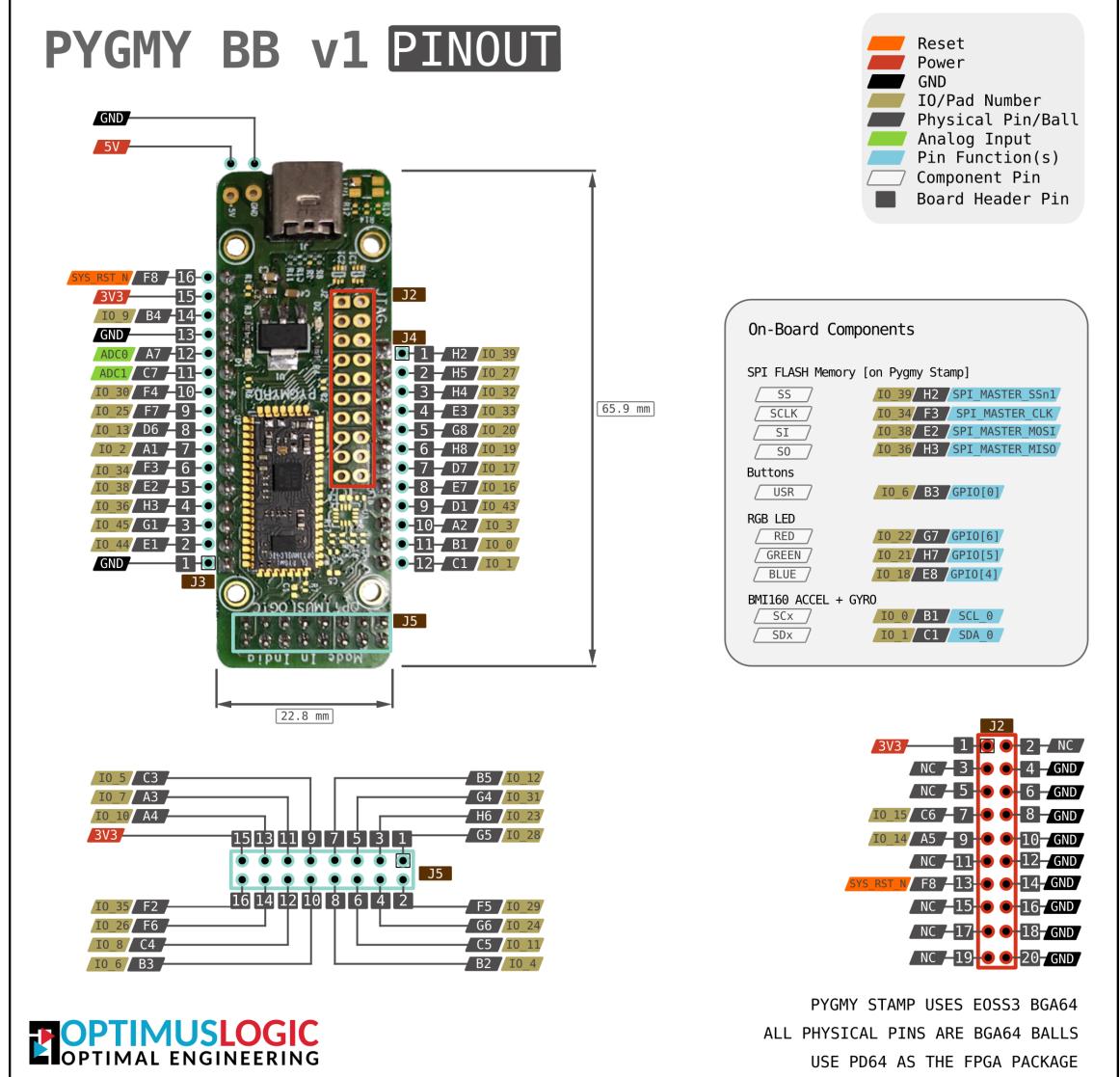


Figure 9.3.2.1.2: Pin diagram

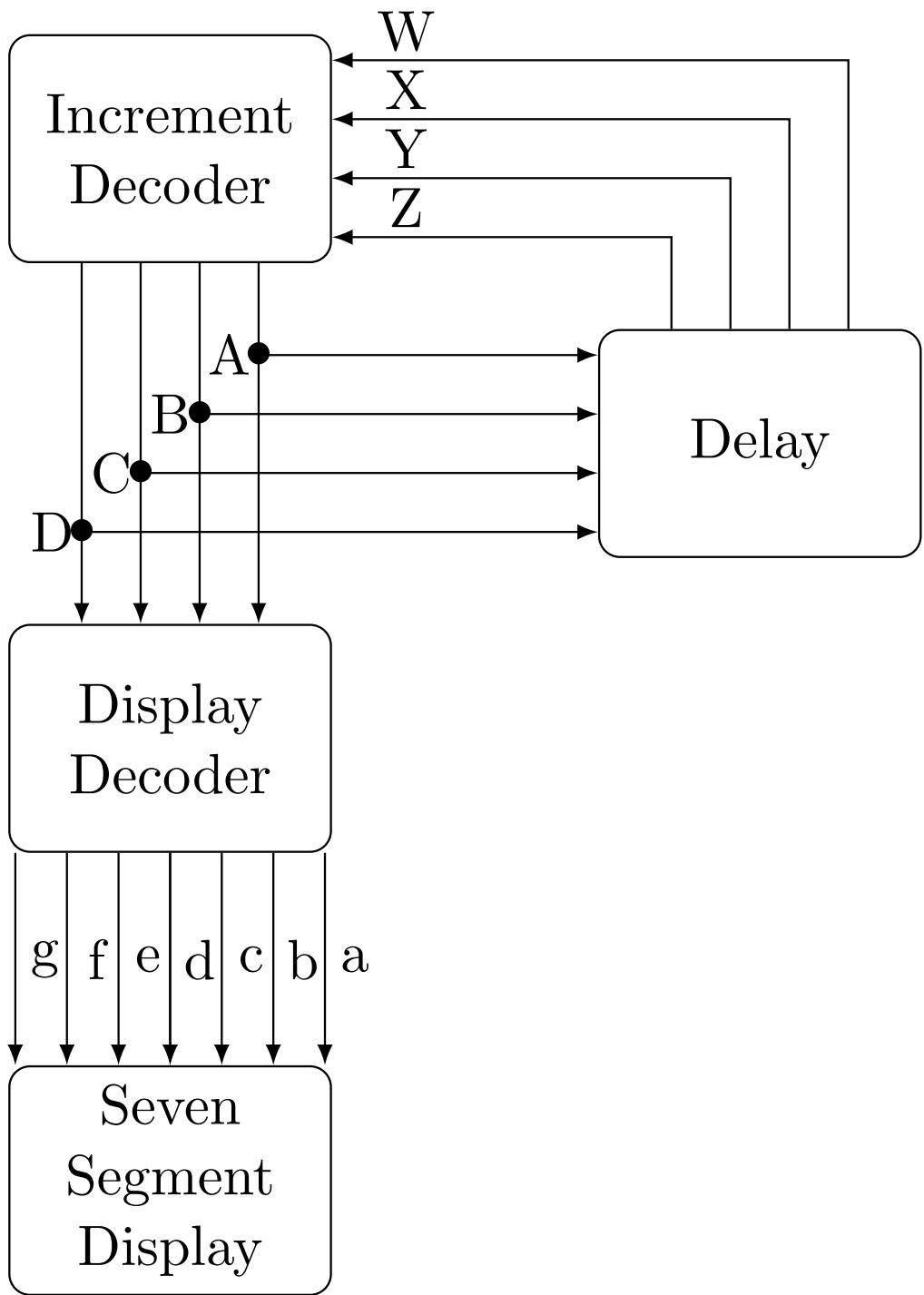


Figure 9.3.3.1.1: Block diagram of the decade counter.

# Chapter 10

## Vaman-ARM

### 10.1. Setup

#### 10.1.1. Software

All codes used in this document are available at

```
vaman/arm/codes/setup
```

#### 10.1.2. Setup

10.1.2.1. Follow the instructions at

```
vaman/installation/termuxdebian/termuxdebian_arm.txt
```

#### 10.1.3. Delay

10.1.3.1. See the following lines of the code below

```
codes/setup/blink/src/main.c
```

```
PyHal_Set_GPIO(18,1); //blue  
PyHal_Set_GPIO(21,1); //green  
PyHal_Set_GPIO(22,1); //red  
HAL_DelayUSec(2000000);  
PyHal_Set_GPIO(18,0);  
PyHal_Set_GPIO(21,0);  
PyHal_Set_GPIO(22,0);
```

We may conclude that the blink delay is 2000 000us = 2 s.

#### 10.1.3.2. Replace the following line in 10.1.3.1

```
HAL_DelayUSec(2000000);
```

with

```
HAL_DelayUSec(1000000);
```

and execute. Can you see any difference in the blink period?

#### 10.1.3.3. To obtain red colour, execute the following code.

```
vaman/arm/codes/setup/red/src/main.c
```

Now obtain blue colour.

#### 10.1.3.4. Now obtain green colour without blink.

**Solution:** Execute the following code.

```
vaman/arm/codes/setup/onoff/src/main.c
```

| Type  | Pin  | Destination |
|-------|------|-------------|
| Input | IO_5 | GND         |

Table 10.1.3.5.1: Vaman control through external input.

10.1.3.5. Using Table 10.1.3.5.1 and Fig. 10.1.3.5.1, use an input pin to control the onboard LED.

**Solution:** Execute the following code. You should see the LED blinking pink. Disconnecting the wire from GND will result in the LED blinking white and green alternately.

```
vaman/arm/codes/setup/gpio/src/main.c
```

## 10.2. Seven Segment Display

This document shows how to implement a decade counter using arm-gcc on Vaman.

### 10.2.1. Software

All codes used in this document are available at the following link

```
https://github.com/gadepall/vaman/tree/master/arm/vaman/arm/codes/sevenseg/
```

### 10.2.2. Setup

10.2.2.1. Fig.10.2.2.1.3 shows all the pin banks of the Vaman. Connect the pins of the display in Fig. 10.2.2.1.1 to bank J5 of the Vaman using Table 10.2.2.1.1. The COM pin should be connected to 3.3V through a resistor.

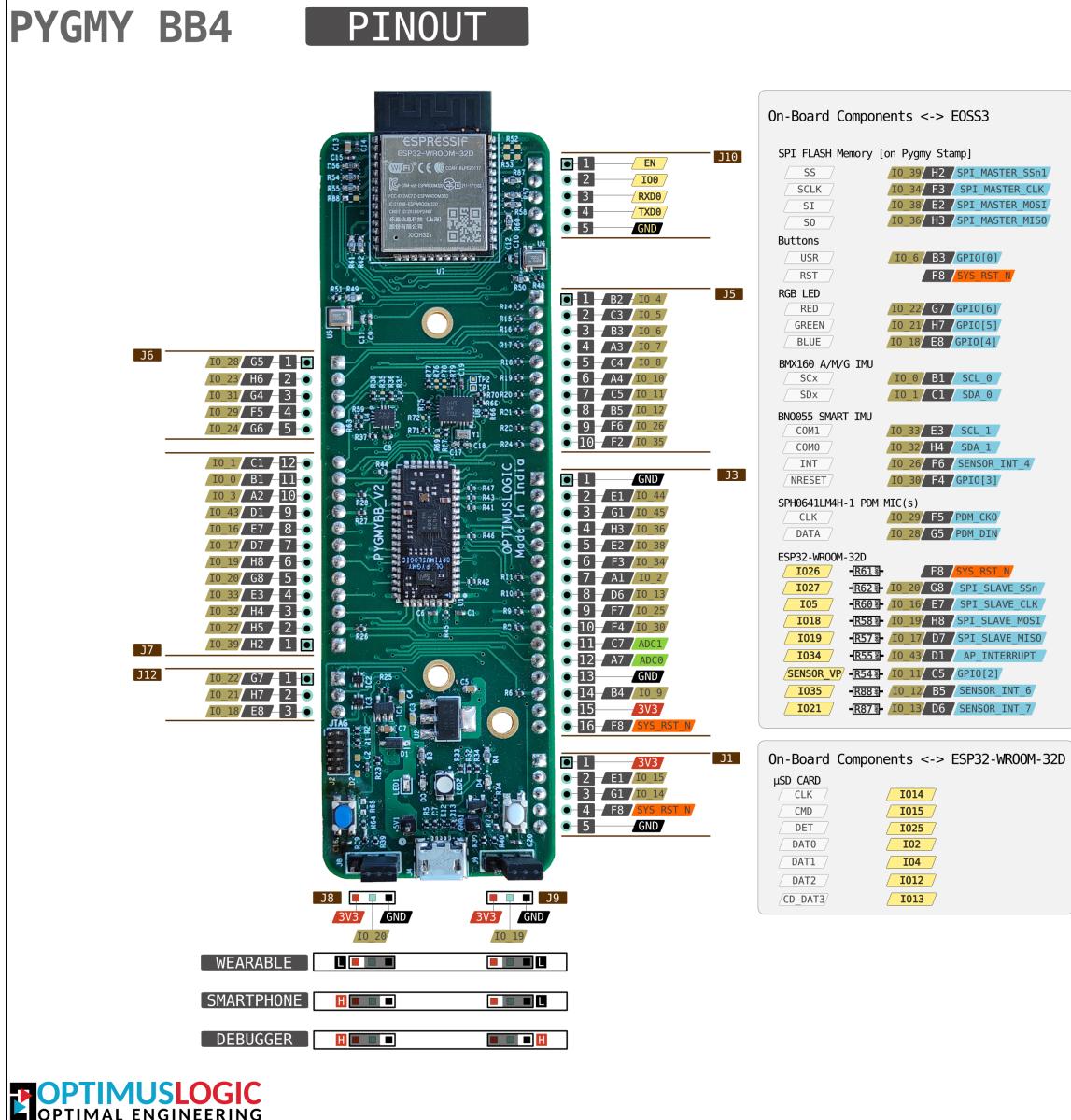


Figure 10.1.3.5.1: Pin Diagram

10.2.2.2. Now execute the following code

```
vaman/arm/codes/sevenseg/static/src/main.c
```

| Display | Vaman |
|---------|-------|
| a       | IO_4  |
| b       | IO_5  |
| c       | IO_6  |
| d       | IO_7  |
| e       | IO_8  |
| f       | IO_10 |
| g       | IO_11 |
| COM     | 3.3 V |

Table 10.2.2.1.1: Display-Vaman connection.

Flash static.bin obtained upon execution of the above code to the Vaman. You should see the number 7 on the display. The following function generates this number.

```
sevenseg(0,0,0,1,1,1,1);
void sevenseg(int a, int b, int c, int d, int e, int f, int g)
{
    //Seven Segment GPIO
    PyHal_GPIO_Set(4,a);//a
    PyHal_GPIO_Set(5,b);//b
    PyHal_GPIO_Set(6,c);//c
    PyHal_GPIO_Set(7,d);//d
    PyHal_GPIO_Set(8,e);//e
    PyHal_GPIO_Set(10,f);//f
    PyHal_GPIO_Set(11,g);//g
```

```
}
```

```
}
```

10.2.2.3. Modify the above program using Table 10.2.2.3.1 and Fig. 10.2.2.1.2 to display 0-9.

| a | b | c | d | e | f | g | decimal |
|---|---|---|---|---|---|---|---------|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1       |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2       |

Table 10.2.2.3.1: Decimal number generation on the display.

## 10.2.3. Examples

10.2.3.1. Table 10.2.2.1.1 and PU 64 Table in Fig. 10.2.3.1.1 show how to use the pins of the Vaman to drive the seven segment display.

10.2.3.2. Use a function taking decimal input in the code in 10.2.2.2 to generate numbers on the display.

**Solution:** Execute the following file.

```
vaman/arm/codes/sevenseg/decimal/main.c
```

10.2.3.3. Program the Vaman to function as a decade counter.

**Solution:** Execute the following code.

```
vaman/arm/codes/sevenseg/loop/main.c
```

## 10.3. FSM

This document shows how to use the Vaman to design a decade counter using a finite state machine (FSM).

### 10.3.1. Software

All codes in this document are available at the following links.

<https://github.com/gadepall/vaman/tree/master/arm/codes/decoders>

<https://github.com/gadepall/vaman/tree/master/arm/codes/fsm>

### 10.3.2. Setup

10.3.2.1. Execute Table 10.3.2.1.1 using the Vaman and a seven segment display.

**Solution:** The outputs  $a, b, c, d, e, f, g$  in Table 10.3.2.1.1 are expressed in terms of the inputs  $A, B, C, D$  through the following equations.

$$a = AB'C'D' + A'B'CD' \quad (10.3.2.1.1)$$

$$b = AB'CD' + A'BCD' \quad (10.3.2.1.2)$$

$$c = D'C'BA' \quad (10.3.2.1.3)$$

$$d = AB'C'D' + A'B'CD' + ABCD' + AB'C'D \quad (10.3.2.1.4)$$

$$e = AB'C'D' + ABC'D' + A'B'CD' + AB'CD' \\ + ABCD' + AB'C'D \quad (10.3.2.1.5)$$

$$f = AB'C'D' + A'BC'D' + ABC'D' + ABCD' \quad (10.3.2.1.6)$$

$$g = A'B'C'D' + AB'C'D' + ABCD' \quad (10.3.2.1.7)$$

| D | C | B | A | a | b | c | d | e | f | g | Decimal |
|---|---|---|---|---|---|---|---|---|---|---|---------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0       |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1       |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2       |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3       |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 4       |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5       |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6       |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 7       |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8       |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 9       |

Table 10.3.2.1.1: Truth table for the display decoder

Now execute the following code.

```
codes/decoders/dispdec/main.c
```

For different values of  $A, B, C, D$ , execute the above code to verify Table 10.3.2.1.1.

10.3.2.2. Table 10.3.2.2.1 shows the logic for the incrementing decoder. Express  $A, B, C, D$  in terms of  $W, X, Y, Z$ .

| Z | Y | X | W | D | C | B | A |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 10.3.2.2.1: Truth table for the incrementing decoder

**Solution:** The desired expressions are available below.

$$\begin{aligned}
 A = & W'X'Y'Z' + W'XY'Z' + W'X'YZ' \\
 & + W'XYZ' + W'X'Y'Z
 \end{aligned} \tag{10.3.2.2.1}$$

$$\begin{aligned}
 B = & WX'Y'Z' + W'XY'Z \\
 & + WX'YZ' + W'XYZ'
 \end{aligned} \tag{10.3.2.2.2}$$

$$\begin{aligned}
 C = & WXY'Z' + W'X'YZ' \\
 & + WX'YZ' + W'XYZ'
 \end{aligned} \tag{10.3.2.2.3}$$

$$D = WXYZ' + W'X'Y'Z \tag{10.3.2.2.4}$$

Execute the following code. You should see the next number displayed.

codes/decoders/incdec/main.c

10.3.2.3. Fig. 10.3.2.3.2 shows the pin diagram for the Vaman. Connect the pins in bank J5 to the seven segment display using Fig. 10.3.2.3.1 and Table 10.3.2.3.1. Do not forget to

put a resistor between COM and 3.3V. Then execute the following code

```
codes/fsm/dispdec/main.c
```

| Display | Pygmy |
|---------|-------|
| a       | IO_4  |
| b       | IO_5  |
| c       | IO_6  |
| d       | IO_7  |
| e       | IO_8  |
| f       | IO_10 |
| g       | IO_11 |
| COM     | 3.3 V |

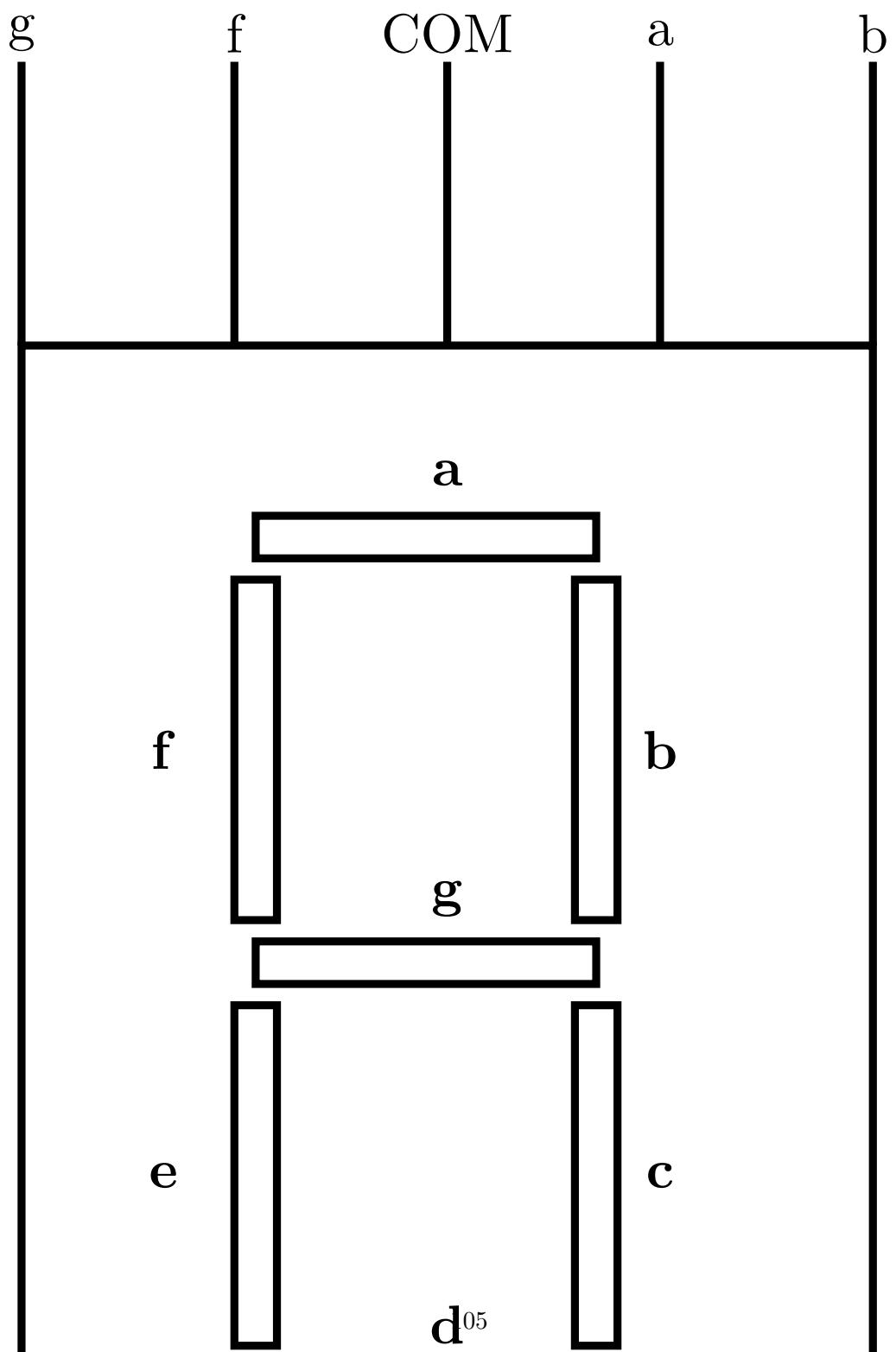
| Input Variable | Pin   |
|----------------|-------|
| W              | IO_28 |
| X              | IO_23 |
| Y              | IO_31 |
| Z              | IO_12 |

Table 10.3.2.3.1: Pin connection between the Vaman and seven segment display

10.3.2.4. Modify the above code to obtain a decade counter.

### 10.3.3. Decade Counter

10.3.3.1. Use the Vaman to implement all the decoders in Fig. 10.3.3.1.1. Implement the delay using a flip flop. This is an example of an FSM which is implemented using a sequential circuit.



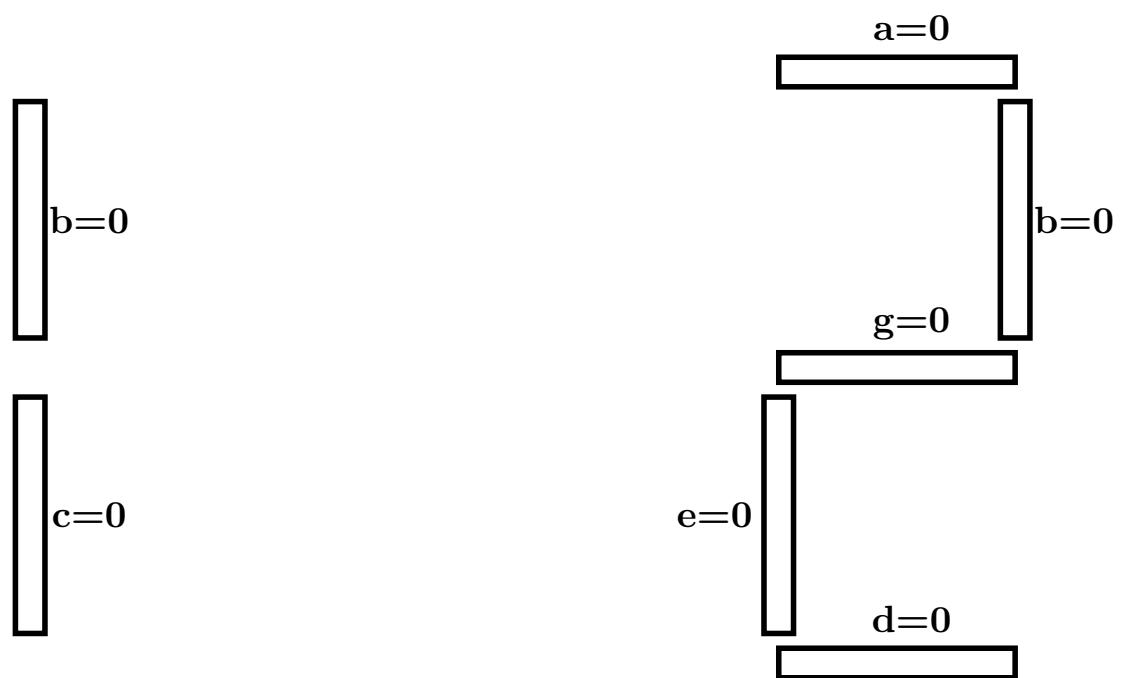


Figure 10.2.2.1.2: Pictorial representation of Table 10.2.2.3.1.

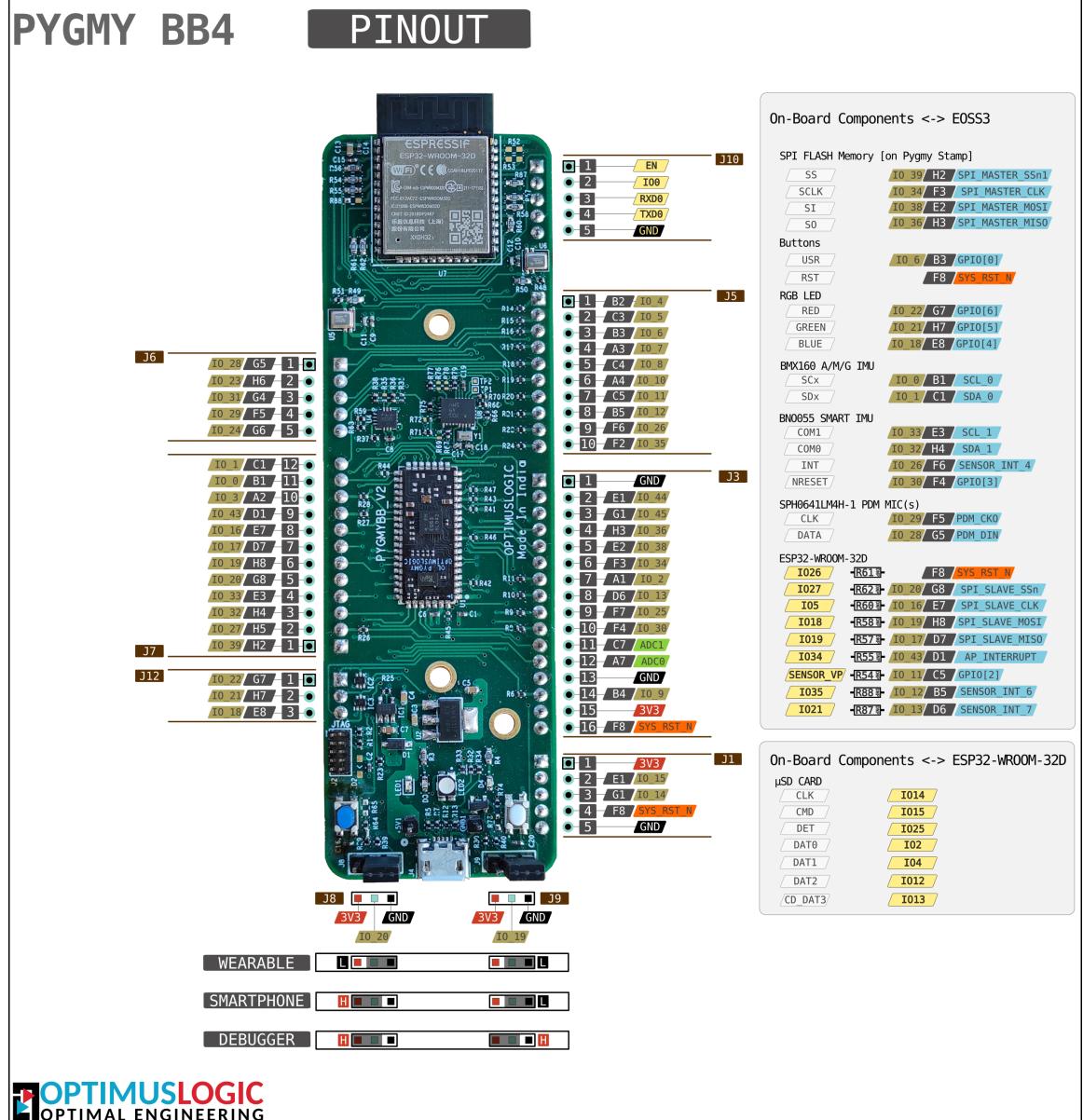
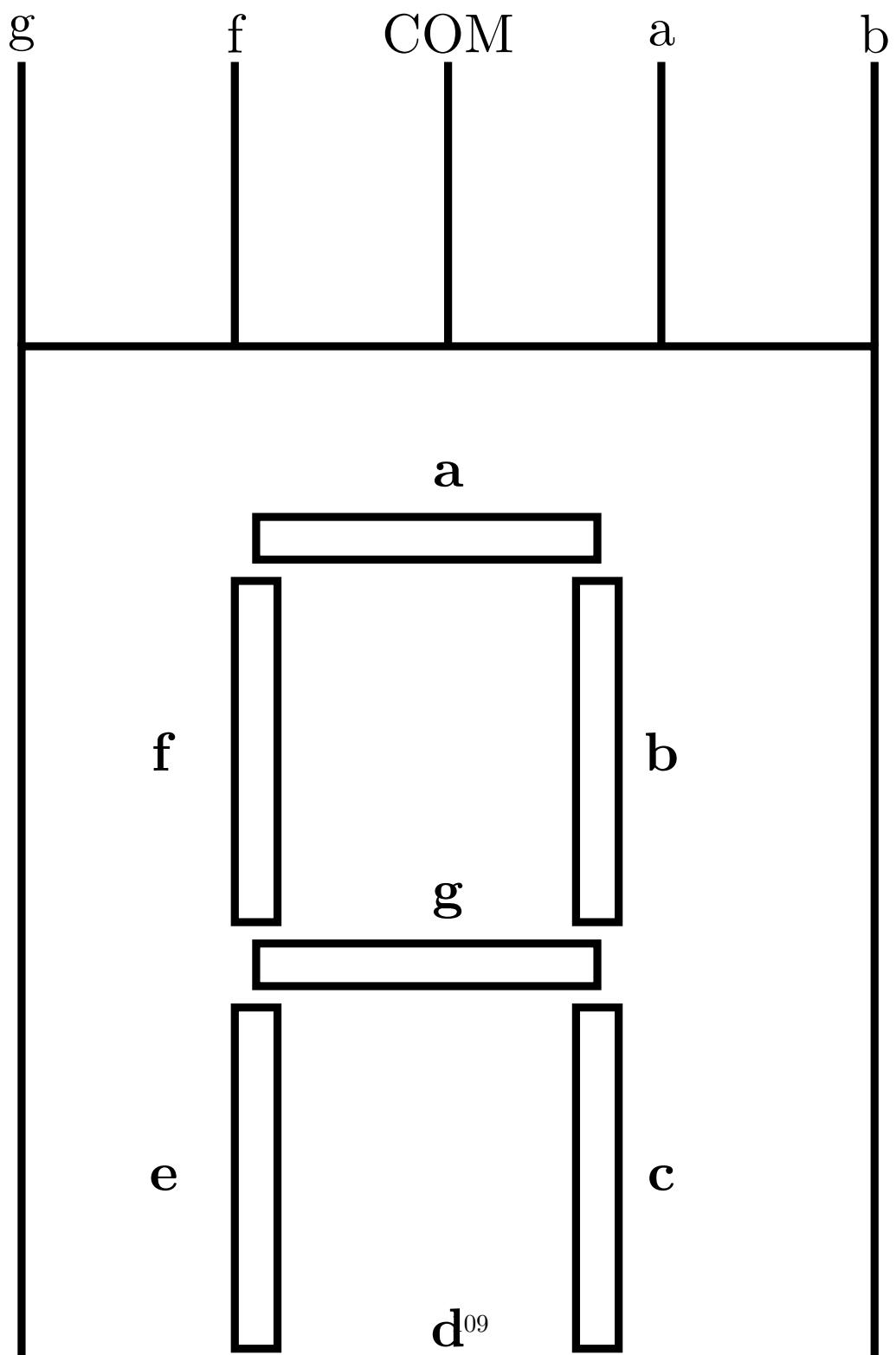


Figure 10.2.2.1.3: Pin Diagram

| PD64        |       |             | PU64        |       |             | WR42        |       |             |
|-------------|-------|-------------|-------------|-------|-------------|-------------|-------|-------------|
| IO Location | Alias | IO Type     | IO Location | Alias | IO type     | IO Location | Alias | IO Type     |
| B1          | IO_0  | BIDIR       | 4           | IO_0  | BIDIR       | A7          | IO_0  | BIDIR       |
| C1          | IO_1  | BIDIR       | 5           | IO_1  | BIDIR       | B7          | IO_1  | BIDIR       |
| A1          | IO_2  | BIDIR       | 6           | IO_2  | BIDIR       | C7          | IO_3  | BIDIR       |
| A2          | IO_3  | BIDIR       | 2           | IO_3  | BIDIR       | A6          | IO_6  | BIDIR       |
| B2          | IO_4  | BIDIR       | 3           | IO_4  | BIDIR       | B6          | IO_8  | BIDIR/CLOCK |
| C3          | IO_5  | BIDIR       | 64          | IO_5  | BIDIR       | A5          | IO_9  | BIDIR       |
| B3          | IO_6  | BIDIR       | 62          | IO_6  | BIDIR       | B5          | IO_10 | BIDIR       |
| A3          | IO_7  | BIDIR/CLOCK | 63          | IO_7  | BIDIR/CLOCK | A4          | IO_14 | BIDIR       |
| C4          | IO_8  | BIDIR/CLOCK | 61          | IO_8  | BIDIR/CLOCK | B4          | IO_15 | BIDIR       |
| B4          | IO_9  | BIDIR       | 60          | IO_9  | BIDIR       | E1          | IO_16 | BIDIR       |
| A4          | IO_10 | BIDIR       | 59          | IO_10 | BIDIR       | D1          | IO_17 | BIDIR       |
| C5          | IO_11 | BIDIR       | 57          | IO_11 | BIDIR       | C1          | IO_19 | BIDIR       |
| B5          | IO_12 | BIDIR       | 56          | IO_12 | BIDIR       | F2          | IO_20 | BIDIR       |
| D6          | IO_13 | BIDIR       | 55          | IO_13 | BIDIR       | E2          | IO_23 | BIDIR/CLOCK |
| A5          | IO_14 | BIDIR       | 54          | IO_14 | BIDIR       | D2          | IO_24 | BIDIR/CLOCK |
| C6          | IO_15 | BIDIR       | 53          | IO_15 | BIDIR       | D3          | IO_25 | BIDIR       |
| E7          | IO_16 | BIDIR       | 40          | IO_16 | BIDIR       | F3          | IO_28 | BIDIR       |
| D7          | IO_17 | BIDIR       | 42          | IO_17 | BIDIR       | E3          | IO_29 | BIDIR       |
| E8          | IO_18 | BIDIR       | 38          | IO_18 | BIDIR       | F4          | IO_30 | BIDIR       |
| H8          | IO_19 | BIDIR       | 36          | IO_19 | BIDIR       | E4          | IO_31 | BIDIR       |
| G8          | IO_20 | BIDIR       | 37          | IO_20 | BIDIR       | D5          | IO_34 | SDIOMUX     |
| H7          | IO_21 | BIDIR       | 39          | IO_21 | BIDIR       | F5          | IO_36 | SDIOMUX     |
| G7          | IO_22 | BIDIR/CLOCK | 34          | IO_22 | BIDIR/CLOCK | E6          | IO_38 | SDIOMUX     |
| H6          | IO_23 | BIDIR/CLOCK | 33          | IO_23 | BIDIR/CLOCK | F6          | IO_39 | SDIOMUX     |
| G6          | IO_24 | BIDIR/CLOCK | 32          | IO_24 | BIDIR/CLOCK | D7          | IO_43 | SDIOMUX     |
| F7          | IO_25 | BIDIR       | 31          | IO_25 | BIDIR       | E7          | IO_44 | SDIOMUX     |
| F6          | IO_26 | BIDIR       | 30          | IO_26 | BIDIR       | F7          | IO_45 | SDIOMUX     |
| H5          | IO_27 | BIDIR       | 28          | IO_27 | BIDIR       |             |       |             |
| G5          | IO_28 | BIDIR       | 27          | IO_28 | BIDIR       |             |       |             |
| F5          | IO_29 | BIDIR       | 26          | IO_29 | BIDIR       |             |       |             |
| F4          | IO_30 | BIDIR       | 25          | IO_30 | BIDIR       |             |       |             |
| G4          | IO_31 | BIDIR       | 23          | IO_31 | BIDIR       |             |       |             |
| H4          | IO_32 | SDIOMUX     | 22          | IO_32 | SDIOMUX     |             |       |             |
| E3          | IO_33 | SDIOMUX     | 21          | IO_33 | SDIOMUX     |             |       |             |
| F3          | IO_34 | SDIOMUX     | 20          | IO_34 | SDIOMUX     |             |       |             |
| F2          | IO_35 | SDIOMUX     | 18          | IO_35 | SDIOMUX     |             |       |             |
| H3          | IO_36 | SDIOMUX     | 17          | IO_36 | SDIOMUX     |             |       |             |
| G2          | IO_37 | SDIOMUX     | 15          | IO_37 | SDIOMUX     |             |       |             |
| E2          | IO_38 | SDIOMUX     | 16          | IO_38 | SDIOMUX     |             |       |             |
| H2          | IO_39 | SDIOMUX     | 11          | IO_39 | SDIOMUX     |             |       |             |
| D2          | IO_40 | SDIOMUX     | 13          | IO_40 | SDIOMUX     |             |       |             |
| F1          | IO_41 | SDIOMUX     | 14          | IO_41 | SDIOMUX     |             |       |             |
| H1          | IO_42 | SDIOMUX     | 10          | IO_42 | SDIOMUX     |             |       |             |
| D1          | IO_43 | SDIOMUX     | 7           | IO_43 | SDIOMUX     |             |       |             |
| E1          | IO_44 | SDIOMUX     | 8           | IO_44 | SDIOMUX     |             |       |             |
| G1          | IO_45 | SDIOMUX     | 9           | IO_45 | SDIOMUX     |             |       |             |

Figure 10.2.3.1.1: Pin map.



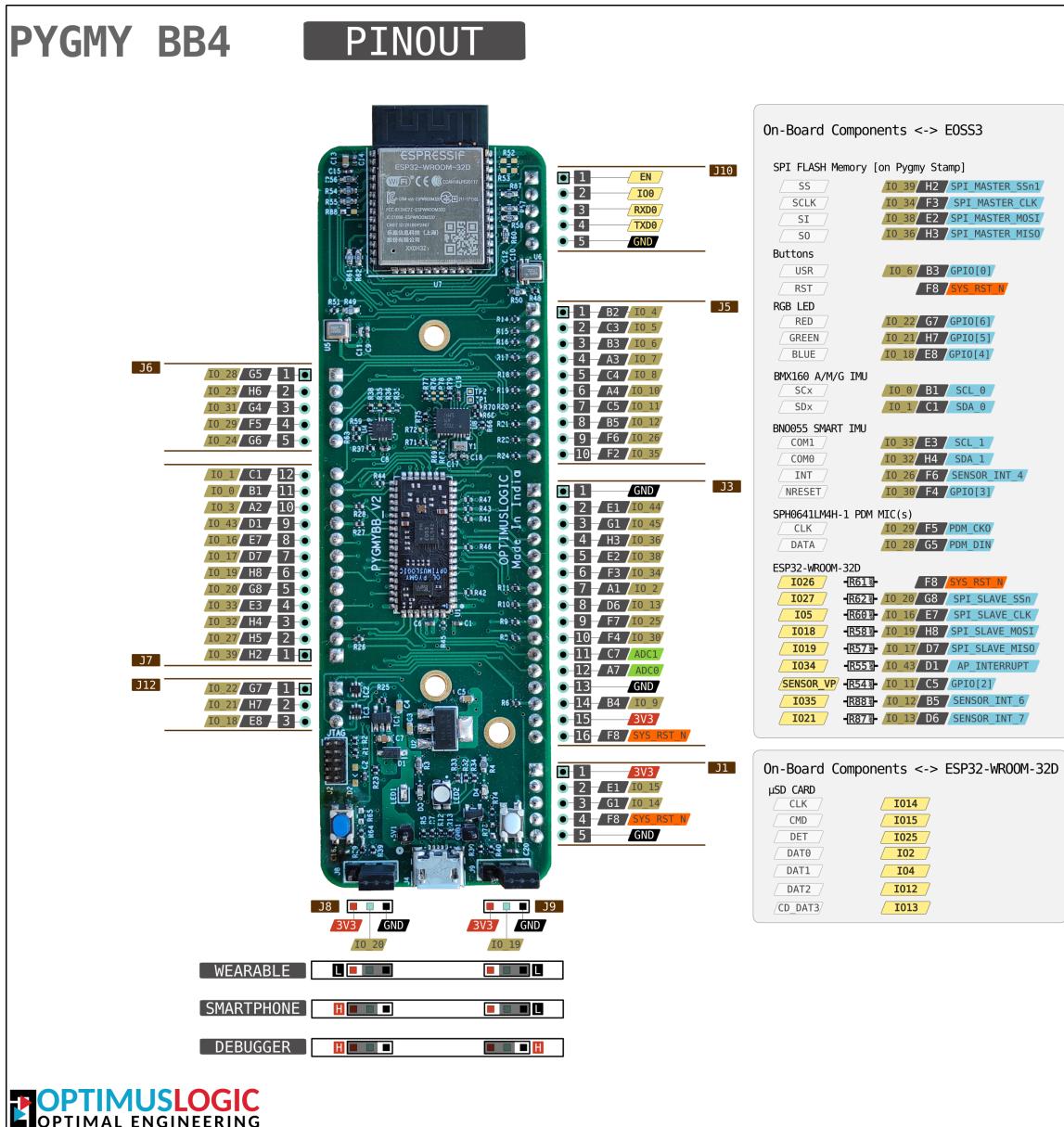


Figure 10.3.2.3.2: Pin diagram

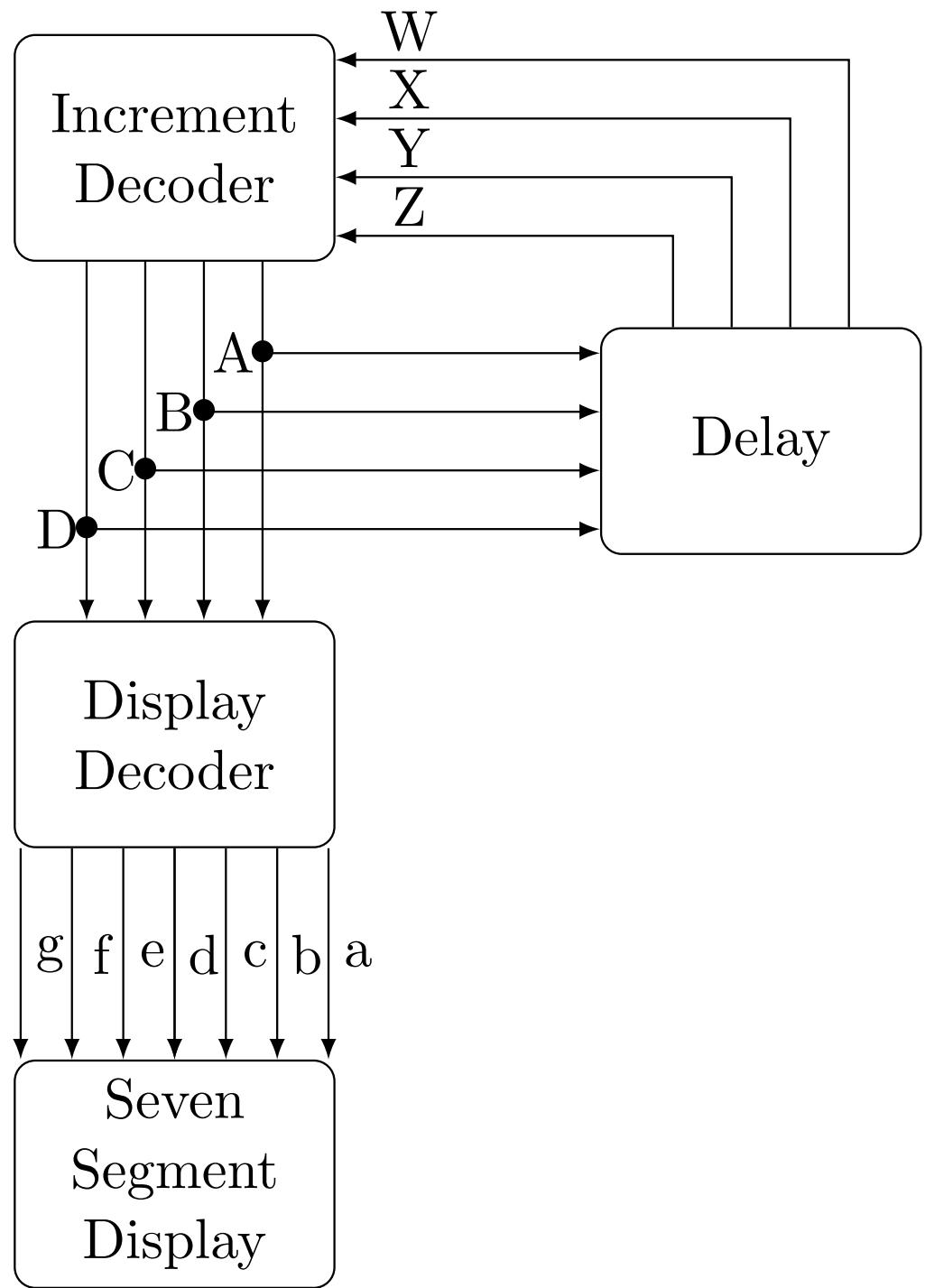


Figure 10.3.3.1.1: Block diagram of a decade counter.



# Chapter 11

## Vaman-ESP32

### 11.1. Software

All codes used in this document are available in the following directory

```
vaman/esp32/codes
```

### 11.2. Flash Vaman-ESP32 using Arduino

11.2.1. Do not power any devices. Make connections as shown in Table 11.2.1.1 and Fig. 11.2.1.1.

| VAMAN LC PINS | ARDUINO PINS |
|---------------|--------------|
| 3.3           | 3.3          |
| GND           | GND          |
| TXD0          | TXD          |
| RXD0          | RXD          |
| 0             | GND          |
| EN            | GND          |

Table 11.2.1.1:

11.2.2. Now power on the devices and measure the voltage across the Vaman-ESP Rx pin

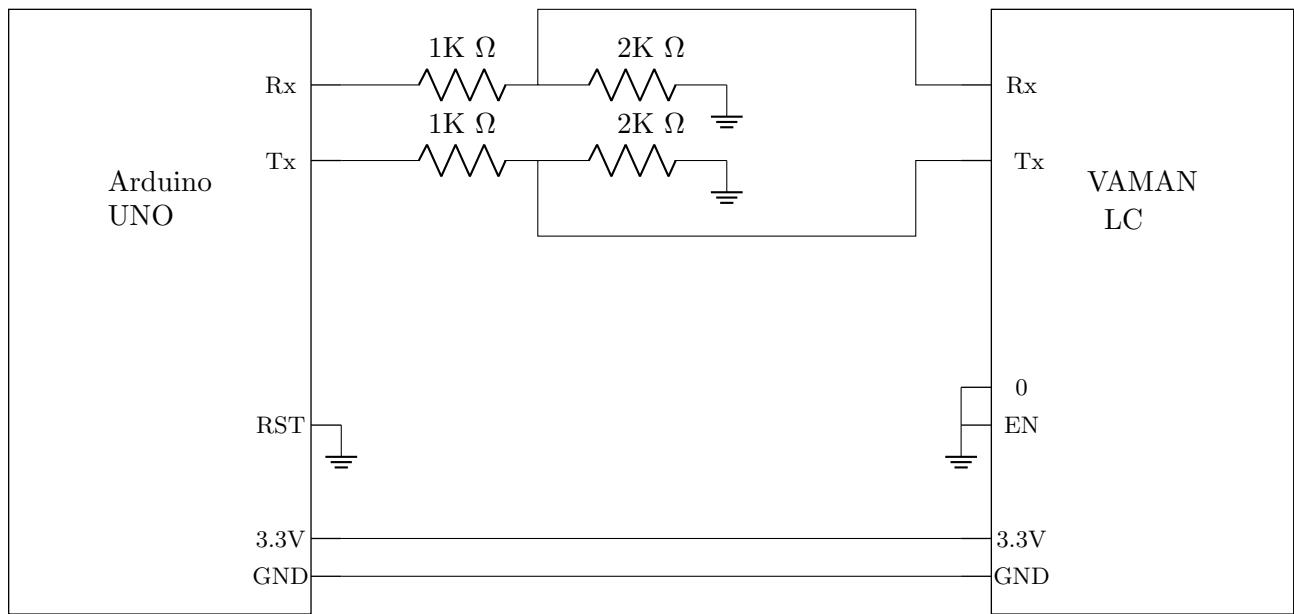


Figure 11.2.1.1: Circuit Connections

using a multimeter. Comment.

#### 11.2.3. For compiling and generating the bin file

```
cd vaman/esp32/codes/ide/blink
pio run
```

#### 11.2.4. For uploading bin file to Vaman through ArduinoDroid application

1. Open the Droid Application
2. Click the three dots **in** the top right corner
3. Navigate to Settings → Board Type
4. Select ESP32 → DOIT ESP32 DEVKIT V1
5. Change the upload speed to 115200
6. Upload the generated .bin file

While the dots are printed on the screen, disconnect the EN wire from GND. Make sure that the Vaman board is not powering any device while flashing. The Vaman-ESP should now flash.

11.2.5. After flashing, disconnect pin 0 on Vaman-ESP from GND. Power on Vaman and the appropriate LED will blink.

11.2.6. Open

```
vaman/esp32/codes/ide/blink/src/main.cpp
```

and change the delay to

```
delay(2000);
```

and execute the code by following the steps above.

## 11.3. OTA

11.3.1. Flash the following code through USB-UART.

```
vaman/esp32/codes/ide/ota/setup
```

after entering your wifi username and password (in quotes below)

```
#define STASSID "...” // Add your network credentials  
#define STAPSK "...”
```

in src/main.cpp file

11.3.2. You should be able to find the ip address of your vaman-esp using

```
ifconfig  
nmap -sn 192.168.231.1/24
```

where your computer's ip address is the output of ifconfig and given by 192.168.231.x

11.3.3. Assuming that the username is gvv and password is abcd, flash the following code wirelessly

```
vaman/esp32/codes/ide/blink
```

through

```
pio run  
pio run -t nobuild -t upload --upload-port 192.168.231.245
```

where you may replace the above ip address with the ip address of your vaman-esp.

11.3.4. Connect pin 2 to an LED to see it blinking.

## 11.4. Onboard LED

11.4.1. Connect the pins between Vaman-ESP32 and Vaman-PYGMY as per Table 11.4.1.1

| ESP32 | Vaman  |
|-------|--------|
| GPIO2 | GPIO18 |
| GPIO4 | GPIO21 |
| GPIO5 | GPIO22 |

Table 11.4.1.1:

11.4.2. Flash the following code OTA

```
vaman/esp32/codes/ide/ota/blinkt
```

You should see the onboard green LED blinking.

#### 11.4.3. Change the blink duration to 100 ms.

