Version: A1

SW Design Description (SDD)

Multi-Color LED Pattern Control SW

3rd Floor, Ajiyal Complex
Kakkanad, Kochi 682 030
Kerala, India

Table of Contents

Introduction **3**

1.1 The project 3

1.2 Architecture Block diagram 3

Detailed Design **3**

2.1 Main() 4

2.1 UI Handler Task 5

2.2 Pattern Execute Task 10

2.2.1 State Diagram 11

2.3 Communication Handler 13

2.3 Inter Task Communication 19

2.3.1 COMMUNICATION QUEUE 21

2.3.2 CONNECTION STATUS QUEUE 21

2.3.3 PATTERN CONTROL QUEUE 21

2.3.3 PATTERN STATUS QUEUE 21

1.Introduction

This document describes the SW (detailed) Design Description (SDD) of the case study exercise “Multi Colour LED controller”. (Refer PRD for product requirement description & SADD for Architecture design description)

1.1 The project

The project is to develop software design for controlling Multicolor LED pattern Controller. MLC (Multicolor LED Pattern Controller) is a pattern controller which acts as a master or a slave depending on the jumper condition. If it is MLC master, then it must be able to accept data from LED Operator and give the data to MLC Slave which is another MLC configured as slave connected via I^2C.

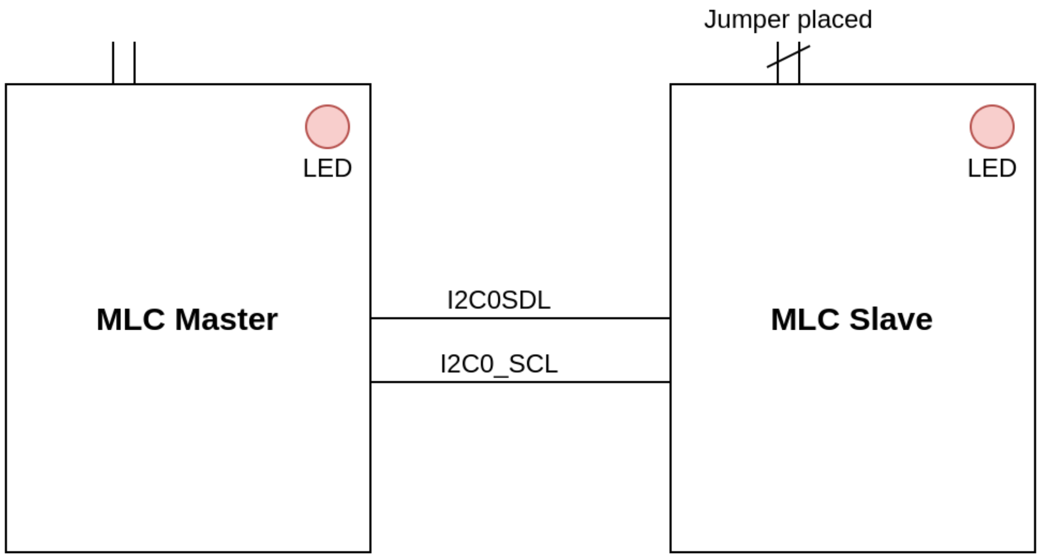


Figure 1.1 - MLC High Level Block Diagram

1.2 Architecture Block diagram

Figure below illustrates the high level block diagram of the SW.

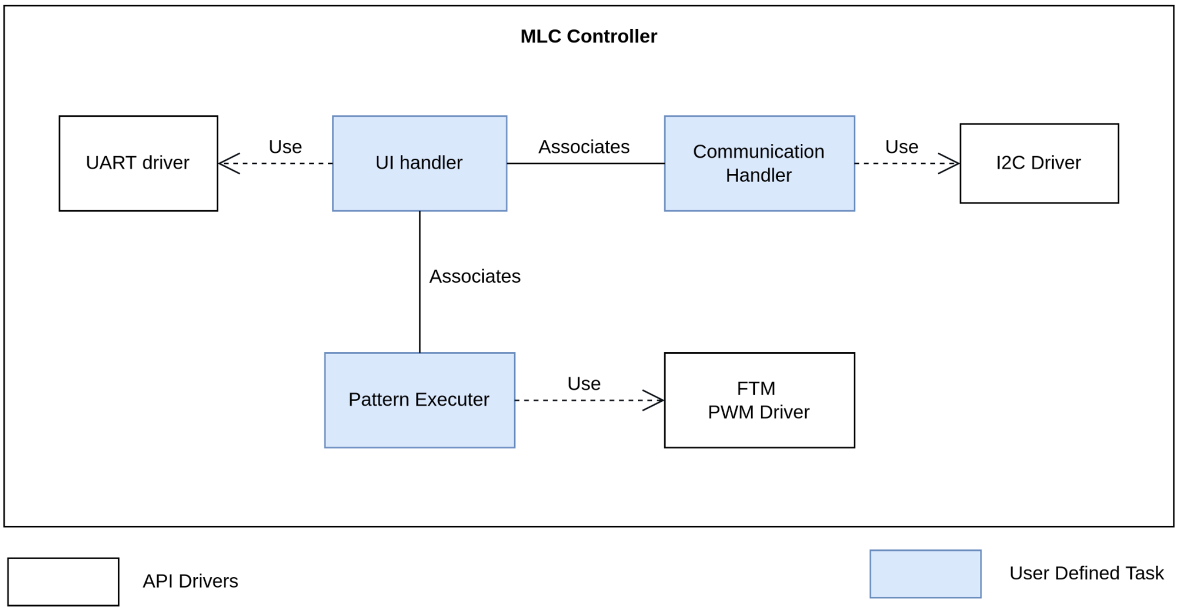


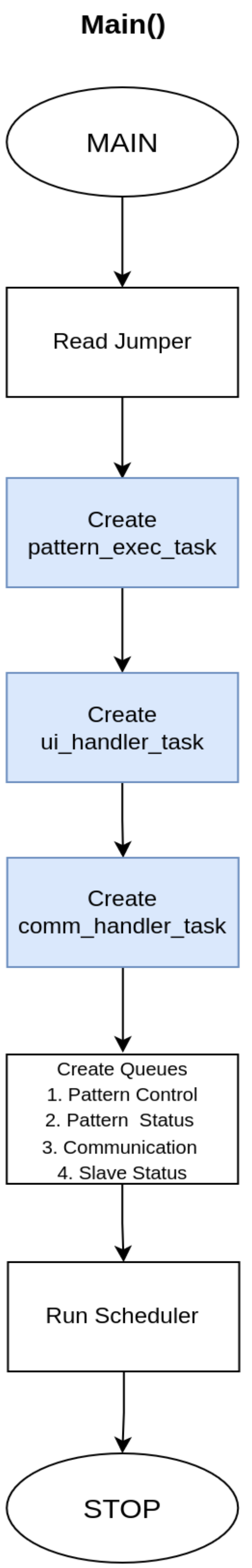
Figure 1.2 - MLC Architecture Design

2. Detailed Design

This section describes each user defined tasks and their logic flow This logical block diagram describes how main() is implemented in the RTOS context. At first the jumper status is checked and then according to the jumper status MLC will be configured as master or slave. For each instance three tasks must be created

1. **pattern_exec_task**
2. **ui_handler_task**
3. **comm_handler_task**

2.1 Main()


Main()
MAIN
Read Jumper
Create pattern_exec_task
Create ui_handler_task
Create comm_handler_task
Create Queues
1. Pattern Control
2. Pattern Status
3. Communication
4. Slave Status
Run Scheduler
STOP

2.1 UI Handler Task

UI Handler Task is the task which handles User Interfaces.

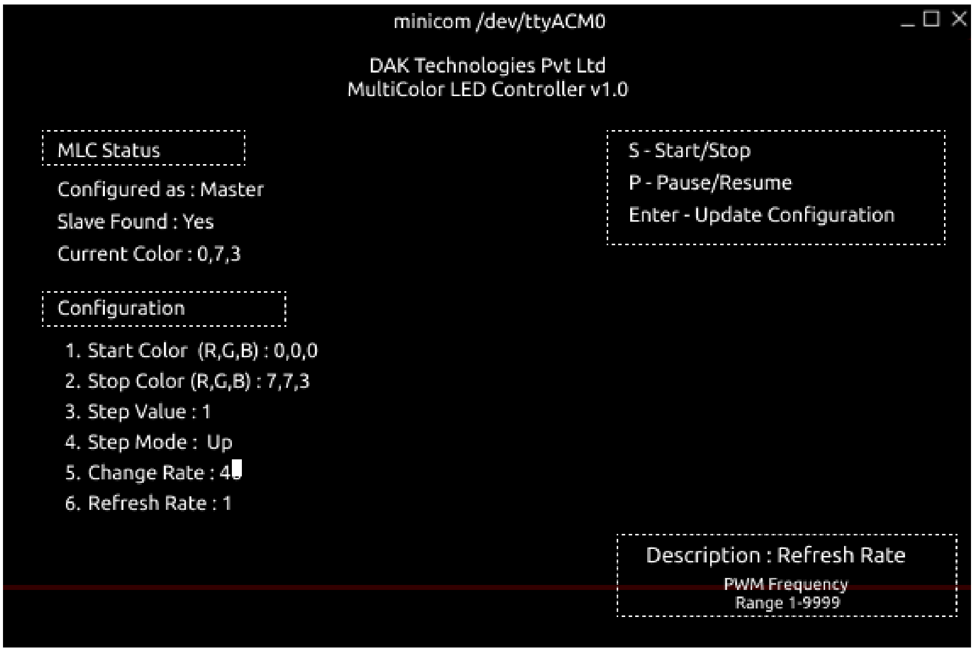


Figure 2.1: User Interface of Master (Default)

In this UI, each configuration can be changed by using the navigation key. For eg: As per the UI shown, in order to change the Refresh rate, we have to press "^" and move the cursor to change rate, then give the value to be reflected. After entering the corresponding value you can change another parameter in the same way or you can apply the configuration by pressing ***ENTER*** key

Figure 2.3: User Interface of Slave (Default)

Logic flow of the UI Handler is given below

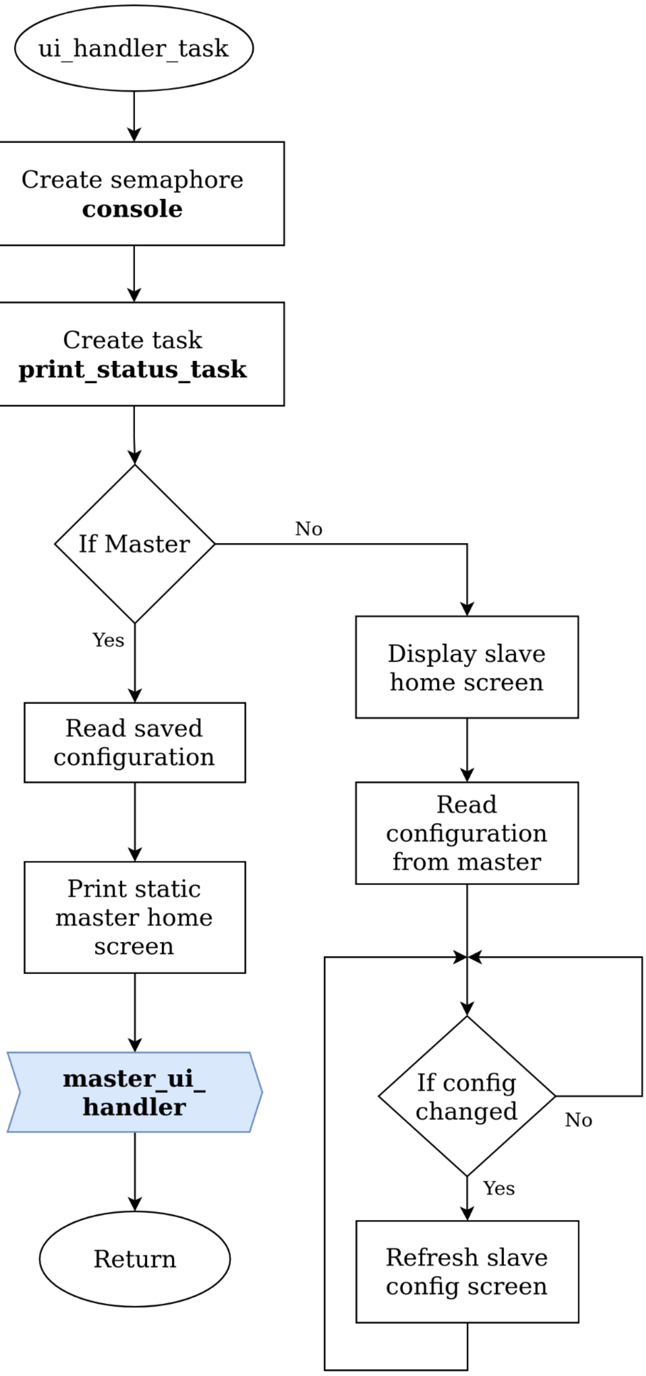


Figure 2.4: UI Handler Abstract view (Flow Chart)

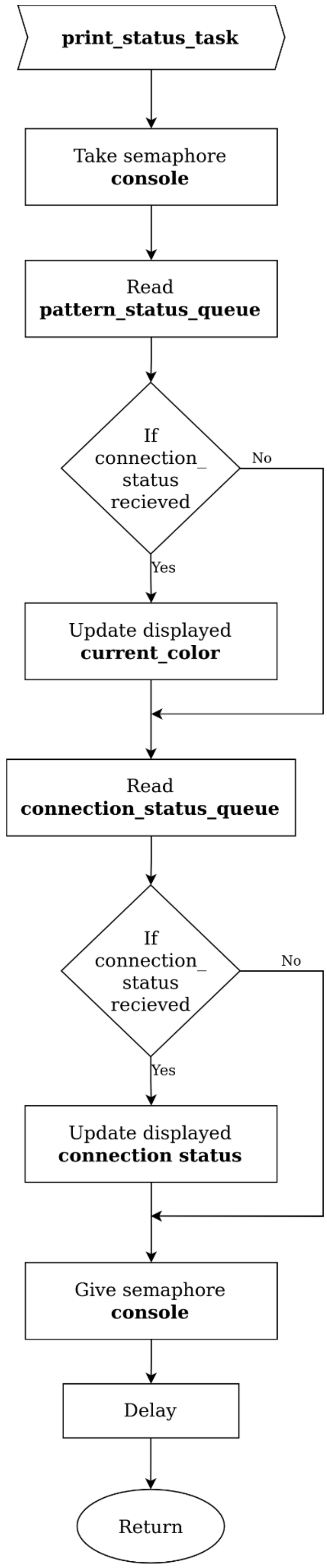


Figure 2.5: Display Current Color

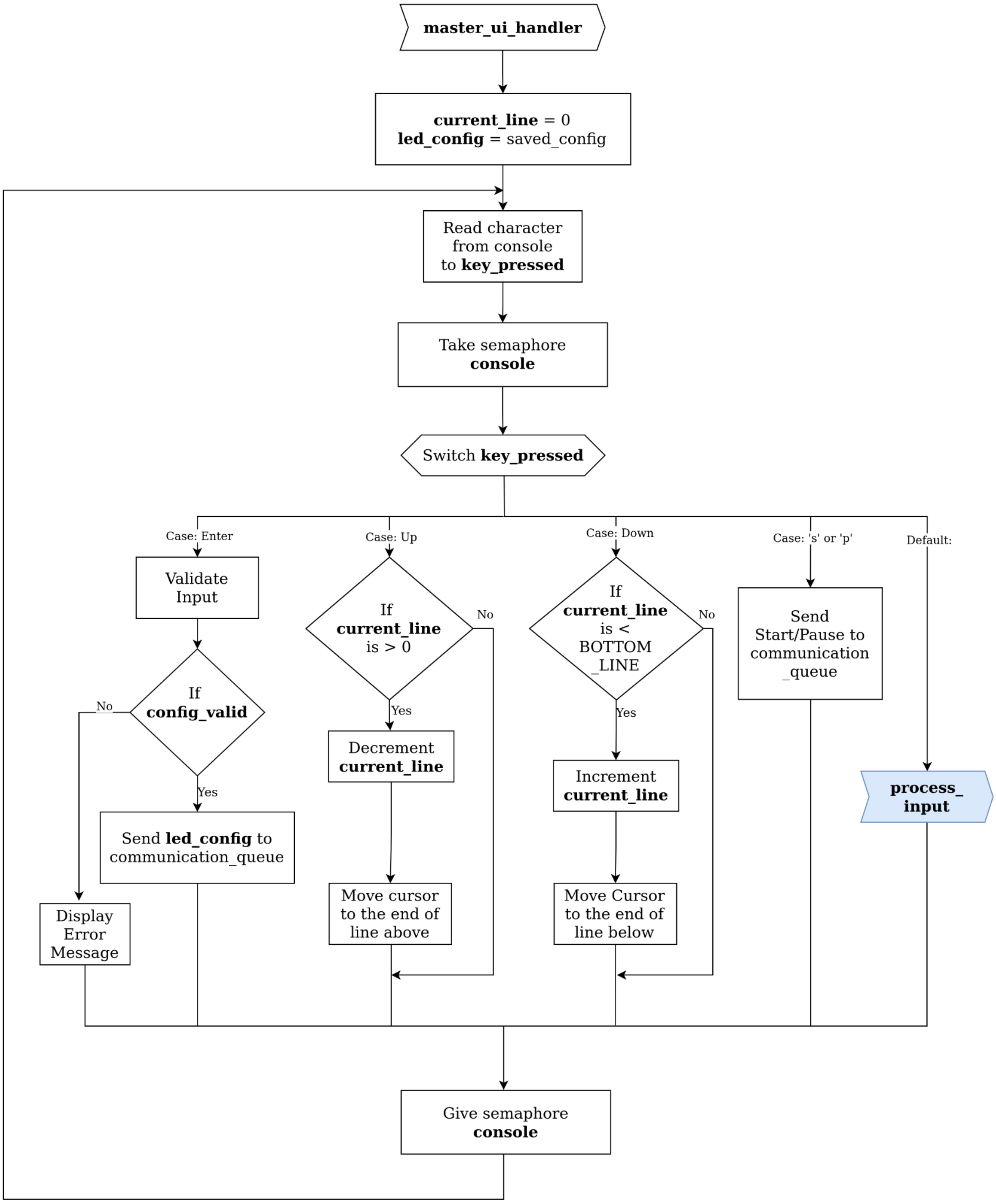


Figure 2.6: Master UI Handler(Flow Chart)

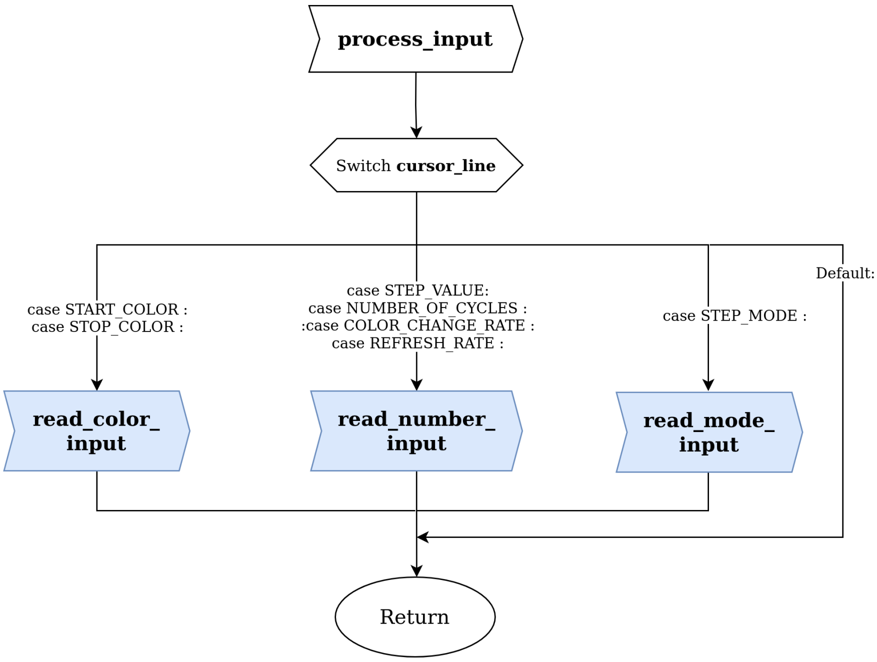


Figure 2.7: Master UI Handler User Input (Flow Chart)

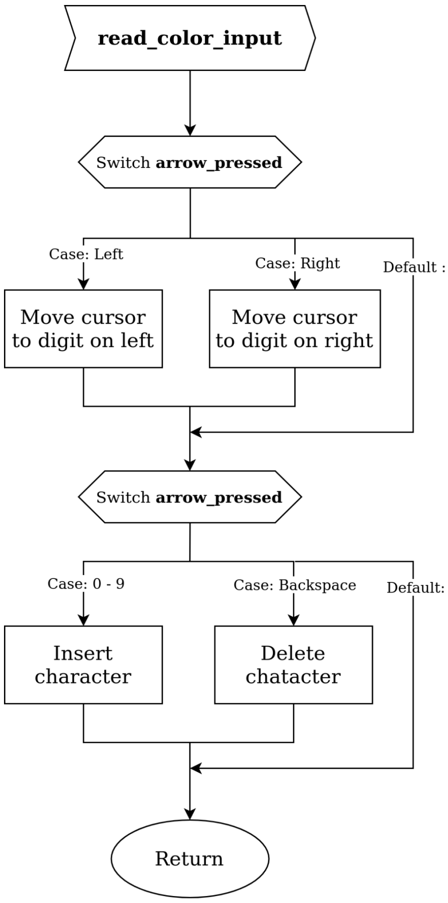


Figure 2.7: Read color input (Flow Chart)

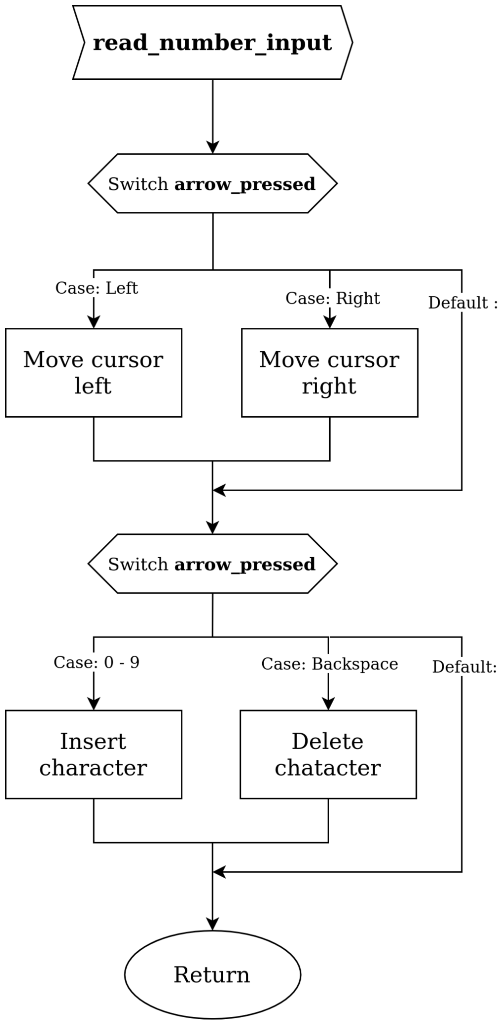


Figure 2.7: Read number input (Flow Chart)

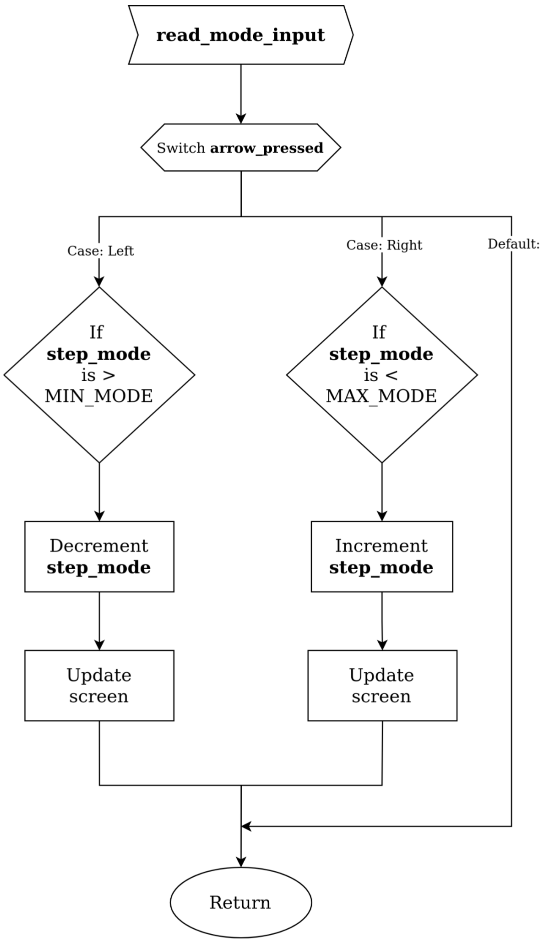


Figure 2.7: Read mode input (Flow Chart)

2.2 Pattern Execute Task

This task executes the user defined pattern using PWM. If the User doesn't enter any data, then default configuration will be used to execute the pattern.

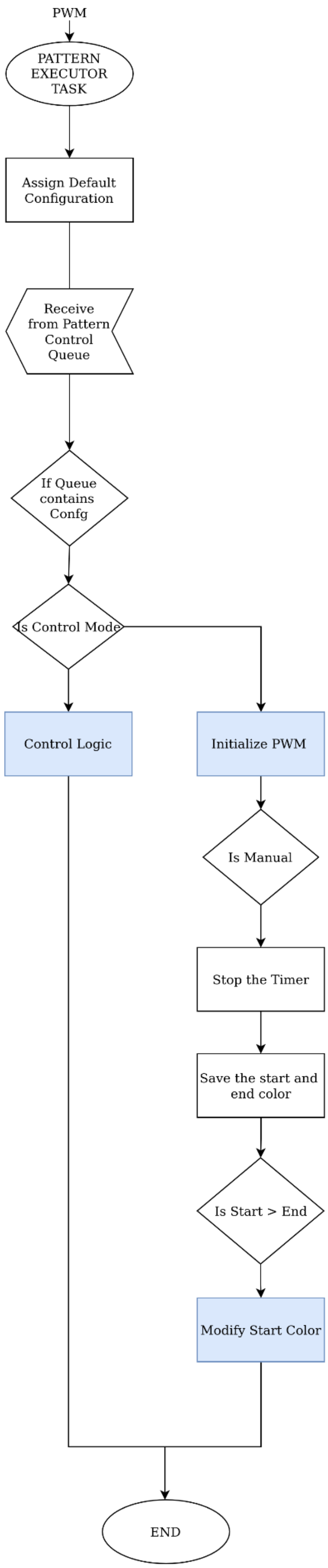

PWM
PATTERN EXECUTOR TASK
Assign Default Configuration
Receive from Pattern Control Queue
If Queue contains Confg
Is Control Mode
Control Logic
Initialize PWM
Is Manual
Stop the Timer
Save the start and end color
Is Start > End
Modify Start Color
END

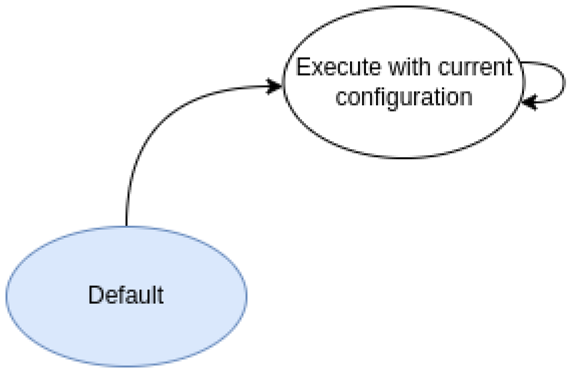
Figure 2.10: Pattern Executor (Flow Chart)

Figure 2.10: Pattern Executor (Flow Chart)

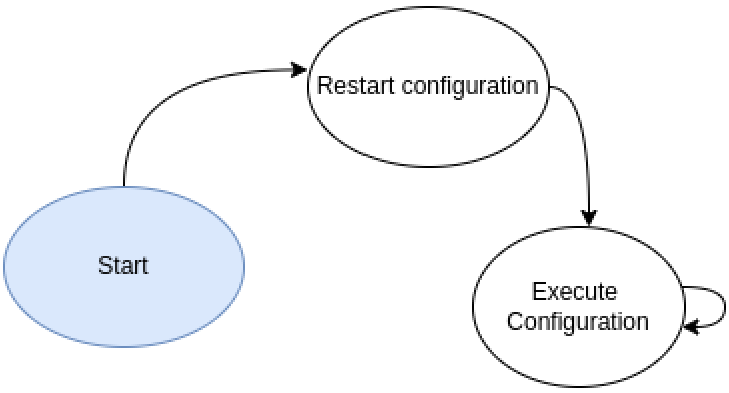
Figure 2.10: Pattern Executor (Flow Chart)

2.2.1 State Diagram

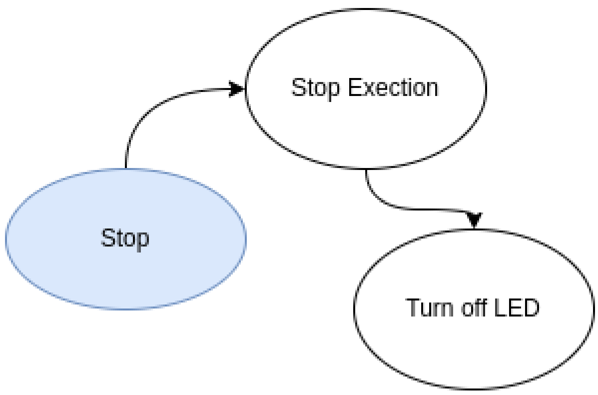
- Execute Default Configuration



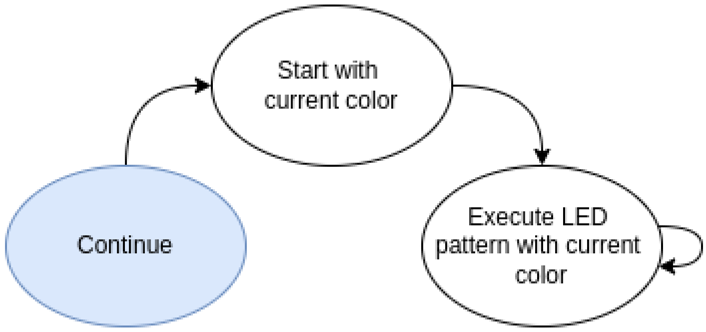
- **START** Command State Diagram



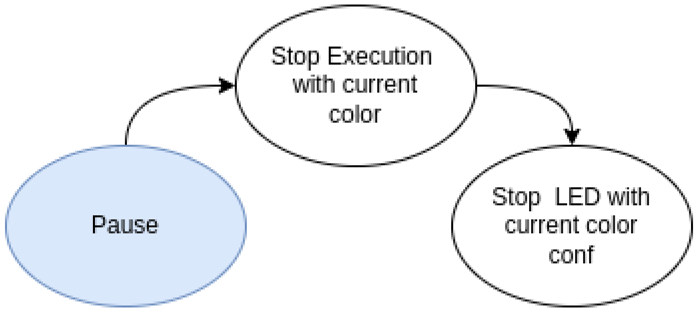
- **STOP** Command State Diagram



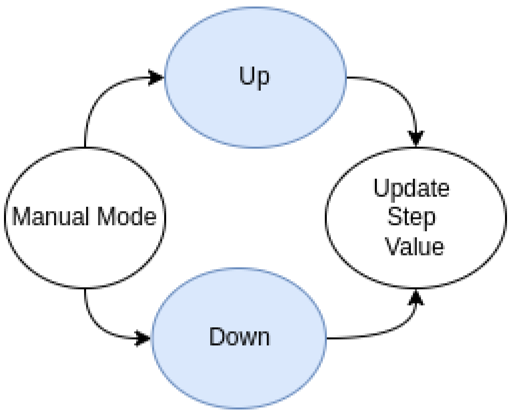
- **CONTINUE** Command State Diagram



- **PAUSE** Command State Diagram



- **MANUAL UP/DOWN** Command State Diagram



2.3 Communication Handler

Communication handler is the task in which the communication between MLC Master and MLC Slave takes place. Communication Handler will continuously check whether a user data received or not. If new data received, then this data will be transferred to MLC Slave..

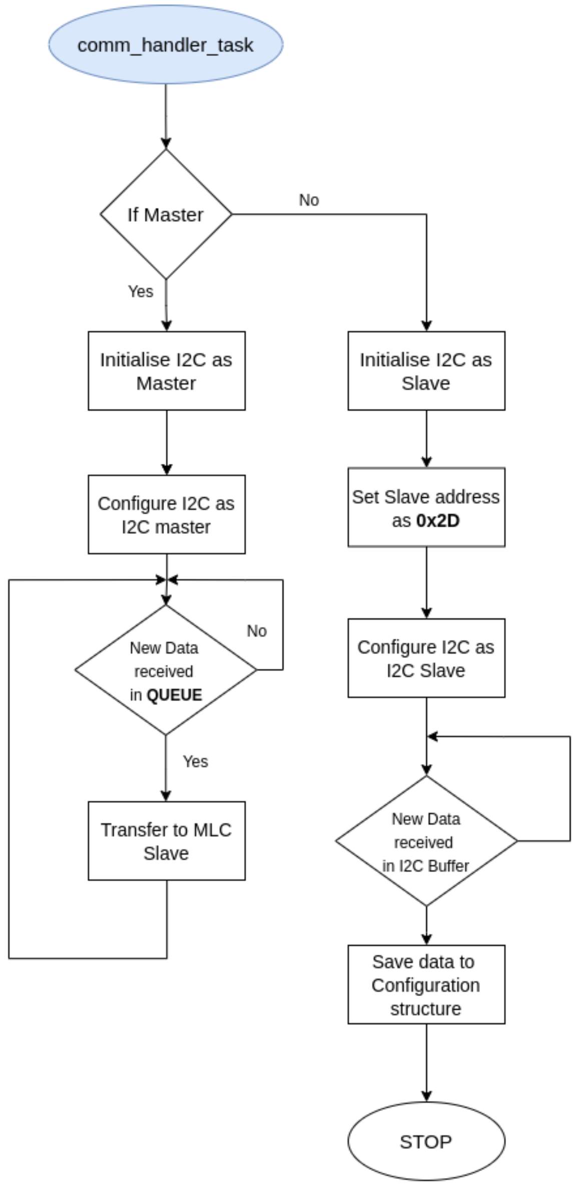


Figure 2.11 I^2C Communication Initialization

In order to identify the presence of MLC slave for master and vice versa, a handshake must be performed to identify the existance of the slave to master and master to slave

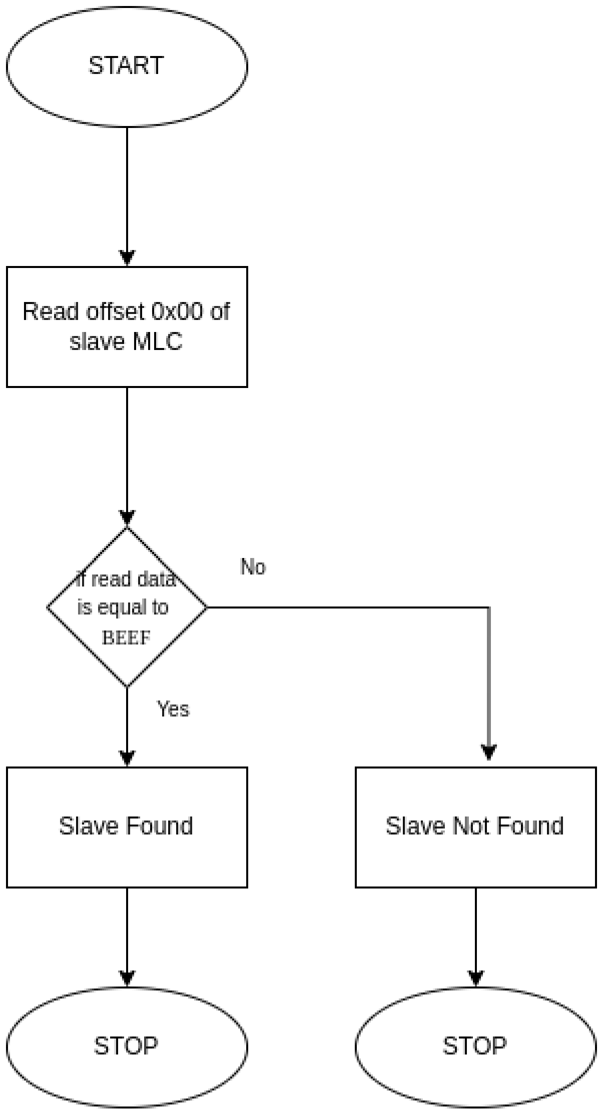


Figure 2.12: Communication Handler (Handshaking) - Detection of slave by master

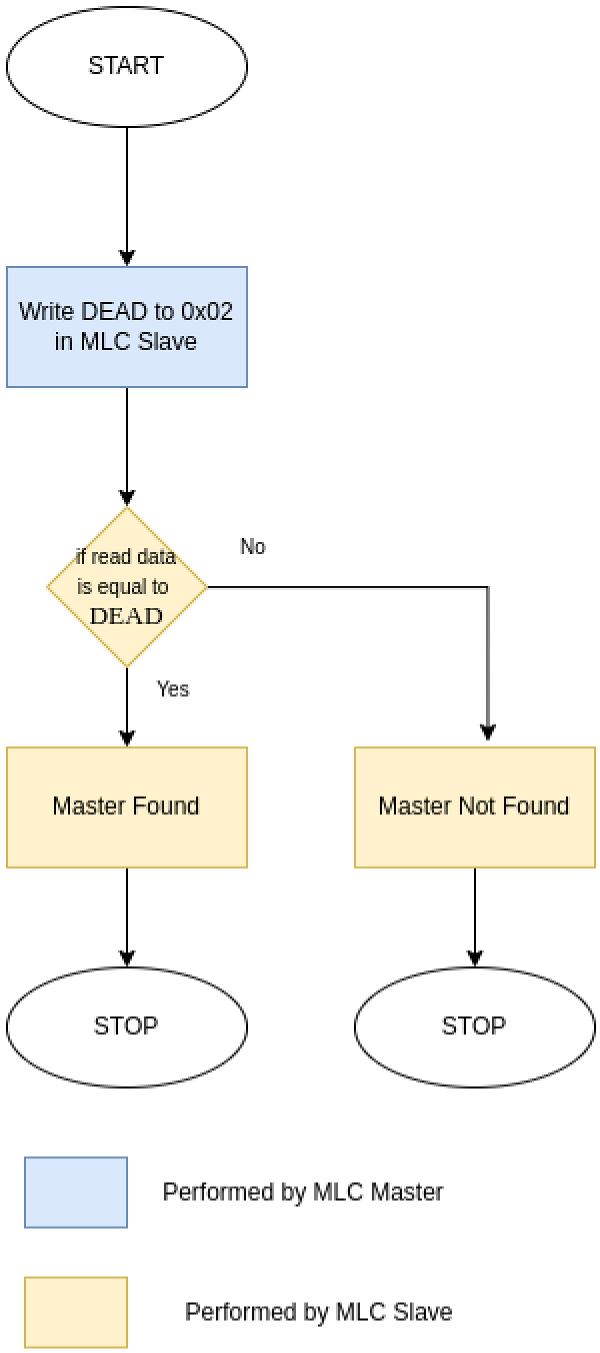


Figure 3.13: Communication Handler (Handshaking) - Detection of Master By Slave

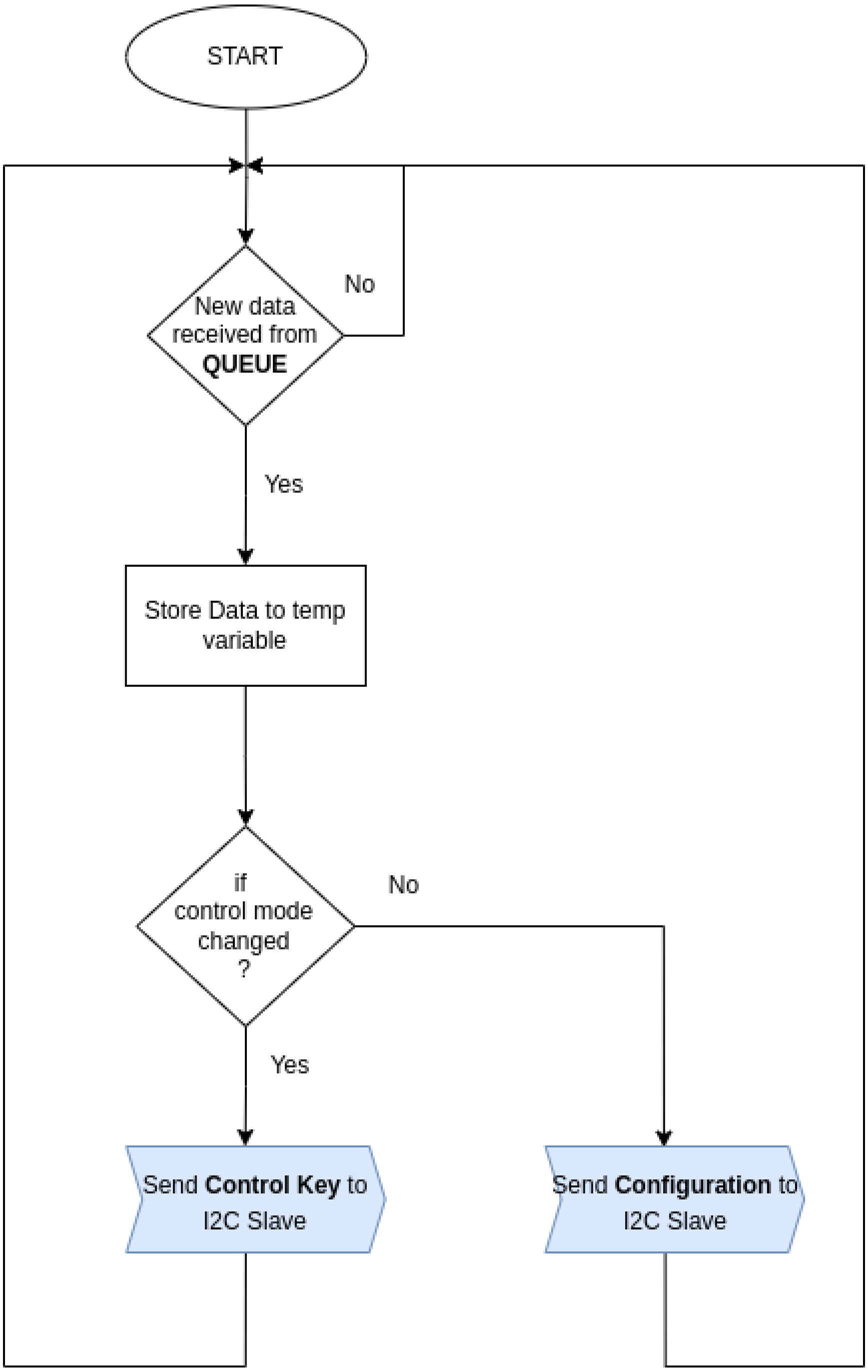


Figure 2.14: Transfering Data to MLC Slave From MLC Master

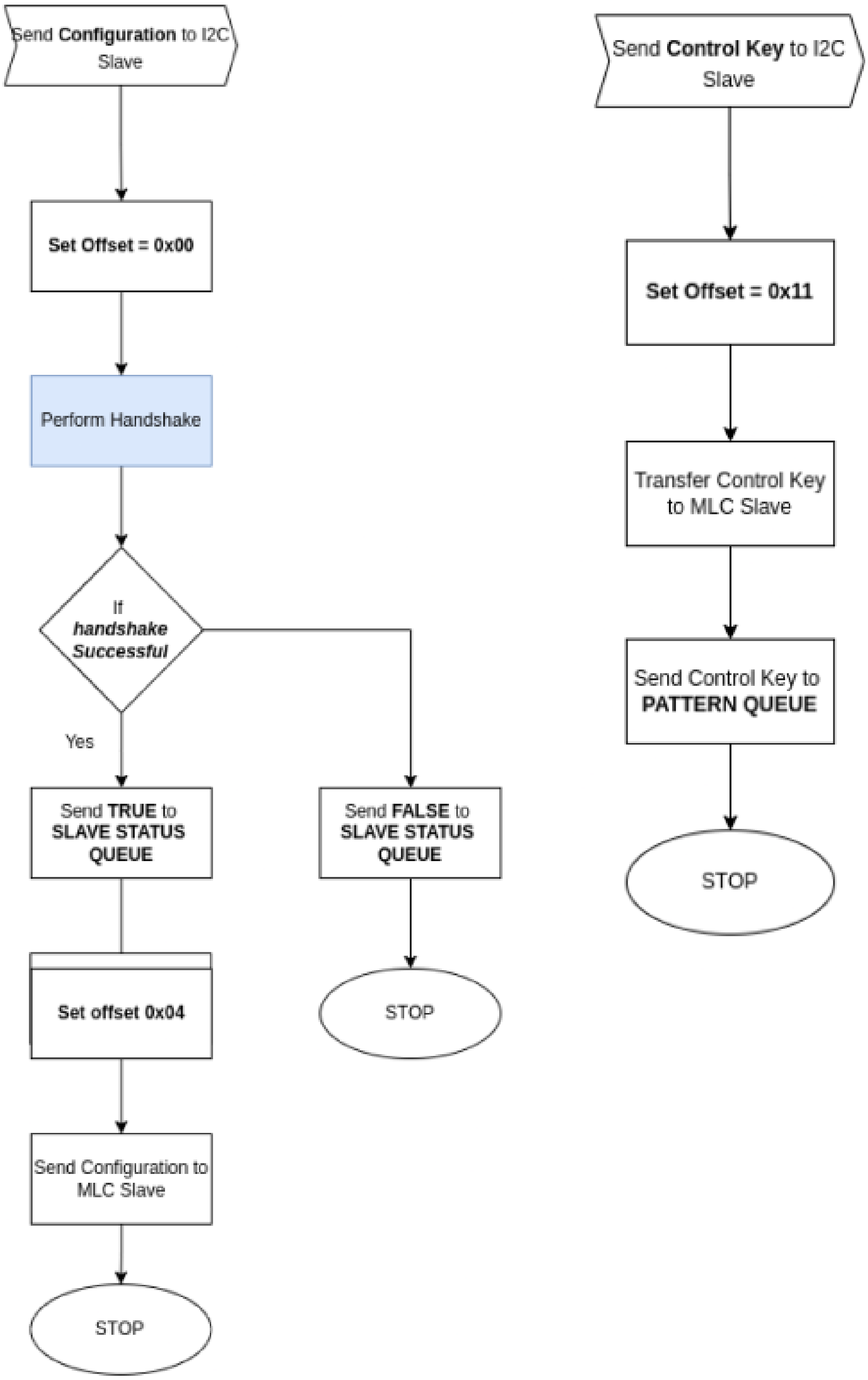


Figure 2.15: Transfering Data to MLC Slave From MLC Master (Detailed View)

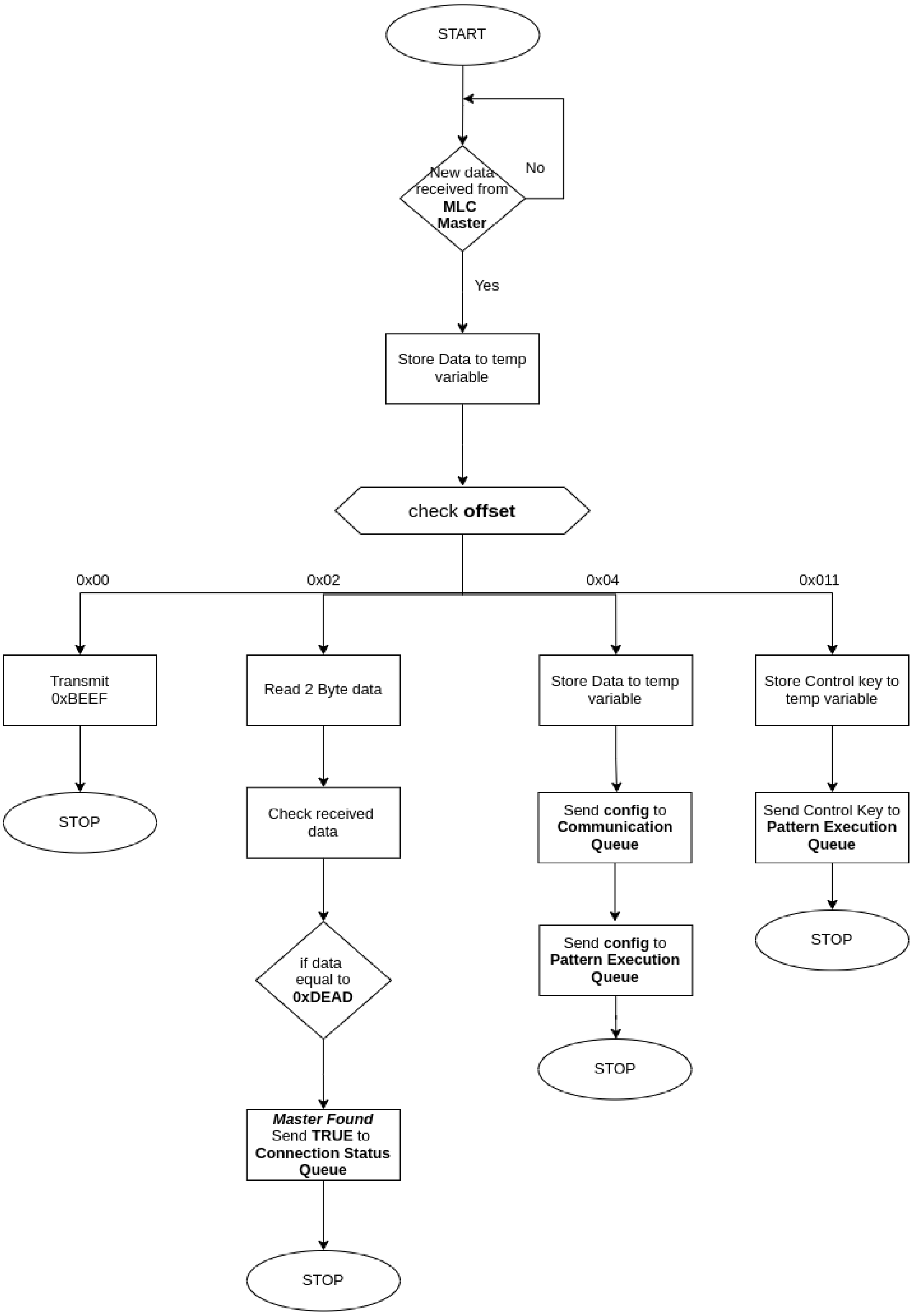


Figure 2.16: MLC Slave Detailed Flow Chart

2.3 Inter Task Communication

This section describes Queue Management, which is used to communicate between tasks.

Queue Name	Data From (Task)	Data To (Task)	Type Of Data
COMMUNICATION	UI HANDLER	COMMUNICATION HANDLER	User Data (Structure)
PATTERN STATUS	PATTERN EXECUTOR	UI HANDLER	Current Color
PATTERN CONTROL	COMMUNICATION HANDLER	PATTERN EXECUTOR	START STOP CONTINUE PAUSE <>
CONNECTION STATUS	PATTERN EXECUTOR	UI HANDLER	Slave Detected or Not

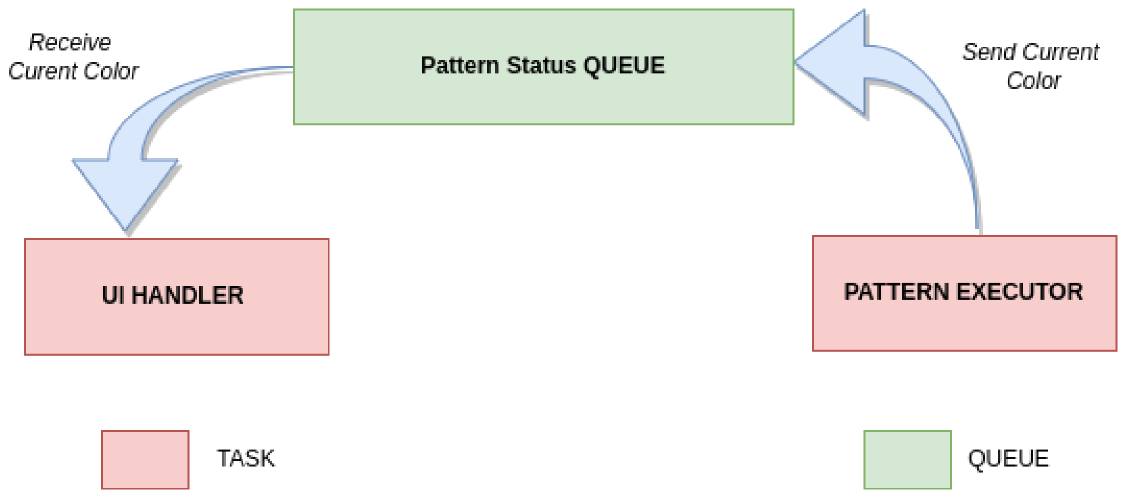


Figure 2.17: Pattern Status Queue

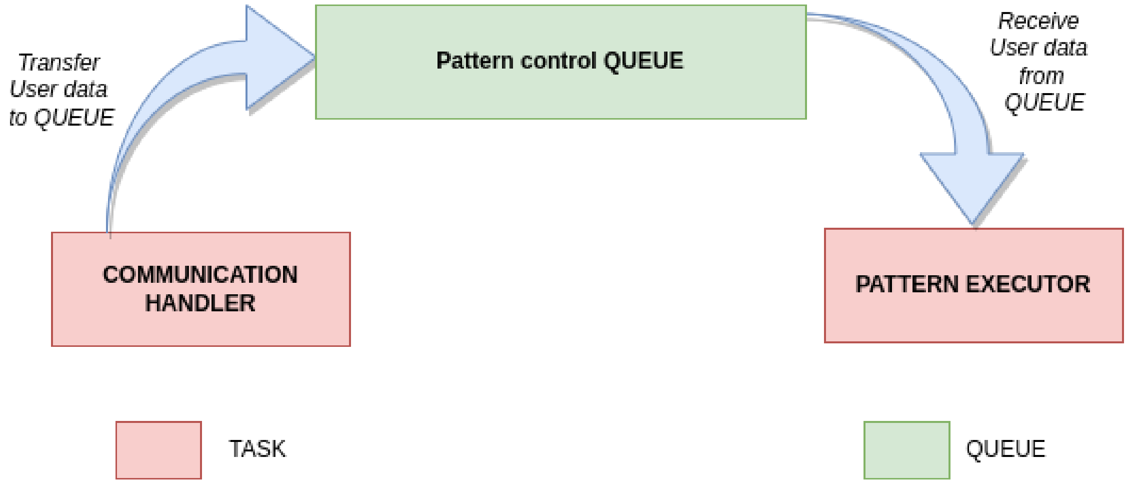


Figure 2.18: Pattern Control Queue

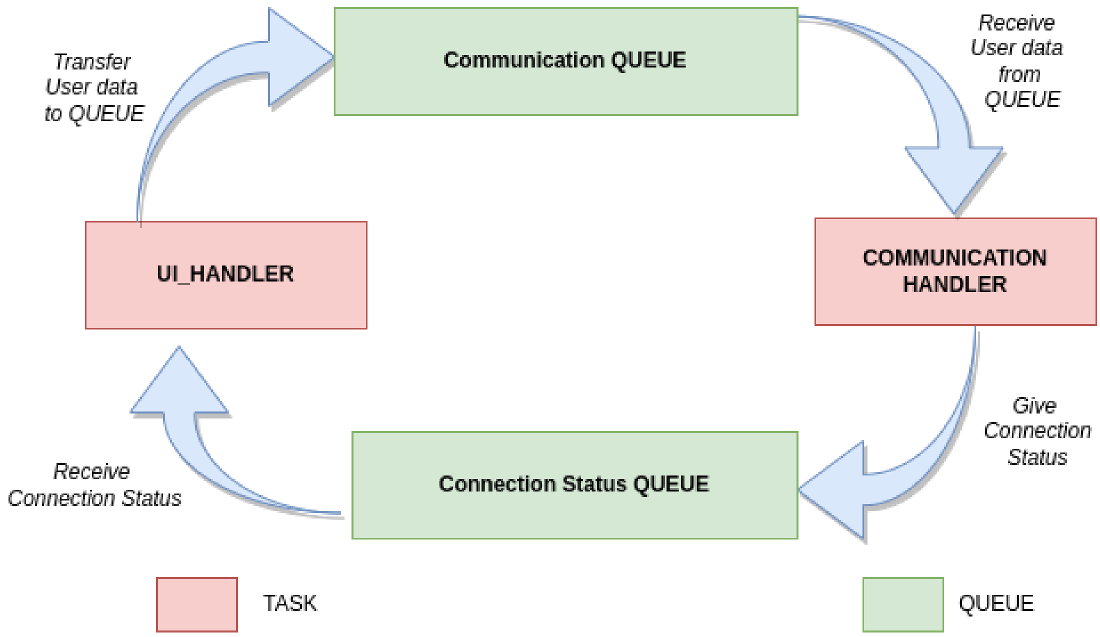


Figure 2.19: Communication & Connection Status Queue

2.3.1 COMMUNICATION QUEUE

In order to communicate between UI HANDLER and COMMUNICATION HANDLER Task, a queue named Communication QUEUE is created. The data entered by the user will be accepted in the UI HANDLER and it will be validated there; after that this data will be sent to Communication QUEUE. COMMUNICATION HANDLER reads the data from this queue and sends it to the MLC Slave

2.3.2 CONNECTION STATUS QUEUE

Connection Status Queue is used to communicate between theUI HANDLER & COMMUNICATION HANDLER, the status of MLC slave or MLC Master will be sent to the UI HANDLER using this QUEUE

2.3.3 PATTERN CONTROL QUEUE

Pattern Queue is used to transfer the user inputted data to PWM controller and start Execution

2.3.3 PATTERN STATUS QUEUE

Pattern Status Queue gives current color to UI HANDLER