Lab 8: Programming with ST, FBD, SFC

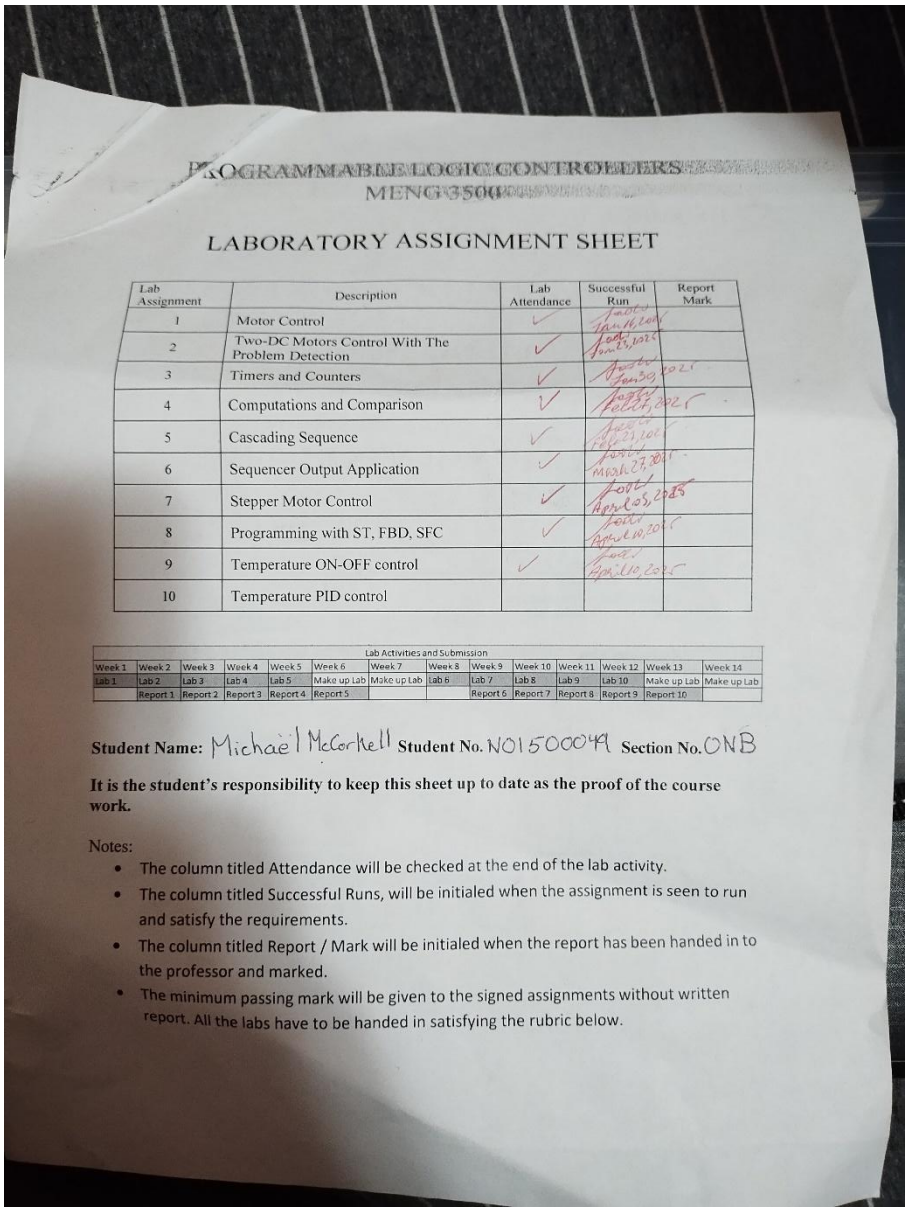
Michael McCorkell

Humber Polytechnics

Programmable Logic Controllers: MENG 3500 0NB

Savdulla Kazazi

April 14, 2025, 2025



PROGRAMMABLE LOGIC CONTROLLERS
MENG 3500

LABORATORY ASSIGNMENT SHEET

Lab Assignment	Description	Lab Attendance	Successful Run	Report Mark
1	Motor Control	✓		
2	Two-DC Motors Control With The Problem Detection	✓		
3	Timers and Counters	✓		
4	Computations and Comparison	✓		
5	Cascading Sequence	✓		
6	Sequencer Output Application	✓		
7	Stepper Motor Control	✓		
8	Programming with ST, FBD, SFC	✓		
9	Temperature ON-OFF control	✓		
10	Temperature PID control			

Lab Activities and Submission													
Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Lab 1	Lab 2	Lab 3	Lab 4	Lab 5	Make up lab	Make up Lab	Lab 6	Lab 7	Lab 8	Lab 9	Lab 10	Make up Lab	Make up Lab
	Report 1	Report 2	Report 3	Report 4	Report 5			Report 6	Report 7	Report 8	Report 9	Report 10	

Student Name: Michael McCorkell **Student No.** N01500049 **Section No.** ONB

It is the student's responsibility to keep this sheet up to date as the proof of the course work.

Notes:

- The column titled Attendance will be checked at the end of the lab activity.
- The column titled Successful Runs, will be initialed when the assignment is seen to run and satisfy the requirements.
- The column titled Report / Mark will be initialed when the report has been handed in to the professor and marked.
- The minimum passing mark will be given to the signed assignments without written report. All the labs have to be handed in satisfying the rubric below.

Assignment Summary: PLC LAB 8 – Structured Text (ST), Function Block Diagram (FBD), and Sequential Function Chart (SFC)

Objectives

This advanced lab explored three major IEC 61131-3 PLC programming languages: Structured Text (ST), Function Block Diagram (FBD), and Sequential Function Chart (SFC). The key objectives included:

1. Establishing stable communication between the PLC and the computer.
2. Designing and programming control logic using ST, FBD, and SFC methodologies.
3. Implementing diverse automation processes using each language's strengths.
4. Executing and monitoring program behavior through simulations and live testing.
5. Troubleshooting and resolving functional errors within each programming environment.

Description of Work Completed

A. Structured Text (ST) Program – Motor Operation Logic

- Motor Run Cycle:
  The motor was programmed to operate for 10 seconds per cycle, repeating 3 times in a row.
- Counter Control:
  After 3 successful cycles, the motor locks out from running again.
  A Reset Pushbutton is used to clear the counter and allow further motor operation.
- Pushbutton Controls:
  - START_PB: Starts the motor operation.
  - STOP_PB: Immediately halts the motor.
  - RESET_PB: Clears the counter to restart operation.
- Indicators:
  - GREEN_LIGHT: On while the motor is running.
  - RED_LIGHT: On when the motor is locked out.
- Program Flow:
  The motor cycles 3 times, with each 10-second run increasing a counter.
  Once the counter reaches 3, the motor is stalled, and only a reset re-enables operation.

B. Function Block Diagram (FBD) – Motor & Counter Logic

- Motor Control:
  Similar to the ST section, START and STOP pushbuttons control the motor's operation.
- Enhanced Counter Logic:
  - Every motor start increments the counter.
  - A Capacitive Sensor decrements the counter when triggered (e.g., for real-time detection or safety).
  - A Reset Pushbutton resets the counter to 0.
- Visual Indicators:
  - GREEN_LIGHT: Indicates motor is ON.
  - RED_LIGHT: Signals motor stall or stop condition.
- Program Flow:
  The motor runs and responds dynamically to both user inputs and sensor triggers.
  This layout demonstrates the flexibility of function blocks for event-based conditions and real-time updates.

C. Sequential Function Chart (SFC) – Cylinder Sequencing

- Cylinder Sequence Overview:
  A timed, step-by-step sequence was developed for three pneumatic cylinders using the SFC programming model.
  - Each step executes at 1-second intervals.
  - Cylinders transition in and out of extended positions in a precise cascading order.
- Seven-Step Sequence:

1. Cylinder A advances
2. Cylinder B advances
3. Cylinder C advances
4. Cylinder A retracts
5. Cylinder B retracts

6. Cylinder C retracts
7. Reset and loop back to step 1

- Control Elements:
  - START_PB: Initiates the full sequence.
  - Timers: Control the duration of each step.
  - Automatic Repeat: After completing all steps, the sequence loops until manually stopped.
- Key Features:
  - The SFC model ensured organized, repeatable state transitions.
  - Each cylinder's action was carefully timed and ordered to avoid mechanical conflicts.

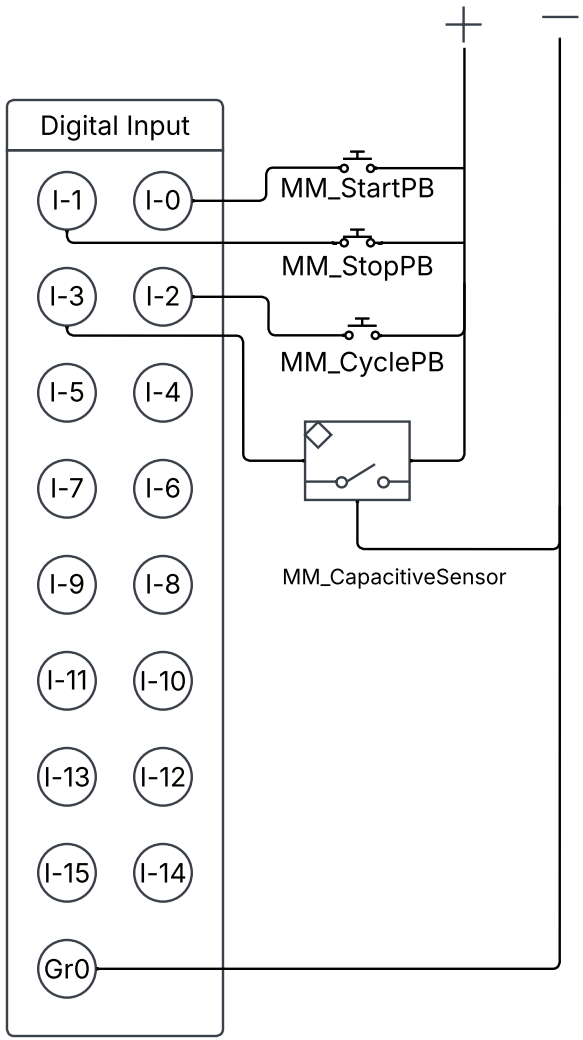
Conclusions

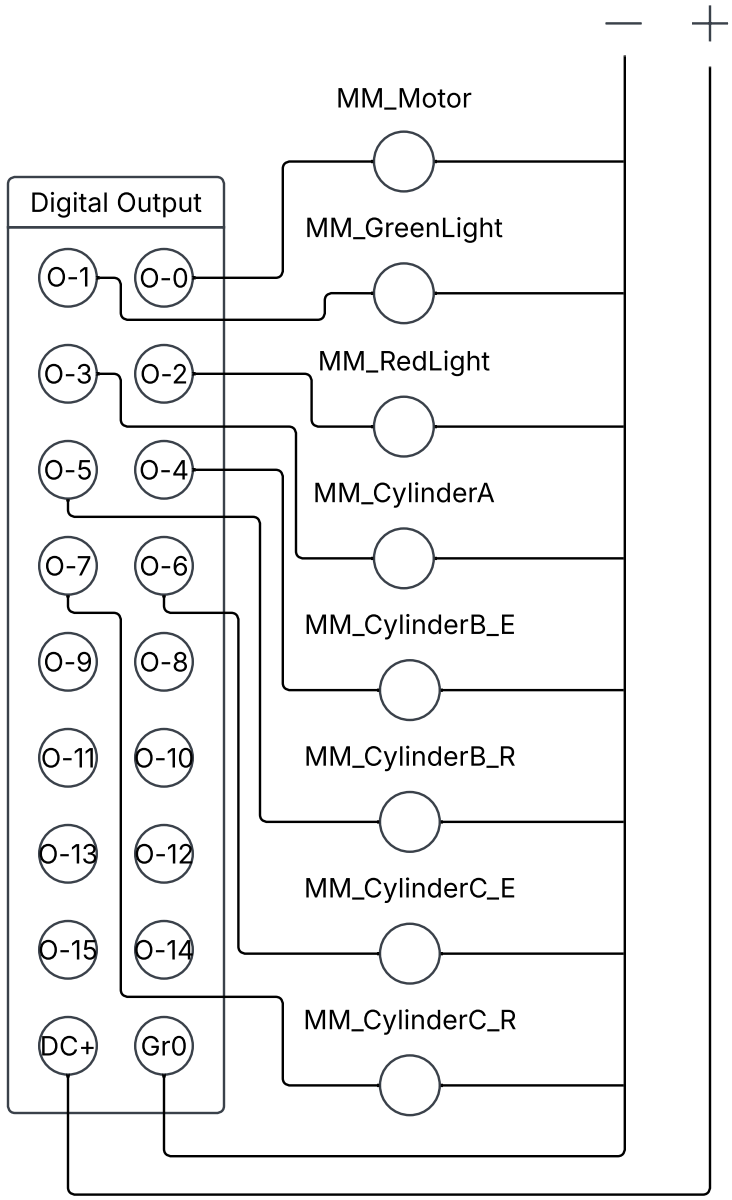
This lab successfully demonstrated the versatility and power of three key PLC programming languages:

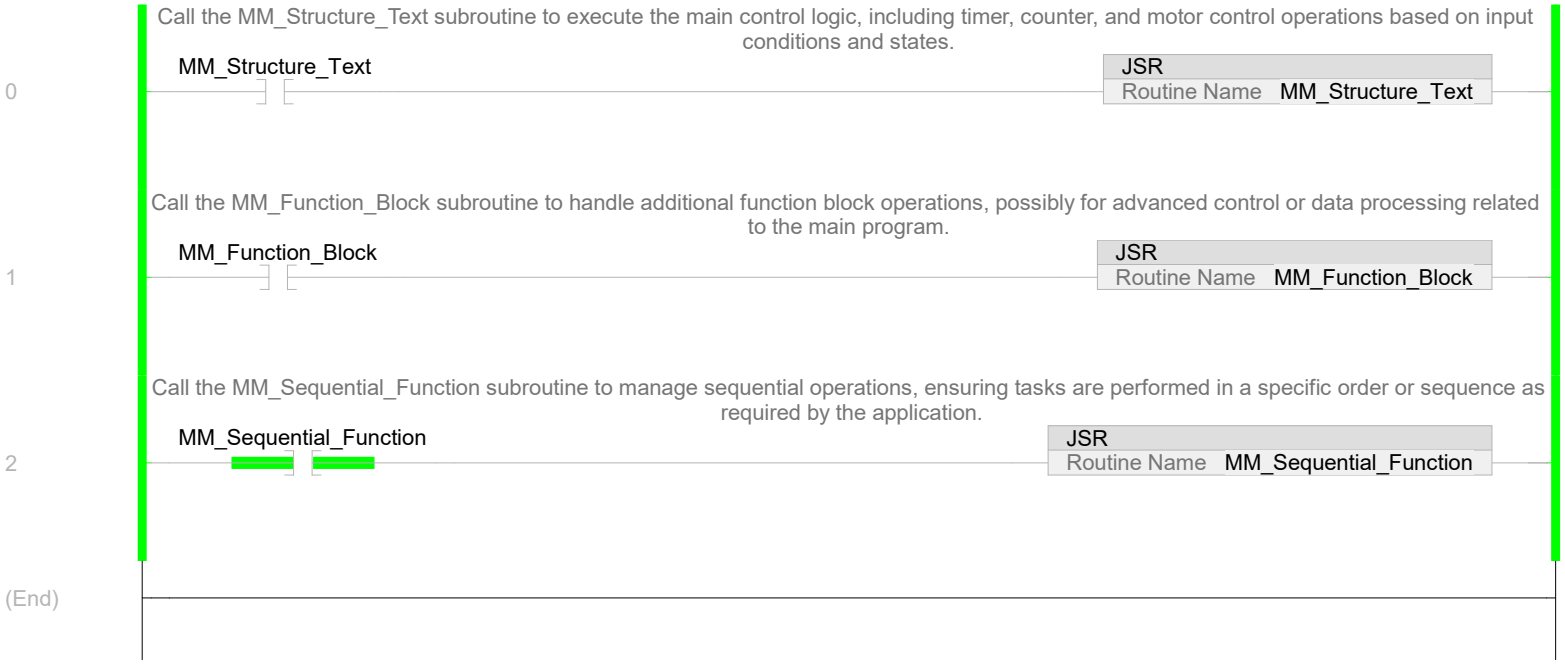
- ST offered compact, code-like control ideal for logic-heavy operations.
- FBD provided an intuitive graphical method for handling input conditions and modular operations.
- SFC allowed for high-level process visualization and step-based machine sequencing.

The motor and cylinder behaviors responded as expected across all platforms.

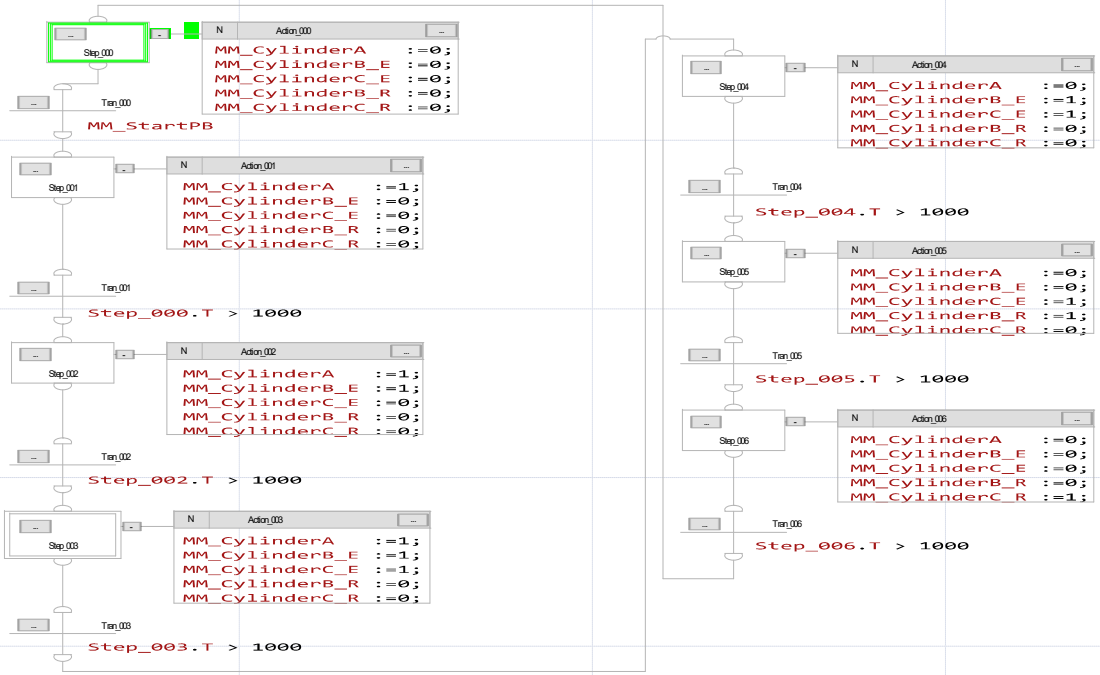
The lab helped reinforce the importance of structured control design, timing accuracy, and visual debugging in modern automation applications.


Digital Input
I-1
I-0
I-3
I-2
I-5
I-4
I-7
I-6
I-9
I-8
I-11
I-10
I-13
I-12
I-15
I-14
Gr0
MM_StartPB
MM_StopPB
MM_CyclePB
MM_CapacitiveSensor


Digital Output
O-1
O-0
O-3
O-2
O-5
O-4
O-7
O-6
O-9
O-8
O-11
O-10
O-13
O-12
O-15
O-14
DC+
Gr0
MM_Motor
MM_GreenLight
MM_RedLight
MM_CylinderA
MM_CylinderB_E
MM_CylinderB_R
MM_CylinderC_E
MM_CylinderC_R


Call the MM_Structure_Text subroutine to execute the main control logic, including timer, counter, and motor control operations based on input conditions and states.
MM_Structure_Text
JSR
Routine Name MM_Structure_Text
0
Call the MM_Function_Block subroutine to handle additional function block operations, possibly for advanced control or data processing related to the main program.
MM_Function_Block
JSR
Routine Name MM_Function_Block
1
Call the MM_Sequential_Function subroutine to manage sequential operations, ensuring tasks are performed in a specific order or sequence as required by the application.
MM_Sequential_Function
JSR
Routine Name MM_Sequential_Function
2
(End)


BOR
BOR_01
In1
In2
Out
MM_StartPB
MM_StopPB
BAND
BAND_01
In1
In2
Out
MM_Motor
MM_GreenLight
CTUD
CTUD_01
CUEnable
CDEnable
PRE
Reset
ACC
DN
MM_StartPB
MM_CapacitiveSensor
5
MM_CyclePB
BNOT
BNOT_01
In
Out
MM_Motor
MM_RedLight


Step_000
N
Action_000
MM_CylinderA :=0;
MM_CylinderB_E :=0;
MM_CylinderC_E :=0;
MM_CylinderB_R :=0;
MM_CylinderC_R :=0;
Tran_000
MM_StartPB
Step_001
Action_001
MM_CylinderA :=1;
MM_CylinderB_E :=0;
MM_CylinderC_E :=0;
MM_CylinderB_R :=0;
MM_CylinderC_R :=0;
Tran_001
Step_000.T > 1000
Step_002
Action_002
MM_CylinderA :=1;
MM_CylinderB_E :=1;
MM_CylinderC_E :=0;
MM_CylinderB_R :=0;
MM_CylinderC_R :=0;
Tran_002
Step_002.T > 1000
Step_003
Action_003
MM_CylinderA :=1;
MM_CylinderB_E :=1;
MM_CylinderC_E :=1;
MM_CylinderB_R :=0;
MM_CylinderC_R :=0;
Tran_003
Step_003.T > 1000

Step_004
N
Action_004
MM_CylinderA :=0;
MM_CylinderB_E :=1;
MM_CylinderC_E :=1;
MM_CylinderB_R :=0;
MM_CylinderC_R :=0;
Tran_004
Step_004.T > 1000
Step_005
Action_005
MM_CylinderA :=0;
MM_CylinderB_E :=0;
MM_CylinderC_E :=1;
MM_CylinderB_R :=1;
MM_CylinderC_R :=0;
Tran_005
Step_005.T > 1000
Step_006
Action_006
MM_CylinderA :=0;
MM_CylinderB_E :=0;
MM_CylinderC_E :=0;
MM_CylinderB_R :=0;
MM_CylinderC_R :=1;
Tran_006
Step_006.T > 1000

```
1  TONR (MM_Timer);
2  MM_Timer.PRE := 10000;
3
4  CTUD(MM_Count);
5  MM_Count.PRE := 3;
6
7  OSRI (MM_OneShot);
8  MM_OneShot.InputBit := MM_StartPB;
9
10 if MM_Timer.DN then
11     MM_Count.Reset := 0;
12     MM_Count.CUEnable := 1;
13 else
14     MM_Count.CUEnable := 0;
15 end_if;
16
17 if MM_CyclePB then
18     MM_Count.Reset := 1;
19 end_if;
20
21 MM_Run := (MM_OneShot.OutputBit OR MM_Run) AND MM_StopPB AND NOT MM_Timer.DN AND NOT MM_Count.DN;
22
23 if MM_Run then
24     MM_Timer.TimerEnable := 1;
25     MM_Motor := 1;
26     MM_GreenLight := 1;
27     MM_RedLight :=0;
28 else
29     MM_Motor := 0;
30     MM_Timer.TimerEnable :=0;
31     MM_RedLight := 1;
32     MM_GreenLight := 0;
33 end_if;
```