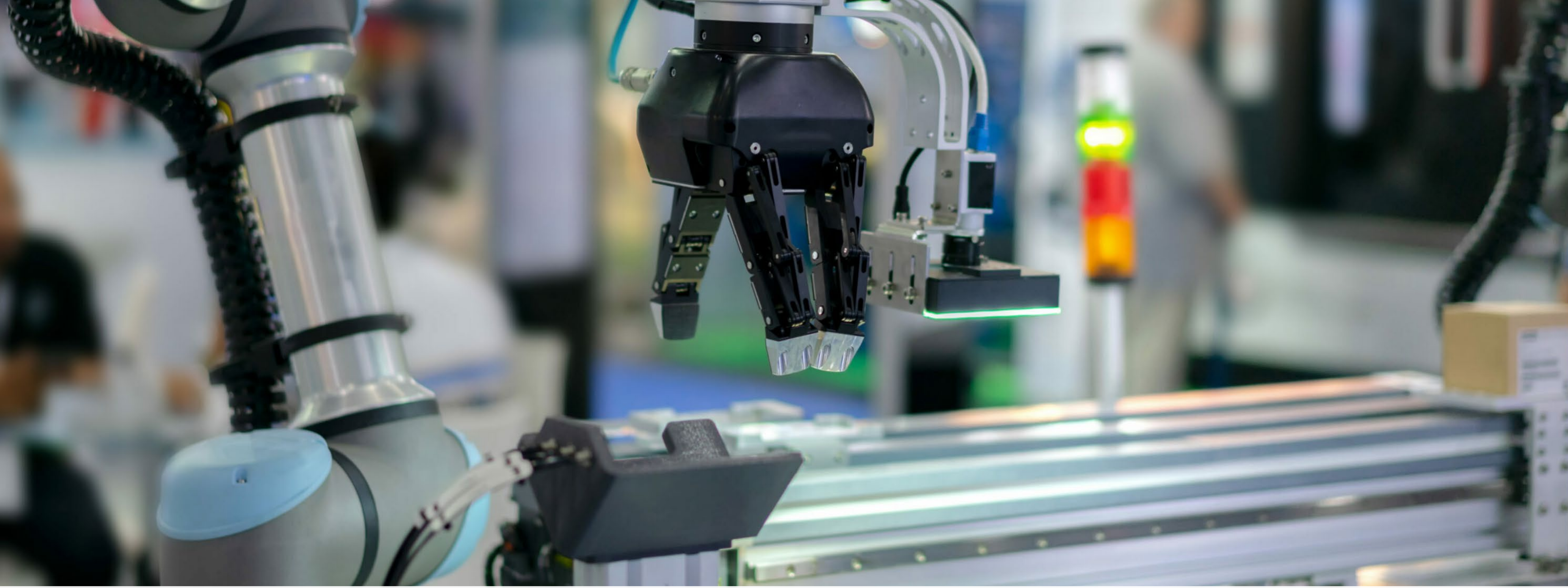


ISE 589-006: INTRODUCTION TO MODERN INDUSTRIAL AUTOMATION

LECTURE 006
Fred Livingston, PhD

LECTURE 006

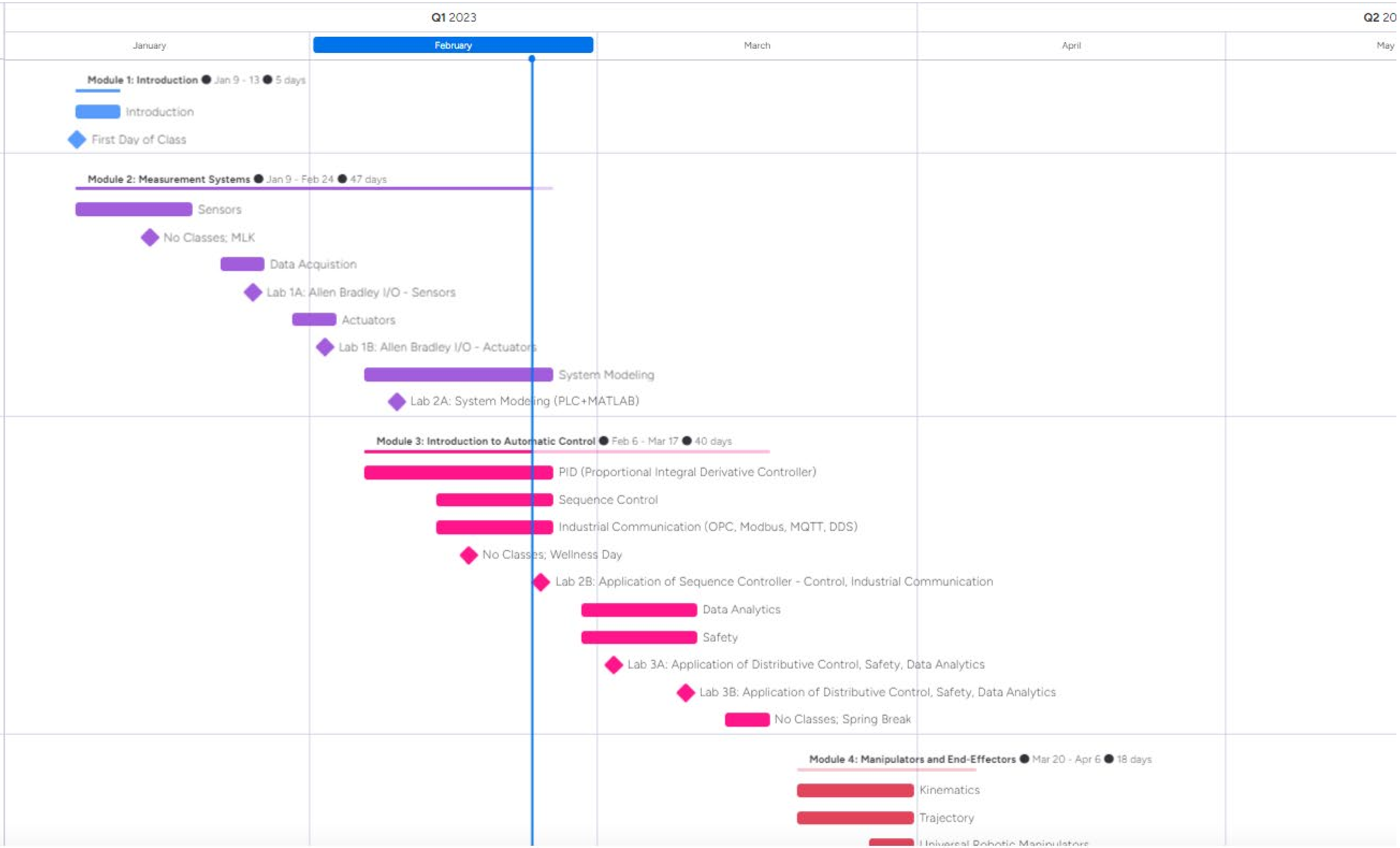
- Schedule Updates
- Modbus (Review)
- Sequential Programming
- Homework 2
- Laboratory 2
  - Open-Loop Speed Control of DC Motor
  - Remote Operation and Monitoring of Controller

Modern Industrial Automation

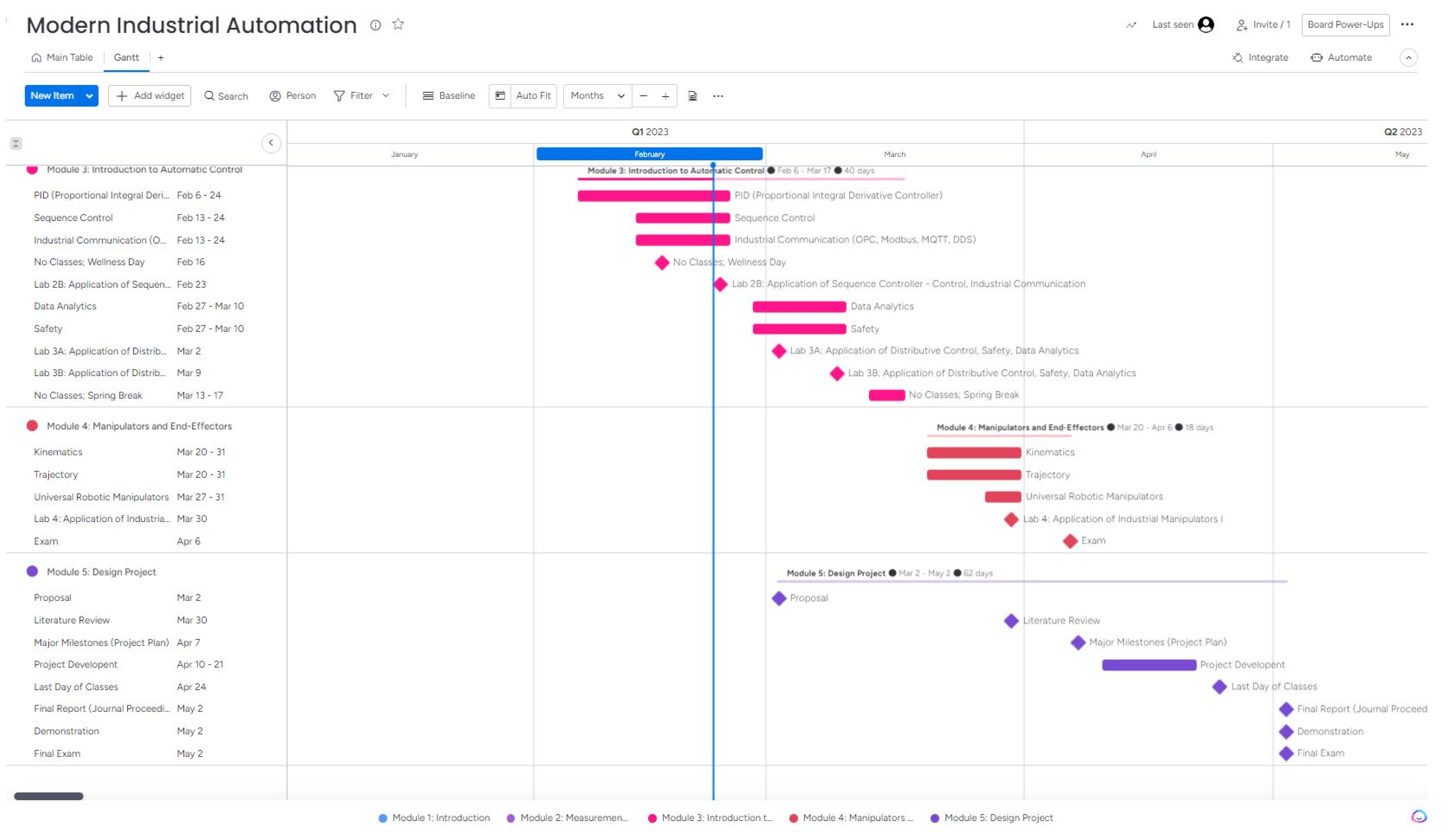
Main Table | Gantt | +

New Item | Add widget | Search | Person | Filter | Baseline | Auto Fit | Months

Module 1: Introduction	
Introduction	Jan 9 - 13
First Day of Class	Jan 9
Module 2: Measurement Systems	
Sensors	Jan 9 - 20
No Classes; MLK	Jan 16
Data Acquistion	Jan 23 - 27
Lab 1A: Allen Bradley I/O - Se...	Jan 26
Actuators	Jan 30 - Feb 3
Lab 1B: Allen Bradley I/O - Act...	Feb 2
System Modeling	Feb 6 - 24
Lab 2A: System Modeling (PL...	Feb 9
Module 3: Introduction to Automatic Control	
PID (Proportional Integral Deri...	Feb 6 - 24
Sequence Control	Feb 13 - 24
Industrial Communication (O...	Feb 13 - 24
No Classes; Wellness Day	Feb 16
Lab 2B: Application of Sequen...	Feb 23
Data Analytics	Feb 27 - Mar 10
Safety	Feb 27 - Mar 10
Lab 3A: Application of Distrib...	Mar 2
Lab 3B: Application of Distrib...	Mar 9
No Classes; Spring Break	Mar 13 - 17
Module 4: Manipulators and End-Effectors	
Kinematics	Mar 20 - 31
Trajectory	Mar 20 - 31



Module 1: Introduction | Module 2: Measuremen... | Module 3: Introduction t... | Module 4: Manipulators ... | Module 5: Design Project



Modern Industrial Automation
Last seen
Invite / 1
Board Power-Ups
Main Table
Gantt
Integrate
Automate
New Item
Add widget
Search
Person
Filter
Baseline
Auto Fit
Months
Q1 2023
Q2 2023
January
February
March
April
May
Module 3: Introduction to Automatic Control
PID (Proportional Integral Deri... Feb 6 - 24
Sequence Control Feb 13 - 24
Industrial Communication (O... Feb 13 - 24
No Classes; Wellness Day Feb 16
Lab 2B: Application of Sequen... Feb 23
Data Analytics Feb 27 - Mar 10
Safety Feb 27 - Mar 10
Lab 3A: Application of Distrib... Mar 2
Lab 3B: Application of Distrib... Mar 9
No Classes; Spring Break Mar 13 - 17
Module 4: Manipulators and End-Effectors
Kinematics Mar 20 - 31
Trajectory Mar 20 - 31
Universal Robotic Manipulators Mar 27 - 31
Lab 4: Application of Industria... Mar 30
Exam Apr 6
Module 5: Design Project
Proposal Mar 2
Literature Review Mar 30
Major Milestones (Project Plan) Apr 7
Project Developent Apr 10 - 21
Last Day of Classes Apr 24
Final Report (Journal Proceedi... May 2
Demonstration May 2
Final Exam May 2
Module 3: Introduction to Automatic Control Feb 6 - Mar 17 40 days
PID (Proportional Integral Derivative Controller)
Sequence Control
Industrial Communication (OPC, Modbus, MQTT, DDS)
No Classes; Wellness Day
Lab 2B: Application of Sequence Controller - Control, Industrial Communication
Data Analytics
Safety
Lab 3A: Application of Distributive Control, Safety, Data Analytics
Lab 3B: Application of Distributive Control, Safety, Data Analytics
No Classes; Spring Break
Module 4: Manipulators and End-Effectors Mar 20 - Apr 6 18 days
Kinematics
Trajectory
Universal Robotic Manipulators
Lab 4: Application of Industrial Manipulators I
Exam
Module 5: Design Project Mar 2 - May 2 62 days
Proposal
Literature Review
Major Milestones (Project Plan)
Project Developent
Last Day of Classes
Final Report (Journal Proceed
Demonstration
Final Exam
Module 1: Introduction
Module 2: Measuremen...
Module 3: Introduction t...
Module 4: Manipulators ...
Module 5: Design Project

UPCOMING MILESTONES

- Projects Descriptions (Rubrics, etc) – March 2nd
- Literature Review – March 30th
- Exam – April 6th

INDUSTRIAL COMMUNICATION PROTOCOL

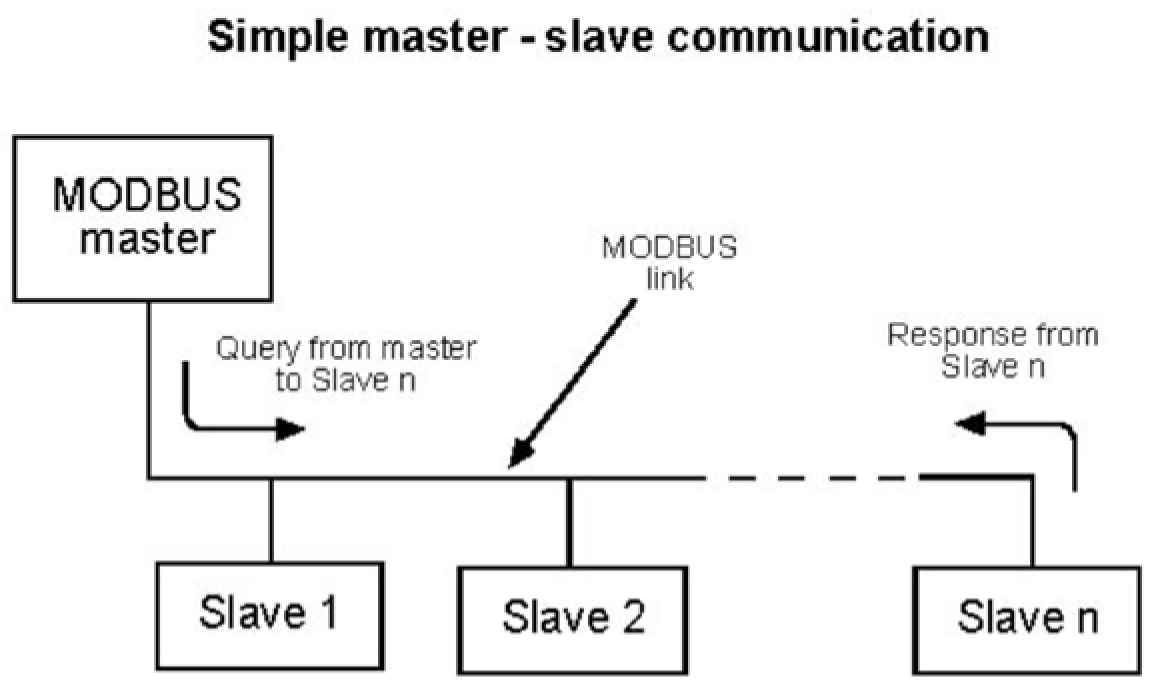
Desired to use high-computing device, such as MATLAB, to control automation process

INDUSTRIAL COMMUNICATION PROTOCOLS

Historically, many industrial components have been connected through different serial fieldbus protocols such as Control Area Network (CAN), **Modbus**®, PROFIBUS® and CC-Link. In recent years, industrial Ethernet has gained popularity, becoming more ubiquitous and offering higher speed, increased connection distance, and the ability to connect more nodes. There are many different industrial Ethernet protocols driven by various industrial equipment manufacturers. These protocols include Ether-CAT®, PROFINET®, EtherNet/IP™, and Sercos® III, among others.

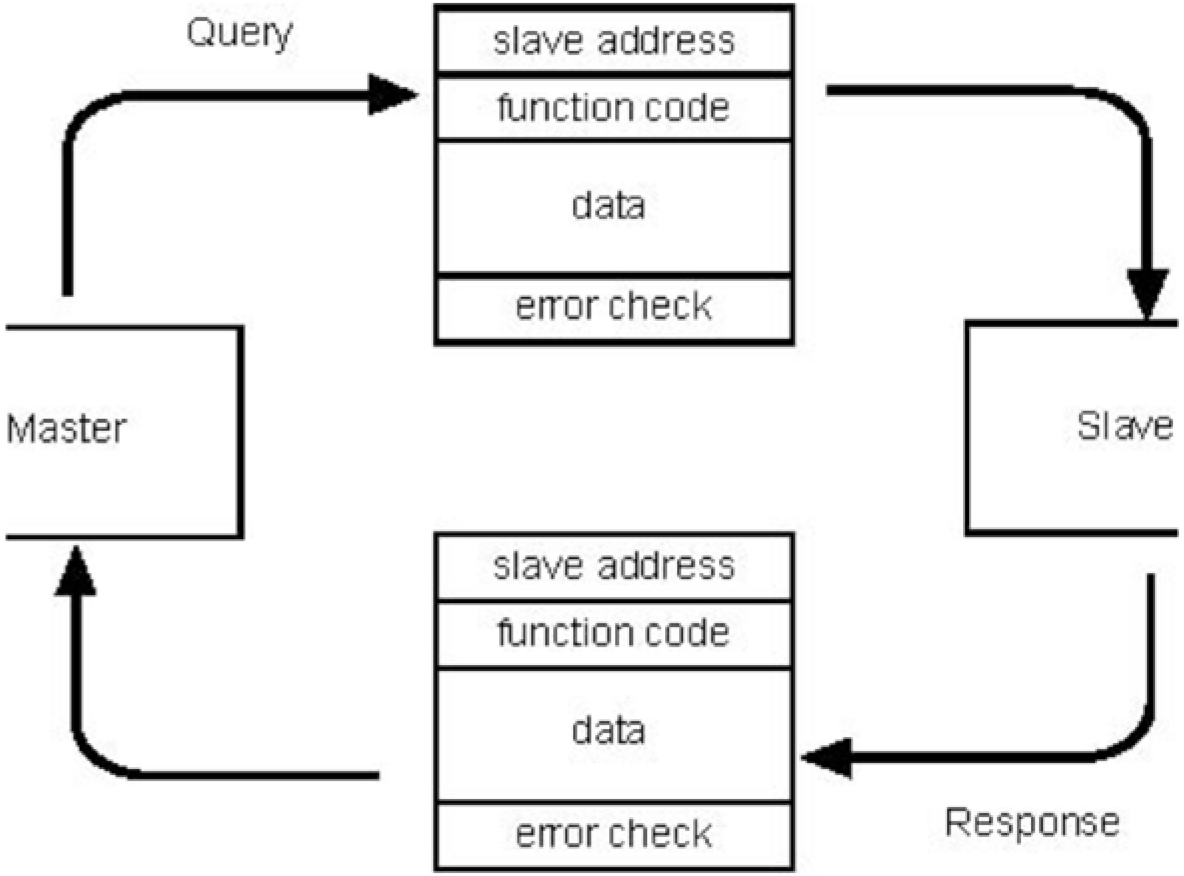
MODBUS TRANSACTIONS

Modbus controllers communicate using a master-slave technique, in which only one device (the master) can initiate a communication sequence.

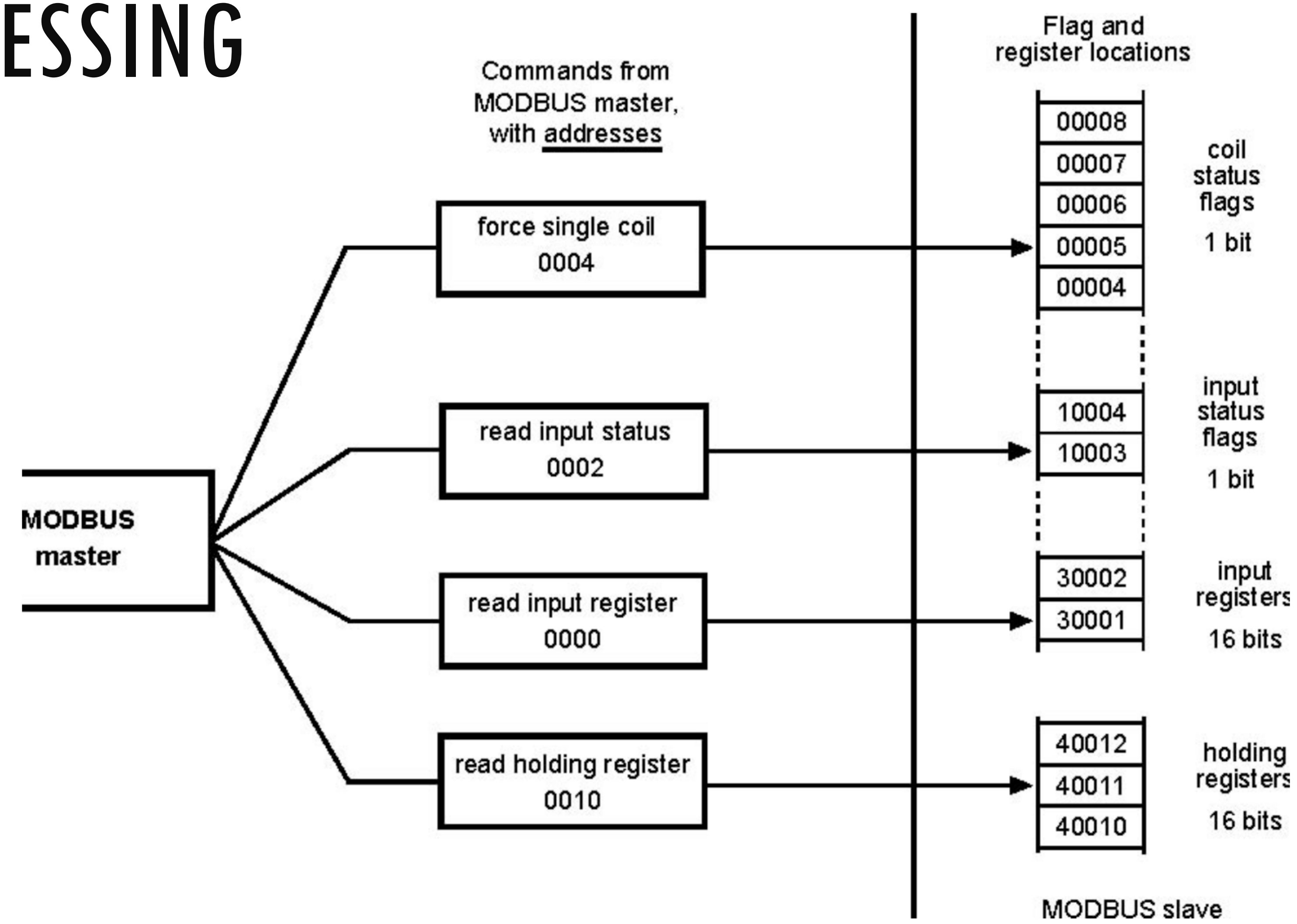


THE QUERY-RESPONSE CYCLE

The query-response cycle forms the basis of all communication on a Modbus network. In all situations it is the master that initiates the query and the slave that responds.

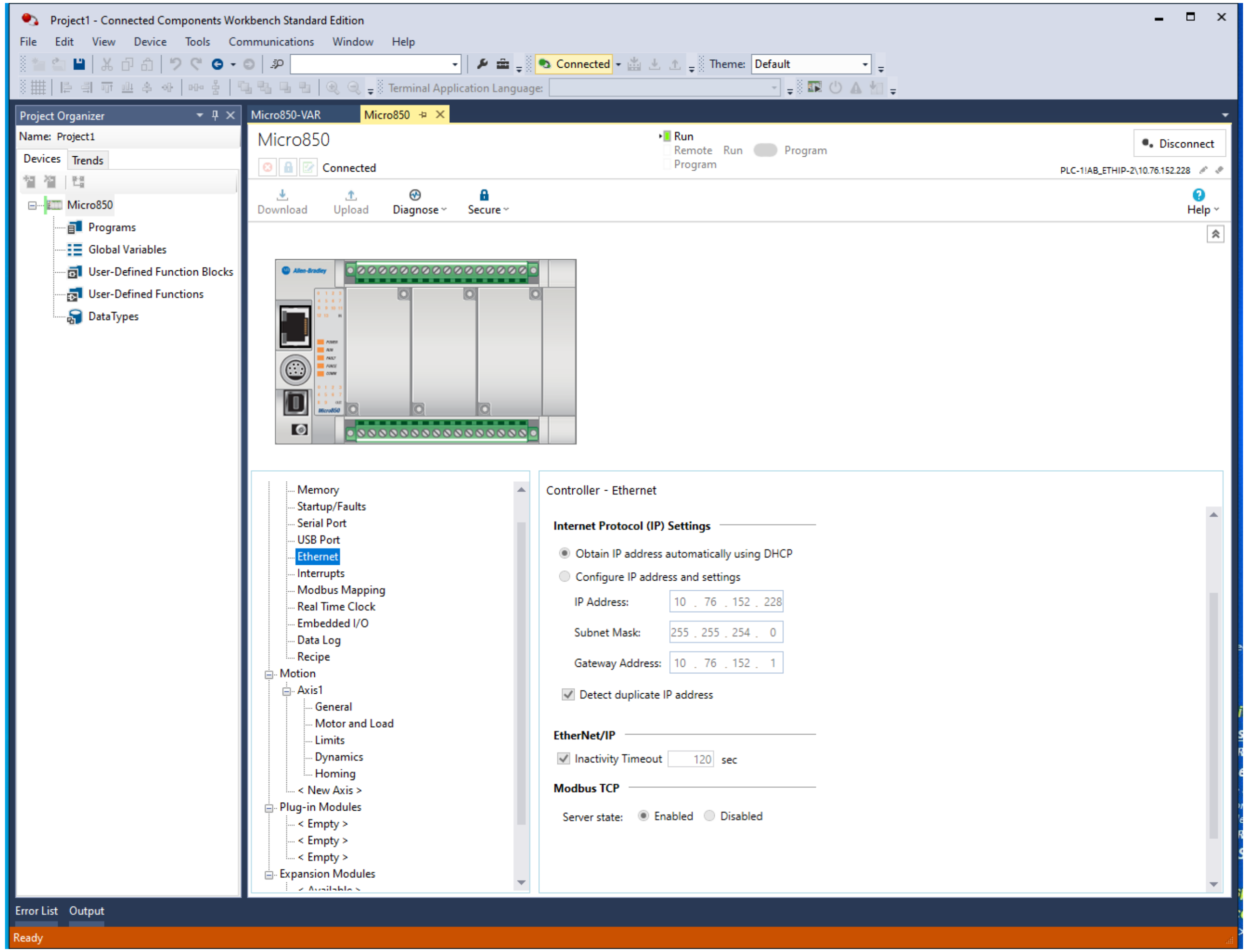


REGISTER AND FLAG ORGANIZATION AND ADDRESSING



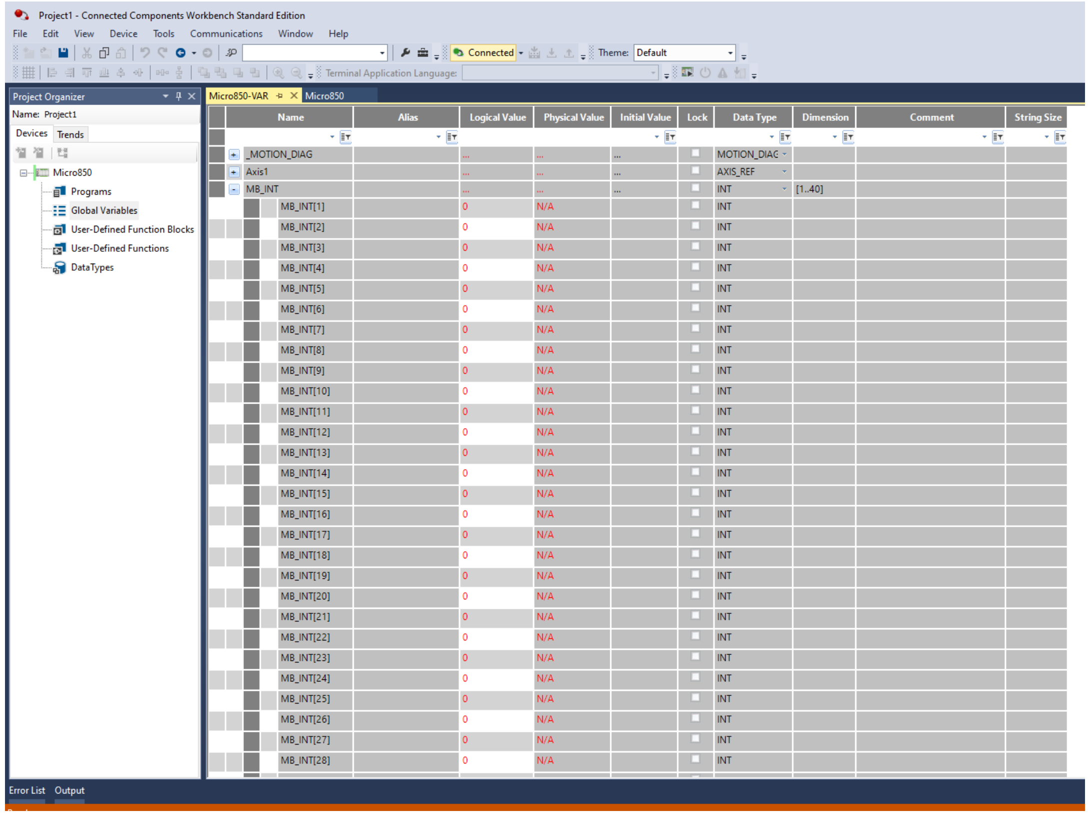
MICRO 850 (2080-LC50-24QBB)

Enable Modbus Server



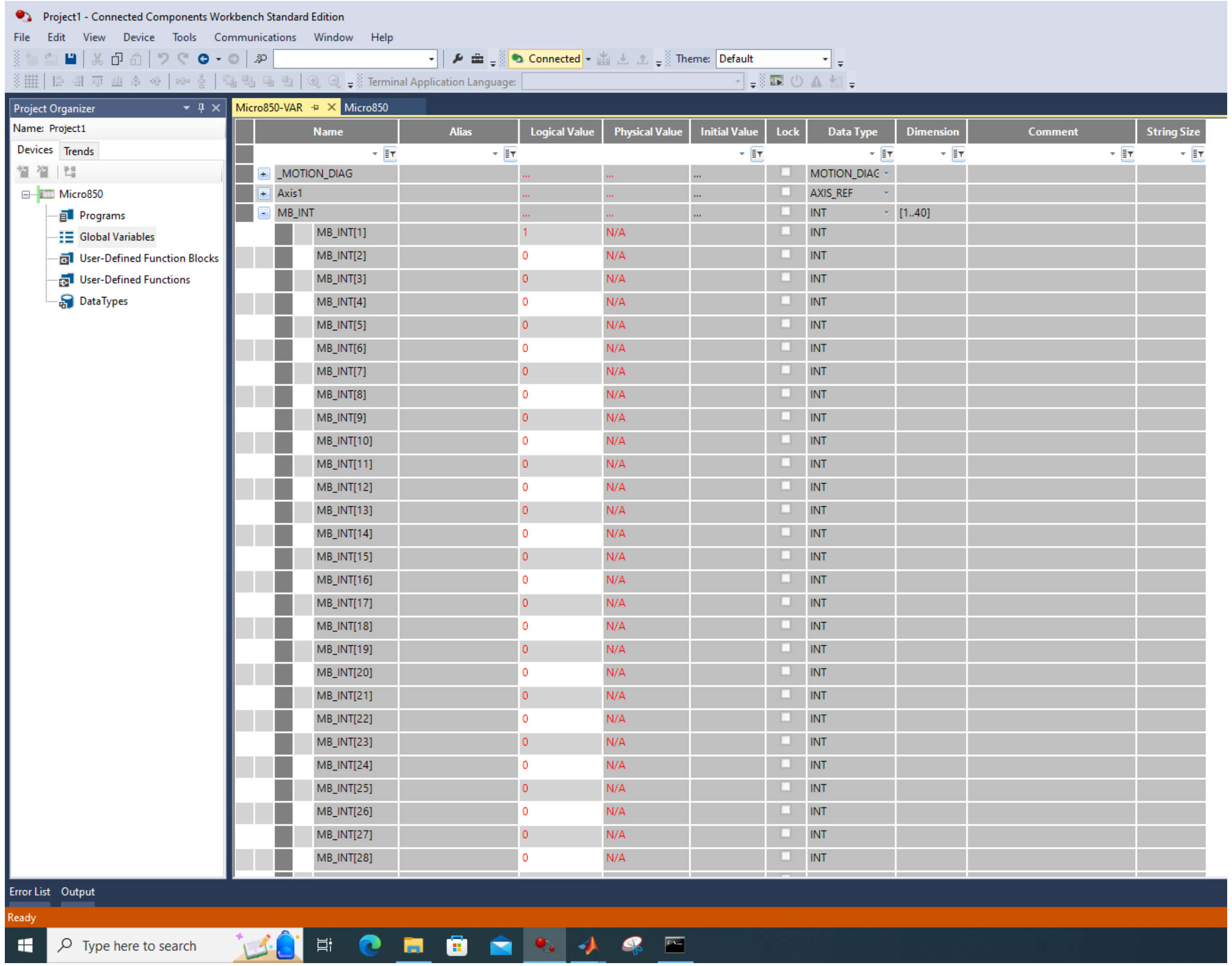
HOLDING REGISTERS

MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007
	...	...
	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046
	...	...
		...
	MB_REAL.30	HDR_40099
		HDR_40100



HOLDING REGISTERS

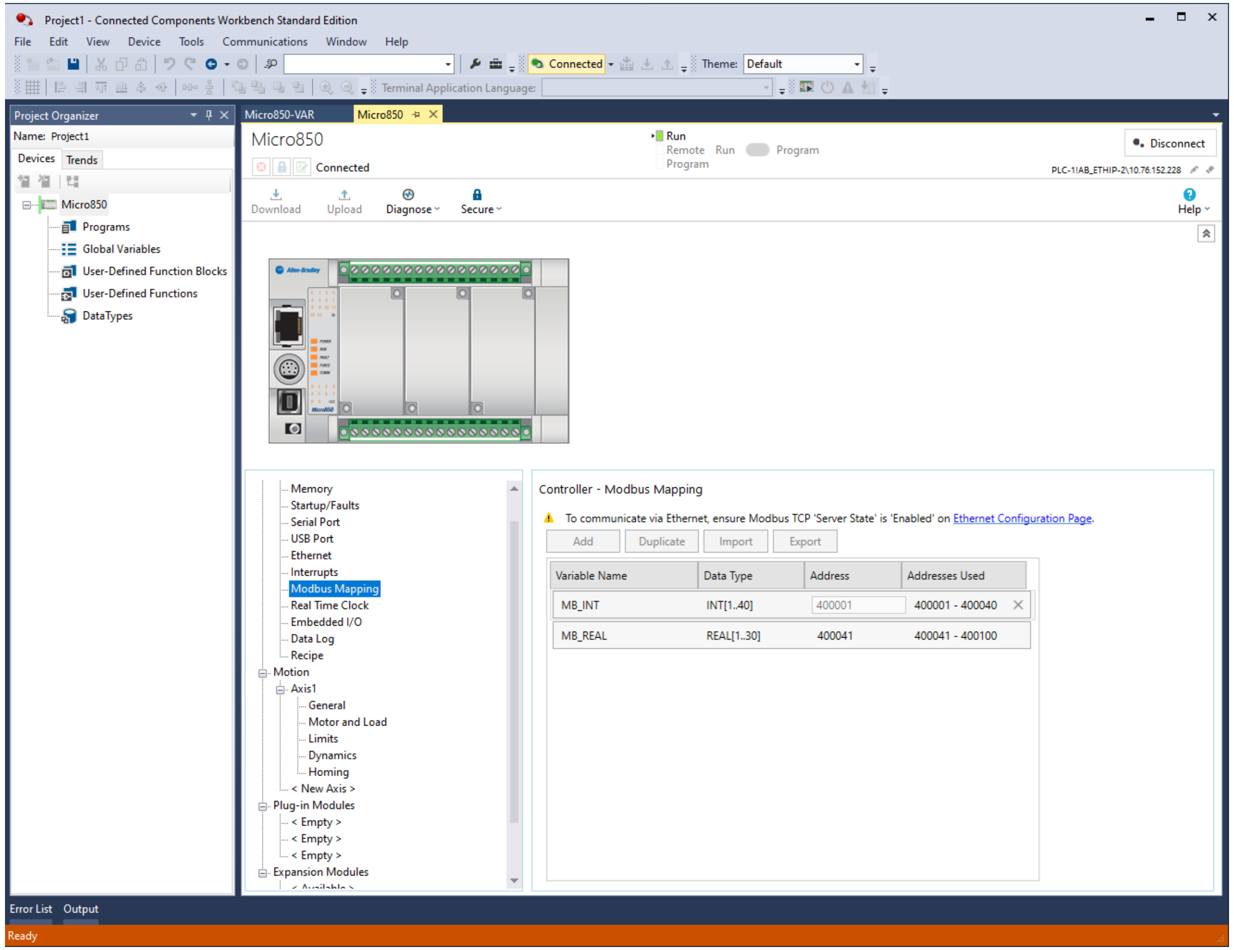
MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007
	...	...
	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046
	...	...
		...
	MB_REAL.30	HDR_40099
		HDR_40100



MICRO 850 (2080-LC50-24QBB)

Modbus Holding Registers

MICRO850 PLC		MODBUS
ARRAYS	ELEMENT	HOLDING REGISTERS
MB_INT[1..40]	MB_INT.1	HDR_40001
	MB_INT.2	HDR_40002
	MB_INT.3	HDR_40003
	MB_INT.4	HDR_40004
	MB_INT.5	HDR_40005
	MB_INT.6	HDR_40006
	MB_INT.7	HDR_40007
	...	...
	MB_INT.39	HDR_40039
	MB_INT.40	HDR_40040
MB_REAL[1..30]	MB_REAL.1	HDR_40041
		HDR_40042
	MB_REAL.2	HDR_40043
		HDR_40044
	MB_REAL.3	HDR_40045
		HDR_40046
	...	...
		...
	MB_REAL.30	HDR_40099
		HDR_40100



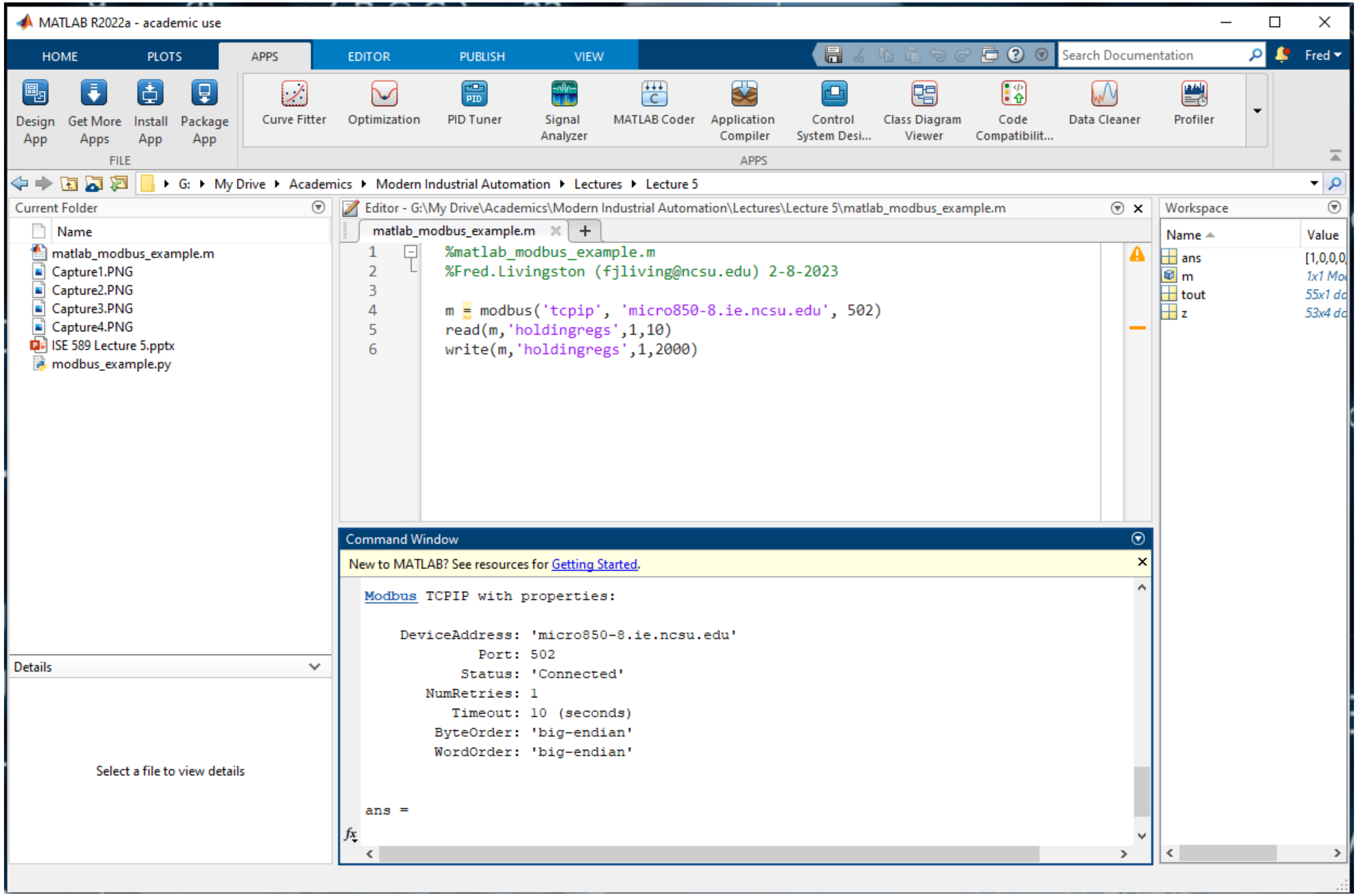
MODBUS CLIENTS

- Python Example
- MATLAB Example

MODBUS PYTHON EXAMPLE

modbus_example.py

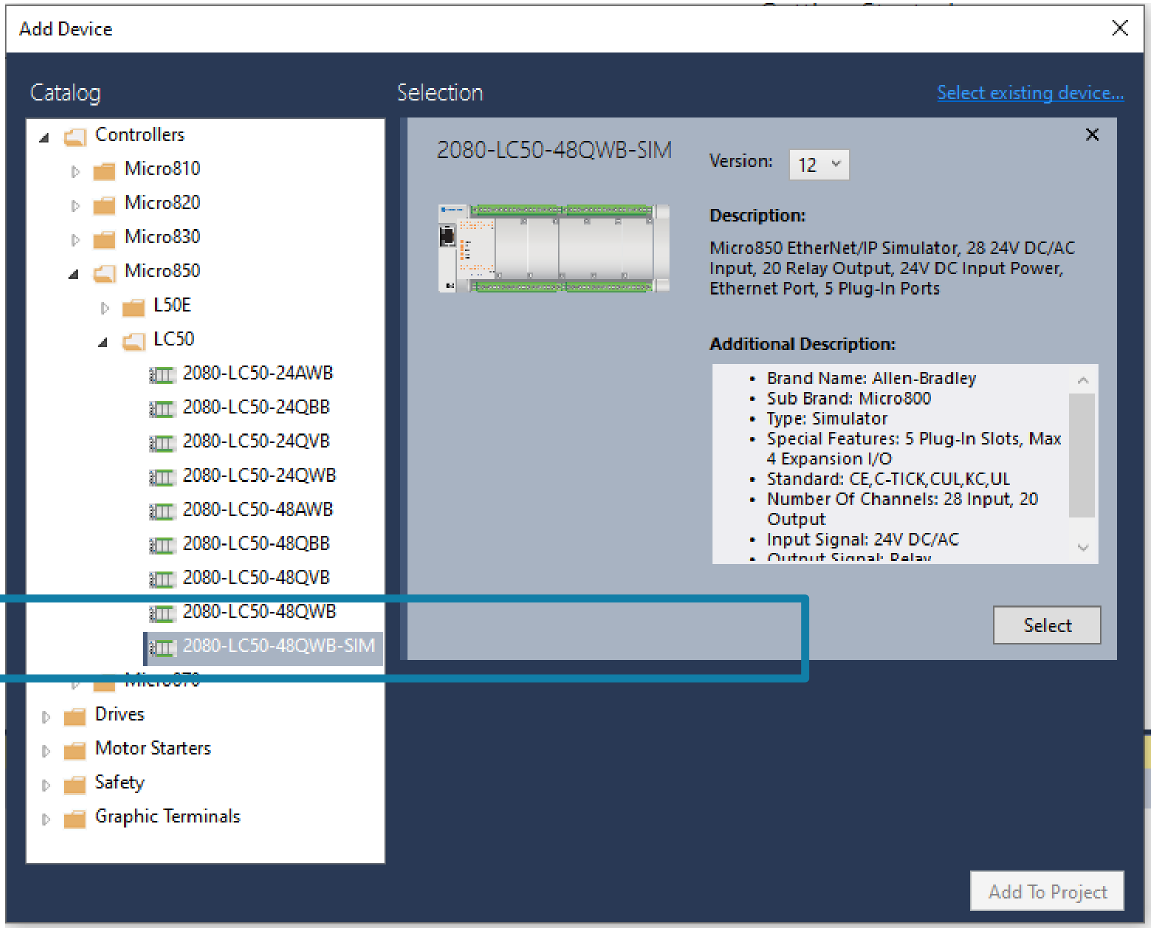
```python
#!/usr/bin/env python3
# modbus_example.py
# Fred Livingston (fjliving@ncsu.edu) 2-8-2023

from pymodbus.constants import [redacted]
from pymodbus.payload import BinaryPayloadDecoder
from pymodbus.payload import BinaryPayloadBuilder
from pymodbus.client import ModbusTcpClient

client = ModbusTcpClient('micro850-8.ie.ncsu.edu')
client.connect()

# Read from Holding Registers
request = client.read_holding_registers(1,1)
result = request.registers
decoder = BinaryPayloadDecoder.fromRegisters(result, Endian.Big, wordorder=Endian.Big)

#    'string': decoder.decode_string(8),
#    'float': decoder.decode_32bit_float(),
#    '16uint': decoder.decode_16bit_uint(),
#    'ignored': decoder.skip_bytes(2),
#    '8int': decoder.decode_8bit_int(),
#    'bits': decoder.decode_bits(),
print("Value: %d" % decoder.decode_16bit_uint())


# Write to Hold Registers
client.write_registers(1,2000)

client.close()
```


MATLAB R2022a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Fred
Design App
Get More Apps
Install App
Package App
FILE
Curve Fitter
Optimization
PID Tuner
Signal Analyzer
MATLAB Coder
Application Compiler
Control System Desi...
Class Diagram Viewer
Code Compatibilit...
Data Cleaner
Profiler
APPS
G: My Drive Academics Modern Industrial Automation Lectures Lecture 5
Current Folder
Name
matlab_modbus_example.m
Capture1.PNG
Capture2.PNG
Capture3.PNG
Capture4.PNG
ISE 589 Lecture 5.pptx
modbus_example.py
Details
Select a file to view details
Editor - G:\My Drive\Academics\Modern Industrial Automation\Lectures\Lecture 5\matlab_modbus_example.m
matlab_modbus_example.m
%matlab_modbus_example.m
%Fred.Livingston (fjliving@ncsu.edu) 2-8-2023
m = modbus('tcpip', 'micro850-8.ie.ncsu.edu', 502)
read(m,'holdingregs',1,10)
write(m,'holdingregs',1,2000)
Command Window
New to MATLAB? See resources for Getting Started.
Modbus TCPIP with properties:
DeviceAddress: 'micro850-8.ie.ncsu.edu'
Port: 502
Status: 'Connected'
NumRetries: 1
Timeout: 10 (seconds)
ByteOrder: 'big-endian'
WordOrder: 'big-endian'
ans =
Workspace
Name
Value
ans
m
tout
z

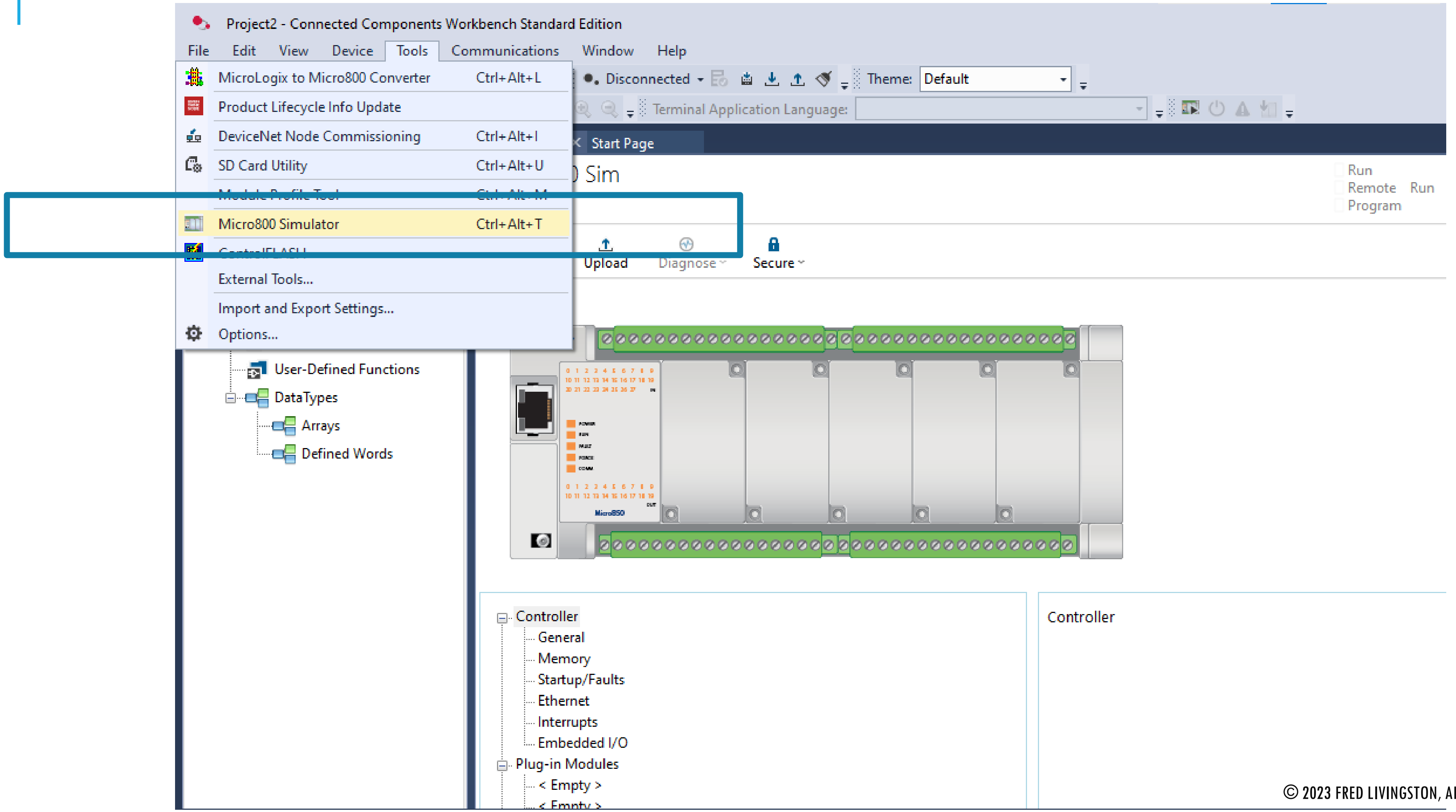
SEQUENTIAL PROGRAMMING

MICRO800 SIMULATOR

2080-LC50-48QWB-SIM

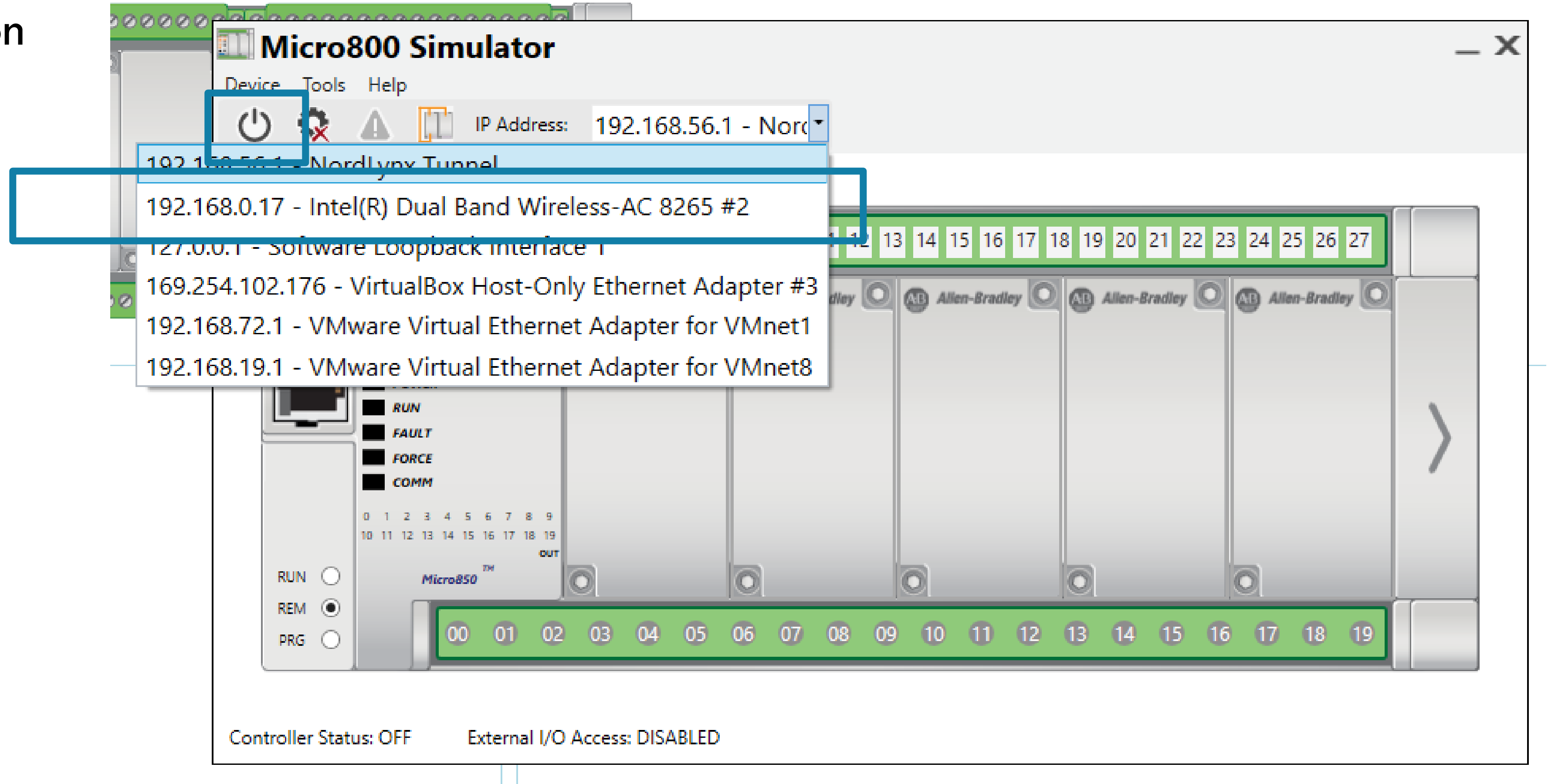


MICRO800 SIMULATOR



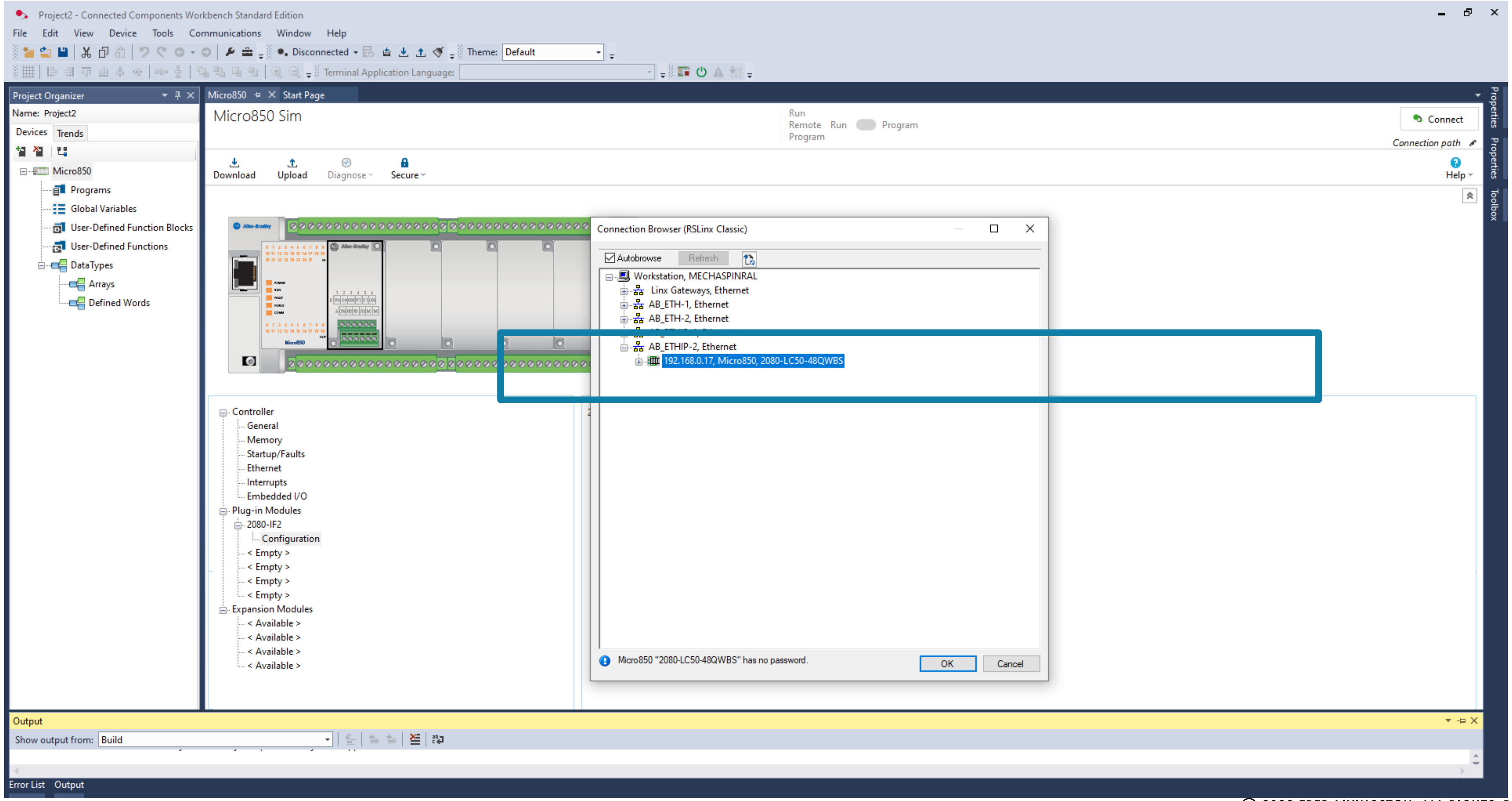
MICRO800 SIMULATOR

- Select correct network device
- Power on



MICRO800 SIMULATOR

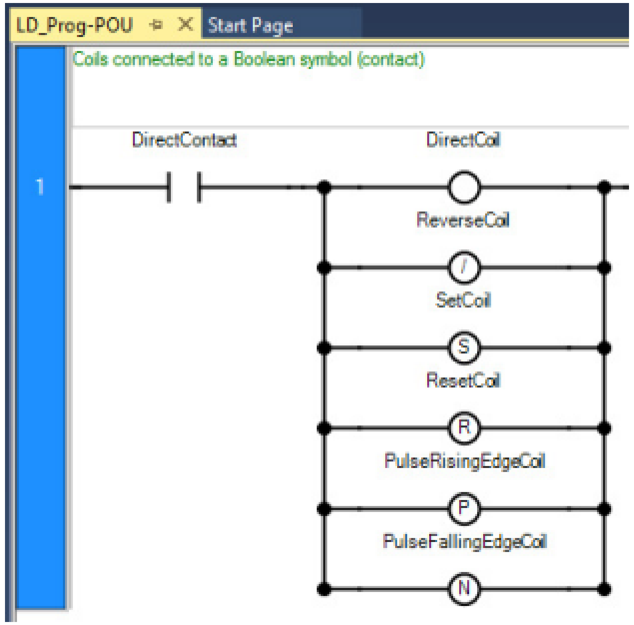
- Connect to Device



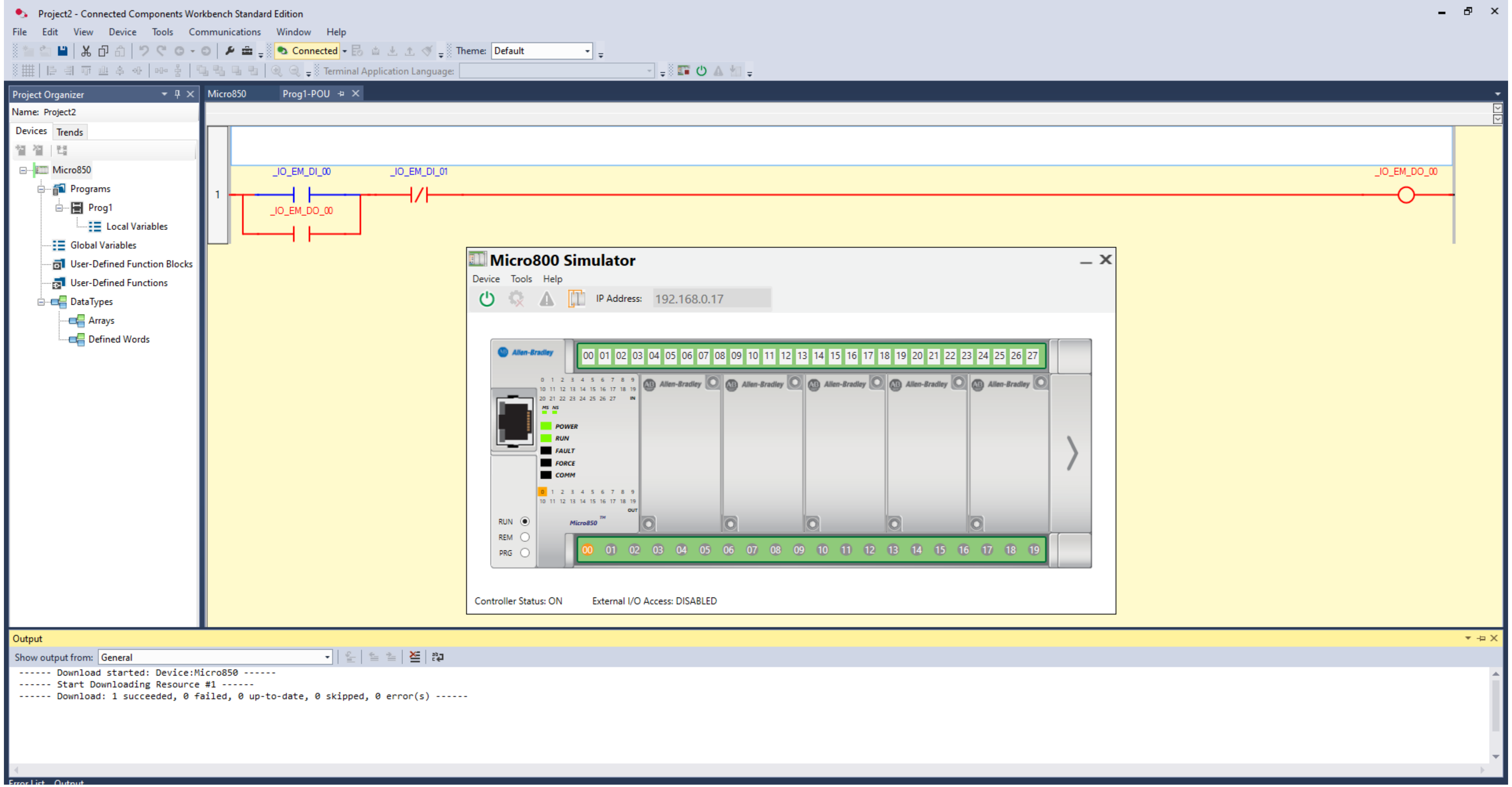
LADDER LOGIC – COILS

Coils are graphic components of Ladder Diagram (LD) programs that represent the assignment of an output or of an internal variable. In LD programs, a coil represents an action. A coil must be connected on the left to a Boolean symbol, such as a contact, or to a Boolean output of an instruction block. Coils can only be added to a defined rung in the LD language editor. The coil definition can be modified after the coil is added to the rung.

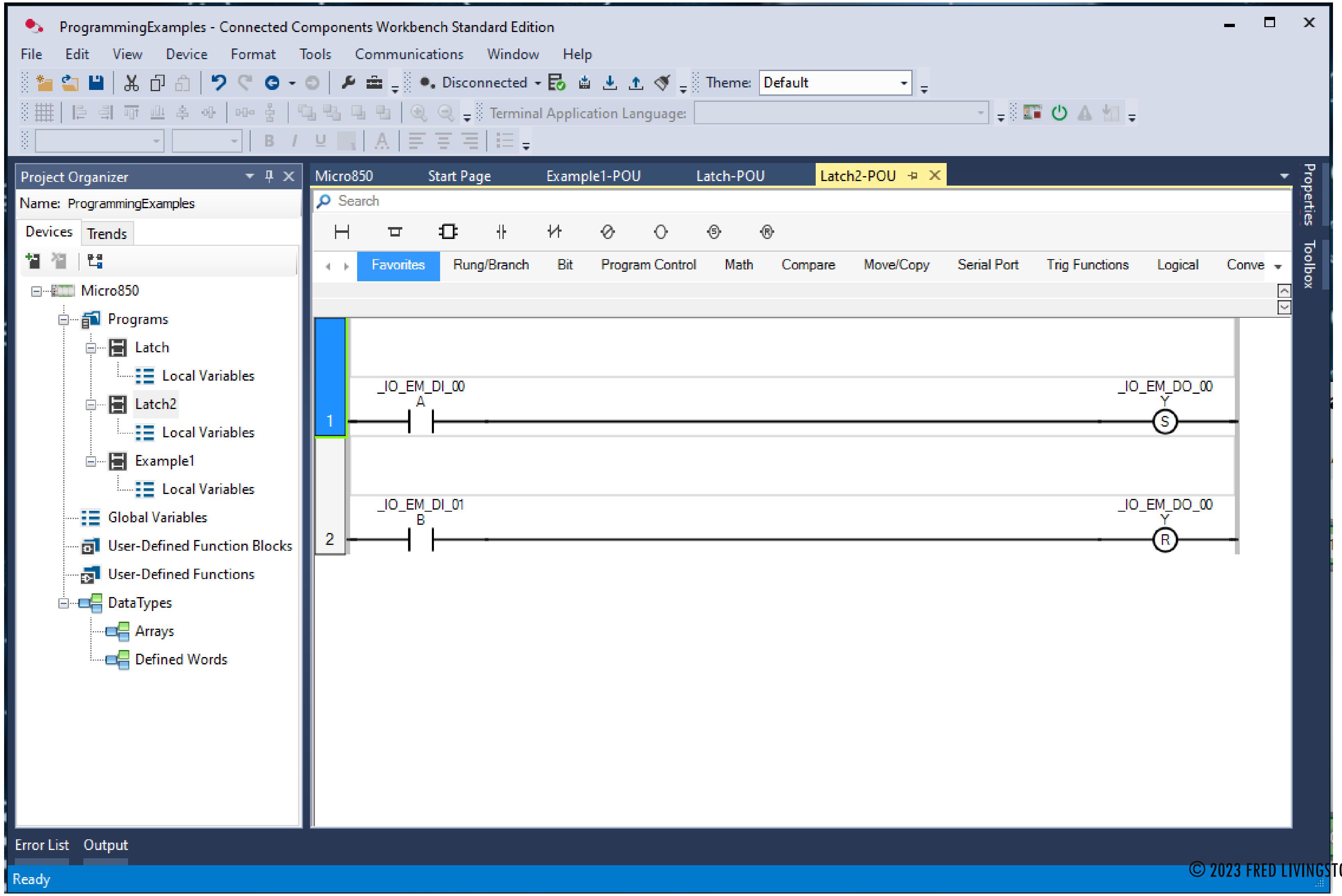
Example: Coils



PROGRAMMING EXAMPLE – LATCH1

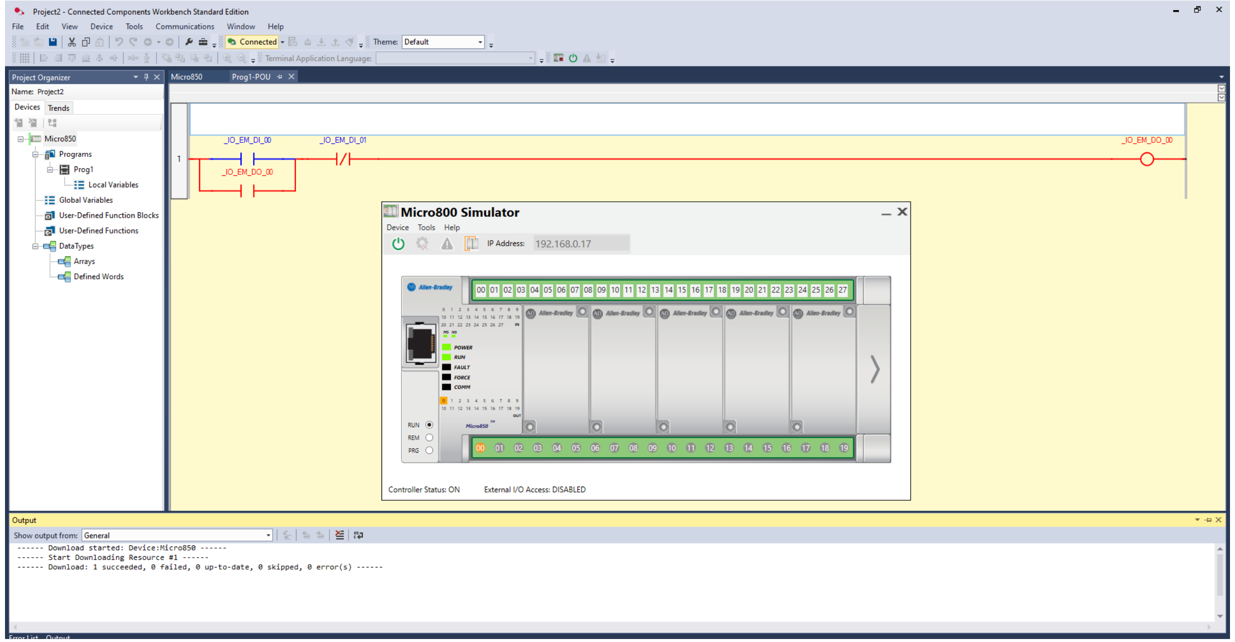


PROGRAMMING EXAMPLE – LATCH2



LOGIC CONTROL REPRESENTATION

$$Y = (A + \acute{Y}) \bullet \bar{B}$$



LOGIC CONTROL REPRESENTATION

$$Y = (\bar{A} \bullet B) + C$$

A	B	C	$\bar{A}$	$\bar{A} \bullet B$	$\bar{A} \bullet B + C$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

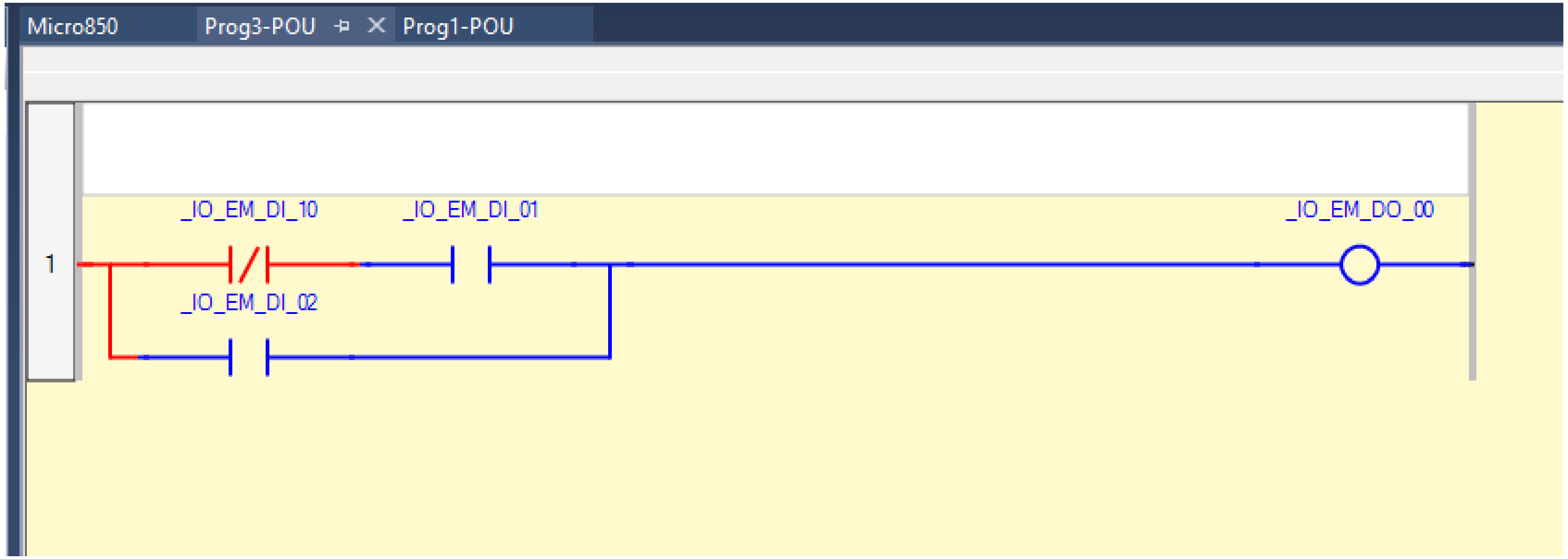
LOGIC CONTROL REPRESENTATION

$$Y = (\bar{A} \bullet B) + C$$

A	B	C	$\bar{A}$	$\bar{A} \bullet B$	$\bar{A} \bullet B + C$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

LOGIC CONTROL REPRESENTATION (LADDER LOGIC EXAMPE)

$$Y = (\bar{A} \bullet B) + C$$



PROGRAMMING – ARITHMETIC

Use the arithmetic instructions to perform mathematical calculations.

Function	Description
ABS on page 63	Returns the absolute value of a Real value.
ACOS on page 65	Calculates the arc-cosine of a Real value.
ACOS_LREAL on page 66	Calculates the arc-cosine of a Long Real value.
Addition on page 68	Adds two or more Integer, Real, Time, or String values.
ASIN on page 69	Calculates the arcsine of a Real value.
ASIN_LREAL on page 71	Calculates the arcsine of a Long Real value.
ATAN on page 72	Calculates the arctangent of a Real value.
ATAN_LREAL on page 74	Calculates the arctangent of a Long Real value.
COS on page 75	Calculates the cosine of a Real value.
COS_LREAL on page 77	Calculates the cosine of a Long Real value.
Division on page 78	Division of two Integer or Real values.
EXPT on page 80	Calculates the Real value of a base number raised to the power of the Integer exponent.
LOG on page 82	Calculates the logarithm (base 10) of a Real value.
MOD on page 83	Performs a Modulo calculation on Integer values.
MOV on page 85	Copies an input value to an output.
Multiplication on page 86	Multiplies two or more Integer or Real values.
Neg on page 88	Converts a value to a negative.
POW on page 89	Calculates the value of a Real number raised to a power of the Real exponent.
RAND on page 91	Calculates random integer values from a defined range.
SIN on page 93	Calculates the sine of a Real value.
SIN_LREAL on page 94	Calculates the sine of a Long Real value.
SQRT on page 96	Calculates the square root of a Real value.
Subtraction on page 97	Subtracts one Integer, Real or Time value from another Integer, Real or Time value.
TAN on page 99	Calculates the tangent of a Real value.
TAN_LREAL on page 100	Calculates the tangent of a Long Real value.
TRUNC on page 102	Truncates Real values, leaving just the Integer.

PROGRAMMING – BINARY INSTRUCTIONS

Use Binary instructions to perform mathematical operations.

Operator	Description
AND_MASK on page 125	Performs a bit-to-bit AND between two Integer values.
NOT_MASK on page 133	Integer bit-to-bit negation mask, inverts a parameter value.
BSL on page 126	Shifts a bit in an array element to the left.
BSR on page 130	Shifts a bit in an array element to the right.
OR_MASK on page 134	Integer OR bit-to-bit mask, turns bits on.
ROL on page 136	For 32-bit integers, rotates integer bits to the left.
ROR on page 137	For 32-bit integers, rotates integer bits to the left.
SHL on page 139	For 32-bit integers, moves integers to the left and places 0 in the least significant bit.
SHR on page 141	For 32-bit integers, moves integers to the right and places 0 in the most significant bit.
XOR_MASK on page 142	Integer exclusive OR bit-to-bit mask, returns inverted bit values.

PROGRAMMING – BOOLEAN INSTRUCTIONS

Use Boolean instructions to determine an output value based on a logical calculation from inputs. The module outputs can be directly controlled from the program or independently controlled by the module using the Boolean instructions

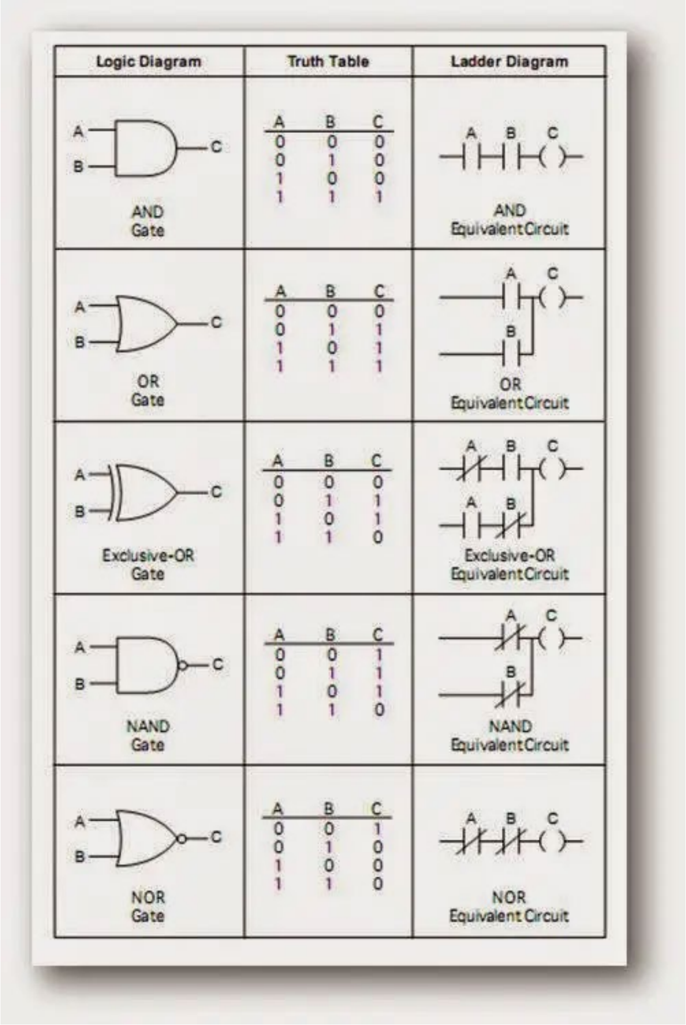
Function	Description
MUX4B on page 160	Multiplexer between four BOOL inputs, outputs a BOOL value.
MUX8B on page 156	Multiplexer between eight BOOL inputs, outputs a BOOL value.
TTABLE on page 153	Provides the value of the output based on the combination of inputs.
Function block	**Description**
F_TRIG on page 145	Detects a falling edge of a Boolean variable.
RS on page 148	Reset dominant (highest priority when determining instruction behavior) bistable.
R_TRIG on page 147	Detects a rising edge of a Boolean variable.
SR on page 152	Set dominant bistable.
Operator	**Description**
AND on page 150	Performs a boolean AND operation between two or more values.
NOT on page 151	Converts Boolean values to negated values.
XOR on page 151	Boolean exclusive OR of two values.
OR on page 149	Boolean OR of two or more values.

LOGIC CONTROL REPRESENTATION

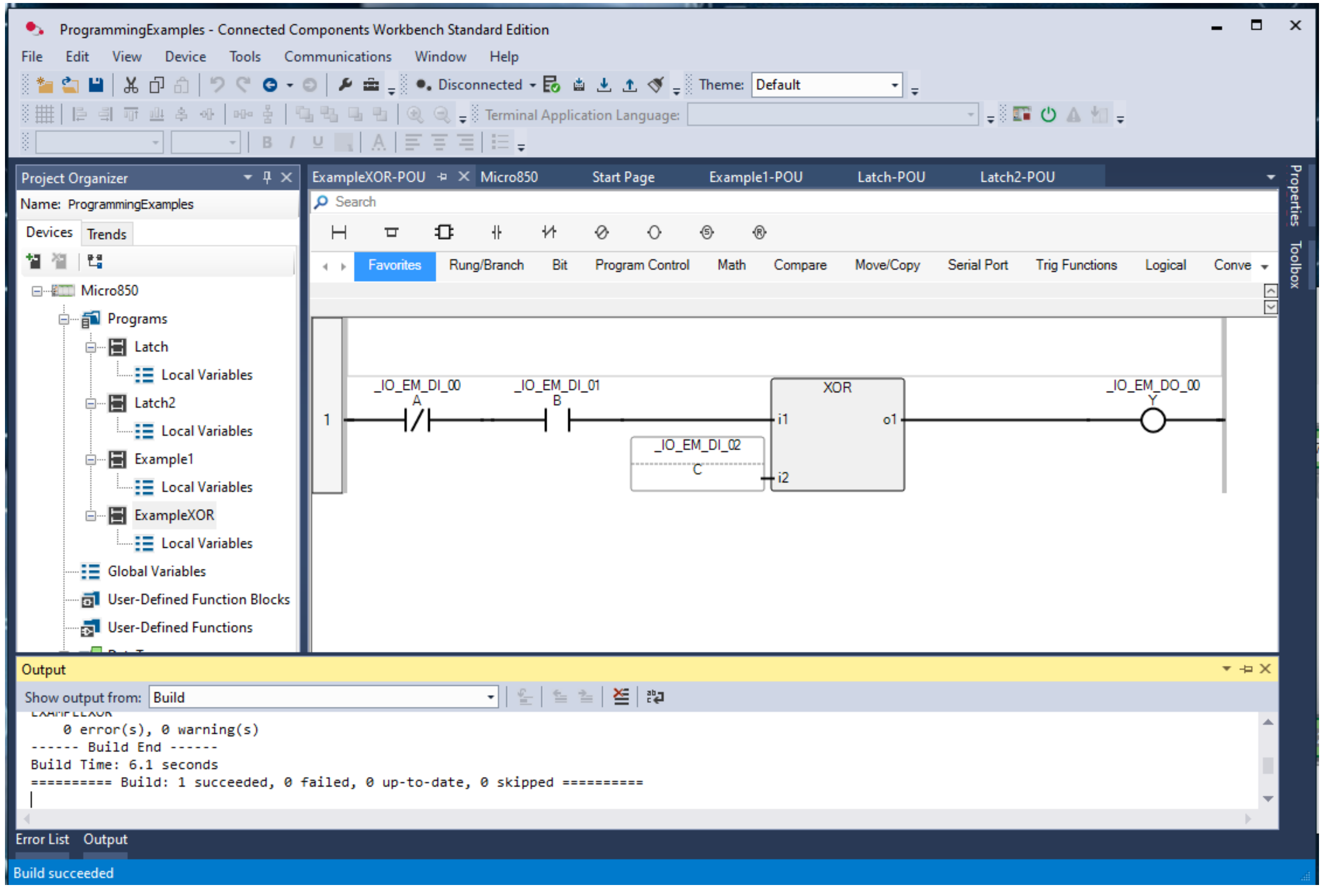
$$Y = (\bar{A} \bullet B) \oplus C$$

A	B	C	$\bar{A}$	$\bar{A} \bullet B$	$\bar{A} \bullet B \oplus C$
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

LOGIC CONTROL REPRESENTATION



$$Y = (\bar{A} \bullet B) \oplus C$$



PROGRAMMING – COMPARE INSTRUCTIONS

Use Compare instructions to compare Integer, Real, Time, Date, and String values using an expression or a specific compare instruction.

Instruction	Description
(=) Equal on page 219	Compares the first input to the second input to determine equality. For Integer, Real, Time, Date, and String data types.
(>) Greater Than on page 222	Compares input values to determine whether the first is greater than the second.
(>=) Greater Than or Equal on page 224	Compares input values to determine whether the first is greater than or equal to the second.
(<) Less Than on page 225	Compares input values to determine whether the first is less than the second.
(<=) Less Than or Equal on page 227	Compare input values to determine whether the first is less than or equal to the second.
(<>) Not Equal on page 228	Compares input values to determine whether the first is not equal to the second.

PROGRAMMING – DATA CONVERSIONS

Use Data conversion instructions to convert the data type of a variable to a different data type.

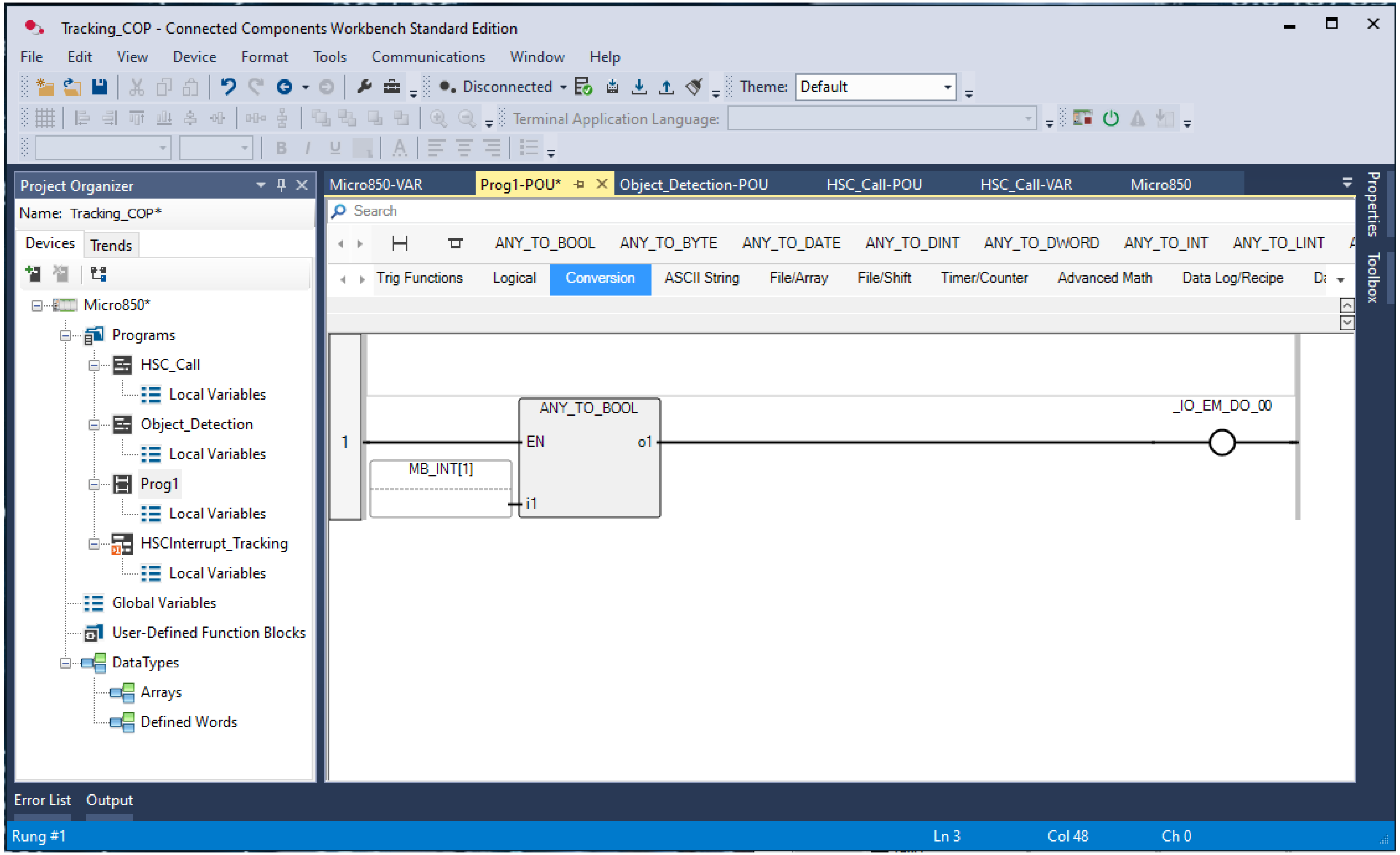
EN	Input	BOOL	Instruction enable. TRUE - execute the conversion to Boolean computation. FALSE - there is no computation. Applies to Ladder Diagram programs.
i1	Input	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Any non-Boolean value.
o1	Output	BOOL	Boolean value.

ANY_TO_BOOL Structured Text example

(* ST Equivalence: *)

```
ares := ANY_TO_BOOL (10);          (* ares is TRUE *)
tres := ANY_TO_BOOL (t#0s);        (* tres is FALSE *)
mres := ANY_TO_BOOL ('FALSE');     (* mres is FALSE *)
```


PROGRAMMING – DATA CONVERSIONS



PROGRAMMING – COUNTERS

Use Counter instructions to control operations based on the number of events.

Instruction	Description
CTD on page 231	Counts integers from a given value down to 0, 1 by 1.
CTU on page 233	Counts integers from 0 up to a given value, 1 by 1.
CTUD on page 235	Counts integers from 0 up to a given value, 1 by 1, or from a given value down to 0, 1 by 1.

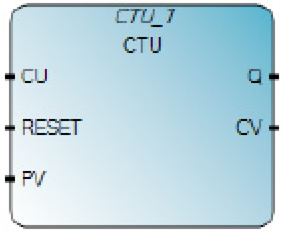
PROGRAMMING – COUNTERS

CTU (count up)

CTU counts (integers) from 0 up to a given value, 1 by 1.

Languages supported: Function Block Diagram, Ladder Diagram, Structured Text.

This instruction applies to the Micro810, Micro820, Micro830, Micro850, Micro870 controllers and Micro800 Simulator.

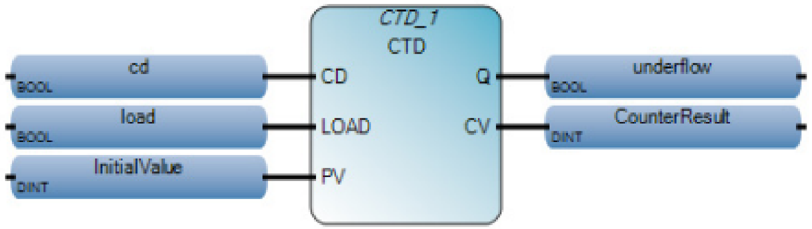


Use this table to help determine the parameter values for this instruction.

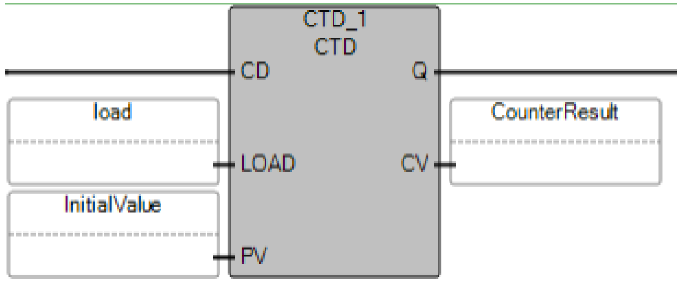
Parameter	Parameter Type	Data Type	Description
CU	Input	BOOL	Counts upward. TRUE - Rising edge detected, count upward in increments of one. FALSE - Falling edge detected, hold the counter value at the same value.
RESET	Input	BOOL	Reset verifies the PV value against the count upward value. TRUE - set the CV value to zero. FALSE - Continue incrementing count upward by one.
PV	Input	DINT	Programmed maximum value of the counter.
Q	Output	BOOL	Indicates whether the count up instruction has resulted in a number greater than or equal to the maximum value of the counter. TRUE - Counter result =>PV (Overflow condition). FALSE - Counter result < PV
CV	Output	DINT	Current counter result.

PROGRAMMING – COUNTERS

CTD Function Block Diagram example



CTD Ladder Diagram example



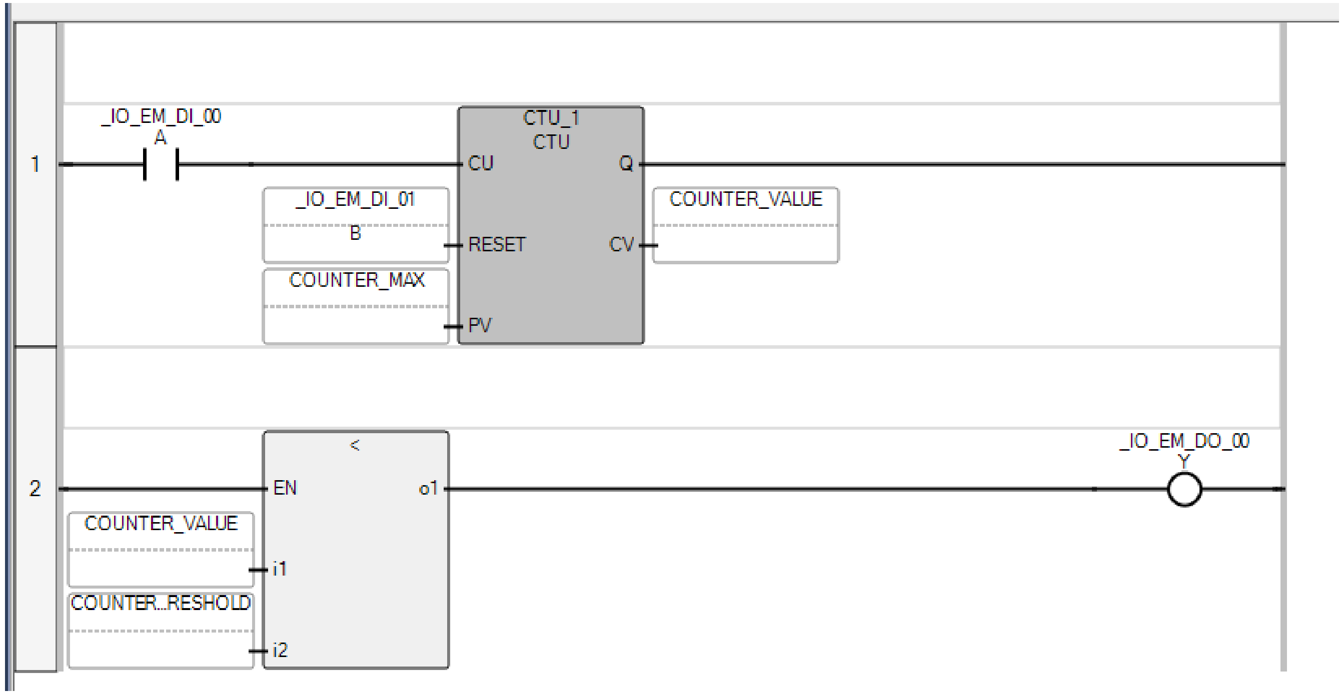
CTD Structured Text example


```
InitialValue := 10;
CTD_1(cd, load, InitialValue);
```

(*ST Equivalence: CTD1 is an instance of block *)

PROGRAMMING – COUNTERS

	Name	Alias	Data Type	Dimension	Project Value	Initial Value	Comment	String Size
	› COUNTER_MAX		DINT			10		
	› COUNTER_VALUE		DINT					
	› COUNTER_THRESHOLD		DINT			5		
	› CTU_1		CTU		...	...		
+	New..							



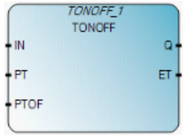
PROGRAMMING – TIMERS

Use Timer instructions to control operations based on time.

Instruction	Description
TOF on page 623	Off-delay timing. Increase an internal timer up to a given value.
TON on page 625	On-delay timing. Increase an internal timer up to a given value.
TONOFF on page 628	Delay turning on an output on a true rung and then delay turning off the output on the false rung.
TP on page 630	Pulse timing. On a rising edge, increases an internal timer up to a given value.
RTO on page 632	Retentive timing. Increases an internal timer when input is active but does not reset the internal timer when input changes to inactive.
DOY on page 634	Turn on an output if the value of the real-time clock is in the range of the Year Time setting.
TDF on page 636	Computes the time difference between TimeA and TimeB.
TOW on page 638	Turns on an output if the value of the real-time clock is in the range of the Time of Week setting.

PROGRAMMING – TIMERS

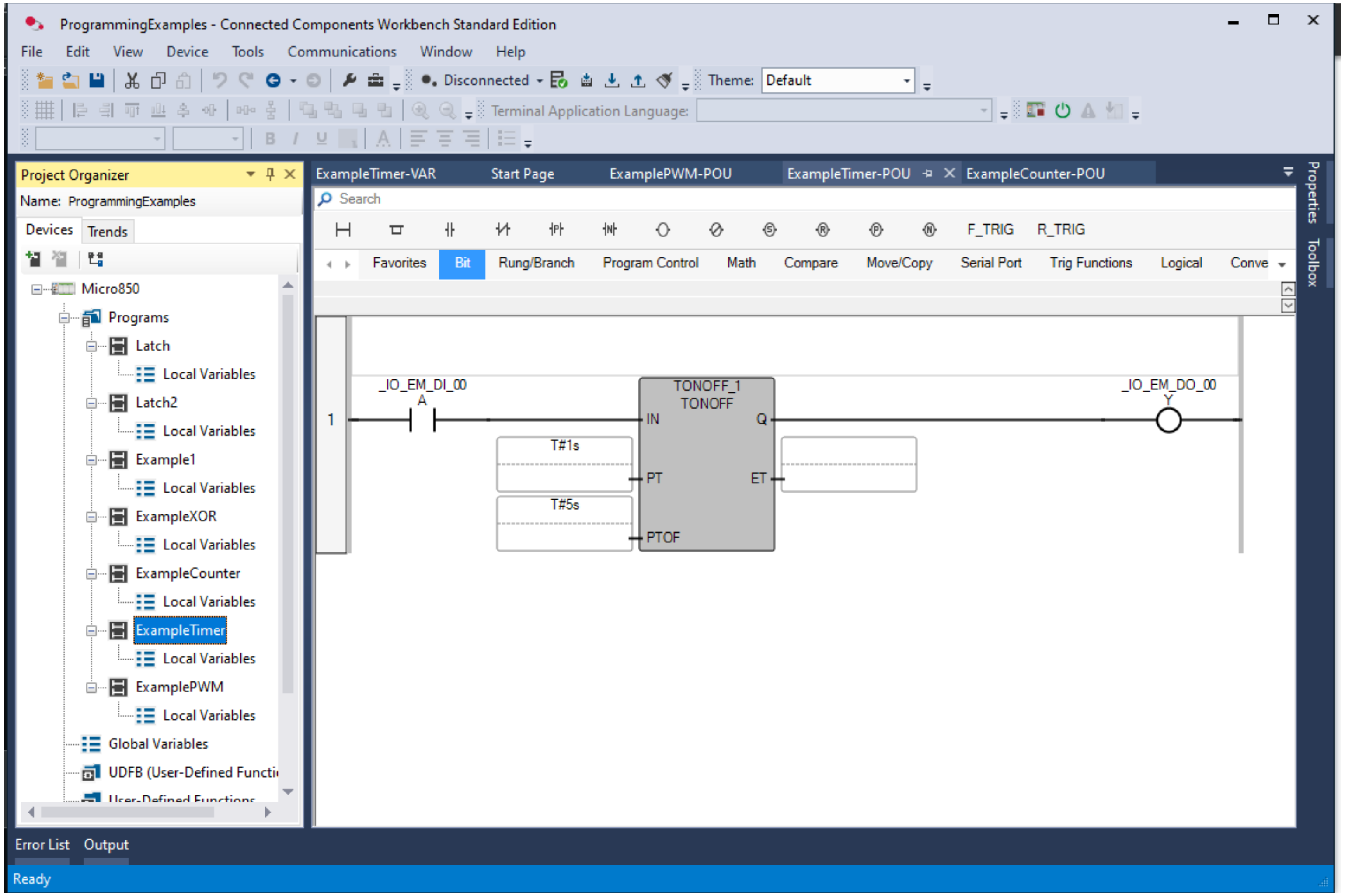
Use Timer instructions to control operations based on time.



Use this table to help determine the parameter values for this instruction.

Parameter	Parameter type	Data type	Description
IN	Input	BOOL	Input control. TRUE - Rising Edge detected (IN turns from 0 to 1): • start the On-delay timer (PT). • if Programmed Off--delay time (PTOF) is not elapsed, restart the On-delay (PT) timer. FALSE - Falling Edge detected (IN turns from 1 to 0): • if Programmed On-delay time (PT) is not elapsed, stop PT timer and reset ET. • if Programmed On-delay time (PT) is elapsed, the start the Off-delay timer (PTOF).
PT	Input	TIME	The on-delay time setting defined using the Time data type.
PTOF	Input	TIME	The off-delay time setting defined using the Time data type.
Q	Output	BOOL	TRUE - the Programmed On-delay time is elapsed and Programmed Off-delay time is not elapsed.
ET	Output	TIME	Current elapsed time. Possible values range from 0ms to 1193h2m47s294ms. If the Programmed On-delay time is elapsed and the Off-delay timer is not starting, the elapsed time (ET) remains at the on-delay (PT) value. If the Programmed Off-delay time is elapsed and the Off-delay timer is not starting, the elapsed time (ET) remains at the off-delay (PTOF) value until the rising edge occurs again.

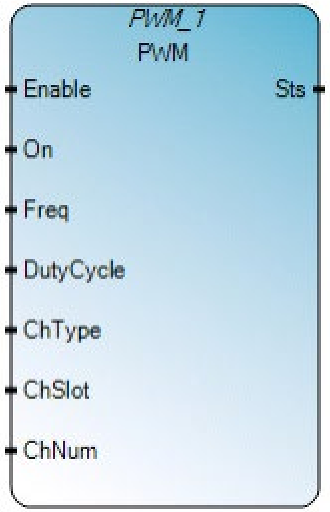
PROGRAMMING – TIMER EXAMPLE

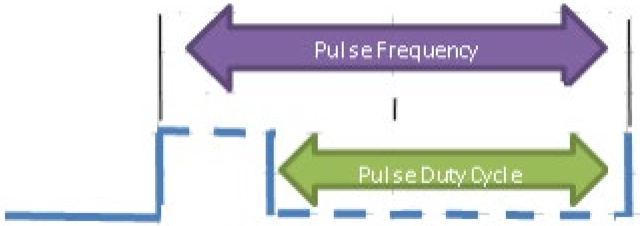


PROGRAMMING – PULSE WIDTH MODULATION

Turns the PWM (Pulse Width Modulation) output for a configured PWM channel ON or OFF. (Micro820 2080-LC20-20QBB)

This instruction applies only to the Micro820 controller.



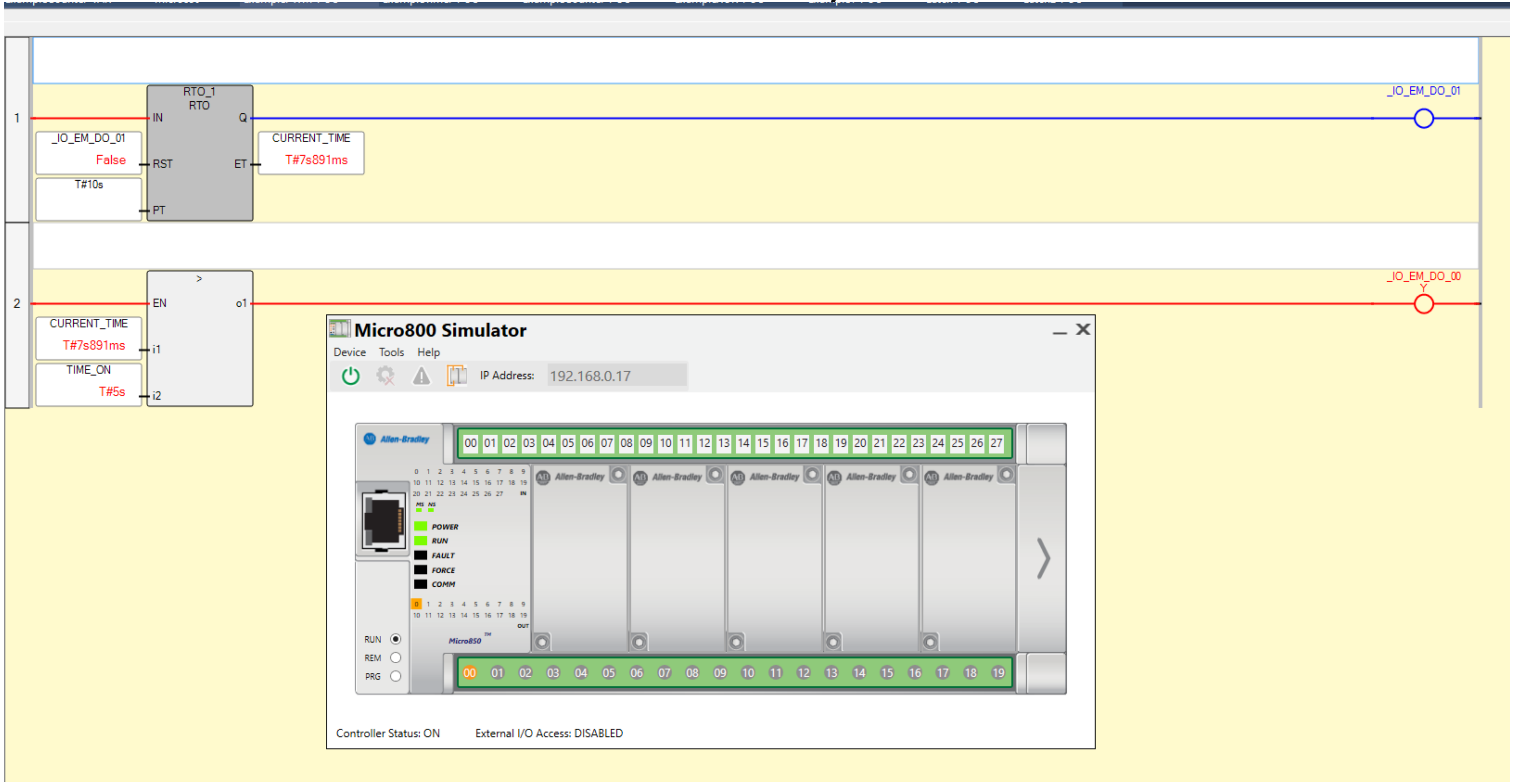


Use this table to help determine the parameter values for this instruction.

PROGRAMMING – PULSE WIDTH MODULATION

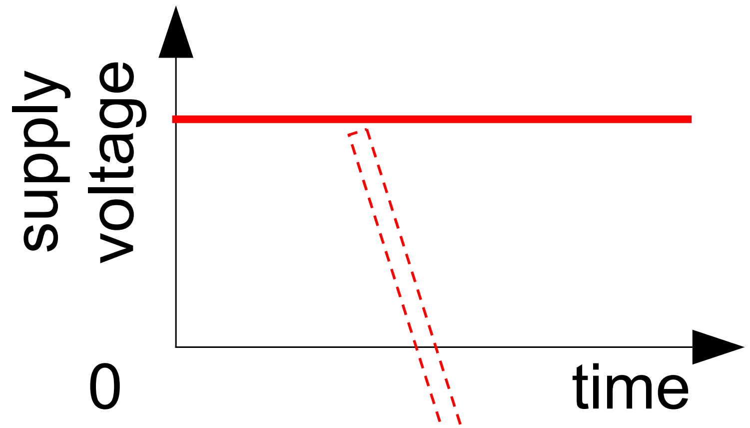
Parameter	Parameter type	Data type	Description
Enable	Input	BOOL	Instruction block enable. This level is instruction block triggered. TRUE - Update Sts. PWM is made active or inactive depending on the On input parameter and valid configuration. FALSE - Sts is only updated. PWM state (active or inactive) is not affected.
On	Input	BOOL	Turns the PWM output ON/Active or OFF/Inactive. TRUE - PWM output is active or continues to be active with latest valid configuration. Output LED is ON when PWM is active, even if duty cycle is set to 0%. FALSE - PWM output is inactive if configuration is also valid.
Freq	Input	UDINT	Pulse Frequency. • 1 - 100000 Hz
DutyCycle	Input	UINT	Pulse Duty Cycle. • 0 - 1000 (0% - 100%)
ChType	Input	UINT	Channel Type • 0 - Embedded • 1 - Plugin • 2 - Expansion
ChSlot	Input	UINT	Channel Slot • 0 - Embedded
ChNum	Input	UINT	Channel Number • 0 - PWM CH0
ENO	Output	BOOL	Enable output. Applies only to Ladder Diagram programs.

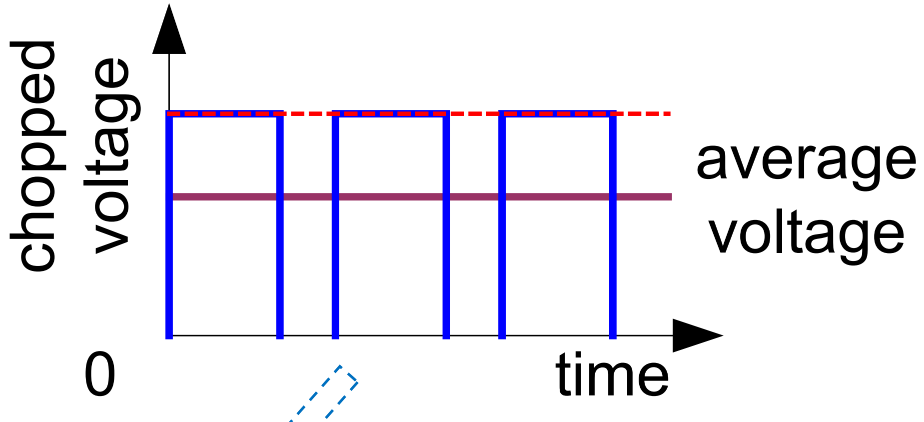
PROGRAMMING – TIMERS/PWM



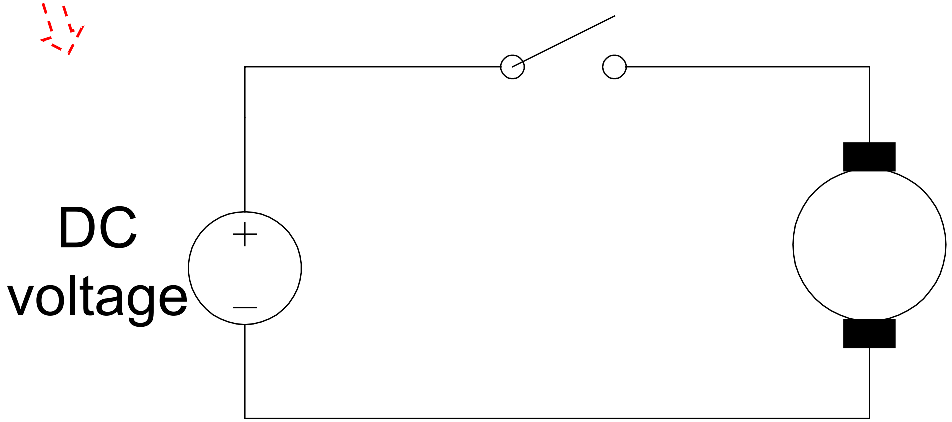
DC MOTOR SPEED CONTROL

CONTROL OF D.C. MOTORS



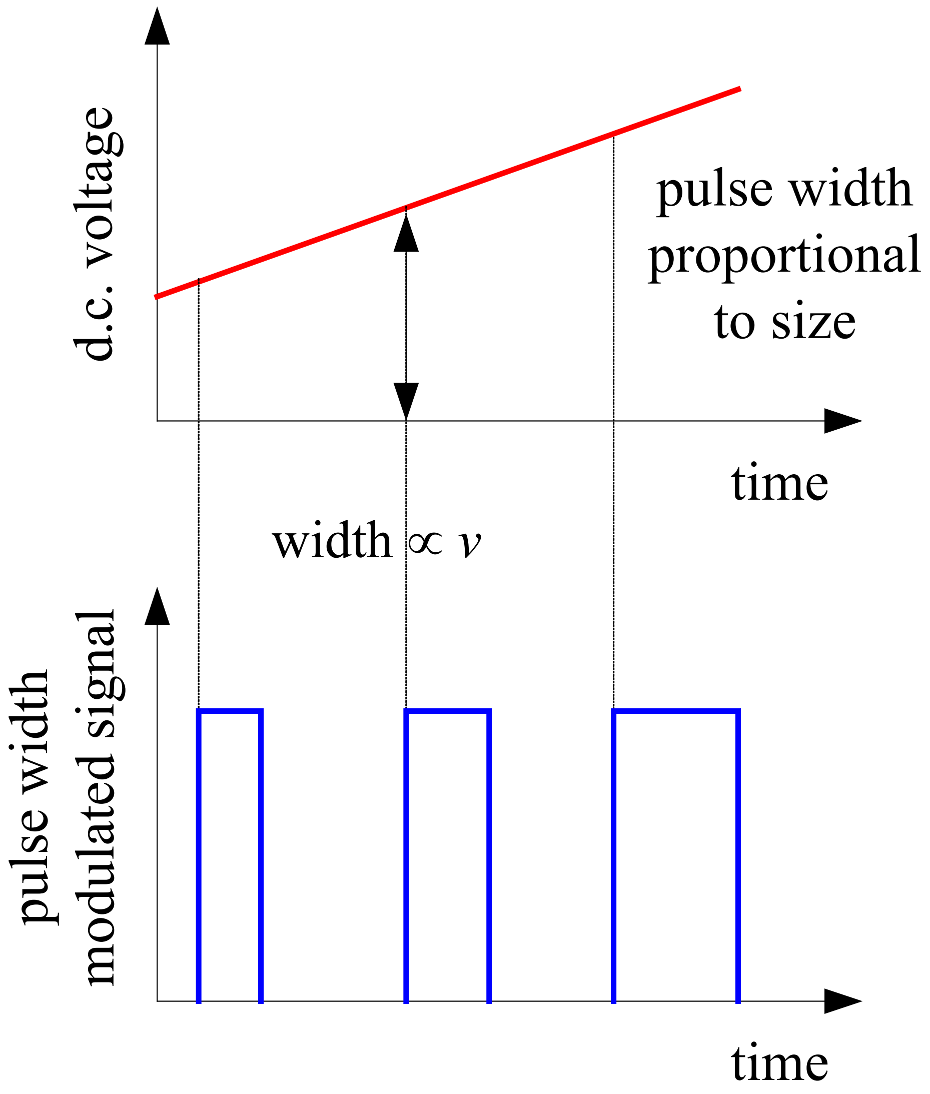


electronically controlled high frequency switch to chop the d.c. voltage



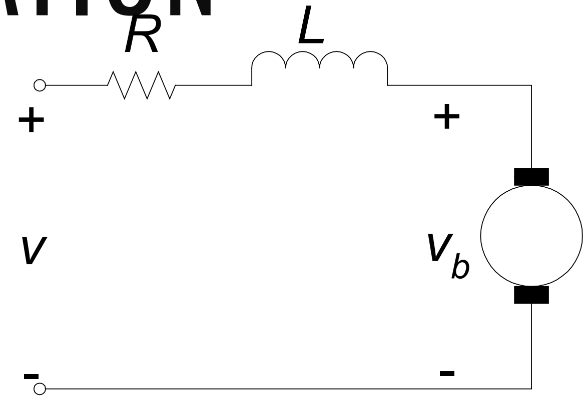
$$\bar{v}(t) \cong \frac{1}{t - t_0} \int_{t_0}^{t} v(t)dt$$

PULSE WIDTH MODULATION (PWM)



DC MOTOR STEADY-STATE OPERATION

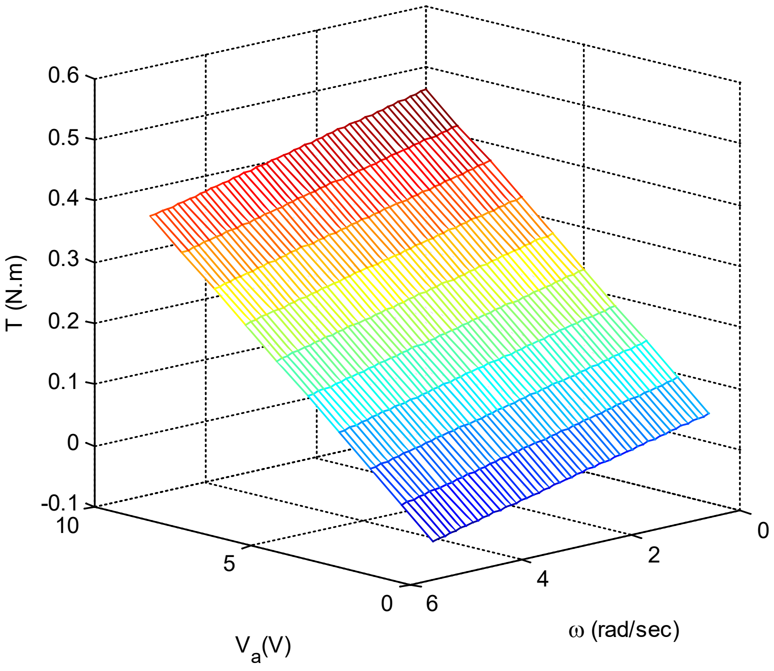
At steady-state: $di/dt = 0$ and $d\omega/dt = 0$



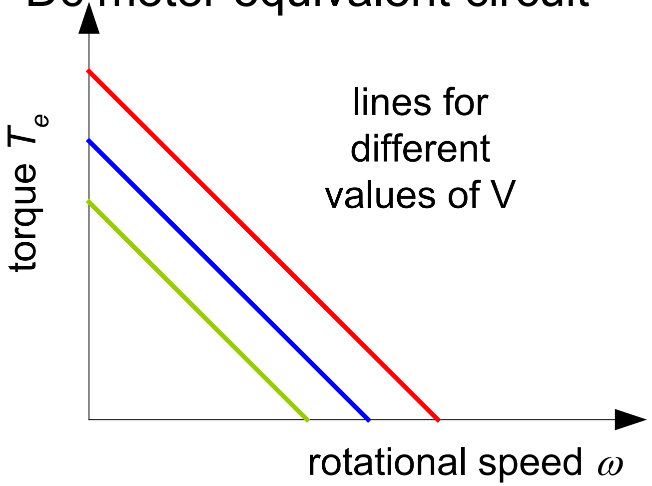
Dc motor equivalent circuit

$$\Rightarrow \quad i_a = \frac{v_a - v_b}{R_a} = \frac{v_a - K_b\omega}{R_a}$$

$$T_e = K_e i_a = K_e \frac{(v_a - K_b\omega)}{R_a}$$

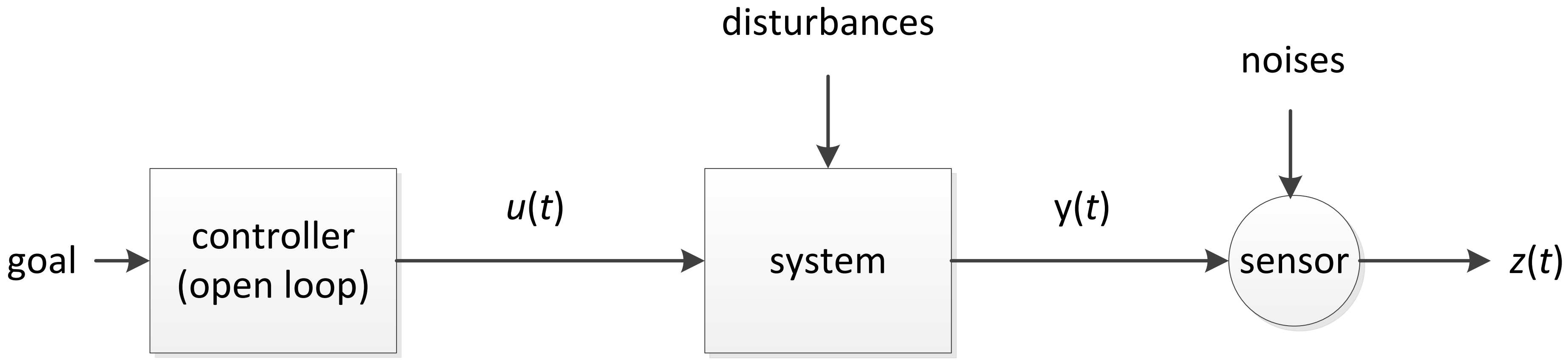



```
counter1 = 1;
counter2 = 1;
for V = 3:0.05:9
% for V = 3
    for spd = 0.1:0.1:6
        T(counter2,counter1) = Ke / R * (V - Kb * spd);
        counter2 = counter2 + 1;
    end
    counter1 = counter1 + 1;
    counter2 = 1;
end
```

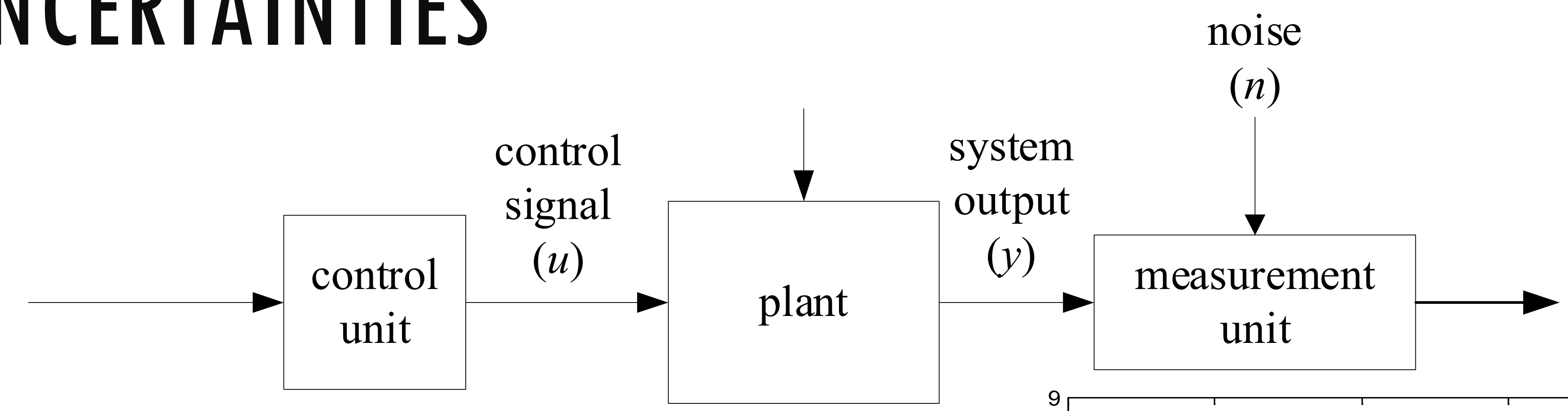


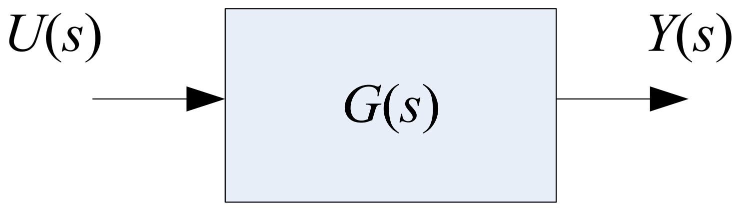
torque-speed characteristic with different input voltage

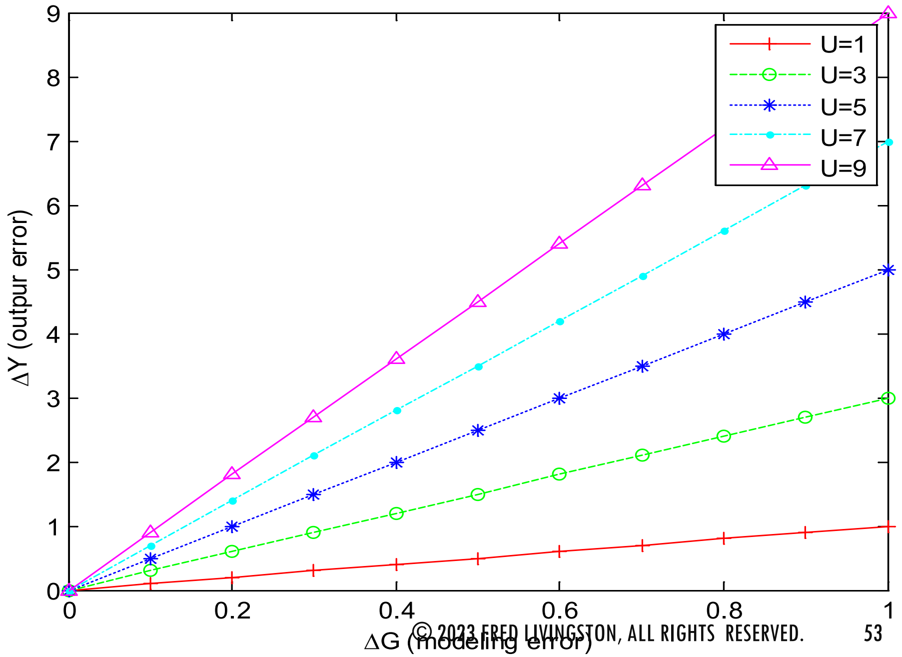
OPEN-LOOP CONTROL



OPEN-LOOP CONTROL – SENSITIVE TO UNCERTAINTIES

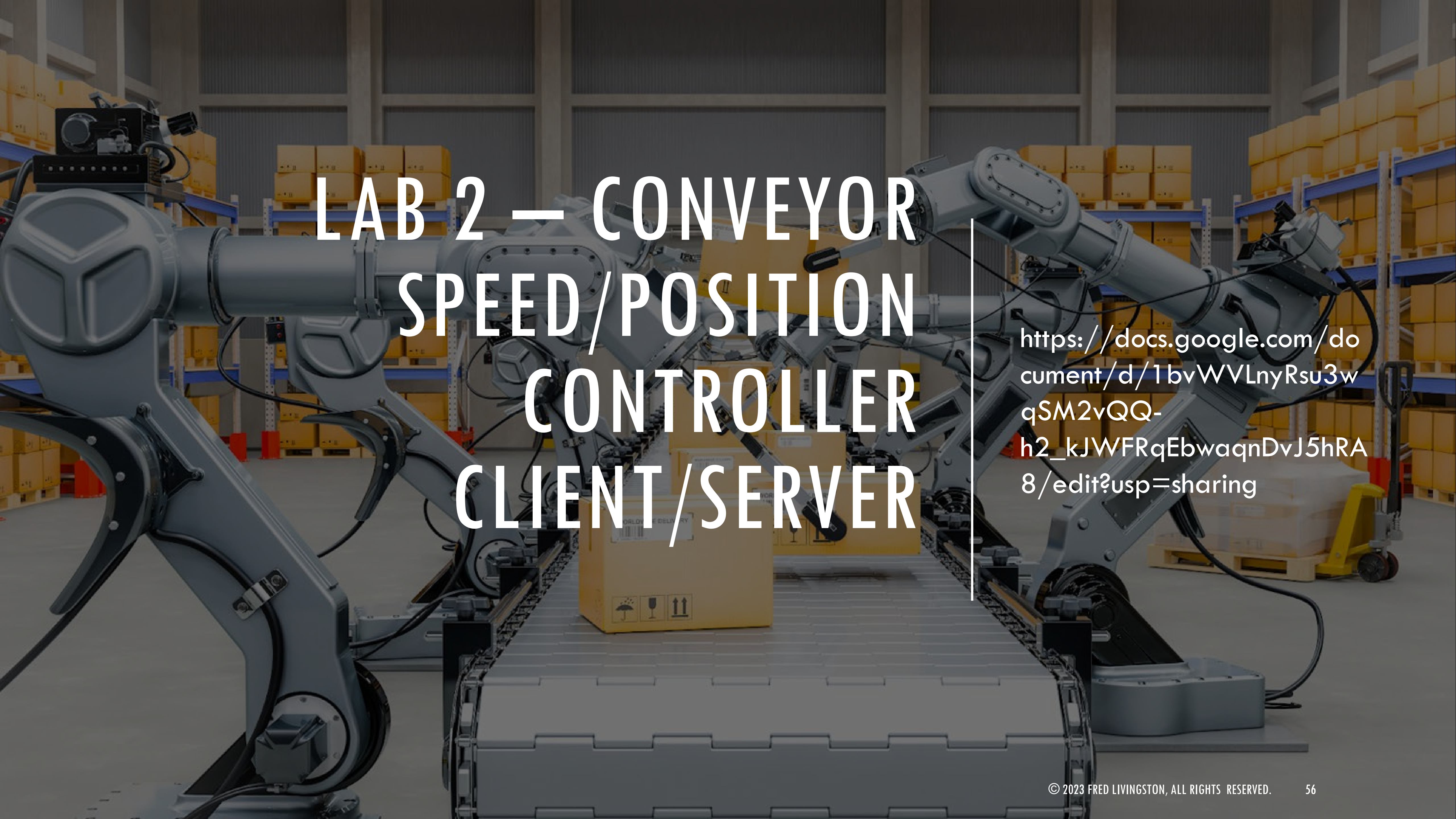






HW2

LABORATORY 2



LAB 2 – CONVEYOR SPEED/POSITION CONTROLLER CLIENT/SERVER

https://docs.google.com/document/d/1bvWVLnyRsu3wqSM2vQQ-h2_kJWFRqEbwaqnDvJ5hRA8/edit?usp=sharing

Conceptual Design [15 points]

- Function Block Diagram - Server (what are the inputs, outputs, main task) (5 pts)
- Function Block Diagram - Client (5 pts)
- Message Diagram (5 pts)

Design Implementation [55 points]

- Using the Connected Component WorkBench, implement an algorithm using the programming language of your choice (Ladder-Logic, Function Block Diagram, Structure Text) to perform the desired task for the server. (Please submit this code) (35 pts)
- Using MATLAB, Simulink, or python to perform the desired task of the client. (Please submit this code) (20 pts)

Automation Demonstration [30 points]

Record an implementation demonstration with a narrative description of the algorithm.

- The client can enable and disable the conveyor system. (10 pts)
- The client can control the speed of the conveyor system (slow, medium, fast) (10 pts)

- The client can control the direction of the conveyor system. (10 pts)

END OF LECTURE

