LabVIEW LINX Arduino using SPI and I2C

Hans-Petter Halvorsen

Table of Contents

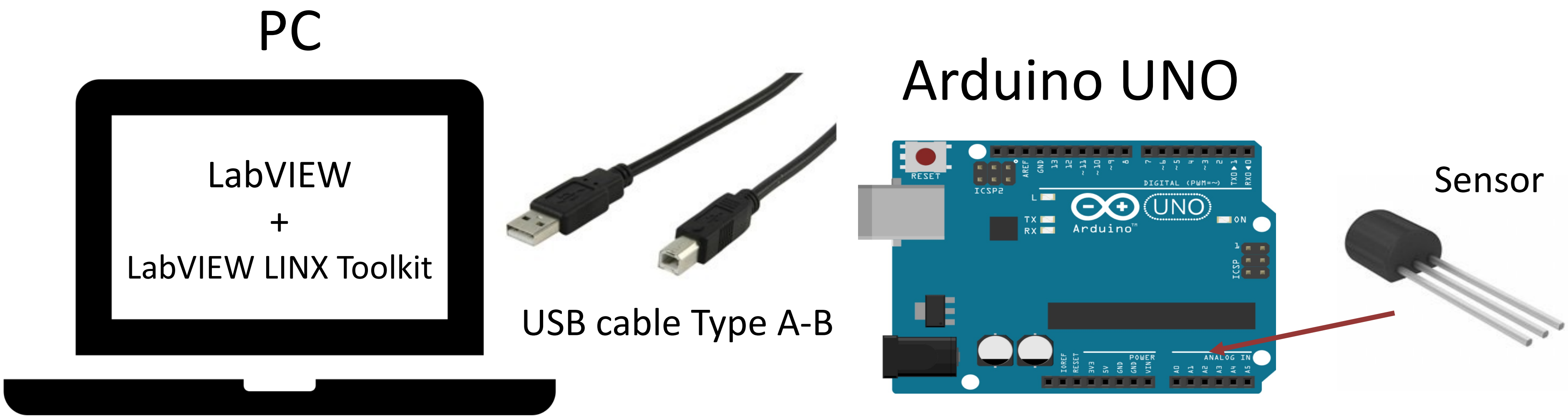
Introduction

Contents

- This Tutorial shows how we can use Arduino in combination with the LabVIEW Programming environment
- **Arduino** is a cheap open-source Microcontroller platform with Input/Output pins that can be used for many purposes like reading Sensor data, Datalogging, Internet of Things Applications, etc.
- **LabVIEW** is a popular Graphical Programming Environment
- **LabVIEW LINX Toolkit** is an add-on for LabVIEW which makes it possible to program the Arduino device using LabVIEW
- If you don't have "LabVIEW Professional" Software, you may use the "LabVIEW Community Edition", which is free for non-commercial use. You then get a very low-cost DAQ/Datalogging System!
- In that way we can create Data Logging Applications, IoT Applications, etc. without the need of an expensive DAQ device or Software.
- In this Tutorial we will use the more advanced features and communicate with Digital Sensors, etc. using **I2C** and **SPI** Communication

Arduino + LabVIEW LINX

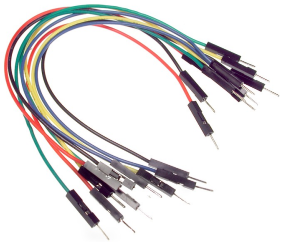
Below we see a typical Application where you can use an Arduino (Hardware) and LabVIEW/LabVIEW LINX (Software) to Log Data from a Sensor:

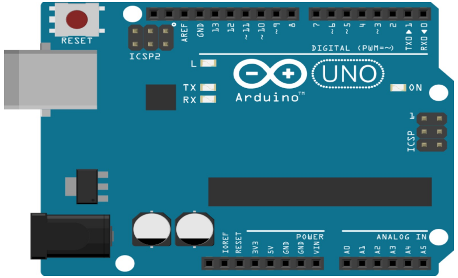


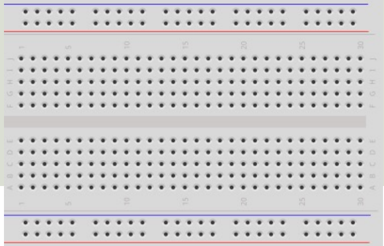
In this Tutorial we will use the more advanced features and communicate with Digital Sensors, etc. using **I2C** and **SPI** Communication

Hardware

- Arduino
- Breadboard
- Wires (Jumper Wires)
- **TC74** Temperature Sensor with **I2C** Interface
- DAC **MCP4911** with **SPI** Interface







TC74

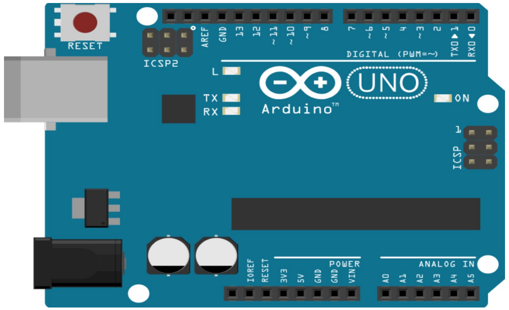
Arduino

Hans-Petter Halvorsen

Table of Contents

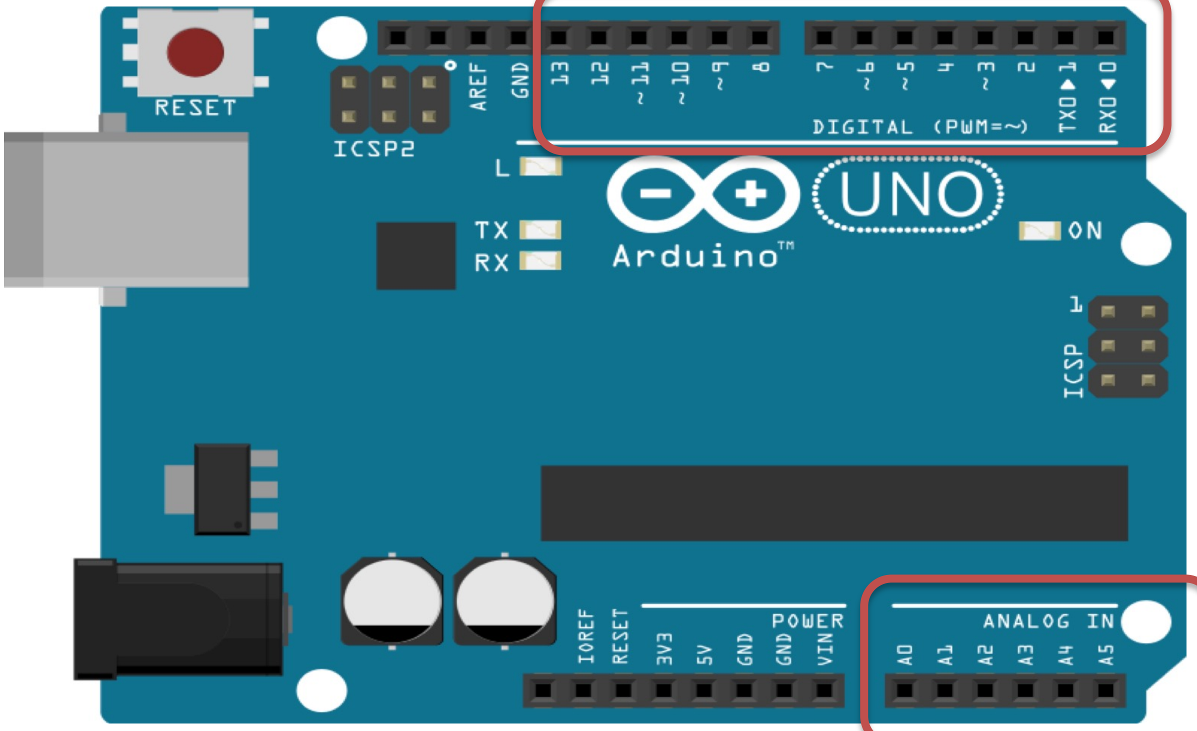
Arduino UNO

- Arduino is a Microcontroller
- Arduino is an open-source platform with Input/Output Pins (Digital In/Out, Analog In and PWM)
- Price about $20
- Arduino Starter Kit ~$40-80
  with Cables, Wires, Resistors, Sensors, etc.



Arduino I/O Channels

Digital Inputs and Digital Outputs



You can choose from the code if they are to be inputs or outputs

Those marked with ~ can also be used as "Analog Outputs", so-called PWM outputs

Analog Inputs

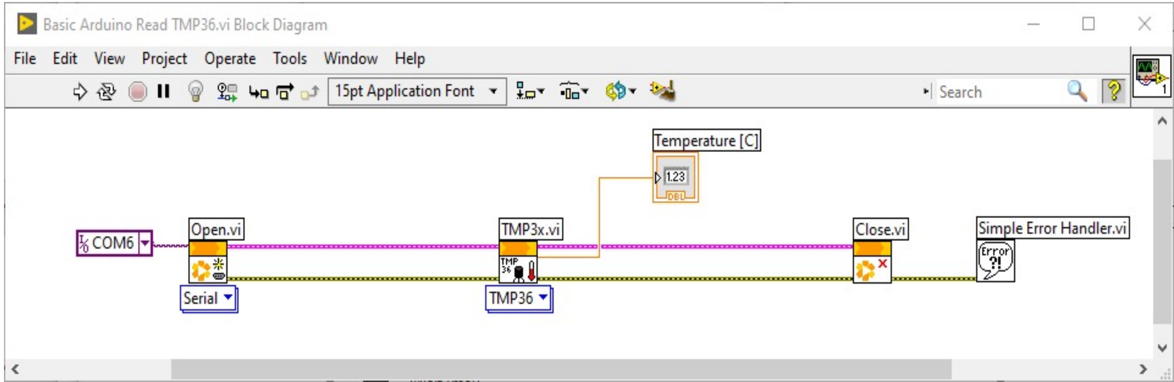
LabVIEW

Hans-Petter Halvorsen

LabVIEW

- LabVIEW is Graphical Software
- LabVIEW has powerful features for Simulation, Control and DAQ applications

Basic LabVIEW Example:



LabVIEW LINX

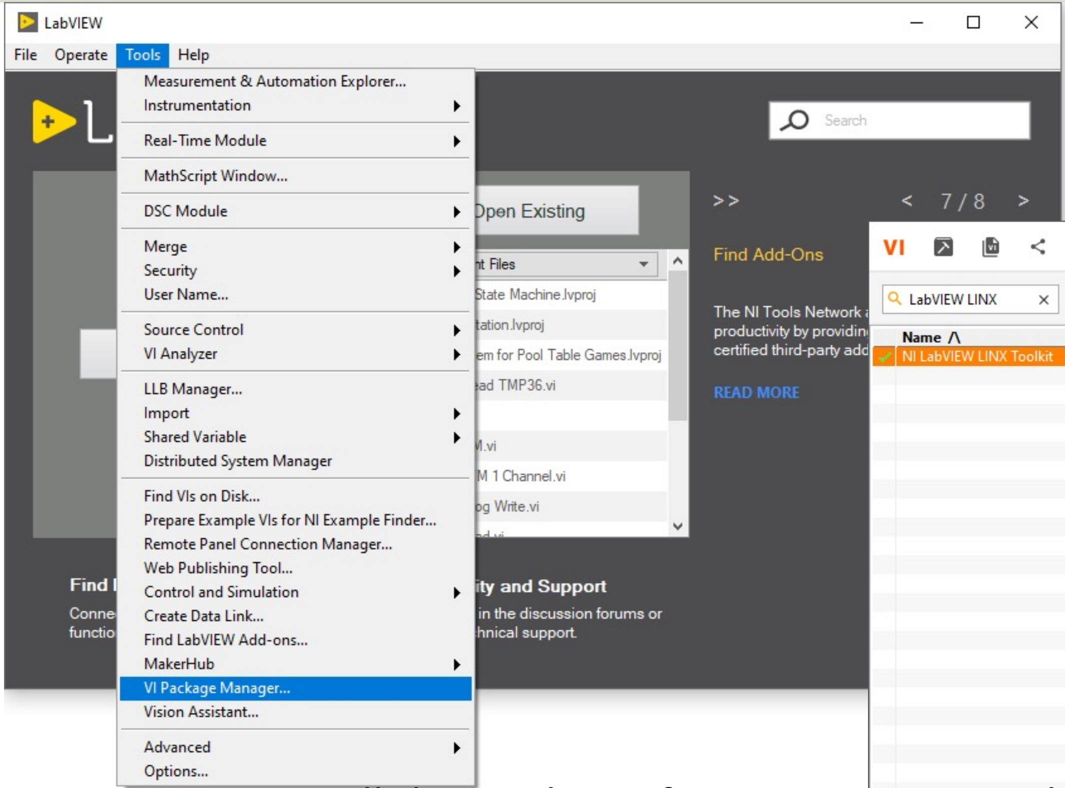
Hans-Petter Halvorsen

Table of Contents

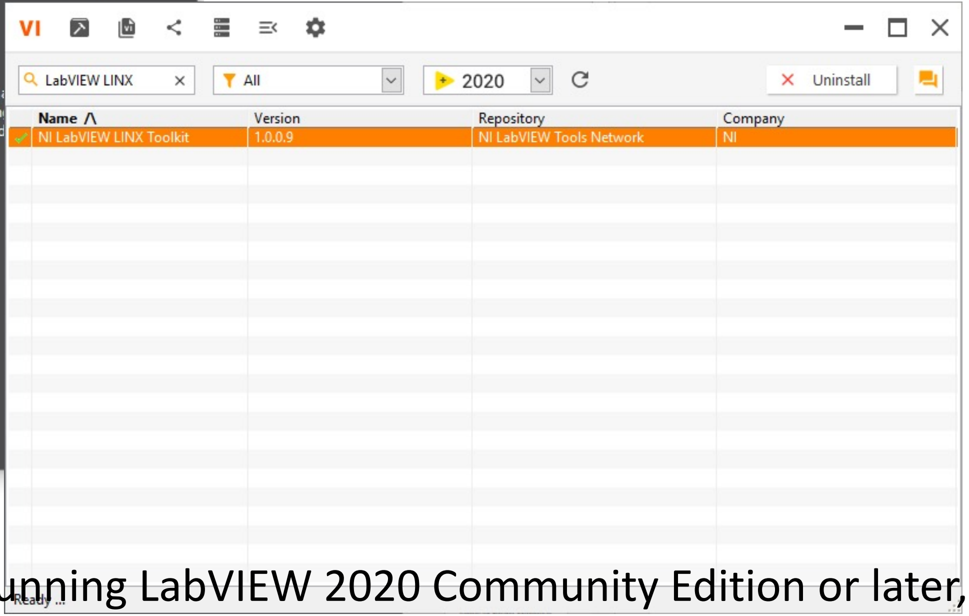
LabVIEW LINX Toolkit

- The LabVIEW LINX Toolkit adds support for Arduino, Raspberry Pi, and BeagleBone embedded platforms
- We will use **Arduino UNO** in this Tutorial

Installing LabVIEW LINX Toolkit

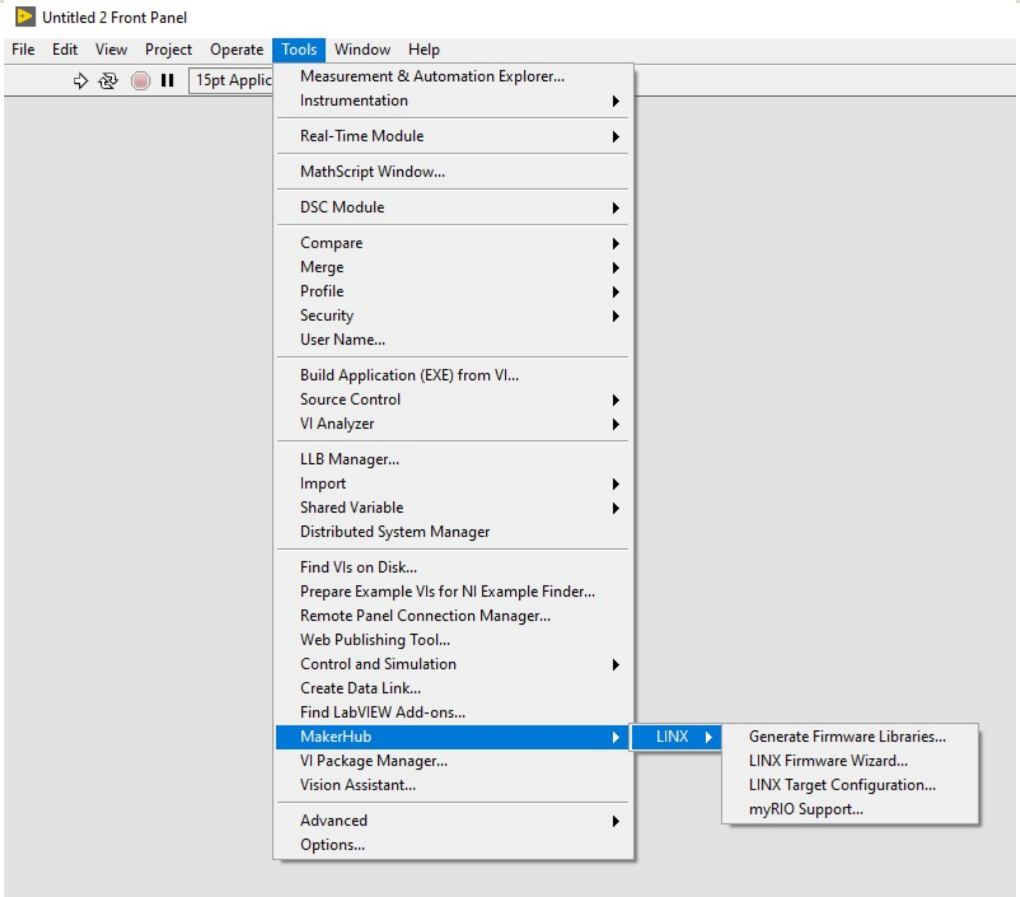


Use **VI Package Manger**

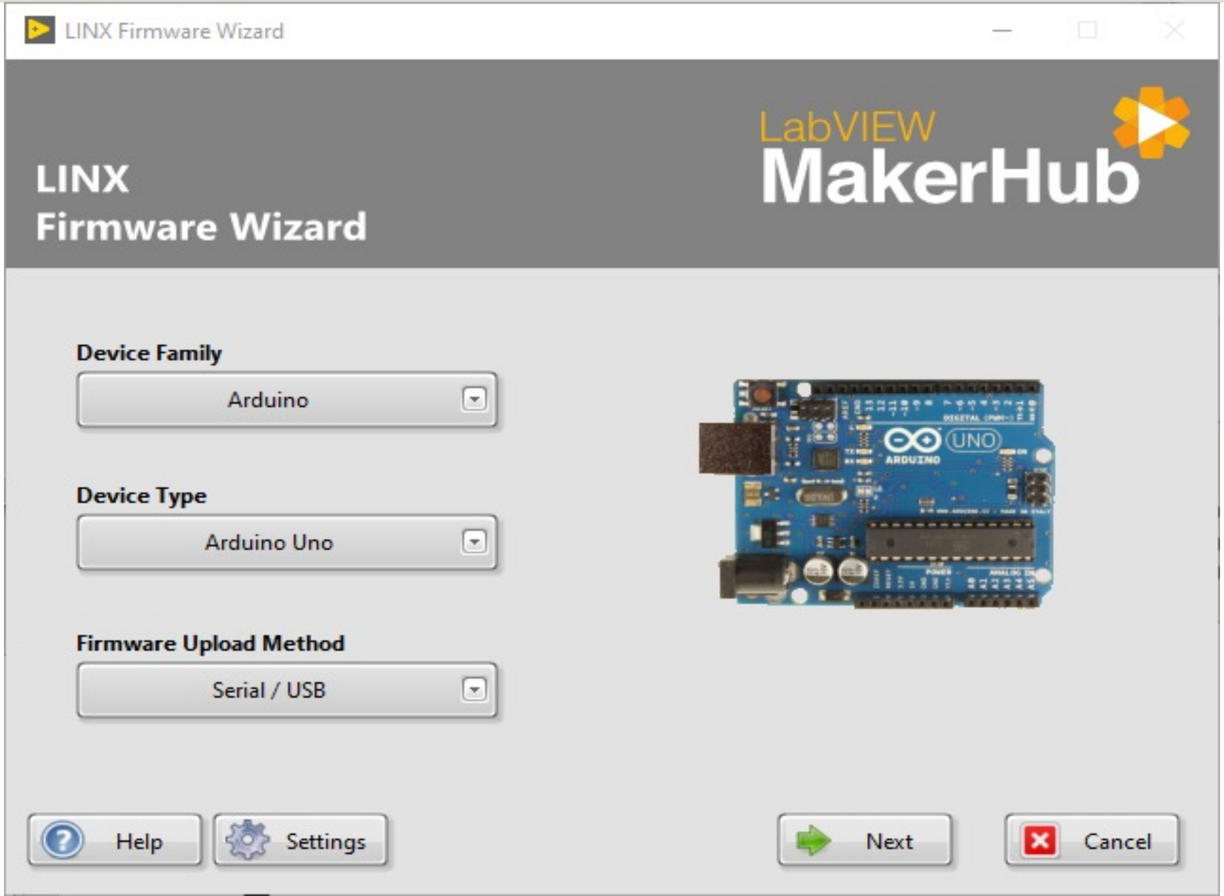


Note: Do not install this package if you are running LabVIEW 2020 Community Edition or later, as the Community Edition already includes the LabVIEW LINX Toolkit

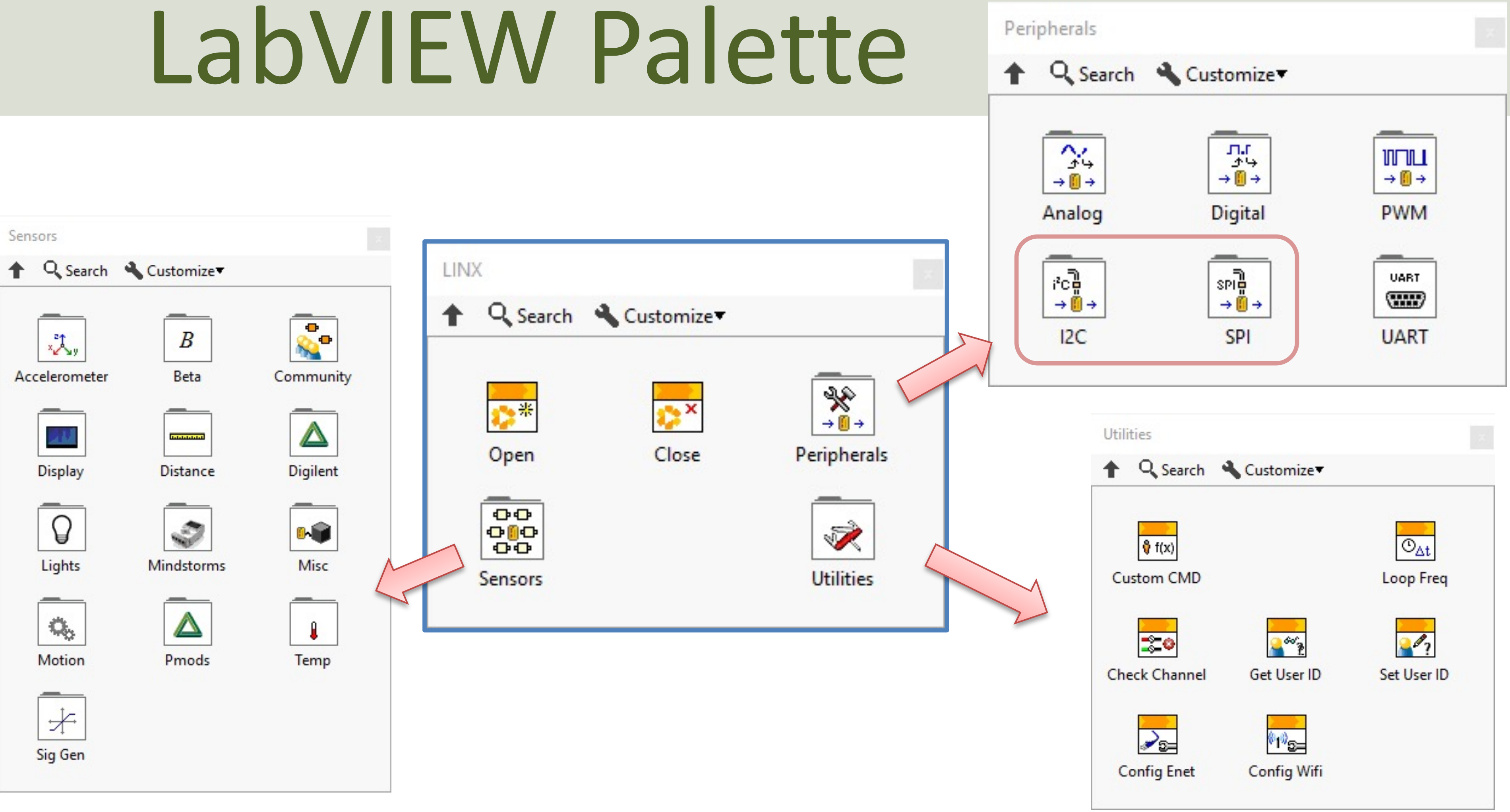
LabVIEW LINX



LINX Firmware Wizard



LabVIEW Palette



SPI and I2C

Hans-Petter Halvorsen

Table of Contents

SPI/I2C

- Digital Sensors typically use either the SPI or the I2C communication protocol
- The Arduino UNO has built-in hardware support for SPI and I2C communication

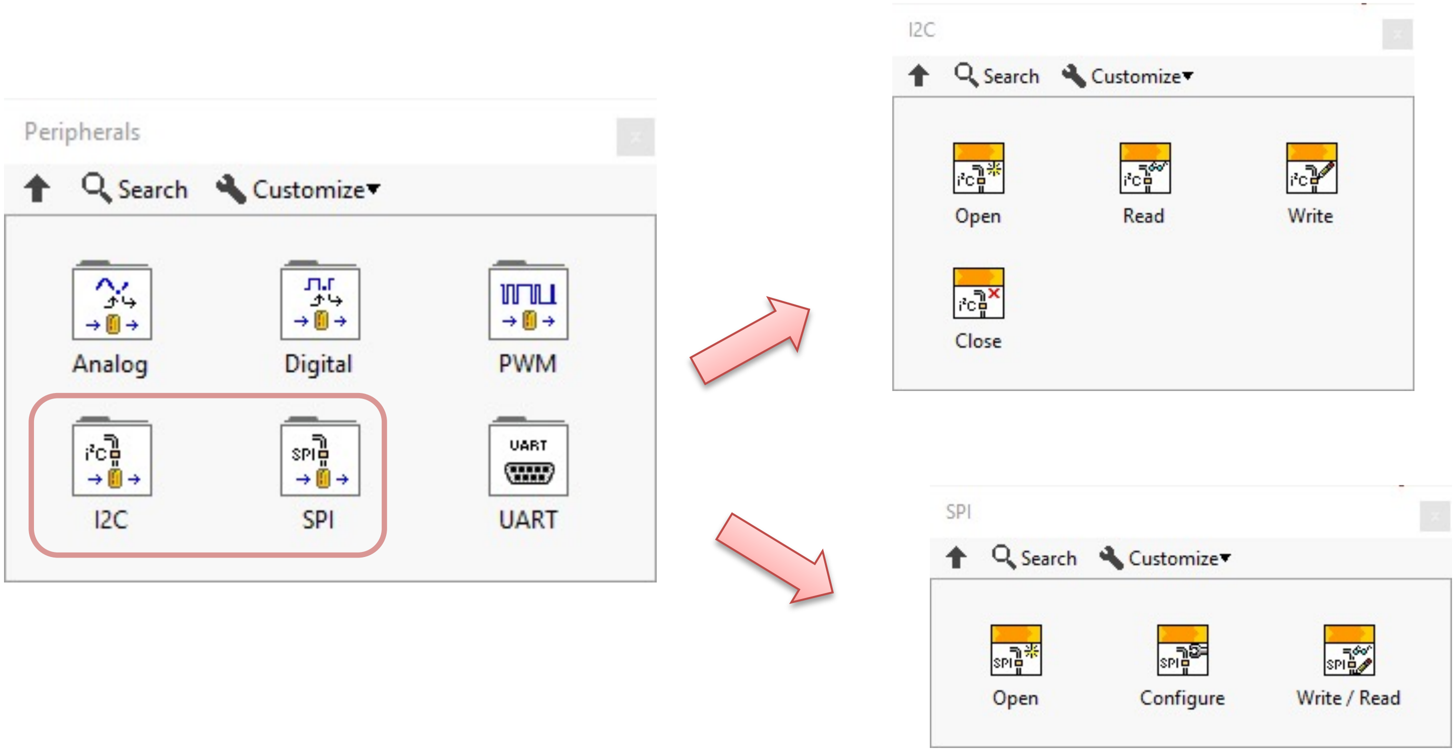
SPI

- 4-Wire Protocol
- SPI supports full-duplex. Data can be sent and received at the same time
- Higher data transfer rate than I2C
- Complex wiring if more than one Slave

I2C

- 2-Wire Protocol
- I2C supports only half-duplex. Data cannot be sent and received at the same time
- Lower data transfer rate than SPI
- Multiple Slaves are easier

SPI/I2C in LabVIEW LINX



I2C

Inter-Integrated Circuit (I^2C)

Hans-Petter Halvorsen

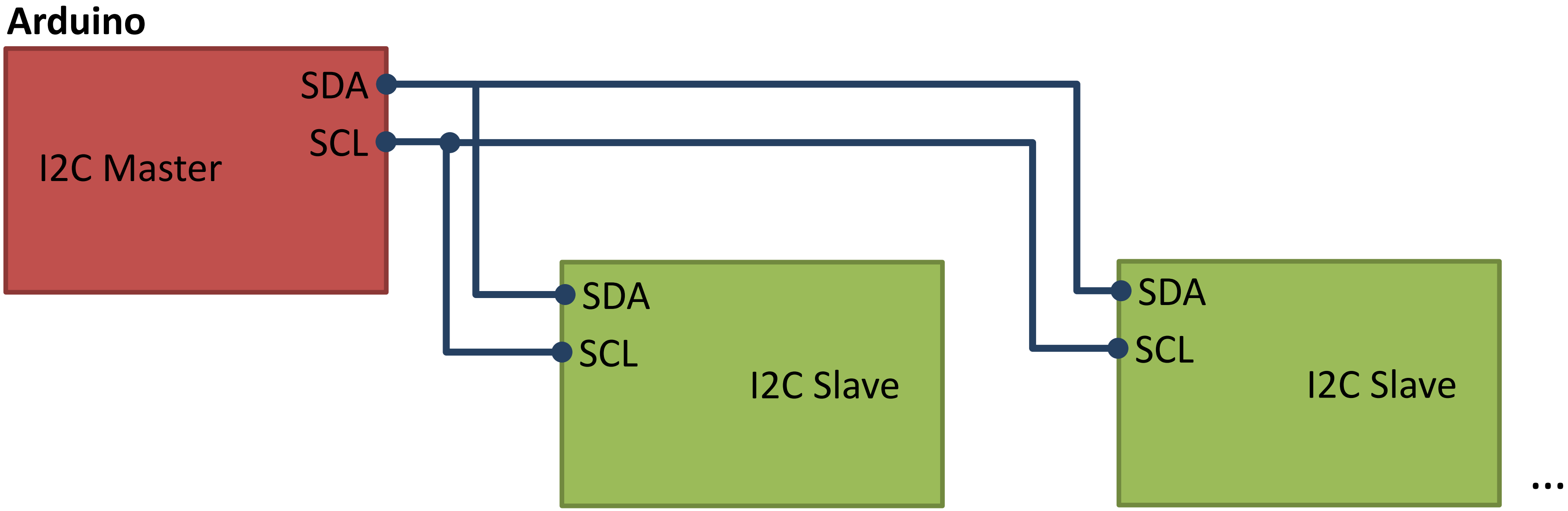
Table of Contents

I2C

- I2C is a multi-drop bus
- 2-Wire Protocol: SCL (Clock) + SDA (Data)
- Multiple devices can be connected to the I2C pins on the Arduino
- Each device has its own unique I2C address

I2C

Multiple devices can be connected to the I2C pins on the Arduino
Master – Device that generates the clock and initiates communication with slaves
Slave – Device that receives the clock and responds when addressed by the master.



ADC, DAC, Sensor, etc. with I2C Interface

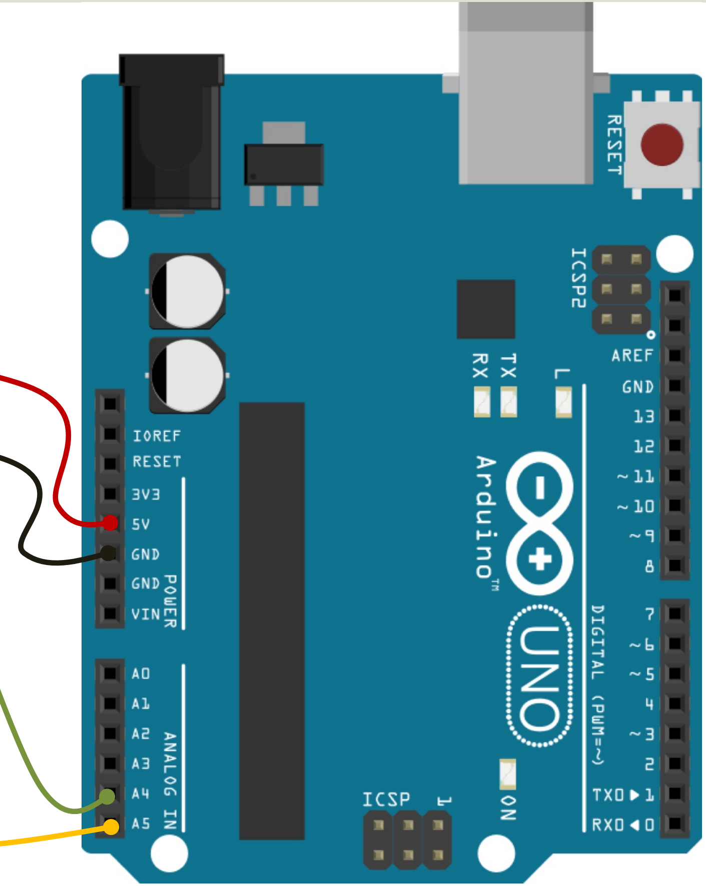
I2C with Arduino

VDD – Power Supply Input

GND – Ground

SDA (A4 pin) - Serial Data – Bidirectional

SCLK (A5 pin) - Serial Clock Input



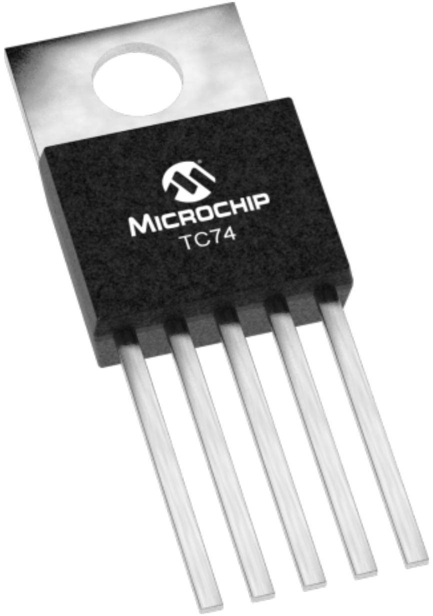
TC74 Temperature Sensor

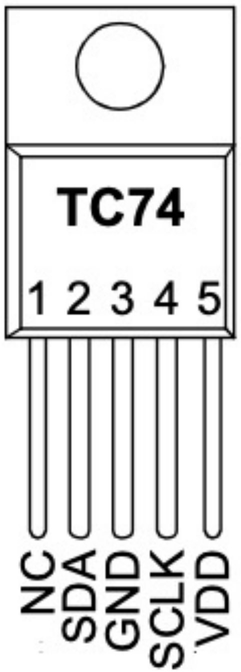
Hans-Petter Halvorsen

TC74 Temperature Sensor

SMBus/I2C Interface

TC74A0-5.0VAT



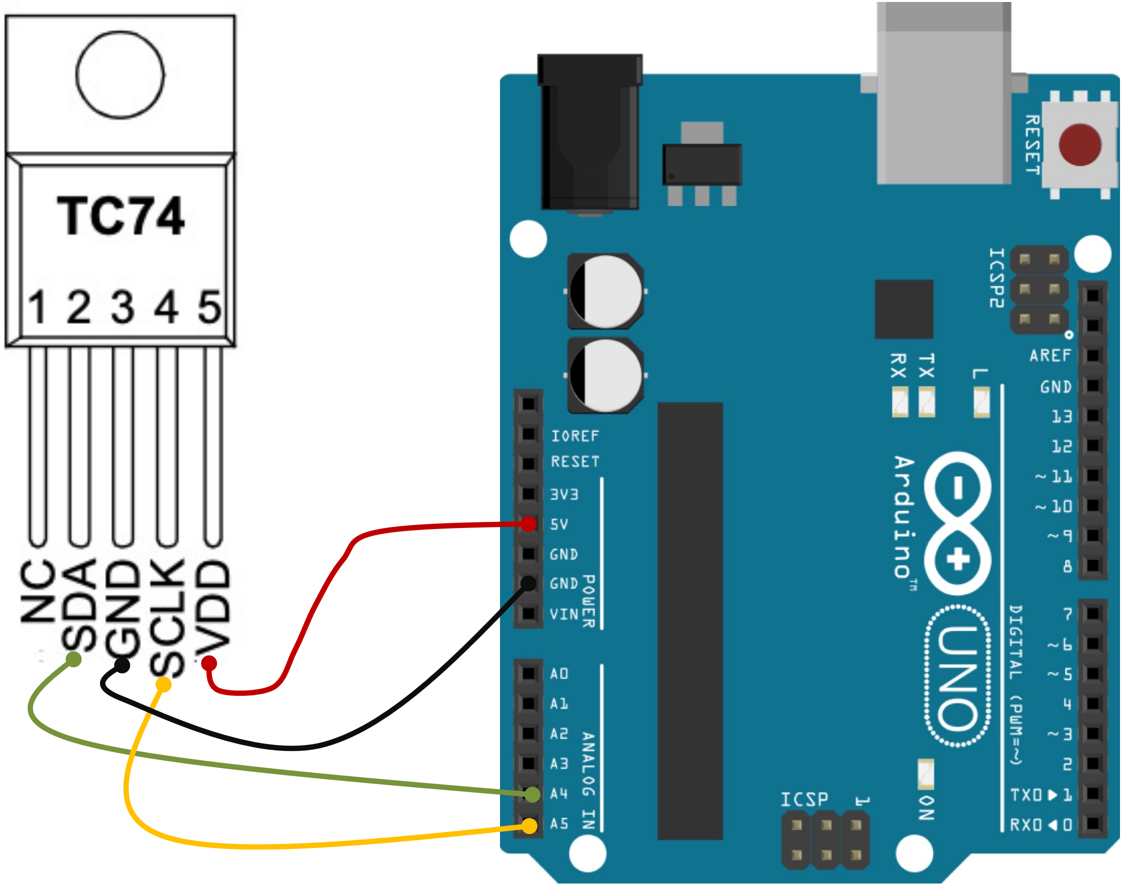


- The TC74 acquires and converts temperature information from its onboard solid-state sensor with a **resolution of ±1°C**.
- It stores the data in an internal register which is then read through the serial port.
- The system interface is a slave SMBus/I2C port, through which temperature data can be read at any time.
- Device Address: **0x48**

Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

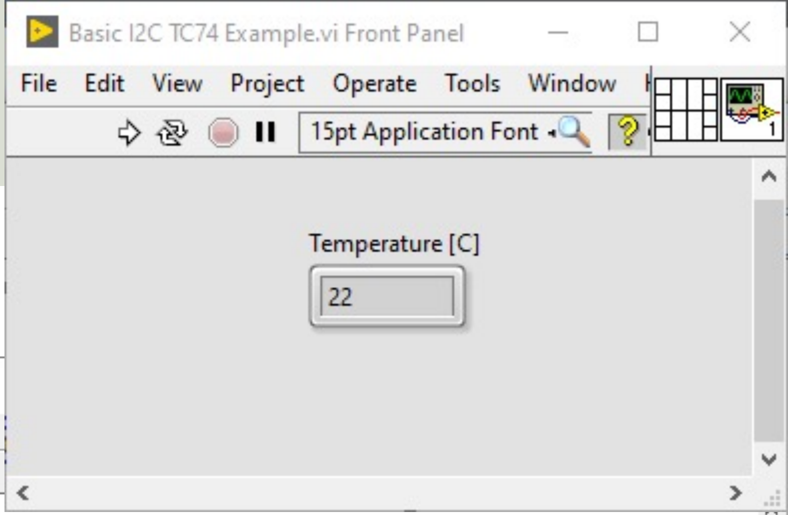
TC74 Example - Wiring

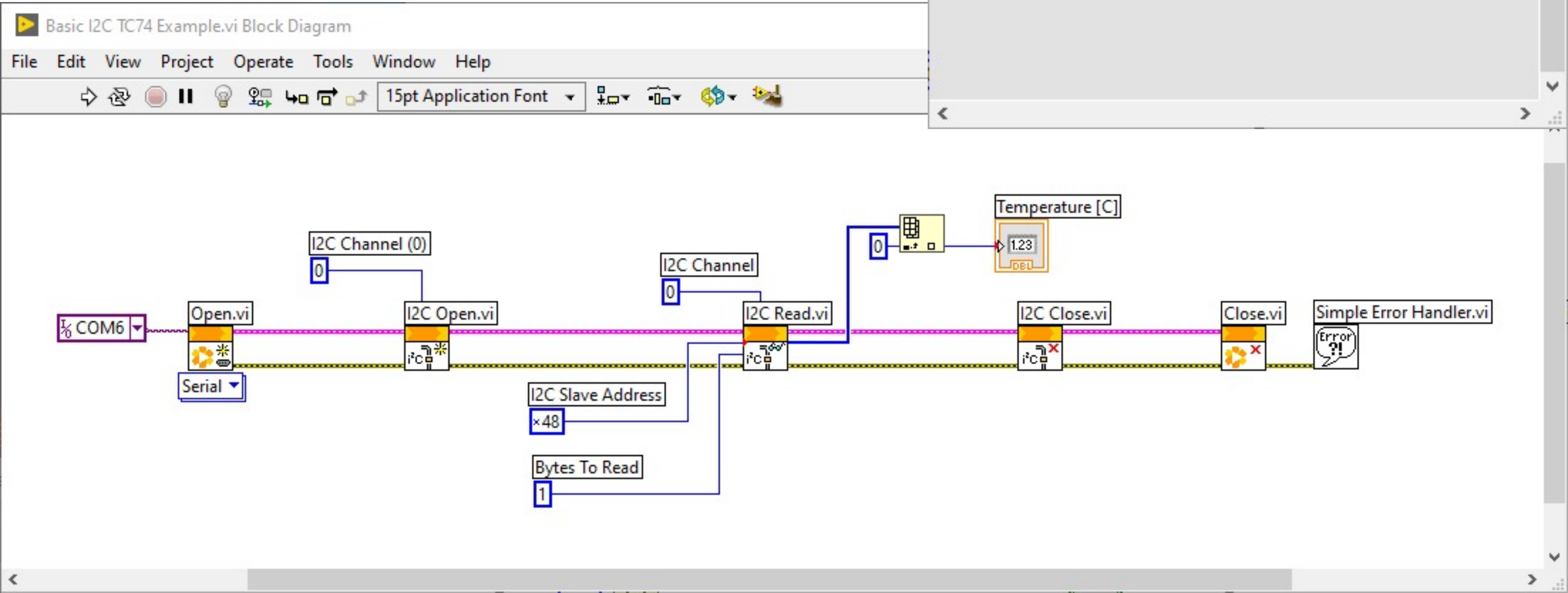
SDA - Serial Data – Bidirectional
SCLK - Serial Clock Input
VDD – Power Supply Input
GND – Ground
NC - Not in use (Not Connected)



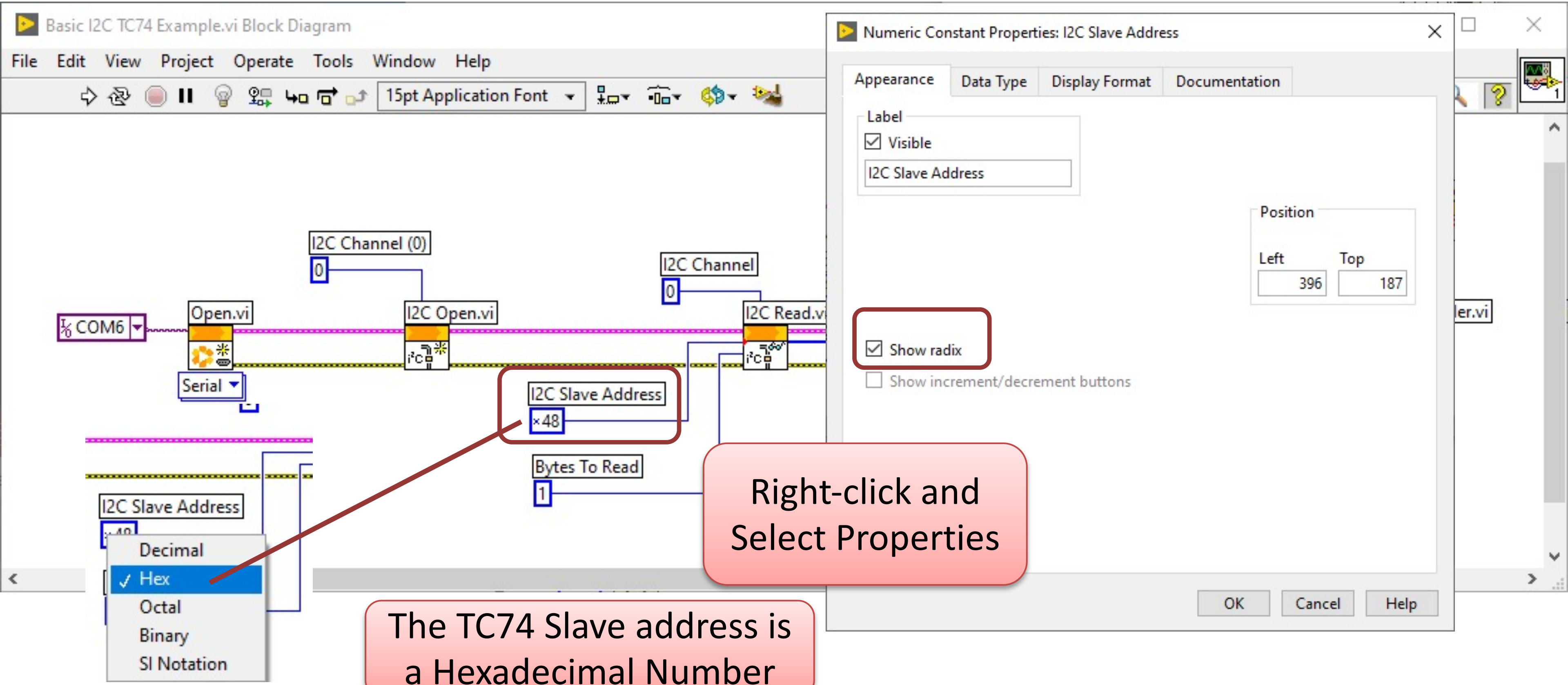
LabVIEW

Basic Example:





Slave Address

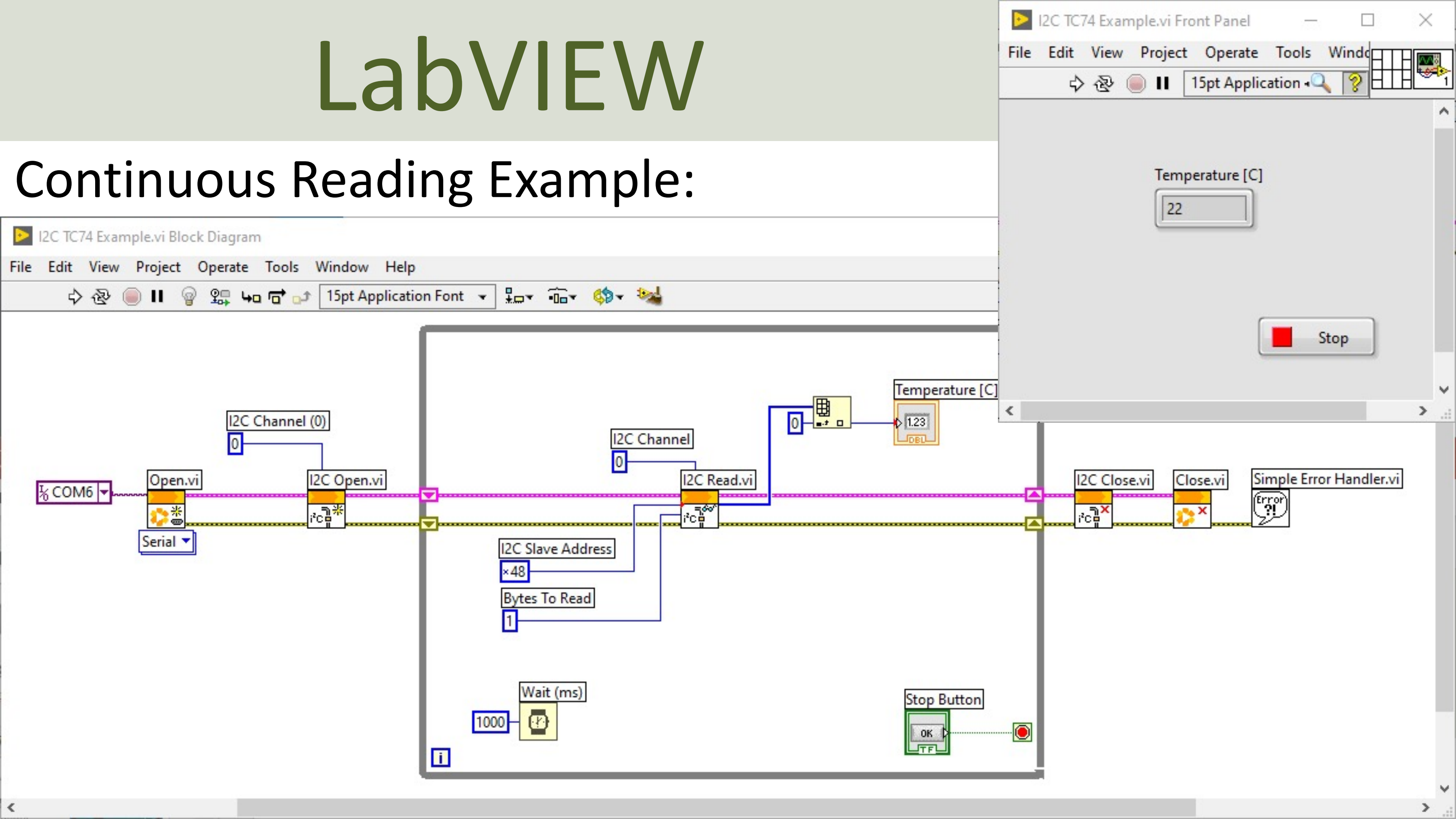


Right-click and Select Properties

The TC74 Slave address is a Hexadecimal Number

LabVIEW

Continuous Reading Example:



SPI

Serial Peripheral Interface (SPI)

Hans-Petter Halvorsen

Table of Contents

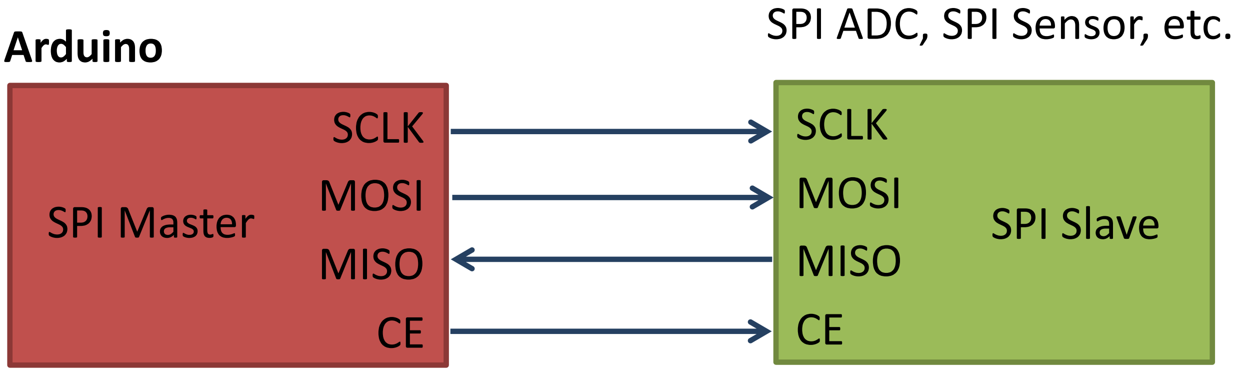
SPI

- Serial Peripheral Interface (SPI)
- 4–Wire Protocol (SCLK, CE, MOSI, MISO)
- SPI is an interface to communicate with different types of electronic components like Sensors, Analog to Digital Converts (ADC), etc. that supports the SPI interface
- Thousands of different Components and Sensors supports the SPI interface

https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/

SPI

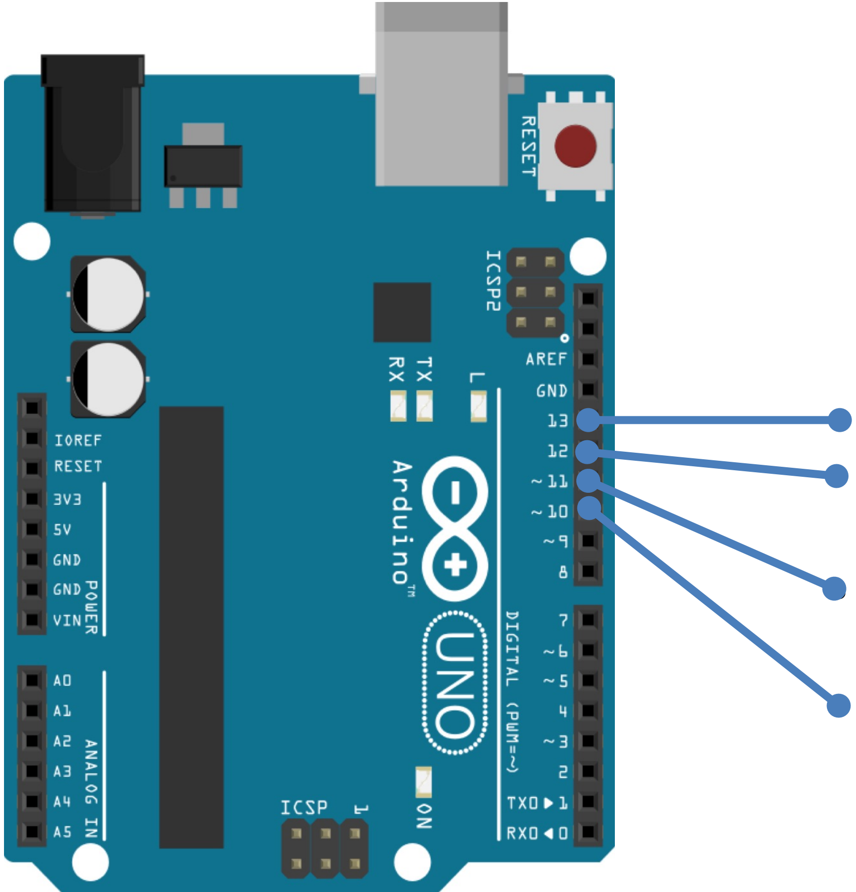
SPI devices communicate in full duplex mode using a master-slave architecture with a single master



The SPI bus specifies four logic signals:

- **SCLK**: Serial **Clock** (output from master)
- **MOSI**: Master Out Slave In (**data output from master**)
- **MISO**: Master In Slave Out (**data output from slave**)
- **CE** (often also called SS - Slave Select): Chip Select (often active low, output from master)

SPI with Arduino



SCLK: Serial Clock (output from master)

MISO: Master In Slave Out (data output from slave)

MOSI: Master Out Slave In (data output from master)

CE (often also called **SS** - Slave Select): Chip Select (often active low, output from master)

DAC - MCP4911

DAC – Digital to Analog Converter

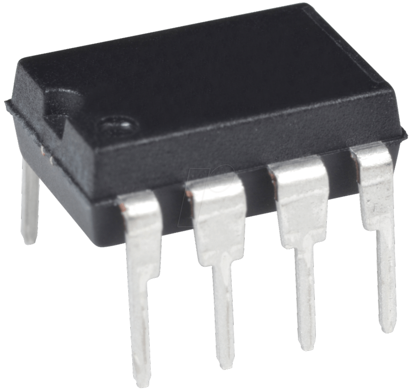
DAC – MCP4911

- DAC – Digital to Analog Converter
- Arduino UNO has no real Analog Out Channel – only Digital PWM channels
- We can use an external DAC in order to provide a real Analog Out
- MCP4911 is a single channel, 10-bit DAC with an external voltage reference and SPI interface

MCP49xx

MCP49xx is a family of DAC ICs:

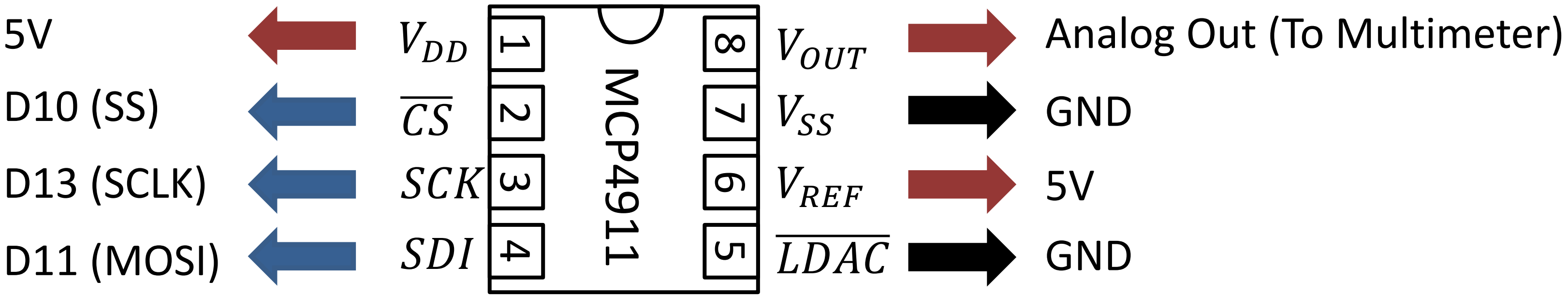
- MCP4901: 8-Bit Voltage Output DAC
- **MCP4911: 10-Bit Voltage Output DAC**
- MCP4921: 12-Bit Voltage Output DAC



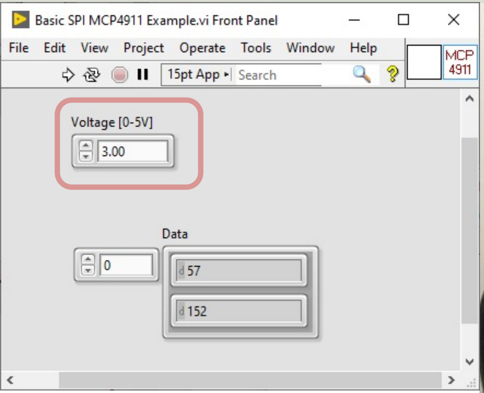
The different MCP49xx DACs work in the same manner, the only difference is the resolution (8, 10, or 12 resolution)

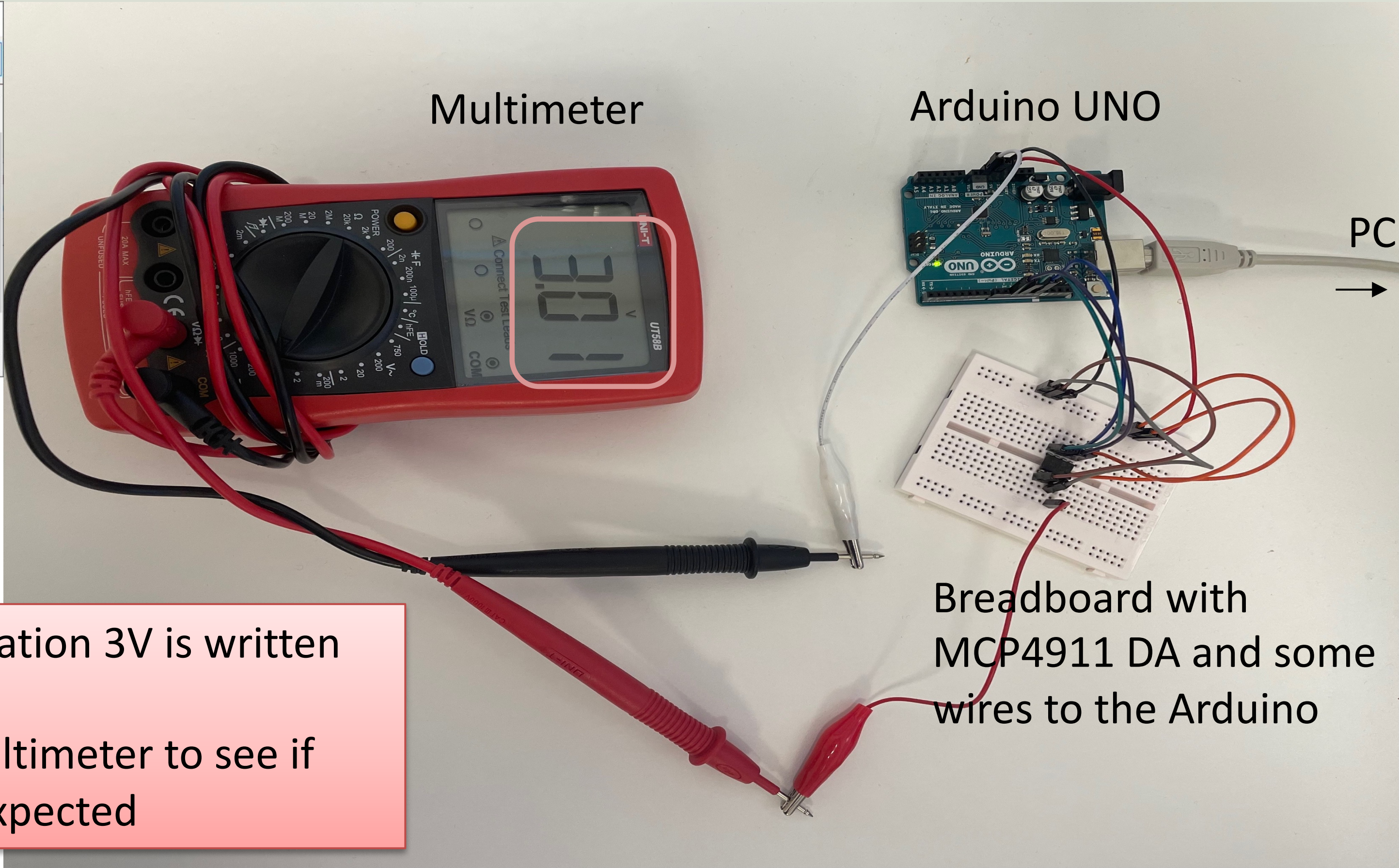
Datasheet: https://www.microchip.com/en-us/product/MCP4911

MCP4911 - Arduino Wiring



Test Setup





In the LabVIEW Application 3V is written to the MCP4911 DAC.
Then we can use a Multimeter to see if everything works as expected

MCP4911 – Write Data

5.2 Write Command

The write command is initiated by driving the $\overline{CS}$ pin low, followed by clocking the four Configuration bits and the 12 data bits into the SDI pin on the rising edge of SCK. The $\overline{CS}$ pin is then raised, causing the data to be latched into the DAC's input register.

Write Command Register:

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
0	BUF	$\overline{GA}$	$\overline{SHDN}$	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	x	x
bit 15															bit 0

bit 15 — 0 = Write to DAC register
1 = Ignore this command

bit 14 — **BUF:** V_{REF} Input Buffer Control bit
1 = Buffered
0 = Unbuffered

bit 13 — **$\overline{GA}$:** Output Gain Selection bit
1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 — **$\overline{SHDN}$:** Output Shutdown Control bit
1 = Active mode operation. VOUT is available.
0 = Shutdown the device. Analog output is not available.

bit 11-0 — **D11:D0:** DAC Input Data bits. Bit x is ignored.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0

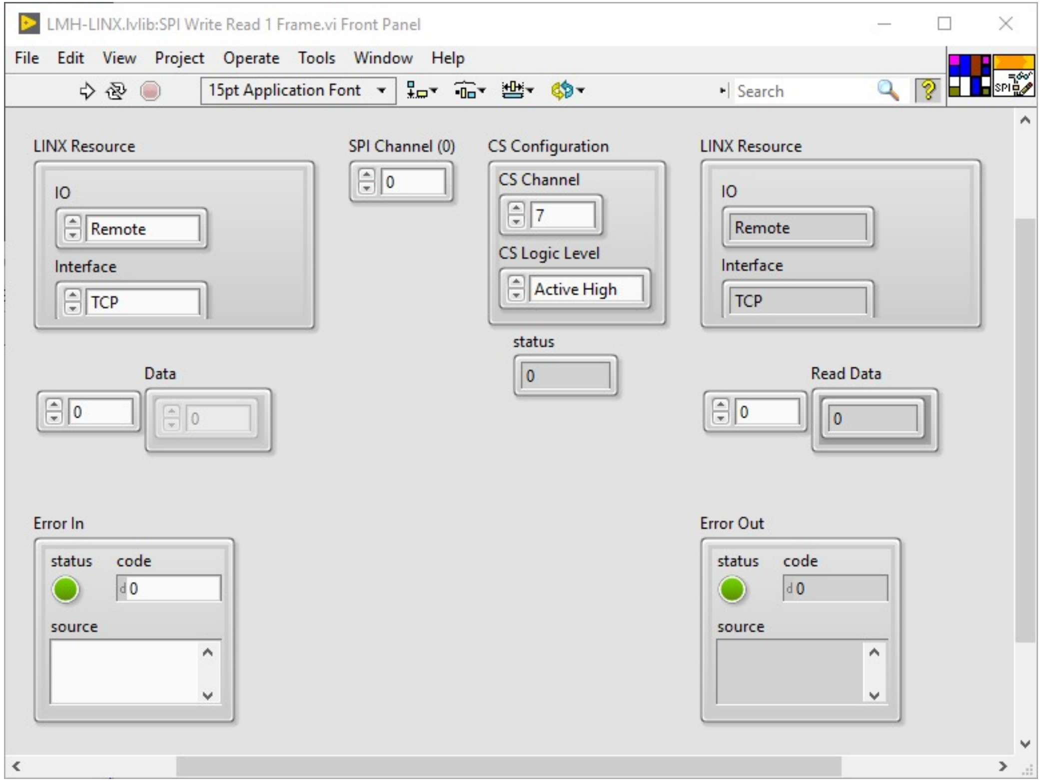
10-bit data

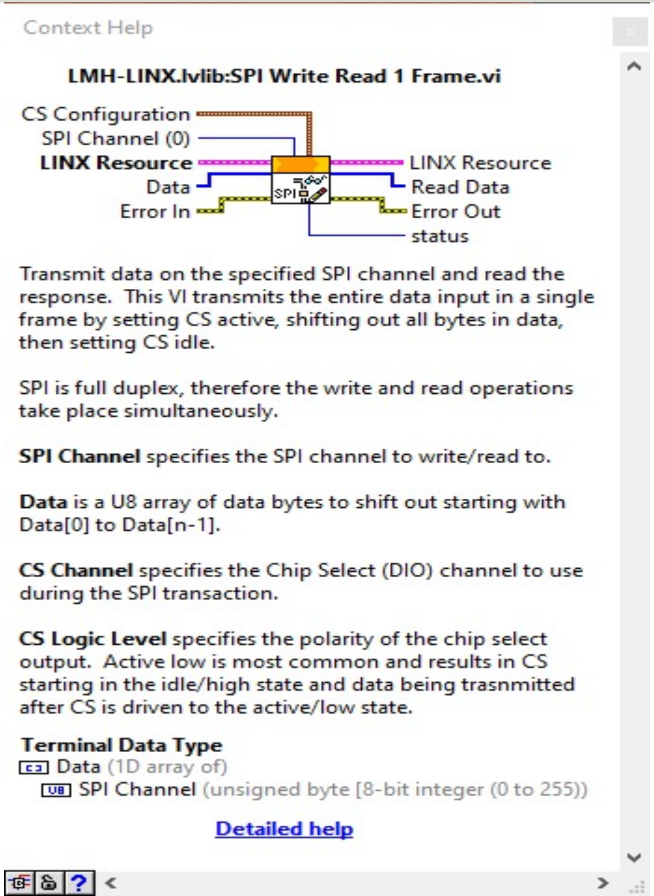
$$2^{10} = 1024 \text{ DAC Levels}$$

$$0V - 5V \rightarrow 0 - 1023$$

Datasheet: https://www.microchip.com/en-us/product/MCP4911

SPI Write in LabVIEW LINX





MCP4911 – Write Data Examples

0V => DAC Value = 0

10-bit data

$2^{10} = 1024$ DAC Levels

$0V - 5V \rightarrow 0 - 1023$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

0

12288

Need to Write **Word 12288** to the MCP4911 Write Register divided into 2 **Bytes**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

48 **0**

48

0

Note! Word – 16Bits and Byte – 8 Bits

MCP4911 – Write Data Examples

2.5V => DAC Value = 512

10-bit data

$2^{10} = 1024$ DAC Levels

$0V - 5V \rightarrow 0 - 1023$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

512

14336

Need to Write **Word 14336** to the MCP4911 Write Register divided into 2 **Bytes**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

56 **0**

56
0

Note! Word – 16Bits and Byte – 8 Bits

MCP4911 – Write Data Examples

5V => DAC Value = 1023

10-bit data

$2^{10} = 1024$ DAC Levels

$0V - 5V \rightarrow 0 - 1023$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0

1023

16380

Need to Write **Word 16380** to the MCP4911 Write Register divided into 2 **Bytes**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0

63 **252**

63

252

Note! Word – 16Bits and Byte – 8 Bits

Split Word into 2 Bytes in LabVIEW

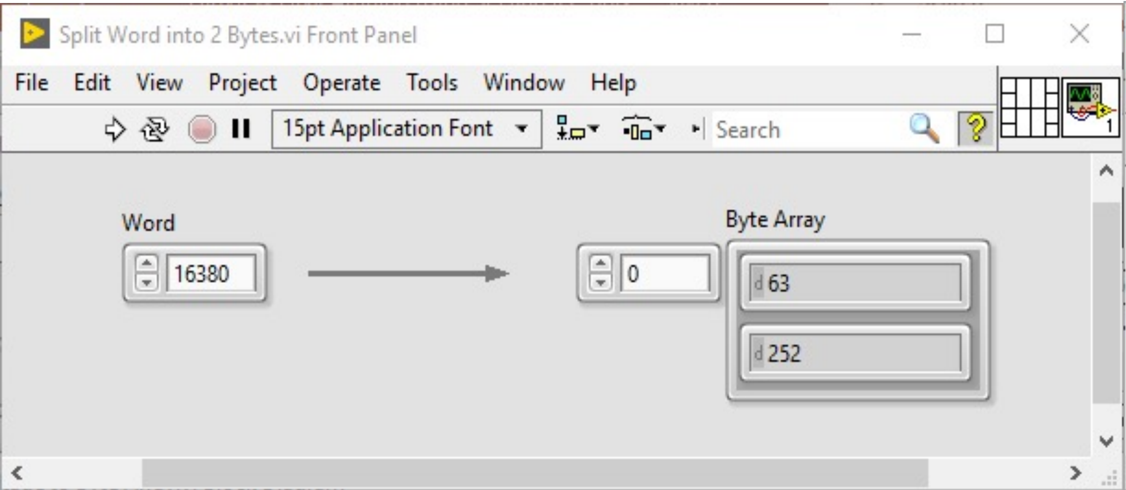
15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0

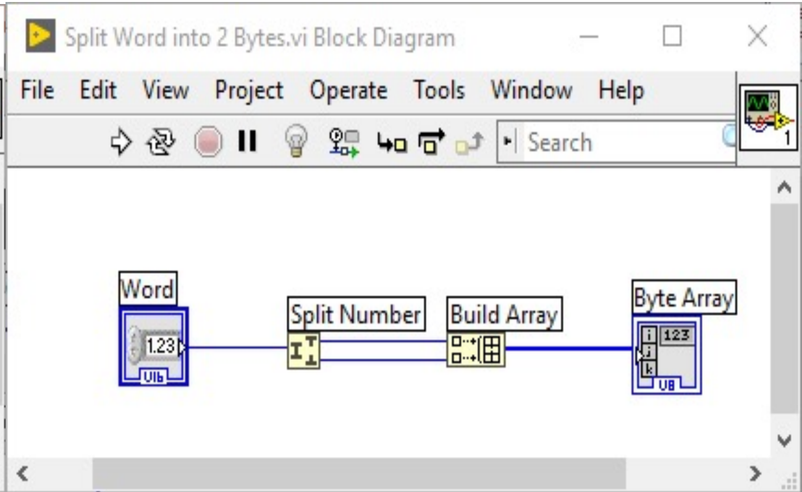
63

252

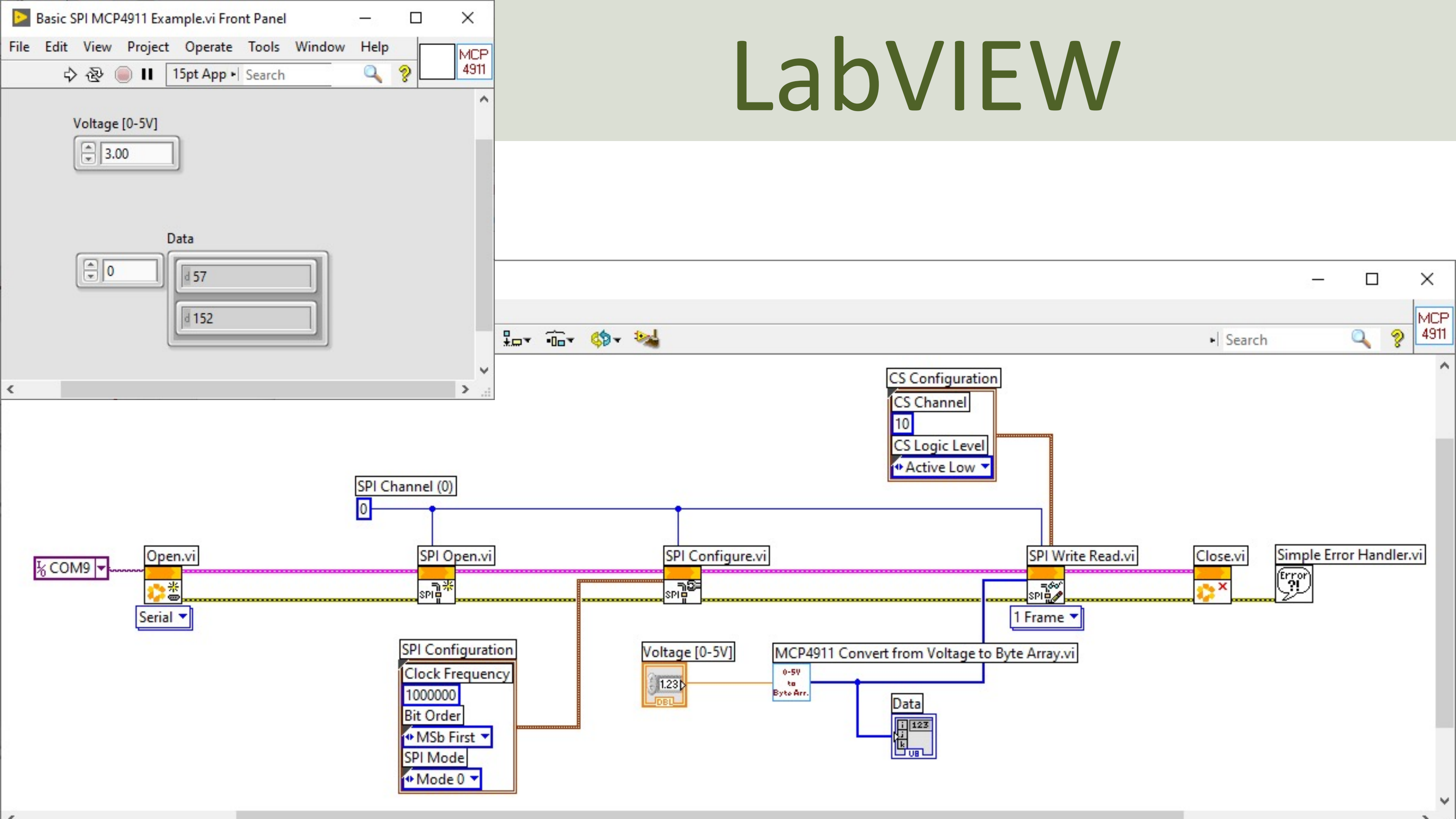
63
252

Note! Word – 16Bbits and Byte – 8 Bits



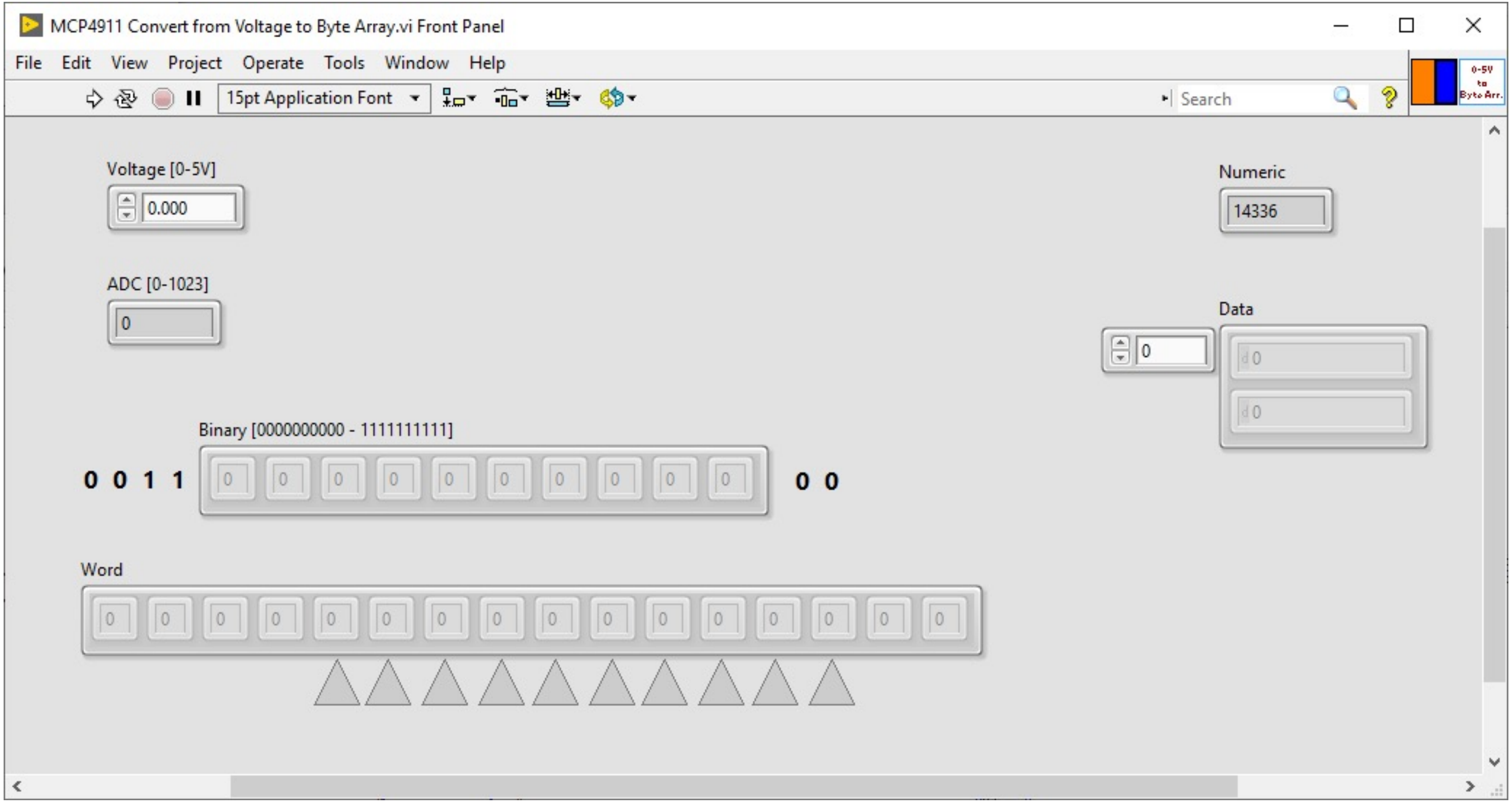


LabVIEW

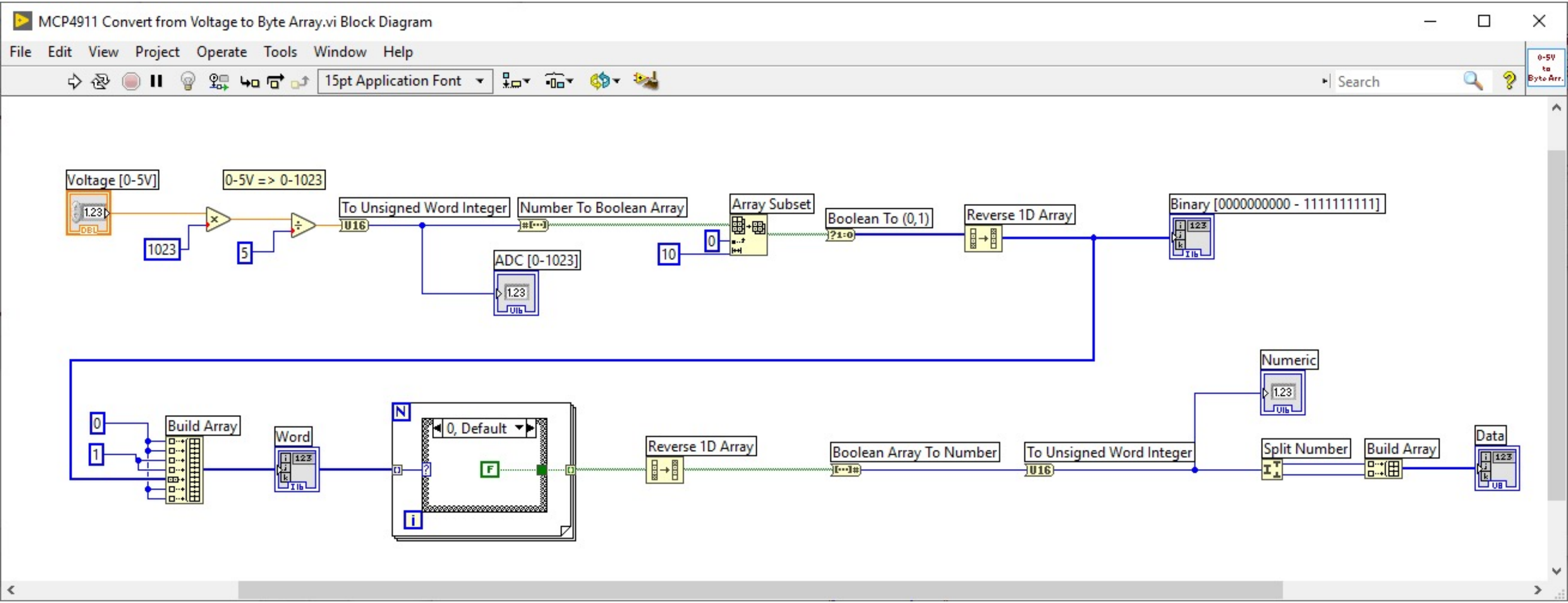


Basic SPI MCP4911 Example.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt App
Search
MCP 4911
Voltage [0-5V]
3.00
Data
0
57
152
CS Configuration
CS Channel
10
CS Logic Level
Active Low
SPI Channel (0)
0
COM9
Open.vi
Serial
SPI Open.vi
SPI Configure.vi
SPI Write Read.vi
1 Frame
Close.vi
Simple Error Handler.vi
SPI Configuration
Clock Frequency
1000000
Bit Order
MSb First
SPI Mode
Mode 0
Voltage [0-5V]
MCP4911 Convert from Voltage to Byte Array.vi
Data

Convert from Voltage to Byte Array

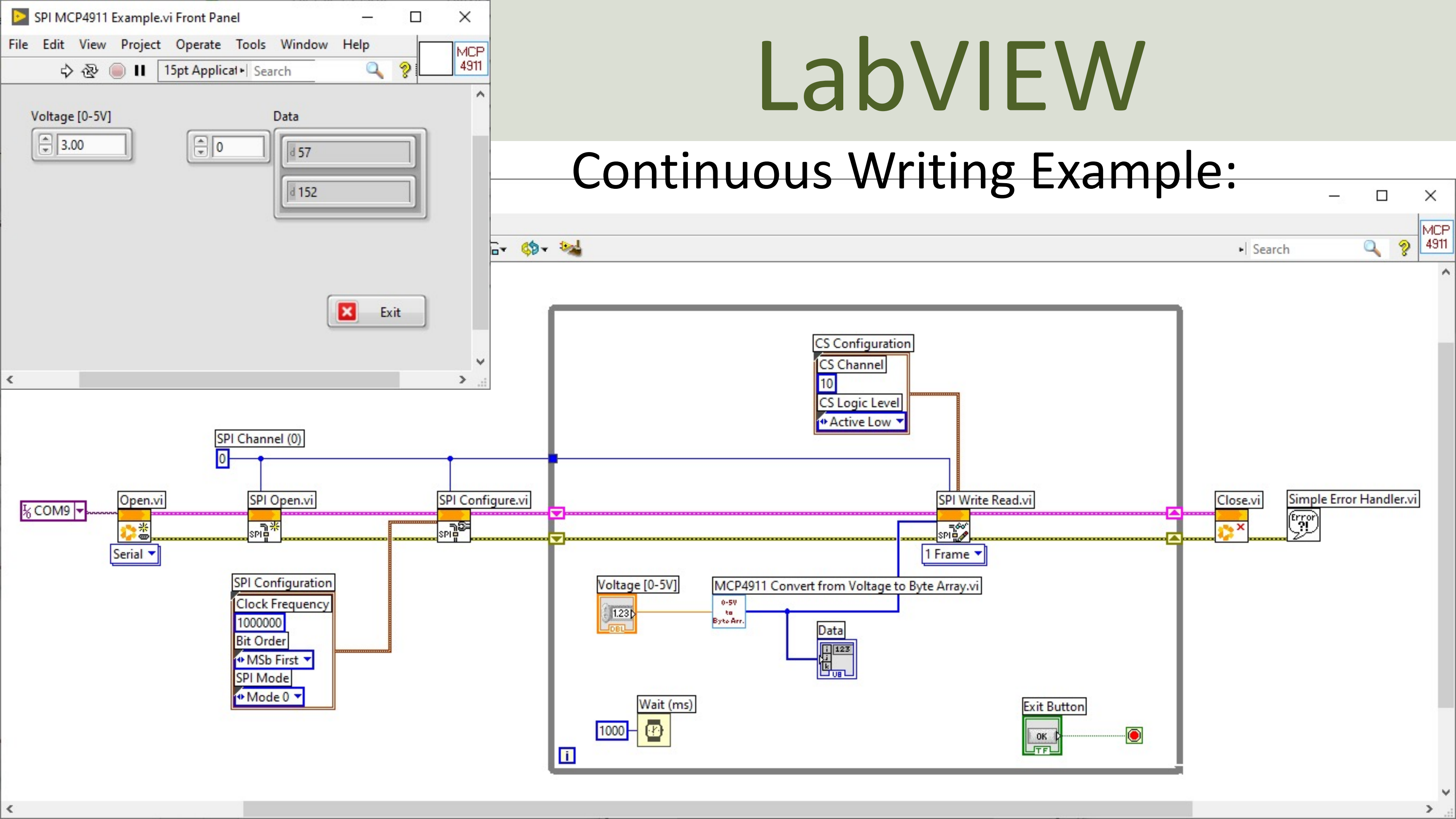


Convert from Voltage to Byte Array



LabVIEW

Continuous Writing Example:



Combined System

TC74 + MCP4911

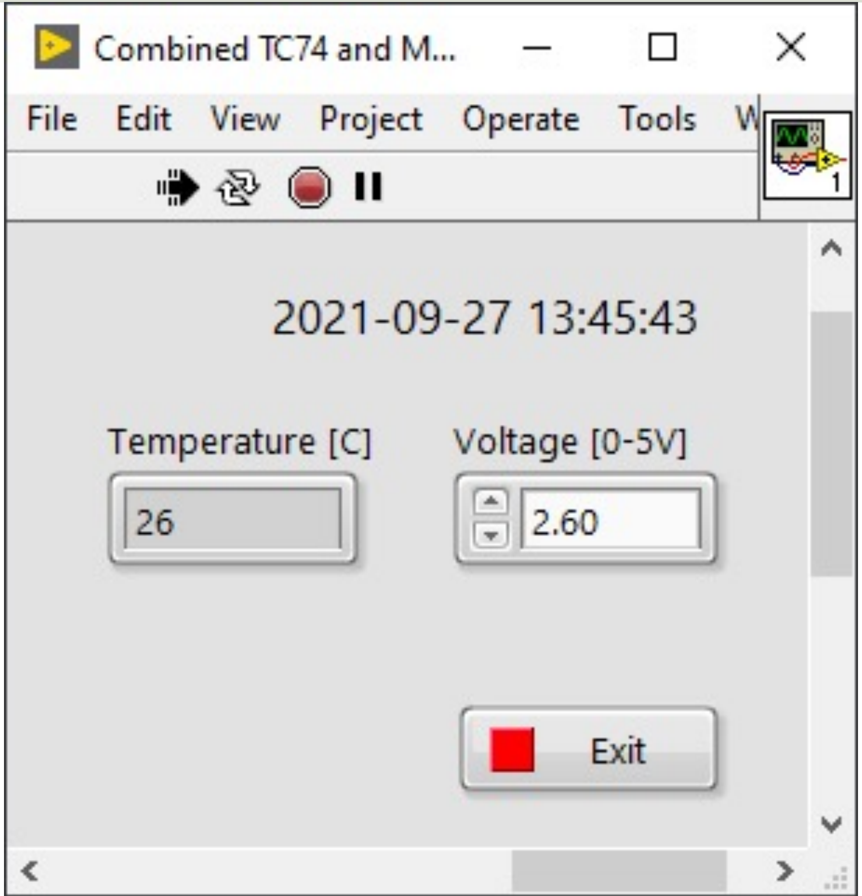
I2C

SPI

Hans-Petter Halvorsen

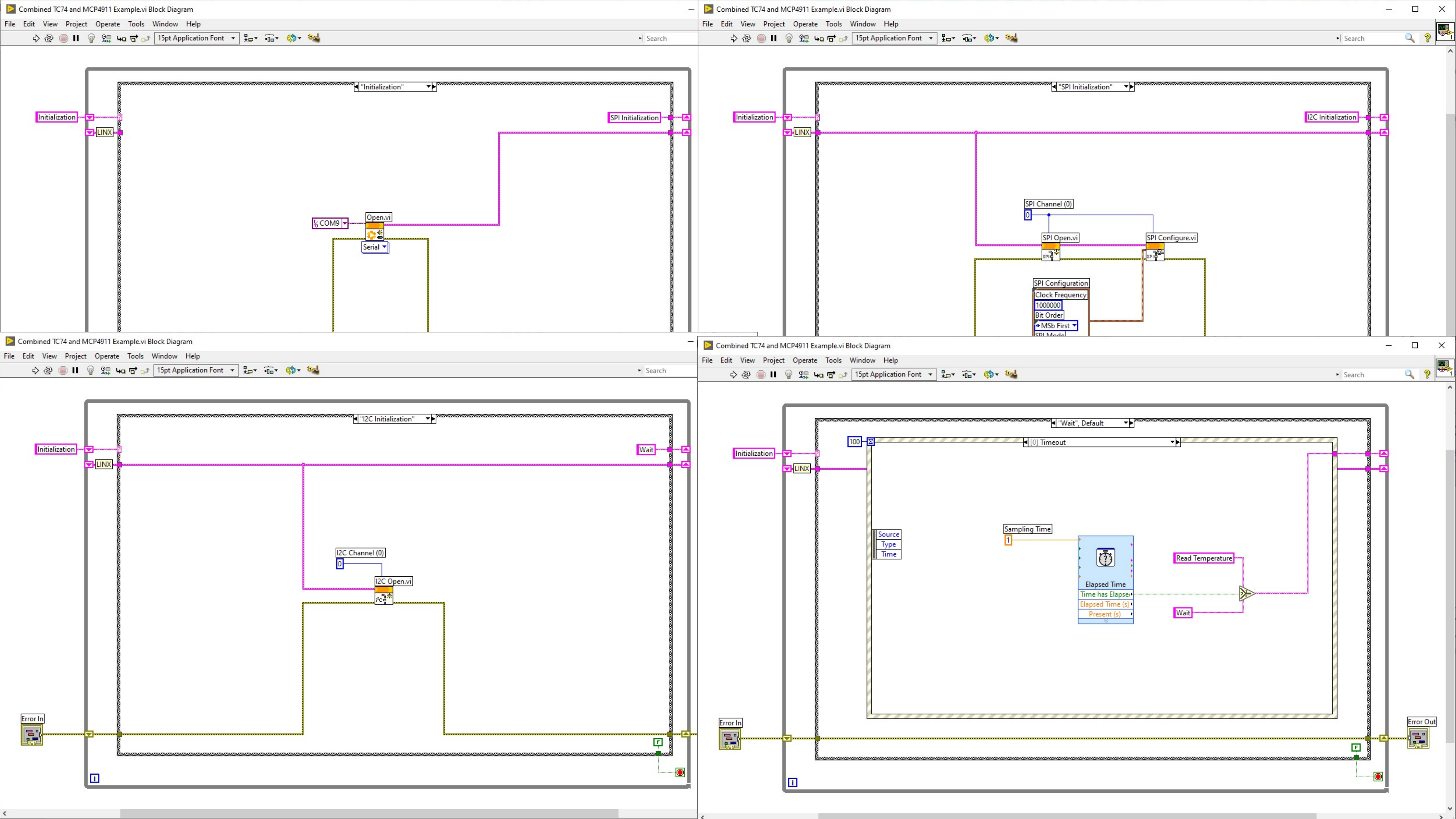
Table of Contents

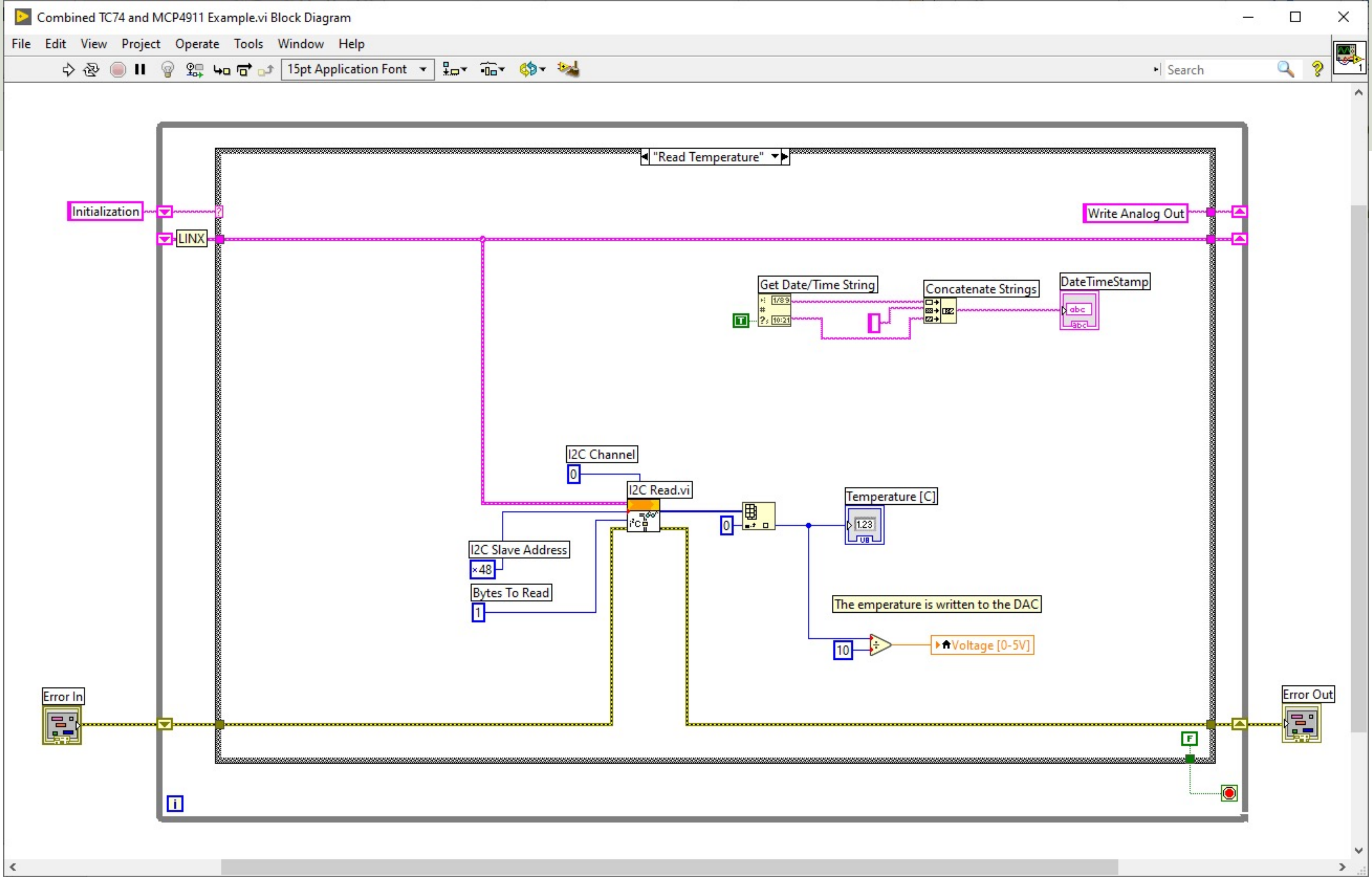
TC74 (I2C) + MCP4911 (SPI)

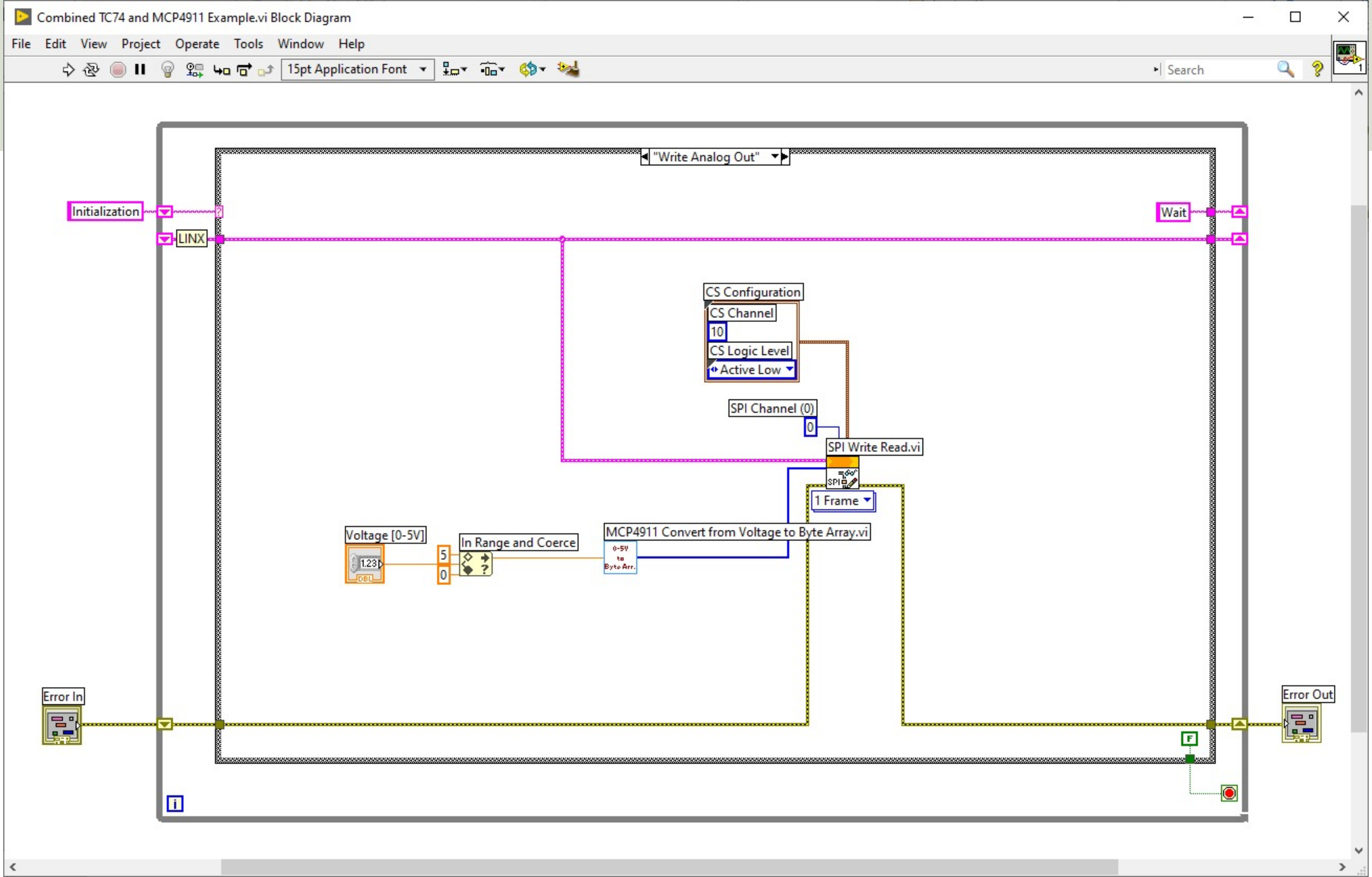


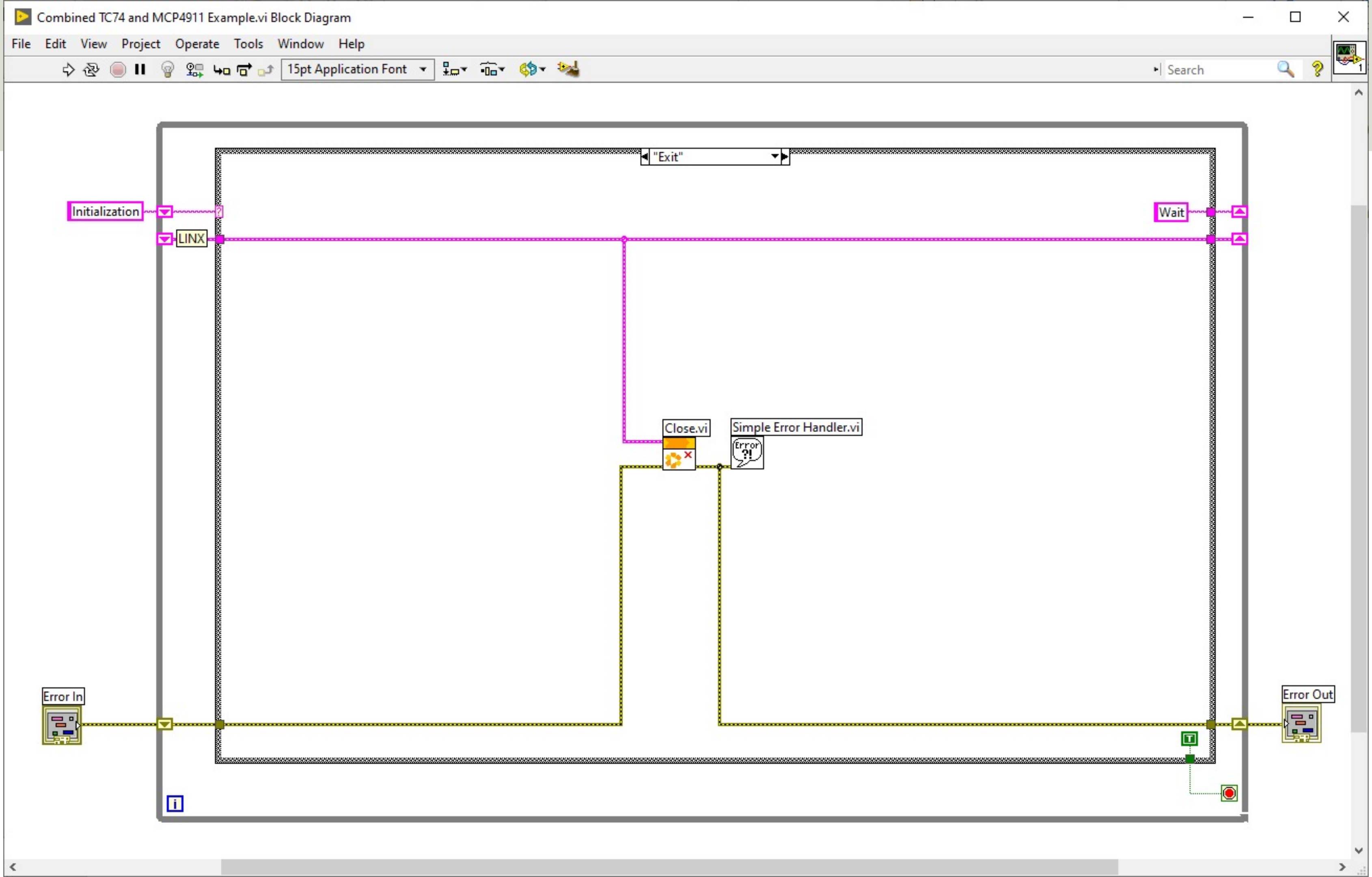
Here is a basic example presented where reading TC74 Temperature Data is combined with writing values to the MCP4911 DAC.

It can easily be extended with, e.g., a PID Control System


Combined TC74 and MCP4911 Example.vi Block Diagram
"Initialization"
Initialization
LINX
SPI Initialization
COM9
Open.vi
Serial
"SPI Initialization"
I2C Initialization
SPI Channel (0)
0
SPI Open.vi
SPI Configure.vi
SPI Configuration
Clock Frequency
1000000
Bit Order
MSb First
"I2C Initialization"
Wait
I2C Channel (0)
I2C Open.vi
Error In
"Wait", Default
100
[0] Timeout
Source
Type
Time
Sampling Time
1
Elapsed Time
Time has Elapsed
Elapsed Time (s)
Present (s)
Read Temperature
Error Out


Combined TC74 and MCP4911 Example.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
"Read Temperature"
Initialization
LINX
Write Analog Out
Get Date/Time String
Concatenate Strings
DateTimeStamp
I2C Channel
0
I2C Read.vi
Temperature [C]
I2C Slave Address
x48
Bytes To Read
1
The emperature is written to the DAC
10
Voltage [0-5V]
Error In
Error Out
F
i


Combined TC74 and MCP4911 Example.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
"Write Analog Out"
Initialization
LINX
Wait
CS Configuration
CS Channel
10
CS Logic Level
Active Low
SPI Channel (0)
0
SPI Write Read.vi
1 Frame
Voltage [0-5V]
1.23
In Range and Coerce
5
0
MCP4911 Convert from Voltage to Byte Array.vi
0-5V to Byte Arr.
Error In
Error Out
F
i


Combined TC74 and MCP4911 Example.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
"Exit"
Initialization
LINX
Wait
Close.vi
Simple Error Handler.vi
Error In
Error Out

Summary

- **TC74** Temperature Sensor with **I2C** Interface
- DAC **MCP4911** – Digital to Analog Converter with **SPI** Interface
- **Arduino** is a cheap open-source Microcontroller platform with Input/Output pins that can be used for many purposes like reading Sensor data, Datalogging, Internet of Things Applications, etc.
- Arduino and **LabVIEW/LabVIEW LINX** is a powerful combination
- If you don't have "LabVIEW Professional" Software, you may use the "LabVIEW Community Edition", which is free for non-commercial use.
- You then get a very low-cost DAQ/Datalogging System!

Table of Contents

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog