# Inter-Integrated Circuit (I2C) Communication

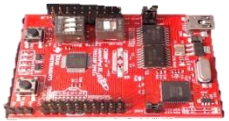
*Using the Texas Instruments F28027 LaunchPad and the Microchip "PICkit Serial I2C Demo Board" with the "MCP23008 8-Bit I/O Expander" & "24LC02B 2Kbit Serial EEPROM"*

sT-Embed Training

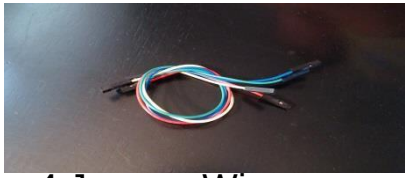
Ric Kolk
Altair Engineering
rkolk@altair.com

# Hardware Required:

Microchip “PICkit Serial I2C Demo Board” part number “PKSERIAL-I2C1”



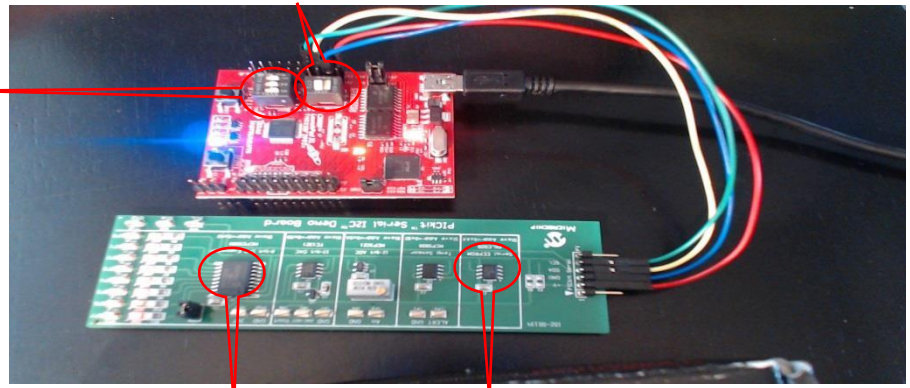
Texas Instruments LAUNCHXL-F28027 LaunchPad Board



4 Jumper Wires

Large switch switched to left

3 microswitches switched to right



MCP23008 8-Bit I/O Expander

24LC02B 2Kbit Serial EEPROM

F28027 Jumpers
J1 Jumpered
J2 Jumpered
J3 Not Jumpered

PICkit Jumpers
J1 Jumpered

Pin Connections

| PICKit | F28027 |
|---|---|
| +v | J1.1 |
| GND | J5.2 |
| SDA | J1.3 |
| SCL | J1.4 |

# I2C Basics

- Master – Slave Architecture:  One Master Device, Many Slave Devices
- All Devices are wired to the I2C Bus and powered by either 3.3v or 5v
- Two signals are used for communication:  (excluding ground & power)
  - SDA = Serial Data Line (can be input or output)
  - SCL = Serial Clock Line (the Master drives the clock line and the Devices read the clock line)
- Each Device has a 16 bit (2 Byte) HEX address, the rightmost bit is the “read/write” bit, 1/0 = read/write.
- sT-Embed uses a 15 bit decimal address, the rightmost “read/write” bit is deleted and the remaining 15 bit HEX address is converted to decimal to obtain the sT-Embed address for the Device.
  - Example:  Device address = 0x92 (HEX)
    Which is equal to 10010010
    sT-Embed deletes the rightmost bit; 01001001
    Which is equal to 49 in HEX
    Which is equal to 64+9 = 73 in Decimal
    sT-Embed uses 73 as the device address
- Communication always begins with a request from the Master
- Device communications are executed in a Top Down execution order in the sT-Embed model

# MCP23008 8-Bit I/O Expander Example

- The MCP23008 is an 8-bit I/O Expander.
- Refer to the MCP23008/MCP23S08 Data Sheet (DS21919) for complete information.

- The slave address is 0x40.

- The output of the MCP23008 drives LEDs DS1 through DS8.

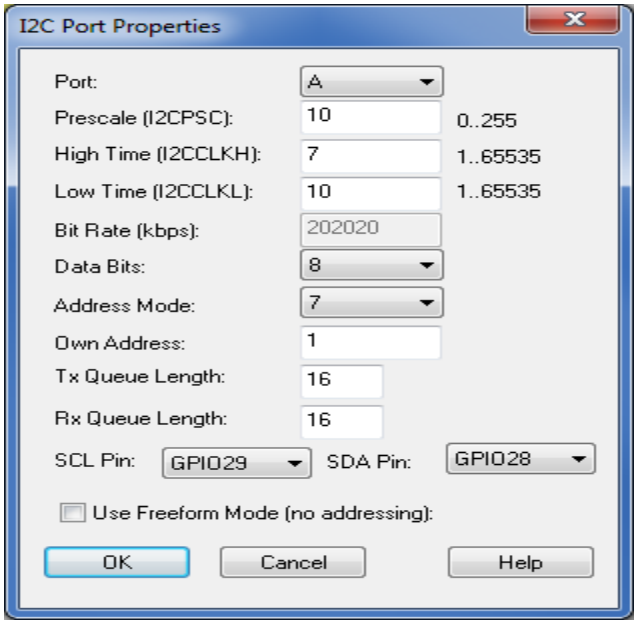
- The LEDs provide an easy to see indication of the MCP23008 operation.
- Jumper JP1 must be closed using a 2-pin shunt for the LEDs to operate.
- The LEDs can be disabled by removing JP1.

- The output of the MCP23008 is connected to test points GP0 through GP7 and GND. These test points can be monitored by a volt meter or connected to an external device. LEDs DS1 through DS8 can be used to monitor the output by closing JP1 with a 2-pin shunt or disable by removing JP1.

sT-Embed Model: (“Embedded/Examples/Piccolo/Launchpad”)
I2C-8-bit-xpander-28027.vsm

# I2C Configuration Block

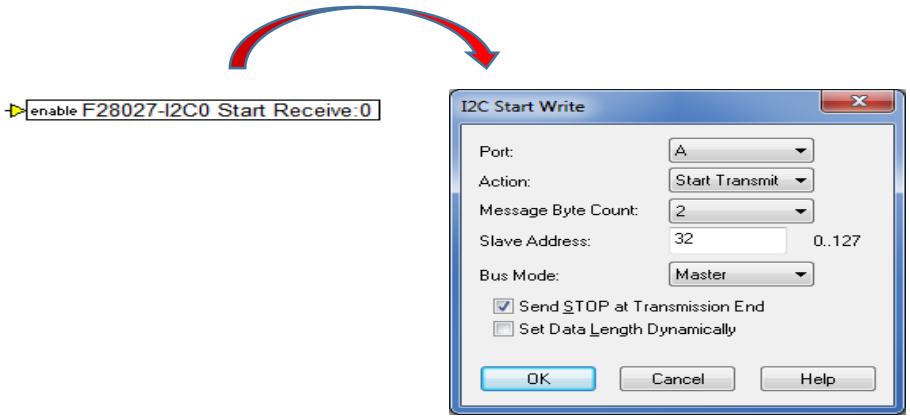
The “I2C Config” block is located under the “Embedded/Piccolo” menu.




- Select “Prescale (I2CPSC)” such that the “Bit Rate (kbps)” value is between 100,000 and 400,000. NOTE: This is the normal operating speed of the I2C bus.
- Select “Data Bits:” to the number of bits of data in your communication.
- Select “Address Mode:” to 7, uses a 7 bit address for the Slave
- Select “Own Address:” to the address you use for the Master device.
- Select the transmit Queue length “Tx Queue Length:” and the receive Queue length “Rx Queue Length” to the number of bits needed for both the address plus data.

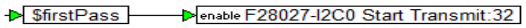
# Configure for Read or Write

Next we must identify the Device we wish to communicate with and the packet size.
Under “Embedded/Piccolo” select the “I2C Start Communication” block:



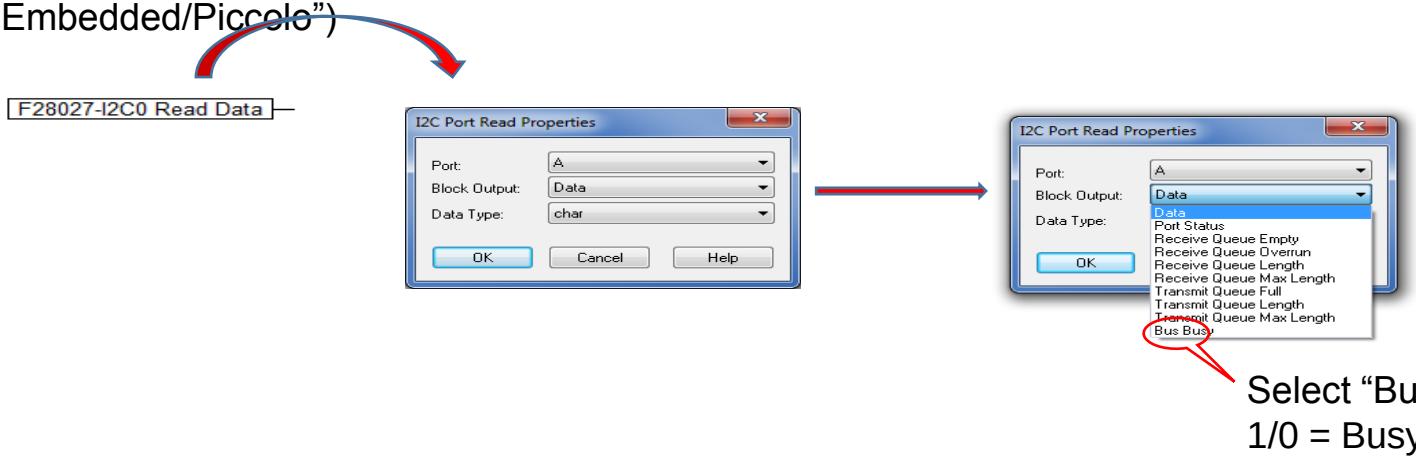

- Set the “Action:” to “Start Transmit” for writing and “Start Receive” for reading. This example will write to the Slave Device. And set the “Message Byte Count” to the packet size of the data in bytes (the address size is not included here).
- “Slave Address”: From the “PICkit Serial I2C Demo Board Users Guide”, the “MCP23008 8-Bit I/O Expander” address is 0x40 (HEX) = [0100 0000]
  - sT-Embed address = [0010 0000] = 20HEX = 32 Decimal
- Select “Master” or “Slave”, normally you are the “Master”.
- Check “Send STOP at Transmission End” is a good policy

Use the following sub-model to set the Read or Write Configuration:

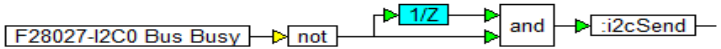


# Initiating Communication

Before any communication can occur, the I2C bus must not be busy.
In sT-Embed, you check whether the bus is busy using the “I2C Read Buffer” block (“Embedded/Piccolo”)




Once the“I2C Read Buffer” block is configured to output “Bus Busy”, the following model can be used to create the variable “:i2cSend”. “:i2cSend” turns and remains ON (value = 1) when the I2C bus becomes Not Busy.

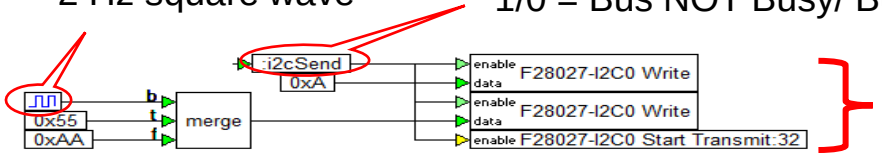


# Writing to the Device

There are 8 LED's on the PICkit. The following model alternately blinks LED 1,3,5,7 and LED 2,4,6,8 at a 2Hz rate.

2 Hz square wave

1/0 = Bus NOT Busy/ Busy




"Top Down" order of execution

Operation: (order of execution is top down)

- The upper "I2C Write Buffer" receives 0xA which equals [1010]. This is place in the TX Queue.
- The lower"I2C Write Buffer" receives 0x55 which equals [0101 0101] when the "b" input = 1 and receives 0xAA which equals [1010 1010] when the "b" input = 0.
- This is placed in the TX Queue.
- "I2C Start Write" is executed to transmit the TX Queue to the Slave Device

- The above process repeats after the "i2cSend" detects the bus is NOT busy (see previous page).

NOTE: This model, named "I2C-8-bit-xpander-28027.vsm" is located under the "Embedded/Examples/Piccolo/LaunchPad/"

# 24LC02B 2Kbit Serial EEPROM Example

The 24LC02B is a 2Kbit Serial EEPROM.

Refer to the 24AA02/24LC02B Data Sheet (DS21709) for complete information.

The slave address is 0xAX (where x = any value).

Data can be read or written to the 24LC02B.

Slave Address in sT-Embed:

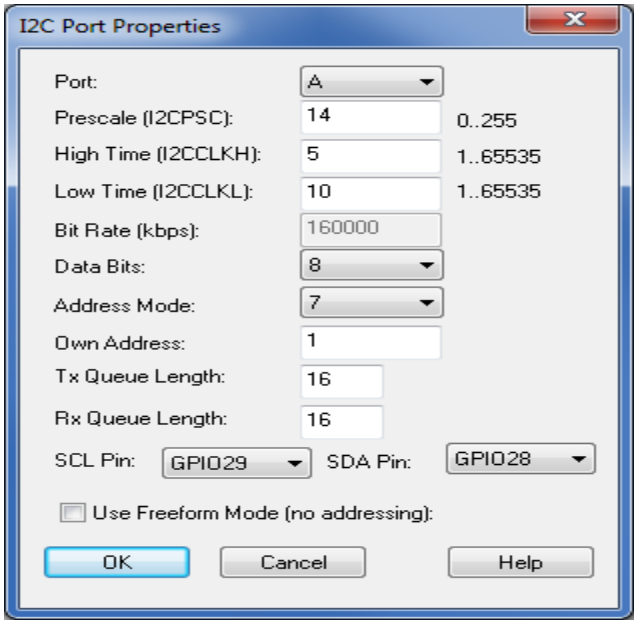
0xA0 HEX = [1010 0000]

sT-Embed discards the rightmost bit = [0101 0000] = 0x50 HEX = 80 Decimal

sT-Embed Model: (“Embedded/Examples/Piccolo/Launchpad”)
I2c-eepromReadWriteF28027.vsm

# I2C Configuration Block

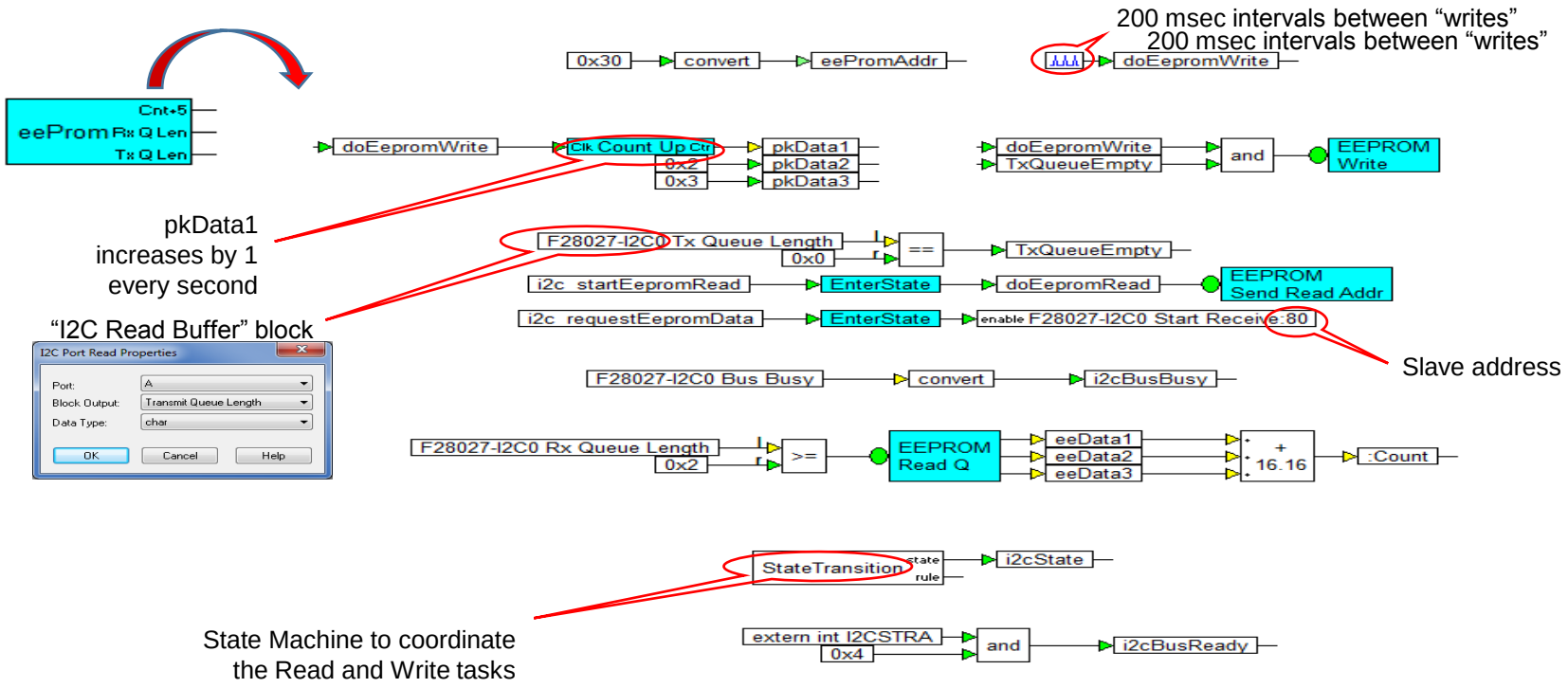
The “I2C Config” block is located under the “Embedded/Piccolo” menu.




- Select “Prescale (I2CPSC)” such that the “Bit Rate (kbps)” value is between 100,000 and 400,000. With “Prescale” = 14, the “Bit Rate” = 160kbps which is in range.
- Select “Data Bits:” to the number of bits of data in your communication.
- Select “Address Mode:” to 7, uses a 7 bit address for the Slave
- Select “Own Address:” to the address you use for the Master device.
- Select the transmit Queue length “Tx Queue Length:” and the receive Queue length “Rx Queue Length” to the number of bits needed for both the address plus data.

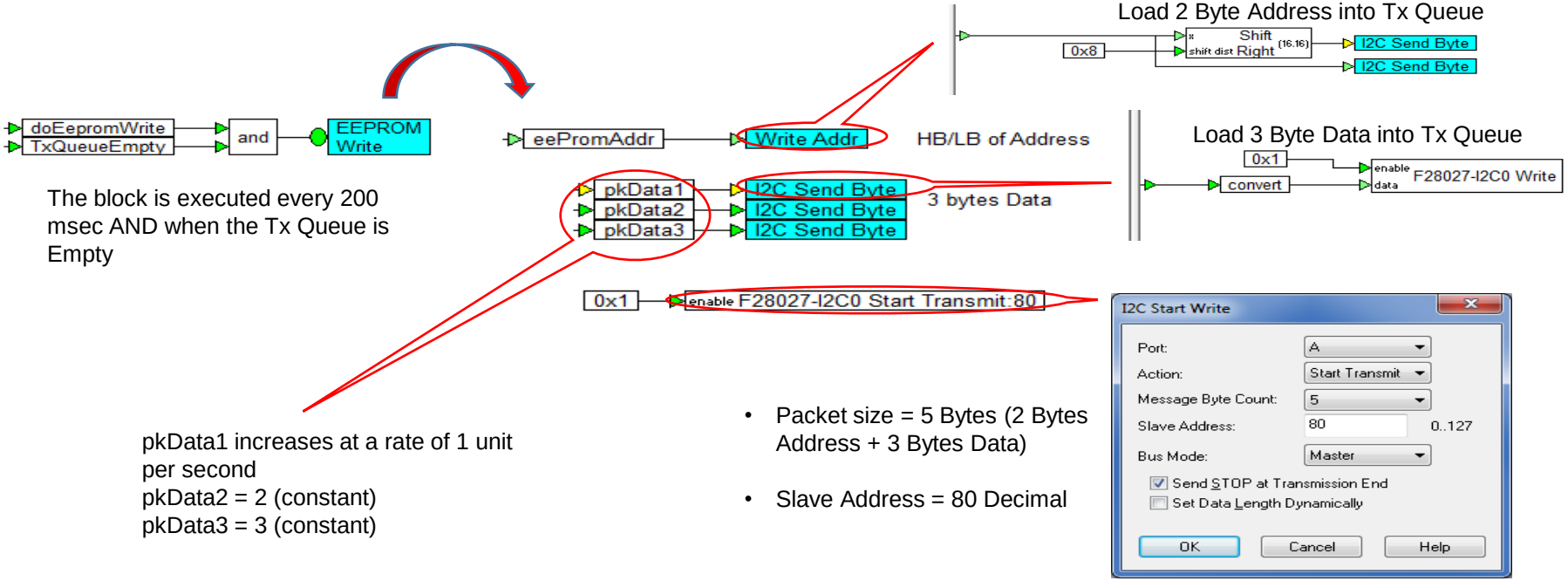
# sT-Embed Model – Top Level

The sT-Embed Read/Write EEPROM model is captured in the Compound Block “eePromRx”;



# sT-Embed Model – EEPROM Write Address & Data

## Operation of the “EEPROM Write” Compound block



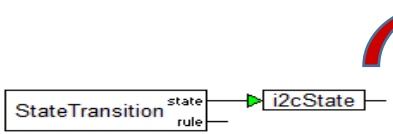

The block is executed every 200 msec AND when the Tx Queue is Empty

pkData1 increases at a rate of 1 unit per second
pkData2 = 2 (constant)
pkData3 = 3 (constant)

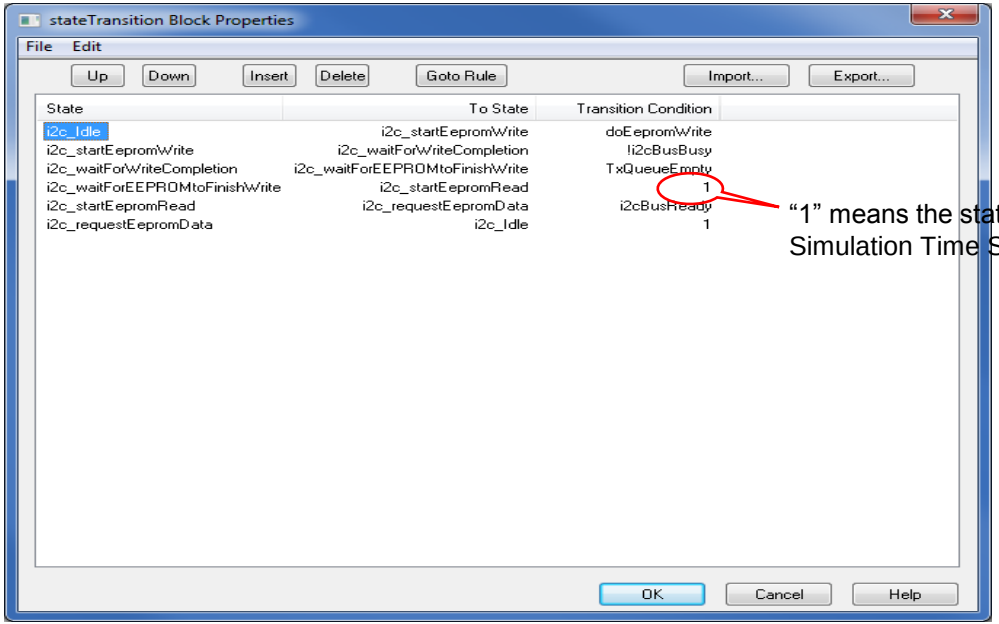
- Packet size = 5 Bytes (2 Bytes Address + 3 Bytes Data)
- Slave Address = 80 Decimal

# sT-Embed Model – State Machine

The “i2cState” variable is determined by a sT-Embed State Machine:



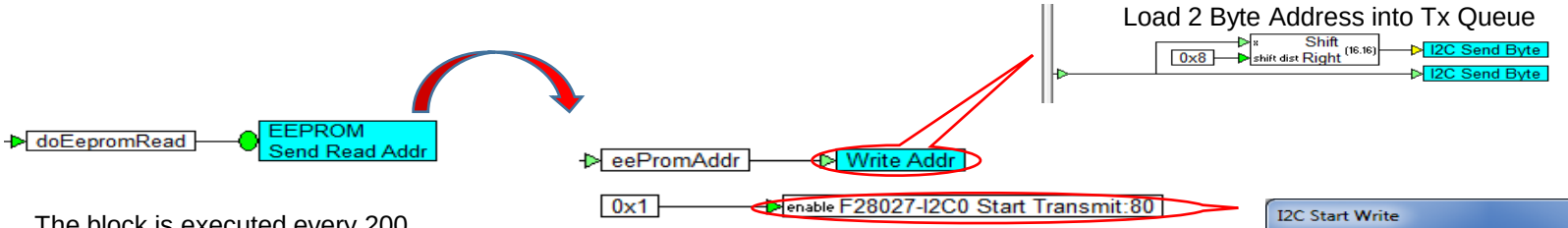

Executed at the Simulation Time Step interval




“1” means the state transitions in 1 Simulation Time Step

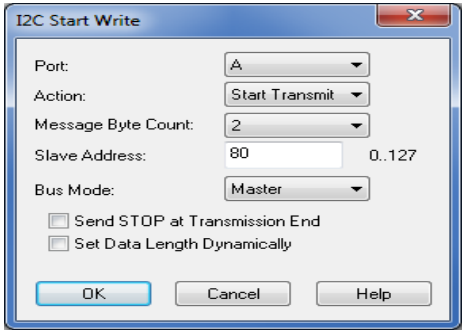
# sT-Embed Model – EEPROM Read Address

## Operation of the “EEPROM Read” Compound block




The block is executed every 200 msec AND when the Tx Queue is Empty

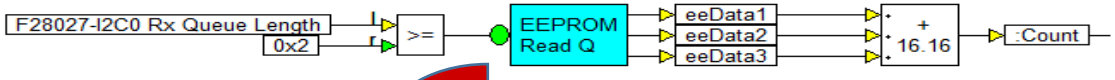
- Packet size = 2 Bytes (address only)
- Slave Address = 80 Decimal



# sT-Embed Model – EEPROM Read Data

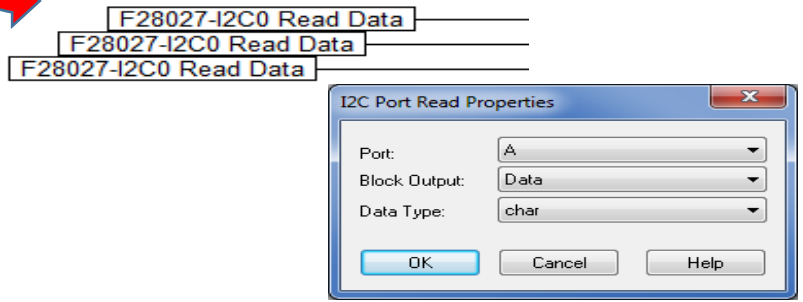
Operation of the “EEPROM Read” Compound block

The block is executed when the Rx Queue length >= 2

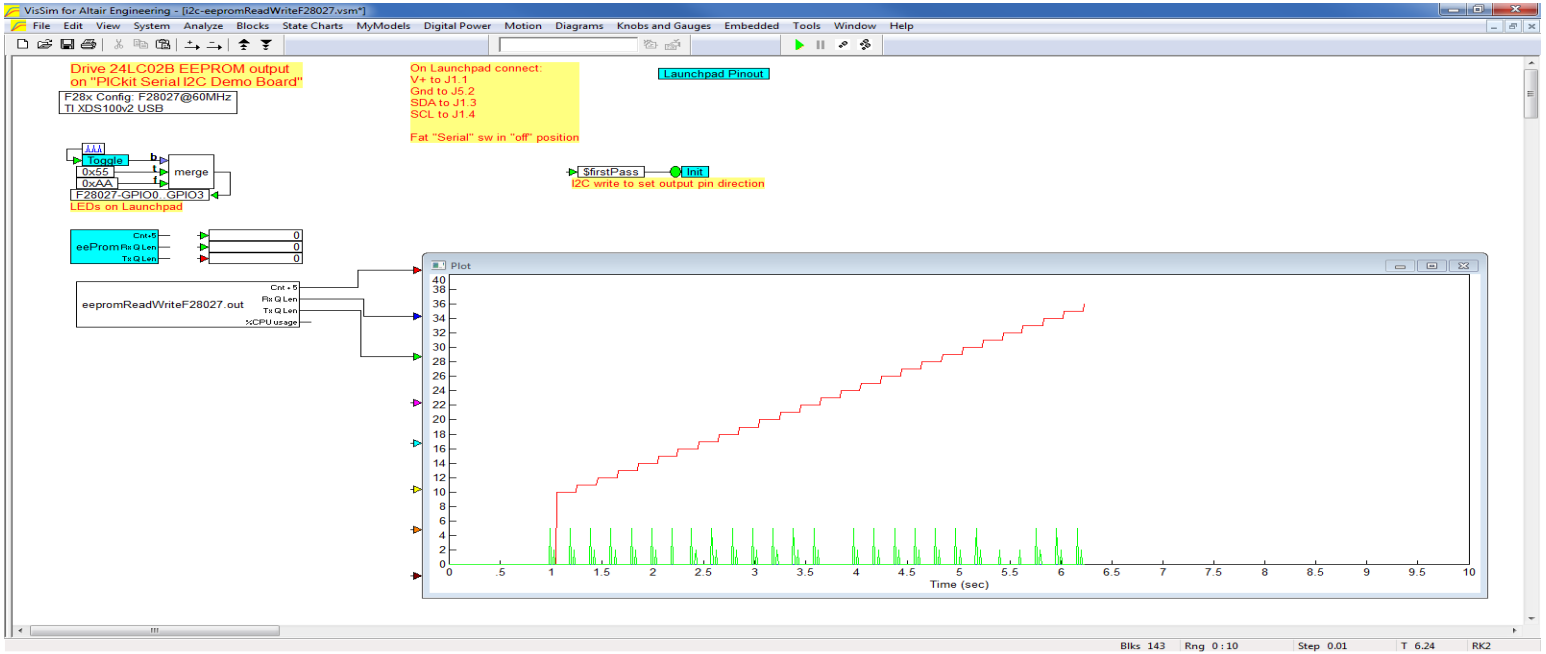



Count begins at 10 and then increments by 1 each second.

eeData1 = pkData1 increases at a rate of 1 unit per second
eeData2 = pkData2 = 2 (constant)
eeData3 = pkData3 = 3 (constant)



# sT-Embed Model – Results



# Additional sT-Embed Example Files

All files are located in the ““Embedded/Examples/Piccolo/LaunchPad/” folder:

- I2C-8-bit-expender-28027.vsm
- I2C-eepromReadWrite28027.vsm
- I2C-MCP3221-ADC-28027.vsm
- I2C-MCP9800-temp-sensor-28027.vsm

# End of Section

EMBED™
solidThinking