AxIO16 EVM User's Guide

IMPORTANT NOTICE

This EVM is provided under the following conditions:
This evaluation kit is for use of ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY. It is provided with a sample integrated circuit mounted to a PCB facilitating connectivity to inputs, outputs and power. This EVM may be used with any development system or other source of I/O signals by connecting it to a host controller or other protocol generating device. This EVM is not a Reference Design. It is not intended to represent a complete circuit design for any particular function or application use. The EVM construction has exposed circuity and is susceptible to a potentially damaging environment. It is the responsibility of the user to take all appropriate precautions for electrostatic discharge prevention.

1 Introduction:

This document describes the features and functionality of the AxIO16 Evaluation board populated with either an APIO16, or an AFIO16. Both devices are radiation hardened by design, level translating, I^2C, SMBUS, SPI integrated IO expanders that are ideally suited for space and other applications requiring high reliability and radiation tolerance for both SEE and TID.

The AxIO16 EVM was developed to allow maximum flexibility for evaluation using a variety of host controllers. Examples for connecting both SPI and I^2C to a Raspberry Pi 4b, and Total Phase Aardvark or Cheetah host adapters are provided.

2 Additional resources:

See Apogee APIO16 product folder: https://apogeesemi.com/products/apio16/

Or scan this QR code

3 Handling requirements:

Caution: This device is sensitive to Electrostatic Discharge (ESD). Proper ESD handling should be used to prevent damage to the integrated circuit.

4 AxIO16 EVM

4.1 AxIO16 Features

- 16 bits of GPIOs available on 2-8bit ports P0 and P1.
- SPI or I²C protocols supported with mode selection jumper
- Jumper selectable biasing to VSS or VCCP on Port 0 on a per bit basis.
- Jumper selectable biasing on port 1 to connect LED, and/or loopback connections to Port 0 on a per bit basis.
- User selectable address jumpers allowing I²C addresses from 0x20 to 0x2F (16 unique addresses on same bus)
- Unpopulated capacitor footprints on SDA/SCL for testing various I²C capacitive bus loadings.
- Jumper selectable for on board pullups for SDA and SCL I²C lines.
- INTb Interrupt LED that can be enabled to visually indicate input port state changes, or header can be used to provide INTb connectivity to host controller.
- Single momentary switch that can be used to trigger a host via GPIO.
- Breakout header for all device pins on standard bread board pitch to facilitate prototyping/testing.
- Auxiliary header for address and protocol pins to allow custom plug in cable connectivity.
- Single jumper option to join VCCP/VCCI to simplify single supply usage.

4.1 EVM image

Figure 1 shows the top of the board with components and jumpers.

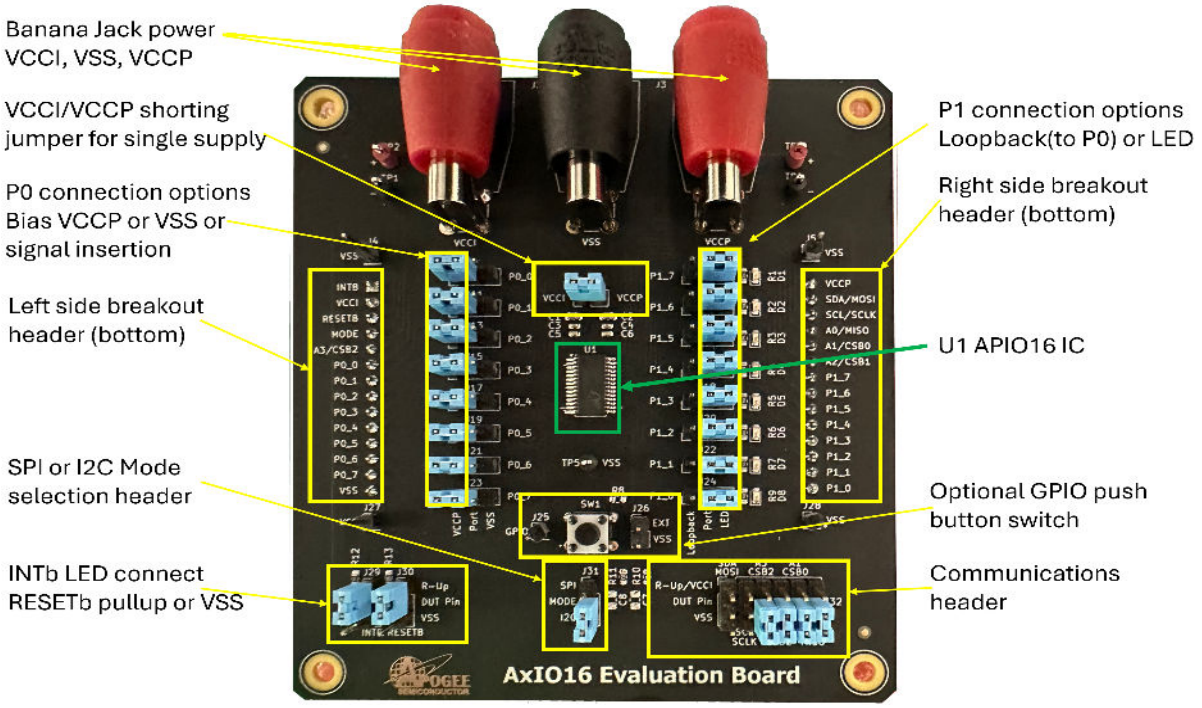

Figure 1 AxIO16 EVM

4.2 EVM power:

J1, J2, and J3 are banana jacks that can be used to provide power to the EVM. J1, and J3 are VCCI, and VCCP respectively. J2 provide VSS for board ground. Alternatively, power can be inserted via test point TP2, TP3. Additionally, when operating with a single supply voltage, power can be applied to any of the J1,J3, TP2, TP3 with the shorting jumper J8 for VCCP to VCCI inserted.

4.4 Breakout headers:

Two bottom side headers are provided at 100mil pitch spacing that represent a 1 to 1 breakout for device pins. J6 exposes pins 1-14, and J7 exposes 15 to 28. This header is spaced to allow insertion into prototyping board or bread board.

4.5 Communications header:

J32 is provided in a 3x6 pin array to allow either signal connectivity with ground, or pullup to VCCI for all the communication signals used for SPI or I^2C control. This provides the capability to bias any of the signals high with resistor pullup, tie to VSS, or provide the signal with ground from host. This connectivity allows for changing I^2C target address or changing which CSBx signal is provided by host for SPI transactions.

If necessary, it is possible to optimize the signal integrity for high speed SPI usage, the signal integrity of SCLK, CSBx, MISO, and MOSI, can be improved with the ability to remove the stubs from the breakout header to J32 . This can be achieved by removing the zero-ohm jumpers the bottom side board. Removal of zero-ohm resistors R14 through R19 will disconnect the following signals from the J32 header respectively. R14 MOSI, R15 SCLK, R16 CSB2, R17 CSB1, R18 CSB2, and R19 disconnects MISO. When disconnected, these signals need to be provided from/to the breakout headers J6 and J7.

4.6 Port Configuration headers:

4.6.1 P0 Configuration header:

Port 0 configuration header has jumpers set to use P0 as an input and by default biased to VCCP. Manually, each port bit can be biased with the jumper to either VCCP, or VSS. Alternatively, a two pin signal cable (signal and vss) can be used to connect to any of the port 0 pins to signal source or destination.

4.6.2 P1 Configuration header:

Port 1 configuration header has jumpers set to use P1 as an output with LEDs connected by default. Manually, each port bit can set with jumper to enable loopback to its equivalent bit on P0, or any bit can be connected to LED for visual representation of signal state. Note, when operating at min VCC, the LED drivers may not have sufficient voltage to illuminate. Operating above ~2.5V will provide sufficient voltage to visually see the LEDs.

4.7 INTb and RESETb header

4.7.1 INTb

J29 header is a 3 pin header with center pin being the INTb signal. Connecting jumper between pins 1 and 2 will connect the INTb open drain output to an LED. The LED will illuminate when any input port changes state. This only applies after it was read at least once after power up. See datasheet for more information. J29 can also be used with pins 2 and 3 to send signal to a host controller to monitor for input state changes.

4.7.2 RESETb

J30 head is a 3 pin header for the RESETb input. Jumper between pin 1 and 2 will connect a 49.9k ohm pull up resistor to VCCI. This is the recommended setting. Alternatively, the RESETb signal can be driven from a host or other reset source.

4.8 SW1 GPIO

J25, SW1, and J26 are provided to allow option on configuring a push button switch. When SW1 is pressed, J25 will be connected to VSS. The circuit has a no populated resistor to J26 that can be used to implement a pull up to an alternate supply for level translating requirements or if the destination requires pullup.

5 AxIO16 EVM Schematic and Jumper configuration.

5.1 AxIO16 Schematic

Figure 2 shows the schematic of the AxIO16 EVM.

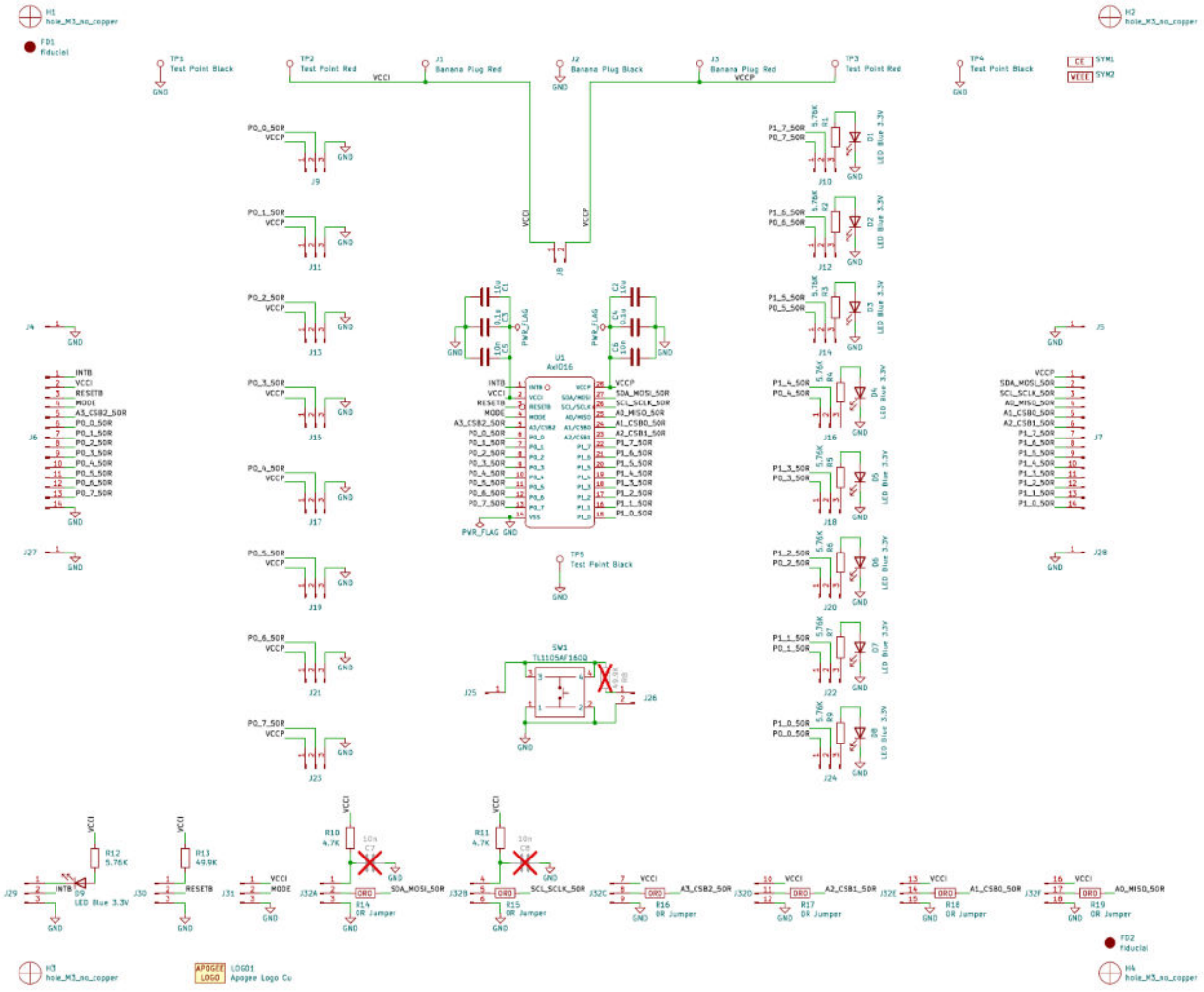

Figure 2 AxIO16 EVM Schematic

5.2 AxIO16 Jumper Configuration

Table 1 contains the descriptions of all jumpers and their supplied default configuration

Jumper	Function	Description	Type	Default Jumper	Default Function
J1	VCCI banana jack	Supply for Interface	Power	n/a	n/a
J2	VSS banana jack	Device Ground	Power	n/a	n/a
J3	VCCP banana jack	Supply for Ports	Power	n/a	n/a
J4,5,27,28	VSS test points	Device Ground	Power	n/a	n/a
J6	Bottom Side header	Pins 1 through 14	*all	n/a	n/a
J7	Bottom Side header	Pins 15 through 28	*all	n/a	n/a
j8	VCCP/VCCI shorting jumper	Single supply usage	Power	1:2	Single Supply
j9	P0_0 Port 0 bit 0	3 pin header enabling Port 0 bit biasing to VSS or VCCP	Signal	2:3	Port 0 bits connected to VSS
j11	P0_1 Port 0 bit 1		Signal	2:3	
j13	P0_2 Port 0 bit 2		Signal	2:3	
j15	P0_3 Port 0 bit 3		Signal	2:3	
j17	P0_4 Port 0 bit 4		Signal	2:3	
j19	P0_5 Port 0 bit 5		Signal	2:3	
j21	P0_6 Port 0 bit 6		Signal	2:3	
j23	P0_7 Port 0 bit 7		Signal	2:3	
j10	P1_7 Port 1 bit 7	3 pin header enabling Port 1 bit connectivity to Loopback to port 0, or to enable LED	Signal	2:3	Port 1 bits connected via loopback to Port 0 bits
j12	P1_6 Port 1 bit 6		Signal	2:3	
j14	P1_5 Port 1 bit 5		Signal	2:3	
j16	P1_4 Port 1 bit 4		Signal	2:3	
j18	P1_3 Port 1 bit 3		Signal	2:3	
j20	P1_2 Port 1 bit 2		Signal	2:3	
j22	P1_1 Port 1 bit 1		Signal	2:3	
j24	P1_0 Port 1 bit 0		Signal	2:3	
j25	GPIO signal	Single pin that shorts to VSS when SW8 pressed. Intended to allow interrupt to host controller	Signal	n/a	
j26	GPIO biasing jumper	Allows provision of pullup to unique voltage for SW8 using DNI R8	Signal	n/a	
j29	INTb	INTb signal can be connected to LED or connection to feed to host.	Signal		
j30	RESETb	Enables connection of pullup to VCCI, or short to VSS, or driven from host.	Signal	1:2	RESETb Pullup enabled to VCCI
j31	MODE selection	Allows biasing MODE pin to VCCI, or VSS	Signal	2:3	I2C enabled
j32	6x3 header for interface signals	Allows for signal connections, and static biasing for interface signals			
J32-A	SDA/MOSI	Option for signal insertion, or pull up to VCCI	Signal	n/a	
J32-B	SCL/SCLK	Option for signal insertion, or pull up to VCCI	Signal	n/a	
J32-C	A3/CSB2	Option for signal, or pull up/down to VCCI/VSS	Signal	2:3	A3 = Low I2C address 0x20
J32-D	A2/CSB1	Option for signal, or pull up/down to VCCI/VSS	Signal	2:3	A2 = Low I2C address 0x20
J32-E	A1/CSB0	Option for signal, or pull up/down to VCCI/VSS	Signal	2:3	A1 = Low I2C address 0x20
J32-F	A0/MISO	Option for signal, or pull up/down to VCCI/VSS	Signal	2:3	A0 = Low I2C address 0x20

Table 1 EVM Jumper Configuration

6 Operation of the AxIO16

Caution: It is possible to configure the board in a manner that will short P0 to P1 outputs. Use caution to ensure that ports are not in conflict when using the loopback connection from P1 to P0.

This EVM does not include pullup or pulldown resistors for the ports. Care should be used to not leave inputs floating. After power up, configure P1 as outputs when using the LED connections. Otherwise there is potential for the LED and resistor parasitics to pull the P1 inputs into the intermediate voltage range that could cause excessive through current for port pins in input mode.

6.1 I²C mode

Operating the AxIO16 in I²C mode is straight forward. The EVM is shipped by default with I²C mode configured for I²C address 0x20. The default address can be readily changed by adjusting jumpers J32C,D, F. These jumpers can be set to VCCI or VSS to change the address to one of 16 possible addresses. See datasheet.

Connect SDA, SCL, and VSS to host controller. Connect SDA to J32A, and SCL to J32B. VSS can be connected from various locations on the board to the ground on the host controller. Connect VCCI and VCCP to appropriate power. Or using default jumper for J8 with VCCI and VCCP shorted together for single supply usage. This has been tested with a Raspberry Pi 4b running python code along with a TotalPhase Aardvark protocol device.

Python code for exercising device with I²C utilizing an Raspberry Pi 4b running Raspberry Pi OS. The below code assumes P0 biased with any pattern, and P1 set to drive LEDs.

```
import smbus
from time import sleep
#  AxIO16 Register Map
#  Reg# : name                : access     : default  (reset)
#  0x00 input port 0          : read only  : xxxxxxxx (unknown)
#  0x01 input port 1          : read only  : xxxxxxxx (unknown)
#  0x02 output port 0         : r/w        : 11111111 (out hi)
#  0x03 output port 1         : r/w        : 11111111 (out hi)
#  0x04 Polarity port 0       : r/w        : 00000000 (no inv)
#  0x05 Polarity port 1       : r/w        : 00000000 (no inv)
#  0x06 Configuration port 0 : r/w        : 11111111 (input)
#  0x07 Configuration port 1 : r/w        : 11111111 (input)

# Rpi setup/configuration.
# Rpi channel
channel = 1
# EVM I2C address.  A0:A3 = Low
address = 0x20
# set bus definition for Rpi
bus = smbus.SMBus(channel)

# Code to write single byte to Rpi I2C channel
```

```
# bus.write_byte_data(address, register, value)

# Code to read single byte to Rpi I2C channel
# val = bus.read_byte_data(address, register)

#Set P0 as input (this is default after power up or reset)
bus.write_byte_data(address, 0x06, 0xff)
#Set P1 as output.  LED jumpers intended to be set on P1 pins.
bus.write_byte_data(address, 0x07, 0x00)

# Read jumper setting on P0
val = bus.read_byte_data(address, 0x00)
print ("P0 read:", val)

# The following while loop toggles LEDs in blinking pattern.
while(True)
       bus.write_byte_data(address, 0x03, 0xaa)
       sleep(delay)
       bus.write_byte_data(address, 0x03, 0x55)
       sleep(delay)
```

TotalPhase Aardvark example.

Connect SDA, SCL, and VSS to Total phase header pins SDA, SCL, and GND respectively.

Using Control Center GUI, connect to Aardvark in batch mode. Load and execute the following:

```
<adapter>
<!---->
<!--Script for setup P0 input, P0 output with simple read of P0 and LED output to P1-->
<configure i2c="1" spi="0" gpio="0" tpower="0" pullups="0"/>
<i2c_bitrate khz="100"/>
<!—Set P0 to input mode-->
<i2c_write addr="0x20" count="2" nostop="0" ten_bit_addr="0" combined_fmt="0" radix="16"> 06 FF
</i2c_write>

<!—Set P1 to output mode-->
<i2c_write addr="0x20" count="2" nostop="0" ten_bit_addr="0" combined_fmt="0" radix="16"> 07 00
</i2c_write>

<!—Read P0-->
<i2c_write addr="0x20" count="2" nostop="0" ten_bit_addr="0" combined_fmt="0" radix="16"> 00 00
</i2c_write>

<!—Write P1-->
<i2c_write addr="0x20" count="2" nostop="0" ten_bit_addr="0" combined_fmt="0" radix="16"> 03 aa
</i2c_write>
<sleep ms="100"/>
<i2c_write addr="0x20" count="2" nostop="0" ten_bit_addr="0" combined_fmt="0" radix="16"> 03 55
</i2c_write>
<adapter>
```

6.2 SPI mode

SPI mode can be configured by setting the mode pin to VCCI using J31 and connecting the protocol signals. SCLK, MOSI, MISO, and CSBx need to be connected to their respective signals on the host controller. For demonstration purposes, 2 of the 3 CSBx signals need to be biased to the active state. All three of the CSBx signals need to be active low simultaneously. Thus, connect one of the

3 CSBx signals to host controller SS or CS line. The other 2 CSBx signals need to have their respective jumper on J32 set to VSS.

Python Raspberry Pi 4b SPI test.

```
import spidev
from time import sleep
#  AxIO16 Register Map
#  Reg# : name                : access     : default  (reset)
#  0x00 input port 0          : read only  : xxxxxxxx (unknown)
#  0x01 input port 1          : read only  : xxxxxxxx (unknown)
#  0x02 output port 0         : r/w        : 11111111 (out hi)
#  0x03 output port 1         : r/w        : 11111111 (out hi)
#  0x04 Polarity port 0       : r/w        : 00000000 (no inv)
#  0x05 Polarity port 1       : r/w        : 00000000 (no inv)
#  0x06 Configuration port 0 : r/w        : 11111111 (input)
#  0x07 Configuration port 1 : r/w        : 11111111 (input)
Rpi setup/configuration.
# Rpi channel
spi = spidev.SpiDev()
spi.open(0,0) # open spi bus 0, device 0
spi.max_speed_hz = 25000000

def SPI_Write(register, value):
    # Create the command byte for the write operation
    # Bits 7:5 = register (0-7)
    # Bit 4 = 1 for write
    # Bit 3 = 0 for read or write operation
    # Bits 2:0 = don't care (set to 000)
    command_byte = (register << 5) | (1 << 4) | (0 << 3)  # Read/Write set to 1, Bit 3 set to 0
    data_byte = value  # Data to write
    #print(register, value, command_byte, data_byte)

    # Send the command byte followed by the value byte
    response = spi.xfer2([command_byte, data_byte])
    return response  # You can handle the response if needed

def SPI_Read(register):
    # Create the command byte for the read operation
    # Bits 7:5 = register (0-7)
    # Bit 4 = 0 for read
    # Bit 3 = 0 for read or write operation
    # Bits 2:0 = don't care (set to 000)
    command_byte = (register << 5) | (0 << 4) | (0 << 3)  # Read/Write set to 0, Bit 3 set to 0

    # Send the command byte and receive the response
    response = spi.xfer2([command_byte, 0])  # Send 0 as dummy data byte for read
    return response[1]  # Return the second byte (data from register)

#Set P0 as input (this is default after power up or reset)
SPI_Write(6, 0xff)
#Set P1 as output.  LED jumpers intended to be set on P1 pins.
SPI_Write(7, 0x00)
# Read jumper setting on P0
val = SPI_Read(0x00)
print ("P0 read:", val)
# The following while loop toggles LEDs in blinking pattern.
while(True)
       SPI_Write( 3, 0xaa)
       sleep(delay)
       SPI_Write( 3, 0x55)
       sleep(delay)
```

TotalPhase Aardvark example SPI operation.

Connect SCLK, MOSI, MISO, CSB(1,2,or 3) and VSS to Total phase header pins SCLK, MOSI, MISO, SS and GND respectively.

Since the SPI protocol is not 1 to 1 mapping of the 8 bit command to the register, some bit manipulations need to be performed to access the AxIO16 device.

The 8 bit command is comprised of 3 bits of command C2,C1,C0, followed by a single write/read bit, then followed by a 0 and then 3 don't care bits. See datasheet for further definition.

For example, a write to register 3 would be comprised of C2, C1, C0 = 111, write bit 1, followed by 0XXX. This would resolve to 1111 0XXX or simply F0.

Using Control Center GUI, connect to Aardvark in batch mode. Load and execute the following:

```
<adapter>
<!---->
<!--Script for setup P0 input, P0 output with simple read of P0 and LED output to P1-->
<!—Aardvark max speed for SPI is 8Mhz -->
<configure i2c="0" spi="1" gpio="0" tpower="0" pullups="0"/>
<spi_bitrate khz="8000"/>

<spi_config polarity="rising/falling" phase="sample/setup" bitorder="msb" ss="active_low"/>

<!--F0 write P1 as outputs. 3 address bit (reg 7 111) followed by r/w bit "1", leading to F. Next
bit must be 0, followed by 3 don't cares resulting in F0 00 to set output port register 7 to all
0's-->
<!—Set P0 to input mode reg 6 binary - 1101 0000 - D0 -->
<spi_write count="2" radix="16" > D0 FF </spi_write>

<!—Set P1 to output mode reg 7 binary - 1111 0000 - F0-->
<spi_write count="2" radix="16" > F0 00 </spi_write>

<!—Write alternating output pattern to P1-->
<!--70 is write to register 3 binary 0111 0000 - 70 -->
<spi_write count="2" radix="16" > 70 55 </spi_write>
<sleep ms="100"/>
<spi_write count="2" radix="16" > 70 AA </spi_write>
</adapter>
```

7 AxIO16 Bill of Material

AxIO16 EVM	PCB #: 401-200-231-A00						
	PCBa #: 400-200-231-A00						
Designator	Qty	Manufacturer Part #	Manufacturer	Description / Value	Package/Footprint	Type	Instructions / Notes
C1, C2	2	CL10B106MQ8NRNC	Samsung Electro-Mechanics	CAP CER 10UF 6.3V X7R 0603	0603 (1608 Metric)	SMD	
C3, C4	2	CL05B104KA5NNNC	Samsung Electro-Mechanics	CAP CER 0.1UF 25V X7R 0402	0402 (1005 Metric)	SMD	
C5, C6	2	CL05B103KA5NNNC	Samsung Electro-Mechanics	CAP CER 10000PF 25V X7R 0402	0402 (1005 Metric)	SMD	
C7, C8	2	n/a	n/a	n/a	0805 (2012 Metric)	DNP	DO NOT POPULATE
D1, D2, D3, D4, D5, D6, D7, D8, D9	9	XZCBD53W-1	SunLED	LED BLUE CLEAR CHIP SMD	0603 (1608 Metric)	SMD	
J1, J3	2	CT3151SP-2	Cal Test Electronics	CONN BANANA JACK SOLDER RED	Connector	TH	
J2	1	CT3151SP 0	Cal Test Electronics	CONN BANANA JACK SOLDER BLACK	Connector	TH	
J4, J5, J25, J27, J28	5	61300111121	Würth Elektronik	CONN HEADER VERT 1POS	Header 1 Pos 0.1" Pitch	TH	
J6, J7	2	61301411121	Würth Elektronik	CONN HEADER VERT 14POS 2.54MM	Header 14 Pos 0.1" Pitch	TH	
J8, J26	2	61300211121	Würth Elektronik	CONN HEADER VERT 2POS 2.54MM	Header 2 Pos 0.1" Pitch	TH	
J9, J10, J11, J12, J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, J23, J24, J29, J30, J31	19	61300311121	Würth Elektronik	CONN HEADER VERT 3POS 2.54MM	Header 3 Pos 0.1" Pitch	TH	
J32	1	TSW 106 07 G T	Samtec Inc.	CONN HEADER VERT 18POS 2.54MM	Header 18 Pos 0.1" Pitch	TH	
R1, R2, R3, R4, R5, R6, R7, R9, R12	9	RMCF0603FT5K76	Stackpole Electronics Inc	RES 5.76K OHM 1% 1/10W 0603	0603 (1608 Metric)	SMD	
R8	1	RMCF0603FT49K9	Stackpole Electronics Inc	RES 49.9K OHM 1% 1/10W 0603	0603 (1608 Metric)	DNP	DO NOT POPULATE
R10, R11	2	RMCF0603FT4K70	Stackpole Electronics Inc	RES 4.7K OHM 1% 1/10W 0603	0603 (1608 Metric)	SMD	
R13	1	RMCF0603FT49K9	Stackpole Electronics Inc	RES 49.9K OHM 1% 1/10W 0603	0603 (1608 Metric)	SMD	
R14, R15, R16, R17, R18, R19	6	RCL0406000Z0EA	Vishay Dale	RES SMD 0 OHM 1/4W 0604 WIDE	0604 WIDE	SMD	
SW1	1	TL1105AF160Q	E-Switch	SWITCH TACTILE SPST NO 0.05A 12V	Switch	TH	
TP1, TP4, TP5	3	5001	Keystone Electronics	PC TEST POINT MINIATURE BLACK	Test Point	TH	
TP2, TP3	2	5000	Keystone Electronics	PC TEST POINT MINIATURE RED	Test Point	TH	
U1	1	APIO16	Apogee Semiconductor	RAD HARD GPIO EXPANDER	TSSOP 28	SMD	
n/a	25	SNT-100-BK-T	Samtec Inc.	CONN JUMPER SHORTING .100" GOLD	Header Jumper/Shunt	Jumper	

Table 2 AxIO16 Bill of Materials

8 Legal Disclaimers for Use of the Evaluation Board

8.1 General Disclaimer This evaluation board is provided "as-is" for evaluation purposes only. The manufacturer does not warrant the performance, suitability, or fitness of the board for any particular purpose. Use of the evaluation board is at your own risk, and the manufacturer will not be liable for any direct or indirect damages, losses, or costs arising from the use or inability to use the evaluation board.

8.2 Intellectual Property The evaluation board and associated software may be subject to intellectual property rights, including patents, trademarks, and copyrights. You acknowledge that no ownership rights are transferred to you upon receiving or using the evaluation board. Any use of the evaluation board or its components in any product for commercial sale or distribution without appropriate licensing is prohibited.

8.3 Not for Production Use The evaluation board is intended solely for testing, prototyping, and evaluation purposes. It is not designed or intended for production environments or end-use applications. The evaluation board is not suitable for use in systems where failure could lead to significant harm, injury, or damage to property.

8.4 Compliance with Regulations It is the user's responsibility to ensure that the use of the evaluation board complies with applicable local laws, regulations, and safety standards. The manufacturer does not guarantee that the evaluation board complies with any specific regional or international regulatory requirements (e.g., CE, UL, FCC).

8.5 Export Control The evaluation board may be subject to export control laws and regulations in certain jurisdictions. By accepting this evaluation board, you agree to comply with all applicable export control laws and regulations. You may not export, re-export, or transfer the evaluation board to any prohibited country or entity under export regulations.

8.6 Support and Warranty The evaluation board is provided without warranty of any kind, either express or implied, including, but not limited to, warranties of merchantability or fitness for a particular purpose. The manufacturer does not provide support or maintenance for the evaluation board beyond what is explicitly described in the documentation or user guides.

By using the evaluation board, you acknowledge that you have read, understood, and agree to these legal disclaimers.