



# winhec

Workshop Taipei 2016

Any questions please contact [winhectepe@microsoft.com](mailto:winhectepe@microsoft.com)



# Access Hardware Buses in User Mode

Devin Wong  
Software Engineer

Partner Enablement Team

# Contents

Overview of Resource Hub Proxy

End-to-End Example on Raspberry Pi 2 and Minnowboard MAX

Verification of ASL

Demo – Reading ADXL345 data in User Mode

Takeaway - you would leave with knowing

- How to modify asl on your own devices to enable hardware buses access in User Mode

- How to use corresponding APIs to access buses

# Low power buses on Windows

## GPIO

- Used as an interrupt or for general I/O
- Buttons, switches, lights, system wake, etc.
- Interrupts for SPB, serial, and other buses

## I<sup>2</sup>C, SPI

- Simple Peripheral Buses (SPB) = I<sup>2</sup>C, SPI
- I<sup>2</sup>C: 100Kbps-3.4Mbps, SPI: Up to 160Mbps
- Typically used for:
  - Input / HID
  - Radios
  - Sensors
  - Power management

## UART

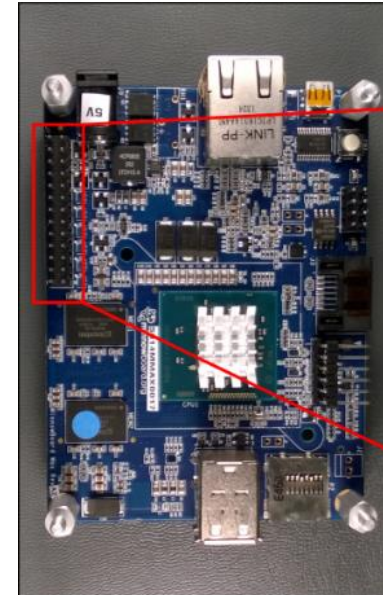
- High Speed UART
- Up to 20Mbps
- Typically used for:
  - Bluetooth,
  - GPS

# Hardware Pins on a dev board



3.3V PWR	1	2	5V PWR
I2C1 SDA	3	4	5V PWR
I2C1 SCL	5	6	GND
GPIO 4	7	8	Reserved
GND	9	10	Reserved
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3.3V PWR	17	18	GPIO 24
SPI0 MOSI	19	20	GND
SPI0 MISO	21	22	GPIO 25
SPI0 SCLK	23	24	SPI0 CS0
GND	25	26	SPI0 CS1
Reserved	27	28	Reserved
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

Raspberry Pi 2



GND	1	2	GND
5V PWR	3	4	3.3V PWR
SPI0 CS0	5	6	UART1 TX
SPI0 MISO	7	8	UART1 RX
SPI0 MOSI	9	10	UART1 CTS
SPI0 SCLK	11	12	UART1 RTS
I2C5 SCL	13	14	GPIO 3
I2C5 SDA	15	16	GPIO 4
UART2 TX	17	18	GPIO 5
UART2 RX	19	20	GPIO 6
GPIO 0	21	22	GPIO 7
GPIO 1	23	24	GPIO 8
GPIO 2	25	26	GPIO 9

MinnowBoard MAX

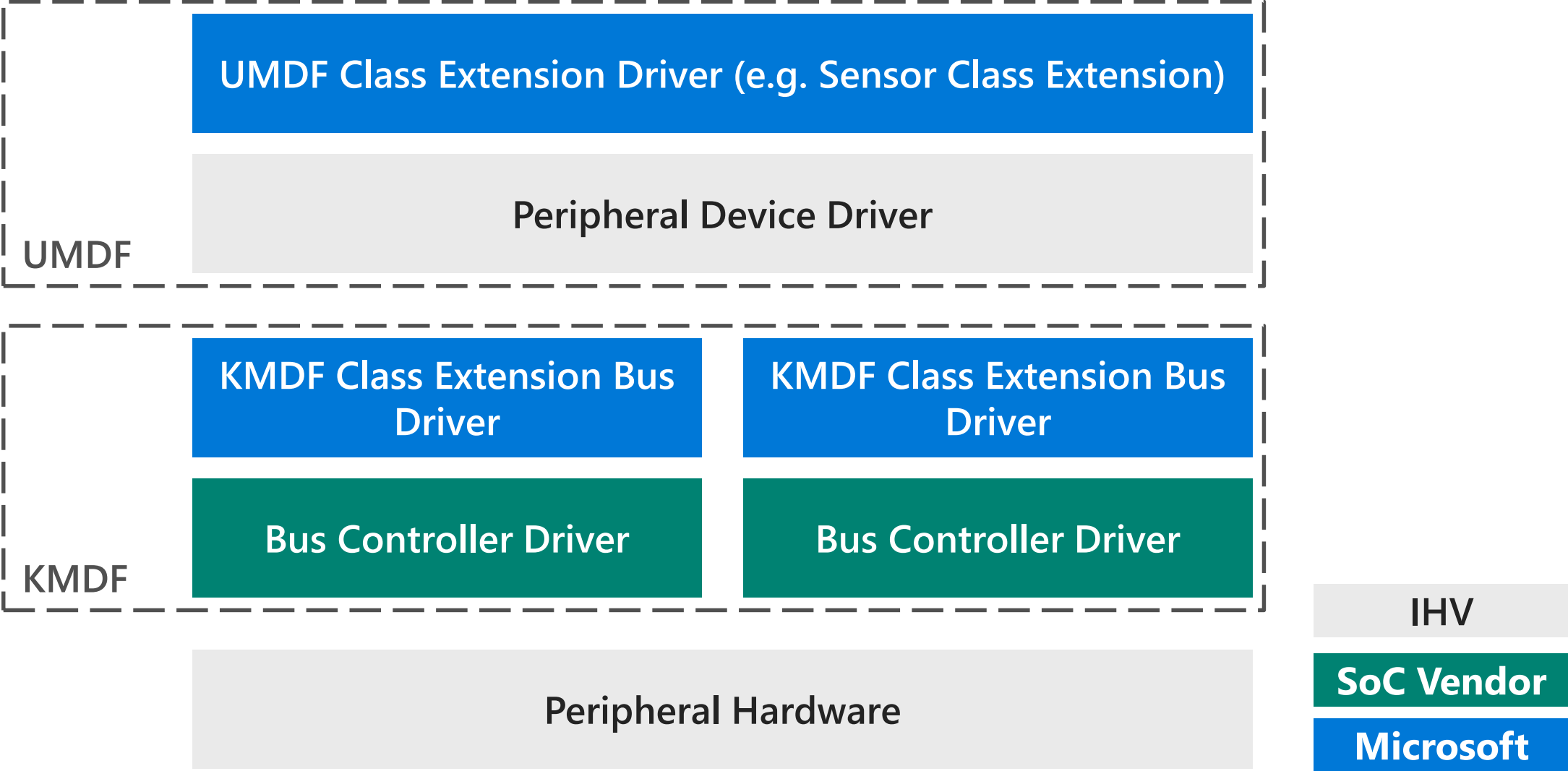


GND	1	2	GND
UART0 CTS	3	4	Reserved
UART0 TX	5	6	Reserved
UART0 RX	7	8	SPI0 CLK
UART0 RTS	9	10	SPI0 MISO
UART1 TX	11	12	SPI0 CS N
UART1 RX	13	14	SPI0 MOSI
I2C0 SCL	15	16	Reserved
I2C0 SDA	17	18	Reserved
I2C1 SCL	19	20	Reserved
I2C1 SDA	21	22	Reserved
GPIO 36	23	24	GPIO 12
GPIO 13	25	26	GPIO 69
GPIO 115	27	28	Reserved
GPIO 24*	29	30	GPIO 25
GPIO 35	31	32	GPIO 34
GPIO 28	33	34	GPIO 33
1.8V PWR	35	36	SYS DC IN
5V PWR	37	38	SYS DC IN
GND	39	40	GND

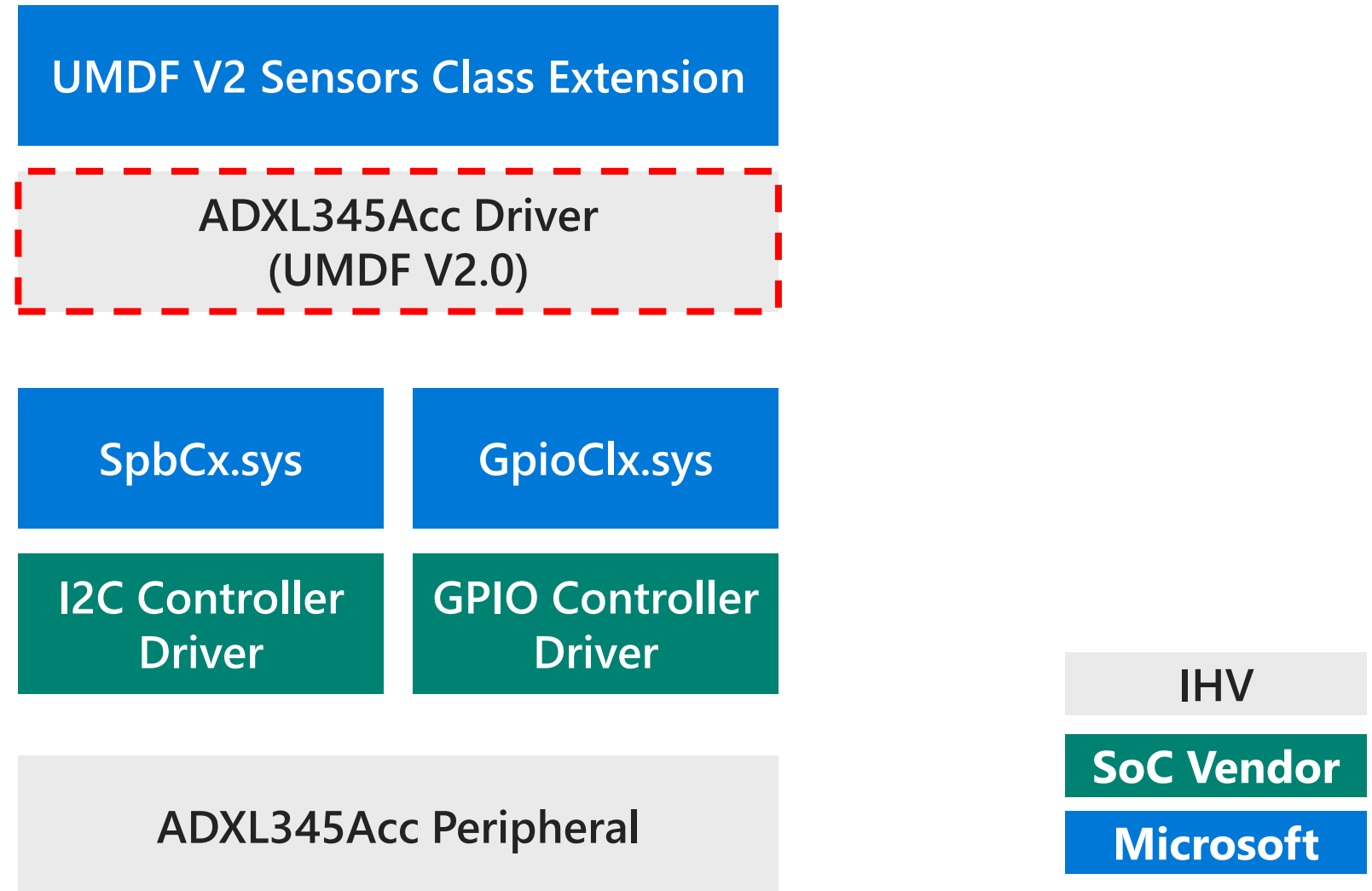
Qualcomm Snapdragon 410

Input Only  
NOTE: GPIO 21 and 120 control onboard LEDs

# Typical device driver architecture in desktop



# Example: Accelerometer sensor in desktop



# It becomes easier in Win 10 IoT

Windows 10 IoT Core contains new APIs for accessing GPIO, I2C, SPI, and UART directly from user mode

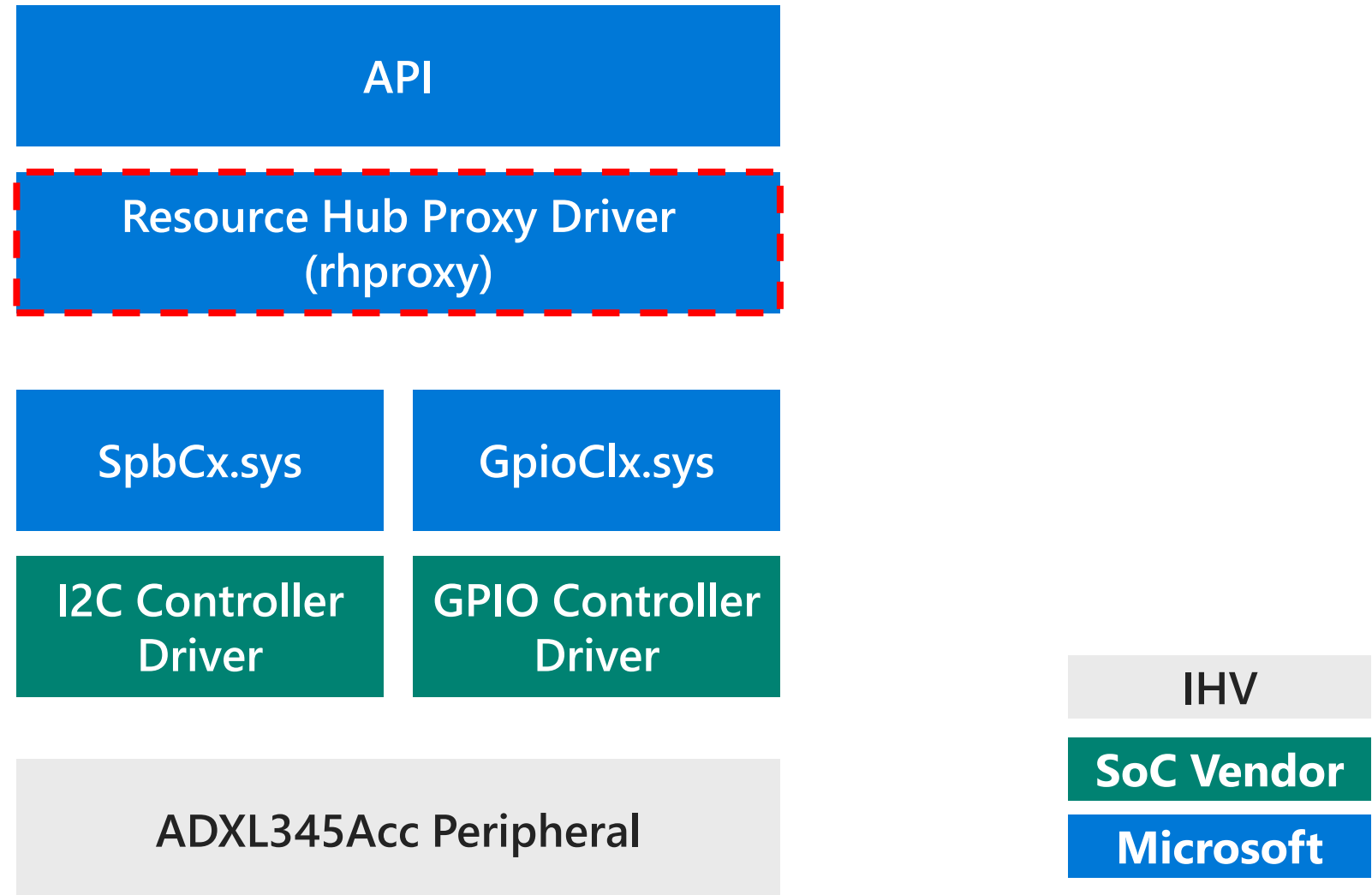
New driver called *rhProxy* exposes *GpioClx* and *SpbCx* resources to user mode

No additional drivers needed

Detailed document "Enabling Usermode Access to GPIO, I2C, SPI, UART" by Jordan Rhee can be obtained from Connect site.



# Example: Accelerometer sensor in Win 10 IoT



# End-to-End Example on Raspberry Pi 2 and Minnowboard MAX

# Downloads and Tools

[http://ms-  
iot.github.io/content/en-  
US/Downloads.htm](http://ms-iot.github.io/content/en-US/Downloads.htm)

Download Windows 10 IoT Core,  
Visual Studio with SDK, WDK or  
any of the other tools available

## Downloads and Tools

Download Windows for IoT, Visual Studio, Software Development Kits or any of the other tools available on this page to get started developing for the Internet of Things today!



### Download Windows 10 IoT Core

Download the November update for Windows 10 IoT Core. Built for devices, Windows 10 IoT Core enables you to create cool connected projects, amazing applications, and much more. It leverages the Windows development environment you know -- adapted to run on embedded devices like the Raspberry Pi 2.

The November update of Windows 10 IoT Core requires Visual Studio 2015 Update 1 for development purposes such as debugging and deploying images.

Download Windows 10 IoT Core for Raspberry Pi 2

Download Windows 10 IoT Core for MinnowBoard Max

Download Windows 10 IoT Core for DragonBoard 410c\*

\*Please see [third party notice for BSP restrictions](#) for DragonBoard 410c.

### Windows 10 IoT Core Dashboard

The Windows 10 IoT Core Dashboard is your best tool for configuring new Windows 10 IoT Core devices, running demo applications, and much more. The Windows 10 IoT Core Dashboard supports the MinnowBoard MAX and the Raspberry Pi 2.

Click below to download the Dashboard today and get started with your Windows 10 IoT Core devices.

Get IoT Core Dashboard

By downloading and using the Windows 10 IoT Core Dashboard you agree to the [license terms](#) and [privacy statement](#) for Windows 10 IoT Core Dashboard.

Need help? [Check out the troubleshooting page.](#)

# ACPI Device Declaration

## Rhproxy device node declaration on Raspberry Pi 2

```
Scope (\_SB)
{
    Device(RHPX)
    {
        Name(_HID, "MSFT8000")
        Name(_CID, "MSFT8000")
        Name(_UID, 1)
    }
}
```

...

**\_HID** – Hardware Id. Set this to a vendor-specific hardware ID.

**\_CID** – Compatible Id. Must be "MSFT8000".

**\_UID** – Unique Id. Set to 1.

# Declaring SPI

Raspberry Pi has one exposed SPI buses

SPI0 has two hardware chip select lines One SPISerialBus() resource declaration is required for each chip select line for each bus

The mapping between bus friendly name and resource index is specified in the DSD(Device Specific Data)

# SPI0 – with 2 chip select lines

```
// Index 0
SPISerialBus(
    // SCKL - GPIO 11 - Pin 23
    // MOSI - GPIO 10 - Pin 19
    // MISO - GPIO 9 - Pin 21
    // CE0 - GPIO 8 - Pin 24
    0, // Device selection (CE0)
    // Device selection polarity
    // wiremode
    // databit len: placeholder
    // slave mode
    // conn. speed: placeholder
    // clock polarity: placeholder
    // clock phase: placeholder
    "\\_SB.SPI0", // RscSource: SPI bus ctrl name
    0, // ResourceSourceIndex
    // Resource usage
    // Vendor Data
)
```

```
// Index 1
SPISerialBus(
    // SCKL - GPIO 11 - Pin 23
    // MOSI - GPIO 10 - Pin 19
    // MISO - GPIO 9 - Pin 21
    // CE1 - GPIO 7 - Pin 26
    1, // Device selection (CE1)
    // Device selection polarity
    // wiremode
    // databit len: placeholder
    // slave mode
    // conn. speed: placeholder
    // clock polarity: placeholder
    // clock phase: placeholder
    "\\_SB.SPI0", // RscSource: SPI bus ctrl name
    0, // ResourceSourceIndex
    // Resource usage
    // Vendor Data
)
```

# SPI0 Corresponding DSD settings

```
Package(2) { "bus-SPI-SPI0", Package() { 0, 1 }},
```

//creates a bus named "SPI0" with two chip select lines – resource indexes 0 and 1

```
Package(2) { "SPI0-MinClockInHz", 7629 },
```

```
Package(2) { "SPI0-MaxClockInHz", 125000000 },
```

//the *MinClockInHz* and *MaxClockInHz* properties specify the minimum and maximum clock speeds that are supported by the controller

```
Package(2) { "SPI0-SupportedDataBitLengths", Package() { 8 }},
```

//the *SupportedDataBitLengths* property lists the data bit lengths supported by the controller





# Declaring GPIO

- Next, we declare all the GPIO pins



- Declare all pins on exposed headers.
- Declare pins that are connected to useful onboard functions like buttons and LEDs.



- Do not** declare pins that are reserved for system functions
- Do not** declare pins that are not connected to anything.

# Declaring GPIO

+ Corresponding  
DSD

```
// Index 4 - GPIO 4
GpioIO(Shared, PullUp, , , , "\\_SB.GPIO0", , , , ) { 4 }
GpioInt(Edge, ActiveBoth, Shared, PullUp, 0, "\\_SB.GPIO0",) { 4 }
// Index 6 - GPIO 5
GpioIO(Shared, PullUp, , , , "\\_SB.GPIO0", , , , ) { 5 }
GpioInt(Edge, ActiveBoth, Shared, PullUp, 0, "\\_SB.GPIO0",) { 5 }
```

# Declaring UART

+ Corresponding  
DSD

- The following UART declaration is from MinnowBoard Max as an example

```
// Index 2
UARTSerialBus(
    115200,
    ,
    ,
    0xfc,
    ,
    ,
    ,
    32,
    32,
    "\\_SB.URT2",
    ,
    ,
    ,
)
```

*(Note: The code above is a simplified representation of the actual DSD declaration, which would include comments for each parameter.)*

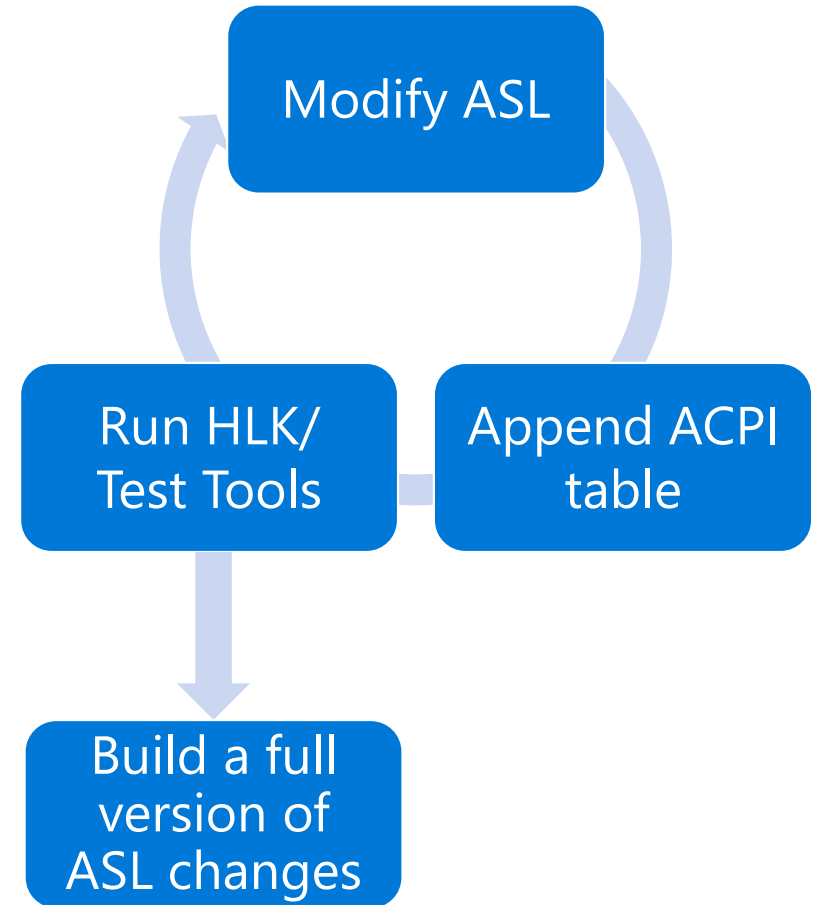
# Verification of ASL

# Verification

May need to modify the ASL file a few times at development stage

We recommend using ACPITABL.dat during development and validation as it does not require a full UEFI rebuild to test ASL changes.

Run the Hardware Lab Kit (HLK) tests to verify that all resources are exposed correctly and the underlying buses meet the functional contract of the API



# 1. Modify your asl file

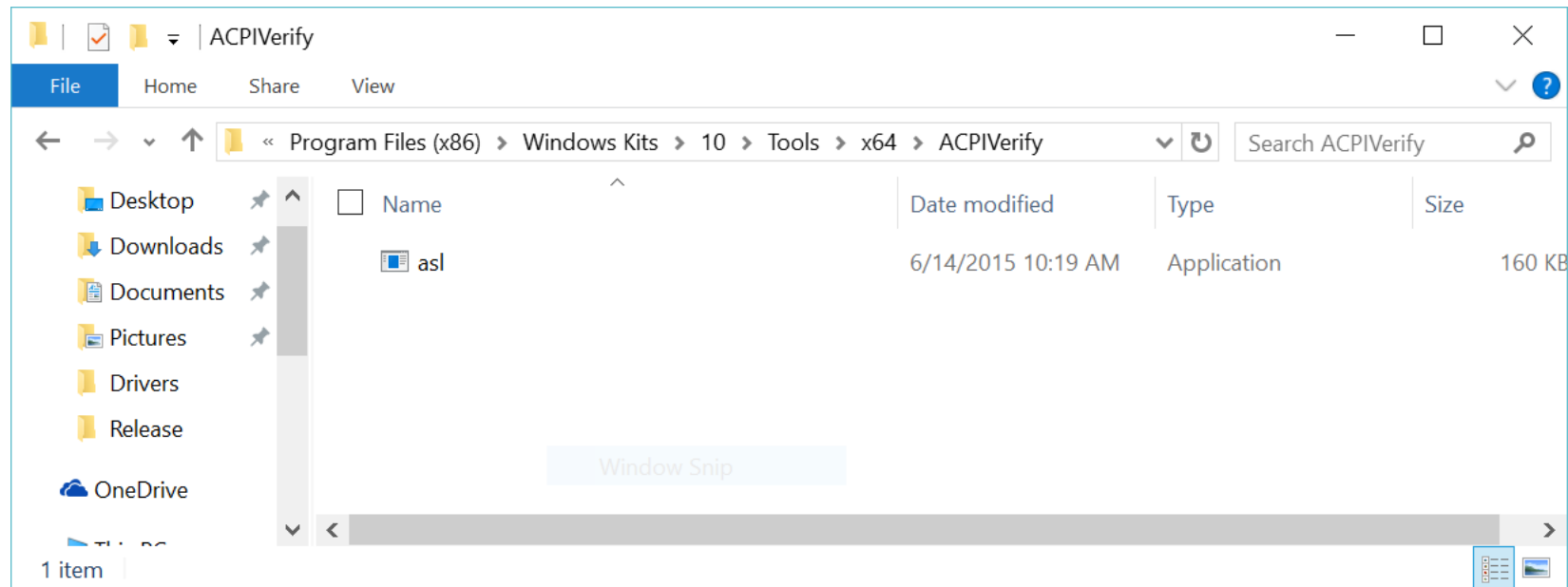
Create a file named `yourboard.asl` and put the RHPX device node inside a `DefinitionBlock`

```
DefinitionBlock ("ACPI.TABL.dat", "SSDT", 1,  
"MSFT", "RHPROXY", 1)  
{  
    Scope (\_SB)  
    {  
        Device(RHPX)  
        {  
            ...  
        }  
    }  
}
```

## 2. Get asl.exe

Download the WDK and get asl.exe

Asl.exe locates at "C:\Program Files (x86)\Windows Kits\10\Tools\x86\ACPIVerify" by default



### 3. Generate ACPITABL.dat

Copy your `yourboard.asl` to the folder of `asl.exe`

Open up a command prompt and navigate to the `asl` compiler directory

Run following command

```
asl.exe yourboard.asl
```



## 4. Copy the resulting ACPITABL.dat file

An ACPITABL.dat file will be generated in the same directory as the ASL file

Copy the resulting ACPITABL.dat file to c:\windows\system32 on your system under test.

# 5. Turn on testsigning

## Turn on testsigning on your system under test

bcdedit /set testsigning on  
Use bcdedit to check status

```
C:\Windows\system32>bcdedit /set testsigning on  
The operation completed successfully.
```

```
C:\Windows\system32>bcdedit
```

### Windows Boot Manager

```
-----  
identifier          <bootmgr>  
device              partition=\Device\HarddiskVolume2  
path                \EFI\Microsoft\Boot\bootmgfw.efi  
description         Windows Boot Manager  
locale              en-US  
inherit             <globalsettings>  
default             <current>  
resumeobject       <42b450c2-3055-11e5-b17d-f46e80558b4b>  
displayorder       <current>  
toolsdisplayorder  <memdiag>  
timeout             30
```

### Windows Boot Loader

```
-----  
identifier          <current>  
device              partition=C:  
path                \Windows\system32\winload.efi  
description         Windows 10  
locale              en-US  
inherit             <bootloadersettings>  
recoverysequence   <42b450c4-3055-11e5-b17d-f46e80558b4b>  
recoveryenabled    Yes  
testsigning         Yes  
isolatedcontext     Yes
```

## 6. Reboot the system under test

The system will append the ACPI tables defined in `ACPITABL.dat` to the system firmware tables.

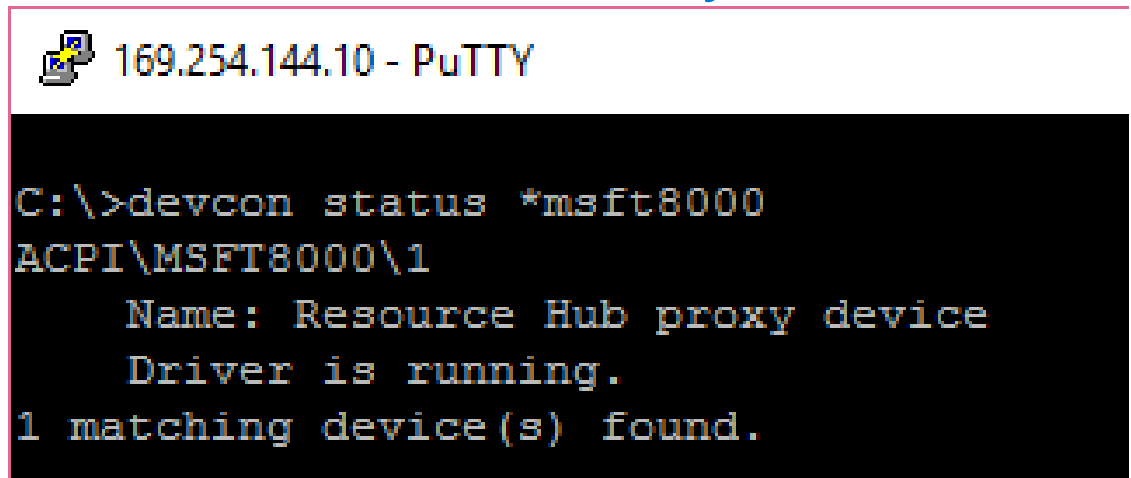
# 7. Verify device node was added

## Sanity check

GpioTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/GpioTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/GpioTestTool.htm</a>
I2cTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/I2cTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/I2cTestTool.htm</a>
SpiTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/SpiTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/SpiTestTool.htm</a>
MinComm (Serial)	<a href="https://github.com/ms-iot/samples/tree/develop/MinComm">https://github.com/ms-iot/samples/tree/develop/MinComm</a>

## Verify that the RHPX device node was added to the system

*devcon status \*msft8000*



```
169.254.144.10 - PuTTY  
C:\>devcon status *msft8000  
ACPI\MSFT8000\1  
    Name: Resource Hub proxy device  
    Driver is running.  
1 matching device(s) found.
```

# Run the HLK Tests

In the HLK manager, select "Resource Hub Proxy device"

Windows Hardware Lab Kit Help | Configuration | Connect

mbm

Project Selection Tests Results Package

\$\mypool All Search

Name	Platform	Machine	Group
<input type="checkbox"/> PCI-to-PCI Bridge	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Plug and Play Software Device Enumerator	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Power Button	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Power Button	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Programmable interrupt controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Realtek PCIe GBE Family Controller	Windows Mobile v10	jordanrh-mbm	
<input checked="" type="checkbox"/> Resource Hub proxy device	Windows Mobile v10	jordanrh-mbm	[Group 2]
<input type="checkbox"/> SD Storage Class Controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> SDA Standard Compliant SD Host Controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> SM Bus Controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> SPI Controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> Standard Power Management Controller	Windows Mobile v10	jordanrh-mbm	
<input type="checkbox"/> System CMOS/real time clock	Windows Mobile v10	jordanrh-mbm	

show selected systems devices and printers device manager software device

show  Inbox  Hidden

Mfg: **Microsoft**  
Device Name: **Resource Hub proxy device**  
Driver Name: **c:\windows\system32\drivers\rhproxy.sys**  
Device Instance ID: **ACPI\MSFT8000\1**

<input type="checkbox"/>	Realtek PCIe GBE Family Controller	Windows Mobile v10	jordanrh-mbm
<input checked="" type="checkbox"/>	Resource Hub proxy device	Windows Mobile v10	jordanrh-mbm
<input type="checkbox"/>	SD Storage Class Controller	Windows Mobile v10	jordanrh-mbm

# Select Desired Tests in HLK

Then click the Tests tab, and select I2C WinRT, GPIO WinRT, and SPI WinRT tests and run

The screenshot displays the Windows Hardware Lab Kit (HLK) interface. The main window is titled "Windows Hardware Lab Kit" and shows a list of tests under the "Tests" tab. The "Tests" tab is selected, and the "GPIO WinRT Functional and Stress Tests" is highlighted. The interface also shows a "Test Status" section on the right, which is currently empty. The "Machine Status" section at the bottom right shows a warning icon and the text "jordanrh-mbm".

Status	Test Name	Type	Le	Ti	M
<input type="checkbox"/>	DF - PNP Surprise Remove Device Test (Development and Integration)	DF	03	Re	joi
<input type="checkbox"/>	DF - PNP Surprise Remove Device Test (Reliability)	DF	08	Re	joi
<input type="checkbox"/>	DF - Reboot restart with IO before and after (Reliability)	DF	04	Re	joi
<input type="checkbox"/>	DF - Reboot Restart with IO During (Development and Integration)	DF	03	Re	joi
<input type="checkbox"/>	DF - Reboot Restart with IO During (Reliability)	DF	05	Re	joi
<input type="checkbox"/>	DF - Sleep and PNP (disable and enable) with IO Before and After (Bring Up)	DF	05	Re	joi
<input type="checkbox"/>	DF - Sleep and PNP (disable and enable) with IO Before and After (Reliability)	DF	45	Re	joi
<input type="checkbox"/>	DF - Sleep with IO Before and After (Bring Up)	DF	04	Re	joi
<input type="checkbox"/>	DF - Sleep with IO During (Development and Integration)	DF	05	Re	joi
<input type="checkbox"/>	DF - Sleep with IO During (Reliability)	DF	45	Re	joi
<input checked="" type="checkbox"/>	<b>GPIO WinRT Functional and Stress Tests</b>	GPIO	05	Re	joi
<input checked="" type="checkbox"/>	I2C WinRT Advanced Functional Tests (mbed LPC1768)	I2C	15	Re	joi
<input checked="" type="checkbox"/>	I2C WinRT IO Stress Tests (EEPROMs Required)	I2C	04	Re	joi
<input checked="" type="checkbox"/>	I2C WinRT Nonexistent Slave Address Tests	I2C	05	Re	joi
<input checked="" type="checkbox"/>	I2C WinRT Read Tests (EEPROM Required)	I2C	05	Re	joi
<input checked="" type="checkbox"/>	I2C WinRT Write Tests (EEPROM Required)	I2C	05	Re	joi
<input type="checkbox"/>	WDK Guardrail Analysis	WDK	02	Re	joi

Test Name	Type	Le	Ti	M
DF - Sleep with IO During (Reliability)	DF	45	Re	joi
<b>GPIO WinRT Functional and Stress Tests</b>	GPIO	05	Re	joi
I2C WinRT Advanced Functional Tests (mbed LPC1768)	I2C	15	Re	joi
I2C WinRT IO Stress Tests (EEPROMs Required)	I2C	04	Re	joi
I2C WinRT Nonexistent Slave Address Tests	I2C	05	Re	joi
I2C WinRT Read Tests (EEPROM Required)	I2C	05	Re	joi
I2C WinRT Write Tests (EEPROM Required)	I2C	05	Re	joi

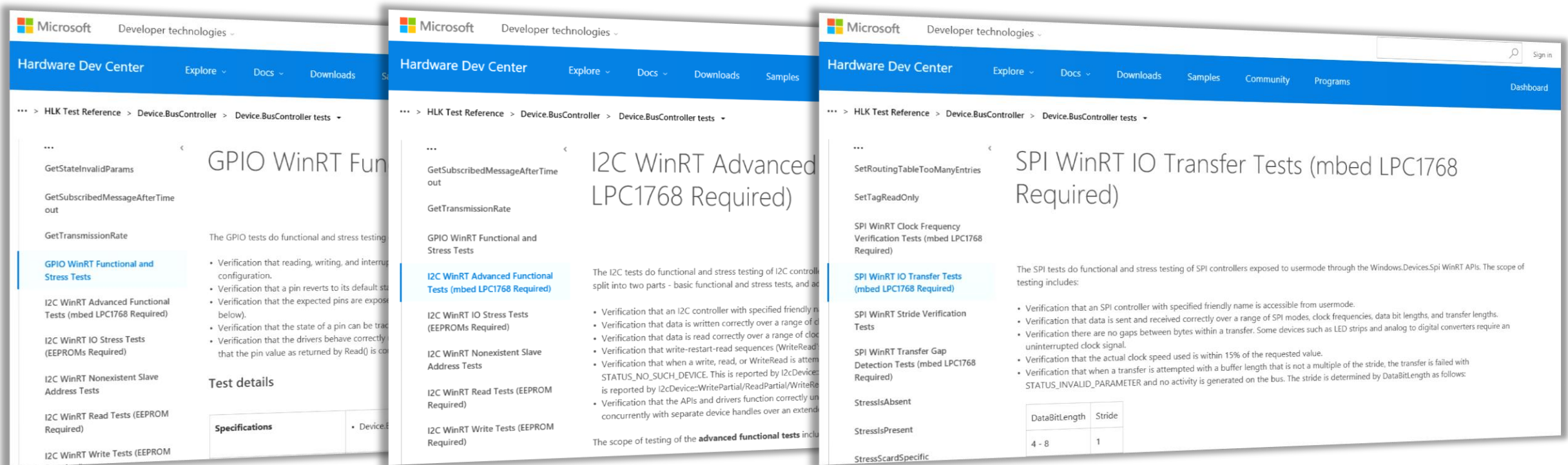
Test Status:

- Passed: 0 test
- Failed: 0 test
- Assessment Passed: 0 test
- Assessment Failed: 0 test
- Not Run: 49 test
- Running: 0 test(s)
- Total: 49 test(s)

Machine Status: jordanrh-mbm

# Full Instructions to run HLK on MSDN

GPIO	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/mt591939(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/mt591939(v=vs.85).aspx</a>
I2C	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/mt591936(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/mt591936(v=vs.85).aspx</a>
SPI	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/mt591929(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/mt591929(v=vs.85).aspx</a>



# Case study - DSD must be a child of Device()

## Erroneous

```
Device(MTST)
{
    Name(_HID, "MSFT8000")
    Name(_CID, "MSFT8000")
    Name(_UID, One)
    Method(_CRS, 0x0, Serialized)
    {
        Name(UBUF, ResourceTemplate()
        {
            SPISerialBus(0, PolarityLow, FourWireMode, 0x8,
ControllerInitiated, 0x7a1200, ClockPolarityLow,
ClockPhaseSecond, "\\_SB.SPI1", 0, ResourceConsumer, , )
            I2CSerialBus(0xff, ControllerInitiated, 0x61a80,
AddressingMode7Bit, "\\_SB.I2C1", 0, ResourceConsumer, , )
        })
        Name(_DSD, Package(0x2)
        //erroneous!! cannot be a child of _CRS. Must be child of
Device()
        {
```

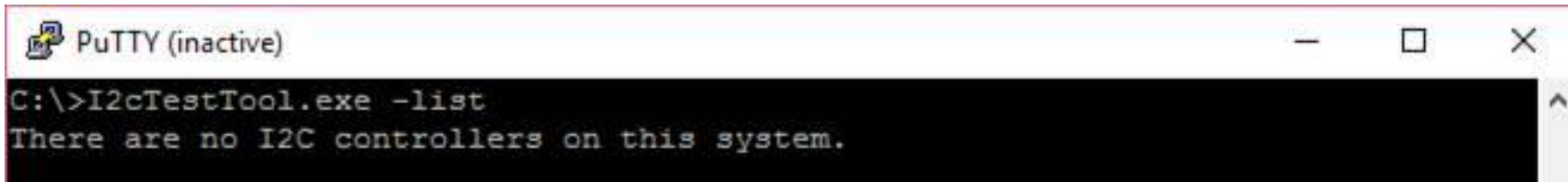
## Correct

```
Device(MTST)
{
    Name(_HID, "MSFT8000")
    Name(_CID, "MSFT8000")
    Name(_UID, One)
    Method(_CRS, 0x0, Serialized)
    {
        Name(UBUF, ResourceTemplate()
        {
            SPISerialBus(0, PolarityLow, FourWireMode, 0x8,
ControllerInitiated, 0x7a1200, ClockPolarityLow,
ClockPhaseSecond, "\\_SB.SPI1", 0, ResourceConsumer, , )
            I2CSerialBus(0xff, ControllerInitiated, 0x61a80,
AddressingMode7Bit, "\\_SB.I2C1", 0, ResourceConsumer, , )
            ...
        })
        ...
    }
    Name(_DSD, Package(0x2) //Correct!!
    {
```



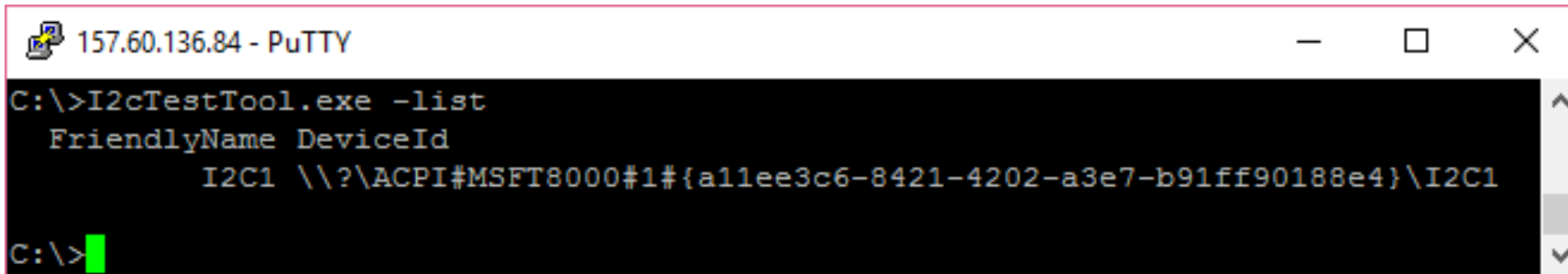
# Use I2cTestTool to verify

If I2C is declared incorrectly



```
PuTTY (inactive)
C:\>I2cTestTool.exe -list
There are no I2C controllers on this system.
```

If declaration is correct



```
157.60.136.84 - PuTTY
C:\>I2cTestTool.exe -list
  FriendlyName DeviceId
      I2C1 \\?\ACPI#MSFT8000#1#{a11ee3c6-8421-4202-a3e7-b91ff90188e4}\I2C1
C:\>
```

# Demo – Reading ADXL345 data in User Mode

**NEXCOM**

Microsoft  
Azure

Certified

## NISE50

The First Microsoft Azure  
IoT Certified Industrial Partner



# Demo – Reading ADXL345 data in User Mode

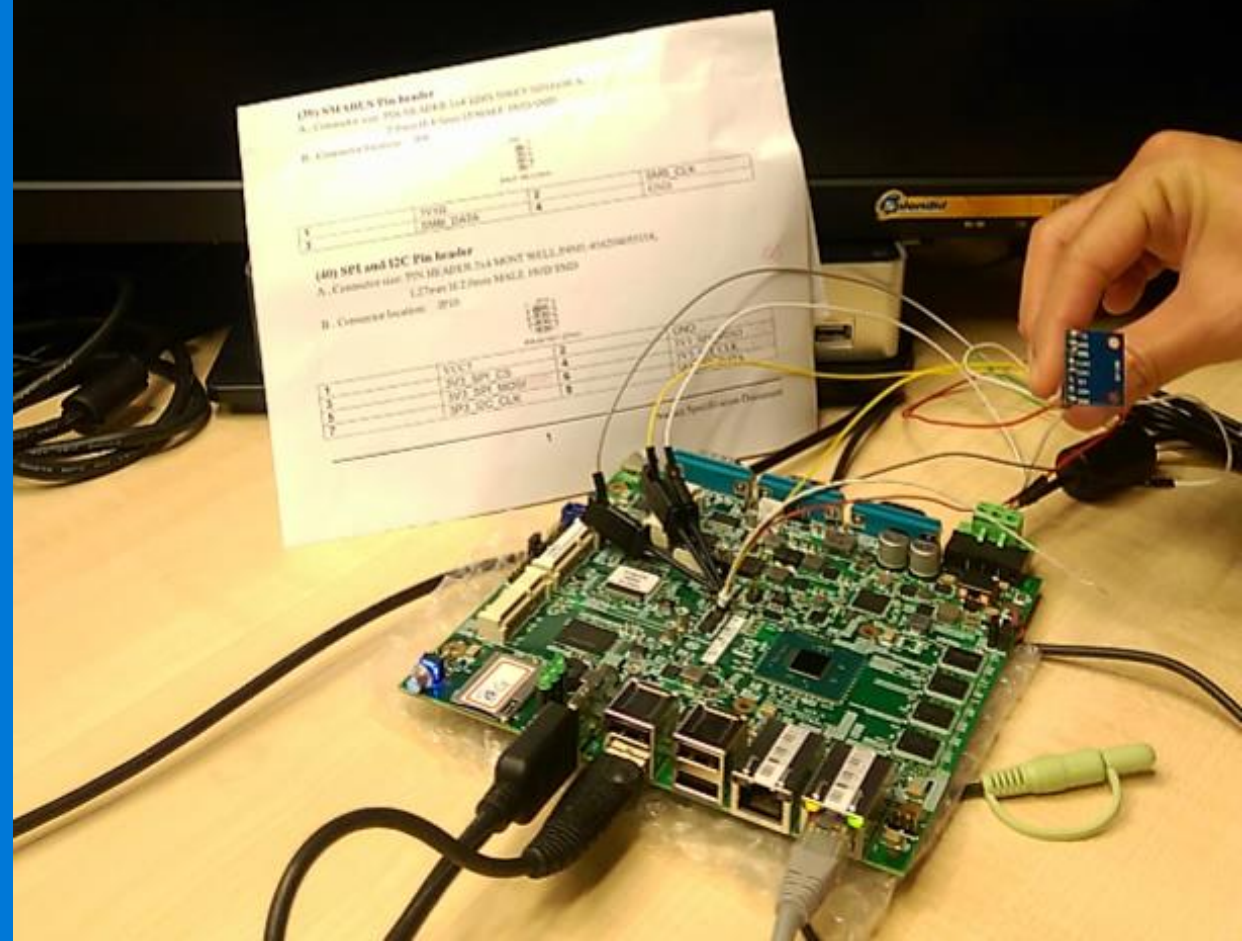
## ADXL345 Accelerometer Data

X Axis: 1.266G

Y Axis: 0.422G

Z Axis: 3.992G

Status: Running



# Resources and References

ACPI 5.0 specification	<a href="http://acpi.info/spec.htm">http://acpi.info/spec.htm</a>
Asl.exe (Microsoft ASL Compiler)	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/dn551195(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/dn551195(v=vs.85).aspx</a>
Windows.Devices.Gpio	<a href="https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.gpio.aspx">https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.gpio.aspx</a>
Windows.Devices.I2c	<a href="https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.i2c.aspx">https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.i2c.aspx</a>
Windows.Devices.Spi	<a href="https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.spi.aspx">https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.spi.aspx</a>
Windows.Devices.SerialCommunication	<a href="https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.serialcommunication.aspx">https://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.serialcommunication.aspx</a>
Test Authoring and Execution Framework (TAEF)	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/hh439725%28v=vs.85%29.aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/hh439725%28v=vs.85%29.aspx</a>
SpbCx	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/hh450906(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/hh450906(v=vs.85).aspx</a>
GpioClx	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/hh439508(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/hh439508(v=vs.85).aspx</a>
SerCx	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/ff546939(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/ff546939(v=vs.85).aspx</a>
MITT I2C Tests	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/dn919852(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/dn919852(v=vs.85).aspx</a>
GpioTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/GpioTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/GpioTestTool.htm</a>
I2cTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/I2cTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/I2cTestTool.htm</a>
SpiTestTool	<a href="http://ms-iot.github.io/content/en-US/win10/samples/SpiTestTool.htm">http://ms-iot.github.io/content/en-US/win10/samples/SpiTestTool.htm</a>
MinComm (Serial)	<a href="https://github.com/ms-iot/samples/tree/develop/MinComm">https://github.com/ms-iot/samples/tree/develop/MinComm</a>
Hardware Lab Kit (HLK)	<a href="https://msdn.microsoft.com/en-us/library/windows/hardware/dn930814(v=vs.85).aspx">https://msdn.microsoft.com/en-us/library/windows/hardware/dn930814(v=vs.85).aspx</a>

# Calls to Action

Join WinHEC LINE Community  
[@winhec](#)



We want to hear from you!

Please Complete the Evaluation Form and return it to our reception.

Your input is highly important to us!  
Thank you!! 😊



