

Intel[®] Atom[™] Processor E3800 Windows* 8 IO Drivers

Programming Guide

February 2014

Revision 1.0
Software Release version: 2.0.0 Beta

Intel Confidential



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Intel product may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2014, Intel Corporation. All rights reserved.

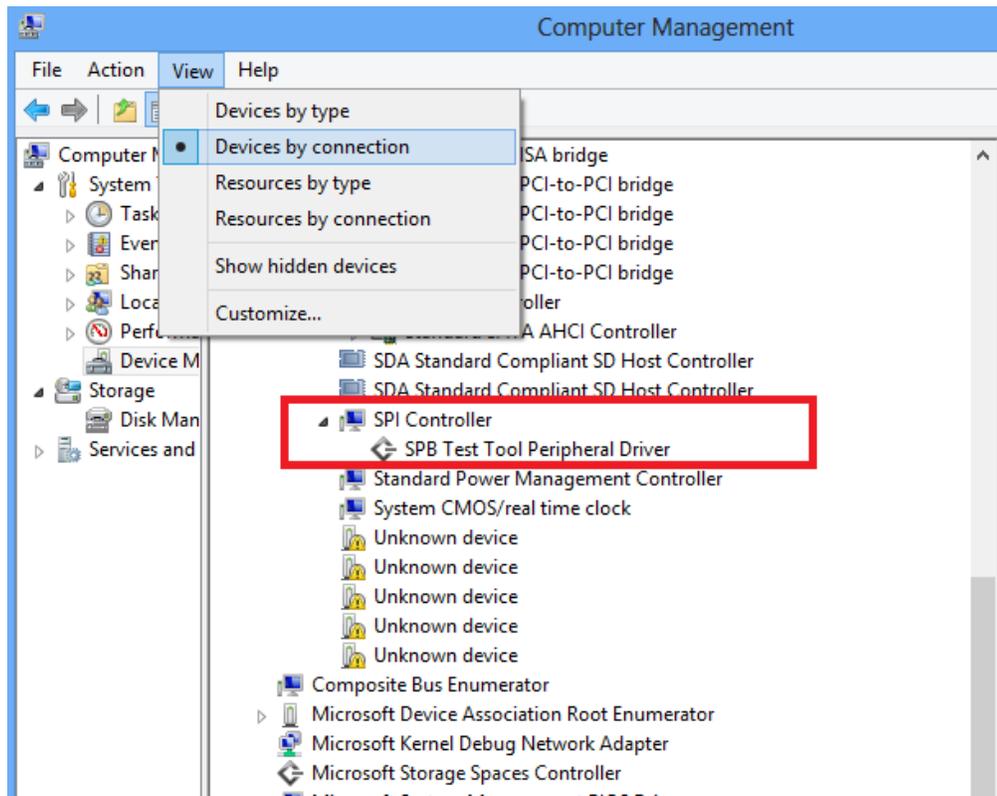


Contents

1	Introduction.....	4
2	GPIO Driver.....	5
3	I ² C and SPI Driver.....	6
4	UART Driver	7

1 Introduction

Microsoft Windows* 8 provided new frameworks for GPIO, I²C, SPI and UART driver. So user-mode applications cannot directly open the controller devices using traditional methods because the GPIO/I²C/SPI/UART controllers do not expose any symbolic links or GUID. So the only way is to mount one sub-device under the controller which is able to open the parent target device and use this sub-device to receive requests from user-mode applications. Overall, all these controllers can be used with a similar method. Below is a structure example of the relationship between the controller (parent) and test device (sub-device).





2 GPIO Driver

The GPIO driver in Windows 8 uses the Microsoft framework called GPIOCx. To use the GPIO controller, there must be a sub-device mounted under the specific GPIO controller. A user mode application can open this sub-device by using its symbolic name or GUID, and send IOCTLs or requests to it. Then only this sub-device can open the parent target (GPIO controller) and forward IOCTLs or requests to the GPIOCx framework, thus to GPIO controller driver.

Refer to the following for Microsoft framework:

<http://msdn.microsoft.com/en-us/library/windows/hardware/hh439508%28v=vs.85%29.aspx>

Here's a **sample code** from Microsoft showing how to write the sub-device driver mounted under GPIO controller, to open its parent and forward requests:

<http://code.msdn.microsoft.com/windowshardware/GPIO-Samples-d25ca63b>

And the description of supported IOCTLs:

<http://msdn.microsoft.com/en-us/library/windows/hardware/hh439470%28v=vs.85%29.aspx>



3 I²C and SPI Driver

I²C and SPI drivers in Windows 8 used the Microsoft framework called the Simple Peripheral Bus (SPBCLX). To use I²C/SPI controller, there must be a sub-device mounted under the specific I²C/SPI controller. A user mode application can open this sub-device by using its symbolic name or GUID, and send IOCTLs or requests to it. Then only this sub-device can open parent target (I²C/SPI controller) and forward IOCTLs or requests to SPBCLX framework, thus to I²C/SPI controller driver.

Refer to the following for Microsoft framework:

<http://msdn.microsoft.com/en-us/library/windows/hardware/hh450906%28v=vs.85%29.aspx>

Here's a **sample code** from Microsoft showing how to write the sub-device driver mounted under the I²C/SPI controller to open its parent and forward requests:

<http://code.msdn.microsoft.com/windowshardware/SpbTestTool-adda6d71>

And the description of supported IOCTLs is here:

<http://msdn.microsoft.com/en-us/library/windows/hardware/hh450915%28v=vs.85%29.aspx>



4 *UART Driver*

The UART driver in Win8 uses the framework of Microsoft called Serial Framework Extension (SerCx). To use UART controller, there must be a sub-device mounted under the specific UART controller. A user mode application can open this sub-device by using its symbolic name or GUID, and send IOCTLs or requests to it. Then only this sub-device can open parent target (UART controller) and forward IOCTLs or requests to SerCx framework, thus to UART controller driver.

Refer to the following for Microsoft framework:

<http://msdn.microsoft.com/en-us/library/windows/hardware/dn265348%28v=vs.85%29.aspx>

And the description of supported IOCTLs is here:

[http://msdn.microsoft.com/en-us/library/windows/hardware/ff547466\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff547466(v=vs.85).aspx)

There are no sample codes of the sub-device provided by Microsoft, but it's quite similar to the SPB or GPIO interface.