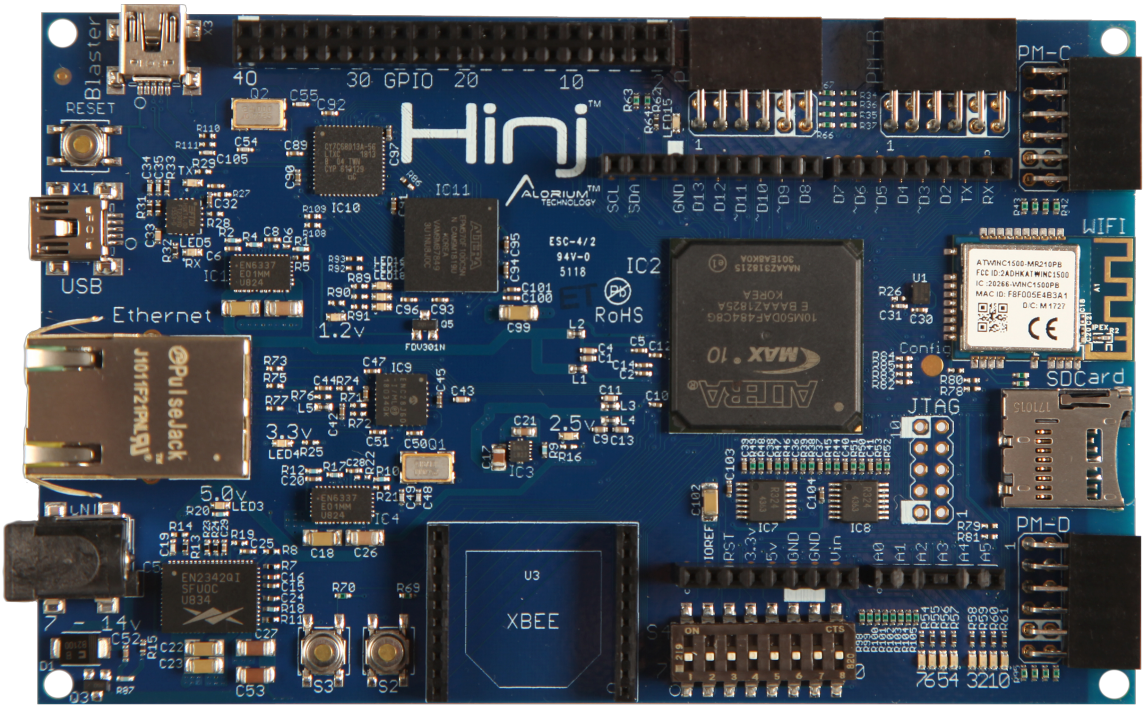
ALORIUM TECHNOLOGY

Hinj



IoT Sensor Hub and Development Board

User Manual

Version 1.2

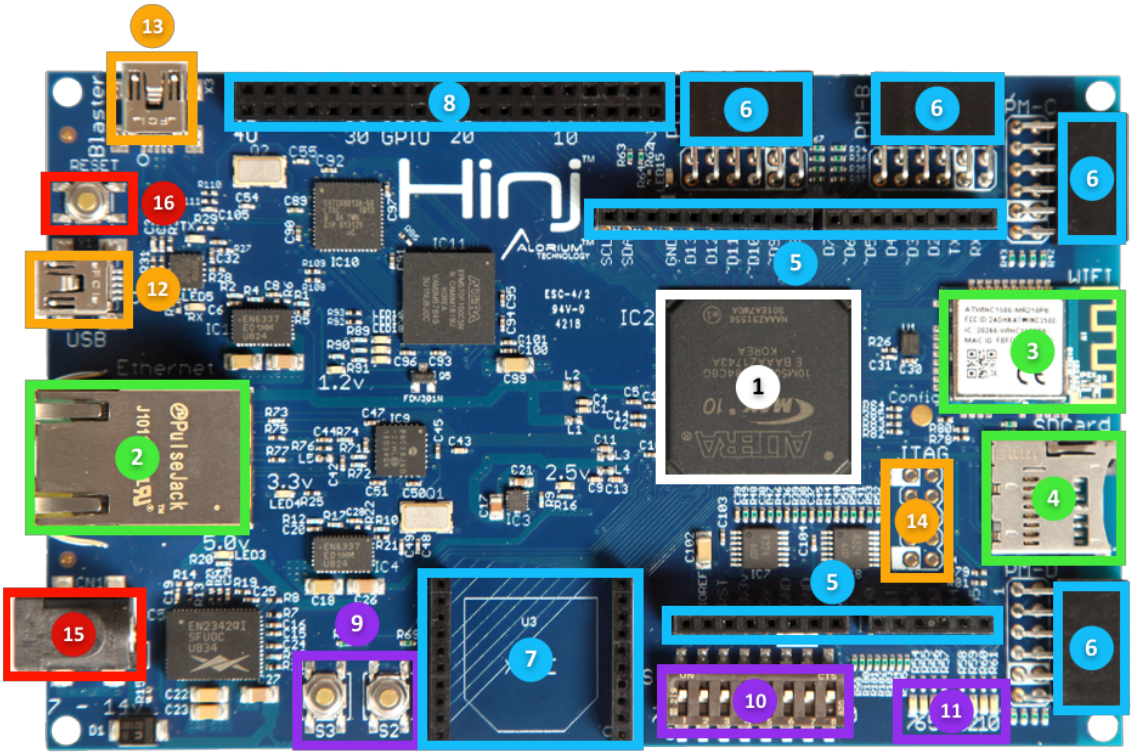
Table of Contents

1 Introduction **4**
1.1 Key Features **4**
2 Usage and Applications **5**
2.1 Example Development Setup: ZigBee Sensor Hub **5**
3 Functional Description **6**
3.1 Intel MAX 10 FPGA **6**
3.2 Integrated Wi-Fi and Ethernet **6**
3.2.1 Wi-Fi 6
3.2.2 Ethernet 6
3.3 Standard Interfaces **7**
3.3.1 Arduino R3 7
3.3.2 Pmod™ 7
3.3.3 XBee® 7
3.3.4 GPIO 7
3.4 User I/O **7**
3.5 microSD Memory Slot **8**
3.6 Programming **8**
3.6.1 Microcontroller and FPGA Programming with USB UART 8
3.6.2 Bare-Metal FPGA Programming with integrated USB Blaster or JTAG 8
JTAG 9
3.7 Power and Reset **9**
4 Technical Specifications **10**
5 Restoring Factory FPGA Image **11**
6 Hinj Library Reference **12**
6.1.1 Base Libraries 12
6.1.2 Library Definitions 12

7 Register Map 16
7.1 XLR8 and XB Register Descriptions 21
7.1.1 XLR8VERL, XLR8VERH, XLR8VERT – Version Number Registers 21
7.1.2 FCFGCID – Chip ID Register 21
7.1.3 CLKSPD – Clock Speed Register 21
7.1.4 XICR, XIFR, XMSK, XACK - XLR8 IRQ Registers 21
7.1.5 OX8ICR, OX8IFR, OXAMSK - OpenXLR8 Pin Change Interrupts 22
7.1.6 FCFGDAT, FCFGSTS, FCFGCTL – FPGA Reconfiguration Registers 23
7.1.7 SVPWH, SVPWL, SVCR – Servo XB Registers 23
7.1.8 NEOD2, NEOD1, NEOD0, NEOCR – NeoPixel XB Registers 24
7.1.9 XFCTRL, XFSTAT, XFR0, XFR1, XFR2, XFR3– Floating Point XB Registers 25
7.1.10 XLR8ADCR – XLR8 ADC Control Register 25
7.1.11 XLR8PID – XLR8 PID 25
7.1.12 XLR8Quad – XLR8 Quadrature 26
7.1.13 Additional Hinj Ports 27
7.1.14 WIFI_SPCR, WIFI_SPSR, WIFI_SPDR - WIFI SPI Interface 31
7.1.15 ETH_SPCR, ETH_SPSR, ETH_SPDR - ETH SPI Interface 31
7.1.16 SD_SPCR, SD_SPSR, SD_SPDR - SD SPI Interface 32
7.1.17 HPCICR, HPCIFR, HPCIMSK -Hinj Pin Change Interrupts 32
7.1.18 MSKX1, MSKX0, MSKG4, MSKG3, MSKG2, MSKG1, MSKG0, MSKPD, MSKPC, MSKPB, MSKPA, MSKPI - Hinj GPIO Port Mask Registers 32
8 Additional Documentation 33
Appendix A – Pin Maps 34
Appendix C - Credits 36

1 Introduction

Hinj is an Intel MAX 10 FPGA development platform that provides a high level of interface flexibility for third-party IoT carrier boards, modules and accessories. In addition, the FPGA on Hinj includes Alorium Technology's AVR compatible 8-bit microcontroller making Hinj easily programmable and compatible with the popular and easy to use Arduino IDE.



1.1 Key Features

Hinj is loaded with the interfaces and built-in functionality that allows you to quickly create projects with a variety of interfaces and connectivity options. This list below provides a quick look at the primary interfaces Hinj offers.

ID	Description
1	Intel® MAX® 10 FPGA
2	Ethernet Port
3	Wi-Fi Module
4	MicroSD Card Slot
5	Arduino R3 Headers
6	PMOD Interfaces
7	XBEE Module Header
8	GPIO Interface

ID	Description
9	User Configurable Buttons
10	Assignable Switches
11	Programmable LED Bank
12	USB UART
13	Integrated USB Blaster
14	JTAG Interface
15	Barrel Connecter Power Jack
16	Reset Button

2 Usage and Applications

Hinj is a great platform for the development and testing of your IoT products and solutions. The integrated Ethernet and Wi-Fi provide a baseline for connectivity to internal networks, cloud based servers and IoT management tools. The Arduino R3 headers, PMOD connectors, and XBEE socket allow you to connect a wide variety of existing shields and modules to the Hinj board creating a highly flexible development platform.

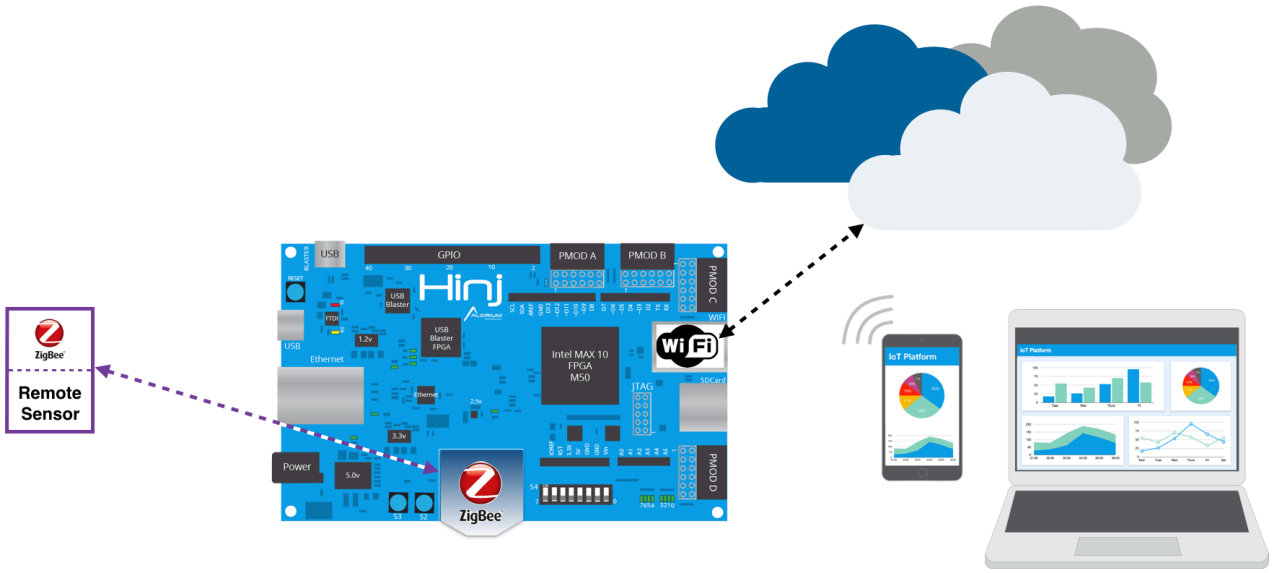
Arduino compatibility makes Hinj very easy to start using out of the box, while the generously sized 50K LE FPGA gives you plenty of room to create your optimized FPGA-based IoT design.

2.1 Example Development Setup: ZigBee Sensor Hub

The following diagram shows Hinj configured with a Digi XBee ZigBee module. The ZigBee interface on Hinj is receiving data from a remote node that is capturing accelerometer data.

Once data is received into Hinj, any kind of custom processing can be performed, and then the resulting data is pushed to a cloud-based IoT visualization platform for remote monitoring and control.

This example shows how easy it can be to leverage existing wireless modules and use them in combination with the integrated connectivity features to create a very simple IoT solution.



3 Functional Description

Hinj offers a rich combination of FPGA processing performance, physical interfaces, and integrated internet connectivity options. This section provides functional descriptions for the primary features of the Hinj board.

3.1 Intel MAX 10 FPGA

Hinj features a 50K LE Intel MAX 10 FPGA as the main processing engine. The FPGA includes an integrated 8-bit microcontroller which is fully instruction- and register-set compatible with the Atmel ATMega328. The remainder of the FPGA fabric is available for customized FPGA hardware functionality which we call Xcelerator Blocks.

An Xcelerator Block (XB) is an optimized hardware implementation of a given function, process or behavior that resides on the programmable portion of FPGA fabric. XBs communicate with the FPGA-based AVR microcontroller through the addressable register interface. Users can select from a library of pre-compiled FPGA images that implement various combinations of XBs and utilize Alorium's XB software libraries to easily take advantage of XB features and functionality.

Alorium's OpenXLR8 methodology allows developers to create custom XBs and chose from a library of XBs that can be combined into unique configurations, or combine Alorium's XBs with user logic to implement even more customized solutions, all the while maintaining compatibility with the standard AVR architecture, as well as with the Arduino ecosystem.

3.2 Integrated Wi-Fi and Ethernet

Hinj includes both Ethernet and Wi-Fi options for internet connectivity. Each interface has a dedicated SPI connection to the FPGA and available libraries that make the interfaces easy to access from the integrated microcontroller.

3.2.1 Wi-Fi

The Wi-Fi module is a Microchip ATWINC1500 supporting single-band 2.4GHz b/g/n connectivity and WPA/WPA2 Personal, TLS, and SSL security protocols. The module is easily configured and controlled from the microcontroller by using a customized version of the Arduino WiFi101 library available through the Hinj board support package.

3.2.2 Ethernet

The Ethernet interface is implemented with the Microchip ENC28J60 10BASE-T controller which includes on-board MAC and PHY, 8KB of buffer RAM, and a standard RJ45 connector. Alorium provides a customized version of the EtherCard library that provides configuration and control of the Ethernet module. The library is available through the Hinj board support package..

3.3 Standard Interfaces

Hinj includes several standard interfaces that make it highly adaptable and compatible with a wide variety of modules.

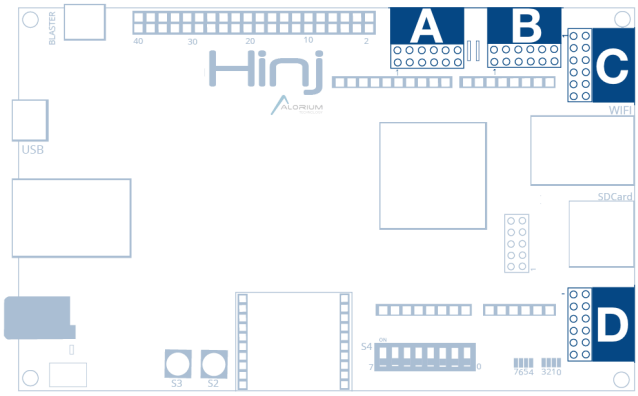
3.3.1 Arduino R3

The Arduino R3 Header allows users to take advantage of the vast number of Arduino shields available in the market to quickly expand the capabilities of the board.

3.3.2 Pmod™

The Peripheral Module interface (Pmod) is a standard developed by Digilent Inc. There are numerous Pmod modules and accessories that can be used with the four Hinj Pmod connectors.

Each connector is compatible with the Pmod standard and supports SPI, I2C, UART, and GPIO functionality. Hinj ships with the Pmod interfaces configured as follows:



Pmod-A: SPI
Pmod-B: I2C
Pmod-C: UART
Pmod-D: GPIO

3.3.3 XBee®

Hinj provides one XBee socket which allows for use of modules that adhere to the Digi XBee standard. There are many wireless radio options – including Wi-Fi, Cellular, and LPWA – that come in the XBee form factor.

3.3.4 GPIO

Hinj also supports 50 GPIO bits that can be configured by the user.

3.4 User I/O

The Hinj board includes two user-configurable buttons, eight assignable switches, and a programmable LED bank with eight LEDs. The factory-installed sketch running on the microcontroller will blink the LEDs in sequence, and the Quick Start Guide uses the on-board buttons, switches, and LEDs to introduce users to the Arduino IDE.

3.5 microSD Memory Slot

The microSD card socket on Hinj accepts a standard microSD card and utilizes a dedicated SPI connection to the FPGA. Alorium provides a version of the *sdfatlib* library to enable integration with the microcontroller. The library is provided as part of the Hinj board support package.

3.6 Programming

Hinj includes three interfaces that can be used to program the board.

3.6.1 Microcontroller and FPGA Programming with USB UART

Microcontroller

For users who want to take advantage of Alorium's AVR-compatible microcontroller, the preferred interface will be the USB UART interface.

Hinj uses the Optiboot bootloader that is used by the Arduino Uno, with a slight modification that allows it to run correctly at different CPU speeds. As an extra indication of Hinj's current CPU speed, you may notice when running at 32MHz that the beginning of an upload sequence blinks the pin 13 LED three times rather quickly just like an Arduino Uno, while at 16MHz you get two slightly slower blinks.

This bootloader is hardcoded into the design and cannot be changed by the user. If you have a need for something different in the bootloader, we'd be interested to hear about it. Consistent with using Optiboot, the BOOTRST "fuse" is hardcoded to zero (programmed) and the IVSEL and IVCE bits of the MCUCR (0x35) register are hardcoded to zero.

FPGA Image Updates

The USB UART interface can also be used to run Alorium's reconfiguration software for uploading new FPGA images to the MAX 10 via the USB UART interface.

3.6.2 Bare-Metal FPGA Programming with integrated USB Blaster or JTAG

For users who don't want or need the integrated 8-bit microcontroller, Hinj provides two methods for more traditional FPGA programming and configuration: the USB Blaster and JTAG.

USB Blaster

The integrated USB Blaster interface provides FPGA developers a path for programming the MAX 10 as a bare-metal FPGA design. This interface allows users to simply plug into Hinj with a USB Mini cable and program directly with Intel's Quartus Prime software. No extra USB Blaster hardware is required.

WARNING:

Connecting both USB ports simultaneously may cause a conflict on the computer. Only one of the USB connections should be used at a time.

JTAG

Hinj includes a standard 10 pin JTAG interface that is compatible with the Intel USB Blaster dongle that can also be used in conjunction with Quartus Prime for bare-metal FPGA configuration. The JTAG interface is not populated with a header from the factory because it is largely redundant with the USB Blaster interface. However, for developers with a JTAG programming workflow, the interface is available.

WARNING:

If a new image is loaded via the integrated USB Blaster or JTAG interfaces, the Alorium factory image that includes the AVR-compatible microcontroller will be overwritten, and Hinj will no longer function as an Arduino-compatible FPGA board.

Alorium Technology does NOT make the factory FPGA image available for download.

3.7 Power and Reset

Hinj ships with a 9V, 2A power supply that has been specifically selected for its output voltage and currently supply capabilities. We recommend using this supply with your Hinj board.

If you prefer to use a different power supply or power Hinj with a lab supply, Hinj uses a standard 5.5 x 2.1mm barrel jack connector, center positive, and requires 7V-14V input at a minimum of 500mA.

The on-board regulators on Hinj are capable of driving more current than what Hinj itself needs – up to nearly 3A on the 3.3V rail, or nearly 4A on the 5V rail, but if your accessories require more than that, or are prone to generating a lot of electrical noise, you should use a separate power supply for the accessories.

Hinj also includes a reset button, which is connected to the FPGA's DEV_CLRn pin, and is used to reset the on-chip microcontroller. As with an Arduino board, the reset input on the chip is also controlled by the on-board FTDI chip to allow for programming the microcontroller via the USB UART interface.

4 Technical Specifications

FPGA	Intel MAX 10 10M50DAF84C8G
Microcontroller	ATmega328 Compatible
I/O Voltage	3.3V
Supply Voltage	7-14V
Digital I/O Pins	50
PWM Digital Output Pins	6
Analog Input Pins	6
PMOD Interfaces (2x6)	4
XBEE Interfaces	1
Network Connectivity	Ethernet, Wi-Fi
On-board Storage	microSD card
User I/O	2 buttons, 8 switches, 8 LEDs
Programming Interfaces	FTDI via USB USB Blaster JTAG
Flash Memory	32 KB
SRAM	2 KB
Clock Speed	16/32 MHz
FPGA	10M50DAF84C8G
Physical Dimensions	5.1″ x 3.2″ 12.95 x 8.13 cm

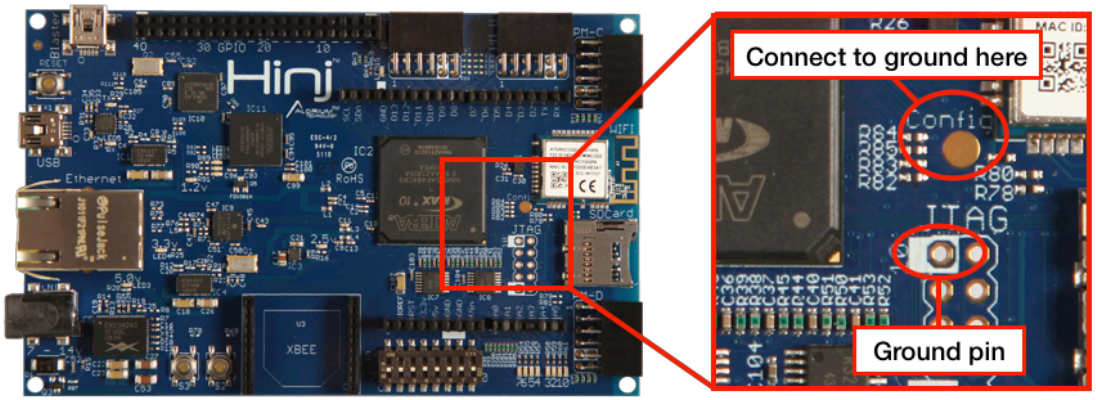
5 Restoring Factory FPGA Image

The Hinj FPGA is able to hold two different FPGA images. One of those, Image 1, is the primary working image and can be reconfigured from the Arduino IDE to take advantage of increased functionality as new XBs are introduced.

The reconfiguration files are obtained by installing our Arduino Board package as described in the instructions at https://github.com/AloriumTechnology/Arduino_Boards.

The other image, Image 0, is never changed and is typically unused unless the primary image 1 becomes corrupted.

If necessary, a “factory reset” of Hinj can be performed by grounding the pad labeled “Config” while applying power to the board.



It only takes a momentary grounding to cause this to happen. The factory image is then loaded. This factory image puts the Hinj board in a state that will allow it to be updated with the user’s desired FPGA image by using the Burn Bootloader FPGA update process with the Arduino IDE.

6 Hinj Library Reference

There are a number of software libraries available for using the Hinj board and accessing all of the integrated interfaces. Some of the libraries are required for basic functionality of the board, and others are needed only if using certain functions or features.

6.1.1 Base Libraries

The following list of libraries are included as part of the board support package for Hinj and do not need to be independently installed by the user:

- XLR8AddrPack
- XLR8HinjAddrPack
- XLR8Wire
- XLR8HardwareSerial
- XLR8SoftwareSerial
- XLR8SPI
- XLR8DigitalIO
- XLR8HinjWiFi101
- XLR8HinjEtherCard
- XLR8HinjGPIO
- XLR8HinjSD
- XLR8HinjXBee
- XLR8Info

6.1.2 Library Definitions

The tables below provide a quick overview of each library including a short description, dependencies, and applicable included examples.

XLR8AddrPack	
Description	Address defines specific to the XLR8 family of boards
Requires	None
Required By	Libraries requiring hardware interrupts
Examples	None

XLR8HinjAddrPack	
Description	Address defines specific to the XLR8 Hinj board
Requires	None
Required By	Libraries accessing the built-in devices on the Hinj board: SD, Ethernet, WiFi, Extra GPIO
Examples	None

XLR8Wire	
Description	Generic access to a hardware I2C interface on an XLR8 family board, such as a Hinj PMOD configured for I2C
Requires	None
Required By	None
Based On	Arduino Wire https://www.arduino.cc/en/Reference/Wire
Examples	None

XLR8HardwareSerial	
Description	Generic access to a hardware UART interface on an XLR8 family board, such as a Hinj PMOD configured for UART
Requires	None
Required By	None
Based On	Arduino Serial
Examples	None

XLR8SoftwareSerial	
Description	Generically create a software serial interface on arbitrary GPIO pins
Requires	None
Required By	XLR8HinjXBee
Based On	
Examples	None

XLR8GPIO	
Description	Generically access additional hardware GPIO on an XLR8 family board
Requires	None
Required By	Libraries using hardware GPIO: Hinj GPIO, SD, Ethernet, WiFi, XBee, XLR8 Software Serial
Examples	None

XLR8SPI	
Description	Generic access to hardware SPI XBs on an XLR8 family board
Requires	None
Required By	Libraries using a hardware SPI interface (SD, Ethernet, WiFi), and used if a Hinj PMOD is configured as SPI
Based On	Arduino SPI https://www.arduino.cc/en/Reference/SPI
Examples	XLR8SPIMaster, initializes a SPI interface using Arduino's built-in SPI, and attempts to write some data.

XLR8DigitalIO	
Description	Drop-in replacement for Digital IO that gives access to extra pins on Snō while still giving access to the standard Arduino Uno pins
Required By	None
Examples	None

XLR8HinjWiFi101	
Description	Access the Wi-Fi radio on a Hinj board
Requires	XLR8SPI
Required By	None
Based On	Arduino WiFi10 https://www.arduino.cc/en/Reference/WiFi101
Examples	HinjWiFiWebClient, uses the Wi-Fi radio to connect to an internet web server.

XLR8HinjEtherCard	
Description	Access the Ethernet port on a Hinj board
Requires	XLR8SPI
Required By	None
Based On	EtherCard for Arduino https://github.com/njh/EtherCard
Examples	HinjPings, uses the Ethernet port to ping a web server. HinjBackSoon, initializes a webserver external users can connect to through the Ethernet port.

XLR8HinjGPIO	
Description	Access the Hinj specific GPIO (40 pin bank, PMODs, switches, LEDs, XBee)
Requires	XRL8GPIO
Required By	XLR8HinjXBee
Examples	HinjLEDs, strobes the LEDs built onto the Hinj board. HinjUserIO, attaches the input of the Hinj built-in switches to the built-in LEDs.

XLR8Wire	
Description	Generic access to a hardware I2C interface on an XLR8 family board, such as a Hinj PMOD configured for I2C
Requires	None
Required By	None
Based On	Arduino Wire https://www.arduino.cc/en/Reference/Wire
Examples	None

XLR8HinjSD	
Description	Access the SD port on a Hinj board
Requires	XLR8SPI
Required By	None
Based On	Arduino SD https://www.arduino.cc/en/Reference/SD
Examples	HinjFiles, initializes the SD card, creates a file, and destroys it.

XLR8HinjXBee	
Description	Access the XBee port on a Hinj board as software serial
Requires	XLR8HinjGPIO, XLR8SoftwareSerial
Required By	None
Based On	Arduino SoftwareSerial https://www.arduino.cc/en/Reference/SoftwareSerial
Examples	HinjXBeePassthrough, passes data directly through the Serial port to the XBee interface.

XLR8Info	
Description	Gather and report information about the configuration of an Alorium board, including board type, clock speed, and installed XBs
Required By	None
Examples	GetXLR8Version, XLR8SelfTest

7 Register Map

The tables below define the Hinj software register map.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	SVPWH	–	–	–	–	Servo Pulse Width High Register				
(0xFC)	SVPWL	Servo Pulse Width Low Register								
(0xFB)	SVCR	SVEN	SVDIS	SVUP	SVCHAN					
(0xFA)	NEOD2	NeoPixel Data 2 register, function depends on NEOCMD								
(0xF9)	NEOD1	NeoPixel Data 1 register, function depends on NEOCMD, often high half of pixel address/length								
(0xF8)	NEOD0	NeoPixel Data 0 register, function depends on NEOCMD, often low half of pixel address/length								
(0xF7)	NEOCR	NEOPIN				NEOCMD				
(0xF6)	PID_OP_L	Low Byte output								
(0xF5)	PID_OP_H	High Byte output								
(0xF4)	PID_PV_L	Process variable low byte								
(0xF3)	PID_PV_H	Process variable high byte								
(0xF2)	PID_SP_L	Set point low byte								
(0xF1)	PID_SP_H	Set point high byte								
(0xF0)	PID_KP_L	KP coefficient low byte								
(0xEF)	PID_KP_H	KD coefficient high byte								
(0xEE)	PID_KI_L	KI coefficient low byte								
(0xED)	PID_KI_H	KI coefficient high byte								
(0xEC)	PID_KD_L	KD coefficient low byte								
(0xEB)	PID_KD_H	KD coefficient high byte								
(0xEA)	PIDCR	PEDEN	PIDDIS	PIDUPD	PIDCHAN					
(0xE9)	QERAT3	Upper 8 bits of quadrature rate data								
(0xE8)	QERAT2	Upper-middle 8 bits of quadrature rate data								
(0xE7)	QERAT1	Lower-middle 8 bits of quadrature rate data								
(0xE6)	QERAT0	Lower 8 bits of quadrature rate data								
(0xE5)	QECNT3	Upper 8 bits of quadrature count data								
(0xE4)	QECNT2	Upper-middle 8 bits of quadrature count data								
(0xE3)	QECNT1	Lower-middle 8 bits of quadrature count data								
(0xE2)	QECNT0	Lower 8 bits of quadrature count data								
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	QECR	QEEN	QEDIS	QECLR	QERATE	QECHAN				
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	**XLR8VERT**	XLR8 Version Number Flags								
(0xD5)	**XLR8VERH**	XLR8 Version Number Register High Byte								
(0xD4)	**XLR8VERL**	XLR8 Version Number Register Low Byte								
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	FCFGDAT	FPGA Reconfiguration Data Register								
(0xD1)	FCFGSTS	FCFGDN	0	FCFGFM	FCFGRDY	–	–	–	–	
(0xD0)	FCFGCTL	–	FCFGSEC			–	FCFGCMD		FCFGEN	
(0xCF)	FCFGCID	Chip ID register								

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	MSKBI	MSKBI[7:0]								
(0xCA)	MSKPA	MSKPA[7:0]								
(0xC9)	MSKPB	MSKPB[7:0]								
(0xC8)	MSKPC	MSKPC[7:0]								
(0xC7)	MSKPD	MSKPD[7:0]								
(0xC6)	UDR0	USART I/O Data Register								
(0xC5)	UBRR0H	–	–	–	–	USART Baud Rate Register High				
(0xC4)	UBRR0L	USART Baud Rate Register Low								
(0xC3)	MSKG0	MSKG0[7:0]								
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01/ UDORD0	UCSZ00/ UCPHA0	UCPOL0	
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	
(0xBF)	MSKG1	MSKG1[7:0]								
(0xBE)	MSKG2	MSKG2[7:0]								
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–	
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	
(0xBB)	TWDR	2-wire Serial Interface Data Register								
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								
(0xB7)	MSKG3	MSKG3[7:0]								
(0xB6)	Reserved	–	–	–	–	–	–	–		
(0xB5)	MSKG4	MSKG4[7:0]								
(0xB4)	OCR2B	Timer/Counter2 Output Compare Register B								
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	
(0xAF)	MSKX0	MSKX0[7:0]								
(0xAE)	MSKX1	MSKX1[7:0]								
(0xAD)	HPCIMSK	--	HPCIMSK[6:0]							
(0xAC)	HPCIFR	--	HPCIFR[6:0]							
(0xAB)	HPCICR	--	HPCICR[6:0]							
(0xAA)	SD_SPDR	SD_SPDR[7:0]								
(0xA9)	SD_SPSR	SD_SPSR[7:0]								
(0xA8)	SD_SPCR	SD_SPCR[7:0]								
(0xA7)	ETH_SPDR	ETH_SPDR[7:0]								
(0xA6)	ETH_SPSR	ETH_SPSR[7:0]								
(0xA5)	ETH_SPCR	ETH_SPCR[7:0]								
(0xA4)	WIFI_SPDR	WIFI_SPDR[7:0]								
(0xA3)	WIFI_SPSR	WIFI_SPSR[7:0]								
(0xA2)	WIFI_SPCR	WIFI_SPCR[7:0]								
(0xA1)	PINBI	PINBI[7:0]								
(0xA0)	DDRBI	DDRBI[7:0]								
(0x9F)	PORTBI	PORTBI[7:0]								
(0x9E)	PINPA	PINPA[7:0]								
(0x9D)	DDRPA	DDRPA[7:0]								
(0x9C)	PORTPA	PORTPA[7:0]								
(0x9B)	PINPB	PINPB[7:0]								

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x9A)	DDRPB	DDRPB[7:0]								
(0x99)	PORTPB	PORTPB[7:0]								
(0x98)	PINPC	PINPC[7:0]								
(0x97)	DDRPC	DDRPC[7:0]								
(0x96)	PORTPC	PORTPC[7:0]								
(0x95)	PINPD	PINPD[7:0]								
(0x94)	DDRPD	DDRPD[7:0]								
(0x93)	PORTPD	PORTPD[7:0]								
(0x92)	PING0	PING0[7:0]								
(0x91)	DDRG0	DDRG0[7:0]								
(0x90)	PORTG0	PORTG0[7:0]								
(0x8F)	PING1	PING1[7:0]								
(0x8E)	DDRG1	DDRG1[7:0]								
(0x8D)	PORTG1	PORTG1[7:0]								
(0x8C)	PING2	PING2[7:0]								
(0x8B)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								
(0x8A)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								
(0x89)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								
(0x88)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								
(0x87)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								
(0x86)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								
(0x85)	TCNT1H	Timer/Counter1 – Counter Register High Byte								
(0x84)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								
(0x83)	DDRG2	DDRG2[7:0]								
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	
(0x7F)	Reserved	–	–	–	–	–	–	–	–	
(0x7E)	DIDR0	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	
(0x7D)	XLR8ADCR	AD12EN	–	–	–	–	–	–	–	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	
(0x7B)	ADCSRB	–	-	–	–	–	ADTS2	ADTS1	ADTS0	
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	
(0x79)	ADCH	ADC Data Register High byte								
(0x78)	ADCL	ADC Data Register Low byte								
(0x77)	PORTG2	PORTG2[7:0]								
(0x76)	PINX0	PINX0[7:0]								
(0x75)	DDRX0	DDRX0[7:0]								
(0x74)	PORTX0	PORTX0[7:0]								
(0x73)	PINX1	PINX1[7:0]								
(0x72)	DDRX1	DDRX1[7:0]								
(0x71)	PORTX1	PORTX1[7:0]								
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	
(0x6A)	OX8MSK	OpenXLR8 Pin Control Interrupt Mask								
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0	
(0x67)	OX8IFR	OpenXLR8 Pin Control Interrupt Flag Register								
(0x66)	OX8ICR	OpenXLR8 Pin Control Interrupt Register								

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x65)	XACK	XLR8 IRQ Acknowledge Register								
(0x64)	PRR	–	–	–	PRINTOSC	–	–	–	–	
(0x63)	XMSK	XLR8 IRQ Mask Register								
(0x62)	XIFR	XLR8 IRQ Flag Register								
(0x61)	XICR	XLR8 IRQ Control Register								
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	
0x3F(0x5F)	SREG	I	T	H	S	V	N	Z	C	
0x3E(0x5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	
0x3D(0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	
0x3C(0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B(0x5B)	**XFR3**	XLR8 Function (floating point) 32 bit Result High Byte								
0x3A(0x5A)	**XFR2**	XLR8 Function (floating point) 32 bit Result Byte								
0x39(0x59)	**XFR1**	XLR8 Function (floating point) 32 bit Result Byte								
0x38(0x58)	**XFR0**	XLR8 Function (floating point) 32 bit Result Low Byte								
0x37(0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	
0x36(0x56)	PING3	PING3[7:0]								
0x35(0x55)	Reserved	–	–	–	–	–	–	–	–	
0x34(0x54)	MCUSR	–	–	–	–	WDRF	–	EXTRF	PORF	
0x33(0x53)	DDRG3	DDRG3[7:0]								
0x32(0x52)	PORTG3	PORTG3[7:0]								
0x31(0x51)	PING4	–	–	–	–	PING4[3:0]				
0x30(0x50)	Reserved	–	–	–	–	–	–	–	–	
0x2F(0x4F)	DDRG4	–	–	–	–	DDRG4[3:0]				
0x2E(0x4E)	SPDR	SPI Data Register								
0x2D(0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	
0x2C(0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	
0x2B(0x4B)	GPIOR2	General Purpose I/O Register 2								
0x2A(0x4A)	GPIOR1	General Purpose I/O Register 1								
0x29(0x49)	CLKSPD	Clock speed programming used by XLR8 bootloader								
0x28(0x48)	OCR0B	Timer/Counter0 Output Compare Register B								
0x27(0x47)	OCR0A	Timer/Counter0 Output Compare Register A								
0x26(0x46)	TCNT0	Timer/Counter0 (8-bit)								
0x25(0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	
0x24(0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	
0x23(0x43)	GTCCR	TSM	–	–	–	–	–	PSRASY	PSRSYNC	
0x22(0x42)	EEARH	EEPROM Address Register High Byte								
0x21(0x41)	EEARL	EEPROM Address Register Low Byte								
0x20(0x40)	EEDR	EEPROM Data Register								
0x1F(0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	
0x1E(0x3E)	GPIOR0	General Purpose I/O Register 0								
0x1D(0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	
0x1C(0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	
0x1B(0x3B)	PCIFR	–	–	–	–	–	PCIF2	PCIF1	PCIF0	
0x1A(0x3A)	PORTG4	–	–	–	–	PORTG4[3:0]				
0x19(0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18(0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17(0x37)	TIFR2	–	–	–	–	–	OCF2B	OCF2A	TOV2	
0x16(0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	
0x15(0x35)	**TIFR0**	**–**	**–**	**–**	**–**	**–**	**OCF0B**	**OCF0A**	**TOV0**	
0x14(0x34)	**PINLD**	**PINLD[7:0]**								
0x13(0x33)	**DDRLD**	**DDRLD[7:0]**								
0x12(0x32)	**PORTLD**	**PORTLD[7:0]**								
0x11(0x31)	**XFSTAT**	**XFDONE**	**XFERR**	**–**	**–**	**–**	**–**	**–**	**–**	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x10(0x30)	XFCTRL	–	XFSTART	–	–	–	XFCMD			
0x0F(0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E(0x2E)	PINSW	PINSW[7:0]								
0x0D(0x2D)	DDRSW	DDRSW[7:0]								
0x0C(0x2C)	PORTSW	PORTSW[7:0]								
0x0B(0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	
0x0A(0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	
0x09(0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	
0x08(0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	
0x07(0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	
0x06(0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	
0x05(0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	
0x04(0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	
0x03(0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	
0x02(0x22)	PINBT	–	–	–	–	–	–	PINBT[1:0]		
0x01(0x21)	DDRBT	–	–	–	–	–	–	DDRBT[1:0]		
0x00(0x20)	PORTBT	–	–	–	–	–	–	PORTBT[1:0]		
		= unchanged from ATmega328p								
		= ATmega328p registers not implemented in XLR8								
		= Some differences in XLR8 compared to ATmega328p								
		= new registers for XLR8 Blocks								
		= Reserved registers that are best not used for XLR8 blocks because ATmega328PB uses them								
		= Hinj specific registers								

7.1 XLR8 and XB Register Descriptions

7.1.1 XLR8VERL, XLR8VERH, XLR8VERT – Version Number Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xD6)	**XLR8VERT**	XLR8 Version Number Flags								
(0xD5)	**XLR8VERH**	XLR8 Version Number Register High Byte								
(0xD4)	**XLR8VERL**	XLR8 Version Number Register Low Byte								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The version number register provides the FPGA design revision, while the version flags register indicates if the build had a mixed or modified version. The registers have a constant value for a particular design, but the value changes for each version. The easiest way to use these registers is with the XLR8Info library (https://github.com/AloriumTechnology/XLR8Info).

7.1.2 FCFGCID – Chip ID Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xCF)	FCFGCID	Chip ID register								
Read/Write		R	R	R	R	R	R	R	R	write-reset
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The chip ID register is a read-only register that provides chip ID information. Multiple bytes of chip ID information are available and each read presents the next byte. Writing the register (with any value) resets the read pointer back to the beginning (and does not store the write data in any way).

7.1.3 CLKSPD – Clock Speed Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x29(0x49)	CLKSPD	Clock speed programming used by XLR8 bootloader								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
0x29(0x49)	CLKSPD	–	–	–	–	–	–	–	OSCOUT	
Read/Write		W	W	W	W	W	W	W	W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	
(0x64)	**PRR**	-	–	–	–	PRINTOSC	–	–	–	
Read/Write		R	R	R	R	R/W	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	

The clock speed register holds a constant value that represents the value to be programmed into the UBRR0L register to run the UART at a baud rate of 115200. It is used by bootloader to allow it to run correctly regardless of whether XLR8 is running 16MHZ, 32MHz, or some other speed.

XLR8 includes in on-chip oscillator that currently isn't being used, but a divide-by-1024 version of it can be output to digital pin 8 by writing bit 0 of the CLKSPD register high. This is a write-only operation, it does not change the value that is read from the CLKSPD register. The internal oscillator can be turned off entirely by setting the PRINTOSC bit of the PRR register. The other bits of this register are currently unused.

7.1.4 XICR, XIFR, XMSK, XACK - XLR8 IRQ Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x61)	XICR	–	–	–	–	–	–	OX8	BI	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	
(0x62)	XIFR	–	–	–	–	–	–	OX8	BI	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	
(0x63)	XMSK	-	–	–	–	-	–	OX8	BI	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	
(0x65)	XACK	-	–	–	–	--	–	OX8	BI	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	

For a description of the XLR8 IRQ Registers, see the Pin Change Interrupt Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

There are some differences of note. The XACK register prevents the XIFR bit from getting set after the ISR has been invoked. Before exiting the ISR, the XACK bit should be cleared.

Bit 0 is for Built-In interrupt sources, such as extra GPIO interface pin change interrupts on the Snō and Hinj boards.

Bit 1 is for interrupts defined by the user in the OpenXLR8 block.

7.1.5 OX8ICR, OX8IFR, OXAMSK - OpenXLR8 Pin Change Interrupts

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x66)	OX8ICR	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x67)	OX8IFR	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x6A)	OX8MSK	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

For a description of the OpenXLR8 Pin Change Interrupt registers, see the Pin Change Interrupt Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

The interrupt sources for the OpenXLR8 interrupts are defined by the user when designing the OpenXLR8 block.

7.1.6 FCFGDAT, FCFGSTS, FCFGCTL – FPGA Reconfiguration Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xD2)	FCFGDAT	FPGA Reconfiguration Data Register								
Read/Write		W	W	W	W	W	W	W	W	
Initial Value		0	0	0	0	0	0	0	0	
(0xD1)	FCFGSTS	FCFGDN	FCFGOK	FCFGFAIL	FCFGRDY	–	–	–	–	
Read/Write		R/W1C	R/W1C	R/W1C	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xD0)	FCFGCTL	–	FCFGSEC			–	FCFGCMD		FCFGEN	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

These registers are used during reconfiguration of the FPGA and are not intended for customer use. FCFGEN auto-clears after a reconfiguration is complete. The Data register is a write-only register.

7.1.7 SVPWH, SVPWL, SVCR – Servo XB Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xFD)	SVPWH	–	–	–	–	Servo Pulse Width High Register				
Read/Write		R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xFC)	SVPWL	Servo Pulse Width Low Register								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xFA)	SVCR	SVEN	SVDIS	SVUP	SVCHAN					
Read/Write		R/W	W	W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

The servo data registers SVPWH and SVPWL represent the desired servo pulse width in microseconds. The value is programmed to the channel selected by SVCHAN when the SVCR register is written with the update (SVUP) bit set. The channel can be enabled to begin at the same time by also setting the enable (SVEN) bit. A channel is disabled by writing SVCR with the desired channel in the SVCHAN field, the SVEN bit clear and the SVDIS set. The pulse width of a channel can be changed without changing its enabled/disabled status by leaving the SVEN and SVDIS bits clear when writing the SVCR register. SVDIS and SVUP are strobes and will always read zero. Reading SVEN will give the current enabled/disabled status of the channel read in the SVCHAN field. The value of SVCHAN corresponds to the Arduino pin to use (i.e. 0=RX, 1=TX, 2=D2, …, 14=A0, etc.). Multiple pins can be driven simultaneously, each with a different pulse width…with a small limitation. The 32 possible values of SVCHAN directly alias to the 16 available timers (e.g. channels 1 and 17 could both be enabled, but they would always have the same pulse width of whichever one was programmed most recently). The easiest way to use these registers is with the XLR8Servo library (https://github.com/AloriumTechnology/XLR8Servo).

7.1.8 NEOD2, NEOD1, NEOD0, NEOCR – NeoPixel XB Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xF7)	NEOD2	NeoPixel Data 2 register, function depends on NEOCMD								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF6)	NEOD1	NeoPixel Data 1 register, function depends on NEOCMD, often high half of pixel address/length								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF5)	NEOD0	NeoPixel Data 0 register, function depends on NEOCMD, often low half of pixel address/length								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF4)	NEOCR	NEOPIN				NEOCMD				
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

The NeoPixel register usage varies depending on the value of the NEOCMD field in the NeoPixel control register (NEOCR).

NEOPIN : For the show command, this specifies which pin to send the data to. For no-op a pin number or other ID can be used. For other commands this field is ignored and not stored. For show the value clears to zero when the command completes and thereby can be used as a busy indication. While the hardware doesn't provide any locks, this may help software manage having multiple objects share this hardware nicely. The outputs are indexed starting at 1 because using the clear to zero to indicate busy/notbusy wouldn't work with pin 0.

NEOCMD : 0000 = no-op, but set PinNum/BusyID (won't autoclear until after show)
0001 = get memsize. D2=cmd buf size (entries), D1/D0= pixel mem size (bytes)
0010 = show WS2812. D2=starting cmd buffer D1/D0=length (bytes)
0011 = show WS2811. D2=starting cmd buffer D1/D0=length (bytes)
0100-0111 = Reserved.
1000 = set color. D2=color value, D1/D0=memory address (autoincrement)
1001 = set cmd buf entry-addr. D2=cmd buf addr, D1/D0=section start addr
1010 = set cmd buf entry-length. D2=cmd buf addr, D1/D0=section length
1011 = set cmd buf entry-bright. D2=cmd buf addr, D1=reserved, D0=brightness
1100 = get color. D2=color value, D1/D0=memory address (autoincrement)
1101 = get cmd buf entry-addr. D2=cmd buf addr, D1/D0=section start addr
1110 = get cmd buf entry-length. D2=cmd buf addr, D1/D0=section length
1111 = get cmd buf entry-bright. D2=cmd buf addr, D1=reserved, D0=brightness

NEOD2/D1/D0 : 8b data registers used as described above

For set, and show operations, D2/D1/D0 are loaded before doing the set/show cmd in the CNTL register (except possibly doing a no-op to set the busy ID).

For get color, D1/D0 are set first, then CNTL, then the color value can be read from D2.

For get cmd buf, D2 is set first, then CNTL, then the cmd buf info can be read from D1/D0.

For get memsize, CNTL is written, and then the memsize can be read from D2/D1/D0.

The easiest way to use these registers is with the XLR8NeoPixel library (https://github.com/AloriumTechnology/XLR8NeoPixel).

7.1.9 XFCTRL, XFSTAT, XFR0, XFR1, XFR2, XFR3– Floating Point XB Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x3B(0x5B)	XFR3	XLR8 Function (floating point) 32 bit Result High Byte								
0x3A(0x5A)	XFR2	XLR8 Function (floating point) 32 bit Result Byte								
0x39(0x59)	XFR1	XLR8 Function (floating point) 32 bit Result Byte								
0x38(0x58)	XFR0	XLR8 Function (floating point) 32 bit Result Low Byte								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
0x11(0x31)	XFSTAT	XFDONE	XFERR	–	–	–	–	–	–	
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
0x10(0x30)	XFCTRL	–	XFSTART	–	–	–	XFCMD			
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

A floating-point calculation is started by writing the XFSTART bit in the XFCTRL register, along with the desired operation in the XFCMD field (1=add, 2=multiply, 3=divide). Operands come directly from the AVR's general-purpose register file (using our library ensures they will be in the right place). When the operation is done, the result appears in the XFR0/1/2/3 registers and the XFDONE status bit is set. If an unsupported XFCMD is used, the XFERR bit is also sets, allowing software to revert to using a software-based calculation. The XFSTAT register auto-clears when it is read, or when the next operation is started via writing the XFSTART bit.
The easiest way to use these registers is with the XLR8Float library (https://github.com/AloriumTechnology/XLR8Float).

7.1.10 XLR8ADCR – XLR8 ADC Control Register

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x7D)	XLR8ADCR	AD12EN	–	–	–	–	–	–	–	
Read/Write		R/W	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	

The AD12EN bit enables the ADC to run in 12 bit mode. The results reported in the ADCL and ADCH registers when running with ADLAR=0 can range from 0-4095, and when running with ADLAR=1, bits 5:4 of ADCL will include the least significant bits of the 12 bit ADC result. When running in 10 bit mode (the default to match Arduino), the result is truncated (not rounded) from the 12 bit result.

7.1.11 XLR8PID – XLR8 PID

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xF6)	PID_OP_L	Low Byte output								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xF5)	PID_OP_H	High Byte output								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xF4)	PID_PV_L	Process variable low byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF3)	PID_PV_H	Process variable high byte								

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF2)	PID_SP_L	Set point low byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF1)	PID_SP_H	Set point high byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xF0)	PID_KP_L	KP coefficient low byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xEF)	PID_KP_H	KP coefficient high byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xEE)	PID_KI_L	KI coefficient low byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xED)	PID_KI_H	KI coefficient high byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xEC)	PID_KD_L	KD coefficient low byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xEB)	PID_KD_H	KD coefficient high byte								
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xEA)	PIDCR	PEDEN	PIDDIS	PIDUPD	PIDCHAN					
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

To start a channel typically the channel is reset first, then the control register with the desired channel indicated and both the enable and update bits set.

7.1.12 XLR8Quad – XLR8 Quadrature

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xE9)	QERAT3	Upper 8 bits of quadrature rate data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE8)	QERAT2	Upper-middle 8 bits of quadrature rate data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE7)	QERAT1	Lower-middle 8 bits of quadrature rate data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE6)	QERAT0	Lower 8 bits of quadrature rate data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE5)	QECNT3	Upper 8 bits of quadrature count data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE4)	QECNT2	Upper-middle 8 bits of quadrature count data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xE3)	QECNT1	Lower-middle 8 bits of quadrature count data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE2)	QECNT0	Lower 8 bits of quadrature count data								
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
(0xE0)	QECR	QEEN	QEDIS	QECLR	QERATE	QECHAN				
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

To start a channel typically the channel is reset first, then the control register with the desired channel indicated and both the enable and update bits set.

7.1.13 Additional Hinj Ports

The Hinj board implements additional ports beyond the standard Arduino ports. They operate the same way as the normal Arduino ports unless otherwise noted in the specific definitions. The standard Arduino behavior is:

The DDRxx bit in the DDRxx Register selects the direction of this pin. If DDxx is written logic one, the pin is configured as an output pin. If DDxx is written logic zero, the pin is configured as an input pin.

If PORTxx is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxx is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

Independent of the setting of Data Direction bit DDxx, the port pin can be read through the PINxx Register bit.

7.1.13.1 PORTBT, DDRBT, PINBT - Port BT – Buttons Port

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x00(0x20)	PORTBT	--	--	--	--	--	--	--	--	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	
0x01(0x21)	DDRBT	--	--	–	–	–	–	–	--	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	0	0	
0x02(0x22)	PINBT	--	--	–	–	--	–	–	–	
Read/Write		N/A	N/A	N/A	N/A	N/A	N/A	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Button values are controlled by the state of the button itself and cannot be set from software. The values can only be read via the PINBT reg.

7.1.13.2 PORTSW, DDRSW, PINSW - Port SW – Switch Port

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x0C(0x2C)	PORTSW	--	--	--	--	--	--	--	--	
Read/Write		R	R	R	R	R	R	R	R	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
Initial Value		0	0	0	0	0	0	0	0	
0x0D(0x2D)	DDRSW	–	–	–	–	–	–	–	--	
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		0	0	0	0	0	0	0	0	
0x0E(0x2E)	PINSW	–	–	–	–	–	–	–	–	
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Switch values are controlled by the state of the button itself and cannot be set from software. The values can only be read via the PINSW reg.

7.1.13.3 PORTLD, DDRLD, PINLD - Port LD – LED Port

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x12(0x32)	PORTLD	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
0x13(0x33)	DDRLD	--	--	--	--	--	--	--	--	
Read/Write		R	R	R	R	R	R	R	R	
Initial Value		1	1	1	1	1	1	1	1	
0x14(0x34)	PINLD	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The LED port pins are always output pins, so the DDLD values are always set to one and are read-only.

7.1.13.4 PORTG4, DDRG4, PING4 - Port G4 – GPIO Port 4

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x1A(0x3A)	PORTG4	--	--	--	--	--	--	--	--	
Read/Write		N/A	N/A	N/A	N/A	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	0	0	0	0	
0x2F(0x4F)	DDRG4	--	--	--	--	--	--	--	--	
Read/Write		N/A	N/A	N/A	N/A	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	0	0	0	0	
0x31(0x51)	PING4	--	--	--	--	--	--	--	–	
Read/Write		N/A	N/A	N/A	N/A	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

GPIO Port 4 is only four bits wide, whereas the other GPIO ports are eight bits wide.

7.1.13.5 PORTG3, DDRG3, PING3 - Port G3 – GPIO Port 3

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
0x32(0x52)	PORTG3	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
0x33(0x53)	DDRG3	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
0x36(0x56)	PING3	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.6 PORTG2, DDRG2, PING2 - Port G2 – GPIO Port 2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x77)	PORTG2	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x83)	DDRG2	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x8C)	PING2	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.7 PORTG1, DDRG1, PING1 - Port G1 – GPIO Port 1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x8D)	PORTG1	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x8E)	DDRG1	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x8F)	PING1	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.8 PORTG0, DDRG0, PING0 - Port G0 – GPIO Port 0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x90)	PORTG0	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x91)	DDRG0	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x92)	PING0	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.9 PORTX1, DDRX1, PINX1 - Port X1 – XBEE Port 1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x71)	PORTX1	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x72)	DDRX1	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x73)	PINX1	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.10 PORTX0, DDRX0, PINX0 - Port X0 – XBEE Port 0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x74)	PORTX0	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x75)	DDRX0	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x76)	PINX0	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.11 PORTPD, DDRPD, PINPD - Port PD – PMOD Port D

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x93)	PORTPD	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x94)	DDRPD	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x95)	PINPD	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.12 PORTPC, DDRPC, PINPC - Port PC – PMOD Port C

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x96)	PORTPC	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x97)	DDRPC	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x98)	PINPC	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.13 PORTPB, DDRPB, PINPB - Port PB – PMOD Port B

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x99)	PORTPB	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x9A)	DDRPB	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x9B)	PINPB	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.14 PORTPA, DDRPA, PINPA - Port PA – PMOD Port A

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x9C)	PORTPA	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x9D)	DDRPA	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0x9E)	PINPA	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.13.15 PORTBI, DDRBI, PINBI - Port BI – BuiltIn Port

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0x9F)	PORTBI	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA0)	DDRBI	--	--	--	--	--	--	--	--	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA1)	PINBI	--	--	--	--	--	--	--	–	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

7.1.14 WIFI_SPCR, WIFI_SPSR, WIFI_SPDR - WIFI SPI Interface

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xA2)	WIFI_SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA3)	WIFI_SPSR	SPIF	WCOL	--	--	--	--	--	SPI2x	
Read/Write		R	R	R	R	R	R	R	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA4)	WIFI_SPDR	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		X	X	X	X	X	X	X	X	

For a description of the SPI registers, see the SPI Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

7.1.15 ETH_SPCR, ETH_SPSR, ETH_SPDR - ETH SPI Interface

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xA5)	ETH_SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA6)	ETH_SPSR	SPIF	WCOL	--	--	--	--	--	SPI2x	
Read/Write		R	R	R	R	R	R	R	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA7)	ETH_SPDR	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		X	X	X	X	X	X	X	X	

For a description of the SPI registers, see the SPI Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

7.1.16 SD_SPCR, SD_SPSR, SD_SPDR - SD SPI Interface

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xA8)	SD_SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xA9)	SD_SPSR	SPIF	WCOL	--	--	--	--	--	SPI2x	
Read/Write		R	R	R	R	R	R	R	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xAA)	SD_SPDR	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		X	X	X	X	X	X	X	X	

For a description of the SPI registers, see the SPI Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

7.1.17 HPCICR, HPCIFR, HPCIMSK -Hinj Pin Change Interrupts

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xAB)	HPCICR	--	PORTBI	PORTG	PORTG	PORTPD	PORTPC	PORTPB	PORTPA	
Read/Write		N/A	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	0	0	0	0	0	0	0	
(0xAC)	HPCIFR	--	PORTBI	PORTG	PORTG	PORTPD	PORTPC	PORTPB	PORTPA	
Read/Write		N/A	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	0	0	0	0	0	0	0	
(0xAD)	HPCIMSK	--	PORTBI	PORTG	PORTG	PORTPD	PORTPC	PORTPB	PORTPA	
Read/Write		N/A	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		N/A	0	0	0	0	0	0	0	

For a description of the Hinj Pin Change Interrupt registers, see the Pin Change Interrupt Register Definition in the Atmel 8-bit AVR Microcontroller Datasheet.

7.1.18 *MSKX1, MSKX0, MSKG4, MSKG3, MSKG2, MSKG1, MSKG0, MSKPD, MSKPC, MSKPB, MSKPA, MSKPI - Hinj GPIO Port Mask Registers*

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
(0xAE)	MSKX1	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xAF)	MSKX0	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xB5)	MSKG4	--	--	--	--	MSB	--	--	LSB	
Read/Write		N/A	N/A	N/A	N/A	R/W	R/W	R/W	R/W	
Initial Value		N/A	N/A	N/A	N/A	0	0	0	0	
(0xB7)	MSKG3	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xBE)	MSKG2	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Notes
Initial Value		0	0	0	0	0	0	0	0	
(0xBF)	MSKG1	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xC3)	MSKG0	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xC7)	MSKPD	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xC8)	MSKPC	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xC9)	MSKPB	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xCA)	MSKPA	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	
(0xCB)	MSKBI	MSB	--	--	--	--	--	--	LSB	
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value		0	0	0	0	0	0	0	0	

Much like the Mask registers for the Hinj Pin Change Interrupt registers, the Mask bits control whether a change to the state of a particular pin will cause a pin change interrupt to be forwarded to the Hinj Pin Change Interrupt Flag register. If the mask bit corresponding to the pin that changes is a one, then the pin change interrupt will be forwarded. If the mask bit is a zero then it will not be forwarded.

8 Additional Documentation

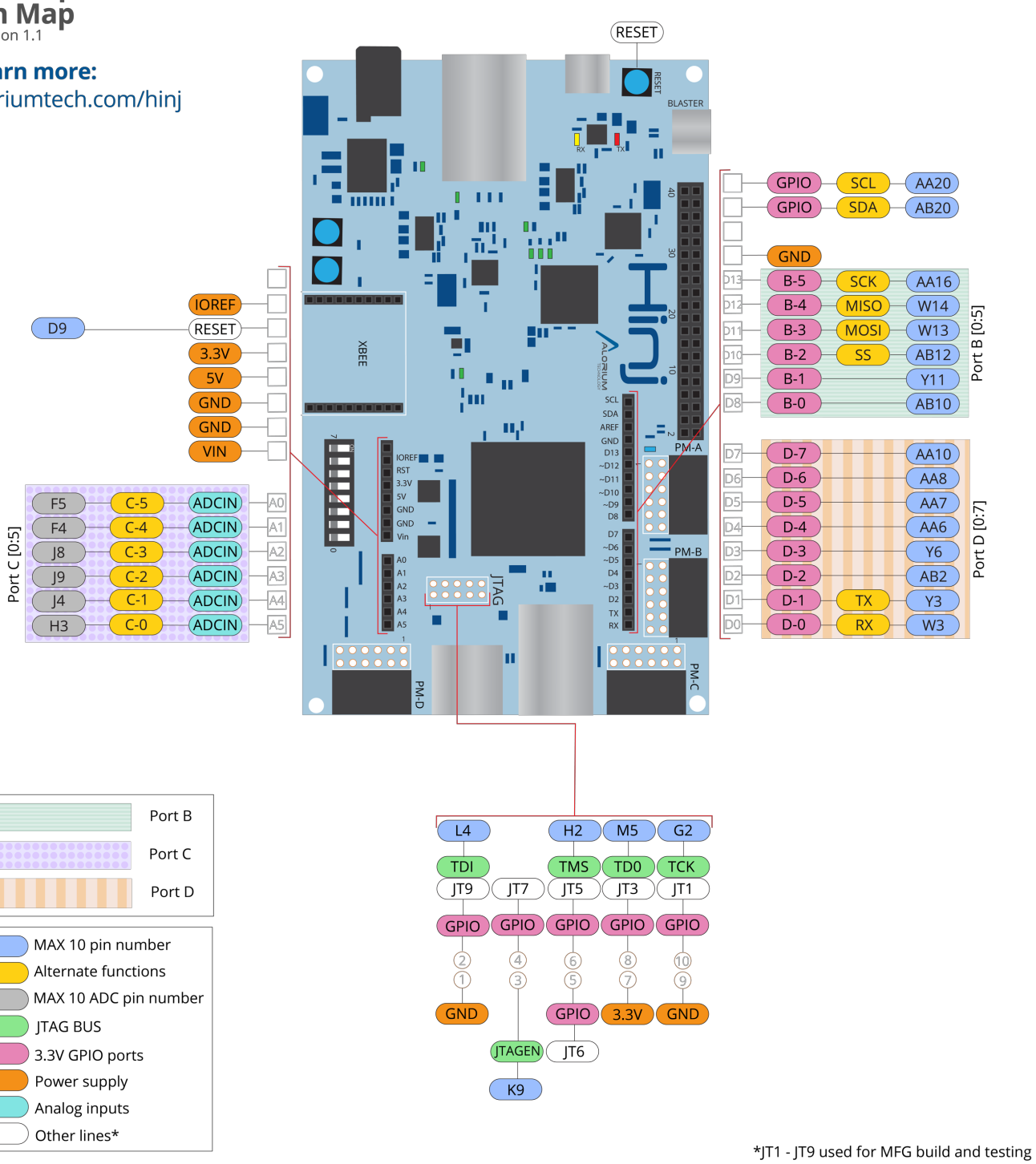
Additional support documentation is available on the Hinj support page:

http://www.aloriumtech.com/hinj-support

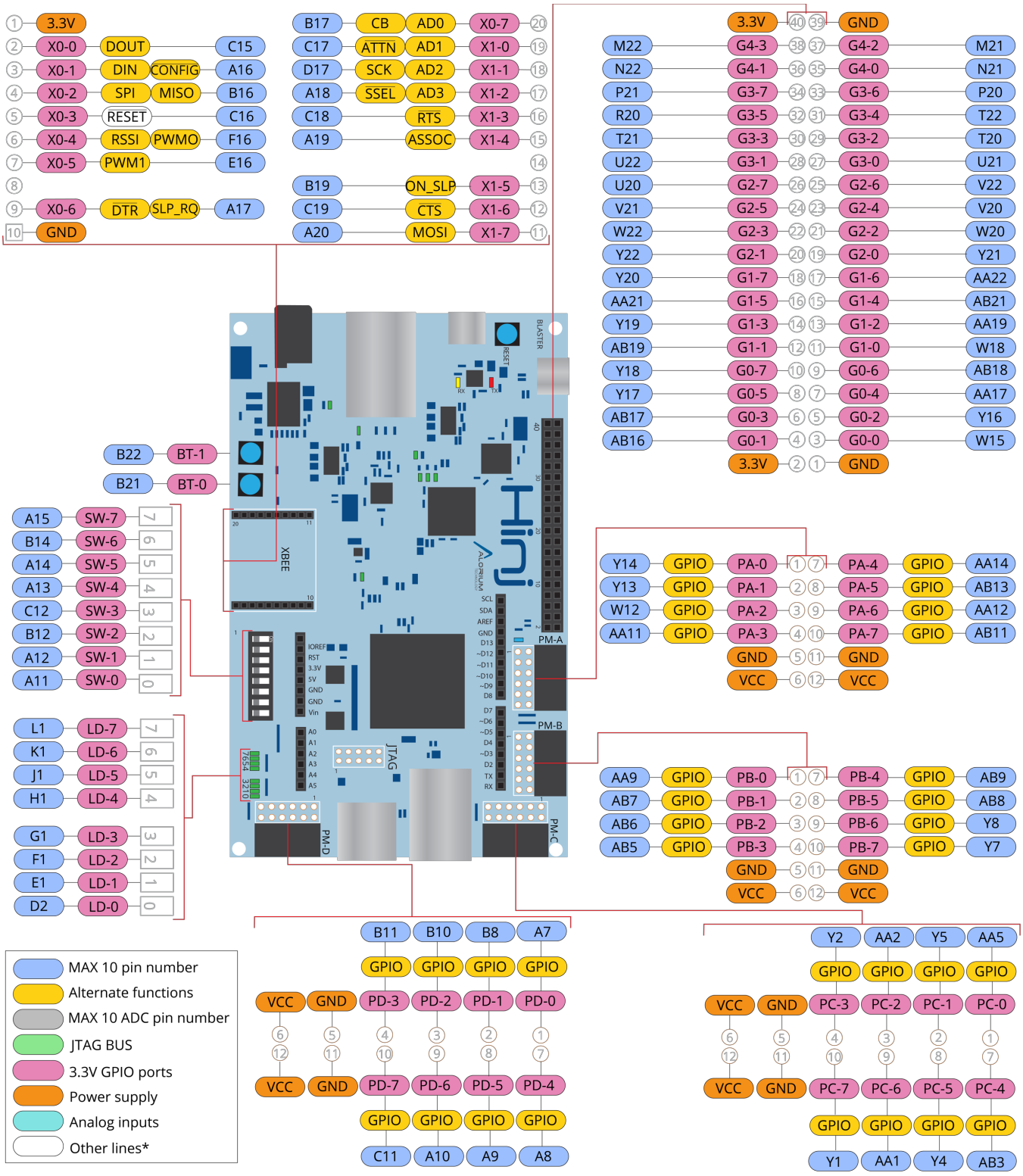
Appendix A – Pin Maps

Hinj
Development Board Pin Map
Version 1.1

Learn more:
aloriumtech.com/hinj



*JT1 - JT9 used for MFG build and testing



Appendix C - Credits

Some code is used and modified from the AVR core written by Ruslan Lepetenok (lepetenokr@yahoo.com) that is available at http://opencores.com/project,avr_core. Ruslan's AVR core does not contain copyright or license notices, but we certainly wish to recognize its contribution to this project.

The I2C module builds upon the I2C core written by Richard Herveille (richard@asics.ws) that is available at http://opencores.org/project,i2c. The I2C core was released under BSD license with the following copyright statement:

Copyright (C) 2001 Richard Herveille richard@asics.ws
This source file may be used and distributed without
restriction provided that this copyright statement is not
removed from the file and that any derivative work contains
the original copyright notice and the associated disclaimer
THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE AUTHOR
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

All other portions of Hinj were designed and verified for Alorium Technology by the excellent team at Superion Technology.