Systronix
JRealTime
Hardware

PRELIMINARY

- Technical Reference for
  Systronix JStik
  Realtime Native Java Network Module
  Revised 08 July 2003

JStik

LIMITED WARRANTY

The information in this manual is subject to change without notice and does not represent a commitment on the part of Systronix, Inc. Systronix, Inc. makes no warranty, express or implied, for the use or misuse of its products, which are provided with the understanding that you, the user, will determine fitness for a particular application. Systronix assumes no responsibility for any errors which may appear in this manual. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of Systronix, Inc.

Systronix reserves the right to revise this documentation and the software and hardware described herein or make any changes to the specifications of the product described herein at any time without obligation to notify any person of such revision or change.

TRADEMARKS

Systronix is a registered trademark of Systronix Inc, JStik, JStamp, JSimm, TILT, STEP, STEP+, STEP.IR, SaJe and SBX2 are trademarks of Systronix Inc, INT_EL and Intel are registered trademarks of Intel Corporation. Microsoft, Windows, and MS-DOS are registered trademarks of Microsoft Corporation. TINI is a trademark of Dallas Semiconductor.

Systronix®, Inc.
555 South 300 East
Salt Lake City, UT 84111
TEL: 801-534-1017
FAX: 801-534-1019
www.systronix.com
jrealtime family: www.jrealtime.com
jstik products: www.jstik.com
email: support@systronix.com

Revised - July 8, 2003

Revisions

2003 July 08 Minor changes to HSIO timing diagram references

Table of Contents

JStik Hardware

Description

JStik is a SIMM30 (Single In-line Memory Module) format board. The Simm30 edge connector conforms to the Systronix JSimm specification and is also compatible with many SimmStick products. JStik is based upon a high speed native execution Java controller, the aJile Sytems aJ-100. JStik is complete - it includes memory (SRAM and flash), a switching power converter, serial I/O, 10BaseT ethernet (including the RJ45 jack), a high speed I/O expansion bus, SPI, and multiple timers, counters, PWM, interrupt inputs, etc..

Purpose and Possible Uses

JStik is the most compact, self contained, native execution Java network module currently available. All you need to use it is an unregulated DC power supply of 9-14 volts. Connect up your desired I/O and it is ready to go.

Streaming Audio and Video on the HSIO bus

The High Speed I/O Interface (HSIO) has byte-wide data, twelve address bits, read and write strobes, two chip selects, 3.3 VDC power and ground. More chip selects can be easily decoded. The bus timing is variable, using a clever scheme developed by Mike Masters, to support most any speed of peripheral, with *no overhead* used in switching speeds.

The JSimm/SimmStick Standard

JStik is a SIMM30 (Single In-line Memory Module 30 contact) form factor native execution realtime Java network module. See www.simmstick.com for simmstick information and www.jsimm.com for JSimm information. SimmStick is a pretty “loose” specification so there are several (slightly incompatible) SimmStick variations in use. To avoid confusion (or maybe create more...) we have standardized and documented the most common version of SimmStick as the Systronix “JSimm” bus.

Before plugging any SimmStick module into a JSimm system check its signal description to be sure it is compatible with JSimm. A list (by no means complete) of known compatible SimmStick I/O boards is maintained at www.jsimm.com.

JStik Addressing and I/O MAP

JStik Memory Interface

JStik uses the aJ100 native execution Java controller which can directly access up to 256 Mbytes of external memory. Eight chip selects are available on the aJ100, but JStik does not use them directly. Instead we use a CPLD to generate memory and I/O control strobes at the same address range(s) of the chip selects.

The JStik memory interface is isolated from the JSimm bus and the HSIO bus. Therefore, loading, or even shorts to power or gorund on the HSIO bus and JSimm interface should not prevent JStik from booting up and running applications.

JemBuilder settings and configurations

We provide JemBuilder configurations with default settings for the various JStik versions. You must select the correct JStik configuration in order for the memory timing to be correct. Do this in the JemBuilder Project->Properties->Target Configuration

Flash Execution

JStik executes in place from Flash memory, so all the SRAM is available for heap or other use. Flash retains its contents through power cycles. You can't set debugger breakpoints when running from Flash, however.

JStik Versions - SRAM and Flash

JStik is initally available in the LP2/4 version only. Very soon after we expect to deliver version HS1/4 and LP2/8.

JStik Versions - Memory Size and Type

Version	SRAM	FLASH
JStik.LP2/4	2 Mbytes 55 nsec low power	4 Mbytes 90 nsec low power
JStik.HS1/4	1 Mbyte 10 nsec high speed	4 Mbytes 90 nsec low power
JStik.LP2/8	2 Mbytes 55 nsec low power	8 Mbytes 90 nsec low power

Future versions might include HS4/8 - 4 Mbytes of 12 nsec SRAM for a large, high speed heap, with 8 Mbytes of Flash. This large, high speed heap is needed in some special applications such as streaming realtime video. At the moment, these memory parts are sole-sourced, have a unique pinout (requiring a custom board layout) and very expensive ($55 per Mbyte – yes, that is $220 our cost for 4 Mbytes) so we have not committed to produce this version HS4/8.

Address and Data Map

The following table describes the addresses used by JStik and the data mapping at those addresses. When studying these tables, it may be helpful to recall that 0x40_0000 = 4,194,304.

JStik Memory Map
Swap memory jumper not installed
(i.e. boot up from Flash memory)

Addresses	Device
0x0000_0000-0x00FF_FFFF	Flash (90 nsec low power)
0x0000_0000-0x003F_FFFF	4 Mbytes using 16 Mbit flash chips
0x0000_0000-0x007F_FFFF	8 Mbytes using 32 Mbit flash chips (option)
0x0000_0000-0x00FF_FFFF	16 Mbytes (possible future version)
0x0180_0000-0x01BF_FFFF	SRAM
0x0180_0000-0x01BF_FFFF	2 Mbytes low power 55 nsec using 16 Mbit chips
	OR, alternatively:
0x0000_0000-0x001F_FFFF	1 Mbyte 10 nsec high speed SRAM
0x0100_0000-0x013F_FFFF	Ethernet controller
0x0100_0000-0x0130_FFFF	memory space
0x0131_0000-0x0131_FFFF	I/O space
0x0140_0000-0x017F_FFFF	High Speed I/O Bus (see HSIO section)
0x0140_0000-0x015F_FFFF	HSIO Chip select 0
0x0160_0000-0x017F_FFFF	HSIO Chip select 1

JStik Memory Map
Swap memory jumper installed
(i.e. boot up from SRAM)

Addresses	Device
0x0000_0000-0x003F_FFFF	SRAM
0x0000_0000-0x003F_FFFF	2 Mbytes low power 55 nsec using 16 Mbit chips
	OR, alternatively:
0x0000_0000-0x001F_FFFF	1 Mbyte 10 nsec high speed SRAM
0x0040_0000-0x00FF_FFFF	Flash (90 nsec low power)
0x0040_0000-0x007F_FFFF	4 Mbytes using 16 Mbit flash chips
0x0040_0000-0x00BF_FFFF	8 Mbytes using 32 Mbit flash chips (option)
0x0040_0000-0x00FF_FFFF and 0x0180_0000-0x01BF_FFFF	16 MBytes (possible future version)
0x0100_0000-0x013F_FFFF	Ethernet controller
0x0100_0000-0x0130_FFFF	memory space
0x0131_0000-0x0131_FFFF	I/O space
0x0140_0000-0x017F_FFFF	High Speed I/O Bus (see HSIO section)
0x0140_0000-0x015F_FFFF	HSIO Chip select 0
0x0160_0000-0x017F_FFFF	HSIO Chip select 1

HSIO Bus

Purpose

We intend the HSIO bus to be a high speed, asynchronous, memory-mapped, byte-wide interface for common I/O devices such as UARTs, CAN controllers, ADC and DAC, digital image data, etc.

Signals

The HSIO Bus has 12 address bits, 8 data bits, two chip selects (both asserted low), and read and write strobes (both asserted low). There are several grounds and two 3.3 VDC pins. There are no handshake or interrupt signals on the HSIO bus itself.. However all the JSimm signals are also available to any card using the HSIO bus - so any combination of these can be used with the HSIO bus for as elaborate an interface as you desire.

All HSIO signals are buffered from the aJ100 busses, are powered by 3.3V logic, are TTL level compatible, and are 5 volt tolerant. This means they will interface directly to 3.3V TTL and CMOS or 5V TTL devices. They are NOT compatible with 5V CMOS (i.e. non-TTL) levels. DO NOT use pullups on HSIO signals to attempt to use them with 5V CMOS devices.

Header

The HSIO Bus uses a 15x2 2mm header, DigiKey part #WM18099 or Molex 87333-3020. It mates with an IDC receptacle DigiKey AKA30K or AMP 111623-7. Note that all 2mm components are not standardized and compatible (as 100-mil .025 square parts are). If you obtain your own mating 2mm parts be sure they are compatible with the ones we use. Don’t use force to try to make incompatible 2mm parts mate.

Bus Timing is Variable

Variable-timing I/O buses typically control timing by adjusting some control register. This means that every bus access (or group of contiguous accesses) requires a write to the control register. This overhead can be significant. This is not how JStik implements variable bus timing.

The HSIO bus uses a clever scheme of zero overhead variable timing developed by Mike Masters. The timing of the bus is controlled by the value of upper address bits within the HSIO bus address space. You can have slow and fast peripherals on the bus and communicate with each at a different speed by simply setting the base address for each to the desired value.

HSIO Timing uses aJ-100 CLKO

The HSIO bus timing uses the aJ-100 CLKO signal, which is enabled and configured in JemBuilder. CLKCO *must* be enabled in order for the HSIO bus to function. If you plan to use HSIO and also plan to use CLKO in your own external logic then you will need to consider its effect on HSIO timing and select configurations which meet your requirements.

You can also use a buffered version of CLKO as a JSimm bus signal, enabled by JSimm JP1. If you are not using JStik HSIO then you can disable CLKO or use it for your own purposes.

HSIO Timing Equations

The HSIO timing circuitry state machine uses the aJ-100 CLKO as its input clock. The HSIO timing granularity is therefore a multiple of the CLKO period. CLKO CLKO can be the aJ-100 internal PCLK divided by two, four, or eight. At its maximum rated frequency of 103 MHz, JStik CLKO options are therefore 51.6, 25.8, or 12.9 MHz., giving CLKO periods of 19.4, 38.8, or 77.5 nsec. The table below summarizes some common values of TCKO.

HSIO Timing Granularity based on aJ-100 CLKO Period

PCLK MHz	TCKO=PCLK/2 period, nsec	TCKO=PCLK/4 period, nsec	TCKO=PCLK/8 period, nsec	Comments
103.2	19.4	38.8	77.5	JStik max PLL speed, PLL mul = 14, div = 1
73.7	27.1	54.3	108	PLL mul =10, div =1
7.37	68	136	271	JStik min PLL speed, PLL mul = 4, div = 4

Note that at the fastest settings, HSIO is capable of operating faster than most common (even high spe peripherals. Do the calculations for your specific hardware before selecting PLL and HSIO timing values HSIO is fast enough to access high speed FIFOs or memory queues on the HSIO bus at their maximum rated speeds.

HSIO Addressing

The HSIO bus uses 12-bits of address, so that's 0-FFF. Call this the "index" within the HSIO bus. HSIO has a byte-wide data path. There are two HSIO chip selects, each one has its own 12-bit address space (4096 locations). You can decode more chip selects within a given HSIO address space. For example, you could decode sixteen eight-address-bit chip selects (256 locations each) within one HSIO chip select by decoding the upper four bits of HSIO address. Typical I/O devices such as UARTs require three to four address bits (eight to sixteen locations), while a CAN controller such as the SJA1000 could require eight address bits (256 locations). ADCs and DACs might need one to four address bits.

JStik is more efficient at word accesses to memory mapped I/O space. Smaller width accesses are implemented by the aJ100 as read-modify writes (these are at least 2X slower than just a write). In other words if you explicitly perform a byte write, the aJ100 first reads the whole word, changes the byte you are writing, then writes out the whole word again. The hardware doesn't know that in the case of HSIO the upper twenty four bits of data aren't connected to anything. So for the best performance we use word access and only the low byte of the word is actually used. The upper bits aren't used and are ignored by the JStik HSIO hardware.

The low two bits of the aJ100 address don't go to the HSIO bus (remember, we're writing on word boundaries, not byte boundaries). Therefore HSIO A0 = aJ100 A2.

This has the effect of shifting the aJ100 address left by 2, so our HSIO addresses are really 0x0, 0x04, 0x08, 0x0C, 0x10, 0x14, etc. This means our HSIO "index" occupies 0-(FFF x 4) = aJ100 addresses 0 thru 3FFC. So A15, A14 are not used, “don't cares” in the current JStik version. So we end up with our HSIO shifted index space appearing FFFF divided by 3FFC = 4 times in aJ100 address space A15..A0.

The HSIO timing logic uses aJ100 address lines A19..A16 as the wait state timing values, so we can't use them as anything other than timing information.

The 0-3FFC address space also will repeat 0x80 times within the address space per HSIO CS which is 0140_0000 - 015F_FFFF for CS0 and 160_0000 - 17F_FFFF for CS1. This is a space of 20_0000 per chip select. A20 is currently used in the wait state equations so this makes 0x140 one Tas and 0x150 the slower one.

It looks like the min CS width with the current JStik configs and 103 MHz clock is too fast to write to 55 nsec SRAM reliably. Wow. This thing is quick! With A1916=1 it's slowed enough to work, A1916=2 is better. CS is about 250 nsec and WE about 100 nsec.

HSIO Bus Timing
note that 0 =not asserted and 1= asserted, regardless of voltage level

CS[0 or 1](L)	A20(H)	A19..A16 (H)	RD(L)	WR(L)	description
1	0	0000	1	0	read at fastest possible speed, no stretch
1	0	0000	0	1	fastest possible write
1	0	0010	X	X	good stretch for accessing 55 nsec SRAM on the HSIO bus.
0	X	XXXX	X	X	chip select not asserted, so no HSIO access

JStik JSimm Interface

Many of the aJ100 I/O bits are tied together and then connected to JSimm I/O pins.

JSimm I/O Bus				
JSimm Pin#	SimmStick description	aJ100 pin	Description	??
1	A1	IOC0/SCS0	GPIOC0 or SPI chip select 0	
2				
3				
4	PWR	N/C	not used on JStik	
1			ROW2	ROW1

■ *JStik HSIO Timing Diagrams*

Complete timing diagrams are available online, along with an interactive timing calculator. Please use the “Technical Data” pulldown list in the www.jstik.com web area.

JStik Java API

See the documentation and JavaDocs online at http://www.jstik.com

Java Packages for JStik

aJile CLDC Runtime

This consists of the aJile\Runtime_cldc\Rts folder and the D:\aJile\Runtime_cldc\classes.jar which provide both the CLDC base classes and aJile-specific support (such as timers, counters, Flash memory, etc). These classes include javaxcomm support for the JStik UARTs, so you don't need to import that. We don't know why Rts is a folder and classes.jar is a JAR file, but if you wish, you can JAR up the Rts folder too - that's how we have done it.

com.systronix.jstik

This is a collection of utility classes for JStik, scheduled for release July 2003