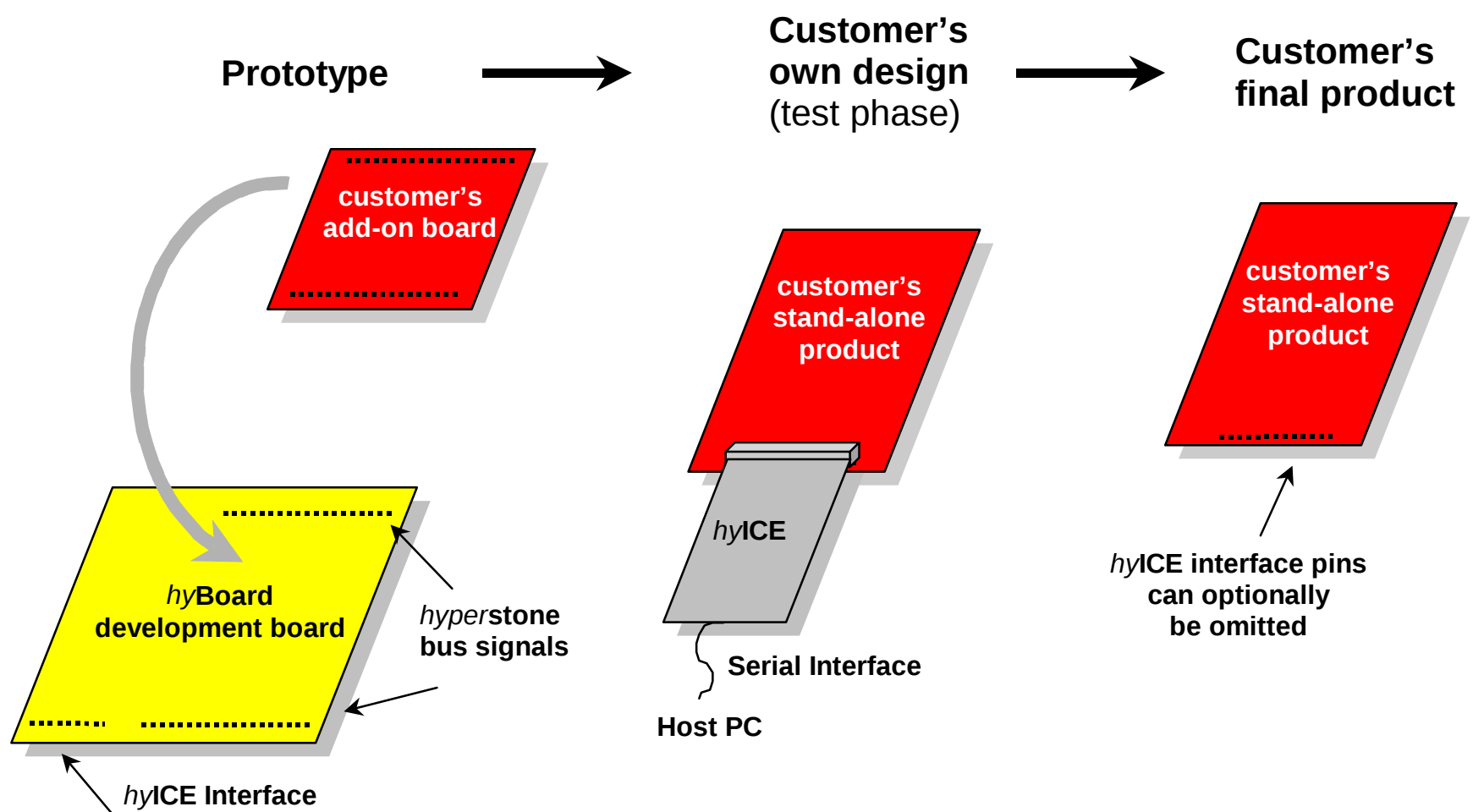


Hyperstone hyBoard

PC-based
Development Board

Installation guide and Function description



hyperstone

Specifications and information in this document are subject to change without notice and do not represent a commitment on the part of **Hyperstone** AG. **Hyperstone** AG reserves the right to make changes to improve functioning. Although the information in this document has been carefully reviewed, **Hyperstone** AG does not assume any liability arising out of the use of the product or circuit described herein.

Hyperstone AG does not authorize the use of the **Hyperstone** microprocessor and tools in life support applications wherein a failure or malfunction of the microprocessor may directly threaten life or cause injury. The user of the **Hyperstone** microprocessor in life support applications assumes all risks of such use and indemnifies **Hyperstone** AG against all damages.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photo-copying and recording, for any purpose without the permission of **Hyperstone** AG.

Hyperstone is a registered trademark of **Hyperstone** AG.

MS-DOS is a registered trademark of Microsoft Corp.

PENTIUM is a registered trademark of Intel Corp.

For further information please contact:



Hyperstone AG

Line-Eid-Str. 3

D-78467 Konstanz

Germany

Phone +49 7531 / 9803-0

Fax +49 7531 / 51725

E-Mail info@hyperstone.de

Hyperstone Taiwan office

7F-5, No. 75, Sec. 1, Hsin Tai Wu Road

Hsi Chih, Taipei Hsien,

Taiwan, R.O.C.

Phone +886 2 2698 2702

Fax +886 2 2698 2872

E-Mail Taiwan@hyperstone.com

URL: <http://www.Hyperstone.com>

© Copyright 1990, 2002 **Hyperstone** AG

Rev. 07/2002

Table of Contents

1. Introduction	1
2. Block Diagram.....	2
3. Technical Data	3
4. Board Layout.....	4
5. Board Installation	5
5.1. Requirements	5
5.2. Board use (with hyICE)	6
5.2.1. Trouble Shooting	9
6. Board Configuration	10
6.1. Jumper numbering convention:	10
6.2. Jumper J3 (Ethernet chip select mapping)	10
6.3. Jumper J4, J13 (EEPROM address select).....	10
6.3.1. 128kB EEPROM.....	10
6.3.2. 512kB EEPROM.....	10
6.4. Jumper J5 (USB chip select mapping)	11
6.5. Jumper J6 (IO1 / IO2 active indicator)	11
6.6. Jumper J7 (Ethernet interrupt mapping).....	11
6.7. Jumper J8 (USB interrupt mapping)	11
6.8. Jumper J9 (USB suspend select)	11
6.9. Jumper J10 (ICE interrupt mapping).....	11
6.10. Jumper J11 (PLD interrupt mapping)	12
6.11. Jumper J12 (ICE chip select mapping)	12
6.12. Jumper R6 (chip 0 Ω resistor).....	12
6.12.1. DRAM Type Selection.....	12
7. Memory Address Space	13
8. Connectors X1...X4	15
8.1. Expansion Connector X1	16
8.2. Expansion Connector X2	18
8.3. Expansion Connector X3	20
8.4. Expansion Connector X4	21
8.5. Connector ST 14 (JTEG connector for XILINX)	22
8.6. HyICE Connector ST1	23
8.7. Power Supply Connector ST16.....	24
9. The XC9572.....	25
9.1. Overview.....	25

9.2. Interrupt Mask Register	26
9.3. Interrupt Polarity Register	26
9.4. Interrupt Identification Register.....	27
9.5. USB Suspend Register.....	27
9.6. Avoid interrupt conflicts on HY_INT4	27
9.7. Example.....	28
10. Board Dimensions	31
11. Appendix.....	32
11.1. hyBoard schematics.....	32
11.2. Board Layout (zoom)	36
11.3. XC9572 Equations	37

1. Introduction

The *hyBoard* development board is a board designed around the *Hyperstone* E1-32XS 32-Bit microprocessor. With this board you can develop and test your software parallel to developing your own *Hyperstone* system. In combination with the *Hyperstone* [software development tools](#) it provides a versatile software development environment.

The *hyBoard* development board operates as a base. In addition to the ICE connector, is equipped with 10 Mbit 10-base-T Ethernet, as well as a USB1.1 interface, for use by applications.

For debugging, an interface card ([hyICE](#)) must be plugged into the development board in order to communicate with the host system. [hyICE](#) provides a serial communication channel from the host PC to the *Hyperstone* target system with a data transfer rate of 115200 bps.

Furthermore, the development board has four expansion connectors which allow connection of additional peripheral devices, hardware testing or custom-made prototype hardware.

2. Block Diagram

Fig. 1 shows a simplified block diagram of the development board:

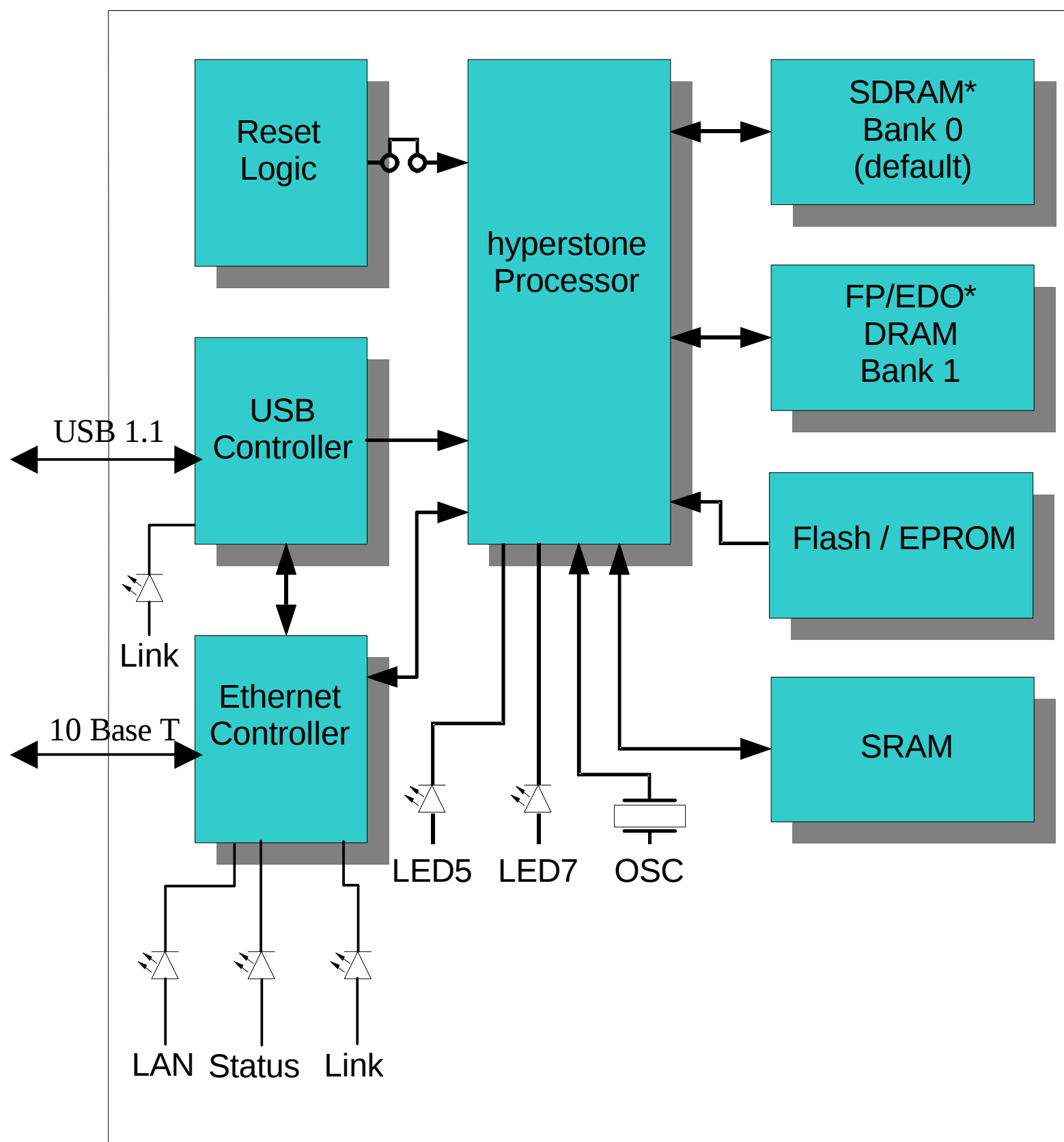


Fig. 1: Block Diagram

* either one of the DRAM banks can be mounted. Please refer to [6.12.1](#) for details.

3. Technical Data

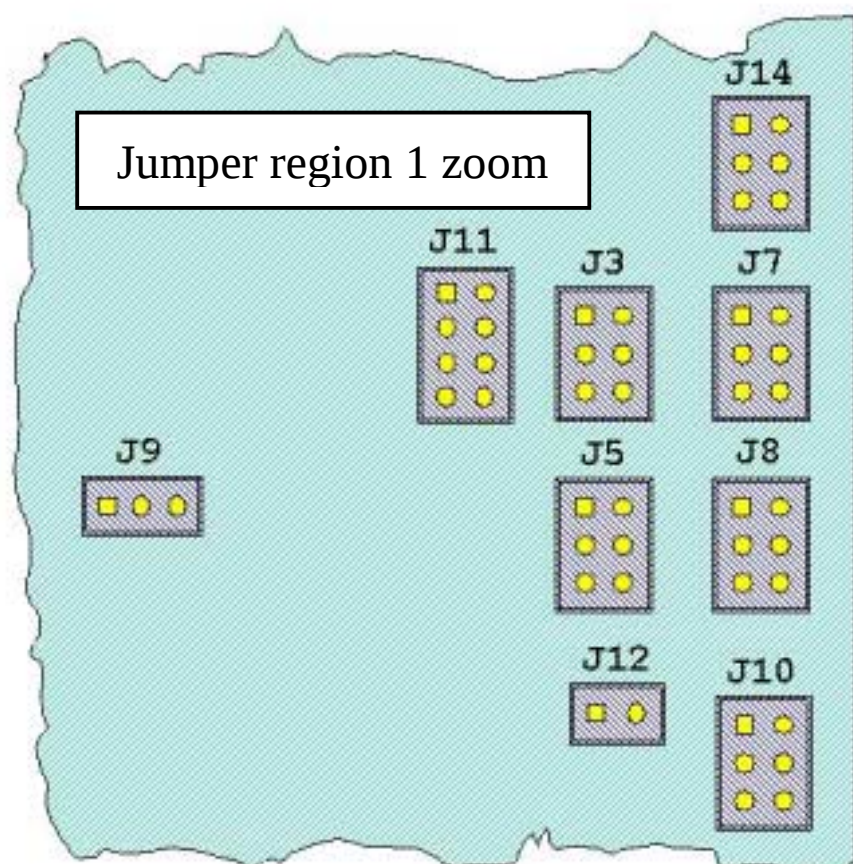
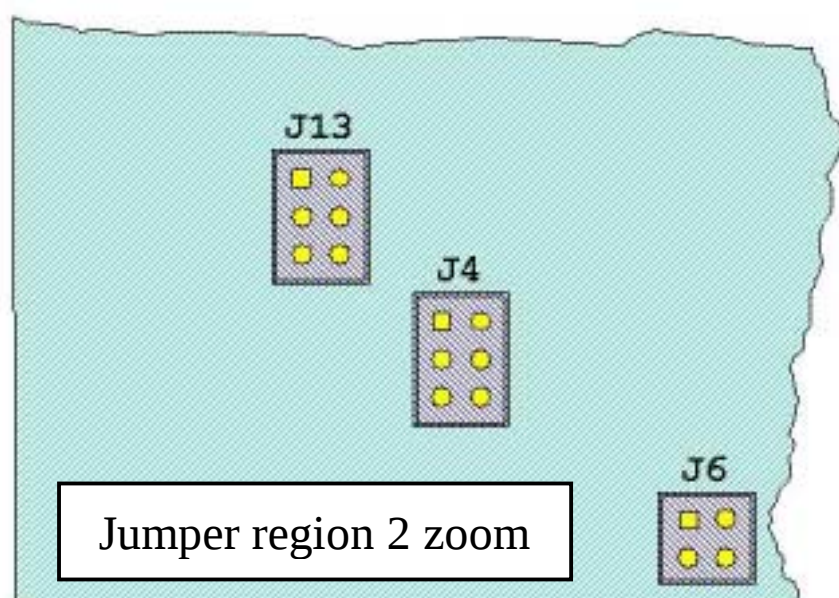
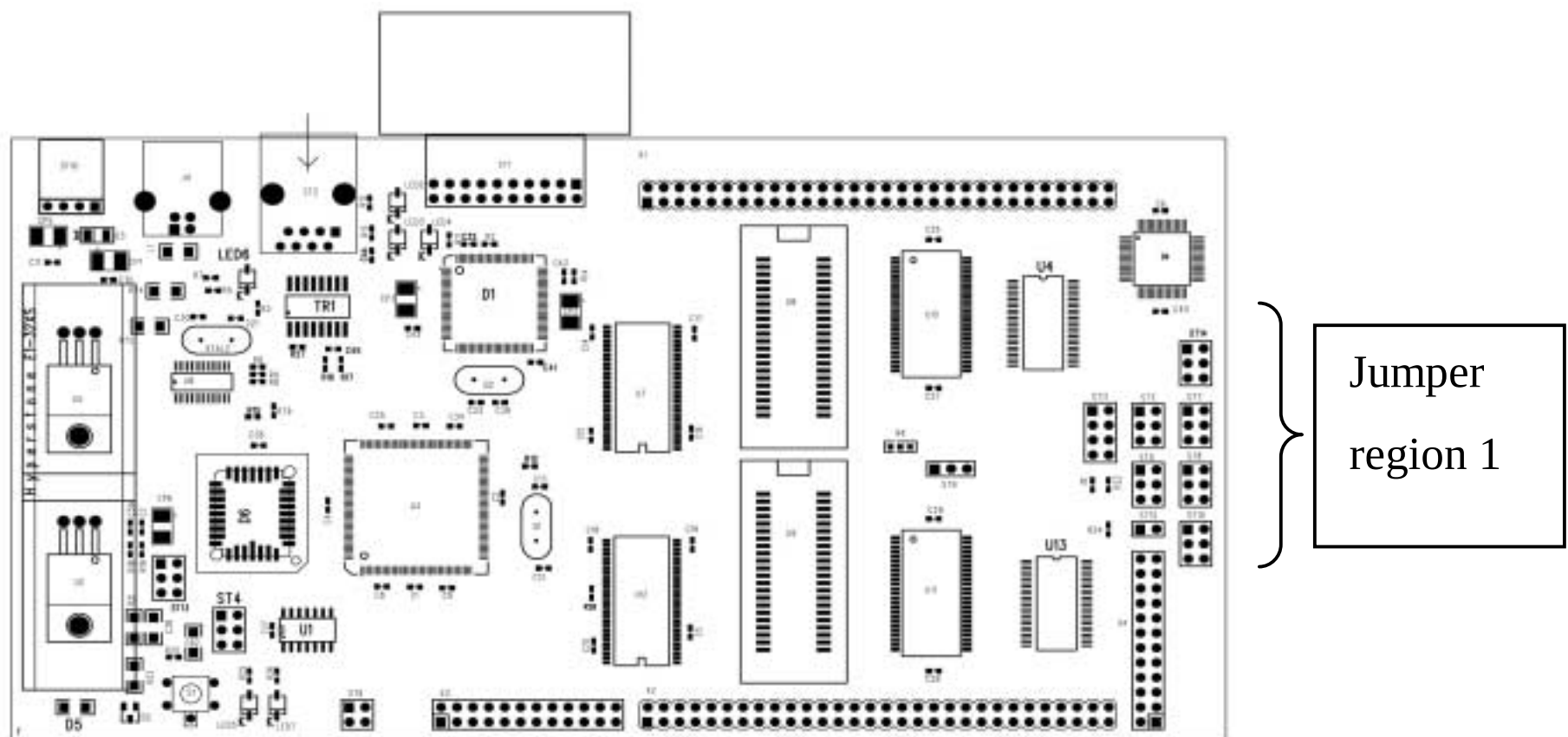
- *Hyperstone* E1-32XS 32-bit RISC/DSP microprocessor
- Either 16 MB SDRAM or 4 MB PS/2 (EDO or FP)
- up to 512 kByte Flash Memory or EPROM, standard 128 kByte Flash Memory
- 256 kB fast SRAM
- up to 11 interrupts (IRQ) for communication with the host PC, selectable by software - 4 (3) provided by CPU, 8 by interrupt controller¹, using either one of the 4 CPU interrupts to forward the user-interrupts. Optionally, the IO1..IO3 can be used as interrupt inputs as well.
- expansion connectors provide all signals for additional hardware extensions
- onboard USB interface v1.1
- onboard LAN 10 Mbit RJ 45 interface
- programmable interrupt controller, basically used for LAN and USB interrupts

¹ Equations and Register description of the XC9572 can be found in section 9

4. Board Layout

Fig. 2 shows the layout of the *hyBoard* development board (see also [schematic](#) of the *hyBoard*):

Fig. 2: Board Layout (top view)



5. Board Installation

5.1. Requirements

This section covers the installation of the *hyBoard* development board. You need the following items to use *hyBoard*:

- o Personal Computer (386, 486 or PENTIUM) running MS-DOS, MS-Windows 3.x or Windows 95², 98 or ME equipped with:
 - at least 4 MByte of extended memory
 - VGA graphics
 - CD-ROM drive for installation of the software
- o *Hyperstone* software development tools.

For installing the software please refer to the installation section of the file **readme.txt** contained on the software distribution CD-ROM. Please do not forget to include the directory where you installed the *Hyperstone* software in the search path of your **autoexec.bat** file.
- o *hyICE* adapter card
- o serial connection cable (included with the *hyICE* adapter card)
- o power cable (included with the *hyICE* adapter card)
- o +5.0V, 800 mA DC power supply with voltage stabilizer, voltage regulators on board.

² The *hyperstone* debugger *hyDebug* doesn't support Windows NT

5.2. Board use (with hyICE)

For installation, proceed with the following steps:

- Switch off your PC host computer.
- Connect the serial connection cable to the SUB-D connector of the [hyICE](#) and to the COM1 or COM2 serial port of the PC. Please use only the serial connection cable supplied with the [hyICE](#).
- Connect the [hyICE](#) to the [HyICE Connector](#) on the **hyBoard** development board or to a *Hyperstone* board developed by yourself.
- The sample Flash/EPROM binary file **rtk32xs.hex** delivered on the distribution disk of the *Hyperstone* software development tools can be used to program a sample Flash/EPROM. Note that the real-time operating system [hyRTK](#) automatically detects whether the [hyICE](#) is present or not. If it is present the system waits for debugging commands from the source-level debugger [hyDebug](#). Otherwise the user program in the Flash/EPROM is downloaded and started. For generating new Flash/EPROM binaries containing your own application program please refer to the *Hyperstone* Software Development Tools manual.
- Connect the development board with the power cable to a +5.0V DC power supply. Please note that the red wire must be connected to Vcc, the black (or blue) wire to GND. You may use the built-in power supply of the PC if you don't have an external power supply.
- Switch on the host PC.
- Switch on the power supply for the **hyBoard** development board.

- Now specify the serial port where you have connected the [hyICE](#) in the debugger configuration file [hydebug.cfg](#) located in the subdirectory **\hstone\bin**. The configuration file can be edited with any ASCII editor.

The debugger configuration file should contain the following entries:

```
[DRIVER]
Driver=SERIAL    ; use this setting for serial
                  ; communication

[SERIAL]
Com=COM1         ; defines the selected COM port
                  ; either COM1 or COM2

Boarddef="c:\hstone\eprom\boarddef.rom"
Boardtype=5
```

The parameter `Boarddef` specifies the directory and filename of the board definition file. The denoted file is read by [hyDebug](#) in order to insert the actual values for the [Bus Control Register BCR](#), [Memory Control Register MCR](#) and [Timer Prescaler Register TPR](#) into the [hyRTK](#) operating system. [hyAdmin](#) allows to generate and edit the board definition file.

The parameter `Boardtype` is used by [hyDebug](#) as an index to search in the board definition file for the board configuration.

- To check the installation, change the default directory to the directory where you installed the *Hyperstone* macro-assembler or C sample files.

Examples:

```
cd ...\hstone\samples\asm      for using the macro-assembler
                                sample program
cd ...\hstone\samples\c        for using the C sample program
```

Start the batch file `sample01.bat` or `sieve.bat` in order to generate the executable file of the sample program. Invoke the source-level debugger and specify the assembler sample program `sample01` or the C sample program `sieve` for debugging.

Examples:

```
hydebug sample01               for using the macro-assembler
                                sample program
hydebug sieve                   for using the C sample program
```

You should now see the debugger displaying a dialog box "Waiting for Reset". Now press the blue RESET button on the *hyBoard* development board. The debugger now loads and displays the source code of the sample program (either `sample01` or `sieve`) in the source window. If you have problems, please refer to section [5.2.1.Trouble Shooting](#).

- You are now ready for loading and debugging *Hyperstone* programs written in assembler or C.

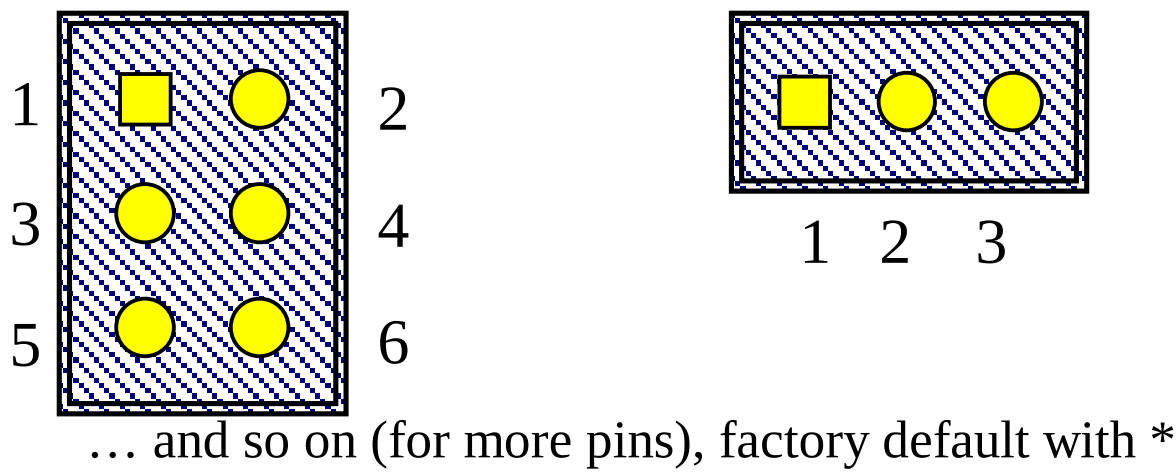
5.2.1. Trouble Shooting

In case you get a message box displaying an error message indicating that **hyDebug** could not establish a connection with your target system, you should check the following:

- Is your **hyICE** properly connected to the host PC and the development board?
- Have the serial connection or power cables become loose?
- Has the debugger configuration file **hydebug.cfg** been set properly? Have you specified the correct serial port (COM1 or COM2)? Make sure that no mouse driver is installed on this serial port.
- Does the board have proper power supply?
- Switch off your host PC, press the RESET button on the development board, switch on the PC, and then try invoking the debugger again.
- Check your **autoexec.bat** and **config.sys** file for memory resident hardware drivers, disk caches or memory managers. Remove them, reboot the host PC and try again. **hyDebug** has been designed to work well with most utility programs, however there may be some resident programs which are incompatible with **hyDebug**.
- Are there multiple debugger configuration files on your hard disk? The debugger first tries to load the configuration file from the current directory, then from the directory **\hstone\bin**.

6. Board Configuration

6.1. Jumper numbering convention:



6.2. Jumper J3 (Ethernet chip select mapping)

1,2*	Mapped from PLD IO9-F2
3,4	Mapped from HY_ADDR24
5,6	Mapped from HY_ADDR12

6.3. Jumper J4, J13 (EEPROM address select)

6.3.1. 128kB EEPROM

J4	1,2*	default, no other configuration valid
J13	1,2	top 64kB mapped to MEM 3
J13	3,4*	128kB mapped to MEM 3
J13	5,6	bottom 64kB mapped to MEM 3

6.3.2. 512kB EEPROM

J13	3,4	default, no other configuration valid
J4	1,2	top 256kB mapped to MEM 3
J4	3,4	512kB mapped to MEM 3
J4	5,6	bottom 256kB mapped to MEM 3

6.4. Jumper J5 (USB chip select mapping)

1,2*	Mapped from PLD IO9-F1
3,4	Mapped from HY_ADDR25
5,6	Mapped from HY_ADDR12

6.5. Jumper J6 (IO1 / IO2 active indicator)

1,2	LED7 (red) driven by HY_IO1 – remove if too much load.
3,4	LED5 (green) driven by HY_IO2 – remove if too much load.

6.6. Jumper J7 (Ethernet interrupt mapping)

1,2	Mapped to SYS_INT (HY_INT4)
3,4	Mapped to HY_INT1
5,6*	Mapped to PLD (interrupt controller)F2-IO10

6.7. Jumper J8 (USB interrupt mapping)

1,2	Mapped to SYS_INT (HY_INT4)
3,4	Mapped to HY_INT2
5,6*	Mapped to PLD (interrupt controller)F1-IO10

6.8. Jumper J9 (USB suspend select)

1,2*	USB suspend controlled by PLD IO1-F1
2,3	USB suspend controlled by HY_IO1

6.9. Jumper J10 (ICE interrupt mapping)

1,2	Mapped to SYS_INT (HY_INT4)
3,4	Not connected
5,6*	Mapped to PLD (interrupt controller)F1- IO2

6.10. Jumper J11 (PLD interrupt mapping)

1,2	mapped to HY_INT1
3,4	mapped to HY_INT2
5,6	mapped to HY_INT3
7,8*	mapped to SYS_INT (HY_INT4)

6.11. Jumper J12 (ICE chip select mapping)

1,2	Close*:Mapped from HY_ADDR12 – open: pulldown GND
-----	---

6.12. Jumper R6 (chip 0Ω resistor)

6.12.1. DRAM Type Selection

Soldered jumper R6 selects the type of DRAM (Fast Page Mode or EDO) connected to the *Hyperstone* processor. By factory default this is unmounted, since there is SDRAM on board. Optionally there might be boards with “standard” DRAM memory instead of SDRAM, and then the jumper will exist, already with correct factory settings.

Fast page: 1-2

EDO: 2-3

7. Memory Address Space

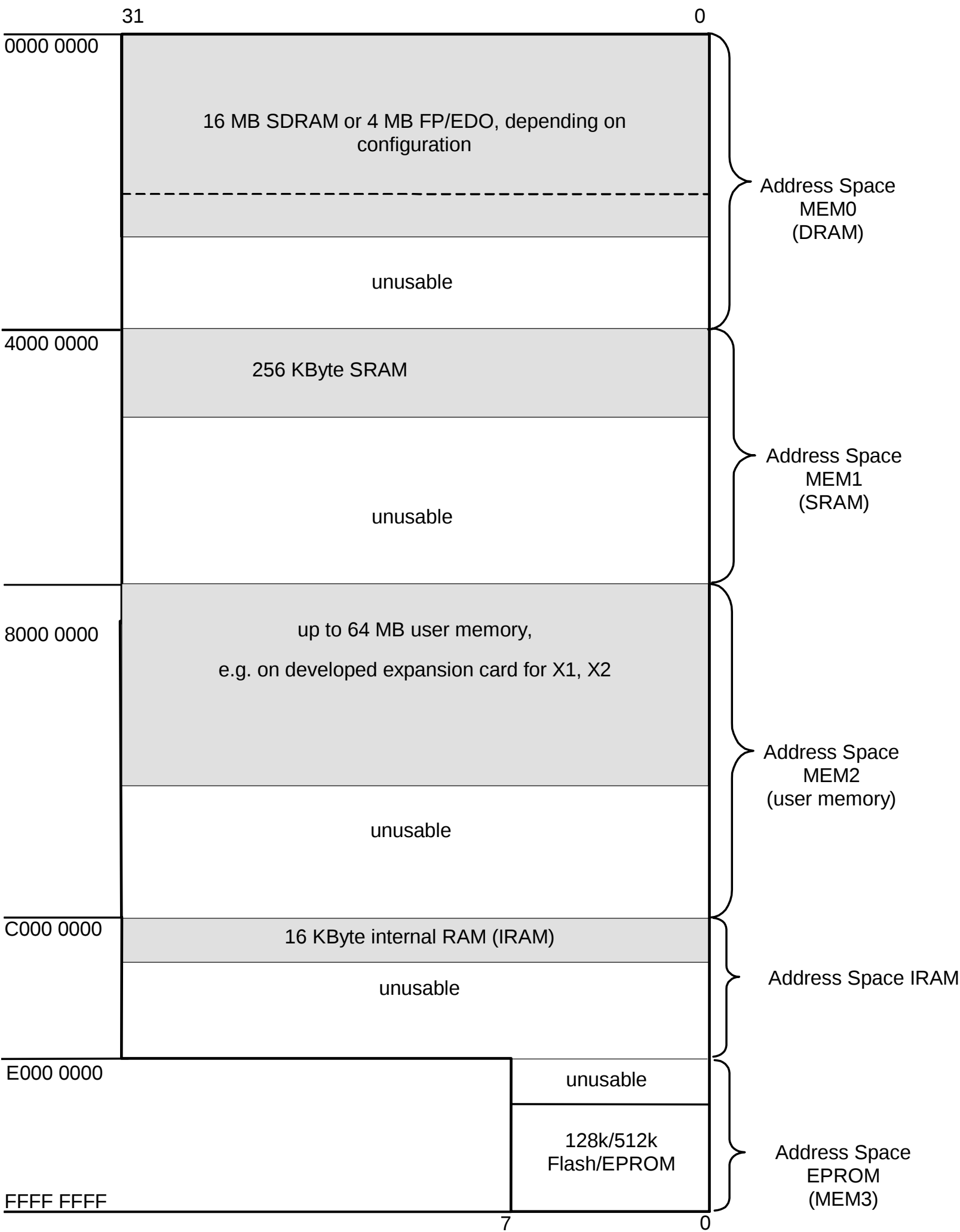
Table 1 shows the memory address map for all possible configurations.

Configuration	Mem Area	Start Address (hex)	End Address (hex)
16 MB SDRAM	MEM0	0000 0000	00FF FFFF
256 kB SRAM	MEM1	4000 0000	4003 FFFF
Free for customer use, max. 64 MB ³	MEM2	8000 0000	BFFF FFFF (83FF FFFF)
16 kB Internal RAM	IRAM	C000 0000	C000 3FFF
128/512 kB Flash/EPROM	MEM3	E000 0000	FFFF FFFF

Table 1: Memory Address Space

³ Physical MEM may be greater but each 64 Meg. will be modulo to the first 64 Meg of this memory (26 Address lines...)

Fig. 3 is a graphical representation of the memory address map of the development board.



g. 3: Memory Address Map

8. Connectors X1...X4

A further expansion card with additional components such as parallel interfaces, AD-converters, EEPROMs etc. can easily be added to the **hyBoard** development board. Three expansion ports (X1...X3) in total with data, address and control signals are provided for this purpose.

X4 mainly provides the connection to the programmable interrupt controller. The user can map up to 6 interrupts to either HY_INT1 .. HY_INT4. As well, it provides 8 chip select signals for customers own products and some more interrupts and chip selects that are normally used by USB, LAN and hyICE.

For stand-alone operating mode, the **hyICE** connector ST1 and the power supply connector ST16 are provided.

The UART is connected to the I/O address space of the **Hyperstone** microprocessor. The address signal A12 acts as chip select, the 8 registers of the UART are selected by address signals A15...A13.

Please contact **Hyperstone** AG for additional information when you are planning to develop an expansion.

8.1. Expansion Connector X1

Table 2 shows the pin out and the corresponding signals for the female 60-pin expansion connector X1.

The signal states are defined as I = input, O = output and Z = three-state (inactive).

Pin-No.	Signal Name	States	Description
1	A18	O/Z	Address Bus
2	VCC		System Power, +3,3 V
3	A17	O/Z	Address Bus
4	GND		Ground
5	A16	O/Z	Address Bus
6	GND		Ground
7	A15	O/Z	Address Bus
8	GND		Ground
9	A14	O/Z	Address Bus
10	GND		Ground
11	A13	O/Z	Address Bus
12	VCC		System Power, +3,3 V
13	A12	O/Z	Address Bus
14	GND		Ground
15	A11	O/Z	Address Bus
16	GND		Ground
17	A10	O/Z	Address Bus
18	GND		Ground
19	A09	O/Z	Address Bus
20	VCC		System Power, +3,3 V
21	A08	O/Z	Address Bus
22	GND		Ground
23	A07	O/Z	Address Bus
24	GND		Ground
25	A06	O/Z	Address Bus
26	GND		Ground
27	A05	O/Z	Address Bus
28	GND		Ground
29	A04	O/Z	Address Bus
30	GND		Ground

8.1. Expansion Connector X1 (continued)

Pin-No.	Signal Name	States	Description
31	A03	O/Z	Address Bus
32	VCC		System Power, +3,3 V
33	A02	O/Z	Address Bus
34	GND		Ground
35	A01	O/Z	Address Bus
36	GND		Ground
37	A00	O/Z	Address Bus
38	VCC		System Power, +3,3 V
39	D15	O/I	Data Bus
40	GND		Ground
41	D13	O/I	Data Bus
42	D14	O/I	Data Bus
43	D11	O/I	Data Bus
44	D12	O/I	Data Bus
45	D09	O/I	Data Bus
46	D10	O/I	Data Bus
47	D08	O/I	Data Bus
48	VCC		System Power, +3,3 V
49	D06	O/I	Data Bus
50	D07	O/I	Data Bus
51	D05	O/I	Data Bus
52	GND		Ground
53	D04	O/I	Data Bus
54	GND		Ground
55	D02	O/I	Data Bus
56	D03	O/I	Data Bus
57	D01	O/I	Data Bus
58	GND		Ground
59	D00	O/I	Data Bus
60	VCC		System Power, +3,3 V

Table 2: Signals at expansion connector X1

8.2. Expansion Connector X2

Table 3 shows the pin out and the corresponding signals for the 60-pin female expansion connector X2.

The signal states are defined as I = input, O = output and Z = three-state (inactive).

Pin-No.	Signal Name	States	Description
1	IO1	O/I	Programmable Input/Output 1
2	INT1	I	Interrupt Input 1
3	IO2	O/I	Programmable Input/Output 2
4	INT2	I	Interrupt Input 2
5	IO3	O/I	Programmable Input/Output 3
6	INT3/WAIT	I	Interrupt Input 3
7	OE#	O/Z	Output Enable for SRAMs and EPROMs
8	INT4	I	Interrupt Input 4
9	CS2#	O/Z	Chip Select for MEM2
10	WE#	O/Z	Write Enable for DRAM and R/W# for I/O
11	CS3#	O/Z	Chip Select for MEM3
12	IORD#	O/Z	I/O Read Strobe, optionally I/O Data Strobe
13	CLKOUT	O	Clock Output
14	IOWR#	O/Z	I/O Write Strobe
15	RESET#	O	Reset Output
16	A19	O/Z	Address Bus
17	ACT	O	Active as Bus Master
18	A20	O/Z	Address Bus
19	RQST	O	Bus Request Output
20	A21	O/Z	Address Bus
21	GRANT#	I	Bus Grant Input
22	A22	O/Z	Address Bus
23	WE0#/BE0#	O/Z	Write Enable for SRAM byte 0
24	A23	O/Z	Address Bus
25	WE1#/BE1#	O/Z	Write Enable for SRAM byte 1
26	A24	O/Z	Address Bus
27	VCC		System Power, +3,3 V
28	A25	O/Z	Address Bus
29	GND		Ground
30	D16	O/I	Data Bus

8.2. Expansion Connector X2 (continued)

Pin-No.	Signal Name	States	Description
31	VCC		System Power, +3,3 V
32	D17	O/I	Data Bus
33	GND		Ground
34	D18	O/I	Data Bus
35	VCC		System Power, +3,3 V
36	D19	O/I	Data Bus
37	GND		Ground
38	D20	O/I	Data Bus
39	GND		Ground
40	D21	O/I	Data Bus
41	VCC		System Power, +3,3 V
42	D22	O/I	Data Bus
43	GND		Ground
44	D23	O/I	Data Bus
45	WE2#/BE2#	O/Z	Write Enable for SRAM byte 2
46	D24	O/I	Data Bus
47	GND		Ground
48	D25	O/I	Data Bus
49	WE3#/BE3#	O/Z	Write Enable for SRAM byte 3
50	D26	O/I	Data Bus
51	GND		Ground
52	D27	O/I	Data Bus
53	VCC		System Power, +3,3 V
54	D28	O/I	Data Bus
55	GND		Ground
56	D29	O/I	Data Bus
57	GND		Ground
58	D30	O/I	Data Bus
59	GND		Ground
60	D31	O/I	Data Bus

Table 3: Signals at expansion connector X2

8.3. Expansion Connector X3

Table 4 shows the pin out and the corresponding signals for the 24-pin female expansion connector X3.

The signal states are defined as I = input, O = output and Z = three-state (inactive).

Pin-No.	Signal Name	States	Description
1	GND	---	signal ground
2	VEXT	O	external Voltage
3	NC	---	NC
4	HY_CS1#	O/Z	chip select 1
5	NC	---	NC
6	HY_RAS#	O/Z	row address strobe
7	NC	---	NC
8	HY_CAS0#	O/Z	column address strobe 0 for SDRAM
9	NC	---	NC
10	HY_CAS1#	O/Z	column address strobe 1 for SDRAM
11	NC	---	NC
12	HY_CAS_CAS2#	O/Z	CAS for SDRAM ; CAS2 for FP/EDO
13	RESET	O	Reset
14	HY_CLK_CAS3X#	O/Z	clock for SDRAM ; CAS3
15	NC	---	NC
16	HY_DP0	O/Z	Parity bit 0
17	NC	---	NC
18	HY_CS1_DP1	O/Z	chip select1 for SDRAM; parity bit 1 for FP/EDO
19	NC	---	NC
20	HY_CS0_DP2	O/Z	chip select0 for SDRAM; parity bit 2 for FP/EDO
21	NC	---	NC
22	HY_CKE_DP3	O/Z	clock enable for SDRAM; parity bit 3 for FP/EDO
23	GND	---	signal ground
24	Vcc	---	System power, +2,5V

Table 4: Signals at expansion connector X3

8.4. Expansion Connector X4

Table 12 shows the pin out and the corresponding signals for the 24-pin female expansion connector X4.

The signal states are defined as I = input, O = output and Z = three-state (inactive).

Pin-No.	Signal Name	States	Description
1	DATA_STR	O	data strobe
2	ICE_INT#	O	hyICE interrupt
3	NC	---	not connected
4	ICE_CS	O	hyICE chipselect
5	ETH_CS	I	Ethernet chipselect
6	USB_CS	I	USB chipselect
7	ETH_INT#	I	Ethernet interrupt
8	USB_INT#	I	USB interrupt
9	ADDR_STR	O	address strobe
10	EXTCS8#	O	free usable chipselect signal
11	GND	---	signal ground
12	EXTCS7#	O	free usable chipselect signal
13	USEINT6	I	free usable interrupt
14	EXTCS6#	O	free usable chipselect signal
15	USEINT5	I	free usable interrupt
16	EXTCS5#	O	free usable chipselect signal
17	USEINT4	I	free usable interrupt
18	EXTCS4#	O	free usable chipselect signal
19	USEINT3	I	free usable interrupt
20	EXTCS3#	O	free usable chipselect signal
21	USEINT2	I	free usable interrupt
22	EXTCS2#	O	free usable chipselect signal
23	USEINT1	I	free usable interrupt
24	EXTCS1#	O	free usable chipselect signal

Tab. 12: Signals at expansion connector X4

8.5. Connector ST 14 (JTEG connector for XILINX)

Please take care, this is not a jumper! It represents the programming pins of the XILINX XC9572XL.

Pin-No.	Signal Name	States	Description
1	Vcc	---	+3,3 V
2	TMS	I	Test Mode Select
3	TDI	I	Test Data In
4	TCK	I	Test Clock
5	TDO	O	Test Data Out
6	GND	---	signal ground

8.6. HyICE Connector ST1

Table 5 shows the pin out and the corresponding signals for the 20-pin female **hyICE** connector ST1.

The signal states are defined as I = input, O = output and Z = three-state (inactive).

Pin-No.	Signal Name	States	Description
1	D00	O/I	Data Bus
2	VCC		System Power, +3,3 V
3	D01	O/I	Data Bus
4	RESET#	O	Reset Output
5	D02	O/I	Data Bus
6	XIO1	O/I	Jumpered IO1 signal
7	D03	O/I	Data Bus
8	A12	O/Z	Address Bus
9	D04	O/I	Data Bus
10	A13	O/Z	Address Bus
11	D05	O/I	Data Bus
12	A14	O/Z	Address Bus
13	D06	O/I	Data Bus
14	A15	O/Z	Address Bus
15	D07	O/I	Data Bus
16	INT4	I	Interrupt Input 4
17	IORD#	O/Z	I/O read Strobe, optionally I/O Data Strobe
18	GND		Ground
19	IOWR#	O/Z	I/O Write Strobe
20	XIO2	O/I	Jumpered IO2 signal

Table 5: Signals at **hyICE** connector ST1

8.7. Power Supply Connector ST16

Figure 4 shows the pin out for the power supply connector ST16.

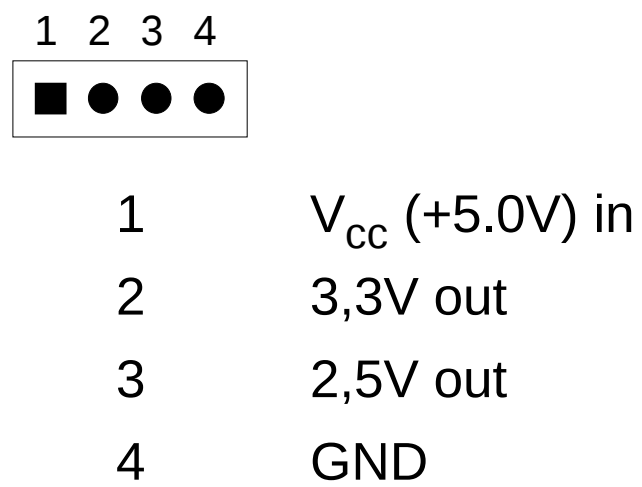


Fig. 4: Power Supply Connector ST16

9. The XC9572

9.1. Overview

The *hyBoard* development board provides eight decoded chip select outputs for I/O accesses and six interrupt input lines for use in add-on logic. Address lines 22..25 are used to decode the I/O accesses according to the following table:

A25	A24	A23	A22	R/W	Description
0	0	0	0		Not used
0	0	0	1	R/W	Interrupt Mask Register
0	0	1	0	R	Interrupt Identification Register
0	0	1	0	W	Interrupt Polarity Register
0	0	1	1	R/W	Chip select for USB Controller
0	1	0	0	R/W	Chip select for Ethernet Controller
0	1	0	1	R/W	Chip select EXTCS1_ (active low)
0	1	1	0	R/W	Chip select EXTCS2_ (active low)
0	1	1	1	R/W	Chip select EXTCS3_ (active low)
1	0	0	0	R/W	Chip select EXTCS4_ (active low)
1	0	0	1	R/W	Chip select EXTCS5_ (active low)
1	0	1	0	R/W	Chip select EXTCS6_ (active low)
1	0	1	1	R/W	Chip select EXTCS7_ (active low)
1	1	0	0	R/W	Chip select EXTCS8_ (active low)
1	1	0	1	R/W	USB Suspend Register

Table 9.1.1, InterruptEnableMask Register

For the eight I/O chip select signals, the XC9572 chip only generates the chip select signal according to address bits 22..25, the data lines are free for the peripheral device using the chip select signal. Chip select EXTCS8_ is handled specially, see the details in the XC9572 equations in the appendix.

9.2. Interrupt Mask Register

This register is for enabling/disabling the 6 user and 2 on board interrupts. A value of 1 for one bit of the Interrupt Mask Register enables the corresponding interrupt, a value of 0 disables the corresponding interrupt.

Bit	Description
0	PUSB_INT enable (USB controller)
1	PETH_INT enable (Ethernet controller)
2	USEINT1 enable
3	USEINT2 enable
4	USEINT3 enable
5	USEINT4 enable
6	USEINT5 enable
7	USEINT6 enable

Table 9.2.1, Interrupt Mask Register

9.3. Interrupt Polarity Register

This register determines the active polarity (high or low) for the 6 user and 2 on-board interrupts. A value of 1 for one bit of the Interrupt Polarity Register configures the polarity as active high, a value of 0 configures active low.

Bit	Description
0	PUSB_INT polarity (USB controller)
1	PETH_INT polarity (Ethernet controller)
2	USEINT1 polarity
3	USEINT2 polarity
4	USEINT3 polarity
5	USEINT4 polarity
6	USEINT5 polarity
7	USEINT6 polarity

Table 9.3.1, Interrupt Polarity Register

9.4. Interrupt Identification Register

This register identifies the active interrupts at the time of the register readout. A value of 1 for one bit of the Interrupt Identification Register signals an active interrupt level, a value of 0 signals an inactive level.

Bit	Description
0	PUSB_INT state (USB controller)
1	PETH_INT state (Ethernet controller)
2	USEINT1 state
3	USEINT2 state
4	USEINT3 state
5	USEINT4 state
6	USEINT5 state
7	USEINT6 state

Table 9.4.1, Interrupt Polarity Register

9.5. USB Suspend Register

This register is used to control and check the USB suspend pin. When a value of 0 has been written into this register, this register can be read to determine the state of the USB suspend pin. When a value of 1 has been written into this register, the USB suspend pin is pulled low.

Bit	Description
0	Read: return value of the USB suspend pin Write: USB suspend pin control. 0 = float, 1 = pull-down

Table 9.5.1, USB Suspend Register

9.6. Avoid interrupt conflicts on HY_INT4

The default jumper settings from chapters 6.6 / 6.7 / 6.9 and 6.10 map all interrupts to the XILINX and it redirects all to HY_INT4. This means that if you operate all in parallel (hyICE, LAN, USB), you will need to write your own interrupt handlers.

9.7. Example

The following C-code shows an example how to set and read the registers, and how to handle an interrupt.

```
#include <io.h>
#include <sys/hyrtk.h>

#define INSTALLUSB      1
#define INSTALLEETHER   2

/* Interrupt Controller Defines */
#define PLD_TIMING      0x3F8      /* !!! the slowest timing */
#define WriteInterruptMaskAddr ((1 << 22) | PLD_TIMING)
#define ReadInterruptMaskAddr  WriteInterruptMaskAddr
#define ReadIntIdentRegister  ((2 << 22) | PLD_TIMING)
#define WriteIntPolarRegister  ReadIntIdentRegister

/* USB */
#define USB_TIMING      0x3F8      /* !!! the slowest timing */
#define USB_ChipSelect  (3 << 22)
#define USB_Data        (USB_ChipSelect | USB_TIMING)
#define USB_Command     ( USB_ChipSelect | (1 << 13) |
                        USB_TIMING)

/*                      Data Flow Commands                      */

/* USB Interrupt */

int volatile  usb_InterruptOccured;

static void  int_routine0() {
    int IntRegByte1, IntRegByte2;

    USB_ReadIntReg(&IntRegByte1, &IntRegByte2);
    usb_InterruptOccured++;
}

/* Ethernet controller Interrupt */
static void  int_routine1()
{
    /* do something */
}

static void  int_routine2() { }
static void  int_routine3() { }
static void  int_routine4() { }
static void  int_routine5() { }
static void  int_routine6() { }
static void  int_routine7() { }
```

```

/*
NewSysInterruptRoutine reads interrupt identification register
and invokes the interrupt routine.
    input:      none
    returns:    1   user interrupt occurred
               0   if spurious or system interrupt (ICE
                   Interrupt) occurred
*/
int NewSysInterruptRoutine(void) {
    int iir, res, IntMask;

    iir = ~inpw(ReadIntIdentRegister); /* Low Polarity !!! */
    IntMask = inpw(ReadInterruptMaskAddr);
    iir &= IntMask;
    res = 1;

    if      (iir & 0x01) { int_routine0(); }
    else if (iir & 0x02) { int_routine1(); }
    else if (iir & 0x04) { int_routine2(); }
    else if (iir & 0x08) { int_routine3(); }
    else if (iir & 0x10) { int_routine4(); }
    else if (iir & 0x20) { int_routine5(); }
    else if (iir & 0x40) { int_routine6(); }
    else if (iir & 0x80) { int_routine7(); }
    else
        res = 0; /* none of the user interrupt is occurred,
                  pass control to the ICE */
    return res;
}

/*
_InstallSysInterrupt installs a new system interrupt routine.
    input: newsysintroutine -- (new interrupt entry of the sys
                                interrupt)
           SetMask          -- sets Interrupt enable Mask
           ClearMask        -- clears Interrupt enable Mask
    returns: 0
    Note: at reset only (system) ICE Interrupt is enable
*/
int _InstallSysInterrupt(int (* newsysintroutine)(), unsigned
                        int SetMask, unsigned int ClearMask)
{
    unsigned int NewIntMask;

    NewIntMask = inpw(ReadInterruptMaskAddr);
    NewIntMask |= SetMask;
    NewIntMask &= ~ClearMask;

    SetSysIntAddr(newsysintroutine);
    /* replaces the address of the hyRTK */
    /* INT4 handler with the one just written... */

    outpw(WriteInterruptMaskAddr, 0xFF & NewIntMask);
    /* writes the Interrupt mask to the XILINX */
    return 0;
}

```

```

/*
_InstallSysInterrupt installs a new system interrupt routine.
    input: newsysintroutine -- (new interrupt entry of the sys
                                interrupt)
           SetMask           -- sets Interrupt enable Mask
           ClearMask        -- clears Interrupt enable Mask
    returns: 0
    Note: at reset only (system) ICE Interrupt is enable
*/
int _setPolarityMask(unsigned int SetMask, unsigned int ClearMask)
{
    unsigned int NewPolMask;

    NewPolMask = inpw(ReadIntIdentRegister);
    NewPolMask |= SetMask;
    NewPolMask &= ~ClearMask;
    outpw(WriteIntPolarRegister, 0xFF & NewPolMask);
    /* writes the polarity mask to the XILINX */
    return 0;
}

/*    Read Interrupt Register */
int USB_ReadIntReg(int *IntRegByte1, int *IntRegByte2)
{
    outpw(USB_Command, 0xF4);
    *IntRegByte1 = 0xFF & inpw(USB_Data); /* read interrupt
                                           Register byte 1 */
    *IntRegByte2 = 0xFF & inpw(USB_Data); /* read interrupt
                                           Register byte 2 */
    return 0;
}

#if 1
#include <stdio.h>
/*
    Test Xilinx CPLD (XC9572XL7VQ64C)
*/
int main() {
    _InstallSysInterrupt (NewSysInterruptRoutine, INSTALLUSB, 0);
    while (!usb_InterruptOccured);
    printf("USB Interrupt occured.\n");

    return 0;
}
#endif

```

10. Board Dimensions

Figure 5 shows the dimensions of the expansion connectors.

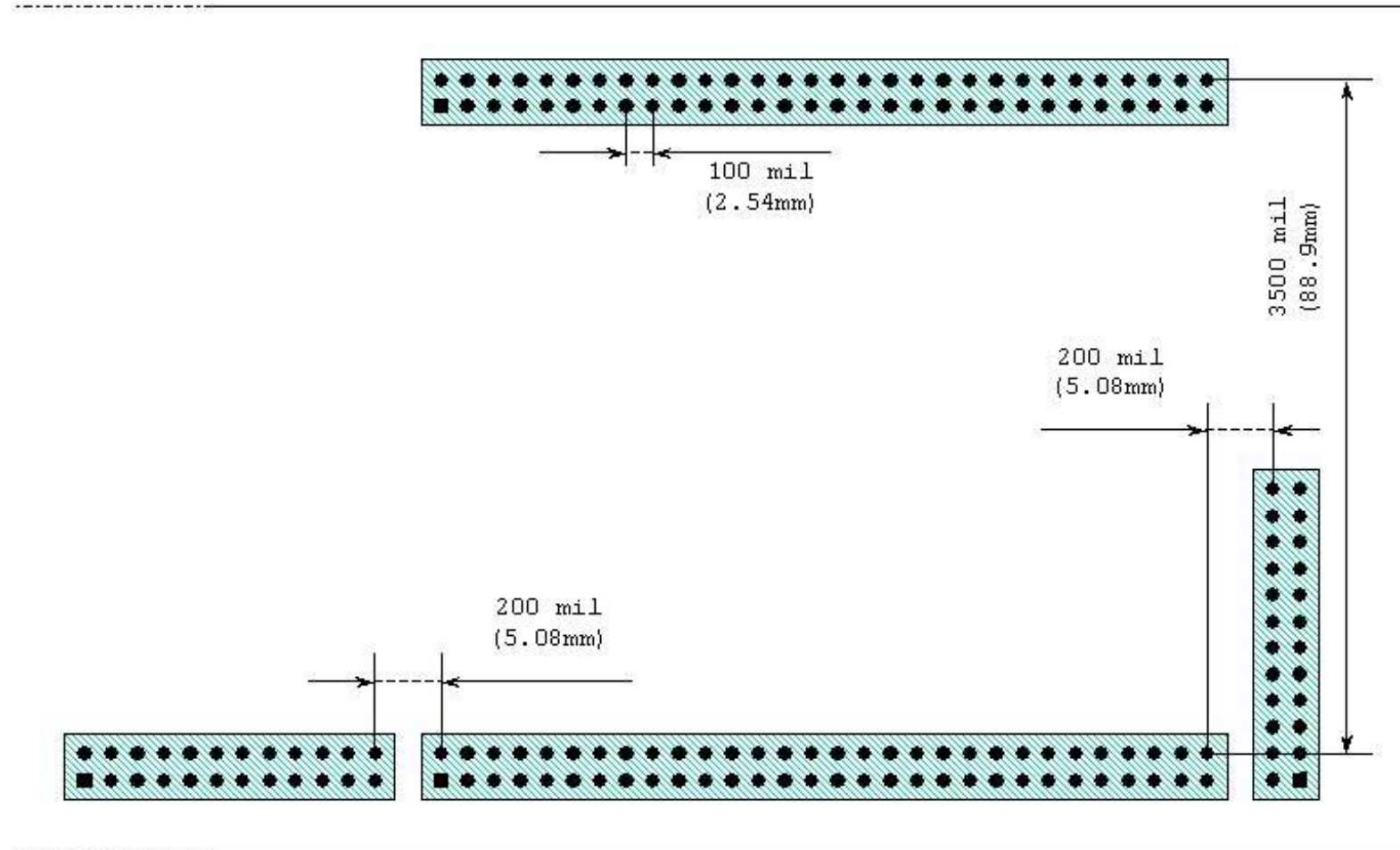


Fig. 5: Board Dimensions

11.1. hyBoard schematics

[illegible]

[illegible]

[illegible]

1

2

3

4

5

6

REVISION RECORD

LTR	ECO NO:	APPROVED:	DATE:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

11.3. XC9572 Equations

```

"
"      hyperstone AG
"      78467 Konstanz
"      Mihajlo Varga

"      05.04.2001

"      Development board for E1-32XS Micro Processor
"      Interrupt controller
"      Chip select generation
"      Tranceiver Enable
"      USB Suspend

MODULE devcs3
TITLE      'DEVBRD-E1-32XS'

development device 'XC9572XL7VQ64C';

DECLARATIONS

"Power Supply
" VCCINT1      Pin   3
" VCCINT2      Pin  37
" VCCI01       Pin  26
" VCCI02       Pin  55
" GND1         PIN   14
" GND2         PIN   21
" GND3         PIN   41
" GND4         PIN   54

" Not Connected
" Pin 16
" Pin 2
" Pin 5
" Pin 52
" Pin 50
" Pin 56
" Pin 57

" Pin Definitions

" Data Bus
HY_D0..HY_D7      Pin  60, 58, 59, 61, 62, 63, 1, 4;

" IO Address Bus
HY_ADDR22 Pin   17;
HY_ADDR23 Pin   11;
HY_ADDR24 Pin   13;
HY_ADDR25 Pin    9;

HY_IOWR_      Pin   15;
HY_IORD_      Pin   10;
RESET_        Pin   64;

HY_WE0_       Pin   43;
HY_WE1_       Pin   46;

```

```

HY_WE2_    Pin  47;
HY_WE3_    Pin  44;
HY_OE_     Pin  49;
HY_CS2_    Pin  45;
HY_CS3_    Pin  51;

```

```

TRANS_EN_  Pin  48;

```

```

ADDR_STR   Pin  18;
PUSB_SUSP  Pin  8;

```

```

" Chip Selects

```

```

PUSB_CS    Pin  19;
PETH_CS    Pin  6;
EXTCS1_    Pin  22;
EXTCS2_    Pin  31;
EXTCS3_    Pin  32;
EXTCS4_    Pin  24;
EXTCS5_    Pin  34;
EXTCS6_    Pin  25;
EXTCS7_    Pin  27;
EXTCS8_    Pin  39;

```

```

" Interrupt Inputs

```

```

PICE_INT_  Pin  12;
PUSB_INT_  Pin  20;
PETH_INT_  Pin  7;
USEINT1..USEINT6  Pin  33, 40, 35, 36, 42, 38;

```

```

" Interrupt Output

```

```

PLD_INT_   Pin  23;

```

```

" JTAG Interface

```

```

"TCK       Pin  30;
"TDI       Pin  28;
"TD0       Pin  53;
"TMS       Pin  29;

```

```

" Internal NODES

```

```

CS_IENMASK NODE;
CS_IDENT    NODE;
IENMASK0..IENMASK7  NODE  istype 'reg';

```

```

INTPOL0..INTPOL7  NODE  istype 'reg';
INTPOL           = [INTPOL0..INTPOL7];

```

```

PUSB_SUSPFF NODE istype 'reg';
CS_PUSB_SUSP NODE;
OUTENABLE   NODE;
ONBOARDINT, EXTENINT1, EXTENINT2  NODE;

```

```

DATA       = [HY_D0..HY_D7];
ADDR        = [HY_ADDR25..HY_ADDR22];
INT         = [PUSB_INT_, PETH_INT_, USEINT1..USEINT6];
IENMASK     = [IENMASK0..IENMASK7];

```

" ----- Boolean Equation Segment -----

EQUATIONS

" Address Mapping

"	HY_A25	HY_A24	HY_A23	HY_A22	
"	0	0	0	0	must not be used
"	0	0	0	1	Read: Int Mask Register
"	0	0	0	1	Write: Int Mask Register
"	0	0	1	0	Read: Int Ident Register
"	0	0	1	0	Write: Int Pol. Register
"	0	0	1	1	PUSB_CS
"	0	1	0	0	PETH_CS
"	0	1	0	1	EXTCS1_
"	0	1	1	0	EXTCS2_
"	0	1	1	1	EXTCS3_
"	1	0	0	0	EXTCS4_
"	1	0	0	1	EXTCS5_
"	1	0	1	0	EXTCS6_
"	1	0	1	1	EXTCS7_
"	1	1	0	0	EXTCS8_
"	1	1	0	1	CS_PUSB_SUSP

CS_IENMASK = ADDR == 1;

CS_IDENT = ADDR == 2;

PUSB_CS = ADDR == 3;

PETH_CS = ADDR == 4;

!EXTCS1_ = ADDR == 5;

!EXTCS2_ = ADDR == 6;

!EXTCS3_ = ADDR == 7;

!EXTCS4_ = ADDR == 8;

!EXTCS5_ = ADDR == 9;

!EXTCS6_ = ADDR == ^HA;

!EXTCS7_ = ADDR == ^HB;

!EXTCS8_ = ADDR == ^HC;

"EXTCS8: Chip select for multiplexed Motorola Interface

CS_PUSB_SUSP = ADDR == ^HD;

" Interrupt Controller

"

" Write Interrupt Enable Mask

"

IENMASK := (IENMASK & !CS_IENMASK) # (DATA & CS_IENMASK);

IENMASK.CLK = HY_IOWR_;

IENMASK.ACLR = !RESET_;

"

" Read Interrupt Enable Mask Register

" Read Interrupt Identification Register

" Read USB_SUSP pin

"

DATA = (IENMASK & CS_IENMASK) # (INT & CS_IDENT) #

(PUSB_SUSP & CS_PUSB_SUSP);

```

OUTENABLE      = ( CS_IENMASK # CS_IDENT # CS_PUSB_SUSP) &
!HY_IORD_;
DATA.OE        = OUTENABLE;

"
"      Write Interrupt Polarity Register
"
INTPOL          := (INTPOL & !CS_IDENT) # (DATA & CS_IDENT);
INTPOL.CLK      = HY_IOWR_;
INTPOL.ACLR     = !RESET_;

"
"      PUSB_SUSP is INPUT and Open Drain Output
"
"
"      Open Drain Output
"
PUSB_SUSPFF     := (PUSB_SUSPFF & !CS_PUSB_SUSP) # (HY_D0 &
CS_PUSB_SUSP);
PUSB_SUSPFF.CLK = HY_IOWR_;
PUSB_SUSPFF.AP  = !RESET_;

PUSB_SUSP       = 0;
PUSB_SUSP.OE    = !PUSB_SUSPFF;

"
"      Bus signal generation for multiplexed Motorola Interface
"      (for example real time clock DS12887)
"
ADDR_STR        = EXTCS8_ & !HY_IOWR_;
"DATA_STR       = ! HY_IORD_; doc only: Data strobe is made external
"

"  Note:   Address access must be Intel mode (IO A10 is 0)
"          Data access must be Motorola mode  (IO A10 is 1)
"
"  for example: read data at address 7
"      MOVI      Lx, 7
"      STW.IOA   0, Lx, ($C << 22) | TIMING ; bit 10 = 0
"      LDW.IOA   0, Lx, ($C << 22) | TIMING | (1 << 10) ; bit 10 = 1
"
"  for example: write data=12 at address 6
"      MOVI      Lx, 6
"      STW.IOA   0, Lx, ($C << 22) | TIMING ; bit 10 = 0
"      MOVI      Lx, 12
"      STW.IOA   0, Lx, ($C << 22) | TIMING | (1 << 10) ; bit 10 = 1
"

"
"      Shared Interrupt Generation
"
ONBOARDINT      = ( IENMASK0 & (PUSB_INT_ !$ INTPOL0)) #
                  ( IENMASK1 & (PETH_INT_ !$ INTPOL1)) #
                  !PICE_INT_ ;

EXTENINT1       = ( IENMASK1 & (USEINT1  !$ INTPOL1)) #
                  ( IENMASK2 & (USEINT2  !$ INTPOL2)) #

```

```
                (IENMASK3 & (USEINT3    !$ INTPOL3));

EXTENINT2      =  (IENMASK4 & (USEINT4    !$ INTPOL4)) #
                  (IENMASK5 & (USEINT5    !$ INTPOL5)) #
                  (IENMASK6 & (USEINT6    !$ INTPOL6));

!PLD_INT_      =  ONBOARDINT # EXTENINT1 # EXTENINT2;

" Tranceiver Enable
!TRANS_EN_    =  !HY_IORD_ # !HY_IOWR_ #
                  ((!HY_OE_ # !HY_WE0_ # !HY_WE1_ # !HY_WE2_ #
                    !HY_WE3_) & (!HY_CS2_ # !HY_CS3_));

END
```

**Hyperstone AG**

Line-Eid-Straße 3

D-78467 Konstanz

Germany

Phone +49 7531 / 9803-0

Fax +49 7531 / 51725

E-Mail info@hyperstone.de**Hyperstone Taiwan office**

7F-5, No. 75, Sec. 1, Hsin Tai Wu Road

Hsi Chih, Taipei Hsien,

Taiwan, R.O.C.

Phone +886 2 2698 2702

Fax +886 2 2698 2872

E-Mail Taiwan@hyperstone.com**URL:** <http://www.Hyperstone.com>