PHILIPS

Philips Semiconductors

Connectivity

Oct 2001

Application Note

Interfacing ISP1161 to Motorola DragonBall™ EZ RISC Processor

Rev 2.0

Revision History:

Version	Date	Descriptions	Author
Ver. 2.0	Oct 8, 2001	Updated schematic to reflect use of ES2	Yuk-lin Ong
Ver. 1.0	August 2001	First release	Socol Constantin

We welcome your feedback. Send it to apic@philips.com.

Interfacing ISP1161 to Motorola DragonBall EZ RISC

Table of Contents

Interfacing ISP1161 to Motorola DragonBall EZ RISC

1. Overview

The unique design of Philips ISP1161 makes it possible to use it as a host controller (with two downstream ports) as well as a device controller (with one upstream port); these ports may be accessed independently. In this way, a simultaneous connection as a host controller and a device controller can be achieved.

When ISP1161 is integrated into a personal digital assistant (PDA) or handheld personal computer (HPC), in most configurations this means connecting it to the external bus interface of a RISC processor. This application note deals with the critical issues in ISP1161's embedded design, using the Motorola DragonBall-EZ RISC processor as a concrete example.

2. ISP1161 Processor Interface Signals

The processor bus interface of ISP1161 was designed for a simple direct connection with a RISC processor. The data transfer can be done in PIO or DMA mode. The estimated maximum data transfer rate on the generic processor bus of the ISP1161—based on an ISP1161 internal clock frequency value equal to 48 MHz—is about 14 MB/s. To achieve the maximum data transfer rate on the host processor bus, the ISP1161 contains a Ping-Pong structured RAM which allows alternative access from the RISC processor or from the internal host/device controller. The "Ping-Pong" memory is allocated separately for the host and the device controllers. The host controller uses 2-KB of "Ping" memory and 2-KB of "Pong" in its allocated memory and the device controller uses 1.5 KB for each of the "Ping" memory and "Pong" in its own memory.

The main ISP1161 signals to consider when connecting to a Motorola DragonBall EZ RISC processor are:

- A 16-bit data bus: D0-D15 for ISP1161. ISP1161 is "little endian" compatible.
- Two address lines A0 and A1 necessary for complete addressing of the ISP1161 internal registers:
  - A0 = 0 and A1 = 0 selects the Data Port of the Host Controller
  - A0 = 1 and A1 = 0 selects the Command Port of the Host Controller
  - A0 = 0 and A1 = 1 selects the Data Port of the Device Controller
  - A0 = 1 and A1 = 1 selects the Command Port of the Device Controller
- One CS_N line used for selection of the ISP1161 in a certain address range of the host system. This input signal is active LOW.
- RD_N and WR_N are common read and write signals. These signals are active LOW.
- Two DMA channel standard control lines: DREQ1/2, DACK1/2 and EOT (one channel used by the host controller and the other channel used by the device controller). As the DragonBall EZ processor does not contain a DMA controller these signals will not be used in a minimal hardware implementation.
- Two interrupt lines: INT1 (used by the host controller) and INT2 (used by the device controller). Both have programmable level/edge and polarity (active HIGH or LOW).
- The CLKOUT signal has a maximum value of 48 MHz.
- The RESET signal is active LOW.

3. Motorola DragonBall EZ

Motorola MC 68EZ328, also called DragonBall EZ, is a processor from the second generation of the DragonBall EZ family. The MC68EZ328 combines a Motorola MC68EC000 processor with intelligent peripheral modules and typical system interface logic. The operating frequency of this processor is 16 MHz, obtained by an internal PLL, which accepts as input a 32.768kHz crystal, oscillator or a clock signal.

The main internal blocks of the DragonBall-EZ processor that should be considered when analyzing the bus interface, on which ISP1161 is connected, are as follows:

- The 8-bit/16-bit 68000 bus interface block
- Clock synthesizer and power control block
- Chip-select generation block
- The interrupt controller

Interfacing ISP1161 to Motorola DragonBall EZ RISC

The DragonBall-EZ processor generates eight programmable general-purpose chip-select signals, which are arranged in four groups of two (CSA [1:0], CSB [1:0], CSC [1:0] and CSD [1:0]). Each chip-select block allows selection of several features, which may be specific to the devices connected to each selected area:

- Bus size: 8 bits or 16 bits can be selected independently for each area. The default setting corresponds to a 16-bit bus size.
- The number of wait states inserted in a bus cycle can be set from zero to six wait states.
- Each area selected by a CSn can be defined as read-only or read/write access.
- Different types of memory are supported for an area selected by a certain CSn: DRAM, ROM, SRAM and FLASH memory.
- The size of any memory area selected by a Chip-Select signal can be selected from a set of predefined ranges (32K, 64K, 128K, 256K, 512K, 1M, 2M, 4M).

For a CSn signal to operate correctly, you must program the “Chip-Select Group Base Address Registers” and the “Chip Select Registers”, accordingly.

4. Considerations in Timing Diagrams and WAIT states

The following is a short study of the timing diagrams of ISP1161:
According to ISP1161 datasheet specifications, a read operation requires the following timing parameters (the write operation is similar), defined in Figure 1:

- tRL = 33 nsec (RD_N low pulse width – minimal value required by ISP1161),
- tRHRL = 110 nsec (RD_N HIGH to next RD_N LOW – minimal value required by ISP1161) and
- tRHDZ = 3 nsec (RD_N hold time, minimal value that can be expected from ISP1161).
- tRC = 143 nsec (will obviously result as a sum of tRL and tRHRL)
- tSHSL = 300 nsec (first RD_N/WR_N after CMD).

Consider the access of one of the internal registers of the ISP1161 (for example the Control Register of the HC) for a detailed analysis of the timing diagram. An access of one of the internal registers of ISP1161 will require two phases: first writing the address (index) of the selected register into the Command Port and then the data transfer access (RD/WR) may take place. Note: the index of each register is different in case of a RD or a WR operation.

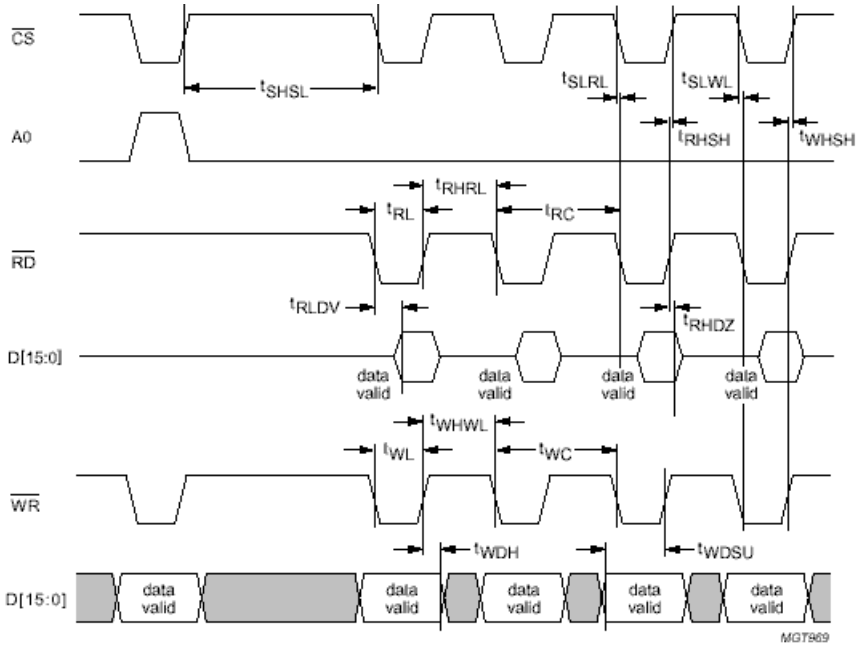


Figure 1. Parallel I/O Interface Timing (16-bit Read/Write)

Defining the connection area of ISP1161 as SRAM, a correct operation of the ISP1161 is ensured for a bus clock CKIO = 33 MHz and timing measurements show that insertion of wait-states in the standard bus cycles of the DragonBall EZ is not necessary. A description of the possible methods of wait-state insertion is done later anyway, taking into consideration the situation when faster bus cycles will be used for accessing the ISP1161.

Wait states insertion may be achieved generally using two solutions: hardware or software implementation. Both solutions will delay the rising edge of RD_N or WR_N to the next CKIO cycle and will determine an elongation of the RD_N or WR_N LOW pulse, that can be calculated as:

tW = W x T(CKIO); where: (W) is the number of wait states desired
T(CKIO) – is the cycle length of CKIO

Note: the value of tRHRL will not be modified by the number of wait states inserted by any of the solutions mentioned earlier. The value of this parameter must be calculated and correctly adjusted according to the number and length of instructions executed by the DragonBall EZ processor between two successive accesses to ISP1161. The “software solution” for wait-state insertion in a bus cycle is simple and is preferable in a minimal configuration if additional wait states are necessary.

5. Using Interrupts

ISP1161 will generate two interrupts on INT1 and INT2 pins, allocated for the Host and the Device controller, respectively. These interrupts will occur depending on the setting of the following registers: *Interrupt Register*, *Interrupt Enable Register* and the *Hardware Configuration Register*. These registers are separately programmed for host controller and for device controller operation.

You can connect the INT1 and INT2 signals directly to any of the available IRQ signals of the DragonBall processor. Both INT1 and INT2 of the ISP1161 are programmable as active on level or edge and HIGH or LOW- as specified in the *Hardware Configuration Registers* of the Host and Device controllers.

6. Suspend/Resume

You can make ISP1161 enter different functional states (Reset, Resume, Operational and Suspend) by programming the *Control Register* of the Host Controller or the *Mode Register* of the Device Controller.

Another way to “wake-up” ISP1161 from Suspend mode is to use the input signals H_WAKEUP (for the Host Controller) and D_WAKEUP (for the Device controller). Monitoring the H_SUSPEND pin (for Host status) and D_SUSPEND pin (for Device status) pins can determine the actual status of the ISP1161, without having to access the internal status registers.

ISP1161 may “wake up” when its CS_N input signal becomes active, if this is desired, by programming a “1” in bit 3 of the *Hardware Configuration Register* of the Device Controller. Alternatively, if the same bit is programmed to a value of “0”, asserting the CS_N signal does not cause ISP1161 to wake up.

7. Schematic Diagram

The following schematic diagram shows the connection of the ISP1161 to a Motorola DragonBall EZ processor, in a minimal hardware configuration. A more detailed description of the connection of ISP1161 to a RISC processor, a study of each category of signals and a description of the timing diagrams can be found in the application note “Interfacing ISP1161 toHitachi SH7709 RISC Processor”.

In this configuration, ISP1161 is selected by CSB0_N signal, which is asserted according to the values programmed in the “Chip Select Group Base Address” and “Chip Select” Registers corresponding to CSB0_N.

Interfacing ISP1161 to Motorola DragonBall EZ RISC

To correctly access the ISP1161, it is assumed that the area selected by CSB0_N is programmed for the SRAM memory type and for 16-bit bus width accesses.

Interrupts INT1 and INT2 are connected to IRQ2 and IRQ3 lines of the DragonBall EZ processor. The interrupt inputs of the DragonBall EZ processor can be set as edge/level sensitive and of positive/negative polarity by programming its *Interrupt Control Register*. The ISP1161 also allows programming the polarity (LOW/HIGH) and the signaling mode (level or pulse) for both generated interrupts INT1 and INT2, by setting the bits of the *Hardware Configuration registers* of the Host and Device controller respectively. This feature of ISP1161 allows a simple connection of the IRQ lines, without any additional logic. The interrupts of the DragonBall processor can be masked in the *Interrupt Mask Register;* checking the status of an interrupt line can be done by monitoring the values of the *Interrupt Status Register* and of the *Interrupt Pending Register* of the DragonBall-EZ processor.

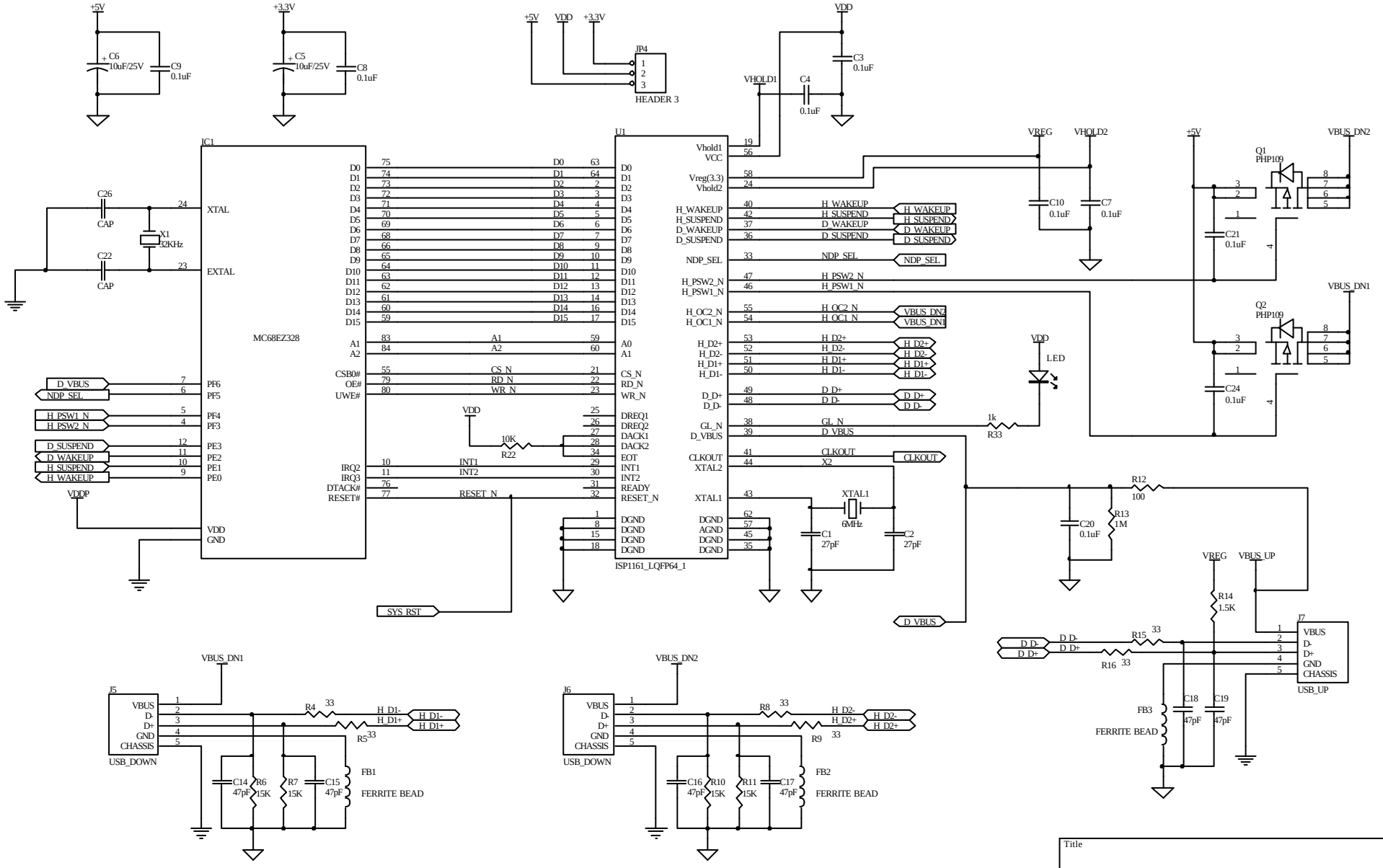
Analysis of the timing diagrams of the ISP1161 and of the DragonBall-EZ processor, running at a standard frequency of 16 MHz, shows that insertion of wait states in a standard bus cycle is not necessary. In case a similar RISC processor with a higher bus frequency is used, if necessary, it is possible to insert a certain number of wait-states during a bus cycle, accessing a certain system resource selected by CSn, by programming the selected value of wait states in the corresponding *(Chip-Select)n Register* of the DragonBall processor.

Input signals H_OC1_N and H_OC2_N are used by the ISP1161 for detecting an overcurrent value on the downstream ports. As separate overcurrent detection and protection circuits are implemented in ISP1161, detection of an overcurrent on one of the downstream ports will power down that port only. Connecting the two voltages of the two downstream ports VBUS_DN1 and VBUS_DN2 to H_OC1_N and H_OC2_N pins enables detection of the current value, by sensing the voltage drop on Q1 and Q2 which are MOSFET transistors with very low switch-on resistance Rds. Selection of Q1 and Q2 will be done depending on the desired maximum current value and that will determine the value of Rds(on). For example, considering that a voltage drop of 75mV will trigger the overcurrent circuitry and the allowed maximum current is about 0.5A, an approximate value of Rds(on) = 150mOhm will result. Connecting the 1161 input pins H_OC1_N and H_OC2_N to +5V will disable the internal overcurrent protection of ISP1161; an external overcurrent protection circuit may also be used.

The number of downstream ports is determined by the setting of NDP_SEL input signal of ISP1161. In a minimal hardware configuration, using a simple jumper option it is possible to set one or two downstream ports accordingly.

Detection of a connection on the upstream port is achieved by connecting VBUS_UP to pin D_VBUS of the ISP1161. R12 and C20 will act as a low-pass filter that eliminates the possibility of sensing false voltage drop due to load current variations or noise on VBUS_UP. It is recommended, if possible, to implement a hybrid power solution, by using VBUS_UP to power the ISP1161 and an external power source for the rest of the system.

The GL_N output signal indicates, through an LED, the status of the USB device and helps in trouble-shooting the USB connection.
The RESET input signal of ISP1161 is connected to the system RESET signal, which also generates the RESET input signal for the DragonBall-EZ processor.


+5V
+3.3V
C6
10uF/25V
C9
0.1uF
C5
10uF/25V
C8
0.1uF
JP4
HEADER 3
VDD
C3
0.1uF
VHOLD1
C4
0.1uF
IC1
MC68EZ328
XTAL
EXTAL
C26
CAP
C22
CAP
X1
32KHz
U1
ISP1161_LQFP64_1
VREG
VHOLD2
C10
0.1uF
C7
0.1uF
Q1
PHP109
Q2
PHP109
VBUS_DN2
VBUS_DN1
C21
0.1uF
C24
0.1uF
LED
1k
R33
D_VBUS
NDP_SEL
H_PSW1_N
H_PSW2_N
D_SUSPEND
D_WAKEUP
H_SUSPEND
H_WAKEUP
VDDP
SYS_RST
10K
R22
RESET_N
CLKOUT
XTAL1
6MHz
C1
27pF
C2
27pF
C20
0.1uF
R12
100
R13
1M
R14
1.5K
VBUS_UP
J7
USB_UP
R15
33
R16
33
FB3
FERRITE BEAD
C18
47pF
C19
47pF
J5
USB_DOWN
R4
33
R5
33
C14
47pF
R6
15K
R7
15K
C15
47pF
FB1
FERRITE BEAD
J6
USB_DOWN
R8
33
R9
33
C16
47pF
R10
15K
R11
15K
C17
47pF
FB2
FERRITE BEAD
Title
Size
B
Number
Revision
Date:
8-Oct-2001
Sheet of
File:
E:\Jason Backup 030701\Philips\ISP1161\Application Note\Dragon ball DDB