# Vortex86EX2

# Fact Sheet

***32-bit x86 Micro Processor / Slave Core***

# 1. Overview

The **Vortex86EX2** is a low-power SoC. It integrates two fully static 32-bit X86 processor, Master System with the compatibility of Windows, Linux and most popular 32-bit RTOS; Slave System with also the compatibility of Windows, Linux, DOS and most popular 32-bit RTOS. Master System integrates I-16KB and D-16KB write through 4-way L1 cache, 128KB write through/write back 4-way L2 cache. Slave System integrates I-16KB and D-16KB write through 4-way L1 cache. It also integrates PCIe Gen1 x1 Port, DDR3, CrossBar Interface, ISA, $I^2C$, SPI, IPC (Internal Peripheral Controllers with DMA and interrupt timer/counter included), Fast Ethernet, FIFO UART, USB2.0 Host, USB2.0 Device, PCIe Device, SD and CAN controller within a single package to form a system-on-a-chip (SoC). It provides an ideal solution for low-cost and power-efficiency embedded system with desired performance.

# 2. Features

## 2.1. Slave SoC Only

- **X86 Processor Core**
  - 6-stage pipeline
  - Clock support up to 400MHz
  - X86 instruction set
- **Floating Point Unit Support**
  - Extends CPU instruction set to include Trigonometric, Logarithmic and Exponential
- **Embedded I/D Separated L1 Cache**
  - 16-KByte I-Cache, 16-KByte D-Cache
- DDR3 Control Interface
  - 16 bit data bus
  - 2 ranks
  - Support ECC
  - DDR clock support up to 400MHz
- **DMA Controller x2**
- **Programmable Interrupt Controller**
- **Counter / Timers**
  - 1 set of 8254 timer controller
- **Real Time Clock**
- **JTAG Interface supported for Software Debugging**



 

## 2.2. Share with Master SoC

- **LCD Interface**
- **SD Interface x3**
  - Support SDSC, SDHC and SDXC card.
  - Compliant upto eMMC Version 5.1
- **Fast Ethernet MAC x2**
- **Fast Ethernet PHY**
- **PCIE Host/Target Gen 1 x1 Interface**
- **PCIE Host Gen 1 x1 Interface**
- **USB Interface**
  - Port 0/1 support Host (HS, FS and LS)
  - Port 1 support HS Device
- **HDA Controller**
  - 4 input streams, 4 output streams
- **12bit ADC Interface x 2**
  - 8 channels
  - Support DMA Controller
- **$I^2C$ bus x2**
  - Compliant with V2.1
  - Some master code (general call, START and CBUS) not support.
  - Up to the fast speed.
- **General Purpose SPI Controller x2**
  - Some master code (general call, START and CBUS) not support.
  - Support SPI Device x2 (SPI_CS# x2)
  - Support DMA Controller
- **CAN Bus Interface x2**
  - Compatible with CAN 2.0A/2.0B
- **Motion Control Interface x3**
  - 1groups of controller, 4 controllers per group
  - Each controller can be configured to PWM/Servo/Sensor Interface mode
  - Controller interconnect to the other with routing network in the same group
- **CrossBar Interface**
  - 16 CrossBar ports for digital function select. (each port is 8 pins, total 128 pins)
- **FIFO UART Port x 10 ( 10 sets COM Port )**
  - Compatible with 16C550 / 16C552
  - Default internal pull-up
  - Support programmable baud rate generator with data rate from 50 to 20M bps
  - The character options are programmable for 1 start bits; 1, 1.5 or 2 stop bits; even, odd or no parity; 5~8 data bits
  - Support TXD_En Signal on COM1-10
  - Port 80h output data could be sent to COM1 by software programming
  - Support half-duplex mode
  - Enhanced low I/O access latency
- **General Programmable I/O**
  - Support 128 programmable I / O pins
  - Each GPIO pin can be individually configured to be an input/output pin
  - GPIO_P0~GPIO_P9 can be programed by 8051A
  - Support GPIO Interrupt Controller (input/output) x2
  - Support DMA Controller



 

# 3. Register Sets

The SoC contains four sets of software accessible registers (Core registers, MSR, I/O Mapped registers and Configuration registers).

## 3.1. Core Registers

The SoC provides 24 Core Registers. The 16 Base Architecture Registers (General-purpose Registers, Segment Registers, Flags Register and Instruction Pointer) are used in general system and application programming. The other 8 system-level registers (Control Registers and System Address Registers) can be used only by system-level programs. These registers are shown below. The details will be described in Register Description in Chapter 4.

### 3.1.1. General-Purpose Registers

| Register Name |
|---|
| EAX Register |
| EBX Register |
| ECX Register |
| EDX Register |
| ESI Register |
| EDI Register |
| EBP Register |
| ESP Register |



 

### 3.1.2. Segment Registers

| Register Name |
|---|
| Code Segment Register – CS |
| Stack Segment Register – SS |
| Data Segment Register – DS |
| Data Segment Register – ES |
| Data Segment Register – FS |
| Data Segment Register – GS |

### 3.1.3. Instruction Pointer Register

| Register Name |
|---|
| Instruction Pointer Register |

### 3.1.4. Flags Register

| Register Name |
|---|
| Flags Register |

### 3.1.5. Control Registers

| Register Name |
|---|
| Control Register 0 |
| Control Register 1 |
| Control Register 2 |
| Control Register 3 |



 

### 3.1.6. System Address Registers

| Register Name |
|---|
| Global Descriptor Table Register |
| Interrupt Descriptor Table Register |
| Local Descriptor Table Register |
| Task State Segment Register |

## 3.2. CPU MSR Registers

### 3.2.1. MSR Registers

| MSR Index | MSR Name |
|---|---|
| 10h | Time-Stamp Counter |
| 1Bh | Reserved |
| 174h | Reserved |
| 175h | Reserved |
| 176h | Reserved |
| 52444300h | Reserved |
| 52444301h | Reserved |
| 52444303h | Reserved |
| 52444304h | Reserved |
| 52444305h | Reserved |
| 52444306h | Reserved |
| CFCFCF00h | Reserved |
| D0D0D000h | Instruction Counter Register |
| D0D0D001h | User Instruction Counter Register |
| D0D0D002h | Instruction Counter Control Register |



 

## 3.3. PCI Configuration Space Registers

The SoC integrated three PCI base bridges – Host-to-PCI Bridge, PCIE-to-PCI Bridge, ISA-to-PCI Bridge, two MAC, two SD, one USB1.1 host, one USB2.0 host, one USB device, one HDA, three Motion Controller, two CAN and one LCD Controller. These three bridges contain their own PCI Configuration Space. Configuration Space Registers reside in PCI Configuration Space and specify PCI configuration, DRAM configuration, operating parameters, and optional system features.

During hardware reset, the SoC sets its internal configuration registers to predetermine default states. The default state represents the minimum functionality feature set required to successfully bring up the system. Hence, it does not represent the optimal system configuration. It is the responsibility of the system initialization software (usually BIOS) to properly determine the DRAM configurations, operating parameters and optional system features that are applicable, and to program the registers accordingly.

### 3.3.1. North Bridge Function 0 Configuration Space Registers

(IDSEL = AD11/Device 0/Function 0)

| Offset | Register Name |
|---|---|
| 01h – 00h | Vendor ID Register |
| 03h – 02h | Device ID Register |
| 05h – 04h | Command Register |
| 07h – 06h | Status Register |
| 08h | Revision ID Register |
| 0Bh – 09h | Class Code Register |
| 0Eh | Header Type Register |
| 2Dh – 2Ch | Subsystem Vendor ID Register |
| 2Fh – 2Eh | Subsystem Device ID Register |
| 43h – 40h | NB SPI Controller Base Address Register |
| 46h | Slave CPU Clock Divided Control Register |
| 47h | Slave PCI Clock Divided Control Register |
| 4Bh | LCD UMA Control Register |
| 50h | Boot SPI Flash Decode Size Control Register |
| 51h | Flash Strap Checksum Status Register |
| 57h – 54h | Slave System Control Register |
| 5Bh – 58h | Device System Access Select Register |



 

| Offset | Register Name |
|---|---|
| 5Fh – 5Ch | PCI Device System Access Select Register |
| 63h – 60h | STRAP Register 1 |
| 67h – 64h | STRAP Register 2 |
| 6Bh – 68h | STRAP Register 3 |
| 6Dh - 6Ch | Memory Bank Register |
| 73h - 70h | NB Control Register |
| 83h | A/B Page and SMM Range Register |
| 87h – 84h | Shadow RAM Register |
| 93h – 90h | Customer ID Register |
| 97h – 94h | Spare 1 Register |
| 9Bh – 98h | Spare 2 Register |
| 9Fh – 9Ch | Spare 3 Register |
| A3h – A0h | Host Control Register |
| B3h – B0h | GPIO Port Configuration IO Base Address Register |
| B7h – B4h | GPIO Interrupt Configuration IO Base Address Register |
| BFh – BCh | FIFO COM Configuration Memory Base Address Register |
| C1h | PIC Latch Enable Register |
| C2h | Master PIC Input Latch Control Register (when North Bridge C1h[1]=1) |
| C3h | Slave PIC Input Latch Control Register (when North Bridge C1h[1]=1) |
| F3h – F0h | Reserved |
| F7h – F4h | Reserved |
| FBh – F8h | Reserved |
| FCh | Reserved |
| FDh | Reserved |



 

### 3.3.2. North Bridge Function 1 Configuration Space Registers

(IDSEL = AD11/Device 0/Function 1 )

| Offset | Register Name |
|---|---|
| 01h – 00h | Vendor ID Register |
| 03h – 02h | Device ID Register |
| 05h – 04h | Command Register |
| 07h – 06h | Status Register |
| 08h | Revision ID Register |
| 0Bh – 09h | Class Code Register |
| 0Eh | Header Type Register |
| 2Dh – 2Ch | Subsystem Vendor ID Register |
| 2Fh – 2Eh | Subsystem Device ID Register |
| A3h – A0h | DRAM ECC Control register |
| A7h – A4h | DRAM ECC Counter Register |

### 3.3.3. South Bridge Function 0 Configuration Space Registers

(IDSEL = AD18/Device 7/Function 0)

| Offset | Register Name |
|---|---|
| 01h – 00h | Vendor ID Register |
| 03h – 02h | Device ID Register |
| 05h – 04h | Command Register |
| 07h – 06h | Status Register |
| 08h | Revision ID Register |
| 0Bh – 09h | Class Code Register |
| 0Eh | Header Type Register |
| 2Dh – 2Ch | Subsystem Vendor ID Register |
| 2Fh – 2Eh | Subsystem Device ID Register |
| 40h | System Fault Count Register |
| 41h | Write Lock and KBC Control Register |
| 43h – 42h | Reserved Register |



 

| Offset | Register Name |
|---|---|
| 46h | Manual Reset Control Register |
| 5Bh – 58h | PCI Interrupt Routing Table Register |
| B7h – B4h | PCI Interrupt Routing Table2 Register |
| C3h – C0h | Internal Peripheral Feature Control Register |
| CEh | South Bridge STRAP Register |
| F0h | RTC Test Register |

### 3.3.4. South Bridge Function 1 Configuration Space Registers

(IDSEL = AD18/Device 7/Function 1)

| Offset | Register Name |
|---|---|
| 01h – 00h | Vendor ID Register |
| 03h – 02h | Device ID Register |
| 05h – 04h | Command Register |
| 07h – 06h | Status Register |
| 08h | Revision ID Register |
| 0Bh – 09h | Class Code Register |
| 0Eh | Header Type Register |
| 2Dh – 2Ch | Subsystem Vendor ID Register |
| 2Fh – 2Eh | Subsystem Device ID Register |
| B7h – B4h | PCI Interrupt Routing Table3 Register |



 

### 3.3.5. Hybrid Function Device Controller Configuration Space Registers

(IDSEL = AD17/Device6)

| Offset | Register Name |
|---|---|
| 01h – 00h | Vendor ID Register |
| 03h – 02h | Device ID Register |
| 05h – 04h | Command Register |
| 07h – 06h | Status Register |
| 08h | Revision ID Register |
| 09h | Program Interface Register |
| 0Ah | Sub-Class Code Register |
| 0Bh | Base Class Code Register |
| 0Ch | Cache Line Size Register |
| 0Dh | Latency Timer Register |
| 0Eh | Header Type Register |
| 0Fh | Built-In Self Test Register |
| 2Dh – 2Ch | Subsystem Vendor ID Register |
| 2Fh – 2Eh | Subsystem Device ID Register |
| 33h – 30h | Expansion ROM Base Address Register |
| 34h | Capabilities Pointer Register |
| 3Ch | Interrupt Line Register |
| 3Dh | Interrupt PIN Register |
| 3Eh | Minimum Grant Register |
| 3Fh | Max Latency Register |
| 43h – 40h | System Access Select Register |
| 47h – 44h | UART Control IO Base Address Register |
| 4Bh – 48h | Hybrid Functions Control IO Base Address Register |



 

## 3.4. I / O Mapped Registers

The I/O Mapped Registers are usually used to control the SoC integrated peripherals or to store the peripherals' data, addresses and statuses. We divided these I/O Mapped Registers into below subsets.

These registers are listed as below. In another chapters, Functions and Registers Description will show more detailed information about these registers.

### 3.4.1. PCI Configuration Registers

| I/O Address | Register Name |
|---|---|
| 0CFBh-0CF8h | PCI Configuration Address Register |
| 0CFFh-0CFCh | PCI Configuration Data Register |

### 3.4.2. Slave DMA Controller Registers

| I/O Address | Register Name |
|---|---|
| 00h | Slave DMA Channel 0 Base/Current Address Register |
| 01h | Slave DMA Channel 0 Base/Current Count Register |
| 02h | Slave DMA Channel 1 Base/Current Address Register |
| 03h | Slave DMA Channel 1 Base/Current Count Register |
| 04h | Slave DMA Channel 2 Base/Current Address Register |
| 05h | Slave DMA Channel 2 Base/Current Count Register |
| 06h | Slave DMA Channel 3 Base/Current Address Register |
| 07h | Slave DMA Channel 3 Base/Current Count Register |
| 08h | Slave DMA Command/Status Register |
| 09h | Slave DMA Command/Request Register |
| 0Ah | Slave DMA Command/Single Mask Register |
| 0Bh | Slave DMA Mode Register |
| 0Ch | Slave DMA Set/Clear First/Last Clear F/F Register |
| 0Dh | Slave DMA Temporary/Master Disable Register |
| 0Eh | Slave DMA Clear Mask/Mode register pointer Register |
| 0Fh | Slave DMA Write Mask Register |



 

### 3.4.3. DMA Page Registers

| I/O Address | Register Name |
|---|---|
| 81h | DMA Page Register – DMA Channel 2 |
| 82h | DMA Page Register – DMA Channel 3 |
| 83h | DMA Page Register – DMA Channel 1 |
| 87h | DMA Page Register – DMA Channel 0 |
| 89h | DMA Page Register – DMA Channel 6 |
| 8Ah | DMA Page Register – DMA Channel 7 |
| 8Bh | DMA Page Register – DMA Channel 5 |

### 3.4.4. Master DMA Controller Registers

| I/O Address | Register Name |
|---|---|
| C0h | Master DMA Channel 4 Base/Current Address Register |
| C2h | Master DMA Channel 4 Base/Current Count Register |
| C4h | Master DMA Channel 5 Base/Current Address Register |
| C6h | Master DMA Channel 5 Base/Current Count Register |
| C8h | Master DMA Channel 6 Base/Current Address Register |
| CAh | Master DMA Channel 6 Base/Current Count Register |
| CCh | Master DMA Channel 7 Base/Current Address Register |
| CEh | Master DMA Channel 7 Base/Current Count Register |
| D0h | Master DMA Command/Status Register |
| D2h | Master DMA Command/Request Register |
| D4h | Master DMA Command/Single Mask Register |
| D6h | Master DMA Mode Register |
| D8h | Master DMA Set/Clear First/Last Clear F/F Register |
| DAh | Master DMA Temporary/Master Disable Register |
| DCh | Master DMA Clear Mask/Mode register pointer Register |
| DEh | Master DMA Write Mask Register |

### 3.4.5. DMA High Page Registers

| I/O Address | Register Name |
|---|---|



 

| | |
|---|---|
| 481h | DMA High Page Register – DMA Channel 2. |
| 482h | DMA High Page Register – DMA Channel 3. |
| 483h | DMA High Page Register – DMA Channel 1. |
| 487h | DMA High Page Register – DMA Channel 0. |
| 489h | DMA High Page Register – DMA Channel 6. |
| 48Ah | DMA High Page Register – DMA Channel 7. |
| 48Bh | DMA High Page Register – DMA Channel 5. |

### 3.4.6. Timer / Counter Registers

| I/O Address | Register Name |
|---|---|
| 40h | Timer / Counter 0 Count Register |
| 41h | Timer / Counter 1 Count Register |
| 42h | Timer / Counter 2 Count Register |
| 43h | Timer / Counter Control Register |

### 3.4.7. CMOS Memory & RTC Registers

| I/O Address | Register Name |
|---|---|
| 70h | CMOS Memory Address Register |
| 71h | CMOS Memory Data Register |

### 3.4.8. WDT Registers

**WDT1 Control Register**

| I/O Address | Register Name |
|---|---|
| A8h | WDT1 Control Register |
| A9h | WDT1 Signal Select Control Register |
| AAh | WDT1 Counter 0 Register |
| ABh | WDT1 Counter 1 Register |
| ACh | WDT1 Counter 2 Register |
| ADh | WDT1 Status Register |

 

**WDT Reload Register**

| I/O Address | Register Name |
|---|---|
| AEh | WDT1 Reload Register |

### 3.4.9. Master Interrupt Controller Registers

| I/O Address | Register Name |
|---|---|
| 20h | Master Interrupt Request/Interrupt Service/Interrupt Command Register |
| 21h | Master Interrupt Mask Register |

### 3.4.10. Slave Interrupt Controller Registers

| I/O Address | Register Name |
|---|---|
| A0h | Slave Interrupt Request/Interrupt Service/Interrupt Command Register |
| A1h | Slave Interrupt Mask Register |

### 3.4.11. Interrupt Edge / Level Control Registers

| I/O Address | Register Name |
|---|---|
| 4D0h | Master Interrupt Edge/Level Control Register |
| 4D1h | Slave Interrupt Edge/Level Control Register |

### 3.4.12. Keyboard / Mouse Control Registers

| I/O Address | Register Name |
|---|---|
| 60h | Output Buffer Register |
| 64h | Input Buffer / Status / Command Register |

### 3.4.13. 8051 Firmware Code Address / Data Registers

| I/O Address | Register Name |
|---|---|
| 62h | Address Register |
| 66h | Data Register |



 

### 3.4.14. Spare Registers

| I/O Address | Register Name |
|---|---|
| 80h | Spare Register |
| 84h | Spare Register |
| 85h | Spare Register |
| 86h | Spare Register |
| 88h | Spare Register |
| 8Ch | Spare Register |
| 8Dh | Spare Register |
| 8Eh | Spare Register |
| 8Fh | Spare Register |
| 480h | Spare Register |
| 484h | Spare Register |
| 485h | Spare Register |
| 486h | Spare Register |
| 488h | Spare Register |
| 48Ch | Spare Register |
| 48Dh | Spare Register |
| 48Eh | Spare Register |
| 48Fh | Spare Register |

### 3.4.15. System Control Register

| I/O Address | Register Name |
|---|---|
| 92h | System Control Register |

### 3.4.16. Indirect Access Registers

| I/O Address | Register Name |
|---|---|
| 22h | Address Index Register for indirect access GPIO & WDT0 |
| 23h | Data Register for indirect access GPIO & WDT0 |



 

### 3.4.17. NMI Status and Control Register

| I/O Address | Register Name |
|---|---|
| 61h | NMI Status and Control Register |

### 3.4.18. Hybrid Function Control Registers

(Base Address Refers to the Register of index 4Bh-48h, Device 6, Function 0, Hybrid Function Device Controller Configuration Space Register)

| Offset | Register Name |
|---|---|
| BA + 00h | SPI 0 Control Base Address Register |
| BA + 04h | SPI 0 Control Register |
| BA + 08h | SPI 1 Control Base Address Register |
| BA + 0Ch | SPI 1 Control Register |
| BA + 10h | I2C 0 Control Register |
| BA + 14h | I2C 1 Control Register |
| BA + 18h | ADC 0 Control Register |
| BA + 1Ch | ADC 1 Control Register |
| BA + 20h | CrossBar Control Registers |
| BA + 24h | Device Power Saving Control Register |
| BA + 28h | SPI 0 DMA Configuration Memory Base Address Register |
| BA + 2Ch | SPI 1 DMA Configuration Memory Base Address Register |
| BA + 30h | ADC 0 DMA Configuration Memory Base Address Register |
| BA + 34h | ADC 0 DMA Configuration Memory Base Address Register |

### 3.4.19. FIFO COM Control Registers

(Base Address Refers to the Register of index BFh-BCh, Device 0, Function 0, North Bridge Configuration Space Register)

| Offset | Register Name |
|---|---|
| BA + 00h | FIFO COM Control Register |
| BA + 04h | FIFO COM Status Register |
| BA + 08h | FIFO COM TX Data Register |
| BA + 0Ch | FIFO COM RX Data Register |



 

## 3.5. The Registers Only Reset by PWRGOOD

| I/O Address | Register Name |
|---|---|
| 23h | Indirect access data register |
| North Bridge Function 0 Configuration 57h – 54h bit[30] | Slave System Not Reset When Master System Reset |
| North Bridge Function 0 Configuration 5Bh – 58h | Device System Access Select Register |
| North Bridge Function 0 Configuration 5Fh – 5Ch | PCI Device System Access Select Register |
| North Bridge Function 0 Configuration B3h – B0h | GPIO Port Configuration I/O Base Address Register |
| North Bridge Function 0 Configuration B7h – B4h | GPIO Interrupt Configuration I/O Base Address Register |
| South Bridge Function 0 Configuration 40h | System Fault Count Register |
| South Bridge Function 0 Configuration 46h | Manual Reset Control Register |
| Hybrid Function Device Configuration 43h – 40h | System Access Select Register |
| Hybrid Function Device Configuration 47h – 44h | UART Configuration I/O Base Address Register |
| Hybrid Function Device Configuration 4Bh – 48h | Hybrid Function Control IO Base Address Register |
| Indirect access registers, Index port 13h | GPIO Lock/unlock function |
| GPIO Data Address | GPIO PORT Data Register |



 

| | |
|---|---|
| GPIO Direction Address | GPIO PORT Direction Register |
| BA + 0h, (Note1) | General-Purpose I/O Data & Direction Decode Enable |
| BA + 04h, 08h, 0Ch, 10h, 14h, 18h, 1Ch, 20h, 24h, 28h, 2Ch, 30h, 34h, 38h, 3Ch, 40h (Note1) | General-Purpose I/O 0,1~,15 Data & Direction Decode Address |
| BA + 44h, 48h, 4Ch, 50h, 54h, 58h, 5Ch, 60h, 64h, 68h, 6Ch, 70h, 74h, 78h, 7Ch, 80h (Note1) | General-Purpose I/O 0,1~,15 Control Base Address |
| BA + 0h, (Note2) | General-Purpose I/O Interrupt Status Decode Address |
| BA + 04h, (Note2) | General-Purpose I/O Interrupt Port Select |
| BA + 08h, (Note2) | General-Purpose I/O Interrupt Control 0 Register |
| BA + 0Ch, (Note2) | General-Purpose I/O Interrupt Control 1 Register |
| BA+00h, (Note3) | GPIO PORT Data Register |
| BA+00h, (Note3) | GPIO PORT Direction Register |
| BA + 0h, (Note4) | GPIO PORT Interrupt Status 0 Register |
| BA + 1h, (Note4) | GPIO PORT Interrupt Status 1 Register |
| BA + 20h (Note5) | CrossBar Control Register |
| BA + 00h, 01h, 02h, …, 0Fh, (Note6) | RichIO G0 Port 0, 1, 2 …, 15 Selection Register |
| BA + 10h, 11h, ..., 17h, (Note6) | Bit-RichIO G0 Port0 Select Register |
| BA + 18h, 19h, ..., 1Fh, (Note6) | Bit-RichIO G0 Port1 Select Register |



 

| | |
|---|---|
| BA + 20h, 21h, ..., 27h, (Note6) | Bit-RichIO G0 Port2 Select Register |
| BA + 28h, 29h, ..., 2Fh, (Note6) | Bit-RichIO G0 Port3 Select Register |
| BA + 30h, 31h, 32h, ..., 3Fh, (Note6) | RichIO G1 Port 0, 1, 2 …, 15 Selection Register |
| BA + 40h, 41h, ..., 47h, (Note6) | Bit-RichIO G1 Port0 Select Register |
| BA + 48h, 49h, ..., 4Fh, (Note6) | Bit-RichIO G1 Port1 Select Register |
| BA + 50h, 51h, ..., 57h, (Note6) | Bit-RichIO G1 Port2 Select Register |
| BA + 58h, 59h, ..., 5Fh, (Note6) | Bit-RichIO G1 Port3 Select Register |
| BA + 60h, 61h, ..., 67h, (Note6) | Bit-RichIO G1 Port4 Select Register |
| BA + 70h, 71h, ..., 8Fh, (Note6) | CrossBar Port 0[7:0], 1[7:0], …, 3[7:0] PAD Attribute Register |
| BA + 90h, 91h, ..., 9Fh, (Note6) | CrossBar Port 4[7:0], 5[7:0] PAD Attribute Register |
| BA + A0h, A1h, ..., EFh, (Note6) | CrossBar Port 6[7:0], 7[7:0], …, 15[7:0] PAD Attribute Register |
| BA + F6h, F7h, ..., F9h, (Note6) | CrossBar Port Group Selection Regsister, Port6, Port7, …, Port9 |
| BA + FAh, F7h, ..., FFh, (Note6) | CrossBar Port Group Selection Regsister, Port10, Port11, …, Port15 |
| BA + 100h, 101h, 102h, ...., 10Fh, (Note6) | CrossBar Bit Group Selection Regsister, Port0[7:0], Port1[7:0] |
| BA + 110h, 111h, 112h, ...., 11Fh, (Note6) | CrossBar Bit Group Selection Regsister, Port2[7:0], Port3[7:0] |
| BA + 120h, 121h, 122h, ...., 12Fh, (Note6) | CrossBar Bit Group Selection Regsister, Port4[7:0], Port5[7:0] |

**Note:**

1. BA refers to the Register of index B3h-B0h, North Bridge Function 0 PCI



 

Configuration Register

2. BA refers to the Register of index B7h-B4h, North Bridge Function 0 PCI Configuration Register
3. GPIO Port0~15, BA refers to the General-Purpose I/O 0,1~,15 Data & Direction Base Address
4. BA refers to the General-Purpose I/O Interrupt Status Decode Address
5. BA refers to the Register of index 4Bh-48h, Hybrid Function PCI Configuration Register
6. BA refers to the CrossBar Control Register



 

# 4. CPU Description

The VORTEX86EX2 is a 32-bit highly integrated SoC with 6-stage pipeline. It provides the ideal solution with low power consumption for embedded system integration. The following sections provide more detail on the sub-functions of the SoC.

## 4.1. SoC Core

The SoC integrates a high speed and high performance CPU core that is designed on advanced 32-bit, 6-stage pipeline architecture. The CPU core of SoC implements an MMU (Memory Management Unit).

### 4.1.1. Bus Unit

The bus unit manages data transformations, instruction prefetches and controls functions between the processor's internal units and the SoC peripheral. Internally, the bus unit communicates with the cache and the instruction prefetch units through the 32-bit bus. Externally, the bus unit provides the processor with bus functions, including external bus cycles, memory read/write, instruction fetch, cache line fill, etc.,

### 4.1.2. Prefetch Unit

When the BUS UNIT is not performing bus cycles to execute an instruction, the instruction prefetch unit uses the BUS UNIT to prefetch instructions. By reading instructions before they are needed, the processor rarely needs to wait for an instruction prefetch cycle on the processor bus.

Instruction prefetch cycles read 32-byte blocks of instructions, starting at addresses numerically greater than the last-fetched instruction. The prefetch unit, which has a direct connection to the paging unit, generates the starting address. The 32-byte prefetched blocks are read into both the prefetch and cache units simultaneously. The prefetch queue in the prefetch unit stores 64 bytes of instructions. As each instruction is fetched from the queue, the code part is sent to the instruction decode unit and (depending on the instruction) the displacement part is sent to the segmentation unit, where it is used for address calculation. If loops are encountered in the program being executed, the prefetch unit gets copies of previously executed instructions from the cache.



 

#### 4.1.3. Decode Unit

The instruction decode unit receives instructions from the instruction prefetch unit and translates them in a two-stage process into low-level control signals and microcode entry points. Most instructions can be decoded at a rate of one per clock.

The decode unit simultaneously processes instruction prefix bytes, opcodes, modR/M bytes, and displacements. The outputs include hardwired microinstructions to the segmentation, and integer units. The instruction decode unit is flushed whenever the instruction prefetch unit is flushed.

## 4.2. L1 Cache

In order to maximize the performance, the SoC integrated a 4-way, 16-KByte code and 16-KByte data cache in it. The level 1 cache supports write through policy. The on-chip L1 cache allows frequently used data and code to be stored on chip reducing accesses to the external bus. It significantly reduces the penalty of performance to access these codes and data from external slower memory devices.



 

## 4.3. 32-Bit Memory Addressing Mode

The SoC core provides several addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high-level languages such as C and C++, and they cover the vast majority of data references needed by high-level languages.

### 4.3.1. Register and Immediate Modes

Two of the addressing modes provide instructions that operate on register or immediate operands:

**Register Operand Mode:** The operand is located in one of the 8-, 16- or 32-bit general registers. For example, the ADD instruction adds the EAX and EBX register value and save the result to the EAX register

**Example: ADD EAX, EBX**

**Immediate Operand Mode:** An operand that is directly encoded as part of an instruction is called an **immediate operand**. For example, the MOV instruction moves the immediate value 12345678h (HEX) to the EAX register.

**Example: MOV EAX, 12345678h**



 

### 4.3.2. 32-bit Memory Addressing Modes

The memory addressing modes provide a mechanism for specifying the effective address of an operand. The effective address is calculated by using combinations of the following four address elements:

**Displacement:** An 8-, 16-, or 32-bit immediate value, following the instruction.

**Base:** The contents of any general-purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

**Index:** The contents of any general-purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.

**Scale:** The index register's value can be multiplied by a scale factor: 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these 4 components make up the 9 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions.

The **Effective Address (EA)** of an operand is calculated according to the following formula.

EA = Base + (Index * Scale) + Displacement

| | | |
|---|---|---|
| **Direct Mode** | : | The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement. |
| **Example** | : | **INC Word PTR [50000]** |

| | | |
|---|---|---|
| **Register Indirect Mode** | : | A **Base** register contains the address of the operand. |
| **Example** | : | **MOV [ECX], EDX** |

| | | |
|---|---|---|
| **Based Mode** | : | A **Base** register's contents are added to a **Displacement** to form the operand's offset. |
| **Example** | : | **MOV ECX, [EAX + 24]** |

| | | |
|---|---|---|
| **Index Mode** | : | An **Index** register's contents are added to a **Displacement** to form the operand's offset. |
| **Example** | : | **ADD EAX, TABLE[ESI]** |

| | | |
|---|---|---|
| **Scaled Index Mode** | : | An **Index** register's contents are multiplied by a **Scaling** factor that is added to a **Displacement** to form the operand's offset. |
| **Example** | : | **IMUL EBX, TABLE[ESI • 4], 7** |

| | | |
|---|---|---|
| **Based Index Mode** | : | The contents of a **Base** register are added to the contents of an **Index** register to form the effective address of an operand. |
| **Example** | : | **MOV EAX, [ESI] [EBX]** |

| | | |
|---|---|---|
| **Based Scaled Index Mode** | : | The contents of an **Index** register are multiplied by a **Scaling** factor and the result is added to the contents of a **Base** register to obtain the operand's offset. |
| **Example** | : | **MOV ECX, [EDX • 8] [EAX]** |

| | | |
|---|---|---|
| **Based Index Mode with Displacement** | : | The contents of an **Index** register and a **Base** register's contents and a **Displacement** are all summed together to form the operand offset. |
| **Example** | : | **ADD EDX, [ESI] [EBP + 00FFFFF0H]** |

| | | |
|---|---|---|
| **Based Scaled Index Mode with Displacement** | : | The contents of an **Index** register are multiplied by a **Scaling** factor and the result is added to the contents of a **Base** register and a **Displacement** to form the operand's offset. |
| **Example** | : | **MOV EAX, LOCALTABLE [EDI • 4] [EBP + 80]** |

### 4.3.3. 32-bit Addressing Map

In the 32-bit addressing mode, the physical memory addresses range from 0000_0000h to FFFF_FFFFh (4 Gbytes).

- CPU start address at 0FFF_FFFF0h after reset
- Interrupt Vector Table with 256 interrupts at 0000_0000h to 0000_03FFh (1 Kbyte)

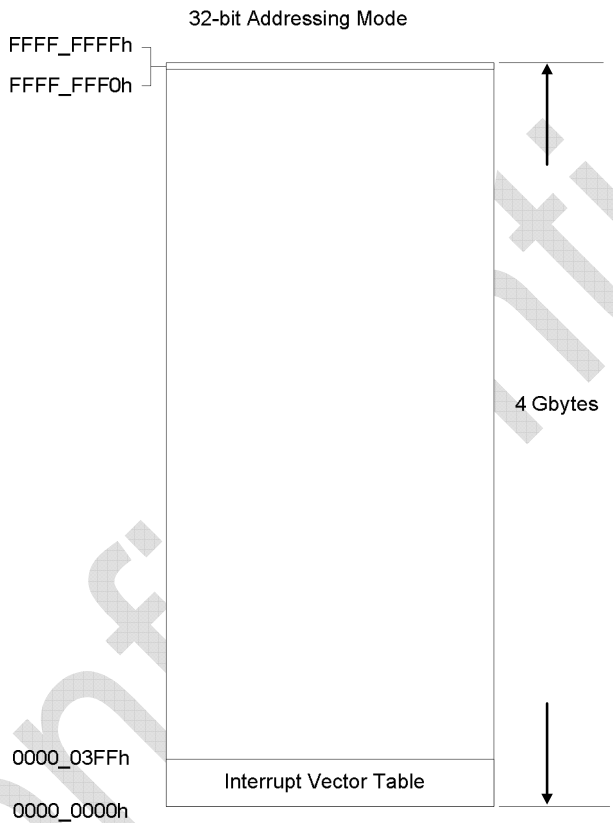


**Figure 4-1. 32-bit Addressing Map and Interrupt Vector Table**



 

### 4.3.4. Interrupt Vector Table

- Interrupt Vector Table with 256 interrupts at 0000_0000h~0000_03FFh (1 Kbyte)
- Each interrupt has 4-byte space to store Interrupt Service Routine (ISR) entry point

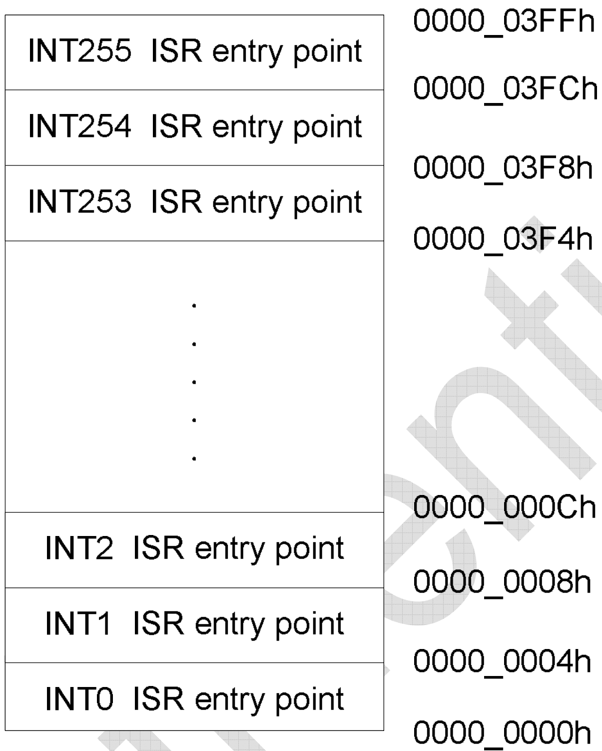


**Figure 4-2. Interrupt Vector Table**



 

### 4.3.5. Interrupt & Exception Flow

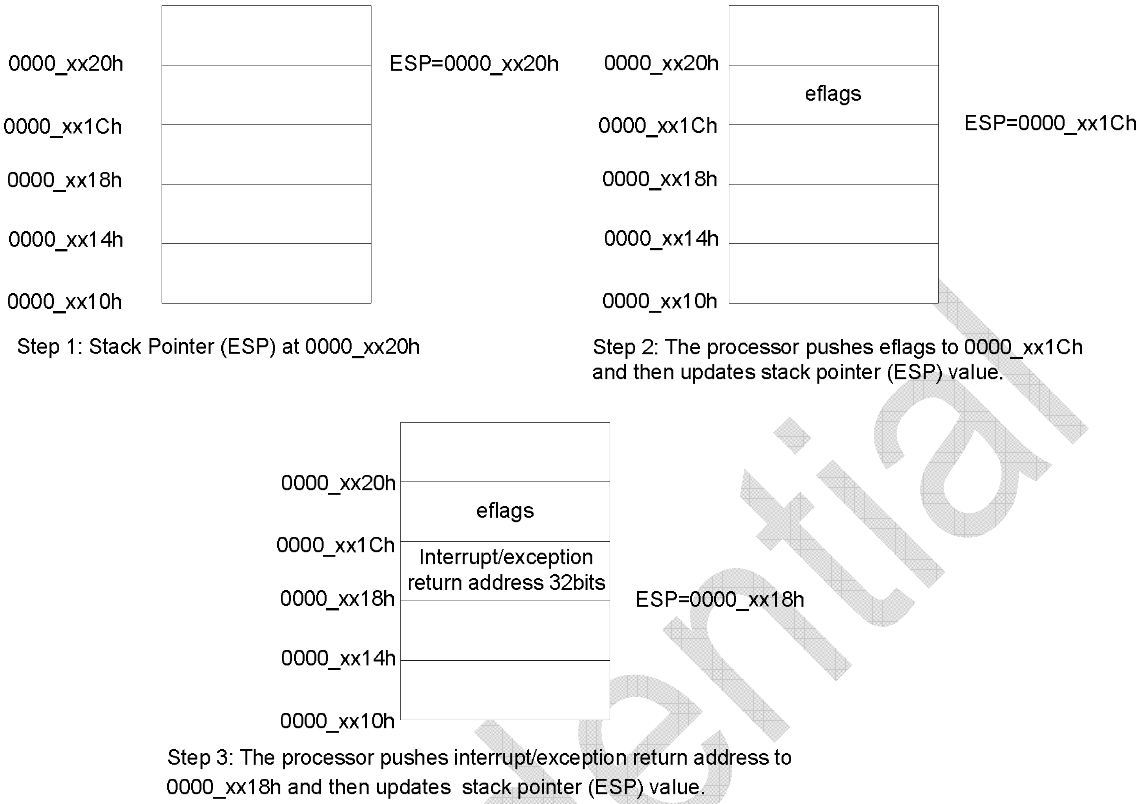


**Figure 4-3. Interrupt & Exception Flow**

PUSH/POP Behavior

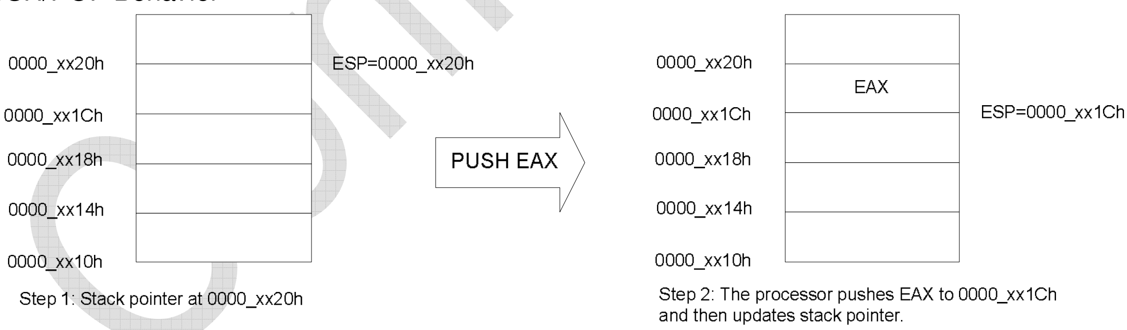


**Figure 4-4. PUSH/POP Behavior**



 

### 4.3.6. Exception Type

| Exception | Vector | Condition |
|---|---|---|
| Division by zero | 0 | Attempting to execute a DIV or an IDIV instruction with a divisor which equals zero. |
| Breakpoint | 3 | A Breakpoint exception occurs when an INT3 instruction is executed. The INT3 is normally used by debug software to set instruction breakpoints by replacing instruction-opcode bytes with the INT3 opcode. |
| Overflow | 4 | It indicates that an overflow trap occurred when an INTO instruction was executed. The INTO instruction checks the state of the OF flag in the EFLAGS register. If the OF flag is set, an overflow trap is generated. |
| Bounds check | 5 | A bound check exception can occur as a result of executing the BOUND instruction. The BOUND instruction compares an array index with the lower bounds and upper bounds of an array. If the array index is not within the array boundary, the bound check exception occurs |
| Invalid opcode | 6 | An invalid opcode exception occurs when an attempt is made to execute an invalid or undefined opcode. |
| Floating-point error | 16 | Indicates FPU has detected a floating-point error conditions:<br>Divide-by-Zero, Underflow, Overflow....etc. |



 