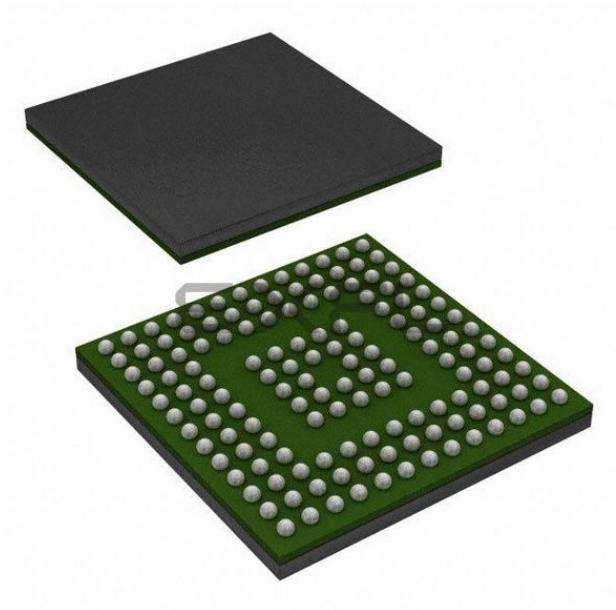


Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Active
Applications	Keyboard and Embedded Controller
Core Processor	ARM® Cortex®-M4
Program Memory Type	-
Controller Series	MEC170x
RAM Size	256KB
Interface	ACPI, EBI/EMI, eSPI, I²C, LPC, PECI, PS/2, QSPI, SPI
Number of I/O	123
Voltage - Supply	1.71V ~ 3.465V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	144-WFBGA
Supplier Device Package	144-WFBGA (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/mec1701h-c1-sz-tr

Agg IRQ	Agg Bit	HWB Instance Name	Interrupt Event	Wake Event	Source Description	Agg NVIC	Direct NVIC
GIRQ10	0	GPIO040	GPIO Event	Yes	GPIO Interrupt Event	2	N/A
	1	GPIO041	GPIO Event	Yes	GPIO Interrupt Event		
	2	GPIO042	GPIO Event	Yes	GPIO Interrupt Event		
	3	GPIO043	GPIO Event	Yes	GPIO Interrupt Event		
	4	GPIO044	GPIO Event	Yes	GPIO Interrupt Event		
	5	GPIO045	GPIO Event	Yes	GPIO Interrupt Event		
	6	GPIO046	GPIO Event	Yes	GPIO Interrupt Event		
	7	GPIO047	GPIO Event	Yes	GPIO Interrupt Event		
	8	GPIO050	GPIO Event	Yes	GPIO Interrupt Event		
	9	GPIO051	GPIO Event	Yes	GPIO Interrupt Event		
	10	GPIO052	GPIO Event	Yes	GPIO Interrupt Event		
	11	GPIO053	GPIO Event	Yes	GPIO Interrupt Event		
	12	GPIO054	GPIO Event	Yes	GPIO Interrupt Event		
	13	GPIO055	GPIO Event	Yes	GPIO Interrupt Event		
	14	GPIO056	GPIO Event	Yes	GPIO Interrupt Event		
	15	GPIO057	GPIO Event	Yes	GPIO Interrupt Event		
	16	GPIO060	GPIO Event	Yes	GPIO Interrupt Event		
	17	GPIO061	GPIO Event	Yes	GPIO Interrupt Event		
	18	GPIO062	GPIO Event	Yes	GPIO Interrupt Event		
	19	GPIO063	GPIO Event	Yes	GPIO Interrupt Event		
	20	GPIO064	GPIO Event	Yes	GPIO Interrupt Event		
	21	GPIO065	GPIO Event	Yes	GPIO Interrupt Event		
	22	GPIO066	GPIO Event	Yes	GPIO Interrupt Event		
	23	GPIO067	GPIO Event	Yes	GPIO Interrupt Event		
	24	GPIO070	GPIO Event	Yes	GPIO Interrupt Event		
	25	GPIO071	GPIO Event	Yes	GPIO Interrupt Event		
	26	GPIO072	GPIO Event	Yes	GPIO Interrupt Event		
	27	GPIO073	GPIO Event	Yes	GPIO Interrupt Event		
	28	Reserved	-	-			
	29	Reserved	-	-			
	30	Reserved	-	-			
	31	Reserved					

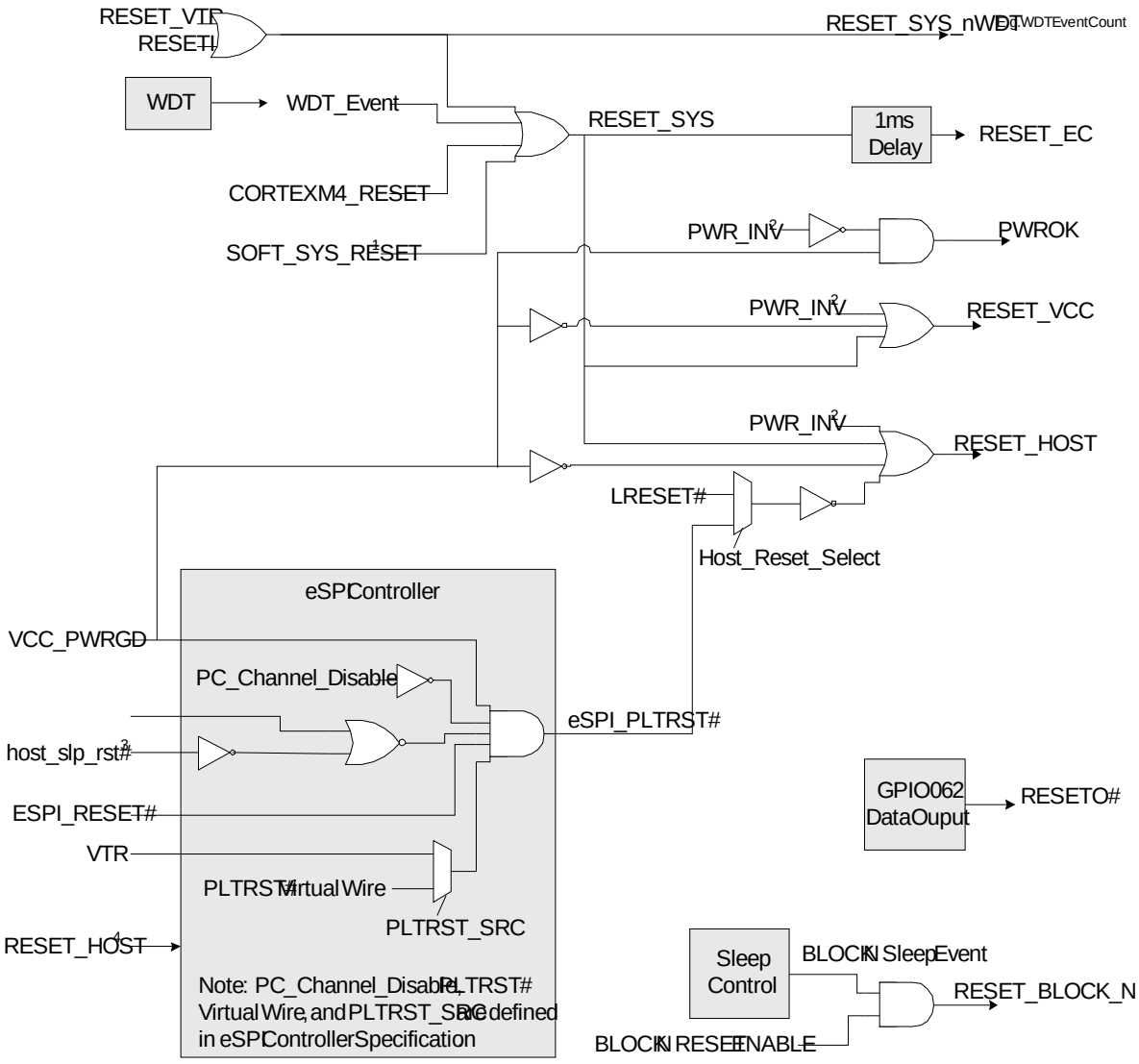
TABLE 4-7: DEFINITION OF RESET SIGNALS (CONTINUED)

Reset	Description	Source
RESET_VTR	Internal VTR Reset signal.	This internal reset signal is asserted as long as the reset generator determines that the output of the internal regulator is stable at its target voltage and that the voltage rail supplying the main clock PLL is at 3.3V. Although most VTR-powered registers are reset on RESET_SYS, some registers are only reset on this reset.
RESET_SYS	Internal Reset signal. This signal is used to reset VTR powered registers.	RESET_SYS is the main global reset signal. This reset signal will be asserted if: • RESET_VTR is asserted • The RESETI# pin asserted • A WDT Event event is asserted • A soft reset is asserted by the SOFT_SYS_RESET bit in the System Reset Register • ARM M4 SYSRESETREQ
RESET_eSPI	System reset signal connected to the eSPI ESPI_RESET# pin.	Pin Interface, ESPI_RESET# pin.
RESET_VCC	Performs a reset when Host power (VCC) is turned off	This signal is asserted if • RESET_SYS is asserted • VCC_PWRGD is low • The PWR_INV bit in the Power Reset Control Register is ‘1b’ The PWROK output pin is an inverted version of this reset; it is asserted when VCC_PWRGD is high and the PWR_INV bit is ‘0b’. **Note:** This reset is referred to as RESET_SIO in the eSPI Block Specification.
RESET_HOST	Performs a reset when VCC_PWRGD is low or when the system host resets the Host Interface.	This signal is asserted if • RESET_SYS is asserted • VCC_PWRGD is low • The PWR_INV bit in the Power Reset Control Register is ‘1b’ • The HOST_RESET_SELECT bit in the Power Reset Control Register is configured for LRESET# and the LRESET# signal is asserted • The HOST_RESET_SELECT bit in the Power Reset Control Register is configured for eSPI_PLTRST# and the eSPI_PLTRST# signal from the eSPI block is asserted.
WDT Event	A WDT Event generates the RESET_SYS event. This signal resets VTR powered registers with the exception of the WDT Event Count Register register. Note that the glitch protect circuits do not activate on a WDT reset. WDT Event does not reset VBAT registers or logic.	This reset signal will be asserted if: • A WDT Event event is asserted This event is indicated by the WDT bit in the Power-Fail and Reset Status Register

TABLE 4-7: DEFINITION OF RESET SIGNALS (CONTINUED)

Reset	Description	Source
RESET_SYS_nWDT	Internal Reset signal. This signal is used to reset VTR powered registers not effected by a WDT Event A RESET_SYS_nWDT is used to reset registers that need to be preserved through a WDT Event like a WDT Event Count Register.	This reset signal will be asserted if: • RESET_VTR is asserted • The RESETI# pin asserted
RESET_EC	Internal reset signal to reset the processor in the EC Subsystem.	This reset is a stretched version of RESET_SYS. This reset asserts at the same time that RESET_SYS asserts and is held asserted for 1ms after RESET_SYS deasserts.
RESET_BLOCK_N	Each IP block in the device may be configured to be reset when it enters the Low Power Mode referred to as SLEEP.	This reset signal will be asserted if Block N SLEEP_ENABLE and Block N RESET_ENABLE are all set to 1, Block N CLOCK_REQUIRED signal is low, and Block N Enters Sleep.

FIGURE 4-4: RESETS BLOCK DIAGRAM



Note 1: SOFT_SYS_RESET is implemented in bit[8] of the System Reset Register
Note 2: PWR_INV is implemented in bit[0] of the Power Reset Control Register
Note 3: host_slp_rst# is asserted if the Host Sleep Enable bit and Host Reset Enable bit are set and the eSPI Controller is put to sleep.
Note 4: RESET_HOST is fed back into the eSPI Controller to reset select register bits

The LPC Interface detects traffic on the bus and requires the clock to be on to process the incoming data. If the LPC_WAKE_ONLY interrupt in GIRQ22 is enabled, the LPC block will be able to autonomously receive data for the programmed I/O ranges without processor intervention. Once the data is loaded into the HOST-to-EC Mailbox Register the Host-to-EC IRQ will trigger an interrupt to the embedded controller to service this command.

4.8 EC Registers

Registers for this block are shown in the following summary table. Addresses for each register are determined by adding the offset to the Base Address for the Power, Clocks, and Resets Block in the Block Overview and Base Address Table in Section 3.0, "Device Inventory".

TABLE 4-9: REGISTER SUMMARY

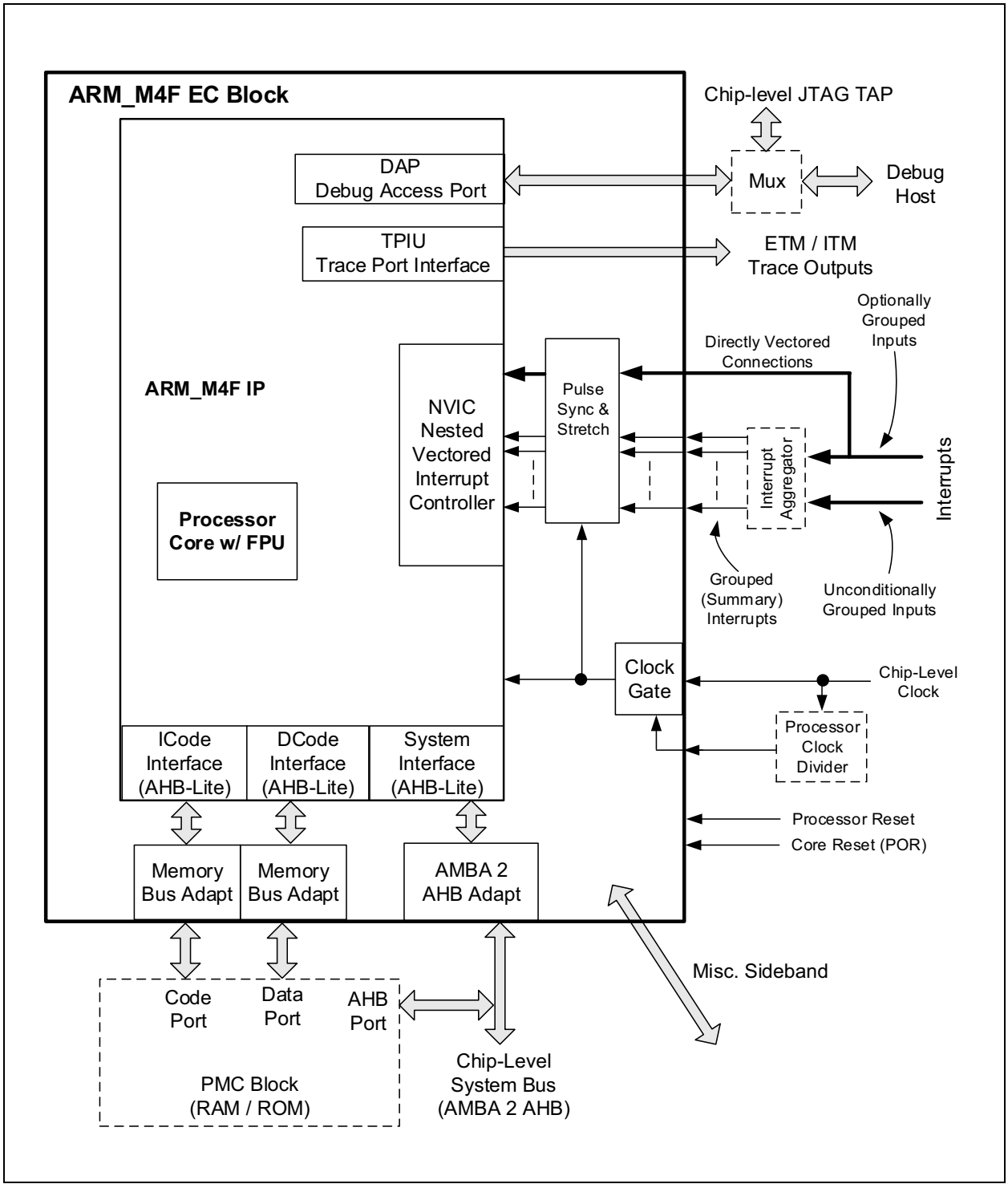
Offset	Name
0h	System Sleep Control Register
4h	Processor Clock Control Register
8h	Slow Clock Control Register
Ch	Oscillator ID Register
10h	PCR Power Reset Status Register
14h	Power Reset Control Register
18h	System Reset Register
1Ch	TEST
20h	TEST
30h	Sleep Enable 0 Register
34h	Sleep Enable 1 Register
38h	Sleep Enable 2 Register
3Ch	Sleep Enable 3 Register
40h	Sleep Enable 4 Register
50h	Clock Required 0 Register
54h	Clock Required 1 Register
58h	Clock Required 2 Register
5Ch	Clock Required 3 Register
60h	Clock Required 4 Register
70h	Reset Enable 0 Register
74h	Reset Enable 1 Register
78h	Reset Enable 2 Register
7Ch	Reset Enable 3 Register
80h	Reset Enable 4 Register

All register addresses are naturally aligned on 32-bit boundaries. Offsets for registers that are smaller than 32 bits are reserved and must not be used for any other purpose.

The bit definitions for the Sleep Enable, Clock Required and Reset Enable Registers are defined in the Sleep Enable Register Assignments Table in Section 3.0, "Device Inventory".

5.5 Block External Interfaces

FIGURE 5-1: ARM M4F BASED EMBEDDED CONTROLLER I/O BLOCK DIAGRAM



Offset	10h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
15:9	HARDWARE_FLOW_CONTROL_DEVICE This is the device that is connected to this channel as its Hardware Flow Control master. The Flow Control Interface is a bus with each master concatenated onto it. This selects which bus index of the concatenated Flow Control Interface bus is targeted towards this channel.	R/W	0h	RESET
8	TRANSFER_DIRECTION This determines the direction of the DMA Transfer. 1=Data Packet Read from MEMORY_START_ADDRESS followed by Data Packet Write to DEVICE_ADDRESS 0=Data Packet Read from DEVICE_ADDRESS followed by Data Packet Write to MEMORY_START_ADDRESS	R/W	0h	RESET
7:6	Reserved	R	-	-
5	BUSY This is a status signal. 1=The DMA Channel is busy (FSM is not IDLE) 0=The DMA Channel is not busy (FSM is IDLE)	R	0h	RESET
4:3	TEST	R	0h	RESET
2	DONE This is a status signal. It is only valid while RUN is Enabled. This is the inverse of the **DMA Channel Control:Busy** field, except this is qualified with the **DMA Channel Control:Run** field. 1=Channel is done 0=Channel is not done or it is OFF	R	0h	RESET
1	REQUEST This is a status field. 1=There is a transfer request from the Master Device 0=There is no transfer request from the Master Device	R	0h	RESET
0	RUN This is a control field. It only applies to **Hardware Flow Control** mode. 1=This channel is enabled and will service transfer requests 0=This channel is disabled. All transfer requests are ignored	R/W	0h	RESET

9.8.7.4 EOI/ISR Read Latency

Any serialized IRQ scheme has a potential implementation issue related to IRQ latency. IRQ latency could cause an EOI or ISR Read to precede an IRQ transition that it should have followed. This could cause a system fault. The host interrupt controller is responsible for ensuring that these latency issues are mitigated. The recommended solution is to delay EOIs and ISR Reads to the interrupt controller by the same amount as the SERIRQ Cycle latency in order to ensure that these events do not occur out of order.

9.8.7.5 AC/DC Specification Issue

All Serial IRQ agents must drive/sample SERIRQ synchronously related to the rising edge of LCLK. The SERIRQ pin uses the electrical specification of the PCI bus. Electrical parameters will follow the PCI Local Bus Specification, Rev. 2.2 definition of “sustained tri-state.”

9.8.7.6 Reset and Initialization

The SERIRQ bus uses LRESET# as its reset signal and follows the PCI bus reset mechanism. The SERIRQ pin is tri-stated by all agents while LRESET# is active. With reset, SERIRQ slaves and bridges are put into the (continuous) Idle mode. The host controller is responsible for starting the initial SERIRQ cycle to collect system’s IRQ/Data default values. The system then follows with the Continuous/Quiet mode protocol (Stop Frame pulse width) for subsequent SERIRQ cycles. It is the host controller’s responsibility to provide the default values to the 8259’s and other system logic before the first SERIRQ cycle is performed. For SERIRQ system suspend, insertion, or removal application, the host controller should be programmed into Continuous (IDLE) mode first. This is to ensure the SERIRQ bus is in Idle state before the system configuration changes.

9.8.7.7 SERIRQ Interrupts

The LPC Controller routes Logical Device interrupts onto SIRQ stream frames IRQ[0:15]. Routing is controlled by the SIRQ Interrupt Configuration Registers. There is one SIRQ Interrupt Configuration Register for each accessible SIRQ Frame (IRQ); all 16 registers are listed in Table 9-9, "Configuration Register Summary".

The format for each SIRQ Interrupt Configuration Register is described in Section 9.9.2.1, "SIRQ Configuration Register Format". Each Logical Device can have up to two LPC SERIRQ interrupts. When the device is polled by the host, each SIRQ frame routes the level of the Logical Device interrupt (selected by the corresponding SIRQ Interrupt Configuration Register) to the SIRQ stream.

Each SIRQ Interrupt Configuration Register controls a series of multiplexers which route to a single Logical Device interrupt as illustrated in Figure 9-7, "SIRQ Routing Internal Logical Devices". The following table defines the Serial IRQ routing for each logical device implemented in the chip.

TABLE 9-8: LOGICAL DEVICE SIRQ ROUTING

SIRQ Interrupt Configuration Register		Logical Device Interrupt Source	
Select	**Frame**	**Logical Device (Block Instance)**	**Interrupt Source**
0h	0h	Mailbox	MBX_Host_SIRQ
1h	0h	Mailbox	MBX_Host_SMI
0h	1h	Keyboard Controller (8042)	KIRQ
1h	1h	Keyboard Controller (8042)	MIRQ
0h	2h	ACPI-EC 0	EC_OBE
0h	3h	ACPI-EC 1	EC_OBE
0h	4h	ACPI-EC 2	EC_OBE
0h	5h	ACPI-EC 3	EC_OBE
0h	6h	ACPI-EC 4	EC_OBE
0h	9h	UART 0	UART
0h	Ah	UART 1	UART
0h	Ch	LPC Interface 0	EC_IRQ
0h	10h	EM Interface 0	EC-to-Host