

# Understanding the Integrated Programmable Interrupt Controller (IPIC)

by: Charles Melear  
Application Engineer, Automotive  
Austin, Texas

## 1 Introduction

The purpose of the integrated programmable interrupt controller (IPIC) is to receive interrupt requests from the peripheral modules of a microcontroller, prioritize these interrupts through the use of various programmable registers, and provide the vector number of the current highest priority pending interrupt. Generally speaking, most of the interrupts can be individually masked. In some cases an interrupt from a peripheral module can be programmed to cause different types of interrupts (normal, system management, and critical) to the CPU core.

It is important to know the IPIC is a standard module. A block diagram of the IPIC is shown in [Figure 1](#). The IPIC can receive 128 interrupting sources and prioritize them according to how various registers within the IPIC are programmed. In the present example, using the MPC5121e, only 56 interrupting sources are brought into the IPIC module.

### Contents

1	Introduction . . . . .	1
2	Register Summary . . . . .	17
3	Programming Example Steps . . . . .	20
	3.1 Assign the Relative Priorities of the Interrupts in . . .	20
	3.2 System Global Interrupt Configuration Register . . .	26
4	Using the IPIC Control and Status Registers . . . . .	28
5	Forcing Interrupts . . . . .	29
6	Summary . . . . .	30

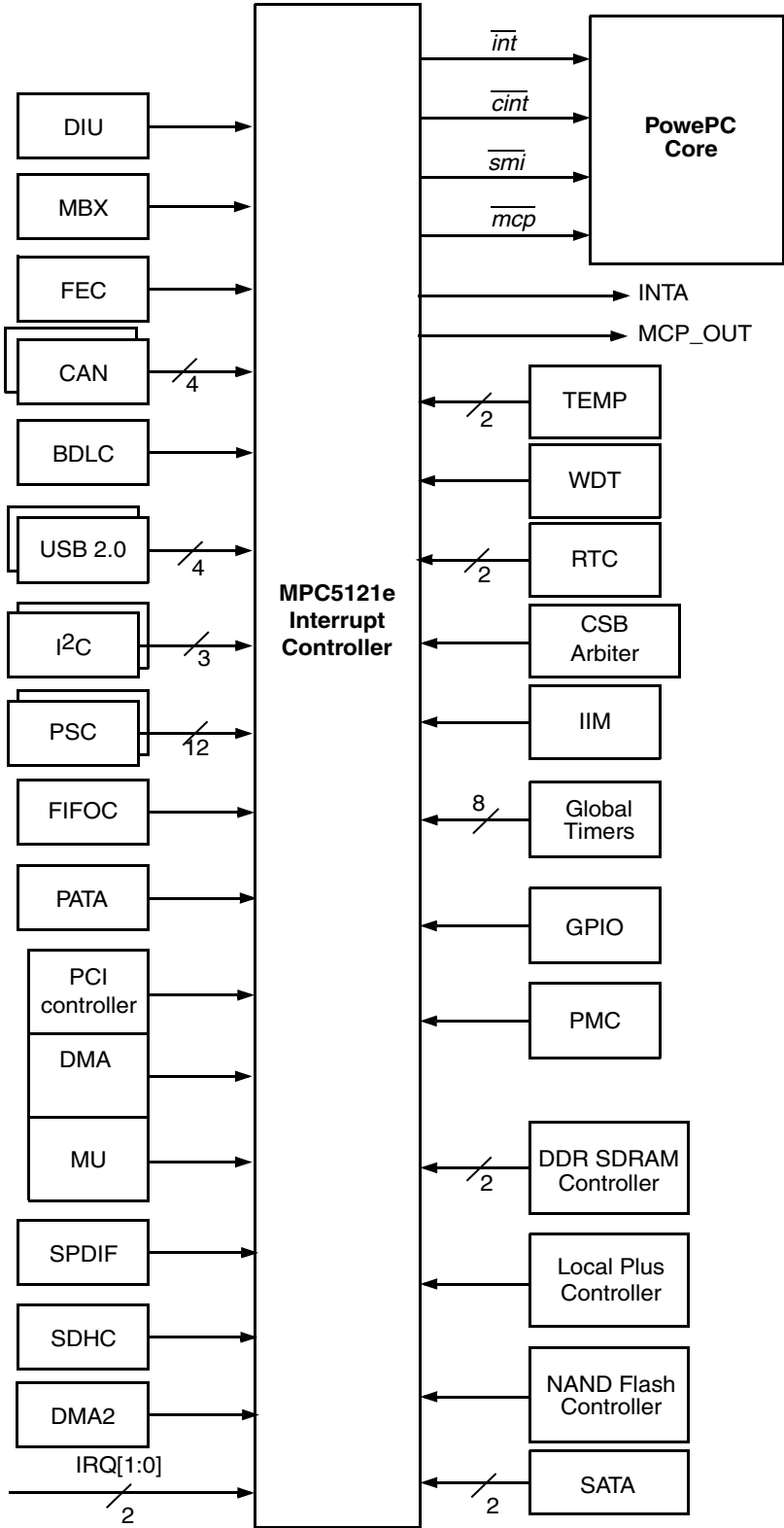


Figure 1. Integrated Peripheral Interrupt Controller

The IPIC assigns some of the various interrupting sources into sets. Other interrupts are individually assigned to fixed priority levels. The sets defined for the IPIC, as used in the MPC5121e are MIXA, MIXB, SYSA, SYSB, SYSC, and SYSD. Each interrupt source is either assigned to one of these sets or to a fixed interrupt priority level. The assignment of interrupts to the various sets is unique for each device and is specified by the integrated circuit system designer. This application note uses the MPC5121e as an example. The assignment of the interrupting sources to the various sets or to fixed priority levels is at the discretion of the integrated circuit system designer. Therefore, the assignment of the interrupts are unique for each device where the IPIC is used.

Each set can accommodate eight interrupt sources. Within a set, the priority of the eight sources is programmable with respect to the other members of the set. Using the MPC5121e as an example, the members of the various sets are shown in [Table 1](#).

**Table 1. Interrupting Sources per Set**

	MIXA	MIXB	SYSA	SYSB	SYSC	SYSD
0	DIU	PSC0	PSC4	FIFOC	PCI	I2C1
1	DMA	PSC1	PSC5	SPCIF	PCI DMA	I2C2
2	MBX	PSC2	PSC6	AXE	PCI MU	I2C3
3	RESERVED_0	PSC3	PSC7	USB ULPI	FEC	MSCAN1
4	IRQ0	RESERVED_3	PSC8	USB UTMI	PATA	MSCAN2
5	IRQ1	RESERVED_4	PSC9	SATA	NFC	BDLC
6	RESERVED_1	RESERVED_5	PSC10	RESERVED_7	LPC	GPT0
7	RESERVED_2	RESERVED_6	PSC11	RESERVED_8	SDHC	GPT1

A partial block diagram of the IPIC is shown in [Figure 2](#). The six sets, MIX A, MIX B, SYS A, SYS B, SYS C, and SYS D are on the left side of the diagram. Each set is routed to a Grouped and a Spread decoder. Only one decoder for each set can be active at any time. Each of the eight outputs of the Group and Spread decoders are assigned one of 128 priority levels. For example, MIXA0 (Spread) is priority 1. MIXA1 (Spread) is priority 11. MIXA0 – MIXA3 (Grouped) are priorities 2 – 5. The entire priority table is shown in [Table 4](#). A particular set, such as MIXA, can only have active Grouped or Spread entries. If a set is programmed to be Spread, the Grouped entries are inactive and vice versa.

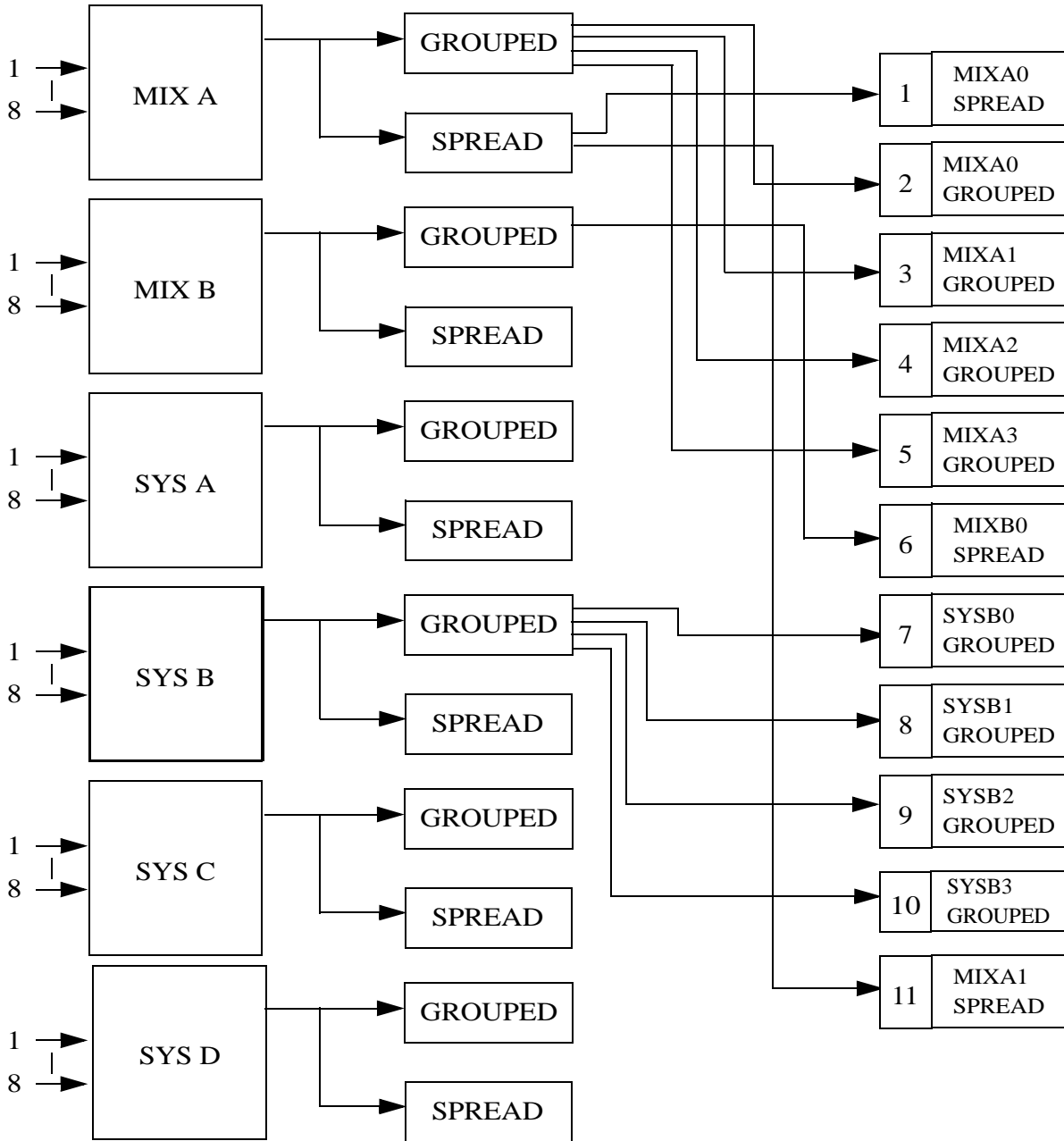


Figure 2. Partial IPIIC Block Diagram

Each set is composed of eight preassigned peripheral interrupt sources. Relative priorities are programmed via the system mixed interrupt group A priority register, the system mixed interrupt group B priority register, the system internal interrupt group A priority register, the system internal interrupt group B priority register, the system internal interrupt group C priority register, and the system internal interrupt group D priority register. In the present example using the MPC5121e, MIXA consists of the DIU, DMA, MBX, Reserved\_0, IRQ0, IRQ1, Reserved\_1, and Reserved\_2.

The sets, MIXA, MIXB, SYSA, SYSB, SYSC, and SYSD remain constant between the various devices where the IPIC module is used. However, the interrupt sources assigned to each set can (almost certainly) vary between the various devices where the IPIC module is used.

Each set has eight possible entries. For instance, the set SYSA has the following fields: SYSA0P, SYSA1P, SYSA2P, SYSA3P, SYSA4P, SYSA5P, SYSA6P, and SYSA7P. The same pattern is true for the other sets listed in [Table 1](#).

[Figure 3](#) shows the system internal interrupt group A priority register (SIPRR\_A) that defines the priority between PSC4, PSC5, PSC6, PSC7, PSC8, PSC9, PSC10, and PSC11 internal interrupt signals. For example, PSC4 can be programmed to any of the eight SYSA priority levels. Likewise, any of the PSCs in the SYSA group can be programmed to any of the eight SYSA priority levels.

#### NOTE

No interrupt source for example, SYSAxP, can be used more than once and each interrupting source (PSCx) must be used, including the reserved entries.

For a particular group, such as MIXA or SYSA, a software selection must be made that only activates a set's Grouped entries or Spread entries. If a set is programmed to be Grouped, then the Spread entries in the priority table are treated as, don't cares. Likewise, if a set is programmed to be SPREAD, then GROUPEd entries in the priority table are treated as, don't cares. Each set can individually be programmed to be SPREAD or GROUPEd.

## Introduction

Offset 0x10Access: User read/write

Power PC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
Conventional	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	SYSA0P			SYSA1P				SYSA2P				SYSA3P				0	0	0	0
W																			
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0			
Power PC	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
Conventional	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	SYSA4P			SYSA5P				SYSA6P				SYSA7P				0	0	0	0
W																			
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0			

= Unimplemented or Reserved

**Figure 3. System Internal Interrupt Group A Priority Register (SIPRR\_A)**

**Table 2. SIPRR\_A Field Descriptions**

Field	Description
SYSA0P - SYSA7P	SYSA0 Priority Order. Defines which interrupt source asserts its request in the (x) priority position. Do not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of SYSA(x)P is shown as follows: 000 PSC4 asserts its request in the SYSA(x) position. 001 PSC5 asserts its request in the SYSA(x) position. 010 PSC6 asserts its request in the SYSA(x) position. 011 PSC7 asserts its request in the SYSA(x) position. 100 PSC8 asserts its request in the SYSA(x) position. 101 PSC9 asserts its request in the SYSA(x) position. 110 PSC10 asserts its request in the SYSA(x) position. 111 PSC11 asserts its request in the SYSA(x) position.
	Each field in this register must have a unique value. This means all 8 codes must be used. It is a programming error to use any of the 3-bit codes multiple times.

Table 3 shows the interrupt vector numbers for each of the interrupting sources. This table is unique to the MPC5121e. Other microcontrollers that use the IPIC module may, and probably assigns different interrupt vector numbers to each or some of the interrupting sources. The interrupt vector number is not to be confused with the interrupt priority number. For the MPC5121e, the PCI module could be any one of eight priority levels within system group C. However, no matter what the interrupt priority level is for the PCI module, it always returns an interrupt vector number of 0x000\_0001. The interrupt vector number is hard-coded into the interrupting peripheral element. The interrupt vector number may change between different microcontrollers where the IPIC module is used.

**Table 3. IVEC Field Values (Sheet 1 of 3)**

Peripheral	Interrupt Vector Number (binary)	Interrupt Vector Number (decimal)	Default Group Programming
Error (No Interrupt)	0x000_0000	0	—
PCI	0x000_0001	1	SYSC0(Grouped)
PCI DMA	0x000_0010	2	SYSC1(Grouped)
PCI MU	0x000_0011	3	SYSC2(Grouped)
FEC	0x000_0100	4	SYSC3(Grouped)
PATA	0x000_0101	5	SYSC4(Grouped)
NFC	0x000_0110	6	SYSC5(Grouped)
LPC	0x000_0111	7	SYSC6(Grouped)
SDHC	0x000_1000	8	SYSC7(Grouped)
I2C1	0x000_1001	9	SYSD0(Grouped)
I2C2	0x000_1010	10	SYSD1(Grouped)
I2C3	0x000_1011	11	SYSD2(Grouped)
MSCAN1	0x000_1100	12	SYSD3(Grouped)
MSCAN2	0x000_1101	13	SYSD4(Grouped)
BDLC	0x000_1110	14	SYSD5(Grouped)
GPT0	0x000_1111	15	SYSD6(Grouped)
GPT1	0x001_0000	16	SYSD7(Grouped)
IRQ1	0x001_0001	17	MIXA5(Grouped)
Reserved	0x001_0010	18	MIXA6(Grouped)
Reserved	0x001_0011	19	MIXA7(Grouped)
Reserved	0x001_0100	20	MIXB4(Grouped)
Reserved	0x001_0101	21	MIXB5(Grouped)
Reserved	0x001_0110	22	MIXB6(Grouped)
Reserved	0x001_0111	23	MIXB7(Grouped)
Reserved	0x001_1000	24	Fixed priority
Reserved	0x001_1001	25	Fixed priority
Reserved	0x001_1010	26	Fixed priority
Reserved	0x001_1011	27	Fixed priority
Reserved	0x001_1100	28	Fixed priority
Reserved	0x001_1101	29	Fixed priority
Reserved	0x001_1110	30	Fixed priority
Reserved	0x001_1111	31	Fixed priority
PSC4	0x010_0000	32	SYSA0(Grouped)

**Table 3. IVEC Field Values (Sheet 2 of 3) (continued)**

Peripheral	Interrupt Vector Number (binary)	Interrupt Vector Number (decimal)	Default Group Programming
PSC5	0x010_0001	33	SYSA1(Grouped)
PSC6	0x010_0010	34	SYSA2(Grouped)
PSC7	0x010_0011	35	SYSA3(Grouped)
PSC8	0x010_0100	36	SYSA4(Grouped)
PSC9	0x010_0101	37	SYSA5(Grouped)
PSC10	0x010_0110	38	SYSA6(Grouped)
PSC11	0x010_0111	39	SYSA7(Grouped)
FIFOC	0x010_1000	40	SYSB0(Grouped)
SPDIF	0x010_1001	41	SYSB1(Grouped)
AXE	0x010_1010	42	SYSB2(Grouped)
USB ULPI	0x010_1011	43	SYSB3(Grouped)
USB UTMI	0x010_1100	44	SYSB4(Grouped)
SATA	0x010_1101	45	SYSB5(Grouped)
reserved	0x010_1110	46	SYSB6(Grouped)
reserved	0x010_1111	47	SYSB7(Grouped)
IRQ0	0x011_0000	48	MIXA4(Grouped)
Reserved	0x011_0001–0x011_1111	49–63	Reserved
DIU	0x100_0000	64	MIXA0
DMA2	0x100_0001	65	MIXA1
MBX	0x100_0010	66	MIXA2
Reserved	0x100_0011	67	MIXA3
PSC0	0x100_0100	68	MIXB0
PSC1	0x100_0101	69	MIXB1
PSC2	0x100_0110	70	MIXB2
PSC3	0x100_0111	71	MIXB3
GPT2	0x100_1000	72	Fixed priority
GPT3	0x100_1001	73	Fixed priority
GPT4	0x100_1010	74	Fixed priority
GPT5	0x100_1011	75	Fixed priority
GPT6	0x100_1100	76	Fixed priority
GPT7	0x100_1101	77	Fixed priority
GPIO	0x100_1110	78	Fixed priority
RTC SEC	0x100_1111	79	Fixed priority



**Table 3. IVEC Field Values (Sheet 3 of 3) (continued)**

Peripheral	Interrupt Vector Number (binary)	Interrupt Vector Number (decimal)	Default Group Programming
RTC ALARM	0x101_0000	80	Fixed priority
DDR	0x101_0001	81	Fixed priority
SBA	0x101_0010	82	Fixed priority
PMC	0x101_0011	83	Fixed priority
USB ULPI WKUP	0x101_0100	84	Fixed priority
USB UTMI WKUP	0x101_0101	85	Fixed priority
SATA CMD	0x101_0110	86	Fixed priority
TEMP 105C	0x101_0111	87	Fixed priority
IIM	0x101_1000	88	Fixed priority
PRIOMON	0x101_1001	89	Fixed priority
MSCAN3	0x101_1010	90	Fixed priority
MSCAN4	0x101_1011	91	Fixed priority
Reserved	0x101_1100	92	Fixed priority
Reserved	0x101_1101	93	Fixed priority
Reserved	0x101_1110	95	Fixed priority
Reserved	0x101_1111	95	Fixed priority

The purpose of [Table 4](#) is to show the priority of each set and each member within a set. Each set can be designated as Spread or Grouped. Some interrupting sources have a fixed priority. The MIX and SYS sets can be individually designated as either Grouped or Spread. If a particular set is designated as Grouped, the Spread entries for that set are not recognized and vice versa. Also, as a programming note within each set, each member of the group must be assigned a unique priority. Even though it is possible to do so, it is a programming violation to assign the same module within a set to two or more different priority levels. All reserved entries in a set must be used and given a unique priority in the set's priority register.

**Table 4. Grouped, Spread and Fixed Priority Assignments**

Priority Level	Interrupt Source Description	Multiple Events
0	Highest	—
1	MIXA0 (Spread)	Yes (No for ext. interrupts)
2	MIXA0 (Grouped)	Yes (No for ext. interrupts)
3	MIXA1 (Grouped)	Yes (No for ext. interrupts)
4	MIXA2 (Grouped)	Yes (No for ext. interrupts)
5	MIXA3 (Grouped)	Yes (No for ext. interrupts)
6	MIXB0 (Spread)	Yes (No for ext. interrupts)
7	SYSB0 (Grouped)	Yes
8	SYSB1 (Grouped)	Yes
9	SYSB2 (Grouped)	Yes
10	SYSB3 (Grouped)	Yes
11	MIXA1 (Spread)	Yes (No for ext. interrupts)
12	SYSB4 (Grouped)	Yes
13	SYSB5 (Grouped)	Yes
14	SYSB6 (Grouped)	Yes
15	SYSB7 (Grouped)	Yes
16	MIXB0 (Grouped)	Yes (No for ext. interrupts)
17	MIXB1 (Grouped)	Yes (No for ext. interrupts)
18	MIXB2 (Grouped)	Yes (No for ext. interrupts)
19	MIXB3 (Grouped)	Yes (No for ext. interrupts)
20	MIXB1 (Spread)	Yes (No for ext. interrupts)
21	SYSA0 (Grouped)	Yes
22	SYSA1 (Grouped)	Yes
23	SYSA2 (Grouped)	Yes
24	SYSA3 (Grouped)	Yes
25	MIXA2 (Spread)	Yes (No for ext. interrupts)
26	SYSA4 (Grouped)	Yes
27	SYSA5 (Grouped)	Yes
28	SYSA6 (Grouped)	Yes
29	SYSA7 (Grouped)	Yes
30	MIXA4 (Grouped)	Yes (No for ext. interrupts)
31	MIXA5 (Grouped)	Yes (No for ext. interrupts)
32	MIXA6 (Grouped)	Yes (No for ext. interrupts)

**Table 4. Grouped, Spread and Fixed Priority Assignments (continued)**

Priority Level	Interrupt Source Description	Multiple Events
33	MIXA7 (Grouped)	Yes (No for ext. interrupts)
34	MIXB2 (Spread)	Yes (No for ext. interrupts)
35	SYSC0 (Grouped)	Yes
36	SYSC1 (Grouped)	Yes
37	SYSC2 (Grouped)	Yes
38	SYSC3 (Grouped)	Yes
39	MIXA3 (Spread)	Yes (No for ext. interrupts)
40	SYSC4 (Grouped)	Yes
41	SYSC5 (Grouped)	Yes
42	SYSC6 (Grouped)	Yes
43	SYSC7 (Grouped)	Yes
44	MIXB4 (Grouped)	Yes (No for ext. interrupts)
45	MIXB5 (Grouped)	Yes (No for ext. interrupts)
46	MIXB6 (Grouped)	Yes (No for ext. interrupts)
47	MIXB7 (Grouped)	Yes (No for ext. interrupts)
48	MIXB3 (Spread)	Yes (No for ext. interrupts)
49	SYSD0 (Grouped)	Yes
50	SYSD1 (Grouped)	Yes
51	SYSD2 (Grouped)	Yes
52	SYSD3 (Grouped)	Yes
53	MIXA4 (Spread)	Yes (No for ext. interrupts)
54	SYSD4 (Grouped)	Yes
55	SYSD5 (Grouped)	Yes
56	SYSD6 (Grouped)	Yes
57	SYSD7 (Grouped)	Yes
58	MIXB4 (Spread)	Yes (No for ext. interrupts)
59	GPT2	Yes
60	SYSB0 (Spread)	Yes
61	SYSA0 (Spread)	Yes
62	GPT3	Yes
63	SYSC0 (Spread)	Yes
64	SYSD0 (Spread)	Yes
65	Reserved	No

**Table 4. Grouped, Spread and Fixed Priority Assignments (continued)**

Priority Level	Interrupt Source Description	Multiple Events
66	GPT4	Yes
67	MIXA5 (Spread)	Yes (No for ext. interrupts)
68	GPT5	Yes
69	SYSB1 (Spread)	Yes
70	SYSA1 (Spread)	Yes
71	GPT6	Yes
72	SYSC1 (Spread)	Yes
73	SYSD1 (Spread)	Yes
74	Reserved	No
75	GPT7	Yes
76	MIXB5 (Spread)	Yes (No for ext. interrupts)
77	GPIO	Yes
78	SYSB2 (Spread)	Yes
79	SYSA2 (Spread)	Yes
80	RTC SEC	Yes
81	SYSC2 (Spread)	Yes
82	SYSD2 (Spread)	Yes
83	Reserved	No
84	RTC ALARM	Yes
85	MIXA6 (Spread)	Yes (No for ext. interrupts)
86	DDR	Yes
87	SYSB3 (Spread)	Yes
88	SYSA3 (Spread)	Yes
89	SBA	Yes
90	SYSC3 (Spread)	Yes
91	SYSD3 (Spread)	Yes
92	Reserved	No
93	PMC	Yes
94	MIXB6 (Spread)	Yes (No for ext. interrupts)
95	USB ULPI WKUP	Yes
96	SYSB4 (Spread)	Yes
97	SYSA4 (Spread)	Yes

**Table 4. Grouped, Spread and Fixed Priority Assignments (continued)**

Priority Level	Interrupt Source Description	Multiple Events
98	USB UTMI WKUP	Yes
99	SYSC4 (Spread)	Yes
100	SYSD4 (Spread)	Yes
101	Reserved	No
102	SATA CMD	Yes
103	MIXA7 (Spread)	Yes (No for ext. interrupts)
104	TEMP 105C	Yes
105	SYSB5 (Spread)	Yes
106	SYSA5 (Spread)	Yes
107	IIM	Yes
108	SYSC5 (Spread)	Yes
109	SYSD5 (Spread)	Yes
110	Reserved	No
111	PRIOMON	Yes
112	MIXB7 (Spread)	Yes (No for ext. interrupts)
113	MSCAN3	Yes
114	SYSB6 (Spread)	Yes
115	SYSA6 (Spread)	Yes
116	MSCAN4	Yes
117	SYSC6 (Spread)	Yes
118	SYSD6 (Spread)	Yes
119	Reserved	No
120	Reserved	Yes
121	Reserved	Yes
122	SYSB7 (Spread)	Yes
123	SYSA7 (Spread)	Yes
124	Reserved	Yes
125	SYSC7 (Spread)	Yes
126	SYSD7 (Spread)	Yes
127	Reserved	No
128	Reserved	Yes

The MIXA set consists of eight entries, three of which are reserved. The entries are DIU, DMA, MBX, RESERVED\_0, IRQ0, IRQ1, RESERVED\_1, and RESERVED\_2.

## Introduction

The SMPRR\_A defines the priority between DIU, DMA2, MBX, Reserved\_0, IRQ0, IRQ1, Reserved\_1, and Reserved\_2. See [Figure 4](#).

Offset 0x30Access: User read/write

Power PC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
Conventional	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
R	MIXA0P				MIXA1P				MIXA2P				MIXA3P				0	0	0	0
W																				
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0				
Power PC	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
Conventional	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
R	MIXA4P				MIXA5P				MIXA6P				MIXA7P				0	0	0	0
W																				
Reset	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0				

= Unimplemented or Reserved

**Figure 4. System Mixed Interrupt Group A Priority Register (SMPRR\_A)**

**Table 5. SMPRR\_A Field Descriptions**

Field	Description
MIXA0P	MIXA0 Priority order. Defines which interrupt source asserts its request in the MIXA0 priority position. Do not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXA0P is as follows: 000 DIU asserts its request to the MIXA0 position. 001 DMA2 asserts its request to the MIXA0 position. 010 MBX asserts its request to the MIXA0 position. 011 Reserved asserts its request to the MIXA0 position. 100 IRQ0 asserts its request to the MIXA0 position. This field for MIXA0 position is valid (must not be ignored) if IRQ0 signal configured as an external maskable interrupt (SEMSR[SIRQ0] = 0). 101 IRQ1 asserts its request to the MIXA0 position. 110 Reserved asserts its request to the MIXA0 position. 111 Reserved asserts its request to the MIXA0 position.
MIXA1P–MIXA7P	Same as MIXA0P, but for MIXA1P–MIXA7P.

By default, the system mixed interrupt group A priority register (SMPRR\_A) is programmed such that MIXA0P – MIXA7P fields are programmed to DIU, DMA, MBX, RESERVED\_0, IRQ0, IRQ1, RESERVED\_1, and RESERVED\_2, respectively. The relative priorities of this set can be changed. The only requirement is each field must be written with a unique value. This means all eight values must be used.

For the present example assume the MIXA set is programmed to be in the GROUPED mode and the default value is being used for the system mixed interrupt group A priority register. The resulting priorities are shown in [Table 6](#).

**Table 6. Priority Assignments for System Interrupt Group A Mixed**

Priority Level	Interrupt Source Description
0	Highest
2	MIXA0 (Grouped)
3	MIXA1 (Grouped)
4	MIXA2 (Grouped)
5	MIXA3 (Grouped)
30	MIXA4 (Grouped)
31	MIXA5 (Grouped)
32	MIXA6 (Grouped)
33	MIXA7 (Grouped)

When the MIXA set is programmed as Grouped, the MIXA0 – MIXA7 fields are assigned interrupt priority numbers of 2, 3, 4, 5, 30, 31, 32, and 33. When a set is programmed to be Grouped, the Spread entries are not used and are treated as, don't cares.

When the MIXA set is programmed as Spread, the MIXA0 – MIXA7 fields are assigned interrupt priority numbers of 1, 11, 25, 39, 53, 67, 85, and 103 as shown in [Table 7](#). When a set is programmed to be Spread, the Grouped entries are not used and are treated as, don't cares.

**Table 7. Priority Assignments for System Interrupt Group A Spread**

Priority Level	Interrupt Source Description
0	Highest
1	MIXA0 (Spread)
11	MIXA1 (Spread)
25	MIXA2 (Spread)
39	MIXA3 (Spread)

**Table 7. Priority Assignments for System Interrupt Group A Spread (continued)**

Priority Level	Interrupt Source Description
53	MIXA4 (Spread)
67	MIXA5 (Spread)
85	MIXA6 (Spread)
103	MIXA7 (Spread)

The interrupt priority levels (Table 7) and interrupt vector numbers (Table 3) are totally independent of each other. Interrupt vector numbers are assigned to each peripheral and they do not change. Interrupt priority levels are programmable. To illustrate this point, an example is presented in Table 8, showing interrupt priority numbers and interrupt vector numbers for the system interrupt group A. MIXA5 Spread is interrupt priority number 67. MIXA5 Group is interrupt priority number 31. In both cases, when MIXA5 is programmed to IRQ1, the default programming case, the interrupt vector number is 17.

Notice, IRQ1 could have been programmed to MIXA6 changing the priority number but not the interrupt vector number.

**Table 8. Interrupt Vector Number and Priority Assignments for System Interrupt Group A**

Interrupt Priority Number	Interrupt Vector Number	Meaning	Default Group Programing
67	17	IRQ1	MIXA5(Spread)
31	17	IRQ1	MIXA5(Grouped)
85	18	Reserved	MIXA6(Spread)
32	18	Reserved	MIXA6(Grouped)
103	19	Reserved	MIXA7(Spread)
33	19	Reserved	MIXA7(Grouped)
53	48	IRQ0	MIXA4(Spread)
30	48	IRQ0	MIXA4(Grouped)
	49–63	Reserved	Reserved
1	64	DIU	MIXA0(Spread)
2	64	DIU	MIXA0(Grouped)
11	65	DMA2	MIXA1(Spread)
3	65	DMA2	MIXA1(Grouped)
25	66	MBX	MIXA2(Spread)
4	66	MBX	MIXA2(Grouped)



**Table 8. Interrupt Vector Number and Priority Assignments for System Interrupt Group A (continued)**

Interrupt Priority Number	Interrupt Vector Number	Meaning	Default Group Programing
39	67	Reserved	MIXA3(Spread)
5	67	Reserved	MIXA3(Grouped)

## 2 Register Summary

The registers of the integrated programmable interrupt controller (IPIC) are fully documented in the *MPC5121e Reference Manual*. Remember, for the purposes of this application note, the MPC5121e is being used as an example. However, the registers for this module are always the same regardless of which particular microcontroller and where it is used. The assignments of the interrupting sources to the various sets or fixed priorities change from device to device. This section is simply to provide a quick overview of the purpose and function of each register. For exact programming details, consult the *MPC5121e Reference Manual*.

In the following discussion, PowerPC bit numbering notation is used.

### System Global Interrupt Configuration Register (SICFR) – 0x00

The HPI field specifies one interrupting source to be promoted to the highest priority interrupt.

The HPIT field determines the type of interrupt the highest priority interrupt generates, normal, system management, or critical interrupt.

The MPSA-B and IPSA-D bits determine whether each individual set of interrupting sources (MIXA, MIXB, SYSA, SYSB, SYSC, and SYSD) are Grouped or Spread. Each individual set must be either Grouped or Spread. All members of a set must be configured the same way.

### System Global Interrupt Vector Register (SIVCR) – 0x04

The lower seven bits of this register provide the vector number of the highest, unmasked pending interrupt to the CPU core.

#### NOTE

Bits 0 – 24 of this register are reset to a logic 0 from the release of reset. However, these bits change during device operation. Therefore, when using the contents of this register as an offset or for any other purpose, it is necessary that bits 0 – 24 be masked.

### System Internal Interrupt Pending Registers High (SIPNR\_H) – 0x08

The SIPNR\_H and SIPNR\_L registers contain one bit for each interrupting module. If a module is currently asserting an interrupt, its bit is set in the SIPNR\_H or SIPNR\_L registers.

### System Internal Interrupt Pending Register Low (SIPNR\_L) – 0x0C

The SIPNR\_H and SIPNR\_L registers contain one bit for each interrupting module. If a module is currently asserting an interrupt, its bit is set in the SIPNR\_H or SIPNR\_L registers.

## Register Summary

### System Internal Interrupt Group A Priority Register (SIPRR\_A) – 0x10

This register specifies the priority order of the modules included in the system internal interrupt group A priority register. There are eight entries for each group. All eight entries must be used. No entry can be used more than once.

### System Internal Interrupt Group B Priority Register (SIPRR\_B) – 0x14

This register specifies the priority order of the modules included in the system internal interrupt group B priority register. There are eight entries for each group. All eight entries must be used. No entry can be used more than once.

### System Internal Interrupt Group C Priority Register (SIPRR\_C) – 0x18

This register specifies the priority order of the modules included in the system internal interrupt group C priority register. There are eight entries for each group. All eight entries must be used. No entry can be used more than once.

### System Internal Interrupt Group D Priority Register (SIPRR\_D) – 0x1C

This register specifies the priority order of the modules included in the system internal interrupt group D priority register. There are eight entries for each group. All eight entries must be used. No entry can be used more than once.

### System Internal Interrupt Mask Register High (SIMSR\_H) – 0x20

The SIMSR\_H and SIMSR\_L registers provide an individual interrupt mask bit for the internal peripheral modules. For an interrupt to be passed to the CPU, the interrupt's respective bit in either the SIPNR\_H or SIPNR\_L register must be set and the interrupt's respective mask bit in either the SIMSR\_H or SIMSR\_L must be unmasked.

### System Internal Interrupt Mask Register Low (SIMSR\_L) – 0x24

The SIMSR\_H and SIMSR\_L registers provide an individual interrupt mask bit for the internal peripheral modules. For an interrupt to be passed to the CPU, the interrupt's bit in either the SIPNR\_H or SIPNR\_L register must be set and the interrupt's respective mask bit in either the SIMSR\_H or SIMSR\_L must be unmasked.

### System Internal Interrupt Control Register (SICNR) – 0x28

This register allows the highest two priority interrupting sources in SYSA, SYSB, SYSC, and SYSD to be individually programmed as normal interrupts, system management interrupts, or critical interrupts.

### System External Interrupt Pending Register (SEPNR) – 0x2C

These two bits in this register provide an indication that IRQ0 and/or IRQ1 have been asserted.

### System Mixed Interrupt Group A Priority Register (SMPRR\_A) – 0x30

This register is used to assign the relative priorities of the peripheral elements assigned to system group A.

### System Mixed Interrupt Group B Priority Register (SMPRR\_B) – 0x34

This register is used to assign the relative priorities of the peripheral elements assigned to system group B.

**System External Interrupt Mask Register (SEMSR) – 0x38**

This register is used as a mask register for IRQ0 and IRQ1. Also, it determines whether IRQ0 generates an external interrupt to the CPU or generate a machine check exception.

**System External Interrupt Control Register (SECNR) – 3C**

This register allows the highest two priority interrupting sources in MIXA and MIXB to be individually programmed as normal interrupts, system management interrupts, or critical interrupts. This register also specifies whether the MIXA and MIXB interrupts are low active or negative edge active.

**System Error Status Register (SERSR) – 0x40**

Each bit in the SERSR register corresponds to a non-maskable error source that generates a machine check exception (MCP). These bits are cleared by writing the bit location with a logic 1. The interrupting source must be cleared before clearing the interrupts error status bit.

**System Error Mask Register (SERMR) – 0x44**

When a machine check exception is signaled from one of the peripheral modules, its corresponding flag bit in this register is set. Any of the interrupting sources can be individually masked.

**System Error Control Register (SERCR) – 0x48**

This register contains a single bit that causes a machine check exception to be routed to the MPC\_OUT pin or the INTA pin.

**System Internal Interrupt Force Register High (SIFCR\_H) – 0x50**

Writing a logic 1 to any bit in this register forces an interrupt for the peripheral. The peripheral does not have to actually be asserting an interrupt itself. The priority and vector number are the same as though the peripheral had asserted the interrupt.

**System Internal Interrupt Force Register Low (SIFCR\_L) – 0x54**

Writing a logic 1 to any bit in this register forces an interrupt for the peripheral. The peripheral does not have to actually be asserting an interrupt itself. The priority and vector number are the same as though the peripheral had asserted the interrupt.

**System External Interrupt Force Register (SEFCR) – 0x58**

Writing a logic 1 to any bit in this register forces an interrupt for the external interrupt. The peripheral does not have to actually be asserting an interrupt itself. The priority and vector number are the same as though the respective peripheral had asserted the interrupt.

**System Error Force Register (SERFR) – 0x5C**

Writing a logic 1 to any bit in this register forces a machine check exception for the peripheral. The respective peripheral does not have to actually be asserting an interrupt itself. The priority and vector number are the same as though the respective peripheral had asserted the interrupt.

**System Critical Interrupt Vector Register (SCVCR) – 0x60**

## Programming Example Steps

The lower seven bits of this register specifies the vector number of the highest pending critical interrupt source. Notice bits 0 – 24 are set to a logic 0 at the release of reset; however, these bits change during device operation.

### NOTE

Bits 0 – 24 of this register are reset to a logic 0 from the release of reset. However, these bits change during device operation. Therefore, when using the contents of this register as an offset or for any other purpose, it is necessary bits 0 – 24 be masked.

## System Management Interrupt Vector Register – 0x60

The lower seven bits of this register specifies the vector number of the highest pending system management interrupt source.

### NOTE

Bits 0 – 24 of this register are reset to a logic 0 from the release of reset. However, these bits change during device operation. Therefore, when using the contents of this register as an offset or for any other purpose, it is necessary bits 0 – 24 be masked.

## 3 Programming Example Steps

### 3.1 Assign the Relative Priorities of the Interrupts in MIXA, MIXB, SYSA, SYSB, SYSC, and SYSD

To begin, the relative priorities of each set must be assigned by the system designer. A hypothetical applications example showing the relative priorities for each of the six sets is shown in [Table 9](#). These priorities were chosen to demonstrate a programming example. In an actual design, the system designer assigns these priorities.

**Table 9. Interrupting Sources per Set Example**

MIXA		MIXB		SYSA	
Relative Priority	Module	Relative Priority	Module	Relative Priority	Module
0	IRQ0	0	PSC2	0	PSC6
1	MBX	1	PSC0	1	PSC8
2	IRQ1	2	PSC1	2	PSC10
3	DMA	3	PSC3	3	PSC4
4	DIU	4	RESERVED_3	4	PSC11
5	RESERVED_0	5	RESERVED_4	5	PSC7
6	RESERVED_1	6	RESERVED_5	6	PSC5
7	RESERVED_2	7	RESERVED_6	7	PSC9
SYSB		SYSC		SYSD	
Relative Priority	Module	Relative Priority	Module	Relative Priority	Module
0	USB ULPI	0	PCI	0	I2C1
1	USB UTMI	1	PCI DMA	1	MSCAN1
2	AXE	2	PCI MU	2	I2C2
3	SATA	3	SDHC	3	MSCAN2
4	SPCIF	4	NFC	4	I2C3
5	FIFOC	5	FEC	5	BDLC
6	RESERVED_7	6	LPC	6	GPT1
7	RESERVED_8	7	PATA	7	GPT0

The assignment of priorities in [Table 9](#) is used to populate [Table 10](#). Each module of each set has both a Spread and a Grouped entry. Therefore, there are two entries in [Table 10](#) for each module. In the present example, IRQ0, MBX, and IRQ1 are the three highest priority items in MIXA. Looking at [Table 10](#), IRQ0 (Grouped) is priority 2, MBX (Grouped) is priority 3, and IRQ1 (Grouped) is priority 4. When looking at the spread column, IRQ0 (Spread), MBX (Spread), and IRQ1 (Spread) are priorities 1, 11, and 25.

**Table 10. Interrupting Sources Example**

Priority Level	Interrupt Source Description	Spread	Grouped
0	Highest		
1	MIXA0 (Spread)	IRQ0	
2	MIXA0 (Grouped)		IRQ0
3	MIXA1 (Grouped)		MBX
4	MIXA2 (Grouped)		IRQ1

**Table 10. Interrupting Sources Example (continued)**

Priority Level	Interrupt Source Description	Spread	Grouped
5	MIXA3 (Grouped)		DMA
6	MIXB0 (Spread)	PSC2	
7	SYSB0 (Grouped)		USB ULPI
8	SYSB1 (Grouped)		USB UTMI
9	SYSB2 (Grouped)		AXE
10	SYSB3 (Grouped)		SATA
11	MIXA1 (Spread)	MBX	
12	SYSB4 (Grouped)		SPDIF
13	SYSB5 (Grouped)		FIFOC
14	SYSB6 (Grouped)		
15	SYSB7 (Grouped)		
16	MIXB0 (Grouped)		PSC2
17	MIXB1 (Grouped)		PSC0
18	MIXB2 (Grouped)		PSC1
19	MIXB3 (Grouped)		
20	MIXB1 (Spread)	PSC0	
21	SYSA0 (Grouped)		PSC6
22	SYSA1 (Grouped)		PSC8
23	SYSA2 (Grouped)		PSC10
24	SYSA3 (Grouped)		PSC4
25	MIXA2 (Spread)	IRQ1	
26	SYSA4 (Grouped)		PSC11
27	SYSA5 (Grouped)		PSC7
28	SYSA6 (Grouped)		PSC5
29	SYSA7 (Grouped)		PSC9
30	MIXA4 (Grouped)		DIU
31	MIXA5 (Grouped)	x	
32	MIXA6 (Grouped)	x	
33	MIXA7 (Grouped)	x	

**Table 10. Interrupting Sources Example (continued)**

Priority Level	Interrupt Source Description	Spread	Grouped
34	MIXB2 (Spread)	PSC4	
35	SYSC0 (Grouped)		PCI
36	SYSC1 (Grouped)		PCI DMA
37	SYSC2 (Grouped)		PCI MU
38	SYSC3 (Grouped)		SDHC
39	MIXA3 (Spread)	DMA	
40	SYSC4 (Grouped)		NFC
41	SYSC5 (Grouped)		FEC
42	SYSC6 (Grouped)		LPC
43	SYSC7 (Grouped)		PATA
44	MIXB4 (Grouped)	X	
45	MIXB5 (Grouped)	X	
46	MIXB6 (Grouped)	X	
47	MIXB7 (Grouped)	X	
48	MIXB3 (Spread)	PSC3	
49	SYSD0 (Grouped)		I2C1
50	SYSD1 (Grouped)		MSCAN1
51	SYSD2 (Grouped)		I2C2
52	SYSD3 (Grouped)		MSCAN2
53	MIXA4 (Spread)	DIU	
54	SYSD4 (Grouped)		I2C3
55	SYSD5 (Grouped)		BDLC
56	SYSD6 (Grouped)		GPT1
57	SYSD7 (Grouped)		GPT0
58	MIXB4 (Spread)	X	
59	GPT2		
60	SYSB0 (Spread)	USB ULPI	
61	SYSA0 (Spread)	PSC6	
62	GPT3		
63	SYSC0 (Spread)	PCI	
64	SYSD0 (Spread)	I2C2	
65	Reserved		

**Table 10. Interrupting Sources Example (continued)**

Priority Level	Interrupt Source Description	Spread	Grouped
66	GPT4		
67	MIXA5 (Spread)	X	
68	GPT5		
69	SYSB1 (Spread)	USB UTMI	
70	SYSA1 (Spread)	PSC8	
71	GPT6		
72	SYSC1 (Spread)	PCI DMA	
73	SYSD1 (Spread)	MSCAN1	
74	Reserved		
75	GPT7		
76	MIXB5 (Spread)	X	
77	GPIO		
78	SYSB2 (Spread)	AXE	
79	SYSA2 (Spread)	PSC10	
80	RTC SEC		
81	SYSC2 (Spread)	PCI MU	
82	SYSD2 (Spread)	I2C2	
83	Reserved		
84	RTC ALARM		
85	MIXA6 (Spread)	X	
86	DDR		
87	SYSB3 (Spread)	SATA	
88	SYSA3 (Spread)	PSC4	
89	SBA		
90	SYSC3 (Spread)	SDHC	
91	SYSD3 (Spread)	MSCAN2	
92	Reserved		
93	PMC		
94	MIXB6 (Spread)	X	
95	USB ULPI WKUP		
96	SYSB4 (Spread)	SPDIF	
97	SYSA4 (Spread)	PSC11	



**Table 10. Interrupting Sources Example (continued)**

Priority Level	Interrupt Source Description	Spread	Grouped
98	USB UTMI WKUP		
99	SYSC4 (Spread)	NFC	
100	SYSD4 (Spread)	I2C3	
101	Reserved		
102	SATA CMD		
103	MIXA7 (Spread)	X	
104	TEMP 105C		
105	SYSB5 (Spread)	SPDIF	
106	SYSA5 (Spread)	PSC7	
107	IIM		
108	SYSC5 (Spread)	FEC	
109	SYSD5 (Spread)	BDLC	
110	Reserved		
111	PRIOMON		
112	MIXB7 (Spread)	X	
113	MSCAN3		
114	SYSB6 (Spread)	X	
115	SYSA6 (Spread)	PSC5	
116	MSCAN4		
117	SYSC6 (Spread)	LPC	
118	SYSD6 (Spread)	GPT1	
119	Reserved		
120	Reserved		
121	Reserved		
122	SYSB7 (Spread)	X	
123	SYSA7 (Spread)	PSC9	
124	Reserved		
125	SYSC7 (Spread)	PATA	
126	SYSD7 (Spread)	GPT0	
127	Reserved		
128	Reserved		

After [Table 10](#) has been filled in using the values in [Table 9](#), some decisions can be made about whether to use the Spread or Grouped programming for the respective sets.

[Table 11](#) presents the possible priorities for the two highest priority modules chosen in each set as shown in [Table 9](#). For the MIXA set, IRQ0 and MBX were chosen as the two highest priority items. If the MIXA set is programmed as Grouped, then these two modules are assigned to priorities 2 and 3. When programmed as Spread, IRQ0 and MBX are assigned to priorities 1 and 11. In this case, the two highest priority items in MIXA have relatively high priorities. At this point, the system designer needs to determine if either MBX or IRQ0 can have a priority as low as 11. If there is an issue that adversely affects assigning either of these modules to priority 11, then the MIXA set needs to be programmed to Group.

**Table 11. SPREAD vs. GROUPED Priorities**

SET	Module	GROUP Priorities	SPREAD Priorities
MIX A	IRQ0	2, 3	1, 11
	MBX		
MIX B	PSC2	16, 17	6, 20
	PSC0		
SYS A	PSC6	21, 22	61, 70
	PSC8		
SYS B	USB ULPI	7, 8	60, 69
	USB UTMI		
SYS C	PCI	35, 36	63, 72
	PCI DMA		
SYS D	I2C1	49, 50	64, 73
	MSCAN1		

As a practical matter, choosing whether a particular set is programmed to Spread or Grouped may be as simple as looking at the two or three highest priority modules within a set.

Now, consider the SYSA set. In the present example, PSC6 and PSC8 are chosen as the two highest priority peripherals. When programmed as Group, these two peripherals are priority levels 21 and 22. when programmed as Spread, these modules are assigned to priority levels 61 and 70. In this case, the system designer probably has a very clear idea about whether these two PSC modules can operate correctly with priority levels as low as 61 and 70.

In this manner, each set can be evaluated and some determination made as to whether the two highest priority items in each set can operate effectively at the priority levels imposed by the IPIC hardware. Then a choice must be made regarding Group or Spread encoding to obtain the optimal system performance.

### 3.2 System Global Interrupt Configuration Register

At this point, the assignment of relative priorities within each set and whether each respective set is designated as Grouped or Spread has been executed. Programming the remainder of the IPIC control

registers is more or less like filling out a table. To program the system global interrupt configuration register, use the following steps.

1. Of all interrupting sources, choose the highest priority interrupt.
2. Specify the type of interrupt the highest priority interrupt is to generate; normal, system management, or critical. There is a priority scheme between these three types of interrupts. Critical is the highest priority, system management is the second highest priority, and normal interrupt is the lowest priority. If SYSD is programmed as SPREAD then SYSD1 is priority 71. However, if SYSD1, which is programmed to MSCAN1 in the present example is specified to generate a critical interrupt, the CPU (not the IPIC) essentially treats it as having a higher priority than all of the modules programmed to system management or normal interrupts. The IPIC has a vector register for critical interrupts, system management interrupts, and all interrupts. This adds another level of complexity when assigning both priority levels and interrupt types to a particular module capable of generating interrupts to the IPIC.
3. Specify whether the MIXA, MIXB, SYSA, SYSB, SYSC, and SYSD sets are to be programmed as Spread or Grouped. In general, for the SYSA, SYSB, SYSC, and SYSD sets, programming any of them to Spread results in the modules in the sets to have a relatively low priority. For instance, for these four sets when programmed in Spread, the highest possible priority is SYSB0 which is priority 60. After filling out [Table 11](#), it is usually obvious whether Spread or Grouped priorities are better suited for a particular application.

System Internal Interrupt Group A Priority Register

System Internal Interrupt Group B Priority Register

System Internal Interrupt Group C Priority Register

System Internal Interrupt Group D Priority Register

System MIXED Interrupt Group A Priority Register

System MIXED Interrupt Group B Priority Register

For each of the above six sets, assign the relative priorities of each set's members. This job should have been executed early in system design and now it is a matter of properly programming these registers.

System Internal Interrupt Mask Register High

System Internal Interrupt Mask Register Low

Specify which interrupts are to be masked and which interrupts are to be passed along to the CPU. From the release of reset most interrupts are masked. This register can be used to dynamically program what interrupts can be recognized and which need to be held off. There are cases where some jobs must run to completion with out interruption. In such a case, the system internal interrupt mask register high / low can be used to hold off interrupts.

System Internal Interrupt Control Register

Specify the type of interrupt, normal, system management, or critical, the two top priority interrupts of SYSA, SYSB, SYSC, and SYSD are to generate. For each set, both interrupts have to generate the same type of interrupt. However, each set can generate any of the three types of interrupts. Normal, system management, and critical have a priority between themselves also to the

## Using the IPIC Control and Status Registers

priority assigned by the IPIC for the various interrupting sources. Programming a low priority source to the IPIC to generate a critical interrupt results in the CPU giving that interrupting source a priority higher than normal and system management interrupts.

### System External Interrupt Mask Register

Specify whether IRQ0 and IRQ1 are masked. Specify whether IRQ0 generates an external interrupt or machine check exception. A machine check can be programmed to cause the CPU to jump to the machine check exception handler or cause program execution to halt.

### System External Interrupt Control Register

Specify the type of interrupt (normal, system management, or critical) the two top priority interrupts of MIXA and MIXB are to generate. For each set, both interrupts have to generate the same type of interrupt. However, each set can generate any of the three types of interrupts.

Also, specify whether the interrupting sources of MIXA and MIXB are active, low level active or negative edge active.

### System Error Mask Register

Specify which of the following sources are to signal a machine check exception: IRQ0, watchdog timer, SBA, MU, and 125° C.

### System Error Control Register

Specify whether a machine check exception causes the MCP\_OUT pin or the INTA pin to assert.

## 4 Using the IPIC Control and Status Registers

### System Global Interrupt Vector Register

### System Critical Interrupt Vector Register

### System Management Interrupt Vector Register

These three registers supply the vector number of the highest priority pending interrupt, the highest priority pending critical interrupt, and the highest priority pending system management interrupt. This programming methodology allows the CPU to determine what peripheral needs to be serviced immediately without polling all the possible interrupting sources.

When the CPU responds to an interrupt, the program flow jumps to the appropriate exception handler. From this point, the exception handler interrogates the appropriate vector register to read the vector offset used to jump to the exception handler for a particular interrupting source. For instance, if the CPU responds to a critical interrupt, the program jumps to the critical interrupt exception handler, reads the system critical interrupt vector register, masks off the upper 23 bits, and then uses the resulting seven bits as a branch offset into a table that has the address of the individual interrupting source's exception handler.

**NOTE**

The upper 23 bits of the system global, system critical, and system management interrupt vector registers are not part of the particular vector number. These bits are reset to zero but during program execution, they can and will change. The upper 23 bits of these three registers must be masked off by software to obtain the particular vector offset value.

System Internal Interrupt Pending Register High

System Internal Interrupt Pending Register Low

System External Interrupt Pending Register

These three registers contain status bits for all of the individual interrupt sources. When reading one of the three vector registers, only the highest priority pending interrupt in each class can be determined. In some cases, it may be beneficial for the program to determine when multiple interrupt sources are present. These registers provide that capability.

As a general programming method, real time operating systems (RTOS) are many times used in systems using the MPC5121e and other processors of this class. In general, the PowerPC architecture is not designed to use nested interrupts. When an interrupt is recognized by the CPU, only a very minimal amount of information is placed on the stack. The PowerPC architecture does provide minimal support for stacking information in response to an interrupt, but this is usually limited to a return address and some status information. When using an RTOS, the CPU responds to an interrupt and then schedules a routine for service via the real time operating system. In this case, the RTOS determines when the interrupt handler is executed with respect to other program modules. Then the status bit of the interrupt is cleared. Using this programming method, there is rarely more than one or two interrupt status bits set at any one time.

By providing interrupt pending registers, the programmer is free to write any type of exception handler and respond to an interrupt in any way deemed appropriate. It is not necessary to use an RTOS. These registers provide the status information about what interrupts are currently asserted. The software writer can use this information in any manner they want to attain maximum system performance.

## 5 Forcing Interrupts

System Internal Interrupt Force Register High

System Internal Interrupt Force Register Low

System External Interrupt Force Register

System Error Force Register

These registers contain bits for all the interrupting sources. Setting a bit in one of these registers for a particular interrupt source causes an interrupt for that source to be asserted without having the conditions present to make the source assert the interrupt, itself. For instance, a programmable serial controller 0 may be idle. However, the PSC0 bit can be asserted in the system internal interrupt force register and the software responds as if PSC0 was actually receiving or transmitting data. Many times this feature is simply used to test hardware. However, there are actual conditions

## Summary

where devices external to the MPC5121e may need to be addressed to put them into an active condition. By forcing an interrupt, the software can be directed to service a particular module as if that module had actually generated the interrupt.

## 6 Summary

The IPIC is used to supply exception vector software with the vector number of the highest priority pending interrupt, critical interrupt, or system management interrupt. This is an effective time saving step for software exception handlers. It is not necessary to poll all peripherals in a particular class to see what peripherals need service. The IPIC can also manage which interrupting sources are passed to the CPU by masking or unmasking the interrupting source's interrupt.

After initialization, the priorities assigned to each interrupting module probably remain fixed. It is possible to change the priorities. However, all of the interrupting sources to a particular set must be disabled at the source before reprogramming the relative priorities for a particular set.

During system operation, various registers can be used to mask and unmask interrupts as necessary. However, after system operation begins, the other IPIC registers are not usually modified.

THIS PAGE IS INTENTIONALLY BLANK

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **Web Support:**

<http://www.freescale.com/support>

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### **Asia/Pacific:**

Freescale Semiconductor China Ltd.  
Exchange Building 23F  
No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022  
China  
+86 10 5879 8000  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### **For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or +1-303-675-2140  
Fax: +1-303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

AN3797  
Rev. 0  
03/2009