Holtek – Interrupts

Interrupts are an important part of any microcontroller system.

- When an **external event** or an **internal function** such as a **Timer Module** or an **A/D converter** requires microcontroller attention, their corresponding interrupt will **enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs**.
- The device contains **several external interrupt** and **internal interrupts functions**. → *HT66F50只有兩個 external interrupt*
- The **external interrupts** are generated by the action of the external **INT0~INT3** and **PINT** pins, while the **internal interrupts** are generated by various **internal functions such as the TMs, Comparators, Time Base, LVD, EEPROM, SIM and the A/D converter**.

Overall interrupt control, which basically means the **setting of request flags** when certain microcontroller conditions occur and the **setting of interrupt enable bits** by the application program, **is controlled by a series of registers**, located in the **Special Purpose Data Memory**.

The **number of registers** depends upon the device chosen but fall into **three categories**.

1. the **INTC0~INTC3 registers** which **setup the primary interrupts**,
2. the **MFI0~MFI3 registers** which **setup the Multi-function interrupts**.
3. Finally there is an **INTEG register** to **setup the external interrupt trigger edge type**.

Each register contains a number of **enable bits** to enable or disable individual registers as well as **interrupt flags** to indicate the presence of an interrupt request.

- The naming convention of these follows a specific pattern.
  - First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an **"E" for enable/disable bit** or **"F" for request flag**.

Interrupt Register Bit Naming Conventions

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Comparator	CPnE	CPnF	n=0 or 1
INTn Pin	INTnE	INTnF	n=0~3
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~5
Time Base	TBnE	TBnF	n=0 or 1
SIM	SIME	SIMF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
$\overline{\text{PINT}}$ Pin	XPE	XPF	—
TM	TnPE	TnPF	n=0~3
	TnAE	TnAF	
	TnBE	TnBF	

Holtek – Interrupts

Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	CP0F	INT1F	INT0F	CP0E	INT1E	INT0E	EMI
INTC1	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
INTC2	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
MFI0	T2AF	T2PF	T0AF	T0PF	T2AE	T2PE	T0AE	T0PE
MFI1	—	T1BF	T1AF	T1PF	—	T1BE	T1AE	T1PE
MFI2	DEF	LVF	XPF	SIMF	DEE	LVE	XPE	SIME
MFI3	—	—	T3AF	T3PF	—	—	T3AE	T3PE

Holtek – Interrupts

```
void __attribute((interrupt(0x24))) isr_tb0 (void)
{   ----------------------------------------
    . . . . . .
    . . . . . .
  [ _???f=0; ]
}   ----------------------------------------
    . . . . . .
    . . . . . .
    . . . . . .
```

0x24是程式記憶體的位址，發生中斷條件時，要跳耀去執行中斷程式的指令 之起點位置。

_emi = 0; //cleared automatically

prevent any further interrupt nesting from occurring

_emi = 1; //set automatically

[_???f=0; for some types of interrupt events]

related MFnF flag will be automatically cleared

When interrupt condition or event occurs and relative interrupt flag, **???f**, is set to high.

⊙中斷服務函式必須遵守下列規定:

→ 返回的資料類型必須是 void

→ 不能有參數

→ 必須使用__attribute__((interrupt(0x0c))) 設定中斷向量值 (interrupt vector)

→ 中斷入口會自動保存暫存器(ACC,BP,STATUS,MP/TBLP), 並在中斷出口時恢復.

注:MP/TBLP只在中斷函式有使用到時才會保存。中斷服務函式的使用注意事項:

⊙ C Compiler V3支援中斷內部調用函式，但不同中斷與main之間不能調用同一個函式，會造成RAM 重疊，對此現象 linker 將偵測出並報warning (若被調用的函式無宣告及使用任何的local 變數可忽略此warning)，如下例子：

```
void fun1(){}
void fun2(){fun1();}
void main()
{
fun1();
}
void __attribute((interrupt(0x04))) isr1(void)
{
fun1();
}
void __attribute((interrupt(0x08))) isr2(void)
{
fun2();
}
```

main呼叫fun1，但是fun2也呼叫fun1

Holtek – INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 INT1S1, INT1S0: Interrupt edge control for INT1 pin

00: Disable

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 1~0 INT0S1, INT0S0: Interrupt edge control for INT0 pin

00: Disable

01: Rising edge

10: Falling edge

11: Rising and falling edges

Holtek – INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	CP0F	INT1F	INT0F	CP0E	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 CP0F: Comparator 0 interrupt request flag
0: No request ; 1: Interrupt request

Bit 5 INT1F: INT1 interrupt request flag
0: No request; 1: Interrupt request

Bit 4 INT0F: INT0 interrupt request flag
0: No request; 1: Interrupt request

Bit 3 CP0E: Comparator 0 interrupt control
0: Disable; 1: Enable

Bit 2 INT1E: INT1 interrupt control
0: Disable; 1: Enable

Bit 1 INT0E: INT0 interrupt control
0: Disable; 1: Enable

Bit 0 EMI: Global interrupt control
0: Disable; 1: Enable

Holtek – INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 ADF: A/D Converter Interrupt Request Flag
0: No request; 1: Interrupt request

Bit 6 MF1F: Multi-function Interrupt 1 Request Flag
0: No request; 1: Interrupt request

Bit 5 MF0F: Multi-function Interrupt 0 Request Flag
0: No request; 1: Interrupt request

Bit 4 CP1F: Comparator 1 Interrupt Request Flag
0: No request; 1: Interrupt request

Bit 3 ADE: A/D Converter Interrupt Control
0: Disable; 1: Enable

Bit 2 MF1E: Multi-function Interrupt 1 Control
0: Disable; 1: Enable

Bit 1 MF0E: Multi-function Interrupt 0 Control
0: Disable; 1: Enable

Bit 0 CP1E: Comparator 1 Interrupt Control
0: Disable; 1: Enable

Holtek – INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 MF3F: Multi-function Interrupt 3 Request Flag
0: No request; 1: Interrupt request

Bit 6 TB1F: Time Base 1 Interrupt Request Flag
0: No request; 1: Interrupt request

Bit 5 TB0F: Time Base 0 Interrupt Request Flag
0: No request; 1: Interrupt request

Bit 4 MF2F: Multi-function Interrupt 2 Request Flag
0: No request; 1: Interrupt request

Bit 3 MF3E: Multi-function Interrupt 3 Control
0: Disable; 1: Enable

Bit 2 TB1E: Time Base 1 Interrupt Control
0: Disable; 1: Enable

Bit 1 TB0E: Time Base 0 Interrupt Control
0: Disable; 1: Enable

Bit 0 MF2E: Multi-function Interrupt 2 Control
0: Disable; 1: Enable

Holtek – MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	T2AF	T2PF	T0AF	T0PF	T2AE	T2PE	T0AE	T0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 T2AF: TM2 Comparator A match interrupt request flag
0: No request; 1: Interrupt request

Bit 6 T2PF: TM2 Comparator P match interrupt request flag
0: No request; 1: Interrupt request

Bit 5 T0AF: TM0 Comparator A match interrupt request flag
0: No request; 1: Interrupt request

Bit 4 T0PF: TM0 Comparator P match interrupt request flag
0: No request; 1: Interrupt request

Bit 3 T2AE: TM2 Comparator A match interrupt control
0: Disable; 1: Enable

Bit 2 T2PE: TM2 Comparator P match interrupt control
0: Disable; 1: Enable

Bit 1 T0AE: TM0 Comparator A match interrupt control
0: Disable; 1: Enable

Bit 0 T0PE: TM0 Comparator P match interrupt control
0: Disable; 1: Enable

Holtek – MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T1BF	T1AF	T1PF	—	T1BE	T1AE	T1PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 T1BF: TM1 Comparator B match interrupt request flag
0: No request; 1: Interrupt request

Bit 5 T1AF: TM1 Comparator A match interrupt request flag
0: No request; 1: Interrupt request

Bit 4 T1PF: TM1 Comparator P match interrupt request flag
0: No request; 1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 T1BE: TM1 Comparator B match interrupt control
0: Disable; 1: Enable

Bit 1 T1AE: TM1 Comparator A match interrupt control
0: Disable; 1: Enable

Bit 0 T1PE: TM1 Comparator P match interrupt control
0: Disable; 1: Enable

Holtek – MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	DEF	LVF	XPF	SIMF	DEE	LVE	XPE	SIME
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 DEF: Data EEPROM interrupt request flag
0: No request; 1: Interrupt request

Bit 6 LVF: LVD interrupt request flag
0: No request; 1: Interrupt request

Bit 5 XPF: External peripheral interrupt request flag
0: No request; 1: Interrupt request

Bit 4 SIMF: SIM interrupt request flag
0: No request; 1: Interrupt request

Bit 3 DEE: Data EEPROM Interrupt Control
0: Disable; 1: Enable

Bit 2 LVE: LVD Interrupt Control
0: Disable; 1: Enable

Bit 1 XPE: External Peripheral Interrupt Control
0: Disable 1: Enable

Bit 0 SIME: SIM Interrupt Control
0: Disable; 1: Enable

Holtek – MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T3AF	T3PF	—	—	T3AE	T3PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 T3AF: TM3 Comparator A match interrupt request flag
0: No request; 1: Interrupt request

Bit 4 T3PF: TM3 Comparator P match interrupt request flag
0: No request; 1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 T3AE: TM3 Comparator A match interrupt control
0: Disable; 1: Enable

Bit 0 T3PE: TM3 Comparator P match interrupt control
0: Disable; 1: Enable

Holtek – Interrupt Structure

- When **an interrupt is generated**, the **Program Counter**, which ***stores the address of the next instruction to be executed***, will **be transferred onto the stack**.

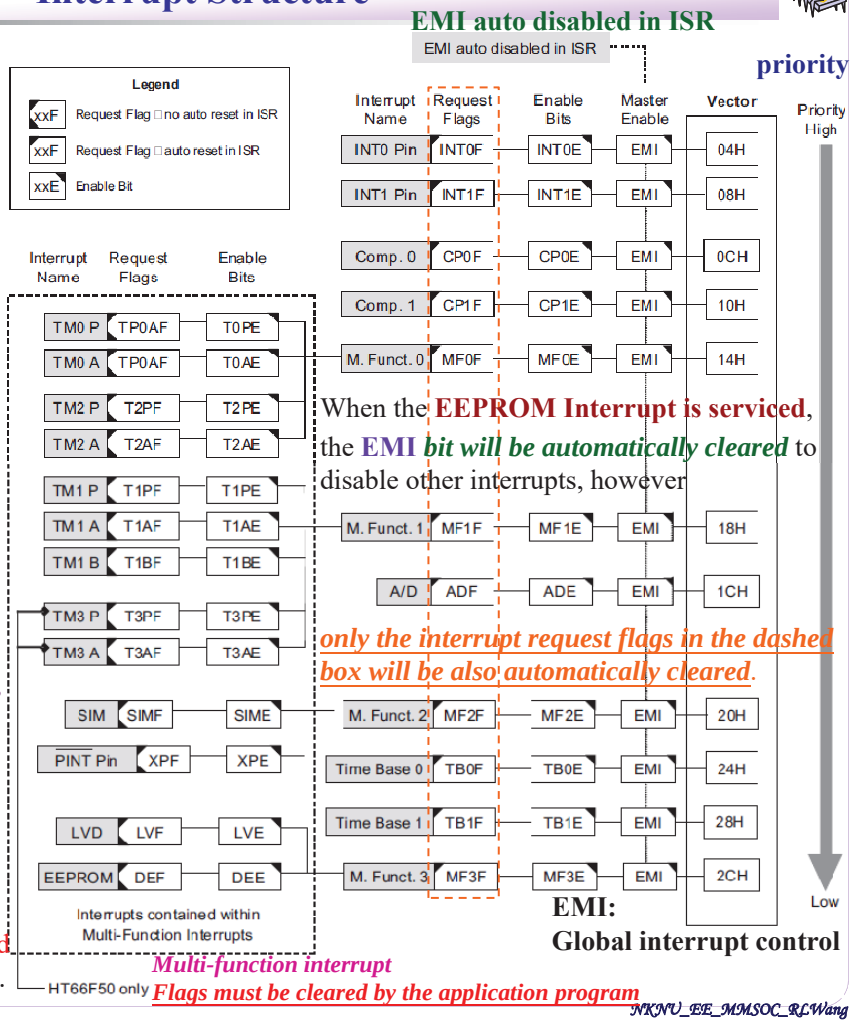
⇨The **Program Counter** will ***then be loaded with a new address*** which will be the value of the ***corresponding interrupt vector***.

⇨The microcontroller will then fetch its next instruction from this interrupt vector.

⇨The ***instruction at this vector will usually be a JMP which will jump to another section of program*** which is known as the ***interrupt service routine***. Here is located the code to control the appropriate interrupt.

⇨The ***interrupt service routine must be terminated with a RETI***, which ***retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution*** at the point where the interrupt occurred.

The **RETI** instruction in addition to **executing a return to the main program also automatically sets the EMI bit high to allow further interrupts**. The **RET** instruction however only executes a return to the main program ***leaving the EMI bit in its present zero state*** and therefore **disabling the execution of further interrupts**.



Holtek – Interrupt Operation

- When the **conditions for an interrupt event occur**, such as a ***TM Comparator P, Comparator A or Comparator B match*** or ***A/D conversion completion*** etc, the **relevant interrupt request flag will be set**. (**irrespective of whether the relevant interrupt enable bit is set high**)
  - ⇨Whether the **request flag actually generates a program jump to the relevant interrupt vector** is determined by the condition of the **interrupt enable bit**.
  - ⇨If the **enable bit** is set **high** then the **program will jump to its relevant vector**; if the enable bit is **zero** then although the interrupt request flag is set, an **actual interrupt will not be generated** and the program will not jump to the relevant interrupt vector.
  - ⇨The **global interrupt enable bit, EMI**, if cleared to zero, will disable all interrupts.
  - ⇨When an interrupt is generated, the **Program Counter**, which stores the address of the next instruction to be executed, will be **transferred onto the stack**.
    - ◈The **Program Counter** will then be **loaded with a new address which will be the value of the corresponding interrupt vector**.
- **Some interrupt sources** have their **own individual vector** while **others share the same multi-function interrupt vector**.
- ***Once an interrupt subroutine is serviced***, ***all the other interrupts will be blocked***, as the **global interrupt enable bit, EMI bit will be cleared automatically**. → ***Default: no interrupt nesting***
  - ⇨**This will prevent any further interrupt nesting from occurring**.
  - ⇨However, **if other interrupt requests occur during this interval**, although the interrupt will not be immediately serviced, **the request flag will still be recorded**.
- **If an interrupt requires immediate servicing while the program is already in another interrupt service routine**, the **EMI bit should be set after entering the routine, to allow interrupt nesting**.
  - ⇨If the **stack is full**, the **interrupt request will not be acknowledged**, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.
  - ⇨**In case of simultaneous requests**, the accompanying diagram shows the **priority that is applied**.

(*see the previous page*)

Holtek – External Interrupt、Comparator Interrupt

- The **external interrupts** are **controlled by signal transitions on the pins INT0~INT3**.
  - ⇨An **external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, are set**, which will **occur when a transition**, **whose type is chosen by the edge select bits**, appears on the external interrupt pins.
- To **allow the program to branch to its respective interrupt vector address**, the **global interrupt enable bit, EMI**, and respective **external interrupt enable bit, INT0E~INT3E**, **must first be set**.
  - ⇨Additionally the correct interrupt edge type must be selected using the **INTEG register** to **enable the external interrupt function** and to **choose the trigger edge type**.
  - ⇨As the **external interrupt pins** are **pin-shared with I/O pins**, they can **only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set**.
  - ⇨The **pin must also be setup as an input** by setting the corresponding bit in the **port control register**.
- **When the interrupt is serviced**, the **external interrupt request flags, INT0F~INT3F,** will be ***automatically reset*** and the **EMI bit** will be ***automatically cleared to disable other interrupts***.
- Note that **any pull-high resistor selections on the external interrupt pins** will **remain valid** even if the pin is used as an external interrupt input.
- The **INTEG register** is used to **select the type of active edge** that will trigger the external interrupt.
  - ⇨A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt.
  - ⇨Note that the INTEG register can also be used to disable the external interrupt function.
- The **comparator interrupt** is **controlled by the two internal comparators**.
  - ⇨A **comparator interrupt request** will take place when the comparator interrupt request flags, **CP0F or CP1F, are set**, **a situation that will occur when the comparator output changes state**.
- To **allow the program to branch** to its respective interrupt vector address, the **global interrupt enable bit, EMI**, and **comparator interrupt enable bits, CP0E and CP1E**, **must first be set**.
- When the **interrupt is serviced**, the **external interrupt request flags**, will be ***automatically reset*** and the **EMI** bit will be ***automatically cleared to disable other interrupts***.

Holtek – Timer Interrupt、EEPROM Interrupt

- The **Compact and Standard Type TMs** have **two interrupts** each, while the **Enhanced Type TM** has **three interrupts.**

⇨A **TM interrupt request will take place when any of the TM request flags are set**, a situation which **occurs when a TM comparator P, A or B match situation happens**.

⇨**When a TM interrupt is generated,** it **can be used to clear the counter and also to change the state of the TM output pin**.

⇨**All of the TM interrupts** are contained **within the Multi-function Interrupts**.

To allow the program to branch to its respective interrupt vector address, the **global interrupt enable bit, EMI**, **respective TM Interrupt enable bit**, and **relevant Multi-function Interrupt enable bit, MFnE**, **must first be set**.

When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place.

When the TM interrupt is serviced, the **EMI *bit will be automatically cleared*** to disable other interrupts, however **only the related MFnF flag will be automatically cleared**. As the ***TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program***. ⇦------*Very important !!!*

- The **EEPROM Interrupt**, is contained within the **Multi-function Interrupt**.

⇨An **EEPROM Interrupt request will take place** when the **EEPROM Interrupt request flag, DEF, is set**, which **occurs when an EEPROM Write or Read cycle ends**.

⇨To **allow the program to branch to its respective interrupt vector address**, the **global interrupt enable bit, EMI**, **EEPROM Interrupt enable bit, DEE**, and **associated Multi-function interrupt enable bit**, **must first be set**.

When the **EEPROM Interrupt is serviced**, the **EMI *bit will be automatically cleared*** to disable other interrupts, however ***only the Multi-function interrupt request flag will be also automatically cleared***. As **the *DEF flag will not be automatically cleared, it has to be cleared by the application program***. ⇦------*Very important !!!*

Holtek – ADC Interrupt、SIM Interrupt、Time Base

- The **A/D Converter Interrupt** is **controlled by the termination of an A/D conversion process**.

⇨An **A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set**, **which occurs when the A/D conversion process finishes**.

⇨To **allow the program to branch to its respective interrupt vector address**, the **global interrupt enable bit, EMI**, and **A/D Interrupt enable bit, ADE**, **must first be set**.

⇨When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place.

When the interrupt is serviced, the **A/D Converter Interrupt flag, ADF, *will be automatically cleared***.

The **EMI *bit will also be automatically cleared*** to disable other interrupts.

- The **Serial Interface Module Interrupt**, also known as the **SIM interrupt**, is contained within the **Multi-function Interrupt**.

⇨A **SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set**, **which occurs when a byte of data has been received or transmitted by the SIM interface**.

⇨To **allow the program to branch to its respective interrupt vector address**, the **global interrupt enable bit, EMI**, and the **Serial Interface Interrupt enable bit**, **SIME**, and **Muti-function interrupt enable bits**, **must first be set**.

⇨When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SIM interface, a subroutine call to the respective Multi-function Interrupt vector, will take place.

When the Serial Interface Interrupt is serviced, the **EMI bit will be automatically cleared** to disable other interrupts, however ***only the Multi-function interrupt request flag will be also automatically cleared***. As the ***SIMF flag will not be automatically cleared, it has to be cleared by the application program*.** ⇦------*Very important !!!*

- When **Time Base interrupt** happens, their respective interrupt request flags, TB0F or TB1F will be set. To **allow the program to branch to their respective interrupt vector addresses**, the **global interrupt enable bit, EMI** and **Time Base enable bits, TB0E or TB1E**, **must first be set**.

⇨**When the Time Base interrupt is serviced**, *the* ***respective interrupt request flag, TB0F or TB1F, will be automatically reset*** and **the EMI bit *will be cleared*** *to disable other interrupts.*

Holtek – Interrupt Wake-up Function、Programming Considerations

- **Each of the interrupt functions has the capability of waking up the microcontroller** when in the **SLEEP** or **IDLE** Mode. *interrupt request flag : low→ high ➔ wake-up is generated*
- A ***wake-up is generated*** when ***an interrupt request flag changes from low to high*** and is **independent of whether the interrupt is enabled or not**.

⇨Therefore, **even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped**, situations such as external edge transitions on the ***external interrupt*** pins, a ***low power supply voltage*** or ***comparator input*** change may **cause their respective interrupt flag to be set high and consequently *generate an interrupt***.

⇨**Care must therefore be taken if spurious wake-up situations are to be avoided**.

✓***If an interrupt wake-up function is to be disabled*** then the ***corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode.***

⇨**The interrupt enable bits have no effect on the interrupt wake-up function**.

Programming Considerations :

- By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.
- Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as **only the Multi-function interrupt request flags, MF0F~MF5F, will be automatically cleared**, the **individual request flag for the function needs to be cleared by the application program**.
- **It is recommended that programs do not use the CALL instruction within the interrupt service subroutine**.
- **Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode**, the **wake up being generated *when the interrupt request flag changes from low to high***.

⇨If it **is required to *prevent a certain interrupt from waking up the microcontroller*** then ***its respective request flag should be first set high before enter SLEEP or IDLE Mode.***

- As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the **contents of the accumulator, status register or other registers are altered by the interrupt service program**, **their contents should be saved to the memory at the beginning of the interrupt service routine.**

Holtek – Time Base Interrupt

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions.

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	LXTLP	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	1	0	1	1	1

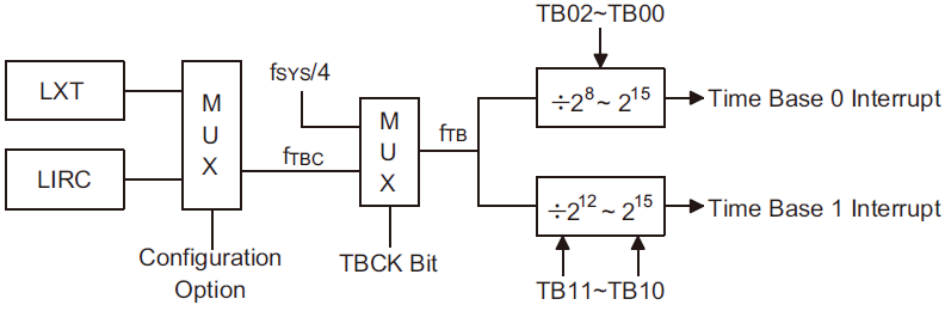
Bit 7 TBON: TB0 and TB1 Control
0: Disable; 1: Enable

Bit 6 TBCK: Select f_{TB} Clock
0: f_{TBC} ; 1: $f_{SYS}/4$

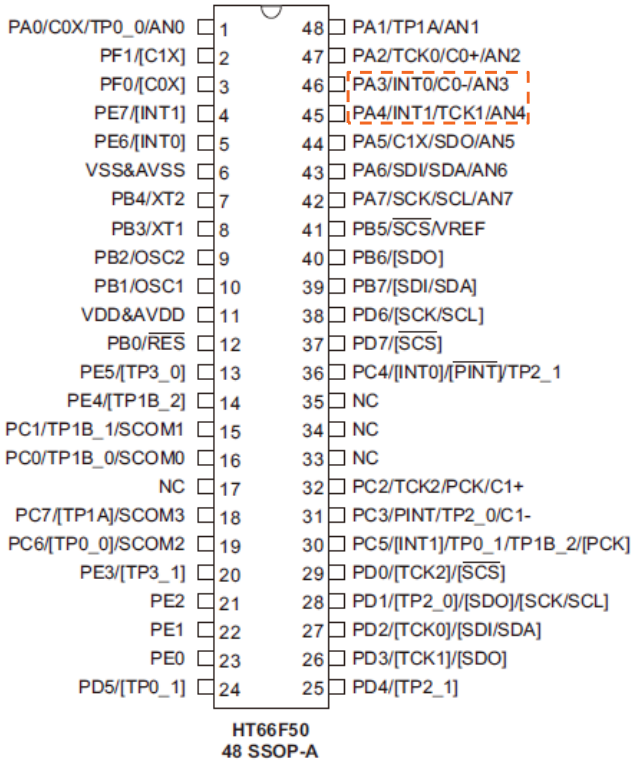
Bit 5~4 TB11~TB10: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$; 01: $8192/f_{TB}$
10: $16384/f_{TB}$; 11: $32768/f_{TB}$

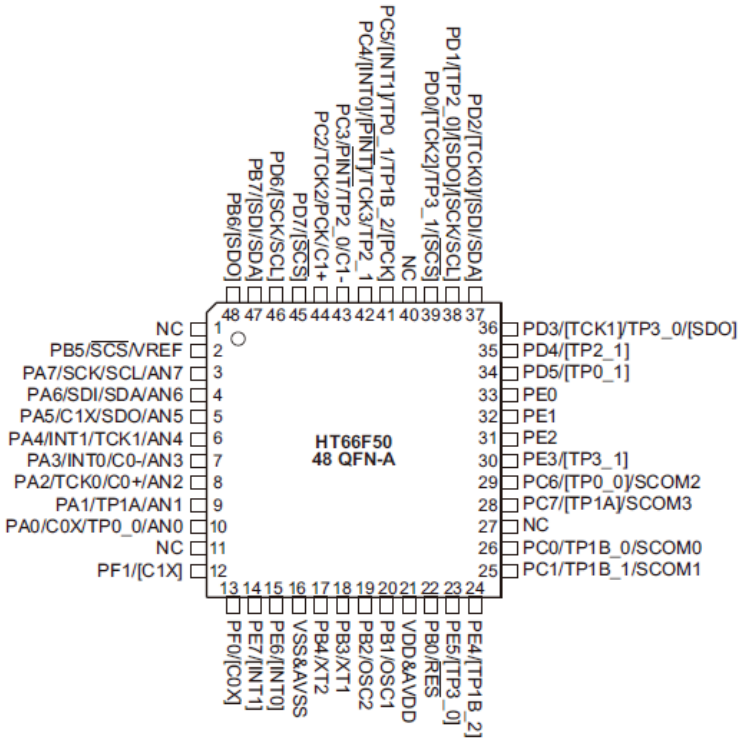
Bit 3 LXTLP: LXT Low Power Control
0: Disable; 1: Enable

Bit 2~0 TB02~TB00: Select Time Base 0 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$



Holtek – Data Memory





Note:

1. Bracketed pin names indicate non-default pinout remapping locations.
2. If the pin-shared pin functions have multiple outputs simultaneously, **its pin names at the right side of the / sign can be used for higher priority**.
3. VDD&AVDD means the VDD and AVDD are the double bonding.

Holtek – RAM Data Memory

- The Data Memory is a volatile area of **8-bit wide RAM internal memory** and is the location where temporary information is stored. The overall Data Memory is subdivided into several banks.
- **Divided into two sections**, the **first** of these is an area of RAM, known as the **Special Function Data Memory**.
  - ⇨Here are located registers which are necessary for correct operation of the device.
  - ⇨**Many** of these registers **can be read from and written to** directly under program control, however, **some remain protected from user manipulation**.
- The **second** area of Data Memory is known as the **General Purpose Data Memory**, which is reserved for general purpose use.
  - ⇨**All locations** within this area are read and write accessible under program control.

Device	Capacity	Banks
HT66F20	64×8	0: 60H~7FH 1: 60H~7FH
HT66F30	96×8	0: 60H~7FH 1: 60H~7FH 2: 60H~7FH
HT66F40	192×8	0: 80H~FFH 1: 80H~BFH
HT66F50	384×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH
HT66F60	576×8	0: 80H~ 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH 4: 80H~BFH

	Bank 0, 1, 2		Bank 0, 1, 2		Bank 0, 2	Bank 1		Bank 0, 2	Bank 1
00H	IAR0	20H	PC	40H	Unused	EEC	60H	Unused	
01H	MP0	21H	PCC	41H	EEA		61H	Unused	
02H	IAR1	22H	PDPU	42H	EED		62H	Unused	
03H	MP1	23H	PD	43H	TMPC0		63H	Unused	
04H	BP	24H	PDC	44H	TMPC1		64H	Unused	
05H	ACC	25H	PEPU	45H	PRM0		65H	Unused	
06H	PCL	26H	PE	46H	PRM1		66H	Unused	
07H	TBLP	27H	PEC	47H	PRM2		67H	Unused	
08H	TBLH	28H	PFPU	48H	TM1C0		68H	Unused	
09H	TBHP	29H	PF	49H	TM1C1		69H	Unused	
0AH	STATUS	2AH	PFC	4AH	TM1C2		6AH	Unused	
0BH	SMOD	2BH	Unused	4BH	TM1DL		6BH	Unused	
0CH	LVDC	2CH	Unused	4CH	TM1DH		6CH	Unused	
0DH	INTEG	2DH	Unused	4DH	TM1AL		6DH	Unused	
0EH	WDTC	2EH	ADRL	4EH	TM1AH		6EH	Unused	
0FH	TBC	2FH	ADRH	4FH	TM1BL		6FH	Unused	
10H	INTC0	30H	ADCR0	50H	TM1BH		70H	Unused	
11H	INTC1	31H	ADCR1	51H	TM2C0		71H	Unused	
12H	INTC2	32H	ACERL	52H	TM2C1		72H	Unused	
13H	Unused	33H	Unused	53H	TM2DL		73H	Unused	
14H	MFI0	34H	CP0C	54H	TM2DH		74H	Unused	
15H	MFI1	35H	CP1C	55H	TM2AL		75H	Unused	
16H	MFI2	36H	SIMC0	56H	TM2AH		76H	Unused	
17H	MFI3	37H	SIMC1	57H	TM2RP		77H	Unused	
18H	PAWU	38H	SIMD	58H	TM3C0		78H	Unused	
19H	PAPU	39H	SIMA/SIMC2	59H	TM3C1		79H	Unused	
1AH	PA	3AH	TM0C0	5AH	TM3DL		7AH	Unused	
1BH	PAC	3BH	TM0C1	5BH	TM3DH		7BH	Unused	
1CH	PBPU	3CH	TM0DL	5CH	TM3AL		7CH	Unused	
1DH	PB	3DH	TM0DH	5DH	TM3AH		7DH	Unused	
1EH	PBC	3EH	TM0AL	5EH	SCOMC		7EH	Unused	
1FH	PCPU	3FH	TM0AH	5FH	Unused		7FH	Unused	