Holtek – C to Assembly Code

```
#include "HT66F50.h"
#define LedPortx _pd // pd 埠
#define LedPorty _pe // pe 埠
void delay (unsigned char n)
{
    unsigned char idy1, idy2, idy3;
    for(idy1 = 0; idy1 < n ; idy1++)  //
    {
                GCC_NOP();
                for(idy2=0; idy2<200 ; idy2++)
                 {
                    GCC_NOP();
                    for(idy3=0; idy3<200 ; idy3++)
                    {
                                GCC_NOP();
                    }
                }
    }
}
void main(void)
{
    _smod=0b11000000;       // [7:5]=100->fH/16,110-->fH/4
    _pdc = 0 ;              // 設定 PD 埠為輸出
    LedPortx = 0x00 ;       // 8x8 LED X-axis (PD) OFF
    _pec=0;                 // 設定 PE 埠為輸出
    LedPorty = 0x00;        // 8x8 LED Y-axis (PE) ON
    while(1)
    {
                LedPortx = 0x01 ;
                LedPorty = 0xff;            // 8x8 LED Y-axis (PE) ON
                delay(10);
                LedPortx = 0x02 ;
                LedPorty = 0x7e;            // 8x8 LED Y-axis (PE) ON
                delay(10);
                LedPortx = 0x04 ;
```

Assembly code

MOV A, [m]	Move Data Memory to ACC
MOV [m], A	Move ACC to Data Memory
MOV A, x	Move immediate data to ACC
SUB A, [m]	Subtract Data Memory from ACC
SDZ [m]	Skip if decrement Data Memory is zero
SNZ [m].i	Skip if bit i of Data Memory is not 0

```
0000  2801  JMP  1H
0001  2816  JMP  16H
0002  4083  MOV  [83H],A
0003  5F00  CLR  [80H]
0004  2811  JMP  11H
0005  0000  NOP
0006  0FC8  MOV  A,0c8H
0007  4081  MOV  [81H],A
0008  0000  NOP
0009  0FC8  MOV  A,0c8H
000A  4082  MOV  [82H],A
000B  0000  NOP
000C  5782  SDZ  [82H]
000D  280B  JMP  0bH
000E  5781  SDZ  [81H]
000F  2808  JMP  8H
0010  5480  INC  [80H]
0011  4700  MOV  A,[80H]
0012  4203  SUB  A,[83H]
0013  390A  SNZ  Z
0014  2805  JMP  5H
0015  0003  RET
0016  0FC0  MOV  A,0c0H
0017  008B  MOV  SMOD,A
0018  1F24  CLR  PDC
0019  1F23  CLR  PD
001A  1F27  CLR  PEC
001B  1F26  CLR  PE
001C  0F01  MOV  A,1H
001D  00A3  MOV  PD,A
001E  1FA6  SET  PE
001F  0F0A  MOV  A,0aH
0020  2002  CALL 2H
0021  0F02  MOV  A,2H
0022  00A3  MOV  PD,A
```


Holtek – Assembly Code to Machine (Binary) Code

```
0000  2801  JMP  1H
0001  2816  JMP  16H
0002  4083  MOV  [83H],A
0003  5F00  CLR  [80H]
0004  2811  JMP  11H
0005  0000  NOP
0006  0FC8  MOV  A,0c8H
0007  4081  MOV  [81H],A
0008  0000  NOP
0009  0FC8  MOV  A,0c8H
000A  4082  MOV  [82H],A
000B  0000  NOP
000C  5782  SDZ  [82H]
000D  280B  JMP  0bH
000E  5781  SDZ  [81H]
000F  2808  JMP  8H
0010  5480  INC  [80H]
0011  4700  MOV  A,[80H]
0012  4203  SUB  A,[83H]
0013  390A  SNZ  Z
0014  2805  JMP  5H
0015  0003  RET
0016  0FC0  MOV  A,0c0H
0017  008B  MOV  SMOD,A
0018  1F24  CLR  PDC
0019  1F23  CLR  PD
001A  1F27  CLR  PEC
001B  1F26  CLR  PE
001C  0F01  MOV  A,1H
001D  00A3  MOV  PD,A
001E  1FA6  SET  PE
001F  0F0A  MOV  A,0aH
0020  2002  CALL 2H
0021  0F02  MOV  A,2H
0022  00A3  MOV  PD,A
```

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 01 28 16 28 83 40 00 5F 11 28 00 00 C8 0F 81 40 ; .(.(  ._.(..?
00000010h: 00 00 C8 0F 82 40 00 00 82 57 0B 28 81 57 08 28 ; ..?  ..  .(  .(
00000020h: 80 54 00 47 03 42 0A 39 05 28 03 00 C0 0F 8B 00 ; T.G.B.9.(..??
00000030h: 24 1F 23 1F 27 1F 26 1F 01 0F A3 00 A6 1F 0A 0F ; $.#.'.&...??..
00000040h: 02 20 02 0F A3 00 7E 0F A6 00 0A 0F 02 20 04 0F ; . ..?~.?... ..
00000050h: A3 00 3C 0F A6 00 0A 0F 02 20 08 0F A3 00 18 0F ; ?<.?... ..?..
00000060h: A6 00 0A 0F 02 20 10 0F A3 00 18 0F A6 00 0A 0F ; ?... ..?..?..
00000070h: 02 20 20 0F A3 00 3C 0F A6 00 0A 0F 02 20 40 0F ; .  .?<.?... @.
00000080h: A3 00 7E 0F A6 00 0A 0F 02 20 80 0F A3 00 A6 1F ; ?~.?... .??
00000090h: 0A 0F 02 20 1C 28 4B 28 00 00 00 00 00 00 00 00 ; ... .(K(........
000000a0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000b0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000d0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000e0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000120h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000130h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000180h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

Holtek – Program Counter & Stack

0x00		MOV	A, 0AH
0x01		MOV	[85H],A
0x02		NOP	[86H],A
0x03		CALL	fun1
0x04		MOV	A,05H
0x05		MOV	[82H],A
0x06		JMP	ini
0x07	fun1	CALL	fun2
0x08		MOV	A,0AH
0x09		MOV	[85H],A
0x0a		INC	[81H]
0x0b		ret	
0x0c	fun2	call	fun3
0x0d		SDZ	[86H]
0x0e		JMP	fun2
0x0f		ret	
0x10	fun3	SDZ	[85H]
0x11		JMP	fun3
0x12		ret	
0x13	ini	SUB	A, [83H]
0x14		MOV	[84H],A

PC counter	0x07
Stack level 1	0x04
Stack level 2	
Stack level 3	
Stack level 4	
Stack level 5	
Stack level 6	
Stack level 7	
Stack level 8	

PC counter	0x0c
Stack level 1	0x08
Stack level 2	0x04
Stack level 3	
Stack level 4	
Stack level 5	
Stack level 6	
Stack level 7	
Stack level 8	

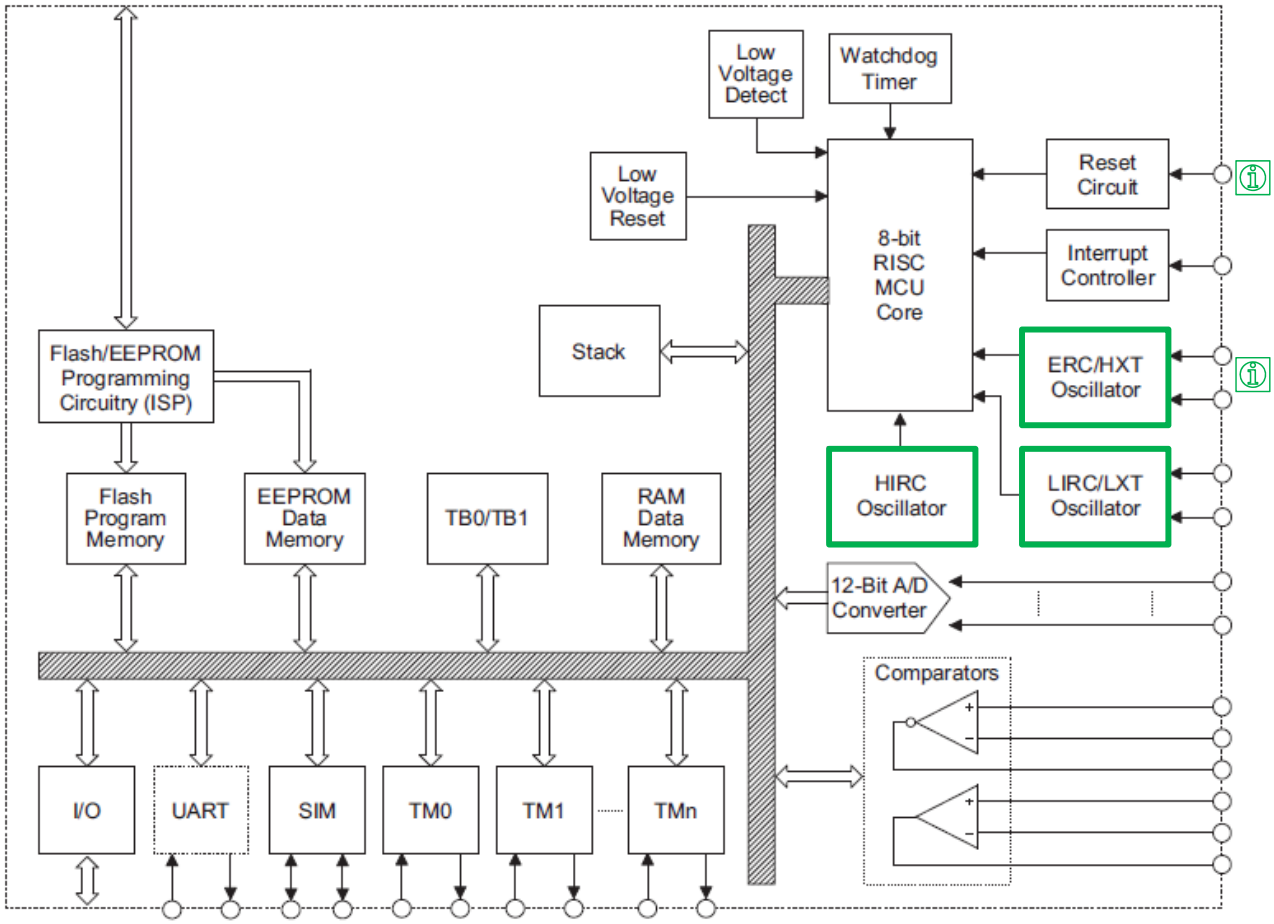
PC counter	0x10
Stack level 1	0x0d
Stack level 2	0x08
Stack level 3	0x04
Stack level 4	
Stack level 5	
Stack level 6	
Stack level 7	
Stack level 8	

PC counter	0x0d
Stack level 1	0x08
Stack level 2	0x04
Stack level 3	
Stack level 4	

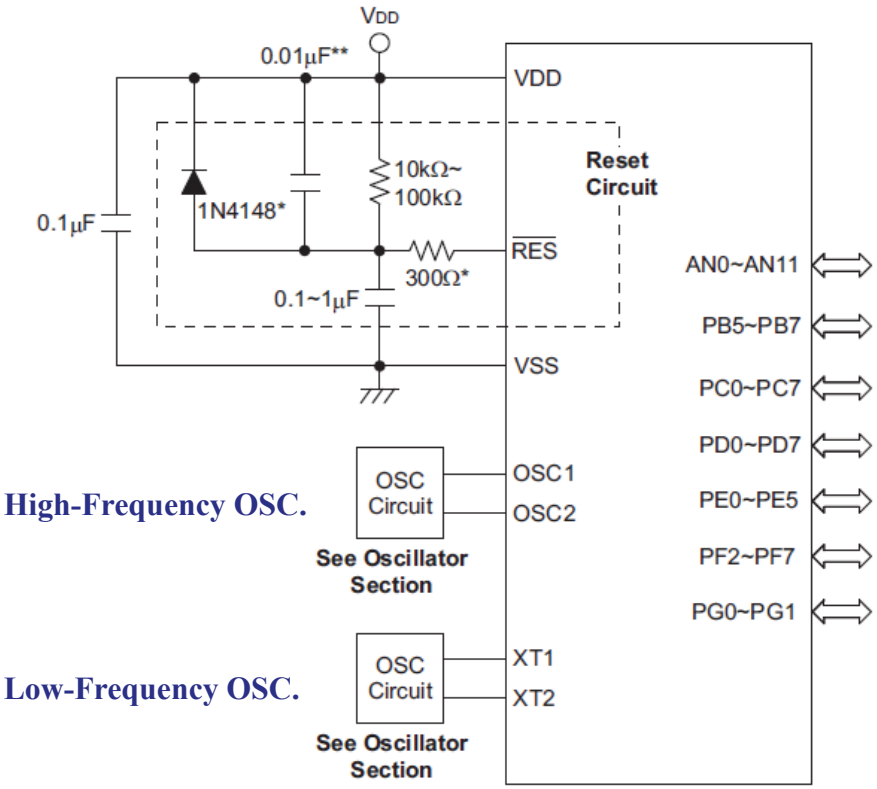
0x01 → 0x08
0x02 → 0x09
0x03 → 0x0a
0x07 → 0x0b
0x0c → 0x04
0x10 → 0x05
0x11 → 0x06
0x12 → 0x13
0x0d → 0x0e
0x0f

NOTE: the program code was written randomly. It means nothing.

Holtek – Architecture



Holtek – Application Circuits



Holtek – Enhanced A/D Flash Type MCU 8-Bit MCU with EEPROM

Most features are common to all devices, the main feature distinguishing them are **Memory capacity**, **I/O count**, **TM features**, **stack capacity** and **package types**. The following table summaries the main features of each device.

Part No.	VDD	Program Memory	Data Memory	Data EEPROM	I/O	Ext. Int.	A/D	Timer Module	Interface (SPI/I²C)	UART	Stack	Package
HT66F20	2.2V~5.5V	1K×14	64×8	32×8	18	2	12-bit×8	10-bit CTM×1, 10-bit STM×1	√	√	4	16DIP/NSOP/SSOP 20DIP/SOP/SSOP
HT66F30	2.2V~5.5V	2K×14	96×8	64×8	22	2	12-bit×8	10-bit CTM×1, 10-bit ETM×1	√	—	4	16DIP/NSOP/SSOP 20DIP/SOP/SSOP 24SKDIP/SOP/SSOP
HT66FU30					14					√		24SKDIP/SOP
HT66F40	2.2V~5.5V	4K×15	192×8	128×8	42	2	12-bit×8	10-bit CTM×1, 10-bit ETM×1, 16-bit STM×1	√	—	8	24/28SKDIP/SOP/SSOP 44LQFP 32/40QFN, 48SSOP/QFN
HT66FU40					34					√		40QFN, 44LQFP, 48SSOP/QFN
HT66F50	2.2V~5.5V	8K×16	384×8	256×8	42	2	12-bit×8	10-bit CTM×2, 10-bit ETM×1, 16-bit STM×1	√	—	8	28SKDIP/SOP/SSOP 44LQFP, 40QFN 48SSOP/QFN
HT66FU50					34					√		44LQFP, 48QFN
HT66F60	2.2V~5.5V	12K×16	576×8	256×8	50	4	12-bit×12	10-bit CTMx2, 10-bit ETMx1, 16-bit STMx1	√	—	12	52QFP, 40QFN, 44LQFP 48SSOP/LQFP/QFN
HT66FU60					42					√		52QFP, 40QFN, 44LQFP 48LQFP/QFN

Holtek – Data Memory

- Divided into **two sections**, the **first** of these is an area of RAM, known as **the Special Function Data Memory (HF66F40、HF66F50、HF66F60: 00H~7FH or HF66F20、HF66F30: 00H~5FH)**.
  - 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 written accessible under program control**.
  - **The overall Data Memory is subdivided into several banks**, the structure of which depends upon the device chosen.
- The **Special Purpose Data Memory registers are accessible in all banks**, with the **exception of the EEC register at address 40H**, **which is only accessible in Bank 1**.
- **Switching between the different Data Memory banks** is achieved **by setting the Bank Pointer** to the correct value.
- The **start address of the Data Memory** for all devices is the **address 00H**.

General Function Data Memory

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	384×8 384=128x3	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH
HT66F60	576×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH 4: 80H~BFH

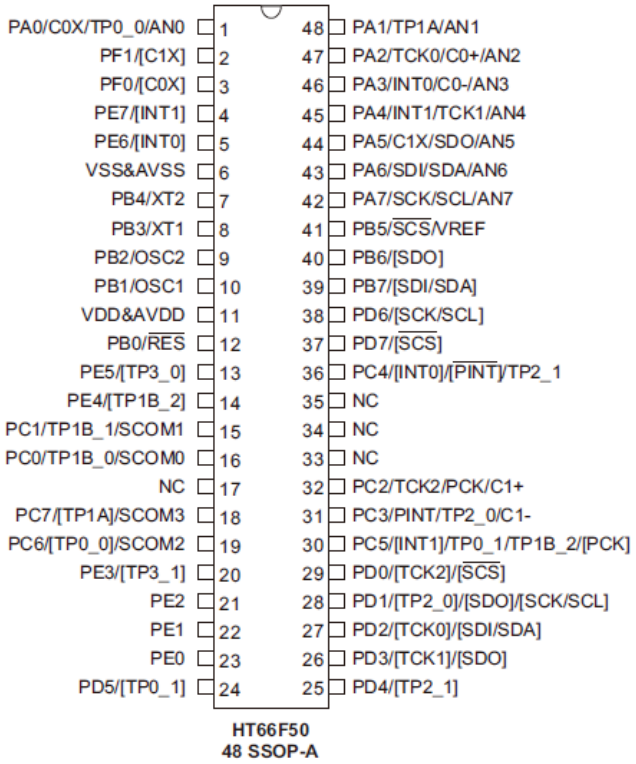
Holtek – RAM Data Memory

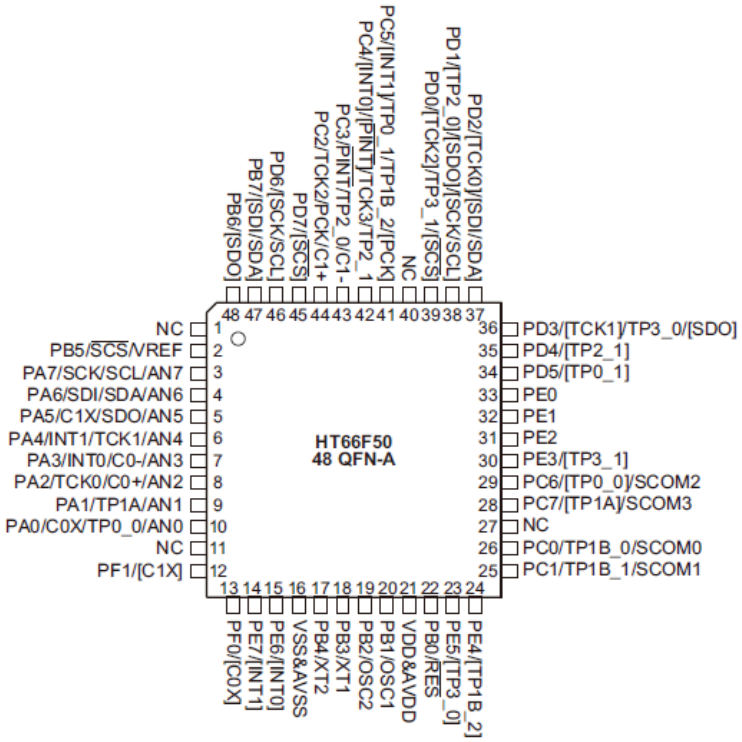
- The **Data Memory is a volatile area** of **8-bit wide RAM internal memory** and is the location where temporary information is stored. The overall Data Memory is subdivided into several banks.
- **Divided into two sections**, the **first** of these is an area of RAM, known as the **Special Function 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	

Holtek – HT66F50 Pins

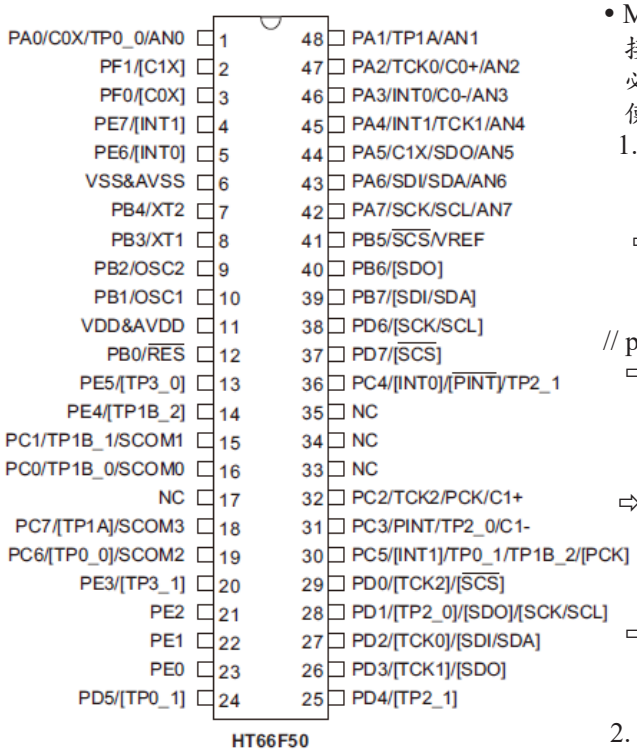




Note:

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 – PIN Function



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.

• MPU的每個接腳均具有多重功能，優先順序為由右至左，也就是說，接腳名稱最右邊的功能，為預設功能，若要將接腳做為純粹數位I/O，必須將其右邊的所有功能關掉。針對目前實驗用到的數位I/O接腳之使用設定，做一些說明。

1. PORT A[7:0]: POR(power on reset)全部預設為ADC的輸入，屬於類比接腳。**必須利用指令下列將PORT A八個接腳的AN功能disable。**

```
acerl =0;
```

⇨**PA2, PA3**第二優先功能為比較器 0的輸入(無pull-high configuration)，若希望規劃是為一般I/O輸出。因此，必須以下列指令關掉此輸入功能。

```
c0sel=0;
// pa2 and pa3 are disconnected c0+ and c0-, respectively, and act as I/O pins
```

⇨**PA0**第二、三優先功能為TM0輸出與比較器 0的輸出，若希望規劃是為一般I/O輸出。因此， 必須以下列指令關掉此輸入功能。

```
_c0os=1; // pa0 is disconnected to c0x and purly acts as an I/Opin
_t0cp0=0; // pa0 is disconnected to TM0 and purly acts as an I/Opin
```

⇨**PA5**第二、三優先功能為SDO輸出與比較器 1的輸出，若希望規劃規劃是為一般I/O輸出。 POR的SDO為浮接狀態，不影響PA5的I/O功能。 因此，必須以下列指令關掉此輸入功能。

```
_c1os=1;  // pa5 is disconnected to c1x and purly acts as an I/Opin
```

⇨**PA1**第二優先功能為TM1輸出，若希望規劃是為一般I/O輸出。因此，必須以下列指令關掉此輸入功能。

```
_t1acp0=0;
```

2. **PC3**: POR的預設為比較器 1的輸入。若希望規劃是為一般I/O輸出，必須利用下列指令使此功能disable。第二優先功能為TM2輸出，預設是開啟的，若希望規劃為一般I/O輸出，也必須關掉此輸入功能。

```
_c1sel=0;
_t2cp0=0;  // 0: turn off, 1: turn on
```

3. **PC4**: POR的預設為TM2的輸出，但是，**預設是關閉。可以直接做為一般I/O輸出。若希望做為TM2的輸出使用，也必須開啟此功能。**

```
_t2cp1=1;
```

Holtek – Configuration Options

Configuration options refer to certain options within the CU that are **programmed into the device during the programming process**. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, **once they are selected they cannot be changed later using the application program**

No.	Options
Oscillator Options	
1	High Speed System Oscillator Selection - fH: 1. HXT 2. ERC 3. HIRC
2	Low Speed System Oscillator Selection - fL: 1. LXT 2. LIRC
3	WDT Clock Selection - fS: 1. f_{SUB} 2. $f_{SYS}/4$
4	HIRC Frequency Selection: 1. 4MHz 2. 8MHz 3. 12MHz
Note: The f_{SUB} and the f_{TBC} clock source are LXT or LIRC selection by the f_L configuration option.	
Reset Pin Options	
5	PB0/RES Pin Options: 1. RES pin 2. I/O pin
Watchdog Options	
6	Watchdog Timer Function: 1. Enable 2. Disable
7	CLRWDT Instructions Selection: 1. 1 instructions 2. 2 instructions

No.	Options
LVR Options	
8	LVR Function: 1. Enable 2. Disable
9	LVR Voltage Selection: 1. 2.10V 2. 2.55V 3. 3.15V 4. 4.20V
SIM Options	
10	SIM Function: 1. Enable 2. Disable
11	SPI - WCOL bit: 1. Enable 2. Disable
12	SPI - CSEN bit: 1. Enable 2. Disable
13	I^2C Debounce Time Selection: 1. No debounce 2. 2 system clock debounce 3. 4 system clock debounce

Holtek – Processor / MIPS Fields

MIPS (Microprocessor without Interlocked Pipeline Stages)

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- *op:* Basic operation of the instruction, traditionally called the **opcode**.
- *rs:* The first register source operand.
- *rt:* The second register source operand.
- *rd:* The register destination operand. It gets the result of the operation.
- *shamt:* Shift amount. (Section 2.6 explains shift instructions and this term; it will not be used until then, and hence the field contains zero in this section.)
- *funct:* Function. This field, often called the *function code*, selects the specific variant of the operation in the op field.

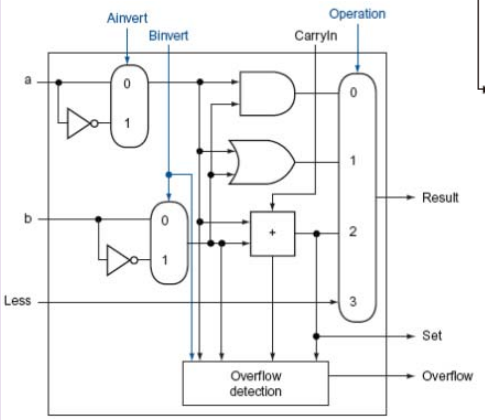
HOLTEK: an executable program code mainly consists of two bytes. The first byte is operation code and the other is operand.

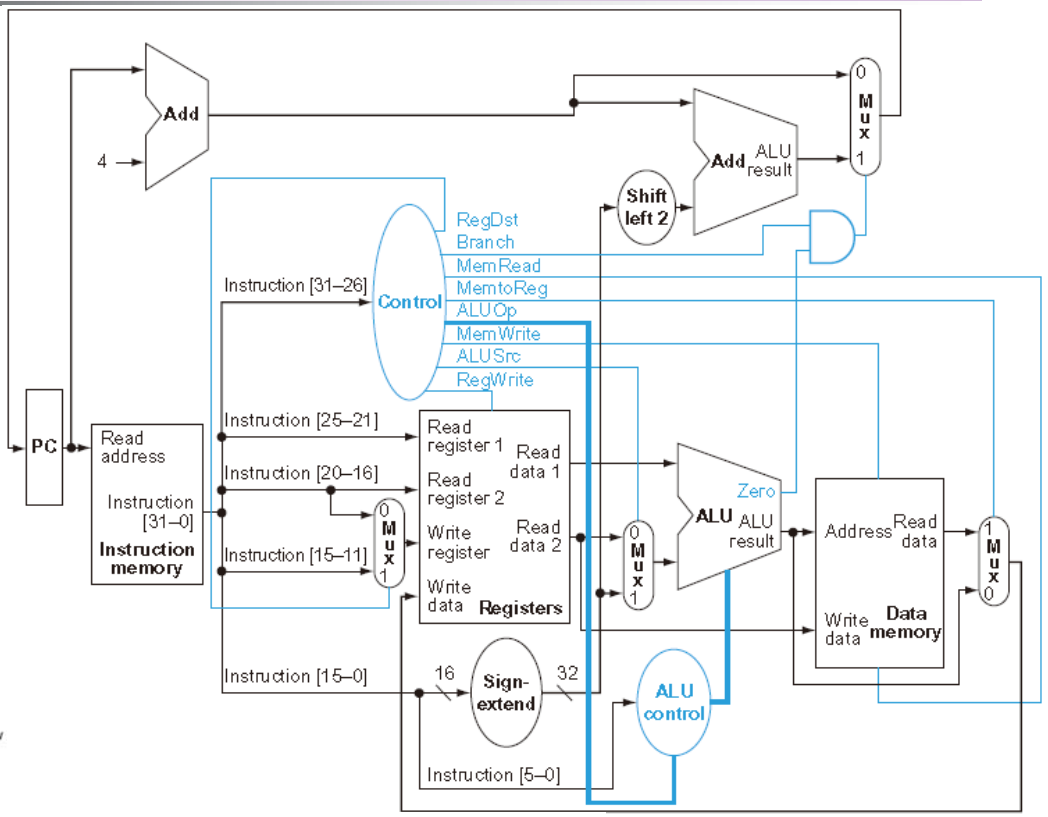
Assembly code	
MOV A, [m]	Move Data Memory to ACC
MOV [m], A	Move ACC to Data Memory
MOV A, x	Move *immediate data* to ACC
RRA [m]	Rotate Data Memory right with result in ACC
XOR A, [m]	Exclusive-OR Data Memory to ACC
XOR A, x	Exclusive-OR *immediate data* to ACC
SUB A, [m]	Subtract Data Memory from ACC
CLR [m]	Clear Data Memory
NOP	No operation
SDZ [m]	Skip if decrement Data Memory is zero
SNZ [m].i	Skip if bit i of Data Memory is not 0

The simple datapath with the control unit.

Control lines

Ainv	Binv	Operation	Function
0	0	0 0	and
0	0	0 1	or
0	0	1 0	add
0	1	1 0	sub
0	1	1 1	slt
1	1	0 0	nor

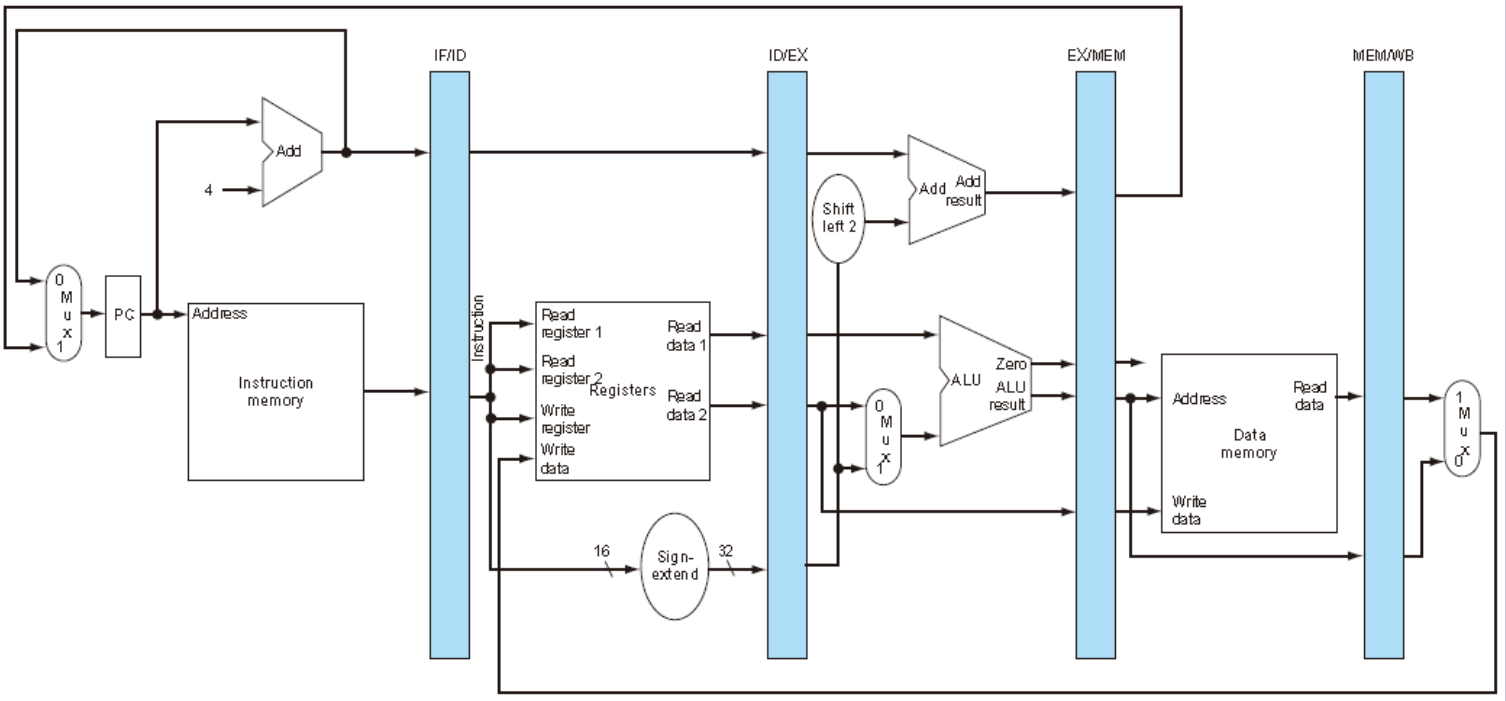




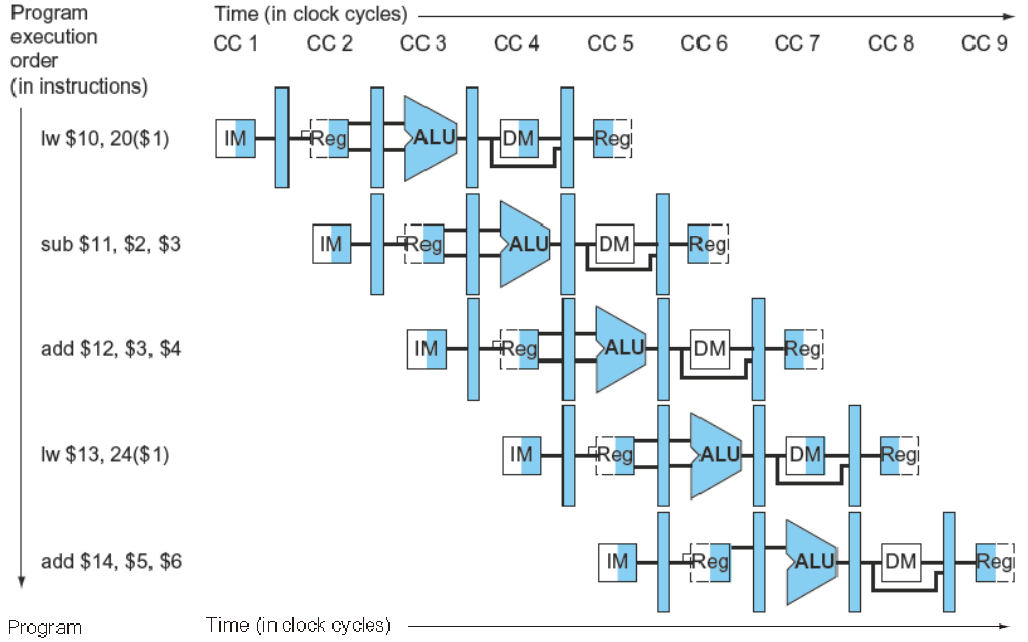
Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

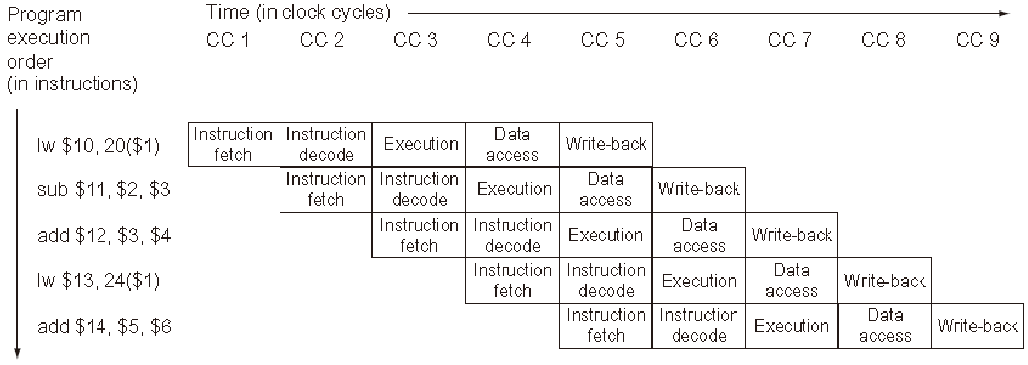
Load word
Store word
Branch on equal

The pipelined version of the datapath



Holtek – Introduction to MIPS Processor / Pipeline



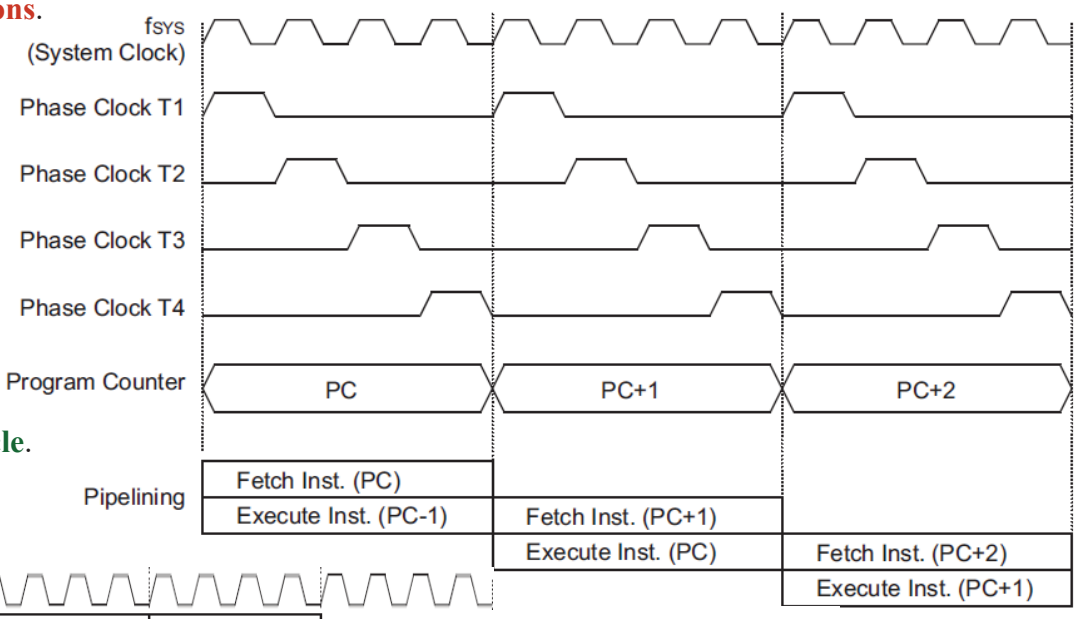


Holtek – System Clocking and Pipelining

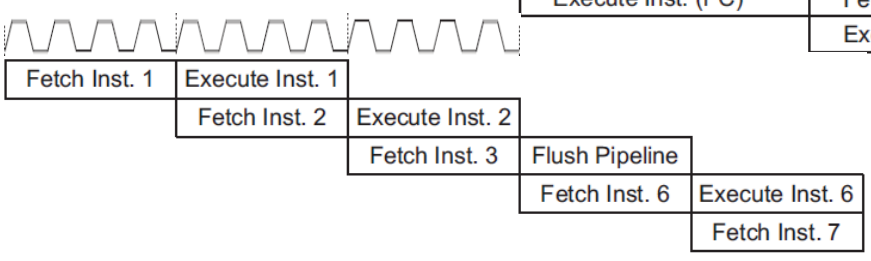
- The main **system clock**, derived from either a **HXT, LXT, HIRC, LIRC or ERC oscillator** is subdivided into **four internally generated non-overlapping clocks, T1~T4.** The **Program Counter is incremented at the beginning of the T1 clock** during which time **a new instruction is fetched**. **The remaining T2~T4 clocks carry out the decoding and execution functions**.

- In this way, one T1~T4 clock cycle forms one instruction cycle.

- Although the fetching and execution of instructions takes place in consecutive **instruction cycles**, the **pipelining structure** of the microcontroller ensures that **instructions are effectively executed in one instruction cycle**.




```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



Holtek – Assembly Code Table

Mnemonic	Description	Cycles	Flag Affected
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRD [m]	Read table to TBLH and Data Memory	2note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note:
1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the CLR WDT1 and CLR WDT2 instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both CLR WDT1 and CLR WDT2 instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

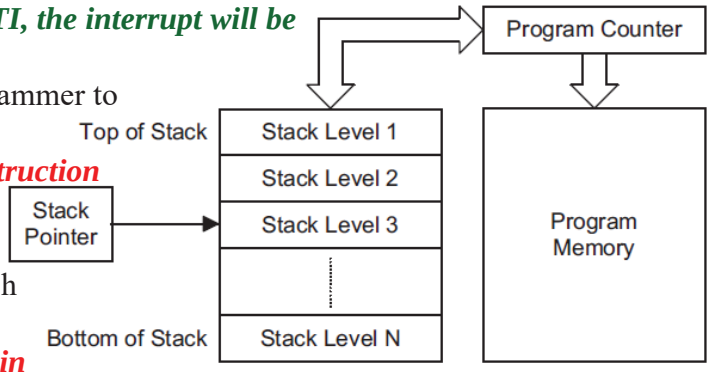
Holtek – Program Counter & Stack

- **During program execution,** the **Program Counter** is used to **keep track of the address of the next instruction** to be executed.
  - ⇨ It is **automatically incremented by one each time an instruction is executed except for instructions, such as JMP or CALL that demand a jump to a non-consecutive Program Memory address**.
- **Only the lower 8 bits**, known as the **Program Counter Low Register**, are **directly addressable by the application program**.
- When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by **loading the required address into the Program Counter**. **For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place** while the correct instruction is obtained.
- The **lower byte** of the Program Counter, known as the **Program Counter Low register or PCL**, is available for program control and is a **readable and writeable register**.
  - ⇨ By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the **jumps are limited to the present page of memory, that is 256** locations.
  - ⇨ **When such program jumps are executed** it should also be noted that **a dummy cycle will be inserted**.
  - ⇨ **Manipulating the PCL register may cause program branching**, so an extra cycle is needed to pre-fetch.

Device	Program Counter	
	Program Counter High Byte	PCL Register
HT66F20	PC9, PC8	PCL7~PCL0
HT66F30	PC10~PC8	
HT66F40	PC11~PC8	
HT66F50	PC12~PC8	
HT66F60	PC13~PC8	

Holtek – Program Counter & Stack

- This is a **special part of the memory** which is used to **save the contents of the Program Counter only**.
- The **stack is neither part of the data nor part of the program space**, and is **neither readable nor writeable**.
- The activated level is indexed by the ***Stack Pointer***, and is **neither readable nor writeable**.
  - ⇨**At a subroutine call** or **interrupt acknowledge signal**, the **contents of the Program Counter are *pushed* onto the stack**.
  - ⇨**At the end of a subroutine** or an **interrupt routine**, **signaled by a return instruction**, **RET or RETI**, the **Program Counter is restored to its previous value from the stack**.
  - ⇨**After a device reset**, the **Stack Pointer will point to the top of the stack**.
- If **the stack is full** and ***an enabled interrupt takes place***, the **interrupt request flag will be recorded but the acknowledge signal will be inhibited**.
  - ⇨***When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced.*** (i.e. ***acknowledge signal will be enabled***)
  - ⇨This feature prevents **stack overflow**, allowing the programmer to use the structure more easily.
  - ⇨However, ***when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow.***
    - **Precautions** should be taken to **avoid such cases** which might ***cause unpredictable program branching***.
  - ⇨***If the stack is overflow, the first Program Counter save in the stack will be lost.***



Device	Stack Levels
HT66F20/HT66F30	4
HT66F40/HT66F50	8
HT66F60	12

Holtek – Arithmetic and Logic Unit (ALU)

- The **arithmetic-logic unit or ALU** is a critical area of the microcontroller that **carries out arithmetic and logic operations** of the instruction set.
- Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register.
- As these ALU calculation or operations may result in **carry**, **borrow** or **other status changes**, the **status register will be correspondingly updated to reflect these changes**.
- The ALU supports the following functions:

⇨**Arithmetic operations** :

ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA

⇨**Logic operations**:

AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA

⇨**Rotation**

RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC

⇨**Increment and Decrement**

INCA, INC, DECA, DEC

⇨**Branch decision**,

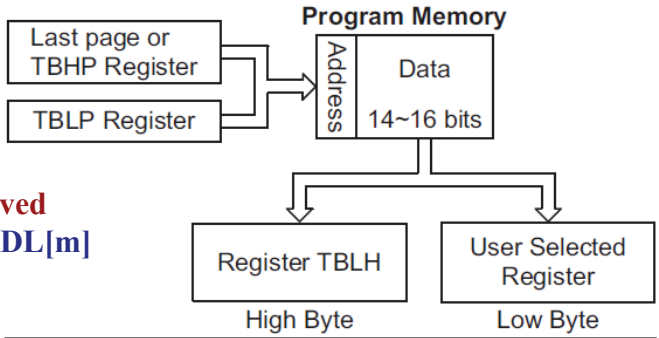
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Holtek – Flash Program Memory / Look-up Table

- The **Program Memory** is the location **where the user code or program is stored**. For this device series, the Program Memory is **Flash type**, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device.
- The **Program Memory** is **addressed by the Program Counter** and **also contains data, table information and interrupt entries**. **Table data**, which can be setup in any location within the Program Memory, is **addressed by a separate table pointer register**.

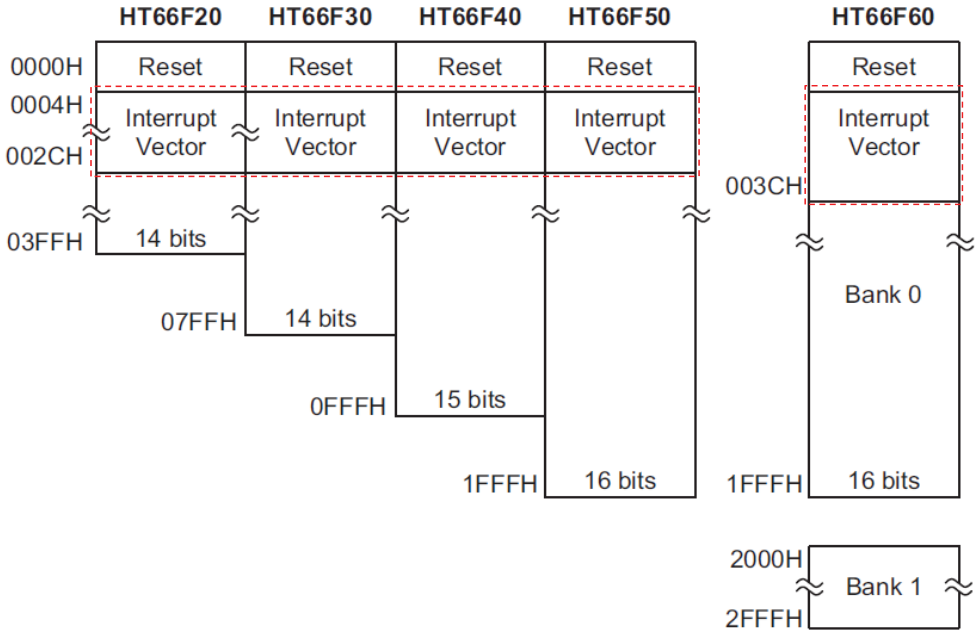
Look-up Table

- **Any location** within the Program Memory can be defined as a **look-up table** where programmers can **store fixed data**.
  - ⇨To use the look-up table, the **table pointer** must first be setup by **placing the address** of the look up data to be retrieved **in the table pointer register**, **TBLP** and **TBHP**.
    - These registers define the total address of the look-up table.
  - ⇨After setting up the table pointer, the **table data can be retrieved from the Program Memory using the TABRD[m] or TABRDL[m] instructions**, respectively.
    - When the instruction is executed, the **lower order table byte** from the Program Memory will be transferred to the user defined **Data Memory register [m]** as specified in the instruction. The **higher order table data** byte from the Program Memory will be transferred to the **TBLH special register**.



Device	Capacity	Banks
HT66F20	1K×14	0
HT66F30	2K×14	0
HT66F40	4K×15	0
HT66F50	8K×16	0
HT66F60	12K×16	0, 1

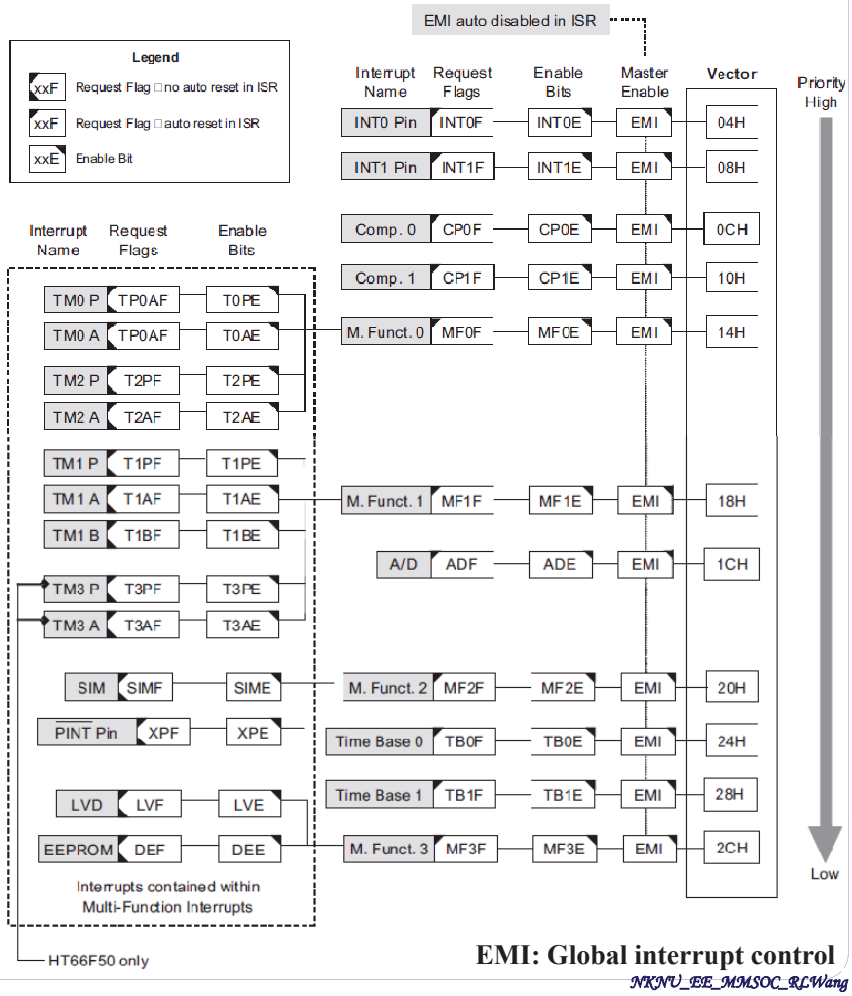
Holtek – Program Memory Structure



The HT66F60 has its Program Memory divided into two Banks, Bank 0 and Bank 1. The required Bank is selected using Bit 5 of the BP Register.

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.

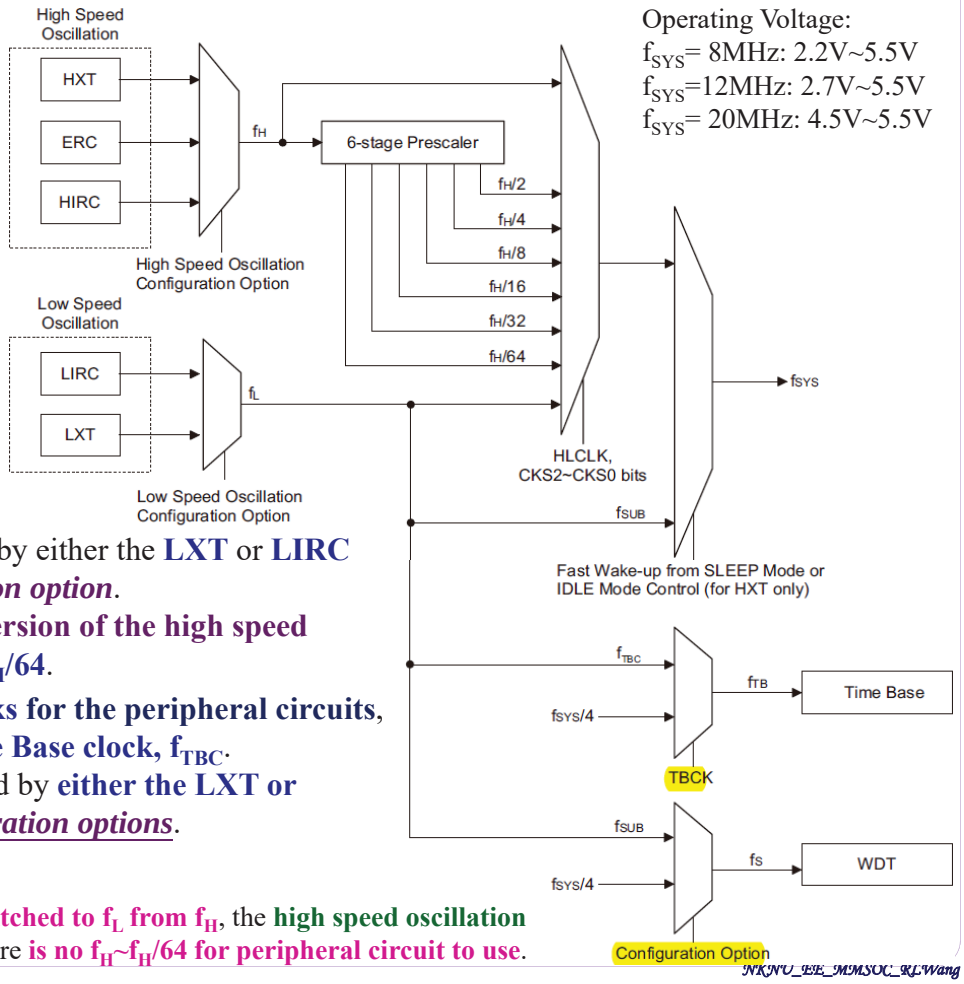


EMI: Global interrupt control

Holtek – System Clock Sources

- The device has **many different clock sources for** both the **CPU** and **peripheral function** operation.
- The **main system clock**, can come from either a **high frequency, f_H**, or **low frequency, f_L**, source, and is **selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register**.
  - The **high speed system clock** can be sourced from either an **HXT**, **ERC** or **HIRC oscillator**, ***selected via a configuration option***.
  - The **low speed system clock** source can be sourced from **internal clock f_L**.
    - If **f_L** is selected then it can be sourced by either the **LXT** or **LIRC oscillators**, ***selected via a configuration option***.
  - The **other choice**, which is a **divided version of the high speed system oscillator** has a range of **$f_H/2$~$f_H/64$**.
- There are **two additional internal clocks for the peripheral circuits**, the **substitute clock, f_{SUB}**, and the **Time Base clock, f_{TBC}**.
  - Each of these internal clocks are sourced by **either the LXT or LIRC oscillators**, ***selected via configuration options***.

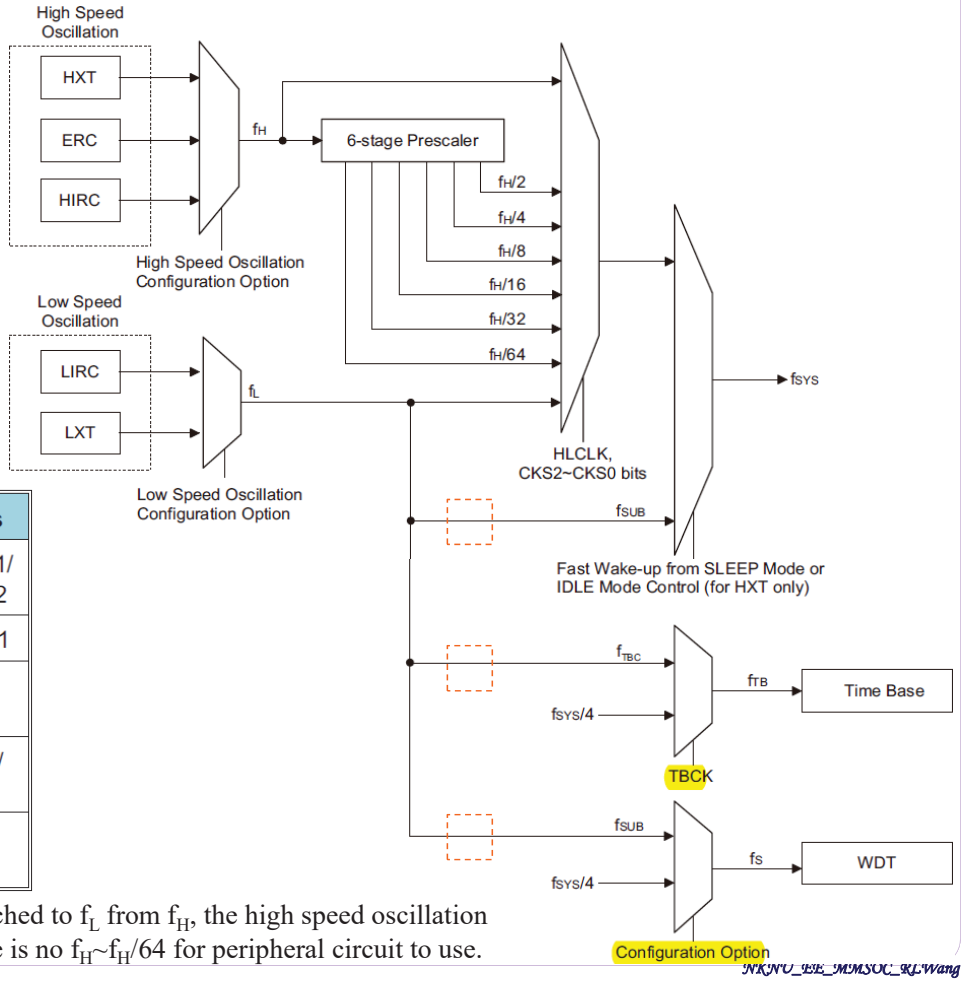
Operating Voltage:
f_{SYS}= 8MHz: 2.2V~5.5V
f_{SYS}=12MHz: 2.7V~5.5V
f_{SYS}= 20MHz: 4.5V~5.5V



Note: **When the system clock source f_{SYS} is switched to f_L from f_H**, the **high speed oscillation will stop to conserve the power**. Thus there **is no f_H~$f_H/64$ for peripheral circuit to use**.

Holtek – System Clock Sources

- The f_{SUB} clock is used to *provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times*.
  - Together with $f_{SYS}/4$ it is also used as one of the clock sources for the Watchdog timer.
- The f_{TBC} clock is used as a source *for the Time Base interrupt functions and for the TMs*.

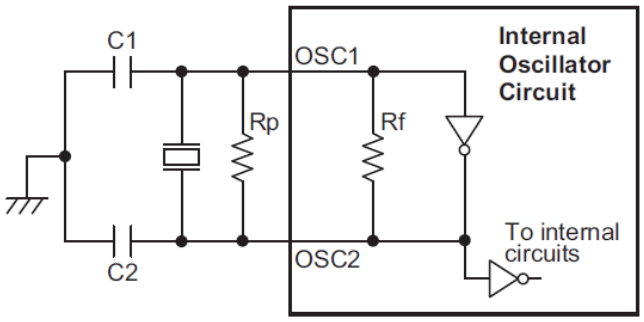


Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~ 20MHz	OSC1/ OSC2
External RC	ERC	8MHz	OSC1
Internal High Speed RC	HIRC	4, 8 or 12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/ XT2
Internal Low Speed RC	LIRC	32kHz	—

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~$f_H/64$ for peripheral circuit to use.

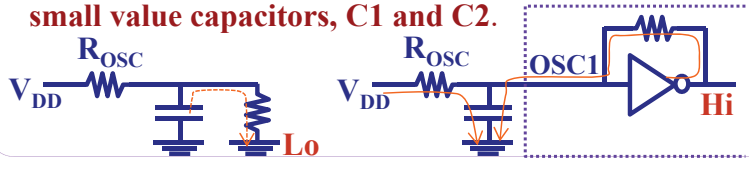
Holtek – System Clock Sources

Crystal/Resonator Oscillator HXT



1. Rp is normally not required.C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7 pF.

- The **External** Crystal/ Ceramic System Oscillator is one of the **high frequency oscillator choices**, which is **selected via configuration option**.
- For most crystal oscillator configurations, the **simple connection of a crystal across OSC1 and OSC2** will **create the *necessary phase shift and feedback for oscillation***, without requiring external capacitors.
- However, for some crystal types and frequencies, **to ensure oscillation**, it may be **necessary to add two small value capacitors, C1 and C2**.

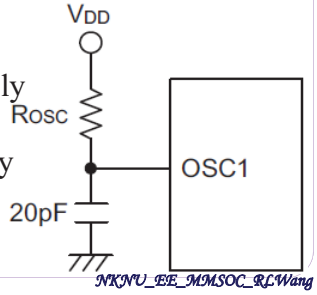


Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

External RC Oscillator ERC

- Using the ERC **oscillator only requires that a resistor, with a value between 56k and 2.4M, is connected between OSC1 and VDD**, and a capacitor is connected between OSC1 and ground, providing a low cost oscillator configuration.
- As a resistance/frequency reference point, it can be noted that with an external **120k** resistor connected and with a 5V voltage power supply and temperature of 25C degrees, the oscillator will have a frequency of **8MHz** within a tolerance of 2%.



Holtek – System Clock Sources

Internal RC Oscillator HIRC

- The internal RC oscillator is **a fully integrated system oscillator requiring no external components**.
- The internal RC oscillator has **three fixed frequencies of either 4MHz, 8MHz or 12MHz.**

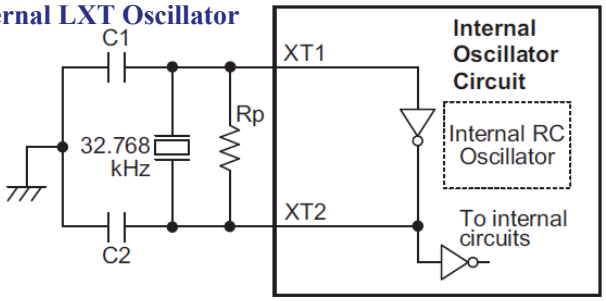
External 32.768kHz Crystal Oscillator LXT

- The **External 32.768kHz Crystal System Oscillator** is one of the low frequency oscillator choices, which is **selected via configuration option**.
- This clock source has a fixed frequency of 32.768kHz **and requires a 32.768kHz crystal to be connected between pins XT1 and XT2**.
- The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation.

Internal 32kHz Oscillator LIRC

- The **Internal 32kHz System Oscillator is one of the low frequency oscillator choices**, which is **selected via configuration option**.
- It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. at a power supply of 5V and at a temperature of 25C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

External LXT Oscillator



Note: 1. Rp, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

- The **LXT oscillator** can function in one of **two modes**, the **Quick Start Mode** and the **Low Power Mode**. **The mode selection is executed using the LXTLP bit in the TBC register**.
- **After power on**, the **LXTLP bit** will be automatically cleared to **zero** ensuring that the **LXT oscillator is in the Quick Start operating mode**.
- However, **after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high**

Holtek – System Operation Modes

- There are **six different modes of operation** for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application.
- There are **two modes allowing normal operation** of the microcontroller, the **NORMAL Mode** and **SLOW Mode**.
- The remaining four modes, the **SLEEP0**, **SLEEP1**, **IDLE0** and **IDLE1 Mode** are used when the microcontroller CPU is switched off to conserve power.

Depend on WDT clock source (f_s) selection
If f_s=f_{SUB}, f_s is on, else f_s=f_{SYS}/4 and f_s is off

Operating Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{TBC}
NORMAL Mode	On	f_H~f_H/64	On	On	On
SLOW Mode	On	f_L	On	On	On
IDLE0 Mode	Off	Off	On	On/Off	On
IDLE1 Mode	Off	On	On	On	On
SLEEP0 Mode	Off	Off	Off	Off	Off
SLEEP1 Mode	Off	Off	On	On	Off

SMOD[1] IDLEN bit=1

IDLEN bit=0

WDT or LVD on/off

IDLE mode: WDT is still on

- **FSYSON(=WDTC[7]) bit ='1' :** in the **IDLE1** mode, the **CPU will stop running but the system clock will continue to keep the peripheral functions operational.**
- **FSYSON(=WDTC[7]) bit = '0':** in **IDLE0** mode, the **CPU and the system clock will all stop**.

Holtek – System Operation Modes / SMOD Register

A single register, **SMOD**, is used **for overall control of the internal clocks within the device**.

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 CKS2~CKS0: The system clock selection when **HLCLK is "0"**

000: f_L (f_{LXT} or f_{LIRC}) 001: f_L (f_{LXT} or f_{LIRC})
010: $f_H/64$ 011: $f_H/32$
100: $f_H/16$ 101: $f_H/8$
110: $f_H/4$ 111: $f_H/2$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 FSTEN: Fast Wake-up Control (**only for HXT**)

0: Disable 1: Enable

This is the **Fast Wake-up Control bit** which **determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock** to provide a faster wake up time as the f_{SUB} clock is available.

Bit 3 LTO: Low speed system oscillator ready flag

0: Not ready
1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The **flag will be low when in the SLEEP0 Mode** but **after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the LXT oscillator is used** and **1~2 clock cycles if the LIRC oscillator is used**.

Holtek – System Operation Modes / SMOD Register

A single register, **SMOD**, is used **for overall control of the internal clocks within the device**.

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 2 HTO: High speed system oscillator ready flag

0: Not ready
1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. **This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable**. Therefore this flag will always be read as "1" by the application program after device power-on. **The flag will be low when in the SLEEP or IDLE0 Mode** but **after a wake-up has occurred**, the flag will **change to a high level after 1024 clock cycles if the HXT oscillator is used** and **after 15~16 clock cycles if the ERC or HIRC oscillator is used**.

Bit 1 IDLEN: IDLE Mode control

0: Disable (→ SLEEP mode)
1: Enable (→ IDLE mode)

This is the **IDLE Mode Control bit** and ***determines what happens when the* HALT *instruction is executed***. **If this bit is high**, when a HALT instruction is executed the device will **enter the IDLE Mode**. In the **IDLE1** Mode the **CPU will stop running but the system clock will continue** to keep the peripheral functions operational, if **FSYSON(=WDTC[7]) bit is high**. **If FSYSON bit is low**, the CPU and the system clock will all stop in **IDLE0** mode. **If the bit is low** the device will **enter the SLEEP Mode** when a HALT instruction is executed.

Bit 0 HLCLK: system clock selection

0: $f_H/2$~$f_H/64$ or f_L 1: f_H

This bit is used to select if the f_H clock or the $f_H/2$~$f_H/64$ or f_L clock is used as the system clock**. When the bit is high the f_H clock will be selected** and **if low the $f_H/2$~$f_H/64$ or f_L clock will be selected**. When **system clock switches from the f_H clock to the f_L clock and the f_H clock will be automatically switched off to conserve power.**

Holtek – Input/Output Ports

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWU	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PBPU	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PCPU	D7	D6	D5	D4	D3	D2	D1	D0
PC	D7	D6	D5	D4	D3	D2	D1	D0
PCC	D7	D6	D5	D4	D3	D2	D1	D0
PDPU	D7	D6	D5	D4	D3	D2	D1	D0
PD	D7	D6	D5	D4	D3	D2	D1	D0
PDC	D7	D6	D5	D4	D3	D2	D1	D0
PEPU	D7	D6	D5	D4	D3	D2	D1	D0
PE	D7	D6	D5	D4	D3	D2	D1	D0
PEC	D7	D6	D5	D4	D3	D2	D1	D0
PFPU	—	—	—	—	—	—	D1	D0
PF	—	—	—	—	—	—	D1	D0
PFC	—	—	—	—	—	—	D1	D0

Holtek – Input/Output Ports

Register	Reset (Power-on)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE)
BP	---- --00	---- --00	---- --00	---- --uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
WDTC	0111 1010	0111 1010	0111 1010	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TMPC0	1001 --01	1001 --01	1001 --01	uuuu --uu
TMPC1	--01 --01	--01 --01	--01 --01	--uu --uu

Holtek – Input/Output Ports

- **ACERL Register**

Bit	7	6	5	4	3	2	1	0
Name	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7 **ACE7: Define PA7 is A/D input or not**
0: Not A/D input
1: A/D input, AN7

Bit n **ACEn: Define PAn is A/D input or not**
0: Not A/D input
1: A/D input, Ann ; n=0~7

- **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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 7~0 **PAWU: Port A bit 7 ~ bit 0 Wake-up Control**
0: Disable 1:enable **wake up the microcontroller as one of the Port A pins changes from high to low.**

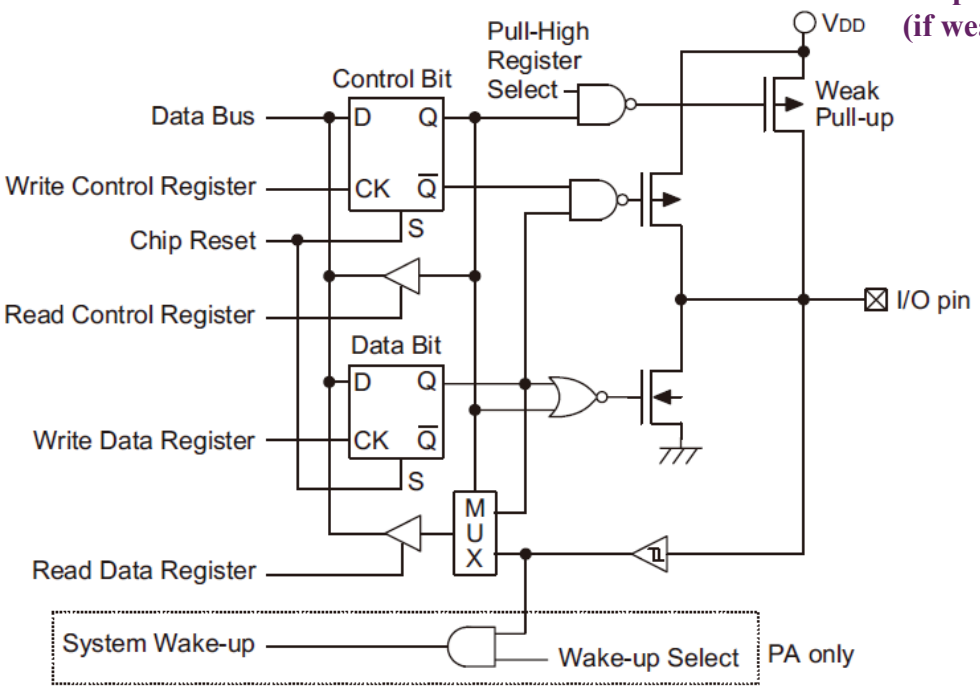
- **PAPU Register** **1: enable pull-up p-MOSFET**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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

Holtek – I/O Structure

Control register : PAC
Data register : PA

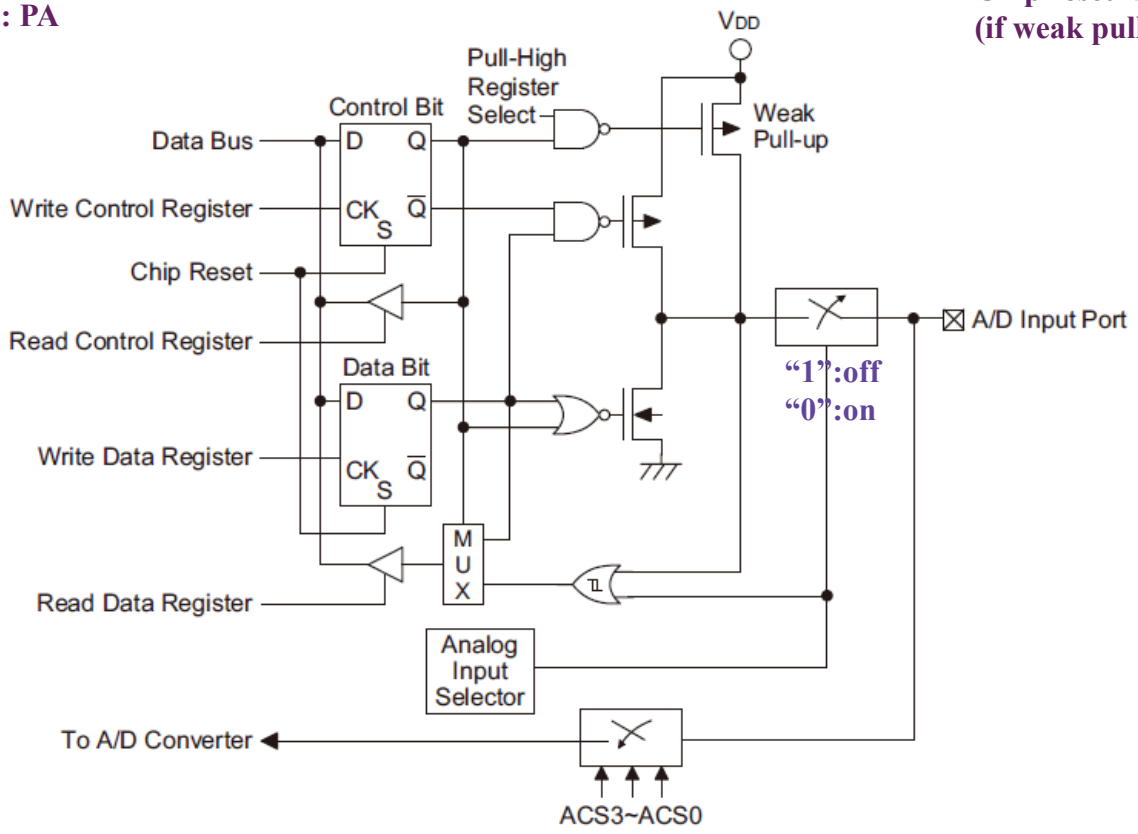
Chip reset → I/O high-Z
(if weak pull-up is off)



Holtek – I/O Structure

Control register : PAC
Data register : PA

Chip reset → I/O high-Z
(if weak pull-up is off)



Holtek – I/O Structure (Pull-up Resistor & Port A Wake-up)

- **PAPU Register**

To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an **internal pull-high resistor**. These pull-high resistors are selected using registers PAPU~PGPU, and are implemented using weak PMOS transistors.

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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 7~0 I/O Port bit 7 ~ bit 0 Pull-High Control
0: Disable 1: Enable

- **Port A Wake-up (PAWU Register)**

The **HALT instruction** forces the microcontroller **into the SLEEP or IDLE Mode which preserves power**, a feature that is important for battery and other low-power applications.

Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low.

⇨ This function is especially suitable for applications that can be woken up via external switches.

⇨ Each pin on Port A can be selected individually to have this wake-up feature using the **PAWU register**.

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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

0: Disable 1: Enable

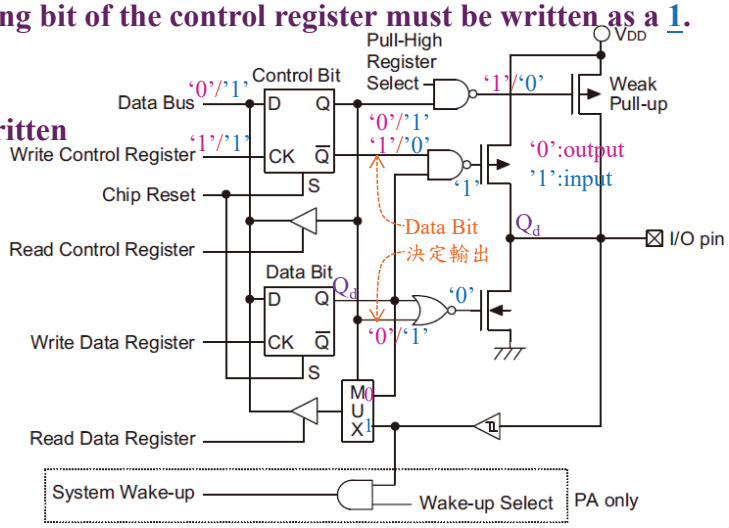
Holtek – I/O Structure (I/O Port Control Registers)

I/O Port Control Registers (PAC Register)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

0: Output 1: Input

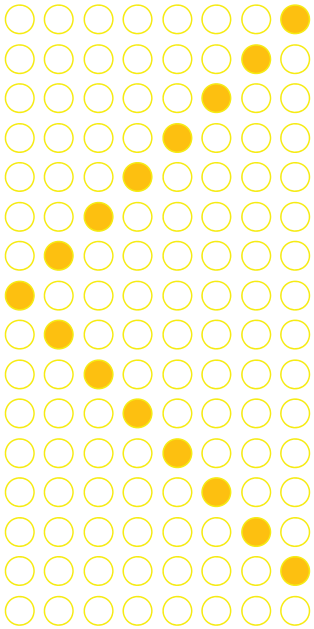
- Each I/O port has its own control register known as PAC~PGC, to control the input/output configuration.
  - ⇨ With this control register, each CMOS output or input can be reconfigured dynamically under software control.
- Each pin of the I/O ports is directly mapped to a bit in its associated port control register.
- **For the I/O pin to function as an input, the corresponding bit of the control register must be written as a 1.**
  - ⇨ This will then allow the logic state of the input pin to be directly read by instructions.
- **When the corresponding bit of the control register is written as a 0, the I/O pin will be setup as a CMOS output.**
- **If the pin is currently setup as an output, instructions can still be used to read the output register.**
  - ⇨ However, it should be noted that the program will in fact **only read the status of the output data latch** and **not the actual logic status of the output pin**.



Holtek – Demo Lab02

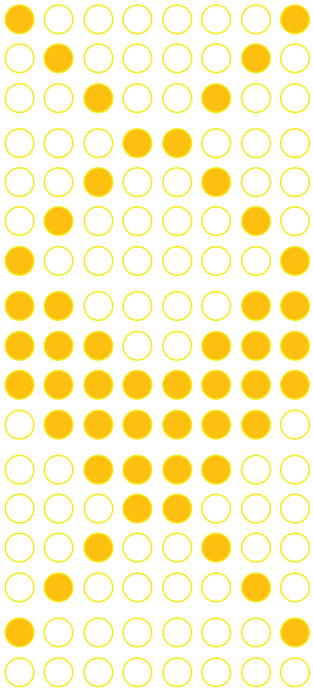
弱光

約0.5秒閃一次



強光

約1秒左右閃一次



```
const unsigned char LEDPatternA[16] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20,
                                        0x10, 0x08, 0x04, 0x02, 0x01, 0x00} ;
const unsigned char LEDPatternB[17] = { 0x81, 0x42, 0x24, 0x18, 0x24, 0x42, 0x81, 0xc3, 0xe7, 0xff,
                                        0x7e, 0x3c, 0x18, 0x24, 0x42, 0x81, 0x00} ;
```

Holtek – Lab. For System Clock, Memory, I/O Port

LAb111_L01_system_clock_LED.c

```
// This is a LED sweeping program。
// In this program, memory, system clock and port are the main lecture topic
// Array declaration with/without "const" maps to the usage of ROM/RAM memory
// Interconnect lines on the Practice Board :
//   (1) CdS light sensor --> MCU : PA2-->PF0, V_D-->VDD, G_D-->VSS
//   (2) LEDs --> MCU :  PC --> PC
//   Under no bright light, LEDs runs by the LEDTab1[1] pattern
//   under bright light, LEDs runs by the LEDTab1[2] pattern
#include "HT66F50.h"
#define LedPort _pc // pc 埠
const unsigned char LEDPatternA[16] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x40, 0x20,
                0x10, 0x08, 0x04, 0x02, 0x01, 0x00} ;
const unsigned char LEDPatternB[17] = { 0x81, 0x42, 0x24, 0x18, 0x24, 0x42, 0x81, 0xc3, 0xe7, 0xff,
                0x7e, 0x3c, 0x18, 0x24, 0x42, 0x81, 0x00} ;
const unsigned int Anum=16;
const unsigned int Bnum=17;
const unsigned int Adelay = 10;
const unsigned int Bdelay = 20;
void delay (unsigned int n)
{
        unsigned int idy1, idy2 ;
        for(idy1 = 0; idy1 < n ; idy1++)   //
                {
                        GCC_NOP();
                        GCC_NOP();
                        for(idy2=0; idy2<4 ; idy2++)
                        {
                                GCC_NOP();
                                GCC_NOP();
                                GCC_NOP();
                        }
                }
}
```

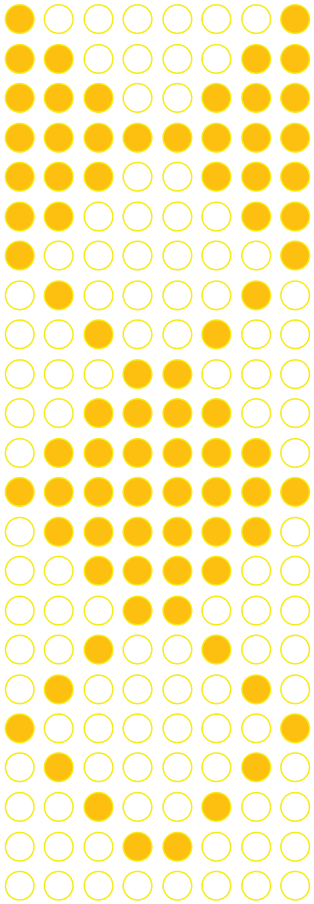

Holtek – Lab. For System Clock, Memory, I/O Port

```
void main(void) // 主函式
{
/* System Operation Mode: _smod */
//              _smod=0b11000000;          //[0]= 1: fH;    0: fH/2~fH/64 or fL, chosen by [7:5]
//      [7:5]= 000: fL (fLXT or fLIRC)      001: fL (fLXT or fLIRC)
//          010: fH/64;         011: fH/32;    100: fH/16      101: fH/8;      110: fH/4;      111: fH/2
        _smod=0b00000000;          // [7:5]=100-->fH/16,110-->fH/4,000,001->fL,[0]=1-->fH
        _pcc = 0;       // 設定 PC 埠為輸出, 0:輸出; 1:輸入
        _pfc0 = 1 ; // 設定 PF0接腳為輸出, 0:輸出; 1:輸入
/*multifunctional I/O pins: must close high-priority functions to assign a desired function to a I/O pin */
/* the rightmost function is the highest-priority function                                          */
//  PA0/C0X/TP0_0/AN0 ; PC5/[INT1]/TP0_1/TP1B_2/[PCK]  ;  _t1bcp2 default=0;
//  PC2/TCK2/PCK/C1+;   PC3/PINT/TP2_0/C1- ; PC4/[INT0]/[PINT]/TCK3/TP2_1
        _c1sel=0;  //set PC3/C1- as I/O
        _t0cp1=0;  //t0cp0:pa0 (default=1), t0cp1:pc5 (default=0);
        LedPort = 0x00 ;                // LEDs port
        int i;
        while(1)
        {
                if (_pf0 == 0)
                {
                  for(i=0; i<Anum; i++)
                        {
                                LedPort = LEDPatternA[i];
                                delay(Adelay);
                        }


                }
                else
                {
                        for(i=0; i<Bnum; i++)
                        {
                                LedPort = LEDPatternB[i];
                                delay(Bdelay);
                        }
                }
        }
}
```

強光

約1秒閃一次



弱光

約2秒左右閃一次