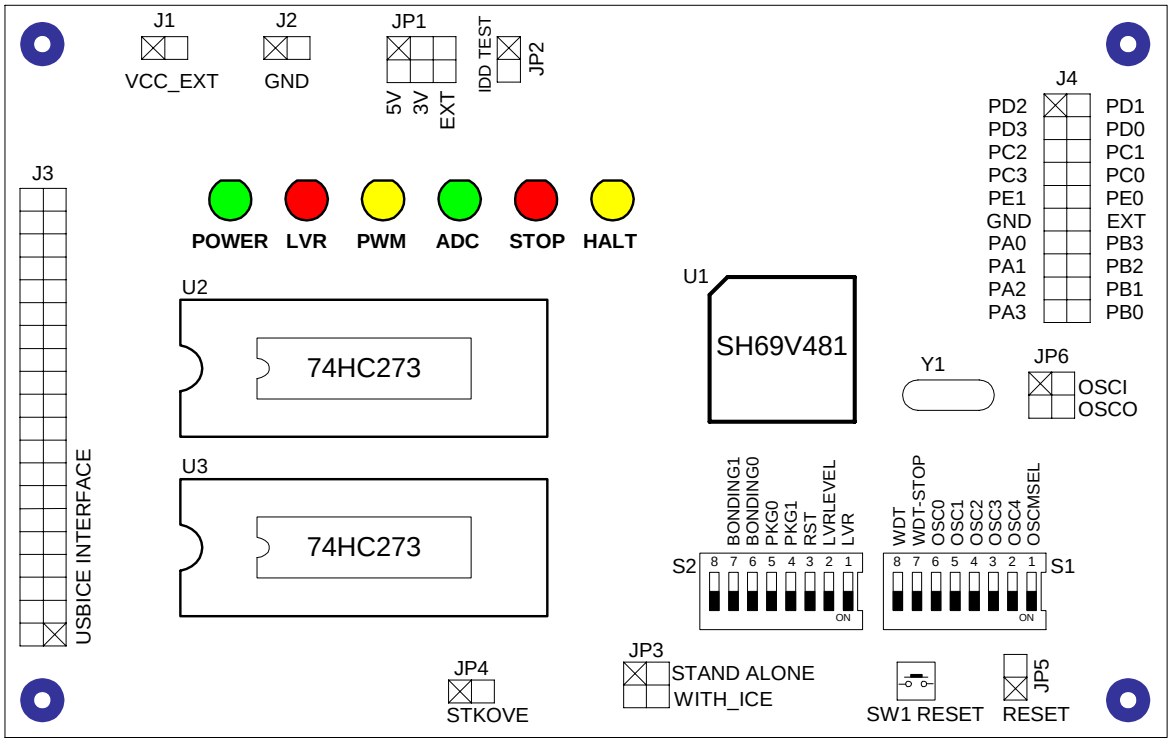
SH69P481 EVB

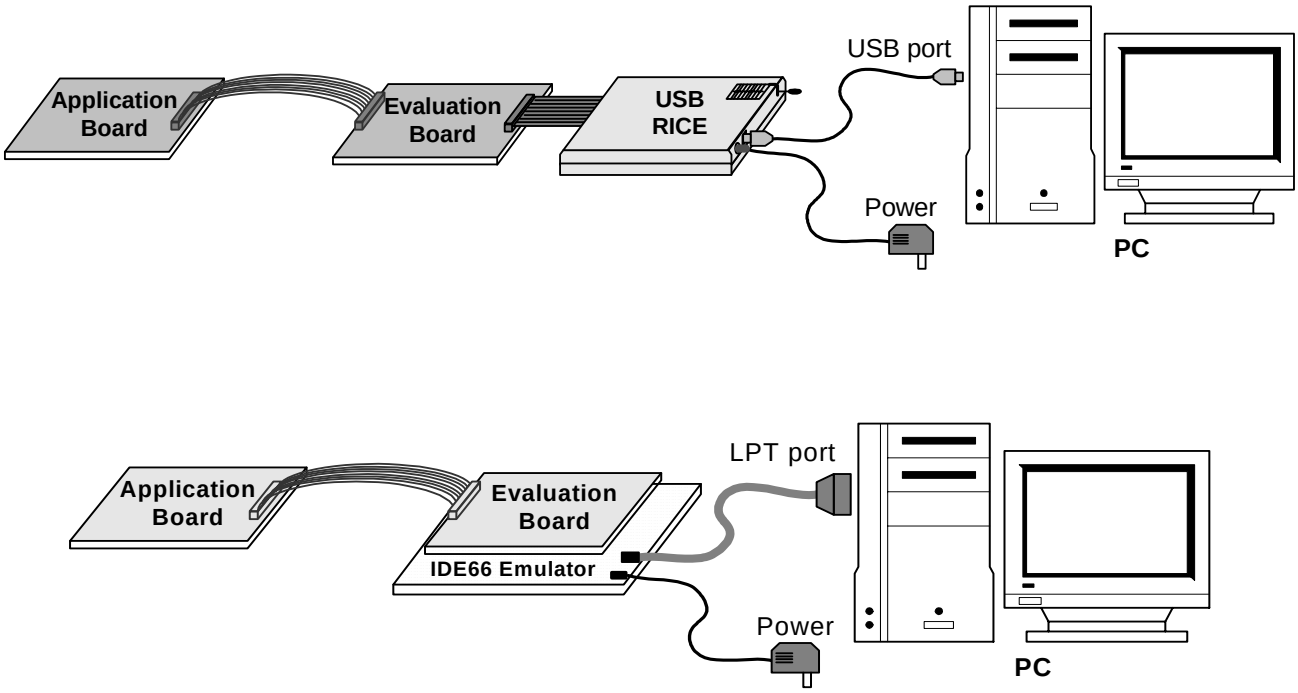
Application Notes for SH69P481 EVB

SH69P481 EVB

The SH69P481 EVB is used to evaluate the SH69P481 chip's function for the development of application program. It contains of a SH69V481 chip to evaluate the functions of SH69P481 including the ADC input and the PWM output. The following figure shows the placement diagram of SH69P481 EVB.

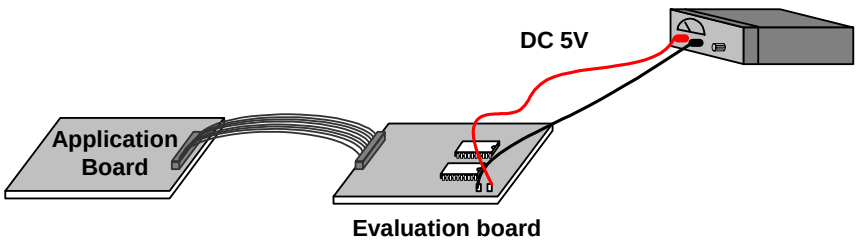


There are two configurations of SH69P481 EVB in application development: ICE mode and stand-alone mode.
In the ICE mode, the SH66xx ICE (motherboard) is connected to the SH69P481 EVB by the ICE interface.



(a) ICE mode

In the standalone mode, the SH69P481 EVB is no longer connected to the motherboard. But the EPROM chip has must be connected to EPROM sockets of the SH69P481 EVB. The EPROM stores the application program; they may be the 27512.



(b) Stand-alone mode

The process of your program's evaluation on SH69P481 EVB

User can use **Sino Wealth Rice66 Integrated Development Environment (IDE)** to emulate the program and produce the obj file. Rice66 IDE is a real-time in-circuit emulator program. It provides real-time and transparent emulation support for the SH6X series 4-bit microcontroller. And integrate assembler can create binary (*.obj) file and the other files.

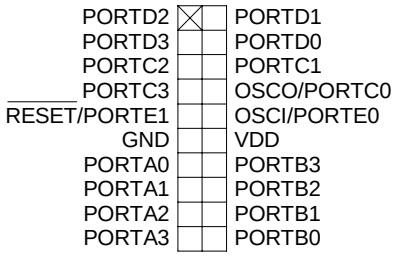
Use Flash (or EPROM) In standalone mode

Rice66 IDE is built-in with an object file depart function. The command "Split object file" can separate the one 16 bits object file into two 8 bits files, which contain the high and low bytes respectively.

Write the high/low byte obj file to Flash (or EPROM) and insert them to EVB (ROMH and ROML). Then, user can evaluate the program in standalone mode.

SH69P481 EVB Interface Connector: (Top View from EVB):

□ **User's interface connector: J4 (Top View from EVB)：**



Most important:

Incorrect power input (The GND is connected to VCC pin J1, and VCC is connected to GND pin J2) will hurt or breakdown the EV board permanently.

□ **External VCC input for stand alone mode:**

J1, J2 -The external power input when the EVB worked in stand-alone mode. The voltage of Vcc must be 5V±5%.

□ **Interface to test the EV chip operating current**

JP2 - User can test the EV chip current through JP2

Note: In ICE mode, the current value is correct only when the IDE66 runs in external clock from EVB mode. (Select the "external clock from EVB" in OSC Frequency configuration manual.)

JP6 (OSCO/PORTC0, OSCI/PORTE0 port type select):

- If select "Internal RC oscillator", short OSCI end and OSCO end of JP6, PORTC0 and PORTE0 can be acquired.
- If select "External RC oscillator", short OSCO end of JP6, PORTC0 can be acquired.
- If select "Crystal oscillator" or "Ceramic resonator", do not short any end.

Jumper setting:

JP1	EV chip power supply select
Short at 3V position	The power of EV chip is set as internal 3V power source
Short at 5V position	The power of EV chip is set as internal 5V power source
Short at EXT position	The EV chip use external power supply that was input from EXT pin.

JP3	EVB ICE/Stand-alone mode select
Short at Stand-alone position	Select stand-alone mode. (The system clock is provided by the on board oscillator.)
Short at With-ICE position	Select with-ICE mode. (The system clock is provided by the ICE.)

JP4	STACK overflow select
Short	The stack overflow function in the ICE mode will on
Open	The stack overflow function in the ICE mode will off

S1

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Remarks
WDT	WDT_ STOP	OSC0	OSC1	OSC2	OSC3	OSC4	OSC MEL	
X	X	Off	Off	Off	Off	Off	Off	External clock
X	X	Off	Off	Off	Off	Off	On	Internal RC 4M
X	X	Off	Off	Off	Off	On	X	Internal RC 8M
X	X	Off	Off	Off	On	X	X	Internal RC 8M
X	X	On	Off	On	Off	Off	Off	Crystal oscillator 8M or 10M,Ceramic resonator 8M or 12M
X	X	On	On	On	Off	Off	On	Crystal oscillator 400K,Ceramic resonator 400K、455K & 2M
X	X	On	On	On	Off	On	Off	Ceramic resonator 1M
X	X	On	On	On	Off	On	On	32.768KHz Crystal oscillator ,Internal RC 4M
X	X	On	On	On	On	Off	Off	32.768KHz Crystal oscillator ,Internal RC 8M
X	X	On	Off	On	On	Off	On	Crystal oscillator 4M,Ceramic resonator 4M
X	X	On	Off	On	On	On	On	Crystal oscillator 8M or 10M,Ceramic resonator 8M or 12M
X	Off	X	X	X	X	X	X	Disable WDT_ STOP
X	On	X	X	X	X	X	X	Enable WDT_ STOP
Off	X	X	X	X	X	X	X	Enable WDT
On	X	X	X	X	X	X	X	Disable WDT

S2

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Remarks
-	BOND ING1	BOND ING0	PKG0	PKG1	RST	LVR LEVEL	LVR	
-	X	X	X	X	X	X	Off	Disable LVR
-	X	X	X	X	X	X	On	Enable LVR
-	X	X	X	X	X	Off	X	High LVR voltage
-	X	X	X	X	X	On	X	Low LVR voltage
-	X	X	X	X	Off	X	X	Enable Reset pin
-	X	X	X	X	On	X	X	Disable Reset pin
-	X	X	Off	Off	X	X	X	Bit 4 和 Bit 5 保持為 Off
-	X	Off	X	X	X	X	X	使用保护方式 1（PB.0 和 PB.1）
-	X	On	X	X	X	X	X	使用保护方式 2（PWM1/2，PC.3）
-	Off	X	X	X	X	X	X	OVPEN, OCPEN, OVPCH 三个寄存器值由 Design Option 中 OP_OVCP 决定
-	On	X	X	X	X	X	X	OVPEN = 1, OCPEN = 1, OVPCH =1, PWMOD=1

SW1 (RESET):
Reset the EVB when push the button.

Diagnostic LED:

Power LED:	The LED will be turned on when the EVB is powered.
LVR LED:	The LED will be turned on when the VDD is lower than LVR voltage.
PWM LED:	The LED will be turned on when the PWM function is enabled.
ADC LED:	The LED will be turned on when the ADC function is enabled.
STOP LED:	The LED will be turned on when the system is in STOP mode.
HALT LED:	The LED will be turned on when the system is in HALT mode.

Notes:

Application notes:

1.1 After entering into the IDE66 and successfully downloaded the user program, use the F5 key on the PC keyboard to reset the EVB before running the program. If abnormal response occurs, the user must switch off the ICE power and quit IDE66, then wait for a few seconds before restarting.

1.2 When running the IDE66 for the first time, the user needs to select the correct MCU type, clock frequency ... then save the settings and restart IDE66 again.

1.3 Can't Step (F8) or Over (F9) a HALT and STOP instruction.

1.4 Can't emulate the interrupt function in Step (F8) operating mode.

1.5 When you want to escape from HALT or STOP (in ICE mode), please press F5 key on the PC keyboard twice.

1.6 The maximum current limit supplied from EVB to the target is 100mA. When the current in the target is over 100mA, please use external power supply.

1.7 Can't emulate the timer function in Step (F8) operating mode.

Programming notes:

2.1 Clear the data RAM and initialize all system registers during the initial programming.

2.2 The "NOP" instruction should be added at the beginning of the program to ensure the IC is stable.

2.3 Never use the reserved registers.

2.4 Do not execute arithmetic operation with those registers that only have 1, 2 or 3 bits. This kind of operation may not produce the result you expected.

2.5 To add "p=69P481" and "romsize=4096" at the beginning of a program. If any problem occurs during the compilation of the program, check the device and set if it was set correctly.

2.6 Both index register DPH and DPM have three bits; so pay attention to the destination address when using them.

2.7 Notes for interrupt:

2.7.1 Please make sure that the IE flag is enabled before entering into a “HALT” or a “STOP mode. It means that the “HALT” or “STOP” instruction must follow the set “IE” instruction closely.

2.7.2 After the CPU had responded to an interrupt, IRQ should be cleared before resetting IE in order to avoid multi-responses.

2.7.3 Interrupt Enable instruction will be automatically cleared after entering into the interrupt-processing subroutine. If setting IE is too early, it is possible to reenter into the interrupt. So the Interrupt Enable instruction should be placed at the last 3 instructions of the subroutine.

2.7.4 CPU will not respond to any interrupt during the next two instructions after the Interrupt Enable flag be set from 0 to 1.

2.7.5 After CPU has responded to an interrupt, IE will be cleared by the hardware. It is recommended to clear the IRQ at the end of interrupt subroutine.

2.7.6 The stack has eight levels. If an interrupt is enabled, there will be only seven levels that can be used.

2.7.7 It is recommended that the last line of program is “END”.

Examples:

1> Description: CPU can not wakeup after executing the “HALT” or “STOP” instruction.

Program: Interrupt Enable instruction is set outside the interrupt subroutine

<Wrong example>	**<Correct example >**
……	……
LDI IE, 0FH ; enable interrupt	LDI IE, 0FH ; enable interrupt
NOP	~~NOP~~
NOP	~~NOP~~
HALT	HALT

Analysis: After two “NOP” instructions, if an interrupt request comes or IRQ is non-zero during the third instruction cycle, CPU will respond to the interrupt and IE will be cleared. Then when returning to main program, CPU starts to execute “HALT” or “STOP” and will not be activated, because IE is cleared to zero and all interrupts are disabled.

Solution: “HALT” or “STOP” are being followed closely by the “LDI IE, 0FH”

2> Description: CPU responds to one interrupt several times.

Program: Interrupt Enable instruction is placed outside the interrupt subroutine.

<Wrong example>

```
L1:
    .....
    LDI     IE, 0FH         ; enable interrupts
    NOP
    NOP
    JUMP    L1
```

Analysis: After executing this two “NOP” instructions, and IRQ is not cleared in time, CPU will respond to the interrupt again when it executes the two instructions followed by “LDI IE, 0FH”. This will happen again and again. So CPU responds to one interrupt several times.

Solution: The relative IRQ flag is cleared in time after responding to the interrupt.

3> Description: CPU is running dead in the interrupt-processing program.

Program: an interrupt subroutine.
<Wrong example>

```
ENTERINT:
    …..
    LDI     IE, 0FH
    NOP
    LDA     STACK, 0
    RTNI
```

Analysis: After executing "LDI IE, 0FH" and the following two instructions, an interrupt request comes or the last relative IRQ flag is not cleared in time, then CPU will respond to the interrupt again, so the interrupt is nesting again. When the stack exceeds over 8 levels, it will run into a dead loop.
Solution: Make sure that the CPU can quit from interrupt subroutine within two instruction cycles after interrupt is enabled; After the interrupt is responded, the relative IRQ flag should be cleared before enabling the interrupt.

2.8 Notes for TIMER

2.8.1 When setting the Timer Counter, write first T0L, then T0H

2.8.2 After setting TM0, T0L, T0H, there is no need to rewrite after the Timer counts overflow, otherwise it will cause a time error every time. The timer is interrupted by the reload registrar that was set in different time.

2.9 Notes for I/O

2.9.1 Each I/O port (excluding those open drain output ports) contains pull-high MOS controllable by the program. Each pull-high MOS is controlled by the value of the corresponding bit in the port pull-high control register (PPCR), independently. When the port is selected as an input port (Write 1 to the relevant bit in the port pull-high control register (PPCR) could turn on the pull-high MOS and write 0 could turn off the pull-high MOS). So the pull-high MOS can be turned on and off individually. But when the port is selected as output port, the pull-high MOS must be turned off automatically, regardless the value of the corresponding bit in the port pull-high control register (PPCR).

2.9.2 When a digital I/O is selected to be an output port, the reading of the associated port bit actually represents the value of the output data latch, not the status on the pad. Only when a digital I/O is selected to be an input port, the reading of the associated port bit represents the status on the corresponding pad.

2.9.3 Setting those I/O ports with open drain output type as input will cause leakage current ranging from tens to hundreds micro-ampere. So do not forget to enable the pull-high MOS or connect these input ports with external resistors (pull-high or pull-low) to prevent the I/O "Floating".

2.9.4 The Key De-bounce time is recommended to be 50ms. But in the Rubber Key application, it is best to test Rubber Key's De-bounce time.

2.10 Refer to the ADC programming notes in the SH69P481 data sheet.

2.11 Refer to the PWM programming notes in the SH69P481 data sheet.

Application notes Revision History

Revision No.	History	Date
0.0	Original	Nov.2013