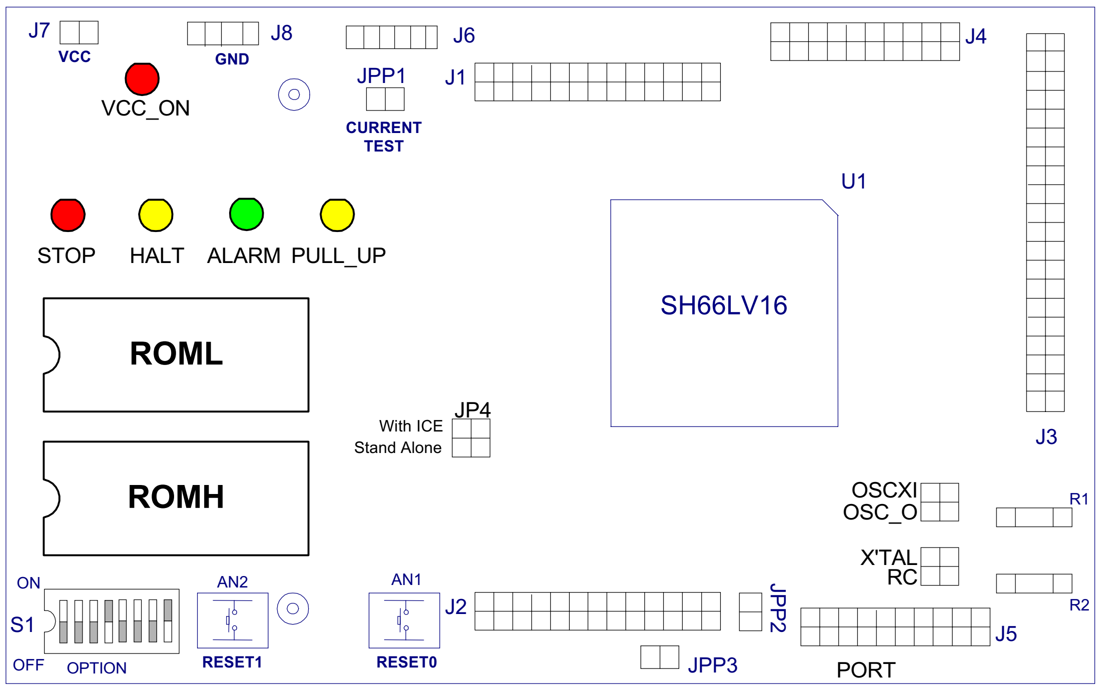
SINO WEALTH

SH66L16 EVB

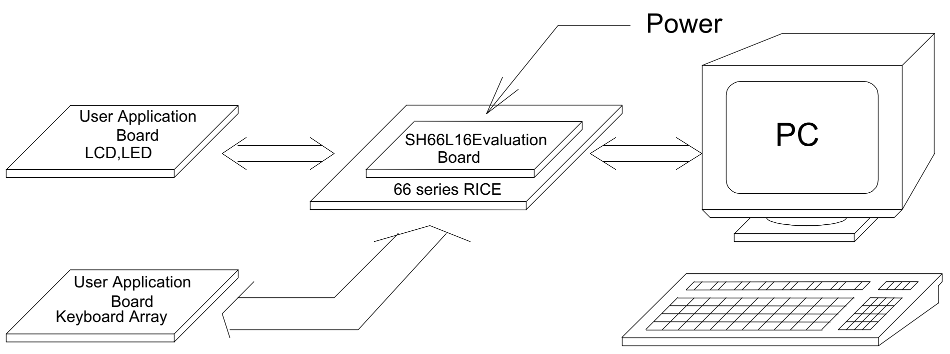
Application Notice for SH66L16 EVB

SH66L16 EVB

The SH66L16 EVB is used to evaluate the SH66L16 chip's function for the development application program. It contains of a SH66LV16 EV chip for evaluating the functions of SH66L16 including the WDT, ALARM and the LCD waveform. The following diagram shows the placement of SH66L16 EVB.

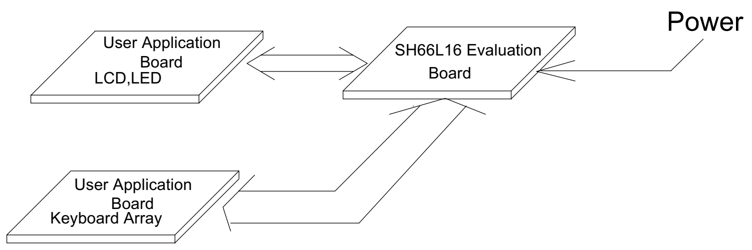


There are two configurations of SH66L16 EVB in application development: ICE mode and stand-alone mode.
In the ICE mode, the SH66xx ICE (motherboard) is connected to the SH66L16 EVB by the ICE interface.


Power
User Application Board LCD,LED
SH66L16Evaluation Board
66 series RICE
PC
User Application Board Keyboard Array

(a) ICE mode

In the stand-alone mode, the SH66L16 EVB is no longer connected to the motherboard. But the EPROM chip (27512 or 27256) which store the application program must be inserted in the sockets of EPROM.


Power
User Application Board LCD,LED
SH66L16 Evaluation Board
User Application Board Keyboard Array

(b) Stand-alone mode

The process of your program's evaluation on SH66L16 EVB

Uasm66.exe: assemble the program, and get binary (*.obj) file and the other files.
Depart the one 16 bits obj file to the two 8 bit files by convert.exe.

Usage example (for example: aaa.asm):

1.Run the NT66 series assemble program:
   C: >uasm66 /p 66L16 aaa.asm ; to produce the obj file: aaa.obj
2. Depart the aaa.obj to two 8 bits files aaah.obj and aaal.obj, for example:
   C: > convert
      Input the 16 bits (.obj) file aaa.obj
      Then aaah.obj and aaal.obj will be created.
3. Write the aaah.obj to EPROM (ROMH)
   Write the aaal.obj to EPROM (ROML)
4. Put the two EPROMs (ROML and ROMH) into the EPROM sockets.

ROM bank switch application Notice

The ROM of SH66L16 is divided to 2K banks. There are 8 banks in this 16K ROM.

CPU Address	**ROM Space**						
	BNK=0	BNK=1	BNK=2	BNK=3	BNK=4	BNK=5	BNK=6
000 ~ 7FF	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)	0000 ~ 07FF (BANK 0)
800 ~ FFF	0800 ~ 0FFF (BANK 1)	1000 ~ 17FF (BANK 2)	1800 ~ 1FFF (BANK 3)	2000 ~ 27FF (BANK 4)	2800 ~ 2FFF (BANK 5)	3000 ~ 37FF (BANK 6)	3800 ~ 3FFF (BANK7)

Bank switch program example:

Example 1:

```
BANK0:ORG 000H; bank 0 program
...
BANK EQU 1FH
BANK2 EQU 01H
...
LDI BANK, BANK2
JMP 1000H; from bank0 jump to bank2
---------------------------------------------------------------------------
BANK2:ORG 1000H; bank2
...
```

Example 2:

```
Program1:
BANK1:...
BANK EQU 1FH
BANK4 EQU 03H
...
LDI BANK, BANK4
JMP B4SR1
-----------------------------------------------------------------------------
Program 2:
BANK4: ORG 2000H
B4SR1:
....
```

SH66L16 EVB Programming Notices

1. Clear data RAM and initialize all system registers at the beginning.
2. Do not perform logical operation with I/O ports. Especially when the I/O ports have external connections.
3. Do not perform logical or arithmetic operation which contains a "read-modify-write" processing with those "write only" system registers.
4. Do not perform arithmetic operation with those registers only have 1, 2 or 3 bits. This kind of operation may not get the result you expected.
5. Never use reserved registers.
6. If "IE" instruction (interrupt enable) is set outside the interrupt processing program and there is "HALT" or "STOP" instruction, this two instructions should be followed "IE" instruction closely.
7. After CPU responding to an interrupt, IRQ should be cleared before resetting IE in order to avoid many responses to one interrupt.
8. Interrupt Enable instruction will be automatically cleared after entering interrupt-processing program. If setting IE too early, there is a possibility of re-entry the interrupt. So the Interrupt Enable instruction should be placed at the end and followed closely by two instructions include "RTNI".
9. During the two successive instruction cycles next to Interrupt Enable instruction, CPU will not respond to any interrupts.
10. After CPU responding to interrupts, each bit of IE will be cleared by hardware while IRQ should be cleared by software.
11. It is necessary to add NOP before and after the HALT instruction, else the CPU will execute error instruction when it wakes up from the HALT.

```
.
. .
NOP
HALT
NOP
.
. .
```

12. It is wise to set the Interrupt Enable flag before you return from subroutine in two instructions.

```
.
. .
LDI     IE, 04H;      Enable timer0 interrupt
LDA     Temp,0
RTNI
```

13. When you set the Interrupt enable flag as the following and your subroutine not be set Interrupt Enable flag, then your system will never wake up if an interrupt entered between the NOPs.

Wrong Example:

```
. .
LDI     IE, 1111B;  IE = Interrupt enable flag
NOP
NOP
NOP
HALT
.
. .
```

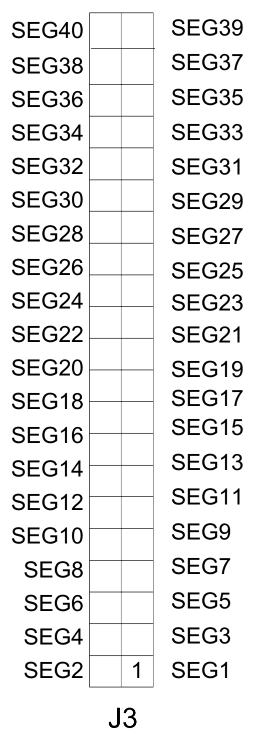
14. To add "p=66L16" or "romsize=16384" at the beginning of a program. If found problem in compiling the program. Check if the NT6566.dev is located at the program directory.
15. Interrupt must be disabled when bank is switching. Otherwise, the program may be failed.

```
.
. .
LDI     IE, 0000B;  IE = Interrupt enable/disable flag
NOP
NOP
LDI     IFH,03H
JMP     LABEL1
.
. .
```

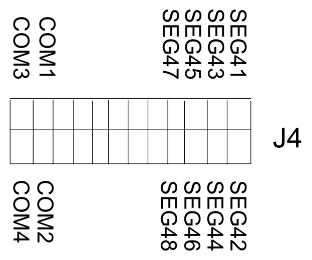
16. Shared interrupt program can be stored in BANK0. When the program runs in other BANK while an interrupt comes, it will automatically jump to the interrupt program in BANK0 and can return to the current BANK automatically after returning from the interrupt.
17. When setting Timer Counter, first fill T0L, then T0H.

18. After setting TM0, T0L, T0H, it is unnecessary to reset them after interrupt each time. If TM0, T0L, T0H are reset after each TIMER interrupt, interrupt interval time will not equal because the interrupt timing is not successive.
19. Any instruction containing writing to or reading from memory, it should not be used to operate with I/O Port. It is best to avoid using those instructions such as "SUB, ADD " which do not contain Write operation with I/O Port but have computation operation.
20. "1" must be written to I/O Port before Reading.
21. Directly reading PORT states ensure the count is correct.
22. Interrupt activating from STOP at the first time can save power.
23. When the Compiler of old version compiles program, the last line will be read twice. So, if the last line is an instruction, two same operations will be occurred. If there is Label in the last line, compiler will give an error named 'repeated definition' . This will happen in main program or included files and it is recommended that the last line should be a blank line or END.
24. The stack has four layers, if an interrupt is enabled, there only have three layers can be used. Otherwise, if an interrupt comes, the stack will be overflowed that will cause CPU Reset or other errors.
25. Key de-bounce time is recommended to be 50ms. If the Rubber Key will be used, it is best to test Rubber Key's de-bounce time.
26. Index register DPH and DPM both have three bits only, so pay attention to the referred address when using them.
27. Please enable 32.768K oscillator before turning LCD on, since LCD clock comes from Basetimer clock source in the condition of the code option (00)(200KHz RC as system clock while 32.768KHz Crystal as Basetimer clock). So do as in the condition of the code option (01) (200KHz RC as system clock while 32.768KHz RC as Basetimer clock).
28. Normally, the LCD driver clock comes from the Basetimer clock source in the condition of code option (00 or 01). Thus the LCD can display ON even in the STOP mode.
29. Writing system register $14 for selecting LCD display mode must be under the condition of "LCD OFF". Otherwise it will be regarded as illegal operation.
30. Since the LCD pump circuits need more current while starting up, please confirm LCD OFF before turning on the Basetimer clock source oscillator in the condition of the code option (00). So do as in the condition of the code option (01).
31. Normally, the Alarm carrier clock comes from the Basetimer clock source in the condition of code option (00 or 01). Thus the alarm signal can output even in the STOP mode.
32. In the condition of the code option (10) (32.768KHz Crystal as system clock) or (11) (200KHz RC as system clock), the Basetimer clock, the LCD driver clock and the Alarm carrier clock are fetched from the system clock. Please pay more attention to these CODE OPTIONs.
33. Since PORTD, E, F are shared with segment49~56 and common5~8,individually, each bit of PORTD can be controlled as input/output simultaneously. So do as the PORTE and PORTF.
34. The Watch-Dog-Timer's clock is fetched from the system clock, so WDT can not run in the STOP mode. To prevent it timing out and generating a device RESET condition, the bit3 of system register $1E must be set as "1" before timing-out. RESET condition caused by WDT, it can be detected by reading bit3 of system register $1E. The value "0" means WDT causing the system reset. The value "1" means another events may cause system reset, such as Power- On, Reset pin pressed during the normal operation or in the HALT mode. Additionally, the device waked up from the STOP mode can set the bit3 of system register $1E as "1". Since the TM0 register is shared with WDT's prescaler division, If theTM0 is changed for some proper use, the WDT's time-out periods will also be changed. Meanwhile, writing TM0 register will cause the WDT's counter reset.
35. If the WDT function is disabled, the value of the bit3 of system register $1E will always be "1" while program reading.
36. The "NOP" instruction should be added at the beginning of the program to ensure the IC for stability.

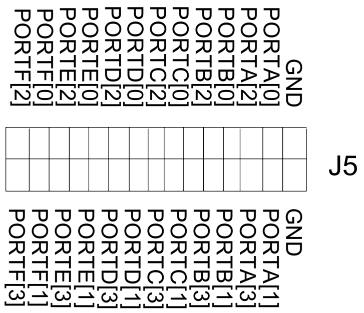
LCD interface connector : J3 (Top View from EVB)



LCD interface connector : J4 (Top View from EVB)



Port interface connector : J5 (Top View from EVB)



External Vcc input: J7, (J8=GND)

(The external power input when the EV. Board worked in stand alone mode. The voltage of Vcc_J7 must be 5V±5%)

CPLD interface: J6

This connector is reserved for CPLD writing. Therefore , it can not be used by application programming.

Most important:

Incorrect power input (GND connected to Vcc pin J7, and Vcc connected to GND pin J8) will hurt or break down the EV board permanently.

Jumper setting:

JP4 (SH66L16 EVB ICE/Stand-alone mode select):

- When the jump of ”With_ICE” position is shorting, the clock of SH66L16 EVB is fed from the ICE.
- When the jump of ”ST_ALONE” position is shorting (only for stand-alone mode), the clock of SH66LV16 is provided from its oscillator (200KHz RC or 32768Hz Crystal).

JPP1 (EV chip Current test):

- To supply 1.5V source to EV chip while short this jump. (Through pin 1 to pin 2.)
- It can be used to test the whole chip current of EV by means of connecting an ammeter.
- It also can be used to input external voltage (1.2~1.7V) through the pin on the right side to feed the EV chip.
- If the whole chip current under the reset condition caused by RESET0/RESET1 is needed to be tested, the real value can be fetched from an ammeter connected between two pins on the JPP1 via the button AN2(RESET1) being pressed directly or the pin on the top side of JPP2(RESET0) being connected to GND simply.

JPP2 (Reset0 pull-high resistor consuming current test):

- It can be used to test the current of the internal pull-high resistor consuming via the RESET0 pin on the EV chip by means of connecting an ammeter between the pin on the top side and GND.
- If the consuming current of the internal pull-high resistor connected to the RESET1 pin on the EV chip is needed to be measured, the user can get the real value by means of connecting an ammeter between a reset1 pin on the button AN2 and GND.

JPP3 (STACK):

- When this jump is open, the ICE will never response while the program stack overflowing.

Switch setting:

S1 (OPTION):

1.Bit 1: (**C**ode **O**ption **C**) ON=1, OFF=0

2.Bit 2: (**C**ode **O**ption **K**) ON=1, OFF=0

Code		**Connection**	
C	**K**	**OSCI/OSCO**	**OSCXI**
0	0	32KHz Crystal	200KHz RC
0	1	32KHz RC	200KHz RC
1	0	32KHz Crystal	
1	1		200KHz RC

when **COC**=0, **COK**=0 Jump (OSCXI =short, OSC_O =short, X'TAL =short, RC =open)

COC=0, **COK**=1 Jump (OSCXI =short, OSC_O =open, X'TAL =open, RC =short)

COC=1, **COK**=0 Jump (OSCXI =open, OSC_O =short, X'TAL =short, RC =open)

COC=1, **COK**=1 Jump (OSCXI =short, OSC_O =open, X'TAL =open, RC =open)

3.Bit 3: (**C**ode **O**ption **W**) ON=1, OFF=0

COW=0	Watchdog timer enable
COW=1	Watchdog timer disable

4.Bit 4: (**C**ode **O**ption **R**) ON=1, OFF=0

COR=0	RESET level triggering (low active)
COR=1	RESET edge triggering (falling edge)

5.Bit 5: Warmup-Skp switch
If the switch is **ON**, the warm-up counter's length is same as the SH66L16 (Main chip). (2^{12}~2^{14})
If the switch is **OFF**, the warm-up counter's length will be reduced. (2^{5})

6.Bit 6: LCD common mode selection
ON: Normal operation

OFF: Prohibited

7. Bit7: (Bonding option **B0**) ON=1, OFF=0

8. Bit8: (Bonding option **B1**) ON=0, OFF=1

AN1 (RESET0):

AN2 (RESET1):
Reset the EVB when the button is pressing.

LED indication:

STOP:

D4 is lighted when the CPU is in STOP mode.

HALT:

D5 is lighted when the CPU is in HALT mode.

Alarm:

D6 is lighted when alarm is working.

PULL_UP:

D7 is lighted when pull_up is enable.

Notice:

Evaluate your program with ICE indicate:

1. After enter to RICE66 and successfully download the user program, push the F5 (Reset) on PC keyboard before run your program when you evaluate your program with ICE. If there were abnormal response, the user should push the F5 key again until the ICE work normally.
2. First time run RICE66, need to select an appropriate MCU type, clock frequency ... save the settings and restart RICE66 again.
3. Can't execute Step (F8) or Step over call (F9) while the ICE worked in HALT or STOP mode.
4. When you want to escape from HALT or STOP (in ICE mode), you should press the F5 key on PC keyboard twice.