



十速

TM52 MCU 使用 STACK 注意事项

Application Note

Rev 1.0

tenx reserves the right to change or discontinue the manual and online documentation to this product herein to improve reliability, function or design without further notice. **tenx** does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. **tenx** products are not designed, intended, or authorized for use in life support appliances, devices, or systems. If Buyer purchases or uses tenx products for any such unintended or unauthorized application, Buyer shall indemnify and hold tenx and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that tenx was negligent regarding the design or manufacture of the part.

AMENDMENT HISTORY

Version	Date	Description
V1.0	Jun, 2015	New Release.

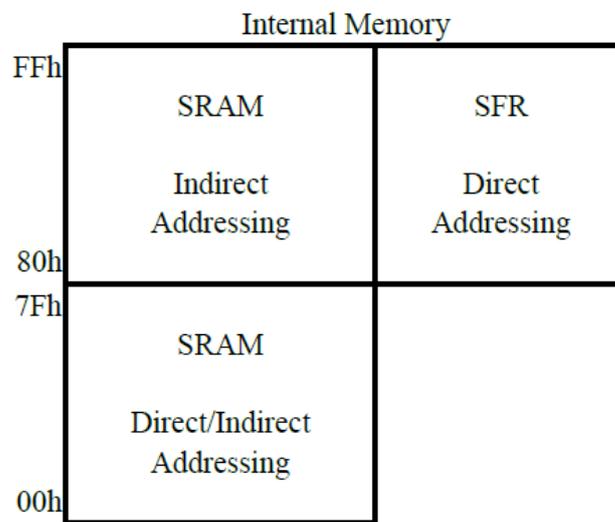
CONTENTS

AMENDMENT HISTORY	2
如何观察和防止 STACK OVERFLOW	4
要如何减少程序 STACK OVERFLOW 的两个方法	13

如何观察和防止 STACK OVERFLOW

在一些没有 XRAM 的 8051 中，要如何防止与观察 STACK OVERFLOW 是很重要的一件事，因为 STACK OVERFLOW 在 KEILC 中不会自动侦测与提示，compiler 的结果也会是正确的，但是在 RUNTIME 时会就会发生不可预期的错误状况。

下图是 5254_58 Internal RAM(IRAM)的架构图，一共有 0x100 个 BYTE 可以使用，而 8051 的 STACK 是存在放 IRAM 中，并且是由 KEILC 依照程序中所使用到的变量来动态分配 SP (stack pointer) 所在位置。



以下提供三个方式来观察程序是否会有 STACK OVERFLOW 的情况。

1. 观察*.M51 档中的 ?STACK 标签，M51 文件会在你的项目底下，档名与你的项目名称相同，寻找 M51 档中的 DATA MEMORY 区段中会有个?STACK 标签，看他的 BASE 值是多少。

TYPE	BASE	LENGTH	RELOCATION	SEGMENT NAME

* * * * * D A T A M E M O R Y * * * * *				
REG	0000H	0008H	ABSOLUTE	"REG BANK 0"
DATA	0008H	0005H	UNIT	_DATA_GROUP_
DATA	000DH	0002H	UNIT	?DT?APP
DATA	000FH	0001H	UNIT	?DT?TOUCHFUN
IDATA	0010H	0005H	UNIT	?ID?SPI_ROUTINES
.....	0015H	000BH		*** GAP ***
BIT	0020H.0	0001H.2	UNIT	?BI?TOUCHFUN
BIT	0021H.2	0000H.2	UNIT	_BIT_GROUP_
BIT	0021H.4	0000H.1	UNIT	?BI?TOUCHSCAN
.....	0021H.5	0000H.3		*** GAP ***
IDATA	0022H	002AH	UNIT	?ID?TOUCHFUN
IDATA	004CH	0028H	UNIT	?ID?TOUCHSETUP
IDATA	0074H	0001H	UNIT	?STACK

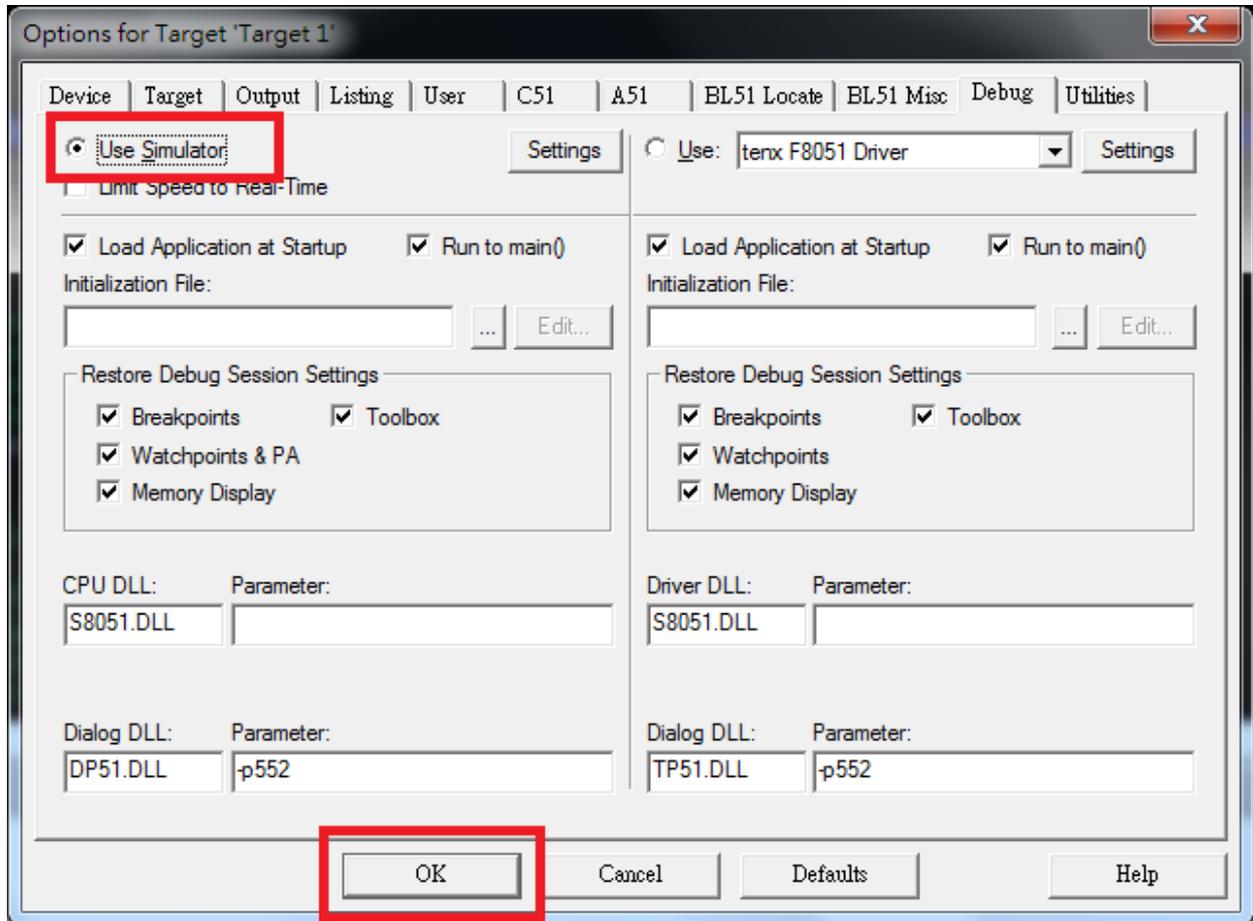
以上图为例，此程序编译完的 SP 值是 0x74，SP 指的是 IRAM 的位置，IRAM 的最大值是 0xFF，

因此，此范例的 SP 深度可以有 0xFF-0x74 = 0x8B, 相当安全。

修改一下此程序，把程序中使用的变量加大，同一个 M51 档中可以看到如下图，STACK 的值是 0xFB 因此 SP 的深度只有 0xFF-0x0xFB = 4 只剩下 4 个 BYTE, 非常危险，只要在程序中 CALL 了三层的 fuction 就会 overflow 了。

* * * * * D A T A M E M O R Y * * * * *				
REG	0000H	0008H	ABSOLUTE	"REG BANK 0"
DATA	0008H	0005H	UNIT	_DATA_GROUP_
DATA	000DH	0002H	UNIT	?DT?APP
DATA	000FH	0001H	UNIT	?DT?TOUCHFUN
IDATA	0010H	0005H	UNIT	?ID?SPI_ROUTINES
.....	0015H	000BH		*** GAP ***
BIT	0020H.0	0001H.2	UNIT	?BI?TOUCHFUN
BIT	0021H.2	0000H.2	UNIT	_BIT_GROUP_
BIT	0021H.4	0000H.1	UNIT	?BI?TOUCHSCAN
.....	0021H.5	0000H.3		*** GAP ***
IDATA	0022H	00AFH	UNIT	?ID?TOUCHSETUP
IDATA	00D1H	002AH	UNIT	?ID?TOUCHFUN
IDATA	00FBH	0001H	UNIT	?STACK

2. 使用 μ Vision Simulator 观察 sp_max 值，设定方式如下



点击 Project/Options for Target 选取 Debug 页面，如上图画面，请选择 Use Simulator 后点选 OK。

进入 DEBUG 模式，出现的画面如下

The screenshot shows a debugger window with the following components:

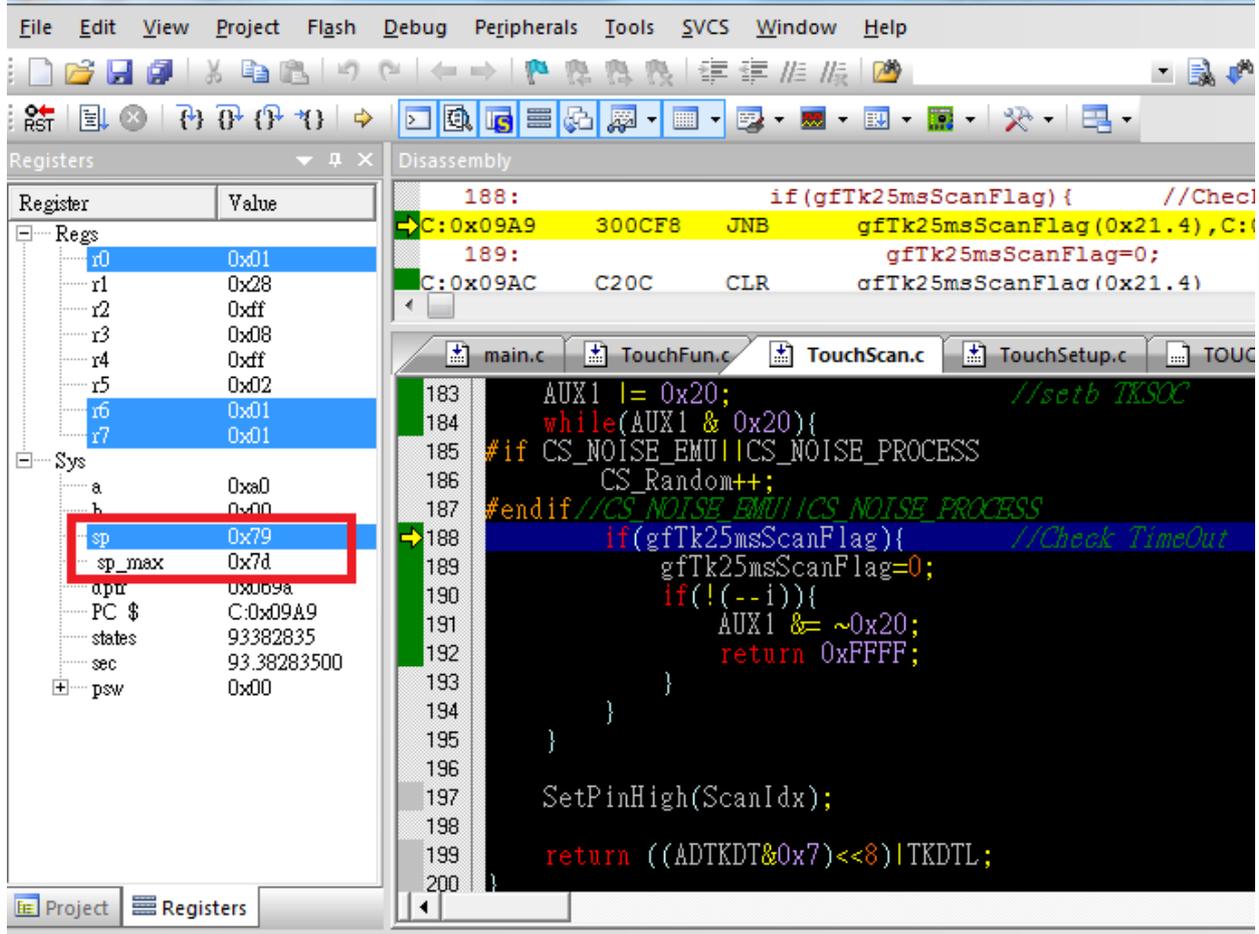
- Registers Panel:** Lists registers and their values. The stack pointer registers are highlighted with a red box:

Register	Value
r0	0x20
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
sp	0x73
sp_max	0x73
cpu	0x0004
PC \$	C:0x0923
states	484
sec	0.00048400
psw	0x40
- Disassembly Panel:** Shows assembly code with the following lines highlighted:

```
15:      InitSpiTkParm();
C:0x092C 1208D6 LCALL  InitSpiTkParm(C:08D6)
16:      TouchConfig(SPI_COMMUNICATION); //StartTouchKey/Enc
17:      #if TK_MODEL_TYPE==1
```
- Source Code Panel:** Shows C code with the following lines:

```
18      TouchThreshold(6/*8*/,4,2,0,3); //Threshold
19  #else
20  // TouchThreshold(8,4,2);
21  // TouchThreshold(17,22,14);
22  // TouchThreshold(9,19,16,19,6);
23  // TouchThreshold(11,22,16,19,6);
24  TouchThreshold(11,22,21,19,6);
25
26  /*
27  TkCnt: Buffer
28  Level0: (?) (F) (?)
29  Level1: (?)F(?)
30  Level2: Trigger threshold
31  Level3: slope(x10)
32  Level4: Water Detect threshold
33  */
34  #endif
35  InitTouchKeyScanTimer(); //Must setup Timer
```

红色框框处，会有目前的 sp 值与曾经最大的 sp 值 sp_max，上图是一开始都还没有 RUN 过程序所以 SP 与 SP_MAX 是一样的，FREE RUN 后再 HALT 停下来观察 SP_MAX 值如下图。



SP_MAX 的值是 0x7D。程序所使用到的 SP 最高用到过 0x7D 还是很安全的。

再看一下会发生 SP OVERFLOW 的状况，如下图

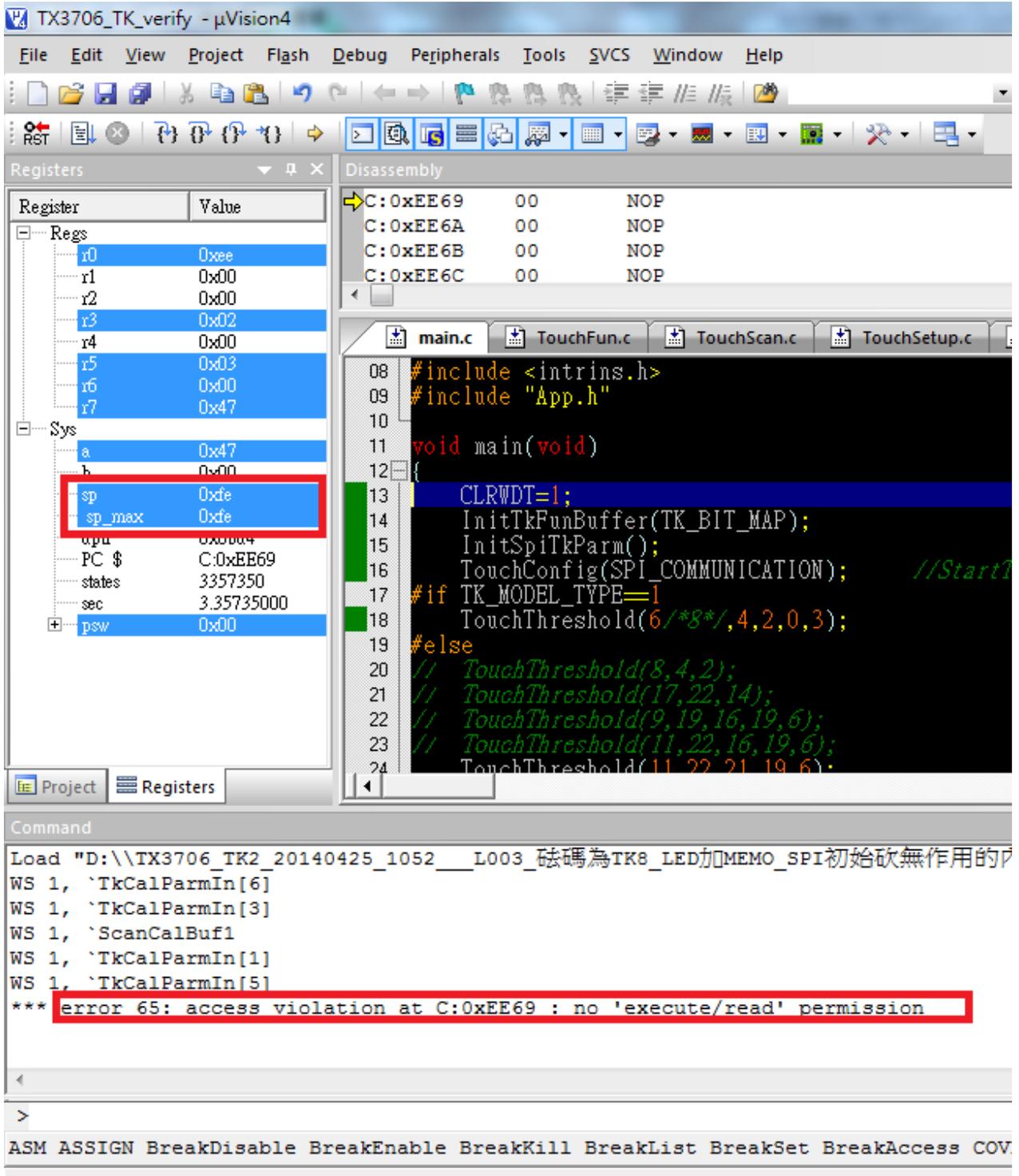
The screenshot shows a debugger interface with three main panes. On the left, the 'Registers' pane displays a list of registers and system variables. The 'sp' register is highlighted with a red box, showing a value of 0xfda. The 'sp_max' register is also highlighted, showing a value of 0xfda. In the center, the 'Disassembly' pane shows the instruction 'CLRWD1=1;' at memory address 'C:0x0923'. On the right, the source code window shows the 'main.c' file with the same instruction highlighted. The code includes headers and sets up touch thresholds.

Register	Value
r0	0x20
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
sp	0xfda
sp_max	0xfda
upu	0x0004
PC \$	C:0x0923
states	484
sec	0.00048400
psw	0x40

```
11: void main(void)
12: {
13:     CLRWD1=1;
14:     InitTkFunBuffer(TK_BIT_MAP);
15:     InitSpiTkParm();
16:     TouchConfig(SPI_COMMUNICATION); //StartTouch
17:     #if TK_MODEL_TYPE==1
18:     TouchThreshold(6/*8*/,4,2,0,3);
19:     #else
20:     // TouchThreshold(8,4,2);
21:     // TouchThreshold(17,22,14);
22:     // TouchThreshold(9,19,16,19,6);
23:     // TouchThreshold(11,22,16,19,6);
24:     TouchThreshold(11,22,21,19,6);
```

此范例程序一开始使用到的 SP 就在 0xfda，离 0xff 不远了，非常危险！

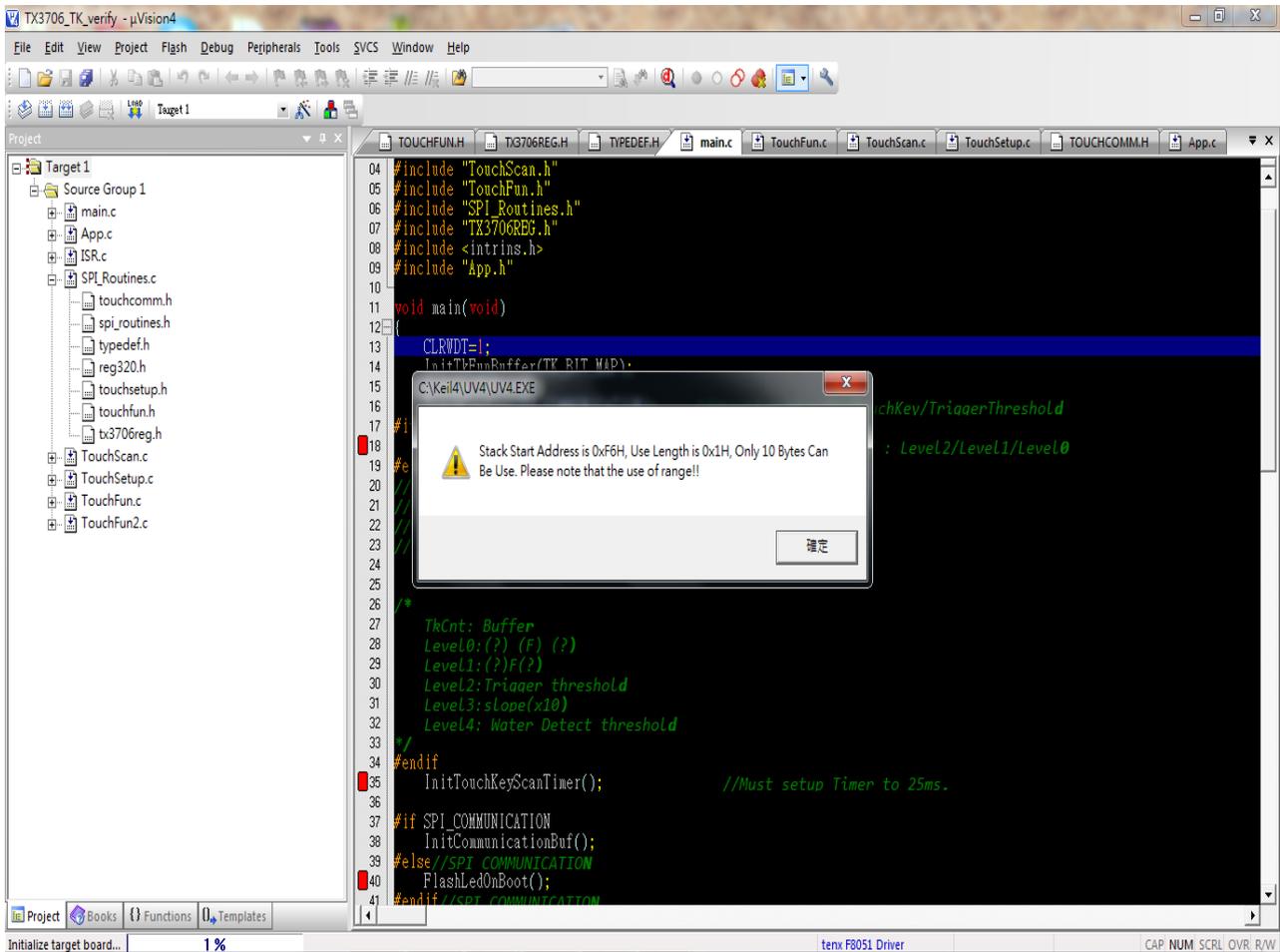
按下 Free Run 钮后，程序马上因为发生 STACK OVER FLOW 自动停止 Free RUN 了，观察 sp_max 与 sp 值都是 0xfe，并且可以看到讯息窗口出现错误讯息。



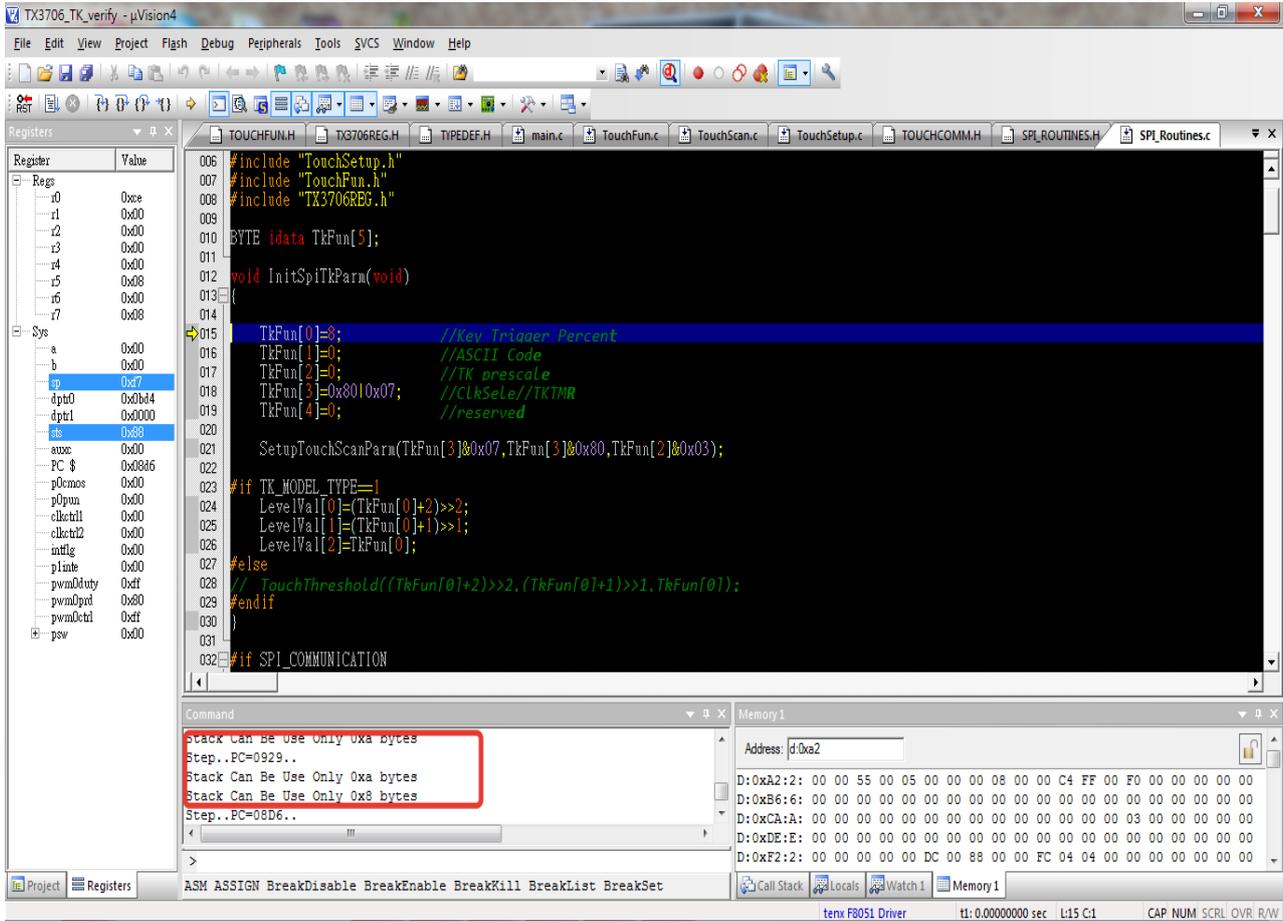
The screenshot displays the uVision4 IDE interface for a project named 'TX3706_TK_verify'. The 'Registers' window on the left shows the stack pointer 'sp' and 'sp_max' both at address 0x0de, highlighted with a red box. The 'Disassembly' window shows a sequence of NOP instructions at addresses C:0xEE69 through C:0xEE6C. The 'Command' window at the bottom shows a red-bordered error message: 'error 65: access violation at C:0xEE69 : no 'execute/read' permission'. The source code in the background shows a C program with various initialization and configuration functions.

此方式可以用 free RUN 的方式检测程序有没有 STACK OVERFLOW。

3. TENX KEILC ICE 插件关于 STACK OVERFLOW 提示



十速的 KEILC 仿真插件会在进入仿真模拟时，自动检查 SP 的所在位置，在 SP 的所在位置大于 0xF0 之后，会跳出警告讯息如上图。



并在进入仿真模式后，在 command 字段会显示出目前最新剩下的 SP 所能使用的 BYTE 数。

要如何减少程序 STACK OVERFLOW 的两个方法

1. 尽量减少声明全局变量，全局变量越多，固定占用的 IRAM 也就越多，非必要请尽量减少。
2. 减少函数调用的层数，函数调用越多层，STACK 占用的越多，尽量减少函数调用的层数。