AN421
Application note

Stack overflow detection using the ST9 timer/watchdog

1 Introduction

In real time applications, implementation of software protection is not always easy, but it is needed to reach a high level of security against software malfunction. This is particularly true for on-board applications in noisy environments, such as automotive, power meter or industrial applications.

To help avoid un-controlled operations and damage to real time systems due to any possible disturbance of the ST9 microcontroller core and I/O ports, a special peripheral acting as a watchdog is available on all ST9 family members: the Timer/Watchdog.

The periodic restarting of the Timer/Watchdog by the application software, associated with the automatic detection of any stack overflow, provide enhanced protection to real time application software.

This application note shows how to detect stack overflow by using the Timer/Watchdog in watchdog mode.

Contents

List of figures

2 Stack overflow detection principle

2.1 Summary of timer/watchdog features

The ST9 core includes a 16-bit down counter with an 8-bit prescaler capable of operating in watchdog mode. This timer, driven by a clock at a frequency of INTCLK divided by 4, is able to provide time periods within the range of 333 ns to 5.59 s (using a 12 MHz internal clock).

In watchdog mode, the Timer/Watchdog generates a fixed time base depending on the Timer/Watchdog registers and prescaler, and to INTCLK. This time base can be modified on the fly by changing the prescaler value. The new value will be taken into account only after an End Of Count event. In watchdog mode, the End Of Count occurrence generates a system reset.

In order to prevent the reset, the byte sequence AAh, 55h should be written into the Timer Watchdog register Low. Once the write of 55h has been performed, the timer reloads the prescaler register and the counting restarts from this value (the prescaler register value may be modified between two End Of Count events).

Note: 1 *For a better understanding of this application note; please refer to the ST9 Technical Manual chapter on the 16-bit programmable Timer/Watchdog.*

2 *INTCLK: Internal Clock. This clock issued from the oscillator circuitry, divided or not by 2, is the ST9 Internal Clock driving the peripherals. The maximum frequency allowed for INTCLK is 12MHz.*

2.2 Stack overflow detection

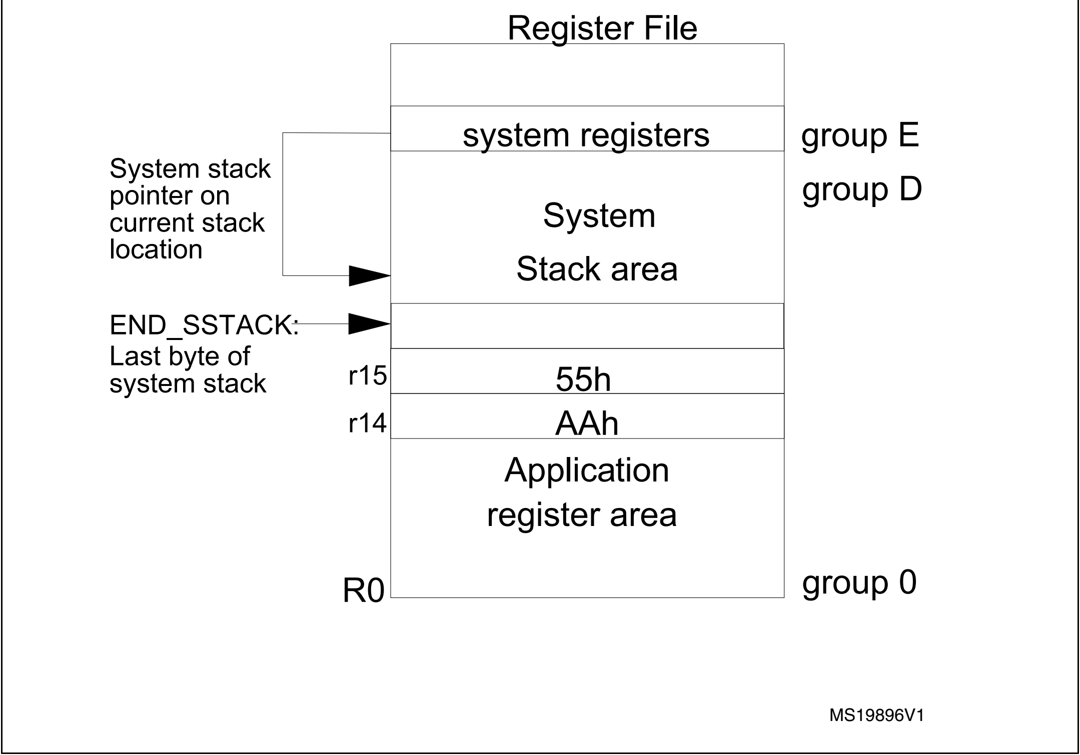
In many software applications, for example when running on ST9 ROMLESS versions or without external memory space, the size of the stack is limited.

On ST9 devices, the system stack may be located in the Register File or in data memory space. The ST9 stack pointer moves from the top to the bottom of the stack area.

A solution to detect stack overflow is to reserve the first two bytes after the bottom of the stack and to store in these locations the Timer Watchdog restart value, AAh, 55h.

In the case of stack overflow, the data will be overwritten and thus destroyed and a system reset will be generated on the next Timer Watchdog End Of Count.

Figure 1. Example of stack overflow detection in register file


Register File
system registers
group E
group D
System stack pointer on current stack location
System Stack area
END_SSTACK: Last byte of system stack
r15
55h
r14
AAh
Application register area
R0
group 0
MS19896V1

3 Software description

3.1 Stack initialization

The following example initializes the system stack in groups D and C of the Register File.

In the stack management of the ST9, the stack pointer is automatically pre-decremented before the data is stored on the stack. So the expression:

```
SSTACK = (BKE * 16) - 1
```

defines the first location of the system stack in group D and C within the Register File, while the instruction:

```
ld SSPLR,#SSTACK + 1
```

initializes the system stack pointer in the system register. The instruction:

```
ldw RR#END_SSTACK,#0AA55h
```

initializes the first two bytes following the bottom of the system stack with the value used to restart the Timer/Watchdog. Initialization

Figure 2. System stack initialization

```
;****************************************************************
; STACK Declaration and end of stack initialisation
; in RAM space or Register File
;****************************************************************
; Initialisation in Register FIle
SSTACK := (BKE * 16) - 1                   ; Sys.stack add.group
LG-SSTACK := 32 ; Sys.stack length
END_SSTACK := (BKE * 16) - LG_SSTACK       ; Last sys.stack byte
ld SSPLR,#SSTACK + 1                       ; Load sys.stack pointer
ldw RR#END_STACK - 2,#0AA55h               ; Init end of stack.
; Initialisation in RAM space
SSTACK := 2000h                            ; top of sys.stack
END-STACK := 1000h                         ; Init end of stack
essp = rr0
sdm
ldw SSPR,#SSTACK                           ; Select data space
ld essp,#END_SSTACK                        ; Init End of sys.stack
ldw -2(essp),#0AA55h
```

3.2 Timer/watchdog programming

As an example, the Timer Watchdog is initialized in order to provide a time base of 10 ms (with a ST9 driven by a clock frequency of 24 MHz internally divided by two). To enable the Watchdog mode, the requirement is to initialize Timer prescaler and counter, to initialize the Timer/Watchdog Control Register with its reset value, and then to enable the watchdog mode by clearing the WDGEN bit in the Wait Control Register in page 0. Resetting this bit causes the counter to start in Watchdog mode regardless of the start/stop, Single/Continuous and input mode bits.

Figure 3. Timer/watchdog initialization

```
;*****************************************************************
**
; WATCHDOG INITIALISATION
;*****************************************************************
**
proc INIT_WGT[PPR] {
spp #0
ld WDTPR,#0             ; TWD prescaler register
ld WDTLR,#-30h        ; ; TWD Timer counter low
ld WDTHR,#075h        ; ; TWD Timer counter high
}
call INIT_WGT           ; call TWD initialisation
spp #0                ; ; select page 0 register
ld WCR,#00111111b     ; ; Enable the Watchdog
ei                    ; ; Enable Interrupt
```

Note: 1 A bit (DIV2 located in the MODE Register MODER, R235 in the system group) controls the divide by two circuit which operates on the OSCIN clock driving the ST9. The maximum Internal Clock (INTCLK) allowable for the ST9 is 12 MHz. This internal clock drives all the ST9 peripherals, while this same clock, optionally slowed down by the ST9 Core clock programmable prescaler and by wait cycle insertion, drives the ST9 Core. After a reset cycle, the clock frequency applied to the ST9 is divided by two and no Core clock prescaling is done.

3.3 Timer/Watchdog restart

This example shows how to restart the Timer Watchdog when the stack is located in Register File or in RAM space. In the register file, the two instructions:

```
ld WDTLR,#END_SSTACK-2
ld WDTLR,#END_SSTACK-1
```

load the restart value of Timer Watchdog.

When the system stack is located in RAM space, a register essp (end of system stack pointer) must be used to load the sequence AAh, 55h in the Timer Watchdog counter register low.

Figure 4. Restarting the timer/watchdog

```
; In Register File
spp #0                  ; TWD register page
ld WDTLR,R#END_SSTACK-2 ; Load AAh
ld WDTLR,R#END_SSTACK-1 ; Load 55h
; In RAM space
spp #0                  ; TWD register page
sdm                     ; Select RAM space
ld essp,#END_SSTACK     ; End stack pointer
ld WDTLR,-2(essp)       ; Load AAh
ld WDTLR,-1(essp)       ; Load 55h
```

4 Summary

Protection of software against externally generated perturbations can be made by additional test routines.

This protection can easily be increased by using the ST9 Timer/Watchdog bringing software reliability and security. With the Timer/Watchdog the ST9 program can control that the software executing properly. Additionally, when restarting the Timer Watchdog from values (AAh, 55h) located at the bottom of the system stack two additional security functions are added:

- Test of the integrity of the Register File or the RAM space
- Provision of a system reset in the case of stack overflow.

5 Revision history

Table 1. Document revision history

Date	Revision	Changes
15-Dec-1992	1	Initial release.
02-Nov-2011	2	Updated format and company logo.