Hitachi America, Ltd. TN-0202
Application Engineering
TechNote

Cristian Tomescu

Implementing software delay loops on the H8/300

The H8/300 family has a concise set of 57 RISC-like instructions that can operate on bit, byte, or word data using up to eight possible addressing modes (although most instructions normally use less than the maximum number of addressing modes). These instructions act upon the CPU general registers (R0-R7), the CPU control register (CCR), the program counter (PC), the memory (either internal/external ROM/RAM memory or memory-mapped on-chip peripheral registers), or on none of the above. Most of the time, these instructions are being used by the programmer for a "direct" task within his program flow. However, the need of using instructions for no direct purpose but to induce a needed delay in the program sometimes arises.

In this endeavor, the programmer must properly select instructions that would not disturb any previous program conditions. Specifically, the CPU registers, the user external/internal memory values, and the on-chip peripheral register values should be held unchanged. Also, no instructions that alter the CCR flags should be used. Of course, there are instances where the programmer will not use a certain CPU register or not care about a certain flag, and in which case he/she can choose a wider range of instructions to achieve the desired delay; however, in this technote, only the instructions that will not alter any previous conditions or registers will be discussed.

Before implementing the desired delay, 2 factors should be taken into considerations: the internal running frequency of the processor and the number of clock cycles the chosen delay instruction is executed. The table below shows the instructions that can be used to implement software delays and their execution time in clock states (and their execution has no effect upon either registers or condition flags).

Instructions	**Operation**	**Execution time**
NOP	no operation	2 states
BRN	never branch	4 states
BRA	always branch	4 states
Bcc	conditional branch	4 states
BCLR, BSET, BIST, BNOT, BST	register-direct bit operation	2 states
	register-indirect bit operation	8 states
	absolute address bit operation	8 states
ANDC #H'FF,CCR	AND CCR contents with 1's	2 states
ORC #H'00,CCR	OR CCR contents with 0's	2 states
BSR *address* and RTS at *address*	branch at address and return	6 + 8 = 14 states
JSR *address* and RTS at *address*	jump at address and return	8 + 8 = 16 states
JMP *location* and instruction at location	absolute address jump	6 states
	memory indirect jump	8 states

1. The Bcc (conditional branch) instructions can be used if the user makes sure that the branch conditions are not satisfied; then, the branch will be ignored.
2. BCLR and BSET can be used if the user makes sure that they act upon an already cleared or, respectively, set register bit.
3. BIST, BNOT, and BST can be used if the user makes sure that they act upon memory locations that are not utilized by the program.
4. ANDC #H'FF,CCR and ORC #H'00,CCR can be used since it does not alter the contents of the CPU control register.
5. BSR and JSR cannot be used alone since they change the PC contents.
6. JMP and BRA are used in conjunction with other instructions above.

If the programmer does not care about the status flags but still wants to keep the register contents intact, the following additional instructions may be utilized:

Instructions	**Operation**	**Execution time**
ADD.B #0,RnL(H)	add 0 to register	2 states
AND #H'FF,RnL(H)	AND register with 1's	2 states
BTST, BOR, BXOR, BLD, BIXOR, BIOR, BILD, BIAND, BAND	register-direct bit operation	2 states
	register-indirect bit operation	6 states
	absolute address bit operation	6 states
CMP.B or CMP.W	compare register with source	2 states
MOV Rn,Rn	move a register into itself	2 states
MOV.B #0,RnL(H)	move a 0 into register	2 states
MOV.W #0,Rn	move a 0 into register	4 states
OR #0,RnL(H)	OR register with 0's	2 states
PUSH and POP	push and pop register into stack	6 + 6 = 12 states
SUB #0,RnL(H)	subtract 0 from register	2 states

The bit instructions in the table above only alter the value of the carry flag and not the register contents.

Example 1: Let's say a delay of 2.2us is needed in a H8/300-based system running at 10MHz before a sleep condition should occur. This translates into a 22 clock states delay. Also, no register contents as well as condition flags may be altered. Given the possibilities above, multiple approaches are possible. However, a programmer can implement this delay using 4 instructions in the following way:

```
        JSR address 1
Address 1:
        RTS                     ; 8 + 8 = 16 states delay
        BRN                     ; 4 states delay
        NOP                     ; 2 states delay
        SLEEP
```

Example 2: The same delay (2.2us) is needed for a H8/300-based system running at 10MHz before a sleep condition should occur. This means that a delay of 11 states is needed. The programmer does not care whether or not the condition flags are changed. This delay may be implemented using 3 instructions as follows:

```
        JSR address 1
Address 1:
        RTS                     ; 16 states delay
        BTST #n,@RnL(H)         ; 6 states delay
        SLEEP
```

The information in this document has been carefully checked; however, the contents of this document may be changed and modified without notice. Hitachi America, Ltd. shall assume no responsibility for inaccuracies, or any problem involving a patent infringement caused when applying the descriptions in this document. This material is protected by copyright laws. © Copyright 1995, Hitachi America, Ltd. All rights reserved. Printed in U.S.A.