```
1  ;-------------------------------------------------------------------------------
2  ; MSP430 Assembler Code Template for use with TI Code Composer Studio
3  ;
4  ;
5  ;-------------------------------------------------------------------------------
6              .cdecls C,LIST,"msp430.h"       ; Include device header file
7
8  ;-------------------------------------------------------------------------------
9
10             .def    RESET                   ; Export program entry-point to
11                                             ; make it known to linker.
12 ;-------------------------------------------------------------------------------
13             .data                           ; space for 50 words
14 output1:    .space 512                      ; space for 50 words
15 output2:    .space 512                      ; space for 50 words
16 ;-------------------------------------------------------------------------------
17             .text                           ; Assemble into program memory
18             .retain                         ; Override ELF conditional linking
19                                             ; and retain current section
20             .retainrefs                     ; Additionally retain any sections
21                                             ; that have references to current
22                                             ; section
23 input:      .space 512
24 ;-------------------------------------------------------------------------------
25 RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
26 StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
27
28 ;-------------------------------------------------------------------------------
29                                             ; Main loop here
30 ;-----------  For Loop 1  -------------------
31             mov.w   #0, R5                  ; init loop counter
32
33 for1_cond_label:
34             cmp.w   #512, R5
35             jge     for1_break
36             ; for1 stuff
37                 mov.w   input(R5), output1(R5)
38                 sub.w   #425, output1(R5)
39             ; end for1 stuff
40             incd.w  R5
41             jmp     for1_cond_label
42 for1_break:
43
44 ;-----------  For Loop 2  -------------------
45             mov.w   #0, R5                  ; init loop counter
46
47 for2_cond_label:
48             cmp.w   #512, R5
49             jge     for2_break
50             ; for2 stuff
51                 push.w  #150                ; N1
52                 push.w  #-250               ; N2
53                 push.w  output1(R5)         ; N
54                 sub.w   #2, SP              ; space for M
55                 call    #LIMIT
56                 pop.w   output2(R5)         ; output[R5] = M
57                 add.w   #6, SP              ; reclaim stack space
```

```
58              ; end for2 stuff
59
60              incd.w  R5
61              jmp     for2_cond_label
62  for2_break:
63
64  loop:       jmp loop
65
66  ;-------------------------------------------------------------------------------
67  ;               Subroutine: LIMIT
68  ;
69  ;   SP ->  Return Address
70  ;          Output M   (word)
71  ;          Input  N   (word)
72  ;          Input  N2  (word)
73  ;          Input  N1  (word)
74  ;   Subroutine assumes that N1 > N2
75  ;-------------------------------------------------------------------------------
76
77  LIMIT:
78              ; no local variables required
79              ; 0(SP) is (return address)
80              ; 2(SP) is M (where the output goes)
81              ; 4(SP) is N
82              ; 6(SP) is N2
83              ; 8(SP) is N1
84
85              cmp.w   8(SP), 4(SP)            ; cmp N1, N
86              jl      ExitIf1
87              mov.w   8(SP), 2(SP)            ; M = N1
88              ret
89  ExitIf1:
90              cmp.w   4(SP), 6(SP)            ; cmp N, N2
91              jge     ExitIf2
92              mov.w   4(SP), 2(SP)            ; M = N
93              ret
94  ExitIf2:
95              mov.w   6(SP), 2(SP)            ; M = N2
96              ret
97
98  ;-------------------------------------------------------------------------------
99  ; Stack Pointer definition
100 ;-------------------------------------------------------------------------------
101             .global __STACK_END
102             .sect   .stack
103
104 ;-------------------------------------------------------------------------------
105 ; Interrupt Vectors
106 ;-------------------------------------------------------------------------------
107             .sect   ".reset"                ; MSP430 RESET Vector
108             .short  RESET
109
```