```
1  ;-------------------------------------------------------------------------------m
2  ; MSP430 Assembler Code Template for use with TI Code Composer Studio
3  ;
4  ;
5  ;-------------------------------------------------------------------------------
6              .cdecls C,LIST,"msp430g2553.h"       ; Include device header file
7
8  ;-------------------------------------------------------------------------------
9              .text                            ; Assemble into program memory
10             .retain                          ; Override ELF conditional linking
11                                              ; and retain current section
12             .retainrefs                      ; Additionally retain any sections
13             .global RESET                    ; that have references to current
14                                              ; section
15 ;-------------------------------------------------------------------------------
16
17 RESET       mov.w   #0x0400,SP                  ; Initialize stackpointer
18
19 StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL      ; Stop WDT
20
21 SetupDCO      clr.b  &DCOCTL                                  ; set DCO to operate at 16MHz
22                    mov.b &CALBC1_16MHZ,&BCSCTL1               ; Set range
23                    mov.b &CALDCO_16MHZ,&DCOCTL                ; Set DCO step + modulation
24
25 SetupTA1      mov.w  #CCIE, &TA1CCTL0                          ; enable timer interrupt
26                    mov.w  #1016,&TA1CCR0                      ; inilize compare value to 1
   line period
27                    mov.w  #TASSEL_2+MC_0, &TA1CTL             ; setup to use SMCLK; up mode
28
29 SetupTA0      mov.w  #CCIE, &TA0CCTL0                          ; enable timer interrupt
30                    mov.w  #1016,&TA0CCR0                      ; inilize compare value to 1
   line period
31                    mov.w  #TASSEL_2+MC_0, &TA0CTL             ; setup to use SMCLK; up mode
32
33                                                               ; setup USCI for SPI mode
34 SetupUSCI     bis.b  #UCSWRST,&UCB0CTL1                       ; Disable USCI
35                    bis.b  #BIT7,&P1SEL                        ; configure P1.7 As USCIB0_SIMO
   (Master Output)
36                    bis.b  #BIT7,&P1SEL2                       ;
37
38                    bis.b  #UCSYNC+UCMST,&UCB0CTL0             ; Master Mode, 3 pin SPI, 8 bit,
   LSB 1st, Syncronous
39                    bis.b  #UCSSEL_2,&UCB0CTL1                 ; SMCLK as input to Bit Clock
40                    mov.b  #4,&UCB0BR0                               ; Low Byte Bit Clock
   Divisor
41                    mov.b  #0,&UCB0BR1                               ; Hi Byte Bit Clock
   Divisor: Bit Clock = SMCLK/(UCB0BR0 + UCB0BR1*256) = 4MHz
42                    bic.b  #UCSWRST,&UCB0CTL1                  ; Enable USCI
43                    bis.b  #UCMSB,&UCB0CTL0                    ; MSB transmit mode
44
45
46 SetupPort1    bic.b  #BIT4+BIT1+BIT2+BIT3, &P2DIR             ; 2.1 - 2.4 Input
47               bis.b  #BIT7+BIT6, &P1DIR                       ; 1.7 and 6 for Output
48                    bic.b  #BIT6, &P1OUT                ; Turn LED OFF
49
50 SetupPort2    bis.b  #BIT7+BIT6+BIT5+BIT0, &P2DIR ; Set P2.7, .6, .5,& .0 as Output. P2.1, .2,
   .3, & .4 as Input. P2.0 only output used.
```

```
51                      bis.b   #BIT0, &P2OUT                    ; Turn P2.0 ON
52                      bic.b   #BIT5, &P2OUT                    ; Turn P2.5 Off
53
54 SetupPortREN  bis.b  #BIT4+BIT1+BIT2+BIT3, &P2REN            ; 2.1 - 2.4 Input (resistor
   enable because of the blasted noise)
55
56
57
58 ; 2.0 on = 0.4
59 ; 2.5 + 2.0 on 1.4
60
61
62
63 ;-------------------------------------------------------------------------------
64                                              ; Main loop here
65 ;-------------------------------------------------------------------------------
66
67 ; Code Goes Here
68
69 ; Paddle Height is 28
70
71
72
73 CurrentLine .EQU R4
74 PaddleLeft_Y .EQU R5
75 PaddleRight_Y .EQU R6
76 Ball_X .EQU R7
77 Ball_Y .EQU R8
78 BallVelocity_X .EQU R9
79 BallVelocity_Y .EQU R10
80 JumpTimer .EQU R11
81 BKTimer .EQU R12
82 LineBufferAddr .EQU R13
83 Unused4 .EQU R15
84 GameReg .EQU R14
85 BallSpeed .EQU 2
86
87 ; Set up scoreboard
88                      mov.w   #ScoreBoard, GameReg
89                      mov.b   #0x00, 0(GameReg)
90                      mov.b   #0x01, 1(GameReg)
91                      mov.b   #0x00, 2(GameReg)
92                      mov.b   #0x80, 3(GameReg)
93                      mov.b   #0x00, 4(GameReg)
94                      mov.b   #0x00, 5(GameReg)
95                      mov.b   #0x00, 6(GameReg)
96                      mov.b   #0x00, 7(GameReg)
97                      mov.b   #0x00, 8(GameReg)
98                      mov.b   #0x00, 9(GameReg)
99                      mov.b   #0x00, 10(GameReg)
100                     mov.b   #0x00, 11(GameReg)
101                     mov.b   #0x00, 12(GameReg)
102                     mov.b   #0x00, 13(GameReg)
103                     mov.b   #0x00, 14(GameReg)
104                     mov.b   #0x00, 15(GameReg)
105                     mov.b   #0x00, 16(GameReg)
106                     mov.b   #0x01, 17(GameReg)
```

```
107                     mov.b   #0x00, 18(GameReg)
108                     mov.b   #0x80, 19(GameReg)
109
110
111                      mov.w #83, JumpTimer
112 TimerInit
113                      mov.w  #TASSEL_2+MC_1,&TA1CTL ;Start Timer A (5, 5)
114 TimerInitJmp         dec.w JumpTimer
115                      jnz TimerInitJmp   ; (58*3)+5 = 179
116                      nop ;180
117                      nop ;181
118                      mov.w  #TASSEL_2+MC_1,&TA0CTL ;Start Timer B
119                      bic.w  #BIT4,&TA0CCTL0                          ; disable timer interrupt
120
121
122
123
124
125             ;Initalize all the posistions of the paddles and stuff
126
127             mov.w   #120, PaddleLeft_Y                  ;
128             mov.w   #120, PaddleRight_Y                 ; They're about halfway down the
    playfield
129             mov.w   #86, Ball_X                         ; The ball is halfway (+/- a pixel or
    two somewhere like that) in the screen
130             mov.w   #42+5, Ball_Y                       ; The ball is on the 5th line from the
    top
131             mov.w   #-1, BallVelocity_X                 ; The ball starts moving left
132             mov.w   #1,  BallVelocity_Y                 ; and down
133             mov.w   #LineBuffer, LineBufferAddr
134             mov.w   #LineBuffer+20, R15
135 IntLoop:    mov.w   #0, 0(LineBufferAddr)
136             add.w   #2, LineBufferAddr
137             cmp.w   R15, LineBufferAddr
138             jl      IntLoop
139
140             mov.w   #0, CurrentLine
141             bis.b   #GIE+CPUOFF,SR
142 CPU_OFF:
143             jmp CPU_OFF
144             nop
145
146
147
148 TIMERRESET:                                 ; Come in with (6) cycles from interrupt
149             bic.b   #BIT0,&P2OUT            ; Turn P2.0 OFF (4, 4)
150             cmp.w   #1,CurrentLine          ; (1, 5)
151             jl      BlankGameCalc           ; (2, 7)
152             cmp.w   #3,CurrentLine          ; (2, 9)
153             jl      Blank                   ; (2, 11)
154             cmp.w   #6,CurrentLine          ; (2, 13)
155             jl      VSyncStart              ; (2, 15)
156             cmp.w   #40,CurrentLine         ; (2, 17)
157             jl      Blank2                  ; (2, 19)
158             cmp.w   #230, CurrentLine       ; (2, 21)
159             jl      VisibleArea             ; (2, 23)
160             cmp.w   #262, CurrentLine       ; (2, 25)
```

```
161             jl      Blank3                  ; (2, 27)
162
163             ; --
164
165             mov.w   #15,JumpTimer       ; (2, 29)
166 ISRJump:    dec.w   JumpTimer           ;
167             jnz     ISRJump             ; 3*15 + 29 = 74
168             nop
169             nop
170             bis.b   #BIT0,&P2OUT        ; Turn P2.0 ON
171             mov.w   #0, CurrentLine
172             reti
173
174             ; --
175
176 BlankGameCalc:                         ; LINE 1 of the VSYNC will have the Game Calcluations
177                                         ; (Come in with 7 cycles)
178             mov.w   #22,JumpTimer       ; (2, 9)
179 GameJMP:    dec.w   JumpTimer           ;
180             jnz     GameJMP             ; 3*22 + 9 = 75
181             nop                         ; 76 cycles
182             bis.b   #BIT0,&P2OUT        ; Turn P2.0 ON
183
184             ; 1.0 is up   p1
185             ; 1.1 is down p1
186             ; 1.2 is up   p2
187             ; 1.3 is down p2
188             ;PaddleLeft_Y
189             ;PaddleRight_Y
190             ;Ball_X             This is what you will be manipulating
191             ;Ball_Y
192             ;BallVelocity_X
193             ;BallVelocity_Y
194
195             ; LINE BOUNDARIES QUICK REF GUIDE---  Line 232 is the bottom / Line 42 is the the
    top
196             ;                                       YOU HAVE 190 LINES TO DO STUFF WITH
197             ;                                       NOW GET TO WORK AND FINISH THIS BEFORE
198             ;                                       THE DEADLINE YOU SLACKER
199
200             ;       Oh yeah, forgot horiziontal boundaries
201             ;       These are values for the top left pixel of the ball
202             ;       Ball_X = 8 Ball Is half in the left goal, half out (Basically a point, but
    this won't be shown)
203             ;       '    ' = 9 Confirmed to be resting on the left goal
204             ;       '    ' = 164 is resting on right goal line (165 will be a point but not
    shown)
205             ;       LASTLY THE BALL_Y IS THE SAME VALUES AS BEFORE,
206             ;                                   42 IS TOP 232 IS BOTTOM (but to rest on the
    bt. it's 232-4 (ball height)
207
208
209             ; AND IN CASE YOU FORGOT, THE PADDLE HEIGHT IS 28
210             ; Ball height is 4 width is 2
211
212
213             ;       GET CRACKIN!
```

```
214
215             bic.w   #BIT4,&TA0CCTL0                         ; disable timer interrupt
216
217 P1Up:       bit.w   #BIT1, &P2IN
218             jnz     P1Down
219             decd.w  PaddleLeft_Y
220
221 P1Down:     bit.w   #BIT2, &P2IN
222             jnz     P2Up
223             incd.w  PaddleLeft_Y
224
225 P2Up:       bit.w   #BIT3, &P2IN
226             jnz     P2Down
227             decd.w  PaddleRight_Y
228
229 P2Down:     bit.w   #BIT4, &P2IN
230             jnz     PaddleLeftTop
231             incd.w  PaddleRight_Y
232
233 PaddleLeftTop:      cmp.w   #42, PaddleLeft_Y
234                     jge     PaddleLeftBottom
235                     mov.w   #42, PaddleLeft_Y
236
237 PaddleLeftBottom:   cmp.w   #233-28, PaddleLeft_Y
238                     jl  PaddleRightTop
239                     mov.w   #232-28, PaddleLeft_Y
240
241 PaddleRightTop:     cmp.w   #42, PaddleRight_Y
242                     jge     PaddleRightBottom
243                     mov.w   #42, PaddleRight_Y
244
245 PaddleRightBottom:  cmp.w   #233-28, PaddleRight_Y
246                     jl      BallMovement
247                     mov.w   #232-28, PaddleRight_Y
248
249 BallMovement:       add.w   BallVelocity_X, Ball_X
250                     add.w   BallVelocity_Y, Ball_Y
251
252 BallBounce:         cmp.w   #42, Ball_Y
253                     jge     BallBounce2
254                     mov.w   #BallSpeed, BallVelocity_Y
255                     mov.w   #42, Ball_Y
256 BallBounce2:        cmp.w   #233-4, Ball_Y
257                     jl      BallPoint
258                     mov.w   #-BallSpeed, BallVelocity_Y
259                     mov.w   #233-4, Ball_Y
260
261 BallPoint:          cmp.w   #9, Ball_X
262                     jge     BallPoint2
263                     mov.w   #86, Ball_X
264                     sub.w   #53, Ball_Y
265                     mov.w   #-1, BallVelocity_X
266                     mov.w   #1, BallVelocity_Y
267                     ;Player Two Score Increase
    Start----------------------------------------------------
268                     mov.w   #ScoreBoard, JumpTimer
269                     mov.b   18(JumpTimer), GameReg
```

```
270                     inv.b   GameReg
271                     rla.b   GameReg
272                     inv.b   GameReg
273                     mov.b   GameReg, 18(JumpTimer)
274                     cmp.b   #0xFF, GameReg
275                     jnz     BallPoint2
276                     mov.w   #0, BallVelocity_X
277                     mov.w   #0, BallVelocity_Y
278                     ;Player Two Score Increase
    End-----------------------------------------------------
279 BallPoint2:         cmp.w   #164, Ball_X
280                     jl      PaddleLeftHit
281                     mov.w   #86, Ball_X
282                     sub.w   #53, Ball_Y
283                     mov.w   #1, BallVelocity_X
284                     mov.w   #1, BallVelocity_Y
285
286                     ;P1 Score INcrease start
    ------------------------------------------------------------
287
288                     mov.w   #ScoreBoard, JumpTimer
289                     mov.b   2(JumpTimer), GameReg
290                     inv.b   GameReg
291                     rla.b   GameReg
292                     inv.b   GameReg
293                     mov.b   GameReg, 2(JumpTimer)
294                     cmp.b   #0xFF, GameReg
295                     jnz     BallPoint2
296                     mov.w   #0, BallVelocity_X
297                     mov.w   #0, BallVelocity_Y
298                     ; P1 Score increase end
    -------------------------------------------------------
299
300 PaddleLeftHit:      cmp.w   #13, Ball_X
301                     jge     PaddleRightHit
302                     mov.w   PaddleLeft_Y, GameReg
303                     add.w   #28, GameReg
304                     cmp.w   GameReg, Ball_Y
305                     jge     PaddleRightHit
306                     mov.w   Ball_Y, GameReg
307                     add.w   #4, GameReg
308                     cmp.w   PaddleLeft_Y, GameReg
309                     jl      PaddleRightHit
310                     mov.w   #BallSpeed, BallVelocity_X
311
312 PaddleRightHit:     cmp.w   #160, Ball_X
313                     jl      OuttaHere
314                     mov.w   PaddleRight_Y, GameReg
315                     add.w   #28, GameReg
316                     cmp.w   GameReg, Ball_Y
317                     jge     OuttaHere
318                     mov.w   Ball_Y, GameReg
319                     add.w   #4, GameReg
320                     cmp.w   PaddleRight_Y, GameReg
321                     jl      OuttaHere
322                     mov.w   #-BallSpeed, BallVelocity_X
323
```

```
324 OuttaHere:
325
326
327             inc.w   CurrentLine
328             reti
329
330 Blank:                                  ; LINE 1 of the VSYNC will have the Game Calcluations
331                                         ; (Come in with 11 cycles)
332             mov.w   #21,JumpTimer       ; (2, 13)
333 BJump:      dec.w   JumpTimer           ;
334             jnz     BJump               ; 3*21 + 13 = 76
335             bis.b   #BIT0,&P2OUT        ; Turn P2.0 ON
336             inc.w   CurrentLine
337             reti
338
339 VSyncStart:                            ; LINE 1 of the VSYNC will have the Game Calcluations
340                                         ; (Come in with 15 cycles)
341             mov.w   #307,JumpTimer      ; (2, 17)
342 VJump:      dec.w   JumpTimer           ;
343             jnz     VJump               ; 3*307 + 17 = 938
344             nop
345             nop                         ;(940)
346             bis.b   #BIT0,&P2OUT        ; Turn P2.0 ON
347             inc.w   CurrentLine
348
349             reti;
350
351
352 Blank2:                                 ; LINE 1 of the VSYNC will have the Game Calcluations
353                                         ; (Come in with 19 cycles)
354             mov.w   #18,JumpTimer       ; (2, 21)
355 BJump2:     dec.w   JumpTimer           ;
356             jnz     BJump2              ; 3*18 + 21 = 75
357             nop                         ; (1, 76)
358
359
360             bis.b   #BIT0,&P2OUT                    ; (4, 4) turn on p2.0
361             mov.w   #BlankLine, LineBufferAddr      ; (2, 6)
362             mov.w   #BlankLine+20, R15              ; (2, 8)
363             mov.w   #51, JumpTimer                  ; (2, 10)
364 BLNKJMP:    dec.w   JumpTimer
365             jnz     BLNKJMP                         ; 10+(51*3) = 163
366
367             inc.w   CurrentLine         ;(164)
368
369             mov.b   @LineBufferAddr+, &UCB0TXBUF
370             nop
371             nop
372             nop
373             nop
374 TXOut2:     mov.b   @LineBufferAddr+, &UCB0TXBUF        ; (5,5)
375             mov.w   #7, JumpTimer           ; (2,7)
376 OutputJMP2: dec.w   JumpTimer
377             jnz OutputJMP2
378             cmp.w   R15,LineBufferAddr                  ; (1, )
379             nop
380             jl      TXOut2                  ; (2, )
```

```
381             reti
382
383 Blank3:                                ; LINE 1 of the VSYNC will have the Game Calcluations
384                                         ; (Come in with 27 cycles)
385             mov.w   #15,JumpTimer       ; (2, 29)
386 BJump3:     dec.w   JumpTimer           ;
387             jnz     BJump3              ; 3*15 + 29 = 74
388             nop                         ; (1, 75)
389             nop                         ; (1, 76)
390
391             bis.b   #BIT0,&P2OUT                    ; (4, 4) turn on p2.0
392             mov.w   #ScoreBoard, LineBufferAddr     ; (2, 6)
393             mov.w   #ScoreBoard+20, R15             ; (2, 8)
394             mov.w   #51, JumpTimer                  ; (2, 10)
395 BLNKJMP3:   dec.w   JumpTimer
396             jnz     BLNKJMP3                        ; 10+(51*3) = 163
397
398             inc.w   CurrentLine         ;(164)
399
400             mov.b   @LineBufferAddr+, &UCB0TXBUF
401             nop
402             nop
403             nop
404             nop
405 TXOut3:     mov.b   @LineBufferAddr+, &UCB0TXBUF        ; (5,5)
406             mov.w   #7, JumpTimer           ; (2,7)
407 OutputJMP3: dec.w   JumpTimer
408             jnz OutputJMP3
409             cmp.w   R15,LineBufferAddr                  ; (1, )
410             nop
411             jl      TXOut3                  ; (2, )
412             bic.w   #BIT4,&TA0CCTL0                         ; disable timer interrupt
413             reti
414
415
416
417 ; 2.0 on = 0.4
418 ; 2.5 + 2.0 on 1.4
419
420
421
422 VisibleArea:                            ; Lines 20-262 will be the visible area
423                                         ;---------------------------------------
424                                         ;
425                                         ; First of all you need to be at 0v for the
    Horiziontal Sync Pulse
426                                         ; That will last for 75 cycles
427                                         ;
428                                         ; Then the Prescan area will be at 0.4v for 94 cycles
429                                         ;
430                                         ; Then the Visible area is next. It lasts for 824
    cycles. (When you activate the 1.4v pin)
431                                         ;
432                                         ; Then the front porch is 0.4v at 22 cycles
433                                         ;
434                                         ; Lather Rinse and repeat 242 times!
435
```

```
436 ; ------------- HORIZIONTAL SYNC (76 CYCLES)
437
438
439                                         ; COME IN with 23 CYCLES
440
441             mov.w   #17,JumpTimer       ; (2, 25)
442 HSyncJump:  dec.w   JumpTimer           ;
443             jnz     HSyncJump           ; 17*3 + 25 = 76
444 ; ------------- BACK PORCH (181 CYCLES)
445             bis.b   #BIT0,&P2OUT        ; (4, 4) turn on p2.0
446             bis.w   #BIT4,&TA0CCTL0                          ; enable timer interrupt
447             mov.w   #LineBuffer, LineBufferAddr ; (2, 6)
448             mov.w   #LineBuffer+20, R15 ; (2, 8)
449             inc.w   CurrentLine         ; (1, 9)
450             mov.w   &TA1R, BKTimer      ; (3, 12)
451             add.w   #169, BKTimer
452
453 CheckLeftP: cmp.w   PaddleLeft_Y, CurrentLine   ;(1, 10)
454             jl      CheckRightP1                ;(2, 12)
455             mov.w   PaddleLeft_Y, GameReg       ;(1, 13)
456             add.w   #28, GameReg                ;(2, 15)
457             cmp.w   GameReg, CurrentLine        ;(1, 16)
458             jge     CheckRightP2                ;(2, 18)
459             bis.b   #BIT4+BIT3+BIT7, 0(LineBufferAddr)      ;(5, 23)
460             jmp     CheckRightP3                ;(2, 25)
461 CheckRightP1: ;12 cycles
462             nop
463
464 CheckRightP2: ;18 cycles
465             bis.b   #BIT7, 0(LineBufferAddr)    ;(5, 23)
466             nop
467 CheckRightP3:   ; COME IN WITH 25 CYCLES
468             cmp.w   PaddleRight_Y, CurrentLine      ;(1, 26)   cmp = dest - src
469             jl      CheckBallX1                     ;(2, 28)   cmp src, dest
470             mov.w   PaddleRight_Y, GameReg          ;(1, 29)
471             add.w   #28, GameReg                    ;(2, 31)
472             cmp.w   GameReg, CurrentLine            ;(1, 32)
473             jge     CheckBallX2                     ;(2, 34)
474             bis.b   #BIT6+BIT5+BIT1, 19(LineBufferAddr)         ;(5, 39)
475             jmp     CheckBallX3                     ;(2, 41)
476 CheckBallX1: ;28 cycles
477             nop
478 CheckBallX2: ;34 cycles
479             bis.b   #BIT1, 19(LineBufferAddr)
480             nop
481 CheckBallX3: ; COME IN WITH 42 CYCLES
482             cmp.w   Ball_Y, CurrentLine
483             jl      DottedLineX
484             mov.w   Ball_Y, GameReg
485             add.w   #4, GameReg
486             cmp.w   GameReg, CurrentLine
487             jge     DottedLineX
488             mov.w   #-1, GameReg
489 StartBallX: sub.w   #1, LineBufferAddr
490             mov.w   Ball_X, GameReg
491             bic.w   #7, GameReg
492             rra.w   GameReg
```

```
493             rra.w   GameReg
494             rra.w   GameReg
495             add.w   GameReg, LineBufferAddr
496 BallBit:    mov.b   #192, GameReg
497             mov.b   Ball_X, JumpTimer
498             bic.b   #248, JumpTimer
499             cmp.b   #0, JumpTimer
500             jz      WriteBall
501 BitDec:     rra.w   GameReg
502             dec.b   JumpTimer
503             jnz     BitDec
504 WriteBall:  bis.b   GameReg, 0(LineBufferAddr)
505             cmp.b   #1, GameReg
506             jnz     DottedLineX
507             bis.b   #128, 1(LineBufferAddr)
508
509 DottedLineX:
510
511             mov.w   #LineBuffer, LineBufferAddr
512
513             reti
514
515
516             ; VISIBLE TRANSMITTER
517
518 TimerBTX:   mov.b   @LineBufferAddr+, &UCB0TXBUF        ; (5,5)
519             mov.b   #0, -1(LineBufferAddr)              ; (4, 9)
520 TXOut:      mov.b   @LineBufferAddr+, &UCB0TXBUF        ; (5,5)
521             mov.b   #0, -1(LineBufferAddr)              ; (4, 9)
522             mov.w   #6, JumpTimer           ; (2,11)
523 OutputJMP:  dec.w   JumpTimer               ;
524             jnz OutputJMP                   ;
525             cmp.w   R15,LineBufferAddr                  ; (1, 30)
526             jl      TXOut                   ; (2, 32)
527             reti
528
529 ; ------------- VISIBLE AREA (777 CYCLES)
530
531
532 ; Line Variables
533
534  .bss LineBuffer, 20
535  .bss PlayerScores, 2 ; The First Byte is the Left Paddle, Second is the right
536  .bss ScoreBoard, 20
537 DottedLine .int 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA, 0xAAAA
538 BlankLine .int 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000
539
540
541 ;-------------------------------------------------------------------------------
542 ;           Stack Pointer definition
543 ;-------------------------------------------------------------------------------
544             .global __STACK_END
545             .sect   .stack
546
547
548
549 ;-------------------------------------------------------------------------------------------
```

```
550 ERRANT_ISR
551               bis.b  #001h, &P1OUT              ; P1.0 = ON
552               jmp    ERRANT_ISR
553 ;-----------------------------------------------------------------------
554 ;           Interrupt Vectors
555 ;-----------------------------------------------------------------------
556             .sect    ".int00"                          ;Not Used
557             .short   ERRANT_ISR
558             .sect    ".int01"                          ;Not Used
559             .short   ERRANT_ISR
560              .sect   ".int02"                          ;PORT1
561             .short   ERRANT_ISR
562              .sect   ".int03"                          ;PORT2
563             .short   ERRANT_ISR
564              .sect   ".int04"                          ;Not Used
565             .short   ERRANT_ISR
566              .sect   ".int05"                          ;ADC10
567             .short   ERRANT_ISR
568              .sect   ".int06"                          ;USCIAB0TX
569             .short   ERRANT_ISR
570              .sect   ".int07"                          ;USCIAB0RX
571             .short   TimerBTX
572              .sect   ".int08"                          ;Timer0_A1
573             .short   TimerBTX
574              .sect   ".int09"                          ;Timer0_A0
575             .short   TimerBTX                        ; TA0_ISR
576              .sect   ".int10"                          ;WDT
577             .short   ERRANT_ISR
578              .sect   ".int11"                          ;COMPA
579             .short   TimerBTX
580              .sect   ".int12"                          ;Timer1_A1
581             .short   TimerBTX
582              .sect   ".int13"                          ;Timer1_A0
583             .short   TIMERRESET                       ;TA1_ISR
584              .sect   ".int14"                          ;NMI
585             .short   ERRANT_ISR
586              .sect   ".reset"                          ;RESET
587             .short   RESET
588
```