



TouchMetal Sample



Sample Program 3 is a simple dice game.

Author : Dr Nicholas J. Macias

```
00001      ;
00002      ; dice game
00003      ; (for entertainment only!)
00004      ; NJM May 2026
00005      ;
00006      ; Press INT to roll the die
00007      ; they will be displayed in the output register (rightmost 2 digits)
00008      ; 7 or 11 wins
00009      ; 2, 3, 12 loses
00010      ; anything else becomes your point.
00011      ;
00012      ; Press INT to keep rolling. Makee your point and win; roll 7 and lose.
00013      ;
00014      ; If you win, the display flashes 077/011
00015      ; If you lose, the display shows 000
00016      ;
00017
00018      ; interrupt handler
00019      .org 376
00020 376:004 003      jmp  IntHandler
00021
00022      .org 0
00023 000:030      inte      ; enable interrupts
00024 001:004 016      jmp  NewGame      ; go to start of game
00025
00026      ; put interrupt code here
00027      IntHandler:
00028 003:101      inc  R1      ; set flag
00029 004:032      rti      ; and return
00030
00031      ; wait for an interrupt
00032      getInt:
00033 005:121      clr  R1      ; interrupt handler will set this
00034 006:070 226      ldi  R0,var_c      ; address of var_c (starting seed)
00035      waitInt:
00036 010:300      inc  (R0)      ; change the salt
00037 011:027 011      mov  R1,R1      ; check flag
00038 013:005 010      jz   waitInt
00039 015:001      ret      ; flag was set - user pressed INT!
00040
00041
00042      ; play a game!
00043      NewGame:
00044 016:003 005      call getInt      ; user asks to roll the dice
```



TouchMetal Sample



```
00045      NewGameGo:
00046 020:120      clr  R0
00047 021:170      out  R0
00048 022:003 143  call rollDice      ; do this and display dice
00049                                     ; return sum in R0
00050                                     ; winner?
00051 024:071 013      ldi  R1,11.
00052 026:021 001      sub  R0,R1
00053 030:005 106      jz   Winner      ; 11
00054 032:071 007      ldi  R1,7.
00055 034:021 001      sub  R0,R1
00056 036:005 106      jz   Winner      ; 7
00057                                     ; lose?
00058
00059 040:071 002      ldi  R1,2.
00060 042:021 001      sub  R0,R1
00061 044:005 125      jz   Lose        ; 2
00062 046:071 003      ldi  R1,3.
00063 050:021 001      sub  R0,R1
00064 052:005 125      jz   Lose        ; 3
00065 054:071 014      ldi  R1,12.
00066 056:021 001      sub  R0,R1
00067 060:005 125      jz   Lose        ; 12.
00068
00069                                     ; save your point
00070 062:027 007      mov  R0,R7      ; R7 is your point
00071      RollMore:
00072 064:003 005      call getInt
00073 066:003 143      call rollDice
00074 070:071 007      ldi  R1,7.
00075 072:021 001      sub  R0,R1
00076 074:005 125      jz   Lose        ; 7 means you lose
00077 076:027 071      mov  R7,R1
00078 100:021 001      sub  R0,R1
00079 102:005 106      jz   Winner      ; made your point!
00080 104:004 064      jmp  RollMore
00081
00082      Winner:
00083 106:121      clr  R1      ; int flag
00084      winnerWait:
00085 107:003 220      call Wait1
00086 111:070 077      ldi  R0,077
00087 113:170      out  R0      ; display 077 for win
00088 114:003 220      call Wait1
00089 116:174      out  R4      ; the last roll
00090 117:027 011      mov  R1,R1
00091 121:005 107      jz   winnerWait  ; still waiting
00092 123:004 020      jmp  NewGameGo
00093
```



TouchMetal Sample



```
00094          Lose:
00095 125:121      clr  R1
00096          loseWait:
00097 126:003 220   call Wait1
00098 130:120      clr  R0
00099 131:170      out  R0                ; display 000 for lose
00100 132:003 220   call Wait1
00101 134:174      out  R4                ; the last roll
00102 135:027 011   mov  R1,R1
00103 137:005 126   jz   loseWait        ; still waiting
00104 141:004 020   jmp  NewGameGo
00105
00106                ; simulate rolling a pair of dice
00107                ; return pair in R4 and sum in R0
00108          rollDice:
00109 143:003 230     call Rand                ; R1=random number
00110 145:070 006     ldi  R0,6
00111 147:023 010     div  R1,R0                ; R0=R1/6
00112 151:072 006     ldi  R2,6
00113 153:022 020     mul  R2,R0                ; R0=6*int(R1/6)
00114 155:021 010     sub  R1,R0                ; R0=original R1 mod 6
00115 157:100        inc  R0
00116 160:027 004     mov  R0,R4                ; 1st digit
00117
00118 162:003 230     call Rand                ; R1=random number
00119 164:070 006     ldi  R0,6
00120 166:023 010     div  R1,R0                ; R0=R1/6
00121 170:072 006     ldi  R2,6
00122 172:022 020     mul  R2,R0                ; R0=6*int(R1/6)
00123 174:021 010     sub  R1,R0                ; R1 mod 6
00124 176:100        inc  R0                ; 2nd digit
00125 177:027 003     mov  R0,R3
00126 201:020 040     add  R4,R0                ; final sum (R0 is return value)
00127                ; display R3 and R4 in output register
00128 203:144        rol  R4
00129 204:144        rol  R4
00130 205:144        rol  R4
00131 206:025 034     or   R3,R4                ; R4 is 00.R4.R3
00132 210:174        out  R4
00133
00134 211:001        ret                ; return sum (R0)
00135
00136
00137 ;***
00138 ;*** Support Functions
00139 ;***
00140
00141 ;
00142 ; Delay(n)
```



TouchMetal Sample



```
00143      ; delays for n seconds
00144      ; (approx 7% fast on my board...)
00145      ;
00146      ; input: R0
00147      ; output: none
00148      ; changes: R0
00149      ;
00150      ; NJM May 2026
00151      ;
00152
00153      Delay:
00154 212:003 220      call Wait1
00155 214:110          dec    R0
00156 215:006 212      jnz    Delay
00157 217:001          ret
00158
00159      Wait1:          ; 1-second delay
00160 220:000          nop
00161 221:000          nop
00162 222:000          nop
00163 223:000          nop
00164 224:000          nop
00165 225:001          ret
00166
00167      ;
00168      ; rand(r)
00169      ; perturbes r and returns it
00170      ;
00171      ; input: none
00172      ; output: R1
00173      ; uses: R0, R1, R2, R3
00174      ; NJM May 2026
00175      ;
00176      ; "var_c" is an incrementing additional element
00177      ;
00178 226:000          var_c:      .byte  0 ; salt
00179 227:000          lastrandnum: .byte  0 ; save last random number here
00180
00181      Rand:
00182 230:071 227      ldi    R1,lastrandnum ; address of last random num
00183 232:027 211      mov    (R1),R1        ; R1 is the last random num
00184 234:070 226      ldi    R0,var_c      ; c increments each time we call this
00185 236:300          inc    (R0)          ; do the inc
00186 237:072 137      ldi    R2,0137       ; first mask
00187 241:026 012      xor    R1,R2        ; twiddle those bits
00188 243:142          rol    R2           ; shift left one bit
00189 244:073 376      ldi    R3,0376      ; mask to clear lsb
00190 246:024 032      and    R3,R2        ; clear lsb
00191 250:073 022      ldi    R3,022       ; 2nd mask
```



TouchMetal Sample



```
00192 252:026 013    xor    R1,R3            ; twiddle that
00193 254:020 023    add     R2,R3            ; sum these
00194 256:020 203    add     (R0),R3        ; and add var_c
00195 260:070 227    ldi     R0,lastrandnum
00196 262:027 130    mov     R3,(R0)        ; save this for next time
00197 264:027 031    mov     R3,R1            ; return value
00198 266:001        ret
00199
00200                .end
```