```
Freescale HC12-Assembler
(c) Copyright Freescale 1987-2010

 Abs. Rel.   Loc    Obj. code   Source line
 ---- ----   ------ ---------   -----------
    1    1                      ;Real Time Interrupt RTI
    2    2                      ;Daniel Noyes, Andrew Haas, Benjamin Doiron
    3    3                      ;Group 11, Lab 3, Febuary 13th, 2013
    4    4
    5    5                      ;*****************************************************************¬
********
    6    6                      ;Revision Notes;
    7    7                      ;
    8    8                      ;Created by Dan [2-20-13]
    9    9                      ;revised and commented by Dan [2-22-13], [3-1-13]
   10   10                      ;
   11   11
   12   12                      ;*****************************************************************¬
********
   13   13                      ; export symbols
   14   14
   15   15                           XDEF Entry, _Startup  ; export 'Entry' symbol
   16   16                           ABSENTRY Entry      ; for absolute assembly: mark this as      ¬
entry point
   17   17
   18   18                      ;*****************************************************************¬
********
   19   19                      ; equates
   20   20
   21   21          0000 1000   RAMStart: EQU    $1000        ; absoolute address of the start of  ¬
RAM: Variables
   22   22          0000 C000   ROMSTART: EQU    $C000        ; absolute Address to place code/    ¬
constant Data
   23   23          0000 3FFF   RAM_END:  EQU    $3FFF        ; absolute address  of the End of RAM:¬
Variables
   24   24
   25   25          0000 0000   PTA:      EQU    $0000        ; Port A
   26   26          0000 0001   PTB:      EQU    $0001        ; Port B
   27   27          0000 0002   DDRA:   EQU    $0002          ; enable port A [bit specific: 1 =   ¬
output, 0 = input]
   28   28          0000 0003   DDRB:   EQU    $0003        ; enable port B [bit specific: 1 =   ¬
output, 0 = input]
   29   29
   30   30          0000 0037   CRG_FLG:  EQU    $0037        ; to clear the interupt by writing 1 ¬
to RTIF (%10000000)
   31   31          0000 0038   CRG_INT:  EQU    $0038        ; enable the interrupt
   32   32          0000 003B   RTI_CTL:  EQU    $003B        ; RTI clock rate ($40 ~ 1.024 ms)
   33   33
   34   34          0000 0001   rti_diag: EQU    %00000001      ;
   35   35          0000 0002   t1_diag:  EQU    %00000010      ; Port A & B diagnostic bits
   36   36          0000 0004   t2_diag:  EQU    %00000100      ;
   37   37          0000 0008   t3_diag:  EQU    %00001000      ;
   38   38          0000 0010   M_DIAG:   EQU    %00010000      ;
   39   39
   40   40                      ;Task flags
   41   41          0000 0001   TSK_1:    EQU %00000001      ;
   42   42          0000 0002   TSK_2:    EQU %00000010      ; Task flags
   43   43          0000 0004   TSK_3:    EQU %00000100      ;
   44   44
   45   45
   46   46                      ;*****************************************************************¬
********
   47   47                      ;variable/data section
   48   48
   49   49                          ORG RAMStart
   50   50
   51   51  a001000             T_FLG:    DS.B  1          ; task flag storage
   52   52  a001001             CNTA:     DS.B  1            ; storage for RTI
   53   53  a001002             CNTB:     DS.B  1            ; storage for RTI
   54   54  a001003             CNT2:     DS.B  1            ; storage for task 2
   55   55
   56   56
   57   57                      ;*****************************************************************¬
********
   58   58                      ; Code section
   59   59
   60   60                          ORG ROMSTART
```

```
   61   61
   62   62                      Entry:
   63   63                      _Startup:
   64   64
•
Freescale HC12-Assembler
(c) Copyright Freescale 1987-2010

 Abs. Rel.   Loc    Obj. code   Source line
 ---- ----   ------ ---------   -----------
   65   65                      ;*****************************************************************¬
********
   66   66                      ; Inilitilizing the program
   67   67
   68   68  a00C000 CF3F FF           LDS    #RAM_END       ; initialize the stack pointer.
   69   69  a00C003 8680             LDAA #%10000000     ; load the 1 in 7th bit to enable the ..¬
.
   70   70  a00C005 5A38             STAA   CRG_INT           ; and set the period counter.
   71   71  a00C007 8640             LDAA #$40           ; loads a value (hex 40) to the clock ..¬
.
   72   72  a00C009 5A3B             STAA RTI_CTL           ; for a 1 ms clock rate for the rti
   73   73
   74   74  a00C00B 86FF             LDAA #$FF           ; loads a FF (11111111) to enable all ¬
ports
   75   75  a00C00D 5A02             STAA DDRA       ; enables port A
   76   76  a00C00F 5A03             STAA DDRB         ; enables port B
   77   77  a00C011 7910 00          CLR    T_FLG           ; reset the task flag
   78   78  a00C014 7910 01          CLR    CNTA         ; reset the count variable A [rti-     ¬
subroutine]
   79   79  a00C017 7910 02          CLR    CNTB         ; reset the count variable B [rti-     ¬
subroutine]
   80   80  a00C01A 8664             LDAA #100         ; set a 100 in decimal...
   81   81  a00C01C 7A10 03          STAA CNT2         ; to set count 2 [task-2-subroutine]
   82   82  a00C01F 10EF             CLI                    ; clear the i bit
   83   83
   84   84                      ;*****************************************************************¬
********
   85   85                      ; Subroutines
   86   86
   87   87                      ;*****************************************************************¬
********
   88   88                      ; Main Routine
   89   89
   90   90  a00C021 B610 00     Main: LDAA T_FLG          ; Loads the Task flag to check for a   ¬
specific task flag
   91   91  a00C024 8501             BITA #TSK_1       ; Check the task flag towards the first¬
task flag
   92   92  a00C026 2703             BEQ    SKIP_1       ; if the flag is not checked will skip¬
to next check
   93   93  a00C028 16C0 41          JSR    TASK_1       ; jumps to task 1
   94   94
   95   95  a00C02B 8502         SKIP_1:BITA  #TSK_2         ;
   96   96  a00C02D 2703             BEQ    SKIP_2       ; skip to task 2
   97   97  a00C02F 16C0 6C          JSR    TASK_2       ; jumps to task 2
   98   98
   99   99  a00C032 8504         SKIP_2:BITA  #TSK_3       ;
  100  100  a00C034 2703             BEQ    SKIP_3       ; skip task 3
  101  101  a00C036 16C0 AC          JSR    TASK_3       ; jumps to task 3
  102  102
  103  103  a00C039 9600         SKIP_3:LDAA  PTA            ;
  104  104  a00C03B 8810             EORA #M_DIAG      ; toggle the main diagonstic bit back a¬
nd forth
  105  105  a00C03D 5A00             STAA PTA            ;    0 -> 1 -> 0 -> 1 -> 0 -> 1
  106  106  a00C03F 20E0             BRA    Main         ; main loop to the beginning and check ¬
the flags again
  107  107
  108  108                      ;*****************************************************************¬
********
  109  109                      ;Task 1
  110  110                      ;
  111  111                      ;for the operation of task one we want it to simulate process time
  112  112                      ;so it goes through a loop then returns
  113  113                      ;
  114  114
  115  115  a00C041 36          TASK_1:PSHA                ; pushes the task flag onto the stack to¬
save it
  116  116  a00C042 1410             SEI                    ; prevent rti from happening while seti¬
ng the bits
```

```
  117  117  a00C044 D600             LDAB PTA              ;
  118  118  a00C046 C802             EORB #t1_diag        ; toggle task one bit on port A
  119  119  a00C048 5B00             STAB PTA              ;
  120  120  a00C04A D601             LDAB PTB              ;
  121  121  a00C04C CA02             ORAB #t1_diag        ; set task one bit on port B                    ¬

  122  122  a00C04E 5B01             STAB PTB              ;
  123  123  a00C050 10EF             CLI                    ; allow rti
  124  124
  125  125  a00C052 8610             LDAA #$10           ; simulate process time operation
  126  126  a00C054 8100       T1.0: CMPA #$0             ; loop $10 times
  127  127  a00C056 2703             BEQ    T1.1          ;
  128  128  a00C058 43               DECA                  ; cycles:(1+1+1+3)*$10[16]+2=98
•
Freescale HC12-Assembler
(c) Copyright Freescale 1987-2010

 Abs. Rel.   Loc    Obj. code   Source line
 ---- ----   ------ ---------   -----------
  129  129  a00C059 20F9             BRA    T1.0          ;
  130  130
  131  131  a00C05B 1410       T1.1: SEI                    ; prevent rti
  132  132  a00C05D D601             LDAB PTB              ;
  133  133  a00C05F C4FD             ANDB #(t1_diag ^ $FF); clear task one bit on port B
  134  134  a00C061 5B01             STAB PTB              ;
  135  135
  136  136  a00C063 32               PULA                  ; returns the task flag from the stack          ¬
to A
  137  137  a00C064 84FE             ANDA #(TSK_1 ^ $FF)  ; Clears the task flag, prevent main            ¬
from running this task again right after
  138  138  a00C066 7A10 00          STAA T_FLG            ;
  139  139  a00C069 10EF             CLI                    ; allow rti
  140  140
  141  141  a00C06B 3D               RTS                    ; RETURN from stack
  142  142
  143  143                       ;*****************************************************************          ¬
********
  144  144                       ;Task 2
  145  145                       ;for this operation we grab a variable from the memory. This            ¬
variable was
  146  146                       ;set to a 100 during initilization. Everytime task 2 starts, it         ¬
checks to
  147  147                       ;see if the value in the memory is 0.
  148  148                       ;true  = reset the value bask to 100 and flag task 3.
  149  149                       ;false = decrease the memory and ends the routine
  150  150                       ;
  151  151
  152  152  a00C06C 36           TASK_2:PSHA               ; pushes the task flag onto the stack to        ¬
save it
  153  153  a00C06D 1410             SEI                    ;
  154  154  a00C06F D600             LDAB PTA              ;
  155  155  a00C071 C804             EORB #t2_diag       ; toggle task two bit on port A
  156  156  a00C073 5B00             STAB PTA              ;
  157  157  a00C075 D601             LDAB PTB              ;
  158  158  a00C077 CA04             ORAB #t2_diag       ; set task two bit on port B
  159  159  a00C079 5B01             STAB PTB              ;
  160  160  a00C07B 10EF             CLI                    ;
  161  161
  162  162  a00C07D B610 03          LDAA CNT2             ; grabs the memory value count 2
  163  163  a00C080 8100             CMPA #$0              ; compare the memory with 0
  164  164  a00C082 2706             BEQ    T2.1         ;
  165  165  a00C084 43               DECA                  ; will decrement until it reaches 0
  166  166  a00C085 7A10 03          STAA CNT2           ;
  167  167  a00C088 2011             BRA    T2.2         ;
  168  168
  169  169  a00C08A 8664       T2.1: LDAA #100         ; resets the value back to 100 and flag          ¬
task 3
  170  170  a00C08C 7A10 03          STAA CNT2           ;
  171  171  a00C08F 1410             SEI                    ;
  172  172  a00C091 32               PULA                  ; grab task from the stack
  173  173  a00C092 84FD             ANDA #(TSK_2 ^ $FF)  ; clear task 2
  174  174  a00C094 8804             EORA #TSK_3         ; set task 3
  175  175  a00C096 7A10 00          STAA T_FLG            ; Store Task flag
  176  176  a00C099 2008             BRA    T2.3           ;
  177  177
  178  178
  179  179
  180  180
```

```
  181  181  a00C09B 32          T2.2: PULA                ; returns the task flag from the stack          ¬
to A
  182  182  a00C09C 1410              SEI                   ;
  183  183  a00C09E 84FD              ANDA #(TSK_2 ^ $FF)  ; Clears the task flag, prevent main           ¬
from running this task again right after
  184  184  a00C0A0 7A10 00           STAA T_FLG            ;
  185  185
  186  186  a00C0A3 D601        T2.3: LDAB PTB              ;
  187  187  a00C0A5 C4FB              ANDB #(t2_diag ^ $FF); clear task two bit on port B
  188  188  a00C0A7 5B01              STAB PTB              ;
  189  189  a00C0A9 10EF              CLI                     ;
  190  190  a00C0AB 3D                RTS                     ; RETURN from stack
  191  191
  192  192                       ;*****************************************************************         ¬
*******
•
Freescale HC12-Assembler
(c) Copyright Freescale 1987-2010

 Abs. Rel.   Loc    Obj. code   Source line
 ---- ----   ------ ---------   -----------
  193  193                       ;Task 3
  194  194                       ;routine task just like task 1
  195  195                       ;
  196  196
  197  197  a00C0AC 36          TASK_3:PSHA                 ; pushes the task flag onto the stack to        ¬
save it
  198  198  a00C0AD 1410              SEI                     ;
  199  199  a00C0AF D600              LDAB PTA              ;
  200  200  a00C0B1 C808              EORB #t3_diag      ; toggle task three bit on port A
  201  201  a00C0B3 5B00              STAB PTA                ;
  202  202  a00C0B5 D601              LDAB PTB              ;
  203  203  a00C0B7 CA08              ORAB #t3_diag      ; set task three bit on port B
  204  204  a00C0B9 5B01              STAB PTB                ;
  205  205  a00C0BB 10EF              CLI                     ;
  206  206
  207  207  a00C0BD 8610              LDAA #$10          ; simulate process time operation
  208  208  a00C0BF 8100        T3.0: CMPA #$0              ; loop $10 times
  209  209  a00C0C1 2703              BEQ    T3.1           ;
  210  210  a00C0C3 43                DECA                  ; cycles:(1+1+1+3)*$10[16]+2=98
  211  211  a00C0C4 20F9              BRA    T3.0           ;
  212  212
  213  213  a00C0C6 1410        T3.1: SEI                     ;
  214  214  a00C0C8 D601              LDAB PTB              ;
  215  215  a00C0CA C4F7              ANDB #(t3_diag ^ $FF); clear task three bit on port B
  216  216  a00C0CC 5B01              STAB PTB              ;
  217  217
  218  218  a00C0CE 32                PULA                  ; returns the task flag from the stack          ¬
to A
  219  219  a00C0CF 84FB              ANDA #(TSK_3 ^ $FF)  ; Clears the task flag, prevent main           ¬
from running this task again right after
  220  220  a00C0D1 7A10 00           STAA T_FLG            ;
  221  221  a00C0D4 10EF              CLI                     ;
  222  222  a00C0D6 3D                RTS                     ; RETURN from stack
  223  223
  224  224                       ;*****************************************************************         ¬
*******
  225  225                       ;RTI
  226  226                       ;maintain two counters that will increment by 1 each time RTI is            ¬
entered
  227  227                       ;CNTA 0->4 : set task flag 1
  228  228                       ;CNTB 0->9 : set task flag 2
  229  229                       ;clear the interrupt when leaving the RTI
  230  230                       ;
  231  231
  232  232  a00C0D7 D601        RTI_ISR:LDAB PTB            ;
  233  233  a00C0D9 CA01              ORAB #rti_diag       ; set task one bit on port B                    ¬

  234  234  a00C0DB 5B01              STAB PTB              ;
  235  235  a00C0DD D600              LDAB PTA              ;
  236  236  a00C0DF C801              EORB #rti_diag       ; toggle task one bit on port A
  237  237  a00C0E1 5B00              STAB PTA              ;
  238  238
  239  239  a00C0E3 B610 01           LDAA CNTA           ;
  240  240  a00C0E6 F610 02           LDAB CNTB           ; loads CNTA & CNTB and proceed with the        ¬
loop checks
  241  241
  242  242  a00C0E9 42                INCA                  ; add to count A and check to see if it         ¬
```

```
is a 4
  243  243  a00C0EA 8104             CMPA #4           ; true  = flag task 1, reset count A
  244  244  a00C0EC 260C             BNE    RTI_A1       ; false = skips and goes to count B
  245  245  a00C0EE 8600             LDAA #0           ;
  246  246  a00C0F0 7A10 01          STAA CNTA         ;
  247  247  a00C0F3 8601             LDAA #TSK_1       ;
  248  248  a00C0F5 7A10 00          STAA T_FLG          ;
  249  249  a00C0F8 2003             BRA    RTI_B          ;
  250  250  a00C0FA 7A10 01     RTI_A1:STAA  CNTA         ;
  251  251
  252  252  a00C0FD 52          RTI_B:INCB                ; adds to count B and check to see if it            ¬
is a 9
  253  253  a00C0FE C109             CMPB #9           ; true  = flag task 2, reset count B
  254  254  a00C100 260C             BNE    RTI_B1       ; false = skip and return from the                  ¬
interupt
  255  255  a00C102 C600             LDAB #0           ;
  256  256  a00C104 7B10 02          STAB CNTB         ;
•
Freescale HC12-Assembler
(c) Copyright Freescale 1987-2010

 Abs. Rel.   Loc    Obj. code   Source line
 ---- ----   ------ ---------   -----------
  257  257  a00C107 8602             LDAA #TSK_2       ;
  258  258  a00C109 7A10 00          STAA T_FLG          ;
  259  259  a00C10C 2003             BRA    RTI_END        ;
  260  260
  261  261  a00C10E 7B10 02     RTI_B1:    STAB CNTB       ;
  262  262
  263  263  a00C111 D601        RTI_END:   LDAB PTB          ;
  264  264
  265  265  a00C113 C4FE             ANDB #(rti_diag ^ $FF) ; clear task one bit on port B
  266  266  a00C115 5B01             STAB PTB            ;
  267  267
  268  268  a00C117 8680             LDAA #%10000000     ;
  269  269  a00C119 5A37             STAA CRG_FLG        ; clear the interupt
  270  270  a00C11B 0B               RTI                 ; RETURN from interupt
  271  271
  272  272                       ;*******************************************************************         ¬
********
  273  273                       ; catch-all isr
  274  274                       ;if any onther task flags are enabled then will return from their           ¬
interupts
  275  275                       ;
  276  276
  277  277  a00C11C A7          DFLT_ISR: NOP               ; let cpu hang for a second
  278  278  a00C11D 0B               RTI                 ; RETURN from the interupt routine
  279  279
  280  280                       ;*******************************************************************         ¬
********
  281  281                       ; Interrupt Vectors [rti interupts]
  282  282                       ;writes the memory locations for various interrupts to prevent the          ¬
system
  283  283                       ;from hanging
  284  284                       ;
  285  285
  286  286                            ORG     $FFF0       ;
  287  287  a00FFF0 C0D7             DC.W RTI_ISR        ;rti
  288  288  a00FFF2 C11C             DC.W DFLT_ISR     ;irq pin
  289  289  a00FFF4 C11C             DC.W DFLT_ISR     ;xirq pin
  290  290  a00FFF6 C11C             DC.W DFLT_ISR     ;swi
  291  291  a00FFF8 C11C             DC.W DFLT_ISR     ;non_inst
  292  292  a00FFFA C11C             DC.W DFLT_ISR     ;cop fail
  293  293  a00FFFC C11C             DC.W DFLT_ISR     ;clk fail
  294  294  a00FFFE C000             DC.W    Entry       ;reset vector
  295  295
```