# How to Use 9S12DEMH1 with Adapt9S12X and CW V4.5

Connect the Adapt9S12DEMH1 and AD9S12XD boards together, with H1 and P1 headers aligned as shown in the drawing. Make sure that Pin 1 of P1 is aligned to Pin1 of H1. Also Pin 50 of both modules must be aligned. The same thing applies when using a backplane. Proper visual inspection of the connections is a must before powering up the boards. See Figure 1.

Powering up the 9S12DEMH1 will also power up the AD9S12XD board with the double ended cable provided. The LED bar is driven by Port H, DIP switch is connected to Port T, CDS, Thermistor, Pushbuttons and Potentiometer are connected to Port AD0, Beeper or Speaker is connector to Port P bit 7. See Figure 1.

**Port pins usage:**

| **SSIM - P1** | **AD9S12X – H1** | |
|---|---|---|
| 1 – LCD DB7 | 1 – PS4 | LCD DATA BUS |
| 2 – LCD DB6 | 2 – PS5 | LCD DATA BUS |
| 3 – LCD DB5 | 3 – PS6 | LCD DATA BUS |
| 4 – LCD DB4 | 4 – PS7 | LCD DATA BUS |
| 5 – LED D2 GRN | 5 – PS1 | |
| 6 – S1 | 6 – PT7 | DIP SWITCH |
| 7 – S2 | 7 – PT6 | DIP SWITCH |
| 8 – S3 | 8 – PT5 | DIP SWITCH |
| 9 – S4 | 9 – PT4 | DIP SWITCH |
| 10 – S5 | 10 – PT3 | DIP SWITCH |
| 11 – S6 | 11 – PT2 | DIP SWITCH |
| 12 – S7 | 12 – PT1 | DIP SWITCH |
| 13 – S8 | 13 – PT0 | DIP SWITCH |
| 14 – SPKR | 14 – PP7 | SPEAKER/BEEPER |
| 15 – LCD Contrast | 15 – PP6 | LCD CONTRAST DRIVER |
| 16 – VOUT | 16 – PP5 | VOUT AS PWM |
| 17 – DRIVER1/SEG8 | 17 – PP4 | MOSFET DRIVER AS PWM |
| 18 – DRIVER2/SEG9 | 18 – PP3 | MOSFET DRIVER AS PWM |
| 19 – LCD RS | 19 – PP2 | LCD REGISTER SELECT |
| 20 – LCD E | 20 – PP1 | LCD ENABLE |
| 21 – LCD R/W* | 21 – PP0 | LCD READ/WRITE |
| L3 – TEMP | 23 – PTAD01 | ANALOG TEMPERATURE |
| 24 – POT | 24 – PTAD02 | ANALOG POT |
| 25 – LIGHT | 25 – PTAD03 | ANALOG CDS |
| 26 – SW5 | 26 – PTAD07 | PUSHBUTTON |
| 27 – SW4 | 27 – PTAD06 | PUSHBUTTON |
| 28 – SW3 | 28 – PTAD05 | PUSHBUTTON |
| 29 – SW2 | 29 – PTAD04 | PUSHBUTTON |

| | | |
|---|---|---|
| 35 – LED SEG7 | 35 – PTH7 | LED BARGRAPH |
| 36 – LED SEG6 | 36 – PTH6 | LED BARGRAPH |
| 37 – LED SEG5 | 37 – PTH5 | LED BARGRAPH |
| 38 – LED SEG4 | 38 – PTH4 | LED BARGRAPH |
| 39 – LED SEG3 | 39 – PTH3 | LED BARGRAPH |
| 40 – LED SEG2 | 40 – PTH2 | LED BARGRAPH |
| 41 – LED SEG1 | 41 – PTH1 | LED BARGRAPH |
| 42 – LED SEG0 | 42 – PTH0 | LED BARGRAPH |

LCD routine not implemented, see DKKI demo example.

Note: The signals LCD Contrast  and VOUT signals are not implemented in the demo software.

**Code Explanations:**

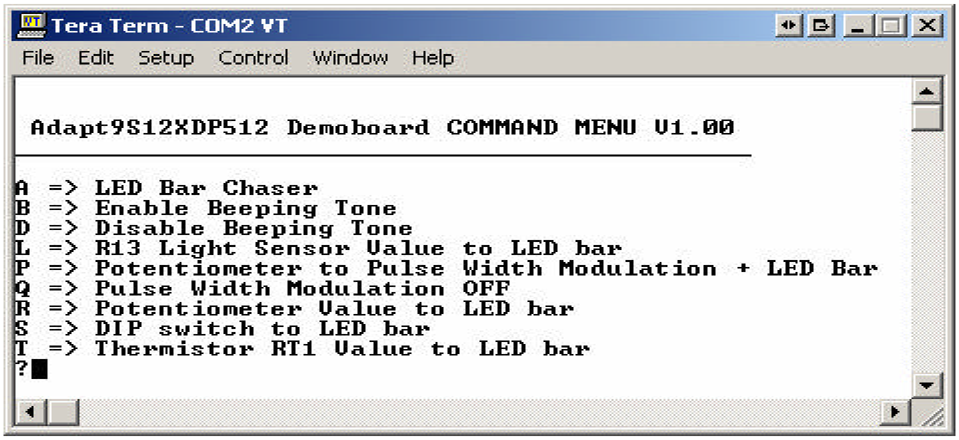
The assembly code is written in Codewarrior IDE.  CW is configured for relocateable coding.  Relocateable coding allows one to work on a single file which makes it easier for de-bugging.  In the sources directory there are 4 separate files that makes up the 9S12DEMH1 Demo.

The compiled source code can be found below this document.  Use an XGATE BDM pod to program the S-record into the MCU program memory.

Upon power or RESET of the board, the code will first initialize the RTI, OC4, A/D and SCI and then set the bus frequency to 24MHz.  This is followed by setting the priorities to be followed by the MCU when interrupts occur.

The BAUD rate is set to 115200,8,N,1.  The Analog Converter mode is set for continuous operation and multiple channels.

After hardware initialization, interrupts are enabled and the command menus are sent out to the SCI.



```
 Adapt9S12XDP512 Demoboard COMMAND MENU V1.00

A  => LED Bar Chaser
B  => Enable Beeping Tone
D  => Disable Beeping Tone
L  => R13 Light Sensor Value to LED bar
P  => Potentiometer to Pulse Width Modulation + LED Bar
Q  => Pulse Width Modulation OFF
R  => Potentiometer Value to LED bar
S  => DIP switch to LED bar
T  => Thermistor RT1 Value to LED bar
?
```

**Project files:**

**Main. Asm** – This file consists of calling the different subroutines to initialize the various hardware subsystems. Once these subsystems are initialized, the code then simply loops in **main** indefinitely. The main loop is intermittently interrupted by SCI (Serial Communication Interface), OC4 (Output Compare 4) and RTI (Real Time Interrupts).

**Int.asm** – This file consists of the RTI, and SCI interrupt service routines. Within the RTI routine, the analog values are updated.

**States.asm** – This file handles parsing of the menu commands. Whenever a command is detected, it is decoded and the appropriate action taken.

**Audio.asm** –This file consists of playing the Beeping routine. The Output Compare interrupt is serviced in this routine.

**LEDBar_Switch.ASM** –This file takes care of driving the LED Bargraph display, checking the Pushbuttons and reading the DIP switch. Anytime the Pushbuttons are pushed, the beeping routine is enabled for a short duration.

**Linker.prm** – This file consists of the Memory map and interrupt Vectors.

**Command menu explanations:**

**A => LED Bar Chaser**
This command will execute a chasing sequence on the LED bargraph. The routine is serviced within **LEDBar_Switch.ASM**

**B => Enable Beeping Tone**
This command will commence a series of Beeping tones starting from High pitch and ending with low-pitched tones. The routine is serviced within **Audio.asm**

**D => Disable Beeping Tone**
This command will abort the beeping tone sequence

**L => R13 Light Sensor Value to LED bar**
This command will copy the analog value of the light sensor to LED Bargraph display. The routine is serviced within **LEDBar_Switch.ASM**

**P => Potentiometer to Pulse Width Modulation + LED Bar**
This command will copy the analog value of the Potentiometer to LED Bargraph display and also output it to the PWM. The routine is serviced within **LEDBar_Switch.ASM**

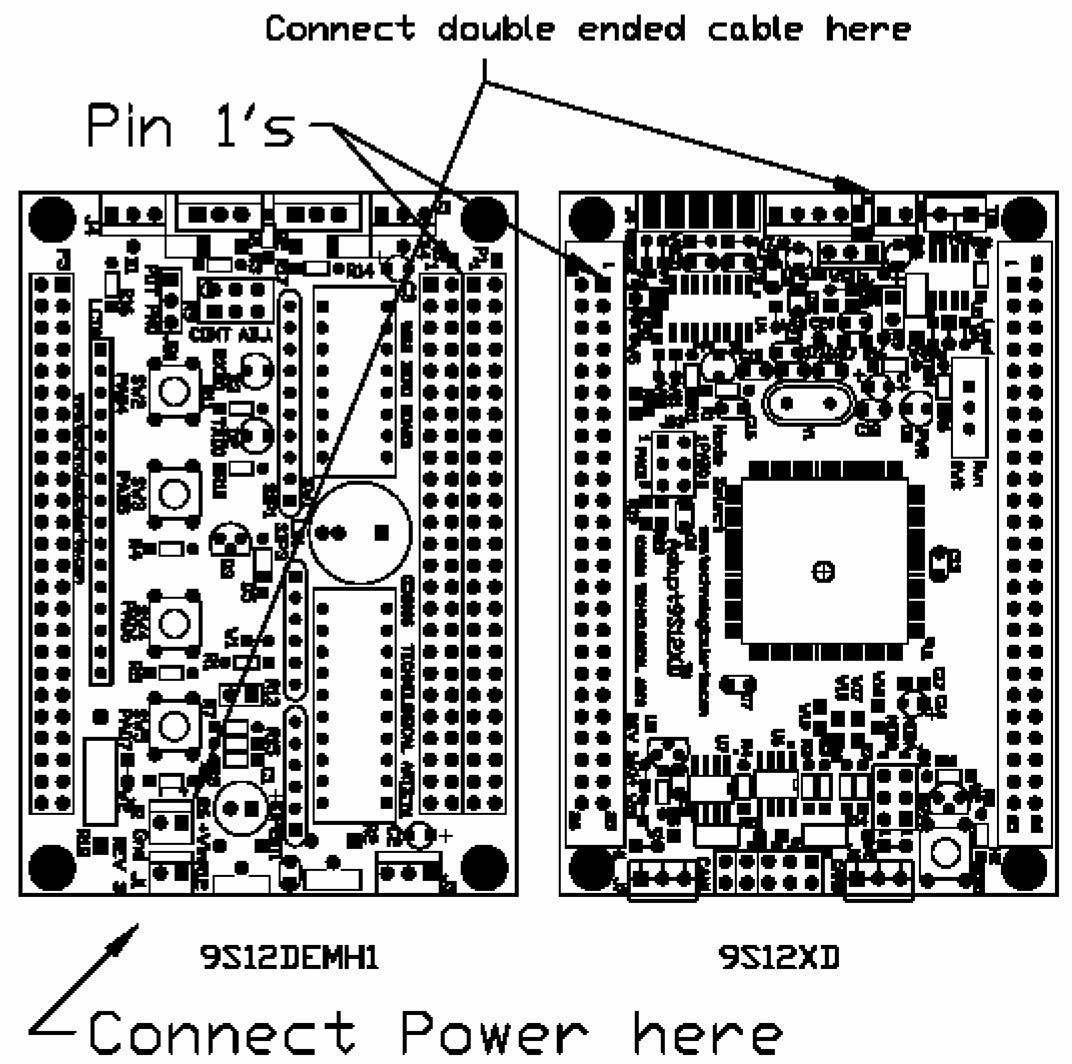
**Q => Pulse Width Modulation OFF**
This command will stop the PWM and Bargraph display.

**R => Potentiometer Value to LED bar**
This command will transfer the analog value of the Potentiometer to the LED Bargraph display only. The routine is serviced within **LEDBar_Switch.ASM**

**S => DIP switch to LED bar**
This command will copy the DIP Switch value to LED Bargraph display. The routine is serviced within **LEDBar_Switch.ASM**

**T => Thermistor RT1 Value to LED bar**
This command will copy the analog value of the Thermistor to the LED Bargraph display. The routine is serviced within **LEDBar_Switch.ASM**

**Figure 1.**



**Source Codes:**

```
;MAIN.ASM
********************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.                DESCRIPTION
*
*June 10, 2006                   1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 Demoboard app
********************************************************************************
;Compiled using CW
;
; ---------------------------
; Demoboard - Main Routine
; ---------------------------

        include "mc9s12xdp512.inc"

;Public Function
        XDEF ResetFunc
        XDEF Entry
        XDEF delay
        XDEF small_delay

;Public Variables
        XDEF Command
        XDEF CommandFlg

;External Function

        XREF RealTimeInit                   ;Initialize RTI
        XREF RealTimeInt

        XREF goPower
        XREF ShowMenu
        XREF ProcessCommand
        XREF audio_init
        XREF LEDSWInit
        XREF CheckPushbuttons

;External Variables

        XREF state
        XREF audio_state
        XREF LEDSWstate

DEFAULT_RAM:SECTION

* System Variables
```

```
Command                 ds      1               ;used by ProcessCommand
CommandFlg              ds      1               ;used by ProcessCommand

*  Operational Parameters

RAM             equ     $1000                   ;9S12DP256 internal RAM
STACK           equ     $4000                   ;Stack at top of internal ram
EEPROM          equ     $400                    ;EEPROM start address
FLASH           equ     $4000                   ;Flash start address

OscFreq         equ     16000           ;Enter Osc speed
initSYNR        equ     $01             ; mult by synr + 1 = 2 (24MHz)
initREFDV       equ     $00             ;
PLLSEL          equ     %10000000       ;PLL select bit
LOCK            equ     %00001000       ;lock status bit
PLLON           equ     %01000000       ;phase lock loop on bit


**************************** Program ******************************

NON_BANKED:SECTION

ResetFunc:
Entry                           ;This is where the RESET vector points to
        sei                             ;Disable Any interrupts

;Initialize Stack
        lds     #STACK                          ;initialize stack pointer

        jsr     RealTimeInit            ;Initialize SCI and RTI
        jsr     goPower
        jsr     audio_init              ;Initialize PORTM bit 4 and OC4 as audio o/p
        jsr     LEDSWInit

; Initialize clock generator and PLL
        bclr    CLKSEL,PLLSEL           ;disengage PLL to system
        bset    PLLCTL,PLLON            ;turn on PLL

        movb    #initSYNR,SYNR          ;set PLL multiplier
        movb    #initREFDV,REFDV        ;set PLL divider

        nop
        nop
        nop
        nop

        brclr   CRGFLG,LOCK,*+0                 ;while (!(crg.crgflg.bit.lock==1))
        bset    CLKSEL,PLLSEL           ;engage PLL to system

        movb    #$D0,INT_CFADDR                 ;Place ATD1 -> TOF into window
        movb    #%00000011,INT_CFDATA3 ;Set SCI0 to level 3 priority

        movb    #$E0,INT_CFADDR        ;Place IC0 -> IC7 into window
        movb    #%00000000,INT_CFDATA0 ;Set Timer 7 disabled
        movb    #%00000000,INT_CFDATA1 ;Set Timer 6 disabled
        movb    #%00000000,INT_CFDATA2 ;Set Timer 5 disabled
        movb    #%00000010,INT_CFDATA3 ;OC4 to level 2 priority
        movb    #%00000000,INT_CFDATA4 ;Set Timer 3 disabled
        movb    #%00000000,INT_CFDATA5 ;Set Timer 2 disabled
        movb    #%00000000,INT_CFDATA6 ;Set Timer 1 disabled
        movb    #%00000000,INT_CFDATA7 ;Set Timer 0 disabled

        movb    #$F0,INT_CFADDR        ;Place RTI -> RESET into window
        movb    #%00000100,INT_CFDATA0 ;Set RTI to level 4 priority

        clr     CommandFlg
        cli                     ;unmask interrupts
        jsr     ShowMenu

;----------------------------------------------------------------------------
main                            ;Main Loop
        jsr     ProcessCommand          ;Check if there are new command
                                ; to execute
        ldy     state
        jsr     0,y

        ldy     audio_state     ;Process audio command
        jsr     0,y

        ldy     LEDSWstate
        jsr     0,y

        jsr     CheckPushbuttons

        bra     main
;----------------------------------------------------------------------------
delay
     pshy
     ldy    #0
     bra     dly

small_delay
     pshy
     ldy    #777

dly
     dbne    y,dly
     puly
     rts

        END
```

```
;STATES.ASM
********************************************************************************
*REVISION HISTORY:
*
*DATE                REV. NO.            DESCRIPTION
*
*June 10, 2006                1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 Demoboard app
********************************************************************************
;Compiled using CW
;
; ---------------------------
; Demoboard - State Routine
; ---------------------------

        include "mc9s12xdp512.inc"

;Public Function
        XDEF ProcessCommand
        XDEF ShowMenu
        XDEF goPower

;Public Variables
        XDEF state
        XDEF state_timer

;External Function
        XREF SerOutput0
        XREF OutStr0
        XREF LEDChaserInit
        XREF SwitchToLEDInit

        XREF PlayAudioInit
        XREF BeepingDisabled

        XREF LightToLEDInit
        XREF ThermistorToLEDInit
        XREF PotentiometerToLEDInit
        XREF PotPWMInit
        XREF PotPWMDisabled


;External Variables

        XREF Command
        XREF CommandFlg

; variable/data section

DEFAULT_RAM:SECTION
```

```
state           ds      2
state_timer     ds      2
temp            ds      1
*************************** Program *****************************

; code section
NON_BANKED:SECTION

goPower
        movw  #goReady,state                ;Loop here until something to do
      movw    #0,state_timer

goReady
        rts


;-----------------------------
;Choose which one to process

ProcessCommand:
        ldaa    CommandFlg          ;commands received via SCI interrupt
        beq     ProcessCommandEx

        clr     CommandFlg

        ldaa    Command
        staa    temp

        anda    #$df                ;simple convert to upper case (only works for alpha
char)

        cmpa    #'A'
        beq     PCA

        cmpa    #'B'
        beq     PCB

        cmpa    #'D'
        beq     PCD

        cmpa    #'L'
        beq     PCL

        cmpa    #'P'
        beq     PCP

        cmpa    #'Q'
        beq     PCQ

        cmpa    #'R'
        beq     PCRR

        cmpa    #'S'
        beq     PCS

        cmpa    #'T'
        beq     PCT


ShowMenu
        ldx     #MenuMSG                ;Send Menu message
        jsr     OutStr0

ShowPrompt
        ldx     #PromptMSG
        jsr     OutStr0

ProcessCommandEx
        rts

;----------------------------------------------------------------------
;Commands executed
PCA
        jsr     LEDChaserInit
        rts

PCB
        jsr     PlayAudioInit
        rts

PCD
        jsr     BeepingDisabled
        rts

PCL
        jsr     LightToLEDInit
        rts

PCP
        jsr     PotPWMInit
        rts

PCQ
        jsr     PotPWMDisabled
        rts

PCRR
        jsr     PotentiometerToLEDInit
        rts

PCS
```

```
        jsr     SwitchToLEDInit
        rts

PCT
        jsr     ThermistorToLEDInit
        rts



*****************************************************************************************
* Messages
MenuMSG                 dc.b    $D,$A,$D,$A
                dc.b    ' Adapt9S12XDP512 9S12DEMH1 COMMAND MENU V1.00
',$D,$A
                dc.b    '_______________________________________________',$D,$A
                dc.b    $D,$A

                dc.b    'A => LED Bar Chaser',$D,$A
                dc.b    'B => Enable Beeping Tone',$D,$A
                dc.b    'D => Disable Beeping Tone',$D,$A

                dc.b    'L => R13 Light Sensor Value to LED bar',$D,$A

                dc.b    'P => Potentiometer to Pulse Width Modulation + LED Bar',$D,$A
                dc.b    'Q => Pulse Width Modulation OFF',$D,$A

                dc.b    'R => Potentiometer Value to LED bar',$D,$A

                dc.b    'S => DIP switch to LED bar',$D,$A
                dc.b    'T => Thermistor RT1 Value to LED bar',$D,$A,0


PromptMSG       dc.b    '?',0
CrlfStr       dc.b      $0a,$0d,$0


        END
```

```
;INT.ASM
***************************************************************************
*REVISION HISTORY:
*
*DATE                   REV. NO.              DESCRIPTION
*
*June 10, 2006                   1.00                     Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 Demoboard app
***************************************************************************
;Compiled using CW
;
; ---------------------------
; Demoboard - ISR Routine
; ---------------------------

        include "mc9s12xdp512.inc"


;Public Function
        XDEF RealTimeInit
        XDEF RealTimeInt

        XDEF OutStr0
        XDEF SerOutput0
        XDEF SerInputInt0

;Public Variables
        XDEF wait_timer

        XDEF ad0
        XDEF ad1
        XDEF ad2
        XDEF ad3

;External Function

        XREF check_audio

;External Variables
        XREF state_timer
        XREF delay_timer
        XREF PBDelay

        XREF Command
        XREF CommandFlg

        XREF audio_delay


DEFAULT_RAM:SECTION
```

```
;
wait_timer      ds       2

ad0            ds    2
ad1            ds    2
ad2            ds    2
ad3            ds    2

;---------------------------------------------------------------------------
;ATD Variables
admask2        equ     %11000000       ;AFFC,ADPU=1 - Enable Analog to Digital
admask3        equ     %00000000       ;FRZ1,FRZ0=0
admask4        equ     %10000001       ;SMP1,SMP0 = 0; S10BM,PRS0=1 - Select
Sample time adn Bit mode
admask5        equ     %01110000       ;S8CM = 1, SCAN = 1, MULT = 1
SCFflag        equ     %10000000       ;SCF - Sequence Complete flag

;RTI Variables
clrmask        equ     %11000000       ;mask for clearing timer flags

ms0064                   equ     %00010000    ;RTI = 16MHz/(2^10) = 0.064ms
ms0128                   equ     %00100000    ;RTI = 16MHz/(2^11) = 0.128ms
ms0256                   equ     %00110000    ;RTI = 16MHz/(2^12) = 0.256ms
ms0512                   equ     %01000000    ;RTI = 16MHz/(2^13) = 0.521ms
ms1024                   equ     %01010000    ;RTI = 16MHz/(2^14) = 1.024ms
ms2048                   equ     %01100000    ;RTI = 16MHz/(2^15) = 2.048ms
ms4096                   equ     %01110000      ;RTI = 16MHz/(2^16) = 4.096ms
ms8192                   equ     %01110001    ;RTI = 16MHz/(2*2^16) = 8.192ms


RTIF           equ     %10000000
RTIE           equ     %10000000

;SCI Variables
scimask        equ     %00101100       ;RIE - SCI Interrupt enable
                                ;RE - Receiver Enable
RDRFflag       equ     %00100000       ;RDRF - Receive Data Register Full flag
TDREflag       equ     %10000000       ;TDRE - Transmit Data Register Empty flag

;Baud rate definitions
OscFreq                  equ     16000            ;Enter Osc speed
initSYNR       equ     $01              ; mult by synr + 1 = 2 (24MHz)
initREFDV      equ     $00              ;

BusFreq                  equ     ((OscFreq/(initREFDV+1))*(initSYNR+1))
baud115200     equ     (BusFreq/16)*10/1152                    ;sets baud rate to
115,200
baud9600       equ     (BusFreq/16)*10/96                 ;sets baud rate to 009,600

* Operational Constants
```

```
TRUE           equ    $FF
FALSE          equ    $00
CR             equ    $D
LF             equ    $A
SPACE          equ    $20

TCIE            equ     $40
RIE             equ     $20
ILIE            equ     $10
TE              equ     $08
RE              equ     $04
RWU             equ     $02
SBK             equ     $01

;---------------------------------------------------------------------------

**************************** Program ******************************

NON_BANKED:SECTION
;
RealTimeInit                            ;Initialize Real Time Interrupt
        movb   #ms0256,RTICTL                   ;and initialize RTI rate
        bset   CRGFLG,RTIF                      ;clear flag
        bset   CRGINT,RTIE              ;Enable RTI

;Initialize Analog To Digital
        movb   #$F0,ATD0DIEN                    ;Make Bit 4 to 7 as input

        movb   #$80,ATD0CTL2                 ;enable ATD
        movb   #$40,ATD0CTL3                    ;
        movb   #$60,ATD0CTL4                    ;Select Sample rate
        movb   #$B0,ATD0CTL5                    ;Select 8 channel mode, Continuous scan

;Initialize first Serial Communication Interface
;
        movb   #0,SCI0BDH
        movb   #baud115200,SCI0BDL          ;..BDH=0 so baud = 9600
        movb   #0,SCI0CR1
        movb   #TE+RE+RIE,SCI0CR2           ;RIE, RE and TE on

        rts

;---------------------------------------------------------------------------
* Real-time Interrupt Routine
;
RealTimeInt
        brclr   ATD0STAT0,SCFflag,*       ;Loop here until SCF of ATD is set
                           ;save ATD

        ldd     ATD0DR0H
        std     ad0

        ldd     ATD0DR1H
        std     ad1

        ldd     ATD0DR2H
        std     ad2

        ldd     ATD0DR3H
        std     ad3

RTI0
        ldx     state_timer
        beq     RTI1
        dex
        stx     state_timer

RTI1
        ldx     wait_timer
        beq     RTI2

        dex
        stx     wait_timer

RTI2
        ldx     delay_timer
        beq     RTI3

        dex
        stx     delay_timer
RTI3
        ldx     PBDelay
        beq     RTI4

        dex
        stx     PBDelay

RTI4
        ldx     audio_delay
        beq     RTI5

        dex
        stx     audio_delay

RTI5



timex
        jsr     check_audio
        movb    #RTIF,CRGFLG
        rti


;=================================================================
============
;Send Character strings
;
OutStr0                                 ; send a null terminated string to the display.
        ldaa    1,x+            ; get a character, advance pointer, null?
        beq     OutStrDone      ; yes. return.
        bsr     SerOutput0      ; no. send it out the SCI.
        bra     OutStr0         ; go get the next character.
;
OutStrDone
        rts

;-----------------------------------------------------------------------------
SerOutput0
        brclr   SCI0SR1,TDREflag,SerOutput0     ;check if buffer is empty
        staa    SCI0DRL
        rts

;  SCI Input Interrupt Handler
;  Gets bytes from SCI.
;  Sets COMMAND_PENDING flag.

SerInputInt0
        ldaa    SCI0SR1                 ;read register to clear flag RDRF
        movb    SCI0DRL,Command         ;read receive buffer
        movb    #1,CommandFlg           ;Set for new data

SIIX
        rti



        END
```

```
;AUDIO.ASM
******************************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.               DESCRIPTION
*
*June 10, 2006                   1.00                        Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 Demoboard app
******************************************************************************************
;Compiled using CW
;
; ---------------------------
; Demoboard - Audio/Beeper Routine
; ---------------------------

        include "mc9s12xdp512.inc"
;
;PortAD bit 0 - A/D analog battery voltage
;PortAD bit 1 - A/D analog Audio from Microphone
;PortAD bit 2 - Left Front Line Sensor
;PortAD bit 3 - Right Front Line Sensor
;PortAD bit 4 - Left Rear Line Sensor
;PortAD bit 5 - Right Rear Line Sensor
;PortAD bit 6 and 7 are for the SpeakJect signal
;
;Port T/P - bit 0 as PWM
;Port T/P - bit 1 as PWM
;Port T/P - bit 2 as PWM
;Port T/P - bit 3 as PWM
;Port T - bit 4 as OC4 for BEEPER
;Port T - bit 5 as IC5 for SONAR
;Port T - bit 6 as IC6 for SONAR
;Port T - bit 7 as IC7 for SONAR
;
;Port M - bit 0 as o/p driving SONAR
;Port M - bit 1 as o/p driving SONAR
;Port M - bit 2 for SPI bus
;Port M - bit 3 for SPI bus
;Port M - bit 4 for SPI bus
;Port M - bit 5 for SPI bus
;
;Port E - bit 0
;Port E - bit 1
;

;Public Function
        XDEF check_audio
        XDEF audio_init
        XDEF set_audio
        XDEF AudioOn
```

```
            XDEF audio_int

            XDEF PlayAudioInit
            XDEF BeepingDisabled

;Public Variables
            XDEF audio_timer       ;audio timer for the beeping sound
            XDEF audio_period
            XDEF audio_status

            XDEF audio_delay
            XDEF audio_counter
            XDEF audio_state

;Public Tones
            XDEF pwr_on_tone

;External Function
            XREF OutStr0


;External variables


DEFAULT_RAM:SECTION

;Audio Variables
audio_timer:    ds      1                   ;audio tone duration timer
audio_ptr:      ds      2                   ;pointer to audio tone table
audio_period:   ds      2                   ;audio tone period
;
rep_addr:       ds      2                   ;repetition address
rep_count:      ds      1                   ;repetition counter
audio_status:   ds      1

audio_delay:    ds      2
audio_counter:          ds      1
audio_state:    ds      2

AudioTblPtr     ds      2
AudioFlg        ds      1

;
OC4mask1        equ     %00010000       ;IOS4 = 1, Bit 4 as output compare
OC4mask2        equ     %00010000       ;C4I = 1, Enable Interrupt
OC4mask3        equ     %00000001       ;OM4 = 0, OL4 = 1, toggel OC output line
OC4flag                 equ     %00010000       ;C4F = 1 to clear Interrupt flag

OM4:            equ     %10
OL4:            equ     %01
```

```
AudioControl   equu     %10000000    ;Port P bit 7 to control Audio
Seconds                 equ     3906            ;Of the RTI rate

;*******************************************************************************************
NON_BANKED:SECTION

;
; Initialization
; --------------
;
audio_init
      movw    #$1ff,audio_period
      movb    #$ff,audio_timer
         movb   #2,audio_counter       ;beep 2 times
         movw   #00,audio_delay
         clr      audio_status

         bset    DDRP,AudioControl   ;Bit 7 output
         bset    TSCR1,%10000000              ;TEN=1 - Enable timer
         bset    TIOS,OC4mask1
         bset    TIE,OC4mask2                    ;enable OC4 interrupt

         movb   #OC4flag,TFLG1                 ;clear flag
         movw   #AudioOn,audio_state
         clr      AudioFlg

audio_ready
         rts


AudioOn
         ldx      audio_delay
         bne     AudioEx

         ldx      #pwr_on_tone
         jsr      set_audio                   ;Make Power on Sound

         movw   #Seconds/2,audio_delay       ;500ms pause
         movw   #audio_ready,audio_state

AudioEx:
         rts

;-----------------------------------------------------------------
PlayAudioInit
         ldx      #AudioTable0
         stx      AudioTblPtr

         movw   #PlayAudio,audio_state
         movw   #00,audio_delay                 ;0
```

```
        ldx     #AudioChaserMsg                 ;Send Menu message
        jsr     OutStr0

        clr     AudioFlg
        rts

AudioChaserMsg          dc.b    $A,$D,'Audio/Beeper enabled',0


PlayAudio
        ldx     audio_delay
        bne     PlayAudioEx

        ldaa    AudioFlg
        beq     Audio1

        cmpa    #1
        beq     Audio2

        cmpa    #2
        beq     Audio3

        cmpa    #3
        beq     Audio4

        clr     AudioFlg

Audio1
        ldx     #AudioTable0
        jsr     set_audio               ;Make Power on Sound
        inc     AudioFlg

        bra     PlayAudio10

Audio2
        ldx     #AudioTable1
        jsr     set_audio               ;Make Power on Sound
        inc     AudioFlg
        bra     PlayAudio10

Audio3
        ldx     #AudioTable2
        jsr     set_audio               ;Make Power on Sound
        inc     AudioFlg
        bra     PlayAudio10

Audio4
        ldx     #AudioTable3
        jsr     set_audio               ;Make Power on Sound
        inc     AudioFlg
```

```
PlayAudio10
        movw    #Seconds/2,audio_delay      ;500ms pause

PlayAudioEx
        rts

AudioTable0
        dc.b    40,40,40
        dc.b    40,40,40
        dc.b    40,40,40
        dc.b    40,40,40

        dc.b    40,40,40
        dc.b    40,40,40
        dc.b    40,40,40
        dc.b    40,40,40

        dc.b    $ff,0,0

AudioTable1
        dc.b    60,60,60
        dc.b    60,60,60
        dc.b    60,60,60
        dc.b    60,60,60

        dc.b    60,60,60
        dc.b    60,60,60
        dc.b    60,60,60
        dc.b    60,60,60

        dc.b    $ff,0,0

AudioTable2
        dc.b    80,80,80
        dc.b    80,80,80
        dc.b    80,80,80
        dc.b    80,80,80

        dc.b    80,80,80
        dc.b    80,80,80
        dc.b    80,80,80
        dc.b    80,80,80

        dc.b    $ff,0,0

AudioTable3
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100
        dc.b    100,100,100

        dc.b    $ff,0,0

;----------------------------------------------------------------
BeepingDisabled
        movw  #00,audio_delay
        jsr     AudioOn

        ldx     #AudioDisabledMsg           ;Send Menu message
        jsr     OutStr0

        clr     AudioFlg
        rts

AudioDisabledMsg    dc.b    $A,$D,'Audio/Beeper disabled',0

;
;----------------------------------------------------------------------------------------
;
set_audio:
     sei
     stx    audio_ptr
     clr    audio_timer
     cli
     rts
;
;------------
; Check Audio
;------------
check_audio:
     ldaa   audio_timer          ;decrement audio duration timer
     beq    aud05                ;  unless already timed out (0)

     inca                        ;  OR infinite duration (FF)
     beq    audex

     dec    audio_timer
     bne    audex

aud05:
     ldx    audio_ptr            ;get pointer to audio table
     ldaa   0,x                  ;get next tone duration time & skip
     staa   audio_timer          ;  if not a repetition indicator
     bne    aud20                ;  (0 = repetition indicator)

     ldaa   1,x                  ;skip if end of repetition (2nd byte
     beq    aud10                ;  = 0)
```

```
        staa    rep_count               ;else, start of repetition:  store
        inx                       ;  number of repetitions & starting
        inx                       ;  address
        stx     rep_addr
        stx     audio_ptr
        bra     audex
;
aud10:
        ldaa    rep_count
        inca
        beq     aud15
        dec     rep_count               ;end of repetition: skip if repetition
        beq     aud50                   ;  counter has counted down
;
aud15:
        ldx     rep_addr                ;else, restore 'start of repetition'
        stx     audio_ptr               ;  address
        bra     audex
;
aud20:
        ldd     1,x                   ;set new audio tone period & duration
        std     audio_period            ;  audio is off (=0)
        beq     aud30

        sei                                       ;disable interrupt a bit
        addd    TC4
        std     TC4
        cli                               ;reenable interrupt again
        bra     aud40

aud30:
            bclr      PTP,AudioControl

aud40:
        inx                       ;increment and save audio table

aud50:
        inx                       ;  pointer
        inx
        stx     audio_ptr

audex:
        rts
;
;--------------------------------------------------------------------------
pwr_on_tone:
            dc.b    40,40,40
            dc.b    40,80,255
            dc.b    40,40,40

            dc.b    220,80,0
            dc.b    240,60,255

            dc.b    160,100,0
            dc.b    160,100,0
            dc.b    160,80,200

            dc.b    220,80,0
            dc.b    240,60,255

            dc.b    $ff,0,0

            dc.b    20,3,20                         ;middle number (3) denotes
            dc.b    20,3,$50                ; frequency from Output compare
            dc.b    20,3,20

            dc.b    $ff,0,0

;-----------------------------------------------------------------------------
; Audio Interrupt Routine
;-----------------------
audio_int:
            ldd     audio_period            ;audio time period for next
            addd    TC4                     ;  output compare interrupt
            std     TC4

            ldx     audio_period
            beq     auiex

            ldaa    audio_status
            eora    #1
            staa    audio_status
            beq     aui10

            bclr    PTP,AudioControl
            bra     auiex

aui10:
            bset    PTP,AudioControl

auiex:
            movb    #OC4flag,TFLG1                  ;clear flag
            rti


            END
```

```
;LEDBar_Switch.ASM
********************************************************************************
*REVISION HISTORY:
*
*DATE                  REV. NO.              DESCRIPTION
*
*June 10, 2006                1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 Demoboard app
********************************************************************************
;Compiled using CW
;
; ---------------------------
; Demoboard - LED Bar graph and Switches Routine
; ---------------------------

        include "mc9s12xdp512.inc"

;Public Function
        XDEF LEDSWInit
        XDEF SwitchToLEDInit
        XDEF LEDChaserInit
        XDEF CheckPushbuttons
        XDEF LightToLEDInit
        XDEF ThermistorToLEDInit
        XDEF PotentiometerToLEDInit

        XDEF PotPWMInit
        XDEF PotPWMDisabled


;Public Variables
        XDEF LEDSWstate
        XDEF PBDelay

        XDEF delay_timer

;Public Tones

;External Function
        XREF SerOutput0
        XREF OutStr0
        XREF state_timer
        XREF set_audio

;External Variables

        XREF ad0
        XREF ad1
        XREF ad2
        XREF ad3
```

```
DEFAULT_RAM:SECTION

LEDSWstate  ds      2
delay_timer ds      2
ChaserFlg   ds      1
ChaserPtr   ds      2
PBDelay             ds      2

SW2    equ     %00010000           ;Port ADL bit 4
SW3    equu    %00100000           ;Port ADL bit 5
SW4    equ     %01000000           ;Port ADL bit 6
SW5    equ     %10000000           ;Port ADL bit 7

Seconds             equ     3906                ;Of the RTI rate

;*******************************************************************************************
NON_BANKED:SECTION
;
LEDSWInit                                       ;Testing the Hardware ports
       movw    #LEDSWReady,LEDSWstate
       movb    #$FF,DDRH                        ;Bar LEDS, Make Port H = o/p
       movb    #$00,DDRT                        ;Switch BAr, Make sure Port T = i/p
       movw    #$00,delay_timer
       clr     ChaserFlg

       rts

LEDSWReady
       rts

;--------------------------------------------------------------------
LEDChaserInit
       movw    #$00,delay_timer
       movb    #0,ChaserFlg

       movw    #LEDChaser,LEDSWstate

       ldx     #LEDChaserMsg                    ;Send Menu message
       jsr     OutStr0
       rts

LEDChaserMsg        dc.b    $A,$D,'LED Chaser enabled',0

LEDChaser
       ldx     delay_timer
       bne     LEDChaserEx

       ldaa    ChaserFlg
       beq     Chaser1
```

```
        cmpa    #1
        beq     Chaser2

        cmpa    #2
        beq     Chaser3

        cmpa    #3
        beq     Chaser4

        clr     ChaserFlg

Chaser1
        ldx     #LEDTable0
        stx     ChaserPtr
        bra     LEDChaser10

Chaser2
        ldx     #LEDTable1
        stx     ChaserPtr
        bra     LEDChaser10

Chaser3
        ldx     #LEDTable2
        stx     ChaserPtr
        bra     LEDChaser10

Chaser4
        ldx     #LEDTable3
        stx     ChaserPtr

LEDChaser10
        movw    #LEDOn,LEDSWstate

LEDChaserEx
        rts


LEDOn
        ldx     delay_timer
        bne     LEDOnEx

        ldx     ChaserPtr
        ldaa    1,x+
        beq     NextTable

        staa    PTH                     ;Turn led on
        stx     ChaserPtr
        bra     LEDOn10


NextTable
```

```
        inc     ChaserFlg
        movw    #LEDChaser,LEDSWstate

LEDOn10
        movw    #Seconds/8,delay_timer

LEDOnEx
        rts

LEDTable0
        dc.b    %00000001
        dc.b    %00000010
        dc.b    %00000100
        dc.b    %00001000
        dc.b    %00010000
        dc.b    %00100000
        dc.b    %01000000
        dc.b    %10000000
        dc.b    %01000000
        dc.b    %00100000
        dc.b    %00010000
        dc.b    %00001000
        dc.b    %00000100
        dc.b    %00000010
        dc.b    %00000001

        dc.b    %00000000     ;Termination

LEDTable1
        dc.b    %00000001
        dc.b    %00000011
        dc.b    %00000111
        dc.b    %00001111
        dc.b    %00011111
        dc.b    %00111111
        dc.b    %01111111
        dc.b    %11111111
        dc.b    %01111111
        dc.b    %00111111
        dc.b    %00011111
        dc.b    %00001111
        dc.b    %00000111
        dc.b    %00000011
        dc.b    %00000001

        dc.b    %00000000     ;Termination


LEDTable2
        dc.b    %01111111
        dc.b    %00111111
        dc.b    %00011111
        dc.b    %00001111
        dc.b    %00000111
        dc.b    %00000011
        dc.b    %00000001
        dc.b    %00000001
        dc.b    %00000011
        dc.b    %00000111
        dc.b    %00001111
        dc.b    %00011111
        dc.b    %00111111
        dc.b    %01111111
        dc.b    %11111111


        dc.b    %00000000    ;Termination

LEDTable3
        dc.b    %11111110
        dc.b    %11111101
        dc.b    %11111011
        dc.b    %11110111
        dc.b    %11101111
        dc.b    %11011111
        dc.b    %10111111
        dc.b    %01111111
        dc.b    %10111111
        dc.b    %11011111
        dc.b    %11101111
        dc.b    %11110111
        dc.b    %11111011
        dc.b    %11111101
        dc.b    %11111110

        dc.b    %00000000    ;Termination


;-----------------------------------------------------------------
SwitchToLEDInit
        movw    #00,delay_timer
        movw    #SwitchToLED,LEDSWstate

        ldx     #SW2LEDMsg              ;Send Menu message
        jsr     OutStr0

        rts

SW2LEDMsg dc.b  $A,$D,'DIP switch to LED Bar enabled',0

SwitchToLED
        ldx     delay_timer
        bne     SwitchToLEDEx

        movw    #Seconds/8,delay_timer
        movb    PTT,PTH

SwitchToLEDEx
        rts

;------------------------------------------------------------------
LightToLEDInit
        movw    #LightToLED,LEDSWstate
        movw    #00,delay_timer

        ldx     #Light2LEDMsg                   ;Send Menu message
        jsr     OutStr0
        rts

Light2LEDMsg            dc.b    $A,$D,'Light sensor to LED Bar enabled',0


LightToLED
        ldx     delay_timer
        bne     LightToLEDEx

        ldd     ad3
        lsrd
        lsrd
        stab    PTH                     ;Turn LED on

        movw    #Seconds/8,delay_timer

LightToLEDEx
        rts

;------------------------------------------------------------------
ThermistorToLEDInit
        movw    #ThermistorToLED,LEDSWstate
        movw    #00,delay_timer

        ldx     #Thermistor2LEDMsg              ;Send Menu message
        jsr     OutStr0
        rts

Thermistor2LEDMsg       dc.b    $A,$D,'Thermistor sensor to LED Bar enabled',0


ThermistorToLED
        ldx     delay_timer
        bne     ThermistorToLEDEx

        ldd     ad1
        lsrd
        lsrd
        stab    PTH                     ;Turn LED on

        movw    #Seconds/8,delay_timer

ThermistorToLEDEx
        rts

;-----------------------------------------------------------------
PotentiometerToLEDInit
        movw    #PotentiometerToLED,LEDSWstate
        movw    #00,delay_timer

        ldx     #Potentiometer2LEDMsg                   ;Send Menu message
        jsr     OutStr0
        rts

Potentiometer2LEDMsg    dc.b    $A,$D,'Potentiometer to LED Bar enabled',0


PotentiometerToLED
        ldx     delay_timer
        bne     PotentiometerToLEDEx

        ldd     ad2
        lsrd
        lsrd
        stab    PTH                     ;Turn LED on

        movw    #Seconds/8,delay_timer

PotentiometerToLEDEx
        rts

;-----------------------------------------------------------------
PotPWMInit
        movb    #%00000000, PWME ;All channels disabled
        movb    #%00000000, PWMPOL      ;Low during duty cycle
        movb    #%00000000, PWMCLK      ;Clock SA & Clock SB
        movb    #%01110111, PWMPRCLK    ;Clock A = Bus Clock / 128,    Clock B = Bus
Clock / 128
        movb    #%00000000, PWMCAE      ;All channels operate in Left Aligned Output
Mode
        movb    #%00001100, PWMCTL      ;No concatenation
        movb    #%00000000, PWMSCLA     ;Clock SA = Clock A / ( 2 * 256)
        movb    #%00000000, PWMSCLB     ;Clock SB = Clock B / ( 2 * 256)

        movb    #%11111111, PWMPER3
        movb    #%11111111, PWMPER4
        movw  #PotPWM,LEDSWstate
        movw  #00,delay_timer

        ldx   #PotPWM2LEDMsg              ;Send Menu message
        jsr   OutStr0

        movb  #%00011000,PWME             ;Enable Port P bit 3 and 4 as PWM
        rts

PotPWM2LEDMsg   dc.b   $A,$D,'Potentiometer to PWM + LED bar enabled',0


PotPWM
        ldx   delay_timer
        bne   PotPWMEx

        ldd   ad2
        lsrd
        lsrd
        stab  PTH              ;Turn LED on

        stab  PWMDTY3
        stab  PWMDTY4

        movw  #Seconds/8,delay_timer

PotPWMEx
        rts

PotPWMDisabled
        movb  #%00000000, PWME ;All channels disabled
        movw  #LEDSWReady,LEDSWstate
        movw  #00,delay_timer

        clr   PTH              ;Turn LED off
        rts



        nop


;------------------------------------------------------------------
CheckPushbuttons
        ldx   PBDelay
        bne   CheckPushbuttonsEx

        brclr ATD0PTAD0,SW2,Switch2
        brclr ATD0PTAD0,SW3,Switch3
        brclr ATD0PTAD0,SW4,Switch4
        brclr ATD0PTAD0,SW5,Switch5
```

```
CheckPushbuttonsEx
        rts

Switch2
        ldx     #SW2tone
        jsr     set_audio
        movw    #Seconds/8,PBDelay
        rts

SW2tone
        dc.b    60,60,60
        dc.b    $ff,0,0


Switch3
        ldx     #SW3tone
        jsr     set_audio
        movw    #Seconds/8,PBDelay
        rts

SW3tone
        dc.b    80,80,80
        dc.b    $ff,0,0

Switch4
        ldx     #SW4tone
        jsr     set_audio
        movw    #Seconds/8,PBDelay
        rts

SW4tone
        dc.b    100,100,100
        dc.b    $ff,0,0

Switch5
        ldx     #SW5tone
        jsr     set_audio
        movw    #Seconds/8,PBDelay
        rts

SW5tone
        dc.b    120,120,120
        dc.b    $ff,0,0



        END
```

## S-record:

```
S224FD80001410CF40001642AB16409B16435516456B4D39804C3A40180B010034180B00007C
S224FD802035A7A7A7A74F3708FC4C3980180BD00127180B03012B180BE00127180B000128FA
S224FD8040180B000129180B00012A180B02012B180B00012C180B00012D180B00012E180B1C
S224FD806000012F180BF00127180B04012879200110EF1640DC1640A8FD20021540FD201DCC
S224FD80801540FD2022154016481E20E935CD0000200435CD03090436FD313D180340A72075
S224FD80A002180300002004 3DB62001273B792001B620007A200684DF8141272D8142272D67
S224FD80C08144272D814C272D8150272D8151272D8152272D8153272D8154272DCE410D166F
S224FD80E04337CE42A61643373D1645863D1643A23D16448B3D16469A3D16477B3D1648085B
S224FD81003D1647313D1646553D1646E33D0D0A0D0A2041646170743953313258445035 3172
S224FD812032203953313244454D483120434F4D4D414E44204D454E552056312E3030200DD7
S224FD81400A5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F5F92
S224FD81605F5F5F5F5F5F5F5F5F5F5F5F5F5F5F0D0A0D0A41203D3E204C4544204261722018
S224FD81804368617365720D0A42203D3E20456E61626C65204265657069 6E6720546F6E659C
S224FD81A00D0A44203D3E2044697361626C652042656570696E6720546F6E650D0A4C203DA3
S224FD81C03E20523133204C69676874205365 6E736F722056616C756520746F204C454420A2
S224FD81E06261720D0A50203D3E20506F74656E74696F6D6574657220746F2050756C73658B
S224FD82002057696474 68204D6F64756C6174696F6E202B204C45442042617 20D0A51203DC6
S224FD82203E2050756C73652057696474 68204D6F64756C6174696F6E204F46460D0A5220F5
S224FD82403D3E20506F74656E74696F6D657465722056616C756520746F204C45442062611A
S224FD8260720D0A53203D3E204449502053776974636820746F204C454420626172 0D0A541E
S224FD8280203D3E20546865726D6973746F7220525431205661 6C756520746F204C454420C4
S224FD82A06261720D0A003F000A0D00180B30003B4C37804C3880180BF002CD180B8002C23C
S224FD82C0180B4002C3180B6002C4180BB002C5180B0000C8180B1100C9180B0000CA180B99
S224FD82E02C00CB3D1F02C680FBFC02D07C2009FC02D27C200BFC02D47C200DFC02D67C2011
S224FD83000FFE20042704097E2004FE20072704097E2007FE20242704097E2024FE202927B0
S224FD832004097E2029FE201A2704097E201A1644C1180B8000370BA63027040703 20F83DE3
S224FD83404FCC80FC5ACF3D96CC180C00CF2000180B0120010B180301FF2014180BFF2011B7
S224FD8360180B02201C18030000201A7920191C025A804C46804C40104C4C10180B10004EC4
S224FD83801803438A201D7920213DFE201A2612CE45181644B6180307A1201A180343892010
S224FD83A01D3DCE441F7E201F180343D5201D18030000201ACE43BE1643377920213D0A0D41
S224FD83C0417564696F2F4265657065722 0656E61626C656400FE201A2644B62021270F81EC
S224FD83E00127168102271D810327247920 21CE441F1644B6722021201FCE443A1644B672EC
S224FD840020212014CE44551644B67220212009CE44701644B6722021180307A1201A3D28EB
S224FD84202828282828282828282828282828282828282828282828 28FF00003C3C3C3C3C3C3B
S224FD84403C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3CFF000050505050505073
S224FD846050505050505050505050505050FF00006464646464646464646464646464646464AB
S224FD84806464646464646464FF000018030000201A16438ACE449E1643377920213D0A0D35
S224FD84A0417564696F2F42656570657220646973 61626C65640014107E201279201110EF71
S224FD84C03DB62011270842274E7320112649FE2012A6007A20112624A601270D7A2018081D
S224FD84E0087E20167E2012202EB620184227057320 18271EFE20167E2012201BEC017C20CC
S224FD850014270A1410D3585C5810EF20041D0258800808087E20123D2828282850FF2828B3
S224FD852028DC5000F03CFFA06400A06400A050C8DC5000F03CFFFF000014031414035014FE
S224FD85400314FF0000FC2014D3585C58FE20142714B6201988017A201927061D0258802018
S224FD8560041C025880180B10004E0B180345852022180BFF0262180B000242180300002024
S224FD8580247920263D3D180300002024180B002026180345B32022CE459E1643373D0A0DCA
S224FD85A04C4544204368617365722 0656E61626C656400FE20242638B62026270F81012708
S224FD85C013810227178103271B792026CE46157E20272016CE46257E2027200ECE46357E29
S224FD85E020272006CE46457E2027180345F120223DFE2024261EFE2027A63027087A026072
S224FD86007E202720097220261803 45B32022180301E820243D010204081020408040201069
S224FD86200804020100010307 0F1F3F7FFF7F3F1F0F070301007F3F1F0F0703010103070F2B
S224FD86401F3F7FFF00FEFDFBF7EFDFBF7FBFDFEFF7FBFDFE00180300002024180346882061
S224FD866022CE46681643373D0A0D44495020737769746368 20746F204C45442042617220 9A
S224FD8680656E61626C656400FE2024260C180301E820241 80C024002603D180346CF2022DA
S224FD86A0180300002024CE46AD1643373D0A0D4C696768742053656E736F7220746F204C93
S224FD86C04544204261722065 6E61626C656400FE2024260EFC200F49497B0260180301E8DB
S224FD86E020243D1803471D2022180300002024CE46F61643373D0A0D546865726D69737499
S224FD87006F722073656E736F7220746F204C454420426172 20656E61626C656400FE202462
S224FD8720260EFC200B49497B0260180301E820243D180347672022180300002024CE474425
S224FD87401643373D0A0D506F74656E74696F6D6574657220746F204C454420426172206512
S224FD87606E61626C656400FE2024260EFC200D49497B0260180301E820243D180B000300D8
S224FD8780180B000301180B000302180B770303180B000304180B0C0305180B000308180B36
```

```
S224FD87A0000309180BFF0317180BFF0318180347EE2022180300002024CE47C516433718BF
S224FD87C00B1803003D0A0D506F74656E74696F6D6574657220746F2050574D202B204C459B
S224FD87E0442062617220656E61626C656400FE20242614FC200D49497B02607B031F7B03C4
S224FD880020180301E820243D180B0003001803458520221803000020247902603DA7FE2028
S224FD88202926141F02CF10101F02CF201E1F02CF402C1F02CF803A3DCE48451644B61803CC
S224FD884001E820293D3C3C3CFF0000CE48581644B6180301E820293D505050FF0000CE48E7
S224FD88606B1644B6180301E820293D646464FF0000CE487E1644B6180301E820293D7878A8
S208FD888078FF00007B
S224FFBF10400040004000400040004000400040004000400040004000400040004000400040000D
S224FFBF3040004000400040004000400040004000400040004000400040004000400040004000ED
S224FFBF5040004000400040004000400040004000400040004000400040004000400040004000CD
S224FFBF7040004000400040004000400040004000400040004000400040004000400040004000AD
S224FFBF90400040004000400040004000400040004000400040004000400040004000400040008D
S224FFBFB0400040004000400040004000400040004000400040004000400040004000400040006D
S224FFBFD040004000400043474000400040004000400040004000454540004000400040004000B9
S214FFBFF042E4400040004000400040004000400057
S9034000BC
```