How to Use DKKI with Adapt9S12X

Connect the DKKI and Adapt9S12X boards with H1 and P1 headers aligned as shown in the drawing. Make sure that Pin 1 of P1 is connected to Pin 1 of H1 and Pin 50 of P1 is connected to Pin 50 of H1, etc. This applies whether they are directly connected, or indirectly connected (e.g. via a backplane). Proper visual inspection of the connections is a must before powering up the boards.

Powering up the AD9S12X will also power up the DKKI board. Once powered up, two lines of dark blocks should appear on the LCD. Turn trimmer R1 on the DKKI clockwise or anticlockwise to adjust the contrast of the display.

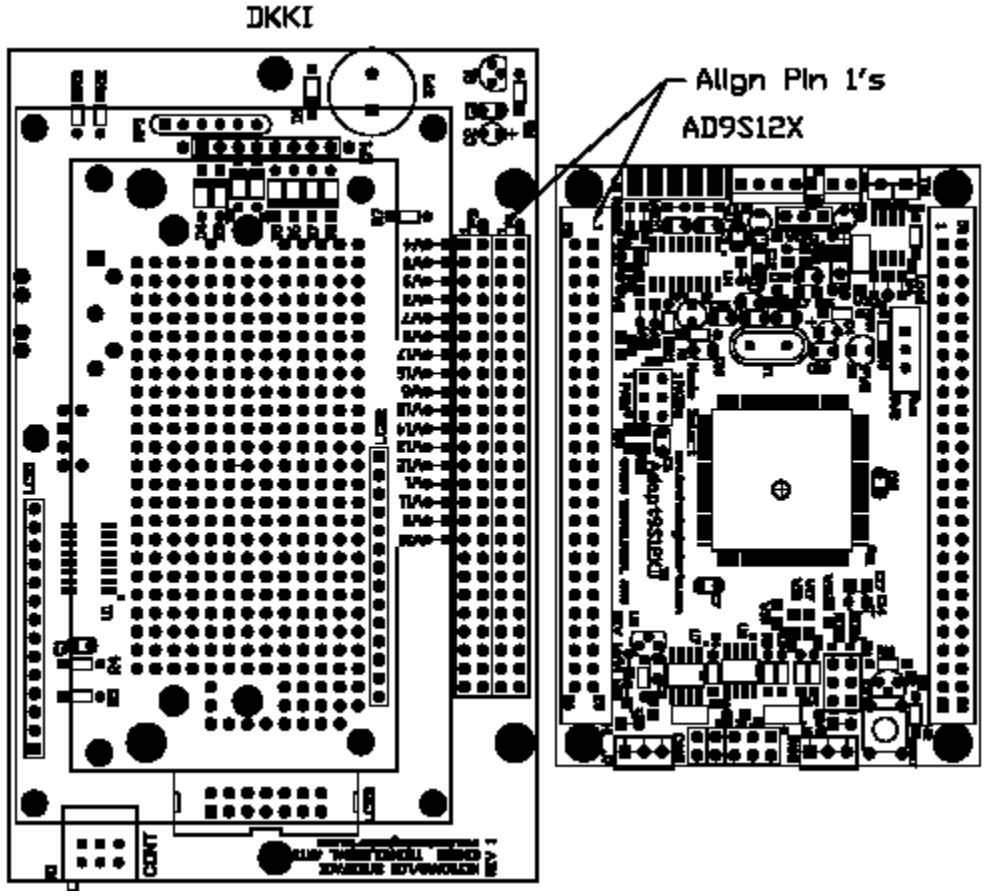
The DKKI demo is configured to drive the LCD in 8bit mode. The data to be sent to the LCD uses the SPI bus (as opposed to direct parallel connections) in order to minimize the quantity of MCU port pins required to send Data to the LCD. On board the DKKI is a shift register, 74HC595, that converts the serial data to parallel, driving the 8-bit bus of the LCD. The LCD EN (enable) and LCD RS (register select) signals are driven by 2 separate port pins.

The keyboard interface use 2 port pins, an input capture (keyboard clock) and the keyboard data line. The Keypad requires 8 port pins configured as 4X4 matrix.

Port pins usage:

DKKI - P1	**AD9S12X – H1**	
2 – 595DAT	2 – PS5/MOSI	used by LCD
3 – 595CLK	3 – PS6/SCK	used by LCD
4 – 595LAT	4 – PS7/SS*	used by LCD
6 – KBD_DAT	6 – PT7	used Keyboard as input data
7 – SPKR	7 – PT6	used to drive Speaker (optional)
10 – KBD_CLK	10 – PT3	use by Keyboard as Input capture
14 – DISP_EN	14 – PP7	used by LCD enable
15 – DISP_RS	15 – PP6	used by LCD register select
35 – KEYY3	35 – PH7	used by Keypad as o/p
36 – KEYY2	36 – PH6	used by Keypad as o/p
37 – KEYY1	37 – PH5	used by Keypad as o/p
38 – KEYY0	38 – PH4	used by Keypad as o/p
39 – KEYX3	39 – PH3	used by Keypad as keywake up i/p
40 – KEYX2	40 – PH2	used by Keypad as keywake up i/p
41 – KEYX1	41 – PH1	used by Keypad as keywake up i/p
42 – KEYX0	42 – PH0	used by Keypad as keywake up i/p
46 – KSTRB	46 – IRQ*	used by Keypad (optional)

Figure 1:



Code Explanations:

The assembly code is written in Codewarrior IDE. CW is configured for relocatable coding. Relocatable coding allows one to work on a single file which makes it easier for de-bugging. In the sources directory there are 5 separate files that makes up the DKKI Demo.

The compiled source code can be found below this document. Use an XGATE-capable BDM pod, such as MicroBDM12XG, to program the S-record into the MCU program memory.

Upon powerup or RESET of the board, the code will first initialize the RTI, SCI, Input capture and KeyWakeup interrupts. Next, the bus frequency is set to 24MHz. Finally, the priority of scheme for vectors to be handled by the MCU when interrupts occur is specified.

After hardware initialization, the interrupt is enabled and the LCD will display the following message:

Technological Arts
LCD Routine Program.
This is a TEST.....

The display is updated in the main loop whenever there is a keyboard event and will change the displayed message to:

Keyboard Test.....
;adslkfj;lkaj any character keyboard press.

Project files:

Main. Asm – This file includes calls to various subroutines to initialize the various hardware subsystems. Once they are initialized, the code then simply loops in indefinitely. The main loop is intermittently interrupted by SCI (Serial Communication Interface), RTI (Real Time Interrupts), Keyboard (input capture) and Keypad (KeyWakeup) interrupts

Int.asm – This file contains the RTI, SCI, and Keyboard interrupt service routines.

LCD.asm – This file contains routines for initializing and updating the LCD. Whenever a new character is to be displayed, routines in this file are called.

Keybrd.asm –This file consists of Keyboard display routine. When the input capture events happen, it is within this file that the keypress character is displayed.

Keypad.asm – This file consists of the Keypad interrupt routine. Whenever the KeyWakeup event happens, it is within this file that the event is processed and the character sent to the SCI

Linker.prm – This file consists of the Memory and Vector interrupt definitions.

```
;MAIN.ASM
*******************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.                DESCRIPTION
*
*April 27, 2006          1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 DKKI apps board
*******************************************************************************
;Compiled using CW
;
; --------------------------
; Demo - Main Routine
; --------------------------

        include "mc9s12xdp512.inc"

;Public Function
        XDEF ResetFunc
        XDEF Entry
        XDEF delay
        XDEF small_delay

;Public Variables
        XDEF COMMAND
        XDEF COMMAND_PENDING
        XDEF CONTINUOUS

        XDEF state_timer

;External Function
        XREF disp_init              ;Initialize LCD
        XREF lcd_state              ;Execute current LCD state
        XREF disp_update                    ;Update LCD

        XREF key_char_disp          ;Display keyboard press to Lcd

        XREF keybd_init                     ;Initialize Keyboard
        XREF RealTimeInit                   ;Initialize RTI

        XREF disp_delay
        XREF SerInputInt0
        XREF KeybrdInt

        XREF KeypadInit
        XREF RealTimeInt

;External Variables


*  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


MY_EXTENDED_RAM:SECTION

* System Variables

COMMAND             ds      1               ;used by ProcessCommand
COMMAND_PENDING ds      1               ;used by ProcessCommand
CONTINUOUS  ds      1               ;flag for real-time a/d screen update

state_timer     ds      2


***************************** 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     keybd_init              ;Initialize keyboard
        jsr     KeypadInit

; 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    #$C0,INT_CFADDR             ;Place PORTH -> PIEH into window
        movb    #%00000000,INT_CFDATA0      ;Set disabled
        movb    #%00000000,INT_CFDATA1      ;Set disabled
        movb    #%00000000,INT_CFDATA2      ;Set disabled
        movb    #%00000000,INT_CFDATA3      ;Set disabled
```

```
        movb    #%00000000,INT_CFDATA4    ;Set disabled
        movb    #%00000000,INT_CFDATA5    ;Set disabled
        movb    #%00000010,INT_CFDATA6    ;Set priority 2 PIEH enabled
        movb    #%00000000,INT_CFDATA7    ;Set disabled PIEJ

        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 OC6 to level 2 priority
        movb    #%00000000,INT_CFDATA2    ;Set Timer 5 disabled
        movb    #%00000000,INT_CFDATA3    ;Set Timer 4 disabled
        movb    #%00000010,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

        cli                               ;unmask interrupts
        jsr     disp_delay                ;Initialize Lcd


;-------------------------------------------------------------------------
main                                      ;Main Loop
        ldy     lcd_state                 ;execute current state
        jsr     0,y

        jsr     disp_update
        jsr     key_char_disp             ;check for keypress

        bra     main

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

small_delay
    pshy
    ldy   #777

dly
    dbne   y,dly
    puly
    rts


        END
```

```
;INT.ASM
*************************************************************************
*REVISION HISTORY:
*
*DATE                   REV. NO.              DESCRIPTION
*
*April 27, 2006         1.00                  Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 DKKI apps board
*************************************************************************
;Compiled using CW
;
; --------------------------
; Demo - ISR Routine
; --------------------------


        include "mc9s12xdp512.inc"


;Public Function
        XDEF RealTimeInit
        XDEF RealTimeInt

        XDEF OutStr0
        XDEF SerOutput0
        XDEF SerInputInt0

        XDEF KeybrdInt

;Public Variables
        XDEF wait_timer
        XDEF SecondsTimers

;External Variables
        XREF state_timer

        XREF lcd_timer
        XREF lcd_state
        XREF clr_flag
        XREF disp_buffer

        XREF keybd_timer
        XREF keyp_timer
        XREF key_ptr
        XREF keybd_value
        XREF keybd_flag
        XREF keybd_count

        XREF COMMAND_PENDING
        XREF COMMAND

        XREF LCDDelay

;External Messages
        XREF disp_menu0
```

```
MY_EXTENDED_RAM:SECTION

;
wait_timer        ds        2
SecondsTimers ds        4

;---------------------------------------------------------------------------
;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    #ms0064,RTICTL          ;and initialize RTI rate
;       movb    #ms4096,RTICTL          ;and initialize RTI rate
        bset    CRGFLG,RTIF             ;clear flag
        bset    CRGINT,RTIE             ;Enable RTI

;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

;test
        movb    #$FF,DDRA
        rts

;--------------------------------------------------------------------------
* Real-time Interrupt Routine

RealTimeInt

;--------
; Timers
;--------
;State timer is decremented here

time10
        com     PORTA
        ldx     state_timer
        beq     time20
        dex
        stx     state_timer

time20
        ldx     wait_timer
        beq     time30

        dex
        stx     wait_timer

time30
        ldx     lcd_timer
```

```
        beq     time40

        dex
        stx     lcd_timer

time40
        ldaa    keybd_timer
        beq     time50
        deca
        staa    keybd_timer
        bne     time50

        ldaa    #11
        staa    keybd_count

time50
        ldx     LCDDelay
        beq     time60

        dex
        stx     LCDDelay

time60
;       ldx     keyp_timer
;       beq     time70

;       dex
;       stx     keyp_timer

;       ldx     keyp_timer
;       bne     time70

;       ldx     #disp_menu0
;       stx     lcd_state

;       ldx     #disp_buffer+20
;       stx     key_ptr
;       clr     clr_flag

time70
        ldx     keyp_timer+0
        bne     time80

        ldx     keyp_timer+2
        beq     time100

        dex
        stx     keyp_timer+2

    ldx   #15625                ;1 Second delay
        stx     keyp_timer+0
        bra     time90

time80
        dex
        stx     keyp_timer+0
```

```
time90
        ldx     keyp_timer+0
        bne     time100

        ldx     keyp_timer+0
        bne     time100

        ldx     #disp_menu0
        stx     lcd_state

        ldx     #disp_buffer+20
        stx     key_ptr
        clr     clr_flag

time100

timex
        movb    #RTIF,CRGFLG
        rti


;*******************************************************************
*  SCI Input Interrupt Handler

*  Gets bytes from SCI.  Sets COMMAND_PENDING flag.


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

SerInputInt0
        ldaa    SCI0SR1                 ;read register to clear flag RDRF
        movb    SCI0DRL,COMMAND         ;read receive buffer
        movb    #TRUE,COMMAND_PENDING

SIIX
        rti

;-------------------------------------------------------------------------
KeybrdInt
      ldaa   #10
      staa   keybd_timer
```

```
        ldaa    PTT
        rola
        ldd     keybd_value
        rora
        rorb
        std     keybd_value

        bset    TFLG1,%00001000      ;Clear input capture 3 interrupt

        dec     keybd_count
        bne     keyex

        ldaa    #11
        staa    keybd_count

        inc     keybd_flag

        ldx     keybd_value+2
        stx     keybd_value+3
        ldaa    keybd_value
        rol     keybd_value+1
        rola
        rol     keybd_value+1
        rola
        staa    keybd_value+2

keyex
        rti


           END
```

```
;lcd.asm
*******************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.                DESCRIPTION
*
*April 27, 2006          1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 DKKI apps board
*******************************************************************************
;Compiled using CW

        include "mc9s12xdp512.inc"

;Public Function
        XDEF dstr
        XDEF disp_init
        XDEF MSG9
        XDEF MSG10
        XDEF drow
        XDEF disp_menu4
        XDEF disp_menu0
        XDEF disp_update
        XDEF disp_flag
        XDEF disp_delay

;External Function
        XREF small_delay
        XREF delay
        XREF SerOutput0

;Public Variables
        XDEF drow_var
        XDEF disp_buffer
        XDEF lcd_state
        XDEF lcd_timer
        XDEF LCDDelay

;External Variables
        XREF SecondsTimers

MY_EXTENDED_RAM:SECTION

; Memory Allocation are define in the main program
disp_buffer     ds      80              ;display buffer (4 lines of 20 chars)
disp_ptr        ds      2               ;pointer to display buffer
disp_flag       ds      1               ;1 = update; 0 = home; -1 = done
drow_var        ds      1
temp            ds      1
lcd_state       ds      2

lcd_timer       ds      2
dcol_var        ds      1
dig_lenght      ds      1
l_flag          ds      1
```

```
LCDDelay        ds        2

;-------------------------------------------------------------------------------
; Local Constants
lcd_ctrl        equ     %01000000       ;Port P bit 6 = 0 - Instruction in
lcd_data        equ     %01000000       ;Port P bit 6 = 0 - Data in
lcd_enable      equ     %10000000       ;Port P bit 7 - enable
BL              EQU     $20
EOT             EQU     $04             ;end of text/table

;
;Spi initialization variables
;
spi_mask1       equ       %01011100       ;SPE,MSTR=1, CPOL,CPHA=1:1
spi_mask2       equ     %11100000       ;Bit 7,6,5=1 the rest=0

spi_baud1       equ     %00000000       ;4.0 Mhz
spi_baud2       equ     %00000001       ;2.0 Mhz
spi_baud3       equ     %00000010       ;1.0 Mhz
spi_baud4       equ     %00000011       ;0.5 Mhz
spi_baud5       equ     %00000100       ;250 kHz
spi_baud6       equ     %00000101       ;125 kHz
spi_baud7       equ     %00000110       ;62.5 kHz
spi_baud8       equ     %00000111       ;31.3 kHz
SPIF            equ       %10000000       ;flag after the 8th clock
SPIE            equ       %10000000       ;Spi interrupt enable
SPRF            equ       %10000000       ;SPI receive Interrupt flag
SPTEF           equ       %00100000       ;SPI transmit empty flag
LSBF            equ       %00000001       ;SPI LSB First enable

SSPIN           equ     %10000000       ;Chip select for 74HC595

NON_BANKED:SECTION
;-------------------------------------------------------------------------------
; Display Initialization
; ----------------------

disp_delay
        bset    DDRP,%11000000              ;Initialize Port P

        bset    PTS,SSPIN                   ;inititalize port to 1
        movb    #spi_mask2,DDRS
        movb    #spi_mask1,SPI0CR1          ;SPE,MSTR=1, SWOM=0, CPOL,CPHA=1

;       movb    #%01011100,SPI0CR1          ;SPE,MSTR=1, SWOM=0, CPOL,CPHA=1
        movb    #spi_baud8,SPI0BR                   ;set clock to 1 Mhz


        bclr    SPI0CR2,%01                 ;SPC0=0, Normal mode

        ldaa    SPI0SR                      ;clear flag and buffer
        ldaa    SPI0DR

        clr     drow_var
        clr     disp_flag
```

```
        movw    #0,LCDDelay

        ldx     #500                    ;Set for 32ms delay
        stx     lcd_timer

        ldx     #disp_init
        stx     lcd_state

init_ex
        rts

;
DispDelay
        ldx     LCDDelay
        bne     DispDelay
        rts

;-------------------------------------------------------------------------
disp_init                               ;Initialize LCD display
        ldx     lcd_timer
        bne     init_ex

;Initialize
        ldaa    #%00110000
        jsr     toggle_lcd_ctrl     ;8 data bits, 2 display lines, 0

        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay

        ldaa    #%00111000
        jsr     toggle_lcd_ctrl     ;8 data bits, 2 display lines, 0

        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay

        ldaa    #%00001100
        jsr     toggle_lcd_ctrl     ;display on, cursor off, blink off

        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay

        ldaa    #%00000001
        jsr     toggle_lcd_ctrl     ;Clear display and return cursor to home

        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay

        ldaa    #%01000000
        jsr     toggle_lcd_ctrl

        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay

        ldaa    #%10000000
        jsr     toggle_lcd_ctrl
```

```
        ldx     #cgram_table
        ldd     #CGRAM_LEN
;       jsr     line_update

        ldx     #disp_menu1
        stx     lcd_state

        jsr     disp_clear

        ldx     #500                    ;Set for 32ms delay
        stx     lcd_timer

        rts

toggle_lcd_ctrl
        jsr     dump_spi_data

        bclr    PTS,SSPIN               ;DRIVE SELECT OF 74HC595 LO
        nop                             ;Low to Hi latches data from shift register
        nop
        bset    PTS,SSPIN               ;DRIVE SELECT OF 74HC595 HI

        bclr    PTP,lcd_ctrl
        nop
        nop
        nop
        nop
        bset    PTP,lcd_enable          ;Pulse lcd enable
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        bclr    PTP,lcd_enable
        rts


toggle_lcd_data
        jsr     dump_spi_data

        bclr    PTS,SSPIN       ;DRIVE SELECT OF 74HC595 LO
        nop                             ;Low to Hi latches data from shift register
        nop
        bset    PTS,SSPIN       ;DRIVE SELECT OF 74HC595 HI

        bset    PTP,lcd_data
        nop
        nop
        nop
```

```
        nop
        bset    PTP,lcd_enable    ;Pulse lcd enable
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        bclr    PTP,lcd_enable
        rts

dump_spi_data
        pshb

        ldab    SPI0SR          ;clear flag
        ldab    SPI0DR                  ;clear buffer

        staa    SPI0DR                  ;dump to spi to start playback
        brclr   SPI0SR,SPIF,*   ;wait here for spi int

        pulb
        rts

;=============================================================================
;These subroutine displays the various messages in 2 second interval and are
; called in the main loop.
;

disp_menu0                  ;dummy display
      ldx    lcd_timer
      bne    lcd_ex0

      ldaa   #0
      staa   drow_var
      jsr    drow
      ldx    #MSG8
      jsr    dstr

      ldaa   #1
      staa   drow_var
      jsr    drow
      ldx    #MSG1
      jsr    dstr

      ldaa   #2
      staa   drow_var
      jsr    drow
      ldx    #MSG2
      jsr    dstr
```

```
        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG3
        jsr     dstr

        ldx     #disp_menu1
        stx     lcd_state

        ldx     #15625              ;1 Second delay
        stx     lcd_timer

lcd_ex0
        rts


disp_menu1
        ldx     lcd_timer
        bne     lcd_ex1

        ldaa    #0
        staa    drow_var
        jsr     drow
        ldx     #MSG1
        jsr     dstr

        ldaa    #1
        staa    drow_var
        jsr     drow
        ldx     #MSG2
        jsr     dstr

        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG3
        jsr     dstr

        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldx     #disp_menu2
        stx     lcd_state

        ldx     #15625              ;1 Second delay
        stx     lcd_timer

lcd_ex1
        rts

disp_menu2
        ldx     lcd_timer
        bne     lcd_ex2
```

```
        ldaa    #0
        staa    drow_var
        jsr     drow
        ldx     #MSG2
        jsr     dstr

        ldaa    #1
        staa    drow_var
        jsr     drow
        ldx     #MSG3
        jsr     dstr

        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldx     #disp_menu3
        stx     lcd_state

        ldx     #15625                  ;1 Second delay
        stx     lcd_timer

lcd_ex2
        rts

disp_menu3
        ldx     lcd_timer
        bne     lcd_ex3

        ldaa    #0
        staa    drow_var
        jsr     drow
        ldx     #MSG3
        jsr     dstr

        ldaa    #1
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr
```

```
        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldx     #disp_menu0
        stx     lcd_state

        ldx     #15625                  ;1 Second delay
        stx     lcd_timer

lcd_ex3
        rts

disp_menu4

        rts

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

; Clear Display
; -------------

disp_clear
        ldx     #disp_buffer            ;starting address of display buffer
        stx     disp_ptr        ;home cursor
        ldab    #80                     ;number of characters
        ldaa    #BL

dcl10
        staa    0,x                     ;fill buffer with spaces
        inx
        decb
        bne     dcl10

        ldaa    #1                      ;set flag for immediate display
        staa    disp_flag               ;  update
        rts

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

; Set Display Position
; --------------------
; This routine sets the display buffer pointer to the selected row
; and column position.  Values are assumed to be valid:  no range checking
; is performed.
;       Entry:  A contains row number (0 - 3)
;               B contains column number (0 - 19)

disp_pos
        ldx     #disp_buffer            ;starting address of display buffer
        abx                             ;add column number
        ldab    #20
        mul
        abx                     ;add 20 x row number
```

```
        stx     disp_ptr
        rts
;-------------------------------------------------------------------------

; Display Character
; -----------------
;       Entry:  A contains character to be displayed
;               B contains number of occurences (DISP_REP only)

disp_chr
        ldy     disp_ptr        ;store character in display buffer
        cmpa    #$a6
        bne     dsc10
        ldaa    #4

dsc10
        staa    0,y
        iny
        bra     dispex

disp_rep
        ldy     disp_ptr
        cmpa    #$a6
        bne     drp10
        ldaa    #4

drp10
        staa    0,y                     ;store character, decrement counter
        iny                             ;  and loop until done
        decb
        bne     drp10

dispex
        cpy     #disp_buffer+80
        blo     dsp10

        ldy     #disp_buffer

dsp10
        sty     disp_ptr

        ldaa    #1                      ;set flag for immediate display
        staa    disp_flag               ;  update
        rts
;-------------------------------------------------------------------------

; Display String
; --------------

disp_str
        ldy     disp_ptr

dst10
        ldaa    0,x
        beq     dst20
```

```
        cmpa    #$a6
        bne     dst15

        ldaa    #4

dst15
        staa    0,y
        inx
        iny
        bra     dst10

dst20
        inx
        bra     dispex

;--------
disp_strn
        ldy     disp_ptr

dsn10
        ldaa    0,x
        cmpa    #$a6
        bne     dsn15
        ldaa    #4

dsn15
        staa    0,y
        inx
        iny
        decb
        bne     dsn10

        bra     dispex
;-------------------------------------------------------------------------------

; Update Display
; --------------
;       This routine updates the LCD display from the display image in
; memory whenever the update flag is set.

disp_update
        ldaa    disp_flag           ;exit if display update flag is clr
        bmi     dupex               ;
        dec     disp_flag
        beq     dup10

        ldaa    #%00000010
        jsr     toggle_lcd_ctrl     ;home cursor

        bra     dupex

dup10
        ldab    #20                             ;number of characters per line

        ldx     #disp_buffer                    ;display 1st line
        jsr     line_update
```

```
        ldx     #disp_buffer+40         ;display 3rd line
        jsr     line_update

        ldx     #disp_buffer+20         ;display 2nd line
        jsr     line_update

        ldx     #disp_buffer+60         ;display 4th line
        jsr     line_update

        ldaa    #%1100
        jsr     toggle_lcd_ctrl         ;display on, cursor off, blink off

dupex
        rts

;-------------------------------------------------------------------------
line_update
        pshb                            ;save character count

lup10   ldaa    0,x                     ;write character to display
        cmpa    #'0'                    ;  (substitue O for 0)
        bne     lup15
        ldaa    #'O'

lup15                   ;Display Char to LCD
        jsr     toggle_lcd_data

        pshx
        movw    #15,LCDDelay            ;1ms delay
        jsr     DispDelay
        pulx

        inx                             ;advance to next character, decrement
        decb                            ;  count & loop if not done
        bne     lup10
        pulb                            ;restore character count

        rts


;=========================================================================
; Set Cursor Position
; -------------------
;       A contains row number
;       B contains column number

drow
        ldaa    drow_var                ;drow_var = 0 =>1st line
;                       ;drow_var = 1 =>2nd line
;                       ;drow_var = 2 =>3rd line
;                       ;drow_var = 3 =>4th line
        ldab    #0
        jsr     disp_pos
        rts
```

```
dpos
        ldaa    drow_var                ;drow_var = 0 =>1st line
;                               ;drow_var = 1 =>2nd line
;                               ;drow_var = 2 =>3rd line
;                               ;drow_var = 3 =>4th line

        ldab    dcol_var                ;dcol_var = 0 => 20 1st line
;                               ;dcol_var = 0 => 20 2nd line
;                               ;dcol_var = 0 => 20 3nd line
;                               ;dcol_var = 0 => 20 4nd line

        jsr     disp_pos
        rts

; Display String
; --------------
;       X contains address of string to be displayed
;       B contains length of string (if absent:  nul terminates string)

dstr
      jsr   disp_str
      rts

dstrn
      jsr   disp_strn
      rts

dvalx
      ldab  #5            ;will display 5 digit value
;      subb   #3            ;lenght of value
      subb   dig_lenght     ;lenght of value to be displayed
      abx                 ;Reg X contain the address to be displayed
      jsr   disp_strn
      rts

;-------------------------------------------------------------------------------
cgram_table
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000
      dc.b    %00000


CGRAM_LEN     EQU   $-cgram_table

;-------------------------------------------------------------------------------
;Lcd Messages

MSG1          dc.b    'LCD Routine Program.'
            dc.b    0
MSG2          dc.b    'This is a TEST.....'
            dc.b    0
```

```
MSG3        dc.b    '--------------------'
            dc.b    0
MSG4        dc.b    'Msg4.....'
            dc.b    0
MSG5        dc.b    'Msg5.....'
            dc.b    0
MSG6        dc.b    'Msg6.....'
            dc.b    0
MSG7        dc.b    'Msg7.....'
            dc.b    0
MSG8        dc.b    'Technological Arts  '
            dc.b    0
MSG9        dc.b    '                    '
            dc.b    0
MSG10       dc.b    'Keyboard Test.......'
            dc.b    0
MSG11       dc.b    'Msg11.....'

            END
```

```
;Keybrd.asm
*******************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.                DESCRIPTION
*
*April 27, 2006          1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 DKKI apps board
*******************************************************************************
;Compiled using CW

        include "mc9s12xdp512.inc"

;
;Public Function
        XDEF keybd_init
        XDEF get_key
        XDEF keybd_count
        XDEF key_char_disp

;External Function
        XREF MSG9
        XREF MSG10
        XREF disp_menu4
        XREF dstr


;Public Variable
        XDEF keybd_state
        XDEF keyp_timer
        XDEF keybd_timer
        XDEF keybd_value
        XDEF clr_flag
        XDEF key_ptr
        XDEF keybd_flag

;External Variable
        XREF drow_var
        XREF drow
        XREF lcd_state
        XREF disp_flag
        XREF disp_buffer

        XREF SecondsTimers


;Keyboard
LEFT    equ     1
RIGHT   equ     2
UP      equ     3
DOWN    equ     4

DEL     equ     7
BS      equ     8
CR      equ     $13
```

```
ESC     equ     $1b
ALT     equ     $11
CTRL    equ     $14
F1      equ     $05

ENTER equ     CR
EXIT    equ     ESC
TEN     equ     $80


MY_EXTENDED_RAM:SECTION

;Keyboard variables
keybd_char      ds.b    1
keybd_value     ds.b    6
keybd_count     ds.b    1
keybd_flag      ds.b    1
keybd_timer     ds.b    1
release_flag    ds.b    1
keybd_state     ds.b    1
key_ptr         ds.b    2
keyp_timer      ds.b    4
keyp_flag       ds.b    1
clr_flag ds.b   1

;=============================================================================
; Keyboard Routines
; -----------------

NON_BANKED:SECTION

keybd_init
        bset    TSCR1,TEN               ;Enable timer
        bset    TIE,%1000               ;Enable input capture 3
        bset    TFLG1,%1000             ;Clear input capture 3 interrupt

        bset    TCTL4,%10000000                 ;Capture on Falling edge
        bclr    TCTL4,%01000000                 ;Capture on Falling edge

        ldaa    #11
        staa    keybd_count

        ldx     #0
        stx     keybd_value
        stx     keyp_timer

clear_key
      clr   keybd_flag
      clr   release_flag
      clr   keybd_char
      clr   keybd_state
      clr   keyp_flag
      clr   keybd_timer
      clr   clr_flag

      ldx    #disp_buffer+20
```

```
        stx     key_ptr

        rts

;
get_key
        ldaa    keybd_flag
        lbeq     gky90

        ldx     #keybd_value+2-1

        sei
        ldab    keybd_flag
        abx
        ldab    0,x
        dec     keybd_flag
        cli

        cmpb    #$e0                ;skip if alternate? prefix
        beq     gky90

        cmpb    #$f0                ;skip if release prefix
        bne     gky10

        inc     release_flag
        bra     gky90

gky10
        ldaa    release_flag
        beq     gky20

        clr     release_flag
        cmpb    #ALT
        bne     gky15

        bclr    keybd_state,%10
        bra     gky90

gky15
        cmpb    #CTRL
        bne     gky90
        bclr    keybd_state,%01
        bra     gky90

gky20
        tstb
        bmi     gky90

gky30
        ldx     #key_table_at
        abx
        ldaa    0,x
        staa    keybd_char

        ldaa    #1
        staa    keyp_flag
```

```
        ldaa    clr_flag
        bne     gky90

        ldaa    #0
        staa    drow_var
        jsr     drow
        ldx     #MSG10
        jsr     dstr
        inc     clr_flag

        ldaa    #1
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

gky90   clra

gkyex
        rts

key_char_disp
        jsr     get_key         ;get keyboard value

        ldaa    keyp_flag       ;display value to lcd
        beq     disp_ex

        ldaa    keybd_char

        ldy     key_ptr
        cpy     #disp_buffer+40
        beq     line1

        cpy     #disp_buffer+80
        beq     line0
        bra     line2

line0
        psha
        ldaa    #1
        staa    drow_var
        jsr     drow
        ldx     #MSG9
```

```
        jsr     dstr

        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        pula

        ldy     #disp_buffer+20
        sty     key_ptr
        bra     line2

line1
        psha
        ldaa    #2
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr

        ldaa    #3
        staa    drow_var
        jsr     drow
        ldx     #MSG9
        jsr     dstr
        pula

        ldy     #disp_buffer+40
        sty     key_ptr

line2
        staa    0,y
        iny
        sty     key_ptr

        ldaa    #1
        staa    disp_flag
        clr     keybd_char

           ldx      #disp_menu4
           stx      lcd_state

;       ldx     #61*30          ;30 second time out if no key press
;       stx     keyp_timer

        ldx     #29             ;29 + 1 = 30 Second delay
           stx      keyp_timer+2

        ldx     #15625                  ;1 Second delay
           stx      keyp_timer+0

           clr      keyp_flag

disp_ex
```

```
        rts


;ConstSec:    SECTION

key_table_at
            dc.b    0,0,0,0,0,F1,0,0,0,0,0,0,0,0,0,0

            dc.b    0,ALT,0,0,CTRL
            dc.b    'Q1'
            dc.b    0,0,0
            dc.b    'ZSAW2'
            dc.b    0

            dc.b    0
            dc.b    'CXDE43'
            dc.b    0,0
            dc.b    ' VFTR5'
            dc.b    0

            dc.b    0
            dc.b    'NBHGY6'
            dc.b    0,0,0
            dc.b    'MJU78'
            dc.b    0

            dc.b    0
            dc.b    ',KIO09'
            dc.b    0,0
            dc.b    './L'
            dc.b    $38
            dc.b    'P-'
            dc.b    0

            dc.b    0,0,0,0
            dc.b    '[='
            dc.b    0,0,0,0,CR
            dc.b    ']'
            dc.b    0,0,0,0

            dc.b    0,0,0,0,0,0,BS,0,0,0,0,0,0,0,0,0
            dc.b    0,0,0,0,0,0,ESC,0,0,0,0,0,0,0,0,0


            END
```

```
;Keypad.ASM
*******************************************************************************
*REVISION HISTORY:
*
*DATE                    REV. NO.                DESCRIPTION
*
*April 27, 2006          1.00                    Initial release
*
*Author:Exequiel Rarama for the ADAPT9S12XDP512 DKKI apps board
*******************************************************************************
;Compiled using CW
;
; --------------------------
; DKKI - Keypad Routine
; --------------------------

        include "mc9s12xdp512.inc"

;Public Function

        XDEF KeypadInit
        XDEF KeypadPress            ;Check if Keypad is press
        XDEF KeypadInt                      ;Keypad interrupt

;Public Variables

        XDEF keypad_timer
        XDEF keypad_var
        XDEF keypad_flag            ;This flag is set for valid keypad press

;External Function

        XREF OutStr0
        XREF SerOutput0

;       XREF Dump_byte
;       XREF set_audio              ;make Sound
;       XREF CheckKey

MY_EXTENDED_RAM:SECTION

*keypad variables
keypad_flag     ds      1
keypad_var      ds      1
keypad_temp     ds      1
keypad_counter  ds      1
keypad_timer    ds      2


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

NON_BANKED:SECTION

KeypadInit
        ldaa    #$FF
        staa    PERH                ;Enable Port H pullup
```

```
        movb    #$00,PIEH               ;Disable Key Wake up on Port H

    ldaa   #$F0
        staa    PTH                     ;Disable keypad int. until ready
        staa    DDRH                    ;Initialize Port P as Keypad input.

    clr    keypad_flag

    ldx    #300                 ;Enable keypad int when this timer is clear
    stx    keypad_timer
    rts

KeypadPress
        ldaa    keypad_flag
        beq     check_p10

        ldaa    keypad_var
;       psha

;       jsr     CheckKey

*do what you need to do here

;       pula
;       jsr     SerOutput0              ;Data is in Reg A, display keypress to sci
;       clr     keypad_flag             ;Keypad interrupt is enabled in Real Time Int.

;       ldx     #KeyPresstone
;       jsr     set_audio               ;make Sound

check_p10
    rts

KeyPresstone
        dc.b    100,50,0
        dc.b    $ff,0,0

;---------------------------------------------------------------------------------
KeypadInt
        movb    #$00,PIEP               ;Disable Key Wake up on Port P
        movb    #$0F,PIFP               ;Clear Port P Key Wake up

    jsr    KeyInitInt

next_keypad10
    dec    keypad_counter
    beq    next_key10

    jsr    check_next_pad
    brset  PTH,%0001,next_keypad10     ;Check next pad

    ldx    #keypad_table0        ;Get keypad value
    jsr    get_keypad_val        ;Reg b has keypad value
    stab   keypad_var

;test
```

```
;           ldaa    keypad_var
;           jsr     SerOutput0              ;Data is in Reg A, display keypress to sci
;           ldaa    PTH
;           jsr     Dump_byte

        ldaa    #1
        staa    keypad_flag         ;set flag for valid keypress
        ldaa    #$FF                ;Disable keypad until present press is service
        staa    PTH

        ldx     #300                ;
        stx     keypad_timer        ;Delay before next keypad interrupt

        lbra        kp_exit10

next_key10
        jsr     KeyInitInt

next_keypad20
        dec     keypad_counter
        beq     next_key20

        jsr     check_next_pad
        brset   PTH,%0010,next_keypad20     ;Check next pad

        ldx     #keypad_table1      ;Get keypad value
        jsr     get_keypad_val      ;Reg b has keypad value
        stab    keypad_var

        ldaa    #1
        staa    keypad_flag         ;set flag for valid keypress
        ldaa    #$FF                ;Disable keypad until present press is service
        staa    PTH

        ldx#300                 ;
        stxkeypad_timer         ;Delay before next keypad interrupt

        bra     kp_exit10

next_key20
        jsr     KeyInitInt

next_keypad30
        dec     keypad_counter
        beq     next_key30

        jsr     check_next_pad
        brset   PTH,%0100,next_keypad30     ;Check next pad

        ldx     #keypad_table2      ;Get keypad value
        jsr     get_keypad_val      ;Reg b has keypad value
        stab    keypad_var

        ldaa    #1
        staa    keypad_flag         ;set flag for valid keypress
        ldaa    #$FF                ;Disable keypad until present press is service
```

```
        staa    PTH

        ldx#300                 ;
        stxkeypad_timer         ;Delay before next keypad interrupt

        bra     kp_exit10

next_key30
        jsr     KeyInitInt

next_keypad40
        dec     keypad_counter
        beq     kp_exit

        jsr     check_next_pad
        brset   PTH,%1000,next_keypad40     ;Check next pad

        ldx     #keypad_table3      ;Get keypad value
        jsr     get_keypad_val      ;Reg b has keypad value
        stab    keypad_var

        ldaa    #1
        staa    keypad_flag         ;set flag for valid keypress
        ldaa    #$FF                ;Disable keypad until present press is service
        staa    PTH

        ldx#300                 ;Delay before next keypad interrupt
        stxkeypad_timer

        bra     kp_exit10

kp_exit
        clr     keypad_flag

kp_exit10
          ldx       #170
          stx       keypad_timer      ;Delay before next keypad interrupt

        rti

KeyInitInt
        ldab    #%11110111
        stab    keypad_temp
        ldaa    #5
        staa    keypad_counter
        rts

check_next_pad
        ldab    keypad_temp
        lslb
        stab    PTH
        stab    keypad_temp
        rts

get_keypad_val
        ldab    keypad_temp
```

```
        lsrb
        lsrb
        lsrb
        lsrb
        comb
        andb    #$0f            ;clear high nibbles

        abx
        ldab    0,x
        rts

keypad_table0
        dc.b    '?'
        dc.b    '1'     ;ok
        dc.b    '2'
        dc.b    '?'
        dc.b    '3'
        dc.b    '?'
        dc.b    '?'
        dc.b    '?'

keypad_table1
        dc.b    'A'     ;ok
        dc.b    '4'     ;ok
        dc.b    '5'
        dc.b    '?'
        dc.b    '6'
        dc.b    '?'
        dc.b    '?'
        dc.b    '?'

keypad_table2
        dc.b    'B'     ;ok
        dc.b    '7'     ;ok
        dc.b    '8'
        dc.b    '*'
        dc.b    '9'
        dc.b    '?'
        dc.b    '?'
        dc.b    '?'

keypad_table3
        dc.b    'C'     ;ok
        dc.b    '*'     ;ok
        dc.b    '0'
        dc.b    '?'
        dc.b    '#'
        dc.b    '?'
        dc.b    '?'
        dc.b    '?'
        dc.b    'D'


            END
```

```
/* This is a linker parameter file for the MC9S12XDP512 */

/*
This file is setup to use the HCS12X core only.
If you plan to also use the XGATE in your project, best create a new project with the
'New Project Wizard' (File|New... menu in the Codewarrior IDE) and choose the appropriate
project parameters.
*/

NAMES
   /* CodeWarrior will pass all the needed files to the linker by command line. But here you may
add your additional files */
END

SEGMENTS /* here all RAM/ROM areas of the device are listed. Used in PLACEMENT below. All
addresses are 'logical' */

/* Register space  */
/*    IO_SEG       = PAGED         0x0000 TO   0x07FF; intentionally not defined */

/* non-paged EEPROM */
     EEPROM       = READ_ONLY     0x0C00 TO   0x0FFF;

/* non-paged RAM */
     RAM          = READ_WRITE    0x2000 TO   0x3FFF;

/* non-banked FLASH */
     ROM_4000     = READ_ONLY     0x4000 TO   0x7FFF;
     ROM_C000     = READ_ONLY     0xC000 TO   0xFEFF;
/*   VECTORS      = READ_ONLY     0xFF00 TO   0xFFFF; intentionally not defined: used for
VECTOR commands below */
   //OSVECTORS    = READ_ONLY     0xFF10 TO   0xFFFF;  /* OSEK interrupt vectors (use
your vector.o) */

/* paged EEPROM                  0x0800 TO   0x0BFF; addressed through EPAGE */
     EEPROM_FC    = READ_ONLY   0xFC0800 TO 0xFC0BFF;
     EEPROM_FD    = READ_ONLY   0xFD0800 TO 0xFD0BFF;
     EEPROM_FE    = READ_ONLY   0xFE0800 TO 0xFE0BFF;
/*   EEPROM_FF    = READ_ONLY   0xFF0800 TO 0xFF0BFF; intentionally not defined:
equivalent to EEPROM */

/* paged RAM:                    0x1000 TO   0x1FFF; addressed through RPAGE */
     RAM_F8       = READ_WRITE  0xF81000 TO 0xF81FFF;
     RAM_F9       = READ_WRITE  0xF91000 TO 0xF91FFF;
     RAM_FA       = READ_WRITE  0xFA1000 TO 0xFA1FFF;
     RAM_FB       = READ_WRITE  0xFB1000 TO 0xFB1FFF;
     RAM_FC       = READ_WRITE  0xFC1000 TO 0xFC1FFF;
     RAM_FD       = READ_WRITE  0xFD1000 TO 0xFD1FFF;
/*   RAM_FE       = READ_WRITE  0xFE1000 TO 0xFE1FFF; intentionally not defined:
equivalent to RAM: 0x2000..0x2FFF */
/*   RAM_FF       = READ_WRITE  0xFF1000 TO 0xFF1FFF; intentionally not defined:
equivalent to RAM: 0x3000..0x3FFF */

/* paged FLASH:                  0x8000 TO   0xBFFF; addressed through PPAGE */
     PAGE_E0      = READ_ONLY   0xE08000 TO 0xE0BFFF;
     PAGE_E1      = READ_ONLY   0xE18000 TO 0xE1BFFF;
```

```
    PAGE_E2      = READ_ONLY   0xE28000 TO 0xE2BFFF;
    PAGE_E3      = READ_ONLY   0xE38000 TO 0xE3BFFF;
    PAGE_E4      = READ_ONLY   0xE48000 TO 0xE4BFFF;
    PAGE_E5      = READ_ONLY   0xE58000 TO 0xE5BFFF;
    PAGE_E6      = READ_ONLY   0xE68000 TO 0xE6BFFF;
    PAGE_E7      = READ_ONLY   0xE78000 TO 0xE7BFFF;

    PAGE_E8      = READ_ONLY   0xE88000 TO 0xE8BFFF;
    PAGE_E9      = READ_ONLY   0xE98000 TO 0xE9BFFF;
    PAGE_EA      = READ_ONLY   0xEA8000 TO 0xEABFFF;
    PAGE_EB      = READ_ONLY   0xEB8000 TO 0xEBBFFF;
    PAGE_EC      = READ_ONLY   0xEC8000 TO 0xECBFFF;
    PAGE_ED      = READ_ONLY   0xED8000 TO 0xEDBFFF;
    PAGE_EE      = READ_ONLY   0xEE8000 TO 0xEEBFFF;
    PAGE_EF      = READ_ONLY   0xEF8000 TO 0xEFBFFF;

    PAGE_F0      = READ_ONLY   0xF08000 TO 0xF0BFFF;
    PAGE_F1      = READ_ONLY   0xF18000 TO 0xF1BFFF;
    PAGE_F2      = READ_ONLY   0xF28000 TO 0xF2BFFF;
    PAGE_F3      = READ_ONLY   0xF38000 TO 0xF3BFFF;
    PAGE_F4      = READ_ONLY   0xF48000 TO 0xF4BFFF;
    PAGE_F5      = READ_ONLY   0xF58000 TO 0xF5BFFF;
    PAGE_F6      = READ_ONLY   0xF68000 TO 0xF6BFFF;
    PAGE_F7      = READ_ONLY   0xF78000 TO 0xF7BFFF;

    PAGE_F8      = READ_ONLY   0xF88000 TO 0xF8BFFF;
    PAGE_F9      = READ_ONLY   0xF98000 TO 0xF9BFFF;
    PAGE_FA      = READ_ONLY   0xFA8000 TO 0xFABFFF;
    PAGE_FB      = READ_ONLY   0xFB8000 TO 0xFBBFFF;
    PAGE_FC      = READ_ONLY   0xFC8000 TO 0xFCBFFF;
/*  PAGE_FD      = READ_ONLY   0xFD8000 TO 0xFDBFFF; intentionally not defined:
equivalent to ROM_4000 */
    PAGE_FE      = READ_ONLY   0xFE8000 TO 0xFEBFFF;
/*  PAGE_FF      = READ_ONLY   0xFF8000 TO 0xFFBFFF; intentionally not defined:
equivalent to ROM_C000 */

END

PLACEMENT /* here all predefined and user segments are placed into the SEGMENTS defined
above. */
    NON_BANKED,           /* runtime routines which must not be banked */
    COPY                  /* copy down information: how to initialize variables */
                        /* in case you want to use ROM_4000 here as well, make sure
                           that all files (incl. library files) are compiled with the
                           option: -OnB=b */
                  INTO  ROM_4000,ROM_C000;

    DEFAULT_ROM      INTO         PAGE_FE,         PAGE_FC, PAGE_FB, PAGE_FA,
PAGE_F9, PAGE_F8,
                       PAGE_F7, PAGE_F6, PAGE_F5, PAGE_F4, PAGE_F3, PAGE_F2,
PAGE_F1, PAGE_F0,
                       PAGE_EF, PAGE_EE, PAGE_ED, PAGE_EC, PAGE_EB, PAGE_EA,
PAGE_E9, PAGE_E8,
                       PAGE_E7, PAGE_E6, PAGE_E5, PAGE_E4, PAGE_E3, PAGE_E2,
PAGE_E1, PAGE_E0;
```

```
//          END

//PLACEMENT
     stack,              /* allocate stack first to avoid overwriting variables on overflow */
     DEFAULT_RAM            /* all variables, the default RAM location */
                    INTO  RAM;

     PAGED_RAM         INTO  /* when using banked addressing for variable data, make sure to
specify
                              the option -D__FAR_DATA on the compiler command line */
                         RAM_F8, RAM_F9, RAM_FA, RAM_FB, RAM_FC, RAM_FD;

   //.vectors          INTO  OSVECTORS; /* OSEK vector table */
END


ENTRIES /* keep the following unreferenced variables */
     //_vectab OsBuildNumber /* always allocate the vector table and all dependent objects */
END

STACKSIZE 0x100          /* size of the stack (will be allocated in DEFAULT_RAM) */

/* use these definitions in plane of the vector table ('vectors') above */
//VECTOR 0 _Startup /* reset vector: this is the default entry point for a C/C++ application. */
VECTOR 0 Entry  /* reset vector: this is the default entry point for a Assembly application. */
INIT Entry      /* for assembly applications: that this is as well the initialization entry point */

//;IVBR + $10
VECTOR 119      ResetFunc      ///* vector 119*/
VECTOR 118      ResetFunc      ///* vector 118*/
VECTOR 117      ResetFunc      // /* vector 117*/
VECTOR 116      ResetFunc      ///* vector 116*/
VECTOR 115      ResetFunc      ///* vector 115*/
VECTOR 114      ResetFunc      ///* vector 114*/
VECTOR 113      ResetFunc      ///* vector 113*/
VECTOR 112      ResetFunc      ///* vector 112*/

//;IVBR + $20
VECTOR 111      ResetFunc      ///* vector 111*/
VECTOR 110      ResetFunc      ///* vector 110*/
VECTOR 109      ResetFunc      ///* vector 109*/
VECTOR 108      ResetFunc      ///* vector 108*/
VECTOR 107      ResetFunc      ///* vector 107*/
VECTOR 106      ResetFunc      ///* vector 106*/
VECTOR 105      ResetFunc      ///* vector 105*/
VECTOR 104      ResetFunc      ///* vector 104*/

//;IVBR + $30
VECTOR 103      ResetFunc      ///* vector 103*/
VECTOR 102      ResetFunc      ///* vector 102*/
VECTOR 101      ResetFunc      ///* vector 101*/
VECTOR 100      ResetFunc      ///* vector 100*/
VECTOR 99       ResetFunc      ///* vector 99 */
VECTOR 98       ResetFunc      ///* vector 98 */
VECTOR 97       ResetFunc      ///* vector 97 */
VECTOR 96       ResetFunc      ///* vector 96 */
```

```
//;IVBR + $40
VECTOR 95       ResetFunc       ///* vector 95 */
VECTOR 94       ResetFunc       ///* vector 94 */
VECTOR 93       ResetFunc       ///* vector 93 */
VECTOR 92       ResetFunc       ///* vector 92 */
VECTOR 91       ResetFunc       ///* vector 91 */
VECTOR 90       ResetFunc       //* vector 90 */
VECTOR 89       ResetFunc       ///* vector 89 */
VECTOR 88       ResetFunc       ///* vector 88 */

//;IVBR + $50
VECTOR 87       ResetFunc       ///* vector 87 */
VECTOR 86       ResetFunc       ///* vector 86 */
VECTOR 85       ResetFunc       ///* vector 85 */
VECTOR 84       ResetFunc       ///* vector 84 */
VECTOR 83       ResetFunc       ///* vector 83 */
VECTOR 82       ResetFunc       ///* vector 82 */
VECTOR 81       ResetFunc       ///* vector 81 */
VECTOR 80       ResetFunc       ///* vector 80 */

//;IVBR + $60
VECTOR 79       ResetFunc       ///* vector 79 */
VECTOR 78       ResetFunc       ///* vector 78 */
VECTOR 77       ResetFunc       //* vector 77 */
VECTOR 76       ResetFunc       ///* vector 76 */
VECTOR 75       ResetFunc       ///* vector 75 */
VECTOR 74       ResetFunc       ///* vector 74 */
VECTOR 73       ResetFunc       ///* vector 73 */
VECTOR 72       ResetFunc       ///* vector 72 */

//;IVBR + $70
VECTOR 71       ResetFunc       ///* vector 71 */
VECTOR 70       ResetFunc       ///* vector 70 */
VECTOR 69       ResetFunc       ///* vector 69 */
VECTOR 68       ResetFunc       ///* vector 68 */
VECTOR 67       ResetFunc       ///* vector 67 */
VECTOR 66       ResetFunc       ///* vector 66 */
VECTOR 65       ResetFunc       ///* vector 65 */
VECTOR 64       ResetFunc       ///* vector 64 */

//;IVBR + $80
VECTOR 63       ResetFunc       ///* vector 63 */
VECTOR 62       ResetFunc       ///* vector 62 */
VECTOR 61       ResetFunc       ///* vector 61 */
VECTOR 60       ResetFunc       ///* vector 60 */
VECTOR 59       ResetFunc       ///* vector 59 */
VECTOR 58       ResetFunc       ///* vector 58 */
VECTOR 57       ResetFunc       ///* vector 57 */
VECTOR 56       ResetFunc       ///* vector 56 */

//;IVBR + $90
VECTOR 55       ResetFunc       ///* vector 55 */
VECTOR 54       ResetFunc       ///* vector 54 */
VECTOR 53       ResetFunc       ///* vector 53 */
VECTOR 52       ResetFunc       ///* vector 52 */
```

```
VECTOR 51      ResetFunc      ///* vector 51 */
VECTOR 50      ResetFunc      ///* vector 50 */
VECTOR 49      ResetFunc      ///* vector 49 */
VECTOR 48      ResetFunc      ///* vector 48 */

//;IVBR + $A0
VECTOR 47      ResetFunc      ///* vector 47 */
VECTOR 46      ResetFunc      ///* vector 46 */
VECTOR 45      ResetFunc      ///* vector 45 */
VECTOR 44      ResetFunc      ///* vector 44 */
VECTOR 43      ResetFunc      ///* vector 43 */
VECTOR 42      ResetFunc      ///* vector 42 */
VECTOR 41      ResetFunc      ///* vector 41 */
VECTOR 40      ResetFunc      ///* vector 40 */

//;IVBR + $B0
VECTOR 39      ResetFunc      ///* vector 39 */
VECTOR 38      ResetFunc      ///* vector 38 */
VECTOR 37      ResetFunc      ///* vector 37 */
VECTOR 36      ResetFunc      ///* vector 36 */
VECTOR 35      ResetFunc      ///* vector 35 */
VECTOR 34      ResetFunc      ///* vector 34 */
VECTOR 33      ResetFunc      ///* vector 33 */
VECTOR 32      ResetFunc      ///* vector 32 */

//;IVBR + $C0
VECTOR 31      ResetFunc      ///* vector 31 */
VECTOR 30      ResetFunc      ///* vector 30 */
VECTOR 29      ResetFunc      ///* vector 29 */
VECTOR 28      ResetFunc      ///* vector 28 */
VECTOR 27      ResetFunc      ///* vector 27 */
VECTOR 26      ResetFunc      //        ;Modulus down counter
VECTOR 25      KeypadInt      //    ;Port H (PIEH)
VECTOR 24      ResetFunc      //    ;Port J (PIEJ)

//;IVBR + $D0
VECTOR 23      ResetFunc      //    ;ATD1 (ATDCTL2 - ASCIE)
VECTOR 22      ResetFunc      //    ;ATD0 (ATDCTL2 - ASCIE)
VECTOR 21      ResetFunc      //    ;SCI1
VECTOR 20      SerInputInt0   //    ;SCI0
VECTOR 19      ResetFunc      //    ;SPI0
VECTOR 18      ResetFunc      //    ;Pulse Accumulator 0 input edge
VECTOR 17      ResetFunc      //    ;Pulse Accumulator 0 overflow
VECTOR 16      ResetFunc      //    ;Standard Timer 0 Overflow

//;IVBR + $E0
VECTOR 15      ResetFunc      //    ;Timer 0 Channel 7
VECTOR 14      ResetFunc      //    ;Timer 0 Channel 6
VECTOR 13      ResetFunc      //        ;Timer 0 Channel 5
VECTOR 12      ResetFunc      //        ;Timer 0 Channel 4
VECTOR 11      KeybrdInt      //        ;Timer 0 Channel 3
VECTOR 10      ResetFunc      //        ;Timer 0 Channel 2
VECTOR 9       ResetFunc      //        ;Timer 0 Channel 1
VECTOR 8       ResetFunc      //        ;Timer 0 Channel 0

//;IVBR + $F0
```

```
VECTOR 7     RealTimeInt   //      ;Real Time Interrupt
VECTOR 6     ResetFunc     //      ;IRQ
VECTOR 5     ResetFunc     //      ;XIRQ
VECTOR 4     ResetFunc     //      ;SWI
VECTOR 3     ResetFunc     //      ;Instruction Trap
VECTOR 2     ResetFunc     //      ;COP failure
VECTOR 1     ResetFunc     //      ;Clock Monitor
```

S-Records:

```
S12340001410CF40001644C31645AC1647824D39804C3A40180B010034180B000035A7A79C
S1234020A7A74F3708FC4C3980180BC00127180B000128180B000129180B00012A180B008A
S1234040012B180B00012C180B00012D180B02012E180B00012F180BD00127180B03012B7B
S1234060180BE00127180B000128180B000129180B00012A180B00012B180B02012C180B66
S123408000012D180B00012E180B00012F180BF00127180B04012810EF1640B8FD205A1525
S12340A04016436116467A20F335CD0000200435CD03090436FD313D1C025AC01C02488022
S12340C0180BE0024A180B5C00D8180B0700DA4DD90196DB96DD7920587920571803000002B
S12340E02061CE01F47E205CCE40F57E205A3DFE206126FB3DFE205C26F4863016415B1855
S123410003000F20611640EF863816415B1803000F20611640EF860C16415B1803000F20D5
S1234120611640EF860116415B1803000F20611640EF864016415B1803000F20611640EF44
S123414086801641 5BCE43D5CC0008CE42067E205A1642E5CE01F47E205C3D1641AF1D027F
S12341604880A7A71C0248801D025840A7A7A7A71C025880A7A7A7A7A7A7A7A7A7A7A722
S12341801D0258803D1641AF1D024880A7A71C0248801C025840A7A7A7A71C025880A7A730
S12341A0A7A7A7A7A7A7A7A7A7A71D0258803D37D6DBD6DD5ADD4FDB80FC333DFE205C26B9
S12341C04486007A20581643AFCE44701643C286017A20581643AFCE43DD1643C286027AEE
S12341E020581643AFCE43F21643C286037A20581643AFCE44071643C2CE42067E205ACE8A
S12342003D097E205C3DFE205C264486007A20581643AFCE43DD1643C286017A20581643DE
S1234220AFCE43F21643C286027A20581643AFCE44071643C286037A20581643AFCE4485DD
S12342401643C2CE42507E205ACE3D097E205C3DFE205C264486007A20581643AFCE43F235
S12342601643C286017A20581643AFCE44071643C286027A20581643AFCE44851643C286B0
S1234280037A20581643AFCE44851643C2CE429A7E205ACE3D097E205C3DFE205C26448614
S12342A0007A20581643AFCE44071643C286017A20581643AFCE44851643C286027A20581F
S12342C01643AFCE44851643C286037A20581643AFCE44851643C2CE41BC7E205ACE3D0914
S12342E07E205C3D3DCE20057E2055C65086206A00085326FA86017A20573DCE20051AE518
S1234300C614121AE57E20553DFD205581A6260286046A4002200FFD205581A6260286040D
S12343206A40025326FA8D20552503CD20057D205586017A20573DFD2055A600270C81A625
S1234340260286046A40080220F00820D9FD2055A60081A6260286046A4008025326F120B3
S1234360C5B620572B2B7320572707860216415B201FC614CE2005164392CE202D164392A2
S1234380CE2019164392CE2041164392860C16415B3D37A60081302602864F1641853418D9
S12343A003000F20611640EF30085326E6333DB62058C6001642FB3DB62058F6205E16429C
S12343C0FB3D1643373D16434D3DC605F0205F1AE516434D3D00000000000000004C434402
S12343E020526F7574696E652050726F6772616D2E0054686973206973206120544553541 8
S12344002E2E2E2E2E2E002D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D004D736734A5
S12344202E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E004D7367352E2E2E2E2E2E2E2E2E2E42
S12344402E2E2E2E2E004D7367362E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E004D7367372EA9
S12344602E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E00546563686E6F6C6F676963616C20417277
S1234480747320200020202020202020202020202020202020202020202020004B6579626F6116
S12344A0726420546573742E2E2E2E2E2E2E004D736731312E2E2E2E2E2E2E2E2E2E2E6F
S12344C02E2E2E180B10003B4C37804C3880180B0000C8180B1100C9180B0000CA180B2CB5
S12344E000CB180BFF00023D710000FE20032704097E2003FE20632704097E2063FE205CF5
S12345002704097E205CB62072270B437A20722605860B7A2070FE20612704097E2061FE2A
S123452020772611FE20792729097E2079CE3D097E20772004097E2077FE20772614FE204A
S1234540772607FCE41BC7E205ACE20197E207579207C180B8000370BA6302704070320F8B1
S12345603D4FCC80FC5ACF3D96CC180C00CF2000180BFF20010B860A7A2072B6024045FC65
S1234580206A46567C206A4C4E08732070261C860B7A2070722071FE206C7E206DB6206A91
S12345A075206B4575206B457A206C0B4C46804C4C084C4E084C4B804D4B40860B7A20707E
S12345C0CE00007E206A7E20777920717920737920697920747920 7B79207279207CCE20A5
S12345E0197E20753DB620711827008CCE206B1410F620711AE5E60073207110EFC1E02788
S123460077C1F02605722073206EB620732717792073C11126061D207402205CC11426589D
S12346201D2074012052D72B4FCE47021AE5A6007A206986017A207BB6207C263B86007AF3
```

S123464020581643AFCE449A1643C272207C86017A20581643AFCE44851643C286027A20AC
S1234660581643AFCE44851643C286037A20581643AFCE44851643C2873D1645E5B6207B9A
S1234680277FB62069FD20758D202D272D8D20552702204A3686017A20581643AFCE44858E
S12346A01643C286027A20581643AFCE44851643C232CD20197D207520243686027A205869
S12346C01643AFCE44851643C286037A20581643AFCE44851643C232CD202D7D20756A40DF
S12346E0027D207586017A2057792069CE42E47E205ACE001D7E2079CE3D097E2077792078
S12347007B3D000000000050000000000000000000001100001451310000005A534157EC
S123472032000043584445343300002056465452350004E42484759360000004D4A553750
S12347403800002C4B494F303900002E2F4C38502D00000000005B3D00000000135D00003F
S123476000000000000000000800000000000000000000000000001B0000000000000012
S1234780000086FF7A0264180B00026686F07A02607A026279207DCE012C7E20813DB620B2
S12347A07D2703B6207E3D643200FF0000180B00025E180B0F025F16486F732080272516D0
S12347C0487A1E026001F3CE48941648857B207E86017A207D86FF7A0260CE012C7E2081E0
S12347E01820008416486F732080272316487A1E026002F3CE489C1648857B207E86017AD3
S1234800207D86FF7A0260CE012C7E2081205916486F732080272316487A1E026004F3CEBC
S123482048A41648857B207E86017A207D86FF7A0260CE012C7E2081202E16486F73208045
S1234840272316487A1E026008F3CE48AC1648857B207E86017A207D86FF7A0260CE012CFF
S12348607E2081200379207DCE00AA7E20810BC6F77B207F86057A20803DF6207F587B0217
S1234880607B207F3DF6207F5454545451C40F1AE5E6003D3F31323F333F3F3F4134353F78
S12348A0363F3F3F4237382A393F3F3F432A303F233F3F3F44FFFFFFFFFFFFFFFFFFFFFF3B
S123FF1040004000400040004000400040004000400040004000400040004000400040004000CD
S123FF3040004000400040004000400040004000400040004000400040004000400040004000AD
S123FF5040004000400040004000400040004000400040004000400040004000400040004000CD
S123FF7040004000400040004000400040004000400040004000400040004000400040004000AD
S123FF9040004000400040004000400040004000400040004000400040004000400040004000CD
S123FFB040004000400040004000400040004000400040004000400040004000400047AD400079
S123FFD040004000400045684000400040004000400040004000400045764000400040002 5
S113FFF044E840004000400040004000400011
S9034000BC