INTRODUCTION

Traditionally, the microcontroller stack has only been used as a storage space for return addresses of subroutines or interrupt routines, where all ‘push’ and ‘pop’ operations were hidden. For the most part, users had no direct access to the information on the stack. The PIC18 microcontroller diverges from this tradition slightly. With the new PIC18 core, users now have access to the stack and can modify the stack pointer and stack data directly. Having such levels of access to the stack allows for some unique and interesting programming possibilities.

This application note describes specific information, registers, and instructions related to accessing the stack. An example is also included demonstrating a very simple task manager, an essential element for a real-time operating system (RTOS).

ACCESSING THE STACK

General Access

The entire stack of the PIC18 microcontroller is not mapped to memory. However, the top of the stack is mapped and is very simple to access during normal program operation. For stack access, four registers are provided in the Special Function Register (SFR) bank.

They are:
- TOSU
- TOSH
- TOSL
- STKPTR

The top of the stack is provided in registers TOSU, TOSH, and TOSL. Each stack memory location is 21-bits wide. Thus, register TOSU is only five-bits wide, while registers TOSH and TOSL are eight-bits wide.

The pointer to the top of the stack is provided in register STKPTR. The pointer is only five-bits wide, which accounts for a stack depth of 32 words. However, the first location is not counted, since it is not physically a memory location in the stack. The first location always contains the value 000000h, which means there are only 31 usable locations in the stack. Figure 1 shows the stack.

To access the data on the stack, the user only has to write the 5-bit pointer to the STKPTR register. The data is available in the TOS registers on the following instruction cycle.

Instructions

Aside from general access, there are two new instructions directly targeted for stack manipulation: PUSH and POP. Executing the PUSH instruction auto-increments the stack pointer and pushes the current program counter (PC) value to the TOS. Executing the POP instruction decrements the stack pointer.

FIGURE 1: THE PIC18 STACK

Note: Interrupts MUST be disabled when modifying the TOS or the STKPTR. If they are not disabled, users run the risk of causing unexpected program redirection.
THOUGHTS ABOUT STACK MANIPULATION

There are several possible applications for using the stack space. Some of them include:

- Program redirection
- Holding data/Passing parameters
- Calculating jumps
- Creating a software return stack

Among a number of possibilities, program redirection is probably the most dominant application for the PIC18 microcontroller. Having access to the stack allows access to the return addresses of interrupts and function calls. Thus, the program direction can be changed by modifying the return addresses or adding to them. The flow chart in Figure 2 presents an example of using the stack manipulation for program redirection.

In Figure 2, program direction is altered based on the number of data samples collected. After X number of samples, the pointer to an analysis function is forced onto the stack. Then, the interrupt ends normally. However, execution does not return to the main routine but to the analysis function. Example 1 outlines how program redirection may occur in code.

There is a distinct advantage to the program flow of Figure 2 versus non-stack manipulating operation. The analysis function is transparent to the main routine. To the main routine, the analysis function remains part of the interrupt, yet from the interrupt perspective, the analysis routine is not part of the interrupt. The net result is the data sampling interrupt routine will never lose data due to long analysis times.

FIGURE 2: MODIFIED RETURN FLOW CHART

EXAMPLE 1: PROGRAM REDIRECTION

```
MyInterruptRoutine
    ; Data collection interrupt
    ;
    decfsz DATA_COUNT, F ; Check for 8 samples
    retfie ; Resume normal execution
    movlw 0x08
    movwf DATA_COUNT ; Reset counter
    incf STKPTR, F ; Increment stack pointer
    movlw low MyAvgRoutine ; Load the TOS to point to averaging routine
    movwf TOSL
    movlw high MyAvgRoutine
    movwf TOSH
    movlw upper MyAvgRoutine
    movwf TOSU
    retfie ; Do average
MyAvgRoutine
    ; Average
    ;
    return
```
A STACK MANIPULATION EXAMPLE: A SIMPLE TASK MANAGER

The simple task manager shown in the appendices (the task manager code in Appendix C, with the supporting files in the other documents) is another example of program redirection. However, TIMER0 is the trigger source to indicate program redirection. Thus, TIMER0 acts as a program timer, or more appropriately, a task timer. When a task runs out of time, the task manager forces a swap to the next task in the list. Therefore, the task manager is preemptive.

The task manager uses the stack a little differently than it was traditionally designed to do. The stack is separated into four user defined blocks, one block for each task. There can be as many as four tasks running simultaneously, where each task has some subroutine, or interrupt return vector space. Figure 3 gives an example of how the stack may be divided. It can be divided differently according to the application. The lowest order block holds the pointers for the first task in the list.

FIGURE 3: AN EXAMPLE OF DIVIDING THE STACK

| Task 4 | 1Fh 18h 1Ah |
| Task 3 | 0Eh 0Dh 0Eh |
| Task 2 | 09h 08h 09h |
| Task 1 | 01h 00h 00h |

The task manager also manages the Special Function Registers (SFRs) to maintain data between task swaps. Without this, each task would have its data destroyed and cease to function as expected. Thus, the SFR data is stored in the General Purpose Registers (GPRs). As in the stack configuration, what SFRs are stored is defined by the user, in order to minimize wasting memory and process time.

There are two levels of priority assigned to each task. One priority is the position in the task list. Thus, Task 1 is the first to run and so on. The second level of priority is time. Each task has a time associated to it; low priority tasks ideally get less time and high priority tasks get more time. Basically, each task is assigned a percentage of the total process time.

This simple task manager gives the user the advantage of writing multiple programs, as if each program were on independent microcontrollers, yet run them on only one microcontroller. The task manager keeps track of the important registers and manages time so the user does not have to address all independent tasks as one large task. Of course, with time and space critical applications, this independent program concept is not always the best option.

MEMORY USAGE

The program memory usage of the task manager in Appendix C varies depending on how it is compiled into the application. Table 1 lists the smallest and largest. The percentages are calculated for the PIC18C452.

<table>
<thead>
<tr>
<th>TABLE 1: PROGRAM MEMORY USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
</tbody>
</table>

Like program memory, data memory is also dependent on the application. Table 2 shows the maximum and minimum data memory usage.

<table>
<thead>
<tr>
<th>TABLE 2: DATA MEMORY USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
</tbody>
</table>

CONCLUSION

Having access to the stack on PIC18 microcontrollers allows the user to apply some advanced programming techniques to 8-bit microcontroller applications. The task manager demonstrated in this application note shows how even sophisticated programming concepts can be executed in a small package.
APPENDIX A: SAMPLE PROGRAM

Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip products.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

; A Simple Task Manager v1.00 by Ross Fosler
; This is a small demonstration of the task manager.

; Definitions
#include <define.inc>
#include PROC_INCLUDE
#include macroins.inc
#include tm_inst.inc

EXTERN ALT_STATUS, ALT_W0

VAR1 UDATA_ACS

INT1 CODE

This is the interrupt handler for all interrupts other than TIMER0.
TIMER0 is dedicated to the task manager. Interrupt latency in the
TM is 8 instruction cycles. The STATUS and WREG is already saved.

InterruptHandler
; btfsc INTCON, INT0IF, A ; Check INT0
; goto HandleINT0
; btfsc INTCON, RBIF, A ; Check interrupt on change
; goto HandlerRBChange

retfint ; Macro to return from interrupt

GLOBAL InterruptHandler ; This line must me included

STP CODE
; Use this section to include any setup code upon power-up or reset.
Setup
    clrf TRISB
    return
GLOBAL Setup

; *******************************************************************
; *******************************************************************

TSK1 CODE
; *******************************************************************
; This is a demonstration task. Each task can trigger a task swap by
; using the 'swptsk' macro. Otherwise, the task manger will
; automatically swap at the end of its cycle.

Task1
    nop
    nop
    btg LATB,5
    nop
    swptsk ; Force the TM to swap

    btg LATB,7
    btg LATB,6
    nop
    swptsk

    bra Task1 ; This line must me included
GLOBAL Task1 ; This line must me included
; *******************************************************************
; *******************************************************************

TSK2 CODE
; *******************************************************************
; This is a demonstration task.

Task2
    btg LATB,4
    ; swptsk ; Force the TM to swap
    bra Task2
GLOBAL Task2 ; This line must me included
; *******************************************************************
; *******************************************************************

END
APPENDIX B: THE START-UP ROUTINE

; ***********************************************
; ; A Simple Task Manager v1.00 by Ross Fosler ;
; ; This is the start-up routine for the task manager.;
; ***********************************************;

; ***********************************************
#include <define.inc>
#include PROC_INCLUDE ; Processor include file
#include <var.inc>
#include <macroins.inc>
; ***********************************************

TEST CODE 0x00
bra 0x200
TEST2 CODE 0x08
bra 0x208

STRT CODE 0x0200
goto TMSetup
INT CODE 0x0208
goto TaskManager

; ***********************************************
;
;This routine sets up all important registers for PIC OS2 to run
;properly.
;
TMSetup
IFDEF SETUP_NAME
    call SETUP_NAME ; Do some user setup
ENDIF

    movlw TIMER_PRESCALE ; Set Prescaler
    movwf T0CON, A
    bsf T0CON, T08BIT, A ; Force 8-bit mode
    bsf T0CON, TMR0ON, A ; Turn TM0 on
    clrf TASK_POINTER, A ; Init the important registers
    clrf TABLE_POINTER, A
    clrf TASK_COMMAND, A
    clrf TASK_BUFFER, A
    clrf TASK_COUNTER, A
    movlw TASK1 ; Prime the task table
    movff WREG, TASK_TABLE
    movlw TASK2
    movff WREG, TASK_TABLE + 1
    movlw TASK3
    movff WREG, TASK_TABLE + 2
    movlw TASK4
    movff WREG, TASK_TABLE + 3
IFDEF TASK1_NAME ; Seed task1
    movff TASK_TABLE, STKPTR
    movlw low TASK1_NAME
    movwf TOSL, A
ENDIF
movlw high TASK1_NAME
movwf TOSH, A
clrf TOSU, A
incf TASK_COUNTER, F, A
ENDIF

IFDEF TASK2_NAME ; Seed task2
movff TASK_TABLE+1, STKPTR
movlw low TASK2_NAME
movwf TOSL, A
movlw high TASK2_NAME
movwf TOSH, A
clrf TOSU, A
incf TASK_COUNTER, F, A
ENDIF

IFDEF TASK3_NAME ; Seed task3
movff TASK_TABLE+2, STKPTR
movlw low TASK3_NAME
movwf TOSL, A
movlwhigh TASK3_NAME
movwf TOSH, A
clrf TOSU, A
incf TASK_COUNTER, F, A
ENDIF

IFDEF TASK4_NAME ; Seed task4
movff TASK_TABLE+3, STKPTR
movlw low TASK4_NAME
movwf TOSL, A
movlw high TASK4_NAME
movwf TOSH, A
clrf TOSU, A
incf TASK_COUNTER, F, A
ENDIF

movlw TASK1 ; Reset the stack pointer
movwf STKPTR, A

movlw high TASK_INFO_TABLE ; Setup priority
movwf FSR0H
movlw low TASK_INFO_TABLE
movwf FSR0L

movlw ((TASK1_TIME * 4) + 0x00)
movwf POSTINC0, A
movlw ((TASK2_TIME * 4) + 0x01)
movwf POSTINC0, A
movlw ((TASK3_TIME * 4) + 0x02)
movwf POSTINC0, A
movlw ((TASK4_TIME * 4) + 0x03)
movwf POSTINC0, A

movlw TASK1_TIME ; Init the timer
comf WREG, W, A
bcf WREG, 0, A
bcf WREG, 1, A
movwf TMR0L, A

bcf RCON, IPEN, A ; No priority levels
bsf INTCON, TMR0IE, A ; Enable timer 0 interrupt
bsf INTCON, GIE, A ; Enable global interrupts

return 0
;

***************************************************************************

END
APPENDIX C: THE TASK MANAGER

; ******************************************************************************
; A Simple Task Manager v1.00 by Ross Fosler ;
; ******************************************************************************

; ******************************************************************************
#include <define.inc>
#include PROC_INCLUDE ; Processor include file
#include <macroins.inc>

; ******************************************************************************
_ATM_SCRATCH UDATA
TEMP res 1

; ******************************************************************************
IFDEF INT_HAND_NAME
EXTERN INT_HAND_NAME
ENDIF
IFDEF SAVE_BSR
EXTERN BACKUP_BSR
ENDIF
IFDEF SAVE_FSR0L
EXTERN BACKUP_FSR0L
ENDIF
IFDEF SAVE_FSR0H
EXTERN BACKUP_FSR0H
ENDIF
IFDEF SAVE_FSR1L
EXTERN BACKUP_FSR1L
ENDIF
IFDEF SAVE_FSR1H
EXTERN BACKUP_FSR1H
ENDIF
IFDEF SAVE_PRODH
EXTERN BACKUP_PRODH
ENDIF
IFDEF SAVE_PRODL
EXTERN BACKUP_PRODL
ENDIF
IFDEF SAVE_FSR2L
EXTERN BACKUP_FSR2L
EXTERN ALT_FSR2L
ENDIF
IFDEF SAVE_FSR2H
EXTERN BACKUP_FSR2H
EXTERN ALT_FSR2H
ENDIF
IFDEF SAVE_TBLPTRU

} **end of source code**
EXTERN BACKUP_TBLPTRU
ENDIF
IFDEF SAVE_TBLPTRH
EXTERN BACKUP_TBLPTRH
ENDIF
IFDEF SAVE_TBLPTRL
EXTERN BACKUP_TBLPTRL
ENDIF
IFDEF SAVE_TABLAT
EXTERN BACKUP_TABLAT
ENDIF
EXTERN TASK_TABLE, TASK_INFO_TABLE
EXTERN BACKUP_WREG, BACKUP_STATUS
EXTERN TASK_POINTER, TABLE_POINTER, TASK_COUNTER
EXTERN TASK_COMMAND, TASK_BUFFER
EXTERN ALT_W0
EXTERN ALT_STATUS

; *******************************************************************
IFDEF LFSR_BUG
; Macro to work around lfsr bug
ldfsr2 macro JUNK, MYLIT
movff WREG, TEMP
movlw high MYLIT
movwf FSR2H
movlw low MYLIT
movwf FSR2L
movff TEMP, WREG
endm
ELSE
ldfsr2 macro _FSR, _REG
lfsr _FSR, _REG
endm
ENDIF
; *******************************************************************

; **********************************************************
; *** Stop the Timer **************************************
bcf T0CON, TMR0ON, A ; Stop the timer
; **********************************************************

; **********************************************************
; *** Save Important Data ********************************
movwf ALT_W0, A ; Copy WREG
movff STATUS, ALT_STATUS ; Copy STATUS
; **********************************************************

; *** Test the Interrupt Source ***
IFDEF INT_HAND_NAME
btfss INTCON, TMR0IF, A
goto NT_HAND_NAME ; Check other interrupt sources
ENDIF

; **********************************************************
; **************************************************

movf TABLE_POINTER, W, A

IFDEF SAVE_FSR2L
movf FSR2L, ALT_FSR2L
ENDIF

IFDEF SAVE_FSR2H
movf FSR2H, ALT_FSR2H
ENDIF

ldfsr2 2, TASK_TABLE ; Save pointer to TOS
movf STKPTR, PLUSW2
ldfsr2 2, BACKUP_WREG ; Save WREG
movf ALT_W0, PLUSW2
ldfsr2 2, BACKUP_STATUS ; Save STATUS
movf ALT_STATUS, PLUSW2

IFDEF SAVE_BSR
ldfsr2 2, BACKUP_BSR ; Save BSR
movf BSR, PLUSW2
ENDIF

IFDEF SAVE_FSR0H
ldfsr2 2, BACKUP_FSR0H ; Save FSR0H
movf FSR0H, PLUSW2
ENDIF

IFDEF SAVE_FSR0L
ldfsr2 2, BACKUP_FSR0L ; Save FSR0L
movf FSR0L, PLUSW2
ENDIF

IFDEF SAVE_FSR1H
ldfsr2 2, BACKUP_FSR1H ; Save FSR1H
movf FSR1H, PLUSW2
ENDIF

IFDEF SAVE_FSR1L
ldfsr2 2, BACKUP_FSR1L ; Save FSR1L
movf FSR1L, PLUSW2
ENDIF

IFDEF SAVE_FSR2H
ldfsr2 2, BACKUP_FSR2H ; Save FSR2H
movf ALT_FSR2H, PLUSW2
ENDIF

IFDEF SAVE_FSR2L
ldfsr2 2, BACKUP_FSR2L ; Save FSR2L
movf ALT_FSR2L, PLUSW2
ENDIF

IFDEF SAVE_PRODH
ldfsr2 2, BACKUP_PRODH ; Save PRODH
movf PRODH, PLUSW2
ENDIF

IFDEF SAVE_PRODL
ldfsr2 2, BACKUP_PRODL ; Save PRODL
movf PRODL, PLUSW2
ENDIF
IFDEF SAVE_TBLPTRU
    ldfrs2 2, BACKUP_TBLPTRU ; Save TBLPTRU
    movff TBLPTRU, PLUSW2
ENDIF

IFDEF SAVE_TBLPTRH
    ldfrs2 2, BACKUP_TBLPTRH ; Save TBLPTRH
    movff TBLPTRH, PLUSW2
ENDIF

IFDEF SAVE_TBLPTRL
    ldfrs2 2, BACKUP_TBLPTRL ; Save TBLPTRL
    movff TBLPTRL, PLUSW2
ENDIF

IFDEF SAVE_TABLAT
    ldfrs2 2, BACKUP_TABLAT ; Save TABLAT
    movff TABLAT, PLUSW2
ENDIF

; *********************************************************
; *** Increment the Task Pointer **************************
IncrementTaskPointer
    incf ASK_POINTER, F, A ; Increment the task pointer
; *********************************************************

; *** Reset Interrupt Flag ********************************
    bcf NTCON, TMR0IF, A ; Clear interrupt
; *********************************************************

; *** Test the Task Pointer *******************************
    movf TASK_COUNTER, W, A
    cpfslt TASK_POINTER, A ; Is the pointer lt the counter?
    clrf TASK_POINTER, A ; No, reset the pointer
; *********************************************************

; *** Find the task **************************************
    clrf WREG2, A
    ldfrs2 2, TASK_INFO_TABLE; Set up pointer to priority table

TstTsk
    movlw 0x03
    andwf POSTINC2, W, A ; Mask off upper 6 bits, get task no#
    cpfseq TASK_POINTER, A ; Does the task numbers match?
    bra NxtTsk ; No

    movff WREG2, TABLE_POINTER ; Yes, store pointer
NxtTsk
    incf WREG2, F, A ; Check the next task
    movlw 0x04
    cpfseq WREG2, A ; Is the last possible task checked?
    bra TstTsk

    movf TABLE_POINTER, W, A
; *********************************************************

; *** Set the Priority ************************************
SetPriorityTimer
    ldfrs2 2, TASK_INFO_TABLE; Set up pointer to priority table
    movf PLUSW2, W, A
    andlw 0xFC ; Pull out priority bits
    bz IncrimentTaskPointer ; Goto next task if no priority
    comf WREG, W, A ; Invert and set TMR0

© 2002 Microchip Technology Inc.
bcf WREG, 0, A
bcf WREG, 1, A
movf TMR0L, A

; *******************************************************
; *** Restore the Saved data *****************************
RecallSavedData
Global RecallSavedData

movf TABLE_POINTER, W, A
ldfsr2 2, TASK_TABLE ; Restore pointer to TOS
movff PLUSW2, STKPTR
ldfsr2 2, BACKUP_WREG ; Restore WREG
movff PLUSW2, ALT_W0
ldfsr2 2, BACKUP_STATUS ; Restore STATUS
movff PLUSW2, STATUS

IFDEF SAVE_BSR
ldfsr2 2, BACKUP_BSR ; Restore BSR
movff PLUSW2, BSR
ENDIF

IFDEF SAVE_FSR0H
ldfsr2 2, BACKUP_FSR0H ; Restore FSR0H
movff PLUSW2, FSR0H
ENDIF

IFDEF SAVE_FSR0L
ldfsr2 2, BACKUP_FSR0L ; Restore FSR0L
movff PLUSW2, FSR0L
ENDIF

IFDEF SAVE_FSR1H
ldfsr2 2, BACKUP_FSR1H ; Restore FSR1H
movff PLUSW2, FSR1H
ENDIF

IFDEF SAVE_FSR1L
ldfsr2 2, BACKUP_FSR1L ; Restore FSR1L
movff PLUSW2, FSR1L
ENDIF

IFDEF SAVE_FSR2H
ldfsr2 2, BACKUP_FSR2H ; Restore FSR2H
movff PLUSW2, ALT_FSR2H
ENDIF

IFDEF SAVE_FSR2L
ldfsr2 2, BACKUP_FSR2L ; Restore FSR2L
movff PLUSW2, ALT_FSR2L
ENDIF

IFDEF SAVE_PRODH
ldfsr2 2, BACKUP_PRODH ; Restore PRODH
movff PLUSW2, PRODH
ENDIF

IFDEF SAVE_PRODL
ldfsr2 2, BACKUP_PRODL ; Restore PRODL
movff PLUSW2, PRODL
ENDIF

IFDEF SAVE_TBLPTRU
ldfsr2, 2, BACKUP_TBLPTRU ; Restore TBLPTRU
movff PLUSW2, TBLPTRU

ENDIF

IFDEF SAVE_TBLPTRH
ldfsr2, 2, BACKUP_TBLPTRH ; Restore TBLPTRH
movff PLUSW2, TBLPTRH
ENDIF

IFDEF SAVE_TBLPTRL
ldfsr2, 2, BACKUP_TBLPTRL ; Restore TBLPTRL
movff PLUSW2, TBLPTRL
ENDIF

IFDEF SAVE_TABLAT
ldfsr2, 2, BACKUP_TABLAT ; Restore TABLAT
movff PLUSW2, TABLAT
ENDIF

IFDEF SAVE_FSR2H
movff ALT_FSR2H, FSR2H
ENDIF

IFDEF SAVE_FSR2L
movff ALT_FSR2L, FSR2L
ENDIF

movff ALT_W0, WREG

; *********************************************************
; *** Start the Timer *************************************
bsf T0CON, TMR0ON, A ; Start the timer
; *********************************************************
retfie 0

; **********************************************************
END
APPENDIX D: VARIABLES

; *****************************************************************
; A Simple Task Manager v1.00 by Ross Fosler
; Variables used for the task manager.
; *****************************************************************

; *******************************************************************
CONSTANT TABLE_DEPTH = 0x04
; *******************************************************************

; *******************************************************************
EXTERN TaskManager
IFDEF TASK1_NAME ; Include any pre-defined tasks
EXTERN TASK1_NAME
ENDIF
IFDEF TASK2_NAME
EXTERN TASK2_NAME
ENDIF
IFDEF TASK3_NAME
EXTERN TASK3_NAME
ENDIF
IFDEF TASK4_NAME
EXTERN TASK4_NAME
ENDIF
IFDEF SETUP_NAME
EXTERN SETUP_NAME
ENDIF
; *******************************************************************

; *******************************************************************
ACS udata_acs
; *******************************************************************
TASK_POINTER res 1 ; Pointer to running task
TABLE_POINTER res 1 ; Pointer to data tables
TASK_COUNTER res 1 ; Number of tasks
GLOBAL TASK_POINTER, TABLE_POINTER, TASK_COUNTER

ALT_W0 res 1 ; An alternate WREG
ALT_STATUS res 1 ; An alternate STATUS
IFDEF SAVE_FSR2L ; An alternate FSR2L
ALT_FSR2L res 1
GLOBAL ALT_FSR2L
ENDIF
IFDEF SAVE_FSR2H ; An alternate FSR2H
ALT_FSR2H res 1
GLOBAL ALT_FSR2H
ENDIF

TASK_COMMAND res 1 ; Register globally available to control tasks
TASK_BUFFER res 1 ; Buffer to hold a new task
GLOBAL TASK_COMMAND, TASK_BUFFER, ALT_W0
GLOBAL ALT_STATUS

; ********************************************************************************

TBL udata ; Tables

; ********************************************************************************

TASK_TABLE res TABLE_DEPTH ; Table for holding pointers
BACKUP_WREG res TABLE_DEPTH
BACKUP_STATUS res TABLE_DEPTH
TASK_INFO_TABLE res TABLE_DEPTH ; Task number and priority table

GLOBAL TASK_TABLE, TASK_INFO_TABLE
GLOBAL BACKUP_WREG, BACKUP_STATUS

IFDEF SAVE_BSR
BACKUP_BSR res TABLE_DEPTH
GLOBAL BACKUP_BSR
ENDIF

IFDEF SAVE_FSR0L
BACKUP_FSR0L res TABLE_DEPTH
GLOBAL BACKUP_FSR0L
ENDIF

IFDEF SAVE_FSR0H
BACKUP_FSR0H res TABLE_DEPTH
GLOBAL BACKUP_FSR0H
ENDIF

IFDEF SAVE_FSR1L
BACKUP_FSR1L res TABLE_DEPTH
GLOBAL BACKUP_FSR1L
ENDIF

IFDEF SAVE_FSR1H
BACKUP_FSR1H res TABLE_DEPTH
GLOBAL BACKUP_FSR1H
ENDIF

IFDEF SAVE_PRODH
BACKUP_PRODH res TABLE_DEPTH
GLOBAL BACKUP_PRODH
ENDIF

IFDEF SAVE_PRODL
BACKUP_PRODL res TABLE_DEPTH
GLOBAL BACKUP_PRODL
ENDIF

IFDEF SAVE_TBLPTRU
BACKUP_TBLPTRU res TABLE_DEPTH
GLOBAL BACKUP_TBLPTRU
ENDIF

IFDEF SAVE_TBLPTRH
BACKUP_TBLPTRH res TABLE_DEPTH
GLOBAL BACKUP_TBLPTRH
ENDIF

IFDEF SAVE_TBLPTRL
BACKUP_TBLPTRL res TABLE_DEPTH
GLOBAL BACKUP_TBLPTRL
ENDIF
ENDIF

IFDEF SAVE_TABLAT
BACKUP_TABLAT  res TABLE_DEPTH
  GLOBAL   BACKUP_TABLAT
ENDIF

IFDEF SAVE_FSR2L
BACKUP_FSR2L  res TABLEDEPTH
  GLOBAL   BACKUP_FSR2L
ENDIF

IFDEF SAVE_FSR2H
BACKUP_FSR2H  res TABLEDEPTH
  GLOBAL   BACKUP_FSR2H
ENDIF

; ***************************************************************************
APPENDIX E: COMPLEX MACRO INSTRUCTIONS

; Some common macros for PIC18 by Ross Fosler
; v1.00 01/05/01
;
; brset MYFILE, MYBIT, MYBANK, WHERE; Bit tests
; brclr MYFILE, MYBIT, MYBANK, WHERE
;
; cffblt MYFILE1, MYFILE2, MYBANK, WHERE; Compare file w/ file
; cffbgd MYFILE1, MYFILE2, MYBANK, WHERE
; cffbdg MYFILE1, MYFILE2, MYBANK, WHERE
; cffbeq MYFILE1, MYFILE2, MYBANK, WHERE
; cffbne MYFILE1, MYFILE2, MYBANK, WHERE
;
; cflblt MYFILE1, MYLIT1, MYBANK, WHERE; Compare file w/ literal
; cflbgd MYFILE1, MYLIT1, MYBANK, WHERE
; cflbdg MYFILE1, MYLIT1, MYBANK, WHERE
; cflbeq MYFILE1, MYLIT1, MYBANK, WHERE
; cflbne MYFILE1, MYLIT1, MYBANK, WHERE
;
; movlf MYLIT, MYFILE, MYBANK ; Move literal to file
; addff MYFILE1, MYFILE2, MYDIRECTION, MYBANK ; Add file to file
; addfl MYFILE1, MYLIT1, MYDIRECTION, MYBANK ; Add file to literal
; andff MYFILE1, MYFILE2, MYDIRECTION, MYBANK ; And file to file
; andfl MYFILE1, MYLIT1, MYDIRECTION, MYBANK ; And file to literal
; iorff MYFILE1, MYFILE2, MYDIRECTION, MYBANK ; Ior file to file
; iorf MYFILE1, MYLIT1, MYDIRECTION, MYBANK ; Ior file to literal
; xorff MYFILE1, MYFILE2, MYDIRECTION, MYBANK ; Xor file to file
; xorf MYFILE1, MYLIT1, MYDIRECTION, MYBANK ; Xor file to literal
;
; *****************************************************************************
; *****************************************************************************

W equ 0 ; To WREG
F equ 1 ; To FILE
A equ 0 ; Use Access Bank
B equ 1 ; Use BSR
WREG2 equ PRODH
WREG3 equ PRODL
;
; *** Common Branch Instructions *******************************************
; Notes:W is destroyed except for brset and brclr.
; All branching is limited to 7 bits in either direction of the
; PC, thus these branch instructions cannot reach all memory.
;
; *****************************************************************************

; *** BRanch if bit is SET
brset macro MYFILE, MYBIT, MYBANK, WHERE
  btfs MYFILE, MYBIT, MYBANK
  bra WHERE
endm
;
; *** BRanch if bit is CLeaR
brclr macro MYFILE, MYBIT, MYBANK, WHERE
  btfsc MYFILE, MYBIT, MYBANK
  bra WHERE
endm
;
; *****************************************************************************

; *** Compare File with File and Branch if Less Than
; *** IF F1 < F2 THEN branch
cffblt macro MYFILE1, MYFILE2, MYBANK, WHERE
    movf MYFILE2, W, MYBANK
    subwf MYFILE1, W, MYBANK
    bn WHERE
endm

; *** Compare File with File and Branch if Greater Than
; *** IF F1 > F2 THEN branch
cffbggt macro MYFILE1, MYFILE2, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    subwf MYFILE2, W, MYBANK
    bn WHERE
endm

; *** Compare File with File and Branch if Equal
; *** IF F1 = F2 THEN branch
cffbeq macro MYFILE1, MYFILE2, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    subwf MYFILE2, W, MYBANK
    bz WHERE
endm

; *** Compare File with File and Branch if Not Equal
; *** IF F1 <> F2 THEN branch
cffbne macro MYFILE1, MYFILE2, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    subwf MYFILE2, W, MYBANK
    bnz WHERE
endm

; *************************************************************************
; *************************************************************************
; *** Compare File with Literal and Branch if Less Than
; *** IF F1 < L1 THEN branch
cflblt macro MYFILE1, MyLIT1, MYBANK, WHERE
    movlw MYLIT1
    subwf MYFILE1, W, MYBANK
    bn WHERE
endm

; *** Compare File with Literal and Branch if Greater Than
; *** IF F1 > L1 THEN branch
cflbggt macro MYFILE1, MyLIT1, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    sublw MYLIT1
    bn WHERE
endm

; *** Compare File with Literal and Branch if Equal
; *** IF F1 = L1 THEN branch
cflbeq macro MYFILE1, MyLIT1, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    sublw MYLIT1
    bz WHERE
endm

; *** Compare File with Literal and Branch if Not Equal
; *** IF F1 <> L1 THEN branch
cflbne macro MYFILE1, MyLIT1, MYBANK, WHERE
    movf MYFILE1, W, MYBANK
    sublw MYLIT1
    bnz WHERE
endm

; *************************************************************************
; ****************************************************************************
; *** Other Instructions ****************************************************
; ****************************************************************************

; *** MOVe Literal to File ****************************************
; Notes:W is destroyed in this macro.
movlf macro MYLIT, MYFILE, MYBANK
    movlw MYLIT
    movwf MYFILE, MYBANK
endm

; ****************************************************************************
; *** ADD File to File ********************************************
; Notes:Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
addff macro MYFILE1, MYFILE2, MYDIRECTION, MYBANK
    movf MYFILE2, W, MYBANK
    addwf MYFILE1, MYDIRECTION, MYBANK
endm

; ****************************************************************************
; *** ADD File to Literal *****************************************
; Notes:Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
addfl macro MYFILE1, MYLIT1, MYDIRECTION, MYBANK
    movlw MYLIT1
    addwf MYFILE1, MYDIRECTION, MYBANK
endm

; ****************************************************************************
; *** AND File to File ********************************************
; Notes:Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
andff macro MYFILE1, MYFILE2, MYDIRECTION, MYBANK
    movf MYFILE2, W, MYBANK
    andwf MYFILE1, MYDIRECTION, MYBANK
endm

; ****************************************************************************
; *** AND File to Literal *****************************************
; Notes:Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
andfl macro MYFILE1, MYLIT1, MYDIRECTION, MYBANK
    movlw MYLIT1
    andwf MYFILE1, MYDIRECTION, MYBANK
endm

; ****************************************************************************
; *** Inclusive OR File to File ***********************************
; Notes:Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
iorff macro MYFILE1, MYFILE2, MYDIRECTION, MYBANK
    movf MYFILE2, W, MYBANK
    iorwf MYFILE1, MYDIRECTION, MYBANK
endm

; ****************************************************************************
; *** Inclusive OR File to Literal ****************************************
; Notes: Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
iорф macro MYFILE1, MYLIT1, MYDIRECTION, MYBANK
    movlw MYLIT1
    iorwf MYFILE1, MYDIRECTION, MYBANK
endm

; *****************************************************************

; *** XOR File to File ********************************************
; Notes: Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
xorф macro MYFILE1, MYFILE2, MYDIRECTION, MYBANK
    movf MYFILE2, W, MYBANK
    xorwf MYFILE1, MYDIRECTION, MYBANK
endm

; *****************************************************************

; *** XOR File to Literal ****************************************
; Notes: Direction selects either the WREG or FILE1.
; W is destroyed in this macro.
xorф macro MYFILE1, MYLIT1, MYDIRECTION, MYBANK
    movlw MYLIT1
    xorwf MYFILE1, MYDIRECTION, MYBANK
endm

; *****************************************************************
; *****************************************************************************
APPENDIX F: TASK MANAGER MACROS

; *****************************************************************
; A Simple Task Manager v1.00 by Ross Fosler
; Commands for the Task Manager
; *****************************************************************

; *******************************************************************
swptsk macro
    bsf INTCON, TMR0IF, A ; Force an interrupt
endm

; *******************************************************************
retfint macro
    movff ALT_STATUS, STATUS ; Return STATUS
    movff ALT_W0, WREG ; Return WREG
    bsf T0CON, TMR0ON, A ; Start the timer
    retfie
endm

; *******************************************************************
APPENDIX G: DEFINITION FILE

; *****************************************************************
; A Simple Task Manager v1.00 by Ross Fosler
; This is a definition file used to incorporate tasks and
; priorities at the start of the task manager.
; *****************************************************************

; The values after correspond to the position in the hardware stack
; used by the tasks. Position 0 is not valid since it is set to
; always return a 0x0000 (reset).

#DEFINE TASK1 0x01
#DEFINE TASK2 0x08
#DEFINE TASK3 0x10
#DEFINE TASK4 0x18

; *******************************************************************

; The following defines the time allotted to the preloaded tasks.
; The value 0x00 corresponds to a null task; values 0x01 through 0x3F
; set the max allowed time for the task to run before it is
; interrupted.

#DEFINE TASK1_TIME 0x3F
#DEFINE TASK2_TIME 0x02
#DEFINE TASK3_TIME 0x00
#DEFINE TASK4_TIME 0x00

; *******************************************************************

; The following defines the names of the preloaded tasks. Uncomment
; or comment these as necessary for preloaded tasks. There must
; be at least one task to pre-load.

#DEFINE TASK1_NAME Task1
#DEFINE TASK2_NAME Task2
;//#DEFINE TASK3_NAME Task3Name
;//#DEFINE TASK4_NAME Task4Name

; *******************************************************************

; This value affects the task time. Valid range from 0x00 to 0x07.

#DEFINE TIMER_PRESCALE 0x04

; *******************************************************************

; Set the name of the interrupt handler. Comment out if none.

;//#DEFINE INT_HAND_NAME InterruptHandler

; *******************************************************************

; Set the name of the setup routine. Comment out if none.

#DEFINE SETUP_NAME Setup

; *******************************************************************
; *******************************************************************
; Set up the SFRs to be managed by the task manager. Comment out the
; registers that are not shared across more than one task. It is best
; to comment out as many as possible to reduce memory usage and
; task manager execution length.

#define save_fsr0h
#define save_fsr0l
#define save_fsr1h
#define save_fsr1l
#define save_fsr2h
#define save_fsr2l
#define save_prodh
#define save_prodh
#define save_bsr
#define save_tblptru
#define save_tblptrh
#define save_tblptrl
#define save_tbllat

; *******************************************************************

; Setup the specific processor file to use.

#define proc_include P18C452.INC

; *******************************************************************

; Uncomment if the device has the lfsr bug.

#define lfsr_bug

; *******************************************************************
APPENDIX H: SOURCE CODE FOR THIS APPLICATION NOTE

In addition to the complete source code listings presented here, all of the programs discussed in this application note are available to users as a Zip file archive. The archive, which also includes all necessary include and assembler files, may be downloaded from the Microchip website at:

www.microchip.com
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.