

Notes on programming

In particular assembly

Hints for programming (1/6)

- **Write the shortest possible program**
 - reduces use of resources,
 - easier to maintain, modify, and understand
 - If long, break it into short modules
- **Evaluate shortcuts before implementing them**
 - Tricks may introduce other problems
 - Don't sacrifice clarity!!!

Hints for programming (2/6)

- **Write programs easy to understand**
 - follow similar structures for similar tasks whenever possible
 - Assign appropriate names to resources and constants
 - Document
- **Write programs easy to modify**
 - different tasks can be achieved by simple changes in code when the design is clear
 - Reduce development costs

Hints for programming (3/6)

- **Document your program as you work on it**
 - Without documentation, programs are difficult to read, understand, modify or debug
 - documenting after finishing the program usually yields a lower quality documentation and higher cost
- **Create your own catalog of codes for common tasks**
 - different projects require similar steps and subtasks to be followed.
 - Associate codes and code structures for the tasks
 - Study the assembler to learn about tools.

Hints for programming (4/6)

- **Know your hardware system**
 - Hardware and program design very often goes in parallel.
 - You can only program for what you have
 - Be sure you know your peripherals
- **Avoid using unimplemented bits of memory or registers**
 - Programs may not run in new platforms
 - The gain in memory is not worth the risks

Hints for programming (5/6)

- **If possible, use a single resource for a single purpose**
 - Multiple purposes only if absolutely required
 - Plan for any multiple use in advance
- **Program for lowest power consumption**
 - choose the mode suitable for your application.
 - Prefer interrupts to polling
 - Put the system and unused components to sleep when possible

Hints for programming (6/6)

- **The most important: PLAN BEFORE CODING!!**
 - Use pseudo codes, flow charts, etc.
 - Assign resources correctly
 - Plan for subroutines connections
 - Plan individual subroutines
 - Reuse tested codes

Source File Format

Advisable but not compulsory

Elementary Components (1/3)

1. Documentation

- Explains purpose, author, etc.
- Very important for debugging and reading.

2. Constant definitions

- Use standard constants with
`#include "msp430.h"`
- Define convenient constant integers and registers to be used as operands in instructions using appropriate directives
- Constants in memory can be defined with memory allocation and initialization directives

Elementary Components (2/3)

3. Memory Management

- Can be used for constants or for initializing and reserving space for data
- MSP430 accepts initialized constant data in ROM space
- Data generated or changed during execution must go in RAM section

4. Executable code

- Main code, subroutines and macros

Elementary Components (3/3)

5. Reset and interrupt vector allocation

- Reset vector allocation **must always go.**
- Interrupt vectors only when necessary

6. Ending Directive END

- Anything after this directive is ignored.

```

1  ;*****
2  MSP430G2xx1 Demo - Software Toggle P1.0
3
4  Description: Toggle P1.0 by xor'ing P1.0 inside of a software loop.
5  ACLK = n/a, MCLK = SMCLK = default DCO
6
7  MSP430G2xx1
8
9  /|\|
10 | | |
11 --| RST
12 |
13 |
14
15 Based on code written by D. Dang
16 Texas Instruments Inc.
17 October 2010
18 Built with IAR Embedded Workbench Version: 5.10
19 ;*****
20 #include "msp430g2231.h" ; standard constants
21 LED EQU 01h ;LED at pin P1.0
22 DELAY EQU 50000
23 #define COUNTER R15 ;using R15 as counter
24
25 -----
26 ORG 0F800h ; Program Reset
27
28 RESET mov.w #0280h, SP ; Initialize stackpointer
29 StopWDT mov.w #WDT0PW+WDTHOLD,&WDTCTL ; Stop WDT
30 SetupP1 bis.b #001h,&P1DIR ; P1.0 output
31
32 Mainloop xor.b #LED,&P1OUT ; Toggle LED
33 Wait mov.w #DELAY,COUNTER ; Delay to counter
34 L1 dec.w COUNTER ; Decrement counter
35 jnz L1 ; Delay over?
36 jmp Mainloop ; Again
37
38 -----
39 ; Interrupt Vectors
40
41 -----
42 ORG 0FFFEh ; MSP430 address for
   DW RESET ; RESET Vector
   END

```

Documentation

Constants

Declaration

Absolute directive

Executable

Code

Reset vector
allocation

```

1  ;*****
2  ;   MSP430G2xx1 Demo - Software Toggle P1.0
3  ;   Description: Toggle P1.0 by xor'ing P1.0 inside of a software loop.
4  ;   ACLK = n/a, MCLK = SMCLK = default DCO
5  ;
6  ;           MSP430G2xx1
7  ;
8  ;           /\|
9  ;           --|RST      XIN| -
10 ;                   XOUT| -
11 ;                   |
12 ;                   P1.0|-->LED
13 ;
14 ;   Based on code written by D. Dang
15 ;   Texas Instruments Inc.
16 ;   October 2010
17 ;   Built with IAR Embedded Workbench Version: 5.10
18 ;*****
19 #include "msp430g2231.h"          ; standard constants
20 LED      EQU      01h              ; LED at pin P1.0
21 DELAY    EQU      50000
22 #define   COUNTER R15              ; using R15 as counter
23 ;-----
24 ;           RSEG      CSTACK          ; creates a stack in RAM
25 ;           RSEG      CODE            ; Program goes to code
26 ;-----
27 RESET     mov.w      #SFE(CSTACK),SP ; Initialize stackpointer
28 StopWDT   mov.w      #WDTFW+WDTHOLD,&WDTCTL ; Stop WDT
29 SetupP1   bis.b      #001h,&P1DIR      ; P1.0 output
30 ;
31 Mainloop  xor.b      #LED,&P1OUT        ; Toggle P1.0
32 Wait      mov.w      #DELAY,COUNTER    ; Delay to R15
33 L1        dec.w      COUNTER            ; Decrement R15
34           jnz        L1                 ; Delay over?
35           jmp         Mainloop          ; Again
36 ;
37 ;-----
38 ;           Interrupt Vectors
39 ;-----
40 ;           RSEG      RESET            ; MSP430 segment for
41 ;           DW         RESET          ; RESET Vector
42 ;           END

```

Documentation

Constants

Declaration

Segment directives

Executable

Code

Reset vector
allocation

Introduction to some IAR directives and operators

Too many directives

Summary of assembler directives

The following table gives a summary of all the assembler directives.

Directive	Description	Section
\$	Includes a file.	Assembler control
#define	Assigns a value to a label.	C-style preprocessor
#elif	Introduces a new condition in a #if...#endif block.	C-style preprocessor
#else	Assembles instructions if a condition is false.	C-style preprocessor
#endif	Ends a #if, #ifdef, or #ifndef block.	C-style preprocessor
#error	Generates an error.	C-style preprocessor
#if	Assembles instructions if a condition is true.	C-style preprocessor
#ifdef	Assembles instructions if a symbol is defined.	C-style preprocessor
#ifndef	Assembles instructions if a symbol is undefined.	C-style preprocessor
#include	Includes a file.	C-style preprocessor
#message	Generates a message on standard output.	C-style preprocessor
#undef	Undefines a label.	C-style preprocessor
/*comment*/	C-style comment delimiter.	Assembler control
//	C++ style comment delimiter.	Assembler control
=	Assigns a permanent value local to a module.	Value assignment

continuation

=	Assigns a permanent value local to a module.	Value assignment
ALIAS	Assigns a permanent value local to a module.	Value assignment
ALIGN	Aligns the location counter by inserting zero-filled bytes.	Segment control
ALIGNRAM	Aligns the program location counter.	Segment control
ASEG	Begins an absolute segment.	Segment control
ASEGN	Begins a named absolute segment.	Segment control
ASSIGN	Assigns a temporary value.	Value assignment

Table 17: Assembler directives summary

And more ...

Directive	Description	Section
CASEOFF	Disables case sensitivity.	Assembler control
CASEON	Enables case sensitivity.	Assembler control
CPI	Specifies call frame information.	Call frame information
COL	Sets the number of columns per page.	Listing control
COMMON	Begins a common segment.	Segment control
DB	Generates 8-bit byte constants, including strings.	Data definition or allocation
DC16	Generates 16-bit word constants, including strings.	Data definition or allocation
DC32	Generates 32-bit long word constants.	Data definition or allocation
DCB	Generates 8-bit byte constants, including strings.	Data definition or allocation
DEFINE	Defines a file-wide value.	Value assignment
DF	Generates a 32-bit floating point constant.	Data definition or allocation
DL	Generates a 32-bit constant.	Data definition or allocation
.double	Generates 32-bit values in Texas Instrument's floating point format.	Data definition or allocation
DS	Allocates space for 8-bit bytes.	Data definition or allocation
DS16	Allocates space for 16-bit words.	Data definition or allocation
DS32	Allocates space for 32-bit words.	Data definition or allocation
DSB	Allocates space for 8-bit bytes.	Data definition or allocation
DW	Generates 16-bit word constants, including strings.	Data definition or allocation
ELSE	Assembles instructions if a condition is false.	Conditional assembly
ELSEIF	Specifies a new condition in an IF...ENDIF block.	Conditional assembly

Table 17: Assembler directives summary (Continued)

Directive	Description	Section
END	Terminates the assembly of the last module in a file.	Module control
ENDIF	Ends an IF block.	Conditional assembly
ENDM	Ends a macro definition.	Macro processing
ENDMOD	Terminates the assembly of the current module.	Module control
ENDR	Ends a repeat structure.	Macro processing
EQU	Assigns a permanent value local to a module.	Value assignment
EVEN	Aligns the program counter to an even address.	Segment control
EXITM	Exits prematurely from a macro.	Macro processing
EXPORT	Exports symbols to other modules.	Symbol control
EXTERN	Imports an external symbol.	Symbol control
.float	Generates 48-bit values in Texas Instrument's floating point format.	Data definition or allocation
IF	Assembles instructions if a condition is true.	Conditional assembly
IMPORT	Imports an external symbol.	Symbol control
LIBRARY	Begins a library module.	Module control
LIMIT	Checks a value against limits.	Value assignment
LOCAL	Creates symbols local to a macro.	Macro processing
LSTCND	Controls conditional assembler listing.	Listing control
LSTCOD	Controls multi-line code listing.	Listing control
LSTEXP	Controls the listing of macro generated lines.	Listing control
LSTMAC	Controls the listing of macro definitions.	Listing control
LSTOUT	Controls assembler-listing output.	Listing control
LSTPAG	Controls the formatting of output into pages.	Listing control
LSTREP	Controls the listing of lines generated by repeat directives.	Listing control
LSTXRF	Generates a cross-reference table.	Listing control
MACRO	Defines a macro.	Macro processing
MODULE	Begins a library module.	Module control
NAME	Begins a program module.	Module control
ODD	Aligns the program location counter to an odd address.	Segment control
ORG	Sets the location counter.	Segment control

Table 17: Assembler directives summary (Continued)

More

Directive	Description	Section
PAGE	Generates a new page.	Listing control
PAGESIZ	Sets the number of lines per page.	Listing control
PROGRAM	Begins a program module.	Module control
PUBLIC	Exports symbols to other modules.	Symbol control
PUBWEAK	Exports symbols to other modules, multiple definitions allowed.	Symbol control
RADIX	Sets the default base.	Assembler control
REPT	Assembles instructions a specified number of times.	Macro processing
REPTC	Repeats and substitutes characters.	Macro processing
REPTI	Repeats and substitutes strings.	Macro processing
REQUIRE	Forces a symbol to be referenced.	Symbol control
RSEG	Begins a relocatable segment.	Segment control
RTMODEL	Declares runtime model attributes.	Module control
SET	Assigns a temporary value.	Value assignment
SFRB	Creates byte-access SFR labels.	Value assignment
SFRTYPE	Specifies SFR attributes.	Value assignment
SFRW	Creates word-access SFR labels.	Value assignment
STACK	Begins a stack segment.	Segment control
VAR	Assigns a temporary value.	Value assignment

Absolute and Relocatable sections

- PLC: Program Location Counter
 - Used by linker to write in memory
 - Controlled by directives
 - **ORG** <Memory Address >
 - **ALIGN** <number>
 - Symbol \$
- An absolute section works with ORG
- **An absolute code works with ORG in all its segments**

Relocatable Segments

- Uses several PLC' s, one for each segment.
Also called Section Location Counter.
 - All initialized at 0.
 - Value is an offset value
 - ORG \$+value clears “value” bytes from current location
- Segment is placed in memory by LINKER
- RSEG Begins a relocatable segment

```

1 #include "msp430.h" ; #define controlled include file
2
3     NAME      main      ; module name
4
5     PUBLIC    main      ; make the main label visible
6                  ; outside this module
7
8     ORG       0FFFFh
9     DC16      init      ; set reset vector to 'init' label
10
11    RSEG       CSTACK    ; pre-declaration of segment
12    RSEG       CODE      ; place program in 'CODE' segment
13
14 init:  MOV     #SFE(CSTACK), SP ; set up stack
15
16
17 main:  NOP      ; main program
18
19        MOV.W   #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer
20
21        JMP     $      ; jump to current location '$'
22                  ; (endless loop)
23
24        END

```

Segment Directives

- RSEG *type*
 - *DATA* for a data segment (in RAM)
 - *CODE* for code segment (in ROM)
 - *CSTACK* for THE stack at the top of RAM
 - <Name of Segment>
 - ***STACK*** for “stack-like” segments
- SFE(XX) and SFB(XX) stand for the last and first addresses of segment XX

Memory Allocation

- **Initialized data:** data stored in memory with a specific given value at the time of compilation
- **Uninitialized data:** Memory space reserved without particular values

directivas

Size	Reserve and initialize memory	Reserve uninitialized memory
8-bit integers	DC8, DB	DS8, DS
16-bit integers	DC16, DW	DS16, DS 2
32-bit integers	DC32, DL	DS32, DS 4
64-bit integers	DC64	DS64, DS 8
32-bit floats	DF32, DF	DS32
64-bit floats	DF64	DS64

Table 27: Using data definition or allocation directives

Directive	Alias	Description	Expression restrictions
DC8	DB	Generates 8-bit constants, including strings.	
DC16	DW	Generates 16-bit constants.	
DC32	DL	Generates 32-bit constants.	
DC64		Generates 64-bit constants.	
DF32	DF	Generates 32-bit floating-point constants.	
DF64		Generates 64-bit floating-point constants.	
.double		Generates 32-bit values in Texas Instrument's floating point format.	

Table 26: Data definition or allocation directives