
Program Subroutines

Text: Chapter 7.16, Chapter 8

Program Subroutines

TABLE 8-3 Subroutine or Function Header

;	Subroutine calling sequence or invocation.
;	Subroutine name.
;	Purpose of subroutine.
;	Name of file containing the source.
;	Author.
;	Date of creation or release.
;	Input and output variables.
;	Registers modified.
;	Global data elements modified.
;	Local data elements modified.
;	Brief description of the algorithm.
;	Functions or subroutines called.

Program Subroutines

Program Element

Subroutines and functions

Program Example

```
; Subroutine "count_em( char *StringP )"
; This routine counts the characters in a string
; until the NULL character is found.
; Entry:    D register pointing to the start
; Exit:     None
; Reg Mod:  CCR
; Data Mod: Data location Counter contains the
;           number of characters
;
count_em
; Save the registers
    psbx
    tfr    d,x    ; Put the address in X
; WHILE the char is not ZERO
while_do:
    tst    0,x
    beq    found_eot
; DO count the chars
    inc    Counter
    inx
    bra    while_do

; ENDO
found_eot:
; ENDWHILE the char is not NULL
; Restore the registers
    pulx
    rts
; END subroutine "count_em( )"
```

Subroutine calls

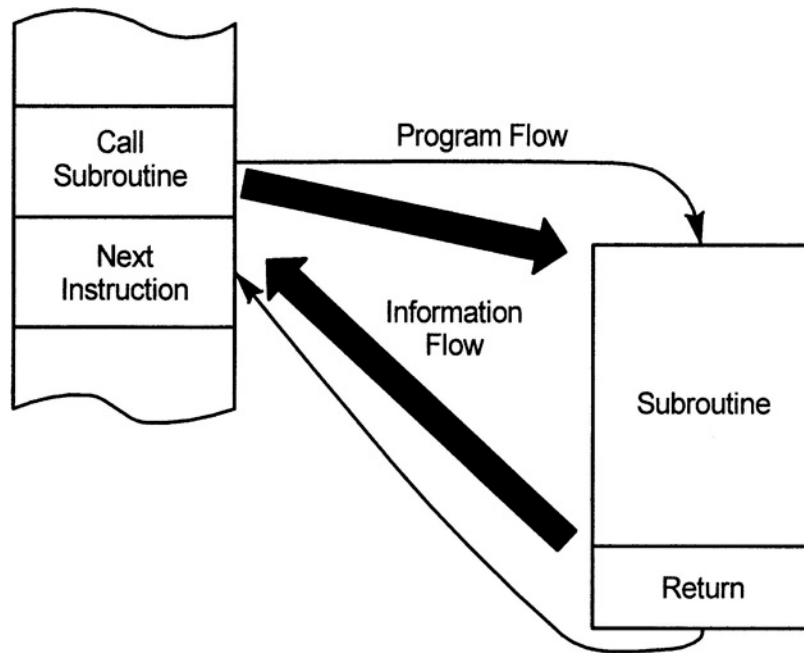


Figure 8-1 Information transfer between modules.

TABLE 8-4 Subroutine Header Comments

```
; *****  
;  
; * Subroutine Name:      SQRT  
; * Author:              F. M. Cady  
; * Date:                July 19, 1993  
; * Function:            Calculate the square root of a 16  
; *                      bit integer number.  
; * Input Registers:     D = 16 bit integer number  
; * Output Registers:    B = 8 bit integer square root  
; *                      Carry flag = 1 if input number is negative  
; *                      Carry flag = 0 if input number is positive  
; * Registers modified:  B, condition code register  
; * Global data modified: none  
; * Functions called:    none  
;  
; *****
```

Example 8-9 Passing Information in Registers

```
11          ;*****
12          ; Parameter passing between modules
13          ;*****
14          ; Passing arguments in Registers
15          ;*****
16          ; . . .
17          ; Get the input argument and pass to the
          subroutine
18 000003 B6xx xx          ldaa   Input_Arg1
19 000006 16xx xx          jsr    sub1
20          ; . . .
21          ;*****
22          ; The subroutine may be local or external
23          ; Input: A = Input Argument
24          ; Output: A = Output Argument
25          ; Registers modified: A
26          sub1:
27          ; Push the registers used on the stack
28          ; . . .
29          ; Use the input argument and/or modify it
30 000009 48              asla
31          ; Pull the registers used from the stack
32          ; Return with the modified data
33 00000A 3D              rts
34          ;*****
35          MyData: SECTION
36          ; Place variable data here
37 000000          Input_Arg1: DS.B 1
```

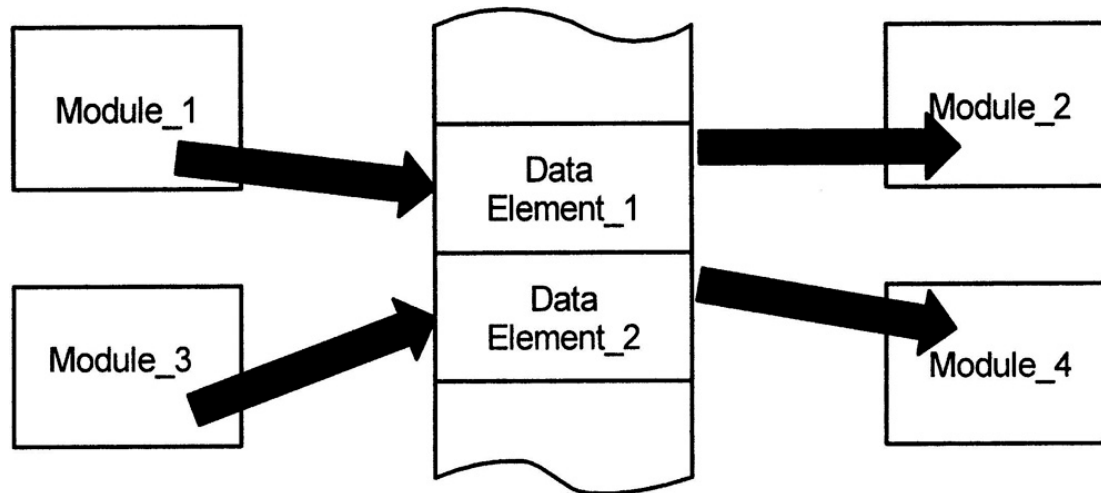


Figure 8-2 Information transfer using global data.

Example 8-10 Passing Information in Global Data Area

Metrowerks HC12-Assembler

(c) COPYRIGHT METROWERKS 1987-2003

```
Rel. Loc   Obj. code Source line
-----
1          ;*****
2          ; Passing arguments in global data
3          ;*****
4          ; Define the entry point for the main program
5          XDEF  Entry, main
6          XREF  __SEG_END_SSTACK
7          ; Define the data names that are external in
8          ; a global data buffer
9          XREF  Data_Element_1, Data_Element_2
10         XREF  Data_element_3, Data_Element_4
11         MyCode: SECTION
12         Entry:
13         main:
14         ;*****
15         ; Initialize stack pointer register
16 000000 CFxx xx         lds  #__SEG_END_SSTACK
17         ;*****
18         ; Module_1 puts data into Data_Element_1
19 000003 7Axx xx         staa Data_Element_1
20         ;*****
```

```

21             ; Module_2 gets data from Data_Element_1
22 000006 B6xx xx         ldaa  Data_Element_1
23             ;*****
24             ; Module_3 puts data into Data_Element_2
25 000009 7Axx xx         staa  Data_Element_2
26             ;*****
27             ; Module_4 gets data from Data_Element_2
28 00000C B6xx xx         ldaa  Data_Element_2
29             ;*****
30

```

Metrowerks HC12-Assembler

(c) COPYRIGHT METROWERKS 1987-2003

Rel. Loc Obj. code Source line

--- ---- -

```

1             ;*****
2             ; This is the global data definition
3             ; The data storage allocations are done
4             ; here and all data names are XDEFed to
5             ; make them globally available
6             ;*****
7             XDEF  Data_Element_1, Data_Element_2
8             XDEF  Data_Element_3, Data_Element_4
9             GlobalData: SECTION
10            ; Place variable data here
11 000000      Data_Element_1: DS.B  1
12 000001      Data_Element_2: DS.B  1
13 000002      Data_Element_3: DS.B  1
14 000003      Data_Element_4: DS.B  1
15

```

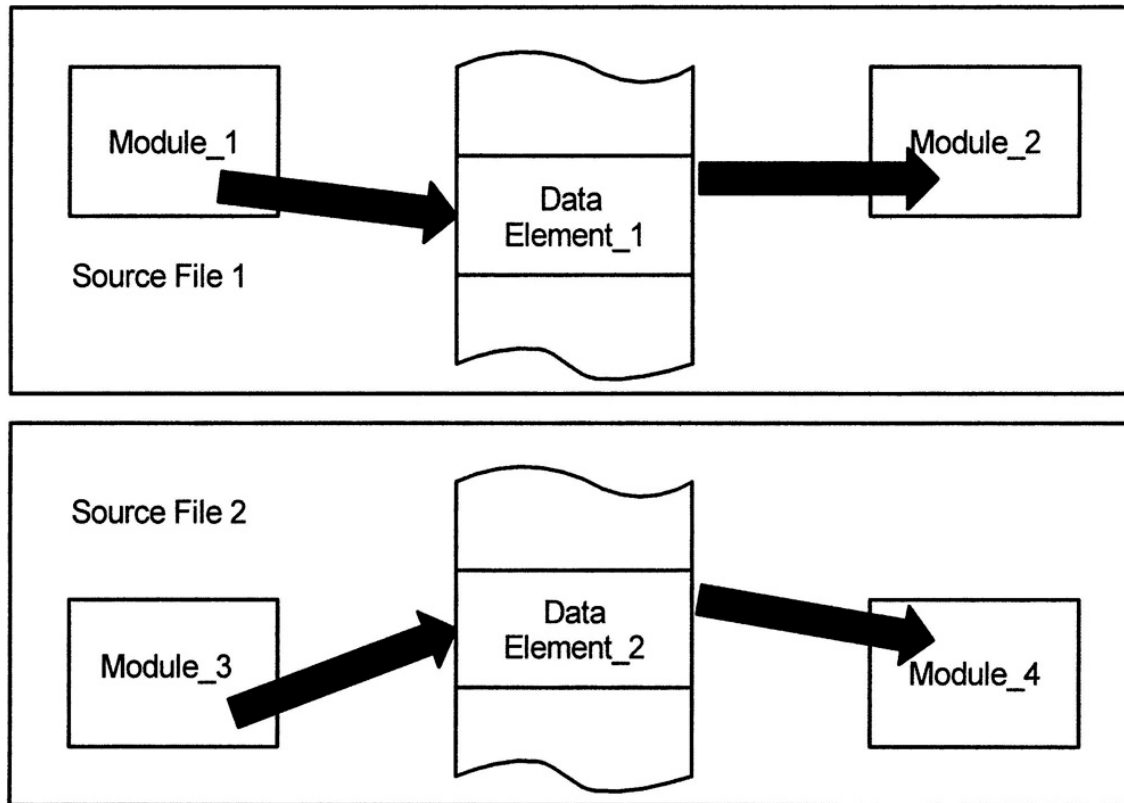


Figure 8-3 Information in local data areas.

Example 8-11 Passing Information on the Stack

Metrowerks HC12-Assembler

(c) COPYRIGHT METROWERKS 1987-2003

```
Rel. Loc   Obj. code Source line
----  ----  -
1          ;*****
2          ; Passing arguments on the stack
3          ;*****
4          ; Define the entry point for the main program
5          XDEF   Entry, main
6          XREF   __SEG_END_SSTACK
7          MyCode: SECTION
8          Entry:
9          main:
10         ;*****
```

```

11                ; Initialize stack pointer register
12 000000 CFxx xx      lds  #_SEG_END_SSTACK
13                ;*****
14                ;
15 000003 CC12 34      ldd  #$1234 ; Demo data
16 000006 CE45 67      ldx  #$4567
17                ; Put the data to be transferred on the stack
18 000009 3B          pshd   ; Two bytes transferred
19 00000A 16xx xx      jsr   subl
20                ; Get the returned data and clean up the
21                ; stack pointer
22 00000D 3A          puld   ; Two bytes returned
23                ; . . .
24                ;*****
25                ;*****
26                ; Subroutine sub
27                ; Input: 16-bit data on the stack
28                ; Output: 16-bit data on the stack
29                ; Registers modified: CCR
30                ;*****
31 0000 0002 Num_B: EQU 2 ; Number of data bytes on stack
32 0000 0004 Reg_B: EQU 4 ; Number of register bytes on
                        stack
33                subl:
34                ; Push registers used in the subroutine onto
                        the stack
35 00000E 3B          pshd
36 00000F 34          pshx
37                ; . . .
38                ; Use indexed addressing to get the data
                        passed in
39                ; from the stack
40 000010 EC86        ldd   Num_B+Reg_B,sp
41                ; . . .
42                ; Put the return data back on the stack
43 000012 CC9A BC      ldd   #$9ABC
44 000015 6C86        std   Num_B+Reg_B,sp
45                ; . . .
46                ; Pull the used registers from the stack
47 000017 30          pulx
48 000018 3A          puld
49 000019 3D          rts
50                ;*****

```

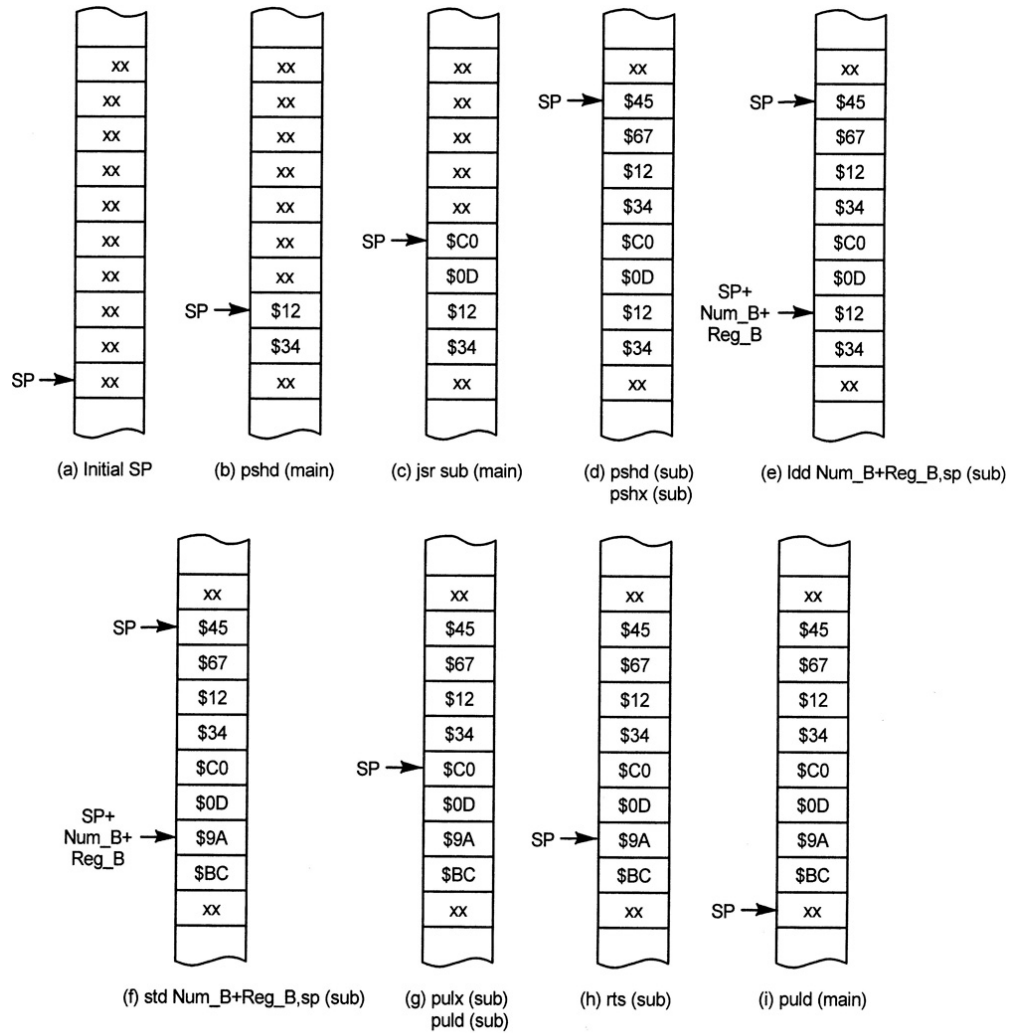


Figure 8-4 Using the stack for information transfer.