APPLICATION NOTE

Performance comparison between ST72254 and PIC16F876

by Microcontroller Division Application Team

INTRODUCTION

STMicroelectronics has developed two sets of test routines related to 8-bit and low-end 16-bit microcontroller applications to evaluate the computing performance of microcontroller cores. These routines have been implemented on ST72254 and PIC16F876 Microcontroller Units.

- The first set of routines has been written in **assembler language** to optimize their implementation and focus on core performance, without being dependent upon compiler code transformation.
- The second set tries to evaluate the performance of the two MCUs and their respective **C compilers**. This benchmark uses a C language program, representative of an automotive application. The C compilers used were from **Hiware** on the ST72 and from **Hi-Tech** on the PIC16.

The speed of the two MCUs has been compared in two ways:

- Firstly, at the maximum frequency commercially available on each MCU. this means at an external frequency of 16MHz on the ST72 and of 20MHz on the PIC16.
- Secondly, at the same current consumption level (10mA).

Table 1: Current Consumption data (taken from datasheets)

	F_{ext}	F_{CPU}	Consumption (Max)	
ST72254	**16MHz**	8MHz	**10 mA**	Run mode
PIC16C72A	20MHz	5MHz	20 mA	Run mode
PIC16C72A	**10MHz***	2.5MHz	**10 mA**	Run mode

* this value is determined by interpolation

As we can see, to reach the same consumption level on the two MCUs, the PIC's running frequency must be lowered to 10Mhz (ext.) and the ST72 can keep its maximum frequency of 16MHz (ext.).

1 ASSEMBLER TEST ROUTINES OVERVIEW

The set of test routines is made of 8 assembly programs which cover all the typical needs of an MCU application. This routine tests only the core performance and does not include any peripheral management.

Table 2. Assembler test routine overview

Abbreviated name	Full name	Description	Features stressed
string	String search	search a 16-byte string in a 128-character array in ROM	8-bit data block manipulation string manipulation
char	Character search	search a byte in a 40-byte array in ROM	8-bit data manipulation char manipulation
bubble	Bubble sort	sort of a one-dimension array of 10 16-bit integers	16-bit data manipulation integer manipulation
blkmov	Block move	move a 64-byte block from a place in RAM to another	8-bit data block manipulation block move
convert	Block translation	translate a 80-byte block in a different format	8-bit data manipulation use of a lookup table
16mul	16-bit integer multiplication	multiplication of two unsigned words giving a 32-bit result	16-bit data computation integer manipulation
shright	16-bit value right shift	shift a 16-bit value five places to the right	16-bit data manipulation bit manipulation
bitsrt	Bit manipulation	set, reset, and test of 3 bits in a 128-bit array	bit computation bit and 8-bit data manipulation

- **Notes on memory accesses used in the test routines:**

The size of the arrays manipulated by the test routines has been chosen in order to minimize RAM bank switching on the PIC16 processor. This means that **the results do not include any overhead for memory bank switching on the PIC16** MCU. But with the complexity-levels of real-world applications, the paginated memory can be a major source of time and code overhead.

For the same reason on the ST72 the data are placed in the zero page, allowing to use the short addressing mode.

2 ASSEMBLER TEST RESULTS

Table 3. Execution Speed

	At Maximum Frequency			At a Current of 10mA			Description
	PIC16 @20MHz	ST72 @16MHz	Speed Ratio ST72/PIC16*	PIC16 @10MHz	ST72 @16MHz	Speed Ratio ST72/PIC16*	
string	371 µs	282 µs	1.32	741 µs	282 µs	2.63	search a 16-byte string in a 128-character array in ROM
char	84 µs	57 µs	1.49	169 µs	57 µs	2.98	search a byte in a 40-byte array in ROM
bubble	752 µs	857 µs	0.88	1504 µs	857 µs	1.76	sort of a one-dimensional array of 10 16-bit integers
blkmov	154 µs	121 µs	1.28	308 µs	121 µs	2.55	move a 64-byte block from one place in RAM to another
convert	256 µs	241 µs	1.06	513 µs	241 µs	2.13	translate a 80-byte block into a different format
shright	6 µs	10 µs	0.65	13 µs	10 µs	1.29	shift a 16-bit value five places to the right
bitsrt	36 µs	62 µs	0.58	72 µs	62 µs	1.17	set, reset, and test of 3 bits in a 128-bit array
32div	124 µs	222 µs	0.56	248 µs	222 µs	1.12	Unsigned division of a 32-bit dividend by a 16-bit divisor
16mul	41 µs	18 µs	2.29	72 µs	18 µs	4.58	multiplication of two unsigned words giving a 32-bit result
AVG.	**203 µs**	**208 µs**	**0.98**	**406 µs**	**208 µs**	**1.95**	**Average results**

*The speed ratio is calculated as follows: (Time PIC16)/(Time ST72).
So, a number higher than 1 means that the ST72 is faster.

- **At their maximum frequency**, we can see that **the execution speed of the two MCUs is truly comparable**. Even though, in this configuration the external frequency of the PIC16 is higher (20Mhz) than that of the ST72 (16MHz).
- **At the same power consumption** level, the ST72254 is significantly better. In other words, for the same power consumption budget, **the ST72254 is nearly 2 times more powerful than the PIC16F87x**.

2.1 RESULTS ANALYSIS

- **Bit manipulation:**

The Microchip architecture is very fast for bit manipulation. Modifying one bit in a data byte can be performed in only 1 CPU cycle which means in 0.2µs@20MHz. To do the same operation the ST72 needs 5 CPU cycles (0.625µs@16MHz).

- **Memory access:**

The PIC16 is faster in direct addressing mode (1 cycle 0.2µs@20MHz compared to 3 cycles 0.375µs@16MHz). This is very useful for many parts of the application where variables are directly used, for example in loop control.

The indirect mode of the PIC16 (needed for table or string manipulation) is very slow, because its index register cannot be loaded without losing the content of the accumulator.

The ST72 with its two index registers and its wider choice of addressing modes allows very easy (and fast) data manipulation.

To summarize, when there is no need for indirect addressing, the PIC16 runs faster. But for more complex algorithms the ST72 is better (faster and easier to program).

Moreover, the test routines use only a small amount memory so, there is no problem of bank switching. But in a real (and large-sized) application, this could be a major limitation of the PIC16 architecture.

- **Use of Constant tables**

The PIC16 cannot read the content of its ROM directly, this means that, constant tables must be handled in a strange way. It needs a sub-routine call and a computed jump. This takes at least 6 CPU cycles (1.2µs@20MHz). On the ST72, the same operation needs 6 CPU cycles (0.625µs@16MHz).

Moreover, due to the need for a 8-bit computed jump, reading a constant table bigger than 256 bytes needs more time and code on the Microchip architecture.

- **Multiplication:**

Here, the comparison is easy: The ST72 has a 8-bit multiply instruction. The PIC16 does not have any. Doing multiplication entirely by software implies considerable time and code overhead.

3 C TEST ROUTINES OVERVIEW

The source program has been provided by a customer. It has 9 modules controlled by the main routine in 'FILE7.C'. It uses all instructions usually found in C language programs.

The source makes heavy use of **unsigned char, bit and table manipulations.** The modules are described in the following table. We have tried to highlight the main features of each module.

Table 4. Module description

Module	#lines	Description	Features stressed
file1	204	after evaluation of a data by a switch, manipulation of "global" data and function calls (~100)	switch/case processing function calls
file2	538	definition of functions with data manipulation, for loop, while statement, if and switch uses	loop statements arithmetic computation
file3	93	definitions of 6 functions manipulating arrays, one function doing intensive calculation	array manipulation bit calculation
file4	251	mainly load of constant tables, and manipulation of structures at the end	constant table manipulation structure use
file5	164	exactly the same file than file8.c but using switch/case statement	switch/case statement
file6	68	if processing and bitwise computation	bit manipulation and if use
file7	133	the file contains the "main", initialises data and calls functions in the other files	function calls data initialisation
file8	88	exactly the same file then file5.c	if/else statements
file9	34	signed char data computation	signed data manipulation

Note: Number of lines: 1918.

3.1 MODIFICATION OF THE SOURCE FILES

The PIC16 data memory is organised in banks which contains up to 96 bytes of RAM. But, the Hi-Tech C compiler (PIC16) does not distribute the variables automatically into these different banks. So, to make the compilation phase work, **the sources files need to be modified**. Two files have been modified: 'FILE2.C' and 'FILE2.H'. The modifications consist of using the key-words `bank1` and `bank2` to place some of the variables in the different memory banks.

For the **ST72** we have made two sets of sources. The first one called **standard does not have any modifications**. The second one called **improved** is modified to take more advantage of the use of the zero page. The only modified file is 'FILE2.H' where two lines are added:

```
#pragma DATA_SEG SHORT ZEROPAGE
#pragma DATA_SEG DEFAULT
```

In order to make heavily-used global variables directly accessible (in short addressing mode)

4 C TEST RESULTS

Table 5. Results

	ST72 (Standard)	ST72 (improved)*	PIC16*
ROM usage	7242 bytes	6849 bytes	5529 words of 14 bits
RAM usage	200 bytes	200 bytes	180 bytes
Execution time			
CPU cycle	96374	94312	62609
Time at the maximum frequency	12.04ms@16Mhz	**11.78ms@16Mhz**	**12.52ms@20Mhz**
Time at a 10mA current	12.04ms@16Mhz	11.78ms@16Mhz	**25.04ms@10Mhz**

* These source files have been modified

- In terms of execution speed, the results are the same as in the assembly routine test: **the execution speed of the two MCUs is truly comparable at their maximum frequency**.
  And for the **same power consumption budget, the ST72 is nearly 2 times more powerful** than the PIC16.

- In terms of program size: the reported size is between 20% and 30% higher on the ST72 than on the PIC16. But this is normal, because the memory of the PIC16 is organised in words of 14 bits and not in bytes (8-bit) like on the ST72.

5 COMPILATION OPTIONS

5.1 ST72 HIWARE C COMPILER

5.1.1 Compiler used:

Hicross+ compiler V5.0.3

Hicross+ Smartlinker V5.0.6

Hiware tool V5.2.6

(all these programs come from the same package)

5.1.2 Compilation options used (for all the sources files):

```
-Cc -Cni -Cu=i3 -Lasm=%n.lst -Ot -Ou -Obfv -Oc

Description:
-Cc                 Constant in ROM
-Cni                No integral promotion
-Cu=i3              Loop unrolling 3
-Lasm=%n.lst
-Ot                 Optimize for time execution
-Ou                 Optimize dead assignement
-Obfv               Optimize bitfield
-Oc                 Do common expression elimination
```

5.1.3 Project file (FILE.PRM)

```
LINK file.abs
NAMES
    file1.o+ file2.o+ file3.o+ file4.o+ file5.o+ file6.o+ file7.o+ file8.o+
file9.o+ Start07.o ansi.lib
END
STACKTOP 0x017F
SECTIONS
    ROM = READ_ONLY 0xE000 TO 0xFFDF;
    RAM = READ_WRITE 0x100 TO 0x17F;
    ZRAM = READ_WRITE 0x80 TO 0xFF;
PLACEMENT
      _ZEROPAGE,_OVERLAP INTO ZRAM;
      ZEROPAGE INTO ZRAM;
      DEFAULT_RAM INTO ZRAM,RAM;
      DEFAULT_ROM INTO ROM;
      ROM_VAR INTO ROM;
END
PRESTART OFF
VECTOR ADDRESS 0xFFFE _Startup  /* main */
```

5.1.4 Results of the compilation

5.1.4.1 Standard sources

```
-----------------------------------------------------
Segmentname   Size  Type  From  To    Name
ROM_VAR       75D   R     F4ED  FC49  ROM
FUNCS         14C1  R     E02C  F4EC  ROM
COPY          2     R     E02A  E02B  ROM
STARTUP       2A    R     E000  E029  ROM
              =1C4A (dec: 7242)
-----------------------------------------------------
Segmentname   Size  Type  From  To    Name
DEFAULT_RAM   90    R/W   B8    FD    ZRAM
                          100   149   RAM
_OVERLAP      18    R/W   A0    B7    ZRAM
_ZEROPAGE     20    R/W   80    9F    ZRAM
              =C8 (dec: 200)

ROM size: 7242
RAM size: 200
Rexecution time: 96374 cpu cycles 12.04ms (@16MHz ext.)
```

5.1.5 Improved sources

This sources files a modified to take a better advantage of the use of the zeropage.

```
-----------------------------------------------------
Segmentname   Size  Type  From  To    Name
ROM_VAR       75D   R     F4ED  FC49  ROM
FUNCS         1336  R     E02C  F361  ROM
COPY          2     R     E02A  E02B  ROM
STARTUP       2A    R     E000  E029  ROM
              =1ABF (dec: 6847)
-----------------------------------------------------
Segmentname   Size  Type  From  To    Name
DEFAULT_RAM   48    R/W   100   147   RAM
ZEROPAGE      48    R/W   B8    FF    ZRAM
_OVERLAP      18    R/W   A0    B7    ZRAM
_ZEROPAGE     20    R/W   80    9F    ZRAM
              =C8 (dec: 200)

ROM size: 6847
RAM size: 200
Execution time: 94312 cpu cycles 11.78ms (@16MHz ext.)
```

5.2 PIC16 HI-TECH C COMPILER

5.2.1 Compiler used

HI-TECH PIC C Compiler Release 7.83PL1

5.2.2 Compilation options used (for all the sources files)

To make the compilation phase work, **the sources files need to be modified**. Two files have been modified: 'FILE2.C' and 'FILE2.H'. The modifications consist of using the keywords `bank1` and `bank2` to place some of the variables in the different memory banks.

Compiler options:

- Processor PIC16F876
- Global optimization level 9
- Peephole optimization enabled
- Assembler optimization enabled

Linker options:

```
-ppowerup=0,intentry=4,intcode,intret,init ,end_init,clrtext,stringta-
ble,strings
-ABANK0=020h-07Fh
-prbss_0=BANK0,rdata_0=BANK0,idata_0=CODE
-ABANK1=0A0h-0EFh
-prbss_1=BANK1,rdata_1=BANK1,idata_1=CODE
-ABANK2=0110h-016Fh
-prbss_2=BANK2,rdata_2=BANK2,idata_2=CODE
-ABANK3=0190h-01EFh
-prbss_3=BANK3,rdata_3=BANK3,idata_3=CODE
-ACOMBANK=070h-07Fh
-ptemp=COMBANK
-ACOMBANK=070h-07Fh
-ptemp=COMBANK
-ACODE=0-7FFhx4
-ACONST=0-0FFhx32
-pconfig=2007h
-pidloc=2000h
-pfloat_text0=CODE,float_text1=CODE,float_ text2=CODE
-pfloat_text3=CODE,float_text4=CODE
-pnvram=BANK0,nvram_1=BANK1,nvram_2=BANK2, nvram_3=BANK3
-Q16F876
```

5.2.3 Results of the compilation

```
Program ROM $0000 - $0FFF $1000 (4096) words
Program ROM $1000 - $109F $00A0 (0160) words
Program ROM $1307 - $17FF $04F9 (1273) words
               $1599 (5529) words total Program ROM

Bank 0 RAM  $0020 - $0062 $0043 (67) bytes
Bank 0 RAM  $0070 - $007C $000D (13) bytes
Bank 1 RAM  $00A0 - $00ED $004E (78) bytes
Bank 3 RAM  $0190 - $01A5 $0016 (22) bytes
               $00B4 (180) bytes total RAM

ROM size: 5529 words of 14 bits
RAM size: 180 bytes
Excution time: 62609 CPU cycles, 12.52ms@20MHz
```

6 SOURCE CODE OF THE ASSEMBLY ROUTINES

You will find here, the source code of the main part of the test routine for each MCU. This will let you see that the routine has been **optimized on the two processor, in order to reach a minimum execution time**. The test routines also contain an initialisation part which is not taken into count for the speed measurement.

6.1 BLKMOV PIC16

```
test:
        bsf             PORTB,0         ;Set I/O pin : beginning of measurement
        movlw           bsize
        movwf           i
        bsf             STATUS,IRP      ;Choose bank 2 or 3 for indirect addressing
mainlp:
        movlw           addrs-1
        addwf           i,W
        movwf           FSR
        movfw           INDF            ;read one byte of the source
        movwf           tmp
        movlw           addrd-addrs     ;Set the address of the byte in the destination block
        addwf           FSR,F
        movfw           tmp
        movwf           INDF            ;write one byte in the destination
        decfsz          i,F
        goto            mainlp

        bcf             PORTB,0         ;Reset I/O pin : end of measurement
        return
```

6.2 BLKMOV ST72

```
.TEST
        bset            pbdr, #0        ;Set I/O pin : beginning of measurement
        ld              x, #bsize
.mainlp
        ld              a, (addrs,x)    ;Read the source byte
        ld              (addrd,x), a    ;Write it to the destination
        dec             x
        jrne mainlp

        bres            pbdr, #0        ;Reset I/O pin : end of measurement
        ret
```

6.3 STRING PIC16

```
test:
  bsf   PORTB,0      ;Set I/O pin: beginning
  clrf  x            ;of measurement
  movlw 0
  call  table_string;Read the first
  movwf strstart     ;char of the string

mainlp:              ;Find the first character
  movfw x
  call  table_array
  incf  x,f
  andlw 0xFF         ;Compare W to zero
  bz    bad
  subwf strstart,W   ;Compare current char
  bnz   mainlp       ;with the first of the
  movfw x            ;string
  movwf x2
  clrf  y
allcp:               ;Follow the comparison
  incf  y,F          ;after match with the
  movfw x2           ;first character
  call  table_array
  andlw 0xFF         ;Compare W to zero
  bz    bad
  movwf tmp
  movfw y
  call  table_string
  andlw 0xFF         ;Compare W to zero
  bz    fin
  subwf tmp,W        ;Comparison between the
  bnz   mainlp       ;two characters
  incf  x2,F
  goto  allcp
bad:                 ;If string not found
  clrf x
fin:                 ;Result is available in X
  bcf   PORTB,0      ;Reset I/O pin: end of
  return             ;measurement
;**************************************
table_array:
  addwf PCL,F
array:
  dt    “xxxxxxxxpatterxxxxxxxxxxxxxxxxxxxx”
  dt    “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”
  dt    “xxxxxxxxxxxxxpattern is here!xxxxxx”
  dt    “xxxxxxxxxxxxxxxxxxxxxxxx”,0
table_string:
  addwf PCL,F
string:
dt ”pattern is here!”,0
```

6.4 STRING ST72

```
.TEST
  bset  pbdr, #0     ;Set I/O pin : beginning
  ld    x, #$ff      ;of measurement
.mainlp
  inc   x
  ld    a, (array,x)
  jreq  bad          ;Check for the end
  cp    a, string    ;Compare the current
  jrne  mainlp       ; char with the first
  push  x            ; of the string
  ld    y, #1
.allcp               ;Follow the comparison
  inc   x            ; after match with the
  ld    a, (array,x) ; first character
  jreq  bad          ;Check for the end
  ld    a, (string,y)
  jreq  end          ;End of the string ?
  cp    a, (array,x)
  jrne  save
  inc   y
  jra   allcp
.bad                 ;If string not found
  clr   x
.end
  pop   x            ;Result is available in X
  bres  pbdr, #0     ;Reset I/O pin:
  ret                ;end of measurement
.save
  pop   x
  jra   mainlp
;**************************************
.array
  dc.b  “xxxxxxxxpatterxxxxxxxxxxxxxxxxxxxx”
  dc.b  “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”
  dc.b  “xxxxxxxxxxxxxpattern is here!xxxxxx”
  dc.b  “xxxxxxxxxxxxxxxxxxxxxxxx”,0
.string
  dc.b  “pattern is here!”,0;searched string
```

6.5 CHAR PIC16

```
test:
  bsf    PORTB,0      ;Set I/O pin : beginning of measurement
  movlw  asize
  movwf  idx          ;Set the index to end of the array
  call   character
  movwf  ch           ;read the character
mainlp:
  movfw  idx
  call   table_array  ;read one character in the array
  subwf  ch,W         ;compare it with the searched one
  bz     fin
  decfszidx,F         ;Do it until the beginning of the array as been reached
  goto   mainlp
fin:                  ;starting position in X if not found X=0
  bcf    PORTB,0      ;Reset I/O pin : end of measurement
  return

table_array:
  addwf  PCL,F
  nop                 ;In order to let idx varies between max and 1 (faster)
array:                ;40 bytes array
   dt "--------o------------------------------"
character: dt "o      ";searched char
```

6.6 CHAR ST72

```
.TEST
  bset   pbdr, #0     ;Set I/O pin : beginning of measurement
  ld     x, #asize    ;Set X to point at the end of the array
  ld     a, character ;Store in A the searched character
.mainlp
  cp     a,({array-1},x);compare the current character to the searched one
  jreq   end          ;If character found, X contain the position
  dec    x            ;go to the previous character
  jrne   mainlp
.end                  ;if not found X=0
  bres   pbdr, #0     ;Reset I/O pin : end of measurement
  ret

.array                ;40 bytes array
  dc.b "--------o------------------------------"
.character dc.b "o    ";searched char
```

6.7 BUBBLE PIC16

```
TEST:
  bsf   PORTB,0;Beginning of measurement
  bsf   flag,0;flag=1

mainlp:
  movf  flag,F
  bz    fin
  clrf  flag

  movlw num-1
  movwf idx
  movlw array
  movwf x     ;Point on the first element
  movlw array+2
  movwf y     ;Point on the second element
intlp:
  movfw x
  movwf FSR
  movfw INDF  ;Store in bufh the msb of the
  movwf bufh  ; element pointed by x
  movfw y
  movwf FSR   ;Load in W the msb of the
  movfw INDF  ; element pointed by y
  subwf bufh,W;compare msbx and msby
  bz    lsb   ;if equal compare the lsb
  bnc   noxch
xch:          ;At this point: msbX is in
  movfw INDF  ;bufh and FSR is set on msbY
  movwf tmph  ;Store msbY in tmph
  incf  FSR,F ;Point lsb Y
  movfw INDF
  movwf tmpl  ;Store lsbY in tmpl
  movfw x
  movwf FSR
  movfw tmph  ;Put msbY in the location of msbX
  movwf INDF
  incf  FSR,F ;Point on lsbX
  movfw INDF
  movwf bufl  ;Store lsbX in bufl
  movfw tmpl
  movwf INDF  ;Put lsbY in the location of lsbX
  movfw y
  movwf FSR   ;Point to msbY
  movfw bufh
  movwf INDF  ;Put msbX in the location of msbY
  incf  FSR,F ;Point on lsbY
  movfw bufl
  movwf INDF  ;Put msbY in the location of msbX
  bsf   flag,0;Set flag to continue sorting
noxch:
  movlw 2
  addwf x,F   ;Set X on the next element
  addwf y,F   ;Set X on the next element
  decfszidx,F
  goto  intlp
  goto  mainlp
lsb:          ;Comparison of the LSB
  incf  x,W
  movwf FSR   ;Point to lsbX
  movfw INDF
  movwf bufl  ;Store lsbX in bufl
  incf  y,W   ;Point to lsbY
  movwf FSR
  movfw INDF  ;Store lsbY in W
  decf  FSR,F ;FSR point on msbY (for xch:)
  subwf bufl,W;Compare lsbX and lsbY
  bc    xch   ;Exchange if needed
  goto  noxch
fin:
  bcf   PORTB,0;End of measurement
  return
```

6.8 BUBBLE ST72

```
.TEST
 bset   pbdr, #0 ;Beginning of measurement
 bset   flag,#0      ;flag=1
.mainlp
 btjf   flag,#0,end  ;IF flag=0 it is finish
 bres   flag,#0
 ld     x, #0
 ld     y, #2
.intlp
 ld     a, (array,x)
 cp     a, (array,y)  ;compare msbx and msby
 jreq   lsb           ;if equal compare the lsb
 jrult  noxch
.xch                  ;Exchange two words
 ld     a, (array,x)
 ld     buf, a        ;save msbX in buf
 ld     a, (array,y)  ;Store msbY in
 ld     (array,x), a  ; the location of msbX
 ld     a, buf        ;Store msbX in
 ld     (array,y), a  ; the location of msbY
 ld     a, ({array+1},x)
 ld     buf, a        ;save lsbX in buf
 ld a, ({array+1},y)  ;Store lsbY in
 ld ({array+1},x), a  ; the location of lsbX
 ld     a, buf        ;Store lsbX in
 ld ({array+1},y),a   ; the location of lsbY
 bset   flag,#0       ;Set flag to continue
.noxch
 inc    y
 inc    y             ;make x and y point
 inc    x             ; to the next integer
 inc    x             ; (16-bit)
 cp     x, #{last - 2};Check for end of array
 jrne   intlp
 jra    mainlp
.lsb                  ;Compare the two lsb
 ld     a,({array+1},x)
 cp a,({array+1},y)   ;Compare lsbY and lsbY
 jrugt  xch
 jra    noxch
.end
 bres   pbdr, #0      ;End of measurement
 ret
```

6.9 CONVERT PIC16

```
test:
       bsf    PORTB,0             ;Set I/O pin : beginning of measurement
       movlw  bsize
       movwf  i
mainlp:
       movlw  srcblock-1
       addwf  i,W
       movwf  FSR
       movfw  INDF                ;read the current value in the source block
       addlw  table               ;Put in W the index in the lookup table
       movwf  FSR
       movfw  INDF
       movwf  tmp                 ;Put in tmp the value in the new format
       movlw  dstblock-1
       addwf  i,W
       movwf  FSR                 ;point to the destination block
       movfw  tmp
       movwf  INDF                ;Write the new value in the destination block
       decfsz i,F
       goto   mainlp

       bcf    PORTB,0             ;Reset I/O Pin : end of the measurement
       return
```

6.10 CONVERT ST72

```
.TEST
       bset   pbdr, #0            ;Set I/O pin : beginning of measurement
       ld     x, #bsize           ;Set the index register to the end of the srcblock
.mainlp
       ld     a,({srcblock-1},x)  ;load the current byte in the source block
       ld     y, a
       ld     a, (table,y)        ;Load the corresponding value from the lookup table
       ld     ({dstblock-1},x), a ;Store it in the destination block
       dec    x                   ;go to previous byte
       jrne   mainlp              ;until the first is reached

       bres   pbdr, #0            ;Reset I/O pin : end of measurement
       ret
```

6.11 SHRIGHT PIC16

```
TEST:
        bsf     PORTB,0             ;Set I/O pin : beginning of measurement
        movlw   00
        movwf   h                   ;Set the high byte of the word
        movlw   0x40
        movwf   l                   ;Set the low byte of the word
        movlw   5
        movwf   idx                 ;Set the number of bit to rotate
        bcf     STATUS,C            ;Reset the carry
mainlp:
        rrf     h,F
        rrf     l,F
        decfsz  idx,F
        goto    mainlp

        bcf     PORTB,0             ;Reset I/O pin : end of measurement
        return
```

6.12 SHRIGHT ST72

```
.TEST
        bset    pbdr, #0            ;Set I/O pin : beginning of measurement
        ld      a, #5               ;Set the number of bit to rotate
        ld      x, #0               ;Set the high byte of the word
        ld      y, #$40             ;Set the low byte of the word
.mainlp
        rrc     x
        rrc     y
        dec     a
        jrne    mainlp

        bres    pbdr, #0            ;Reset I/O pin : end of measurement
        ret
```

6.13 BITSRT PIC16

```
test:
  bsf    PORTB,0      ;Beginning of measurement
  movlw  .10
  movwf  idx          ;bit10 -> bit 2, byte 1
  call setbit
  call resetbit
  call testbit
  movlw  .13          ;bit13 -> bit 5, byte 1
  movwf  idx
  call setbit
  call resetbit
  call testbit
  movlw  .123         ;bit 123 -> bit 3, byte 15
  movwf  idx
  call setbit
  call resetbit
  call testbit
  bcf    PORTB,0      ;End of measurement
  return
setbit:
  movfw  idx
  andlw  b'111'
  movwf  offset       ;bit number
  bcf    STATUS,C
  rlf    idx,w
  andlw  H'F0'
  movwf  FSR
  swapf  FSR,F        ;byte number
  movlw  flags
  addwf  FSR,F        ;Address of the Byte.
  rlf    offset,W     ;Do W=offset*2 (Carry=0)
  addwf  PCL,F        ;Jump into the following
  bsf    INDF,0       ; table
  return
  bsf    INDF,1
  return
  bsf    INDF,2
  return
  bsf    INDF,3
  return
  bsf    INDF,4
  return
  bsf    INDF,5
  return
  bsf    INDF,6
  return
  bsf    INDF,7
  return
```

```
resetbit:
  movfw  idx
  andlw  b'111'
  movwf  offset
  rlf    idx,w
  andlw  H'F0'
  movwf  FSR
  swapf  FSR,F
  movlw  flags
  addwf  FSR,F
  bcf    STATUS,C
  rlf    offset,W
  addwf  PCL,F
  bcf    INDF,0
  return
  bcf    INDF,1
  return
  bcf    INDF,2
  return
  bcf    INDF,3
  return
  bcf    INDF,4
  return
  bcf    INDF,5
  return
  bcf    INDF,6
  return
  bcf    INDF,7
  return
testbit:
  movfw  idx
  andlw  b'111'
  movwf  offset
  rlf    idx,w
  andlw  H'F0'
  movwf  FSR
  swapf  FSR,F
  movlw  flags
  addwf  FSR,F
  bcf    STATUS,C
  swapf  offset,f
  rrf    offset,W
  addwf  PCL,F
  btfsc  INDF,0
  return
  bsf    INDF,0
  bsf    STATUS,Z
  return
  nop                ;The 3 NOPs in the table
  nop                ;allow a good speed
  nop                ;optimization
```

```
btfsc INDF,1
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop

btfsc INDF,2
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop

btfsc INDF,3
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop

btfsc INDF,4
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop
btfsc INDF,5
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop



btfsc INDF,6
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop
btfsc INDF,7
return
bsf   INDF,0
bsf   STATUS,Z
return
nop
nop
nop
```

6.14 BITSRT ST72

```
.TEST
 bset   pbdr, #0     ;Beginning of measurement
 ld     a, #10
 ld     idx,a
 call setbit
 call resetbit
 call testbit
 ld     a, #13
 ld     idx,a
 call setbit
 call resetbit
 call testbit
 ld     a, #123
 ld     idx,a
 call setbit
 call resetbit
 call testbit
 bres   pbdr, #0     ;End of measurement
 ret
.setbit
 ld     a,idx
 sla    a
 and    a,#$F0
 swap   a            ;Number of the byte
 add    a, #flags    ;Address of the byte
 ld     x,a

 ld     a,idx
 and    a,#%111
 ld     y,a          ;Bit number

 ld     a,(bittbl,y)
 or     a,(x)        ;Set the bit in the array
 ld     (x),a
 ret

.resetbit
 ld     a,idx
 sla    a
 and    a,#$F0
 swap   a
 ld     x,a
 ld     a,idx
 and    a,#%111
 ld     y,a

 ld     a,(bittblinv,y)
 and    a,(x)        ;Reset bit in the array
 ld     (x),a
 ret

.testbit
 ld     a,idx
 sla    a
 and    a,#$F0
 swap   a
 add    a, #flags
 ld     x,a
 ld     a,idx
 and    a,#%111
 ld     y,a

 ld     a,(bittbl,y)
 ld     y,a          ;Save the value
 and    a,(x)        ;Test if the bit is Reset
 jrne   isone
 ld     a,y
 or     a,(x)
 ld     (x),a        ;Set bit in the array
 and    a,#0         ;set the Zero flag
.isone
 ret
.bittbl
 dc.b   $01,$02,$04,$08,$10,$20,$40,$80
.bittblinv
 dc.b   $FE,$FD,$FB,$F7,$EF,$DF,$BF,$7F
```

6.15 32DIV PIC16

```
test:
  bsf    PORTB,0      ;Beginning of measurement
  movlw  d'32'
  movwf  cpt
  clrf   reste
  clrf   reste+1
  bcf    STATUS,C
execute:
  rlf    dividend+3,F;Shift  left dividend
  rlf    dividend+2,F
  rlf    dividend+1,F
  rlf    dividend,F
  rlf    reste+1,F
  rlf    reste,F

        ; Test if reste is >= to the divisor
  bc     dividendlsgreater ; Reste is 17 bits
  movfw  reste
  subwf  divisor,W
  bnc    dividendlsgreater
  bnz    nosubstract
  movfw  reste+1
  subwf  divisor+1,W
  bc     nosubstract
dividendlsgreater:  ;Substract divisor
  movfw  divisor+1    ;from left dividend
  subwf  reste+1,F
  btfss  STATUS,C
  decf   reste,F
  movfw  divisor
  subwf  reste,F
  bsf    STATUS,C
  decfszcpt,F         ; Decrement loop counter
  goto   execute       ; if cpt = 0 then exit
  goto   out
nosubstract:
  bcf    STATUS,C
  decfszcpt,F         ; Decrement loop counter
  goto   execute       ; if cpt = 0 then exit
out:
  rlf    dividend+3,F
  rlf    dividend+2,F
  rlf    dividend+1,F
  rlf    dividend,F
  bcf    PORTB,0      ;End of measurement
  return
```

6.16 32DIV ST72

```
.TEST
  bset   pbdr, #0     ;Beginning of measurement
  ld X,#32
  clr    {reste}
  clr    {reste+1}
  rcf
.execute
  rlc    {dividend+3}     ;Shift left dividend
  rlc    {dividend+2}
  rlc    {dividend+1}
  rlc    {dividend}
  rlc    {reste+1}
  rlc    {reste}
    ; Test if reste is >= to the divisor
  jrc    dividendlsgreater ; Reste is 17 bits
  ld     A,{divisor}
  cp     A,{reste}
  jrugt  nosubstract
  jrne   dividendlsgreater
  ld     A,{divisor+1}
  cp     A,{reste+1}
  jrugt  nosubstract
.dividendlsgreater        ; Substract divisor
  ld     A,{reste+1}
  sub    A,{divisor+1}
  ld     {reste+1},A
  ld     A,{reste}
  sbc    A,{divisor}
  ld     {reste},A
  scf
  dec    X            ; Decrement loop counter
  jrne   execute      ; if X = 0 then exit
  jra    out
.nosubstract
  rcf
  dec    X            ; Decrement loop counter
  jrne   execute      ; if X = 0 then exit
.out
  rlc    {dividend+3}
  rlc    {dividend+2}
  rlc    {dividend+1}
  rlc    {dividend}
  bres   pbdr, #0     ;End of measurement
  ret
```

6.17 16MUL PIC16

```
test:
  bsf    PORTB,0      ;Beginning of measurement
;*****************************************
; Double Precision Multiply ( 16x16 -> 32 )
; ( ACCb*ACCa -> ACCb,ACCc )
; 32 bit output
;with   high word in ACCb ( ACCbHI,ACCbLO )
;and    low word in ACCc ( ACCcHI,ACCcLO ).

  movlw  .16       ; for 16 shifts
  movwf  temp
  movf   ACCbHI,W      ;move ACCb to ACCd
  movwf  ACCdHI
  movf   ACCbLO,W
  movwf  ACCdLO
  clrf   ACCbHI
  clrf   ACCbLO

mloop:
  rrf    ACCdHI, F    ;rotate d right
  rrf    ACCdLO, F
  btfss  STATUS,C   ;need to add?
  goto   no_add

  ;------------------------
  ; Addition ( ACCb + ACCa -> ACCb )
  movf   ACCaLO,W
  addwf  ACCbLO, F    ;add lsb
  btfsc  STATUS,C     ;add in carry
  incf   ACCbHI, F
  movf   ACCaHI,W
  addwf  ACCbHI, F    ;add msb
  movlw  0
  ;------------------------

no_add:
  rrf    ACCbHI, F
  rrf    ACCbLO, F
  rrf    ACCcHI, F
  rrf    ACCcLO, F
  decfsz temp, F      ;Until all bits checked
  goto   mloop
;*****************************************
  bcf    PORTB,0      ;End of measurement
  return
```

Note: The source code is taken from a Microchip application note.

6.18 16MUL ST72

```
.TEST
  bset   pbdr, #0     ;Beginning of measurement
  ld     x, op2
  ld     a, op1
  mul    x, a
  ld     res, x
  ld     {res + 1}, a
  ld     x, {op1 + 1}
  ld     a, {op2 + 1}
  mul    x, a
  ld     {res + 2}, x
  ld     {res + 3}, a
  ld     x, op1
  ld     a, {op2 + 1}
  mul    x, a
  add    a, {res + 2}
  ld     {res + 2}, a
  ld     a, x
  adc a, {res + 1}
  ld     {res + 1}, a
  ld     a, res
  adc    a, #0
  ld     res, a
  ld     x, op2
  ld     a, {op1 + 1}
  mul    x, a
  add    a, {res + 2}
  ld     {res + 2}, a
  ld     a, x
  adc    a, {res + 1}
  ld     {res + 1}, a
  ld     a, res
  adc    a, #0
  ld     res, a
  bres   pbdr, #0     ;End of measurement

ret
```

7 SOURCE CODE OF THE C ROUTINES

7.1 COMMNLIB.H

```
/*****************************************************************************
* Module:       commnlib.h
*****************************************************************************/
#ifndef common_lib_h
  #define common_lib_h
  /**** Constant Definitions ***/
  #define TRUE  1
  #define FALSE  0
  #define OK   1
  #define FAULT  0
  /**** Macro Definitions ***/
  /*-------------------------*/
  /* Bit manipulation macros */
  /*-------------------------*/
  #define SETBIT(x,y) (x) |= (y)
  #define CLRBIT(x,y) (x) &= ~(y)
  #define TOGLBIT(x,y) ((x) ^= (y))
  #define TESTBIT(x,y) ((x) & (y))
#endif                  /* common_lib_h */
```

7.2 FILE1.H

```
#ifndef FILE1_H
#define FILE1_H
#ifdef FILE1_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
enum event_type
{
 /** System Events */
 EVENT_ONE,
 EVENT_TWO,
 EVENT_THREE,
 EVENT_FOUR,
 EVENT_FIVE,
 EVENT_SIX,
 EVENT_SEVEN,
 EVENT_EIGHT,
 EVENT_NINE,
 EVENT_TEN,
 EVENT_ELEVEN,
 EVENT_TWELVE,
 EVENT_THIRTEEN,
 EVENT_FOURTEEN,
 EVENT_FIFTEEN,
 EVENT_SIXTEEN
};
```

```
/*****************************************************************************
            Private Function Prototypes
 *****************************************************************************/
/* announce_event connects event announcements to registered responses */
EXTERN_pfx void announce_event(enum event_type newevent, unsigned char data);
EXTERN_pfx void announce_event1(enum event_type newevent);

#undef EXTERN_pfx
```

7.3 FILE1.C

```
/****************************  file1.c  *********************************/
#define FILE1_C
#include  ”commnlib.h”
#include  ”file1.h”
#include  ”file2.h”
#include  ”file3.h”
#ifdef PWR_ANTENNA_SUPPORT
#include  ”antenna.h”
#endif
void announce_event(enum event_type newevent, unsigned char data)
{
 unsigned char temp_var1;
 unsigned char temp_var2;
 unsigned int temp_var3;
 unsigned int temp_var4;

 temp_var1 = 0;
 temp_var2 = 0;
 temp_var3 = 0;
 temp_var4 = 0;
 switch (newevent)
 {
 case EVENT_ONE:
  file2_function1(data);
  SETBIT(global_1, 0x01);
  break;
 case EVENT_TWO:
  file2_function2(data);
  CLRBIT(global_18, 0x02);
  break;
 case EVENT_THREE:
  file2_function3(data);
  SETBIT(global_19, 0x04);
  break;
 case EVENT_FOUR:
  file2_function3(data);
  file2_function4();
  CLRBIT(global_4, 0x10);
  break;
 case EVENT_FIVE:
  file2_function3(data);
  file2_function4();
  CLRBIT(global_15, 0x20);
```

```
 case EVENT_SIX:
  file2_function3(data);
  file2_function4();
  file2_function5();
  file2_function6();
  file2_function7();
  file2_function8();
  file2_function9();
  file2_function10();
  file2_function11(1);
#ifdef PWR_ANTENNA_SUPPORT
  global_19 = file2_function12(3);
  global_18 = file2_function13(2);
  file2_function14(7);
  file2_function15();
  file2_function16();
#endif
  file2_function17();
  file2_function18();
  file2_function19();
  file2_function20();
  file2_function21(sizeof sample_array, &sample_array[0]);
  file2_function22(0, AUX_ON);
  file2_function23(PARAM3);
  SETBIT(global_12, 0x01);
  break;
 case EVENT_SEVEN:
  file2_function3(data);
  file2_function4();
  file2_function5();
  file2_function6();
  file2_function7();
  file2_function8();
  file2_function9();
  file2_function10();
  file2_function11(4);
  global_7 = file2_function12(2);
  global_9 = file2_function13(3);
  global_4 = file2_function14(9);
  file2_function15();
  file2_function16();
  file2_function17();
  file2_function18();
  file2_function19();
  file2_function20();
  file2_function21(sizeof sample_array, &sample_array[0]);
  file2_function22(0, AUX_ON);
  file2_function23(PARAM7);
  global_3 = 99;
  break;
 case EVENT_EIGHT:
  file2_function3(data);
  file2_function4();
  file2_function5();
  file2_function6();
```

```
  file2_function7();
  file2_function8();
  file2_function9();
  file2_function10();
  file2_function11(9);
  global_5 = file2_function12(2);
  global_7 = file2_function13(1);
  file2_function14(4);
  file2_function15();
  file2_function16();
  file2_function17();
  file2_function18();
  file2_function19();
  file2_function20();
  file2_function21(sizeof sample_array, &sample_array[0]);
  file2_function22(10, 0);
  file2_function23(PARAM10);
  break;
 case EVENT_NINE:
  file2_function3(data);
  file2_function4();
  file2_function5();
  file2_function6();
  file2_function7();
  file2_function8();
  file2_function9();
  file2_function10();
  file2_function11(2);
  sample_array[4] = file2_function12(2);
  sample_array[3] = file2_function13(1);
  global_4 = file2_function14(5);
  file2_function15();
  file2_function16();
  file2_function17();
  file2_function18();
  file2_function19();
  file2_function20();
  file2_function22(20, 5);
  file2_function23(PARAM13);
  global_4 = 65535;
  break;
 case EVENT_TEN:
  file2_function11(2);
  temp_var2 = file2_function12(22);
  temp_var3 = file2_function13(1);
  temp_var4 = file2_function14(100);
  global_13 = temp_var4;
  global_12 = temp_var3;
  global_11 = temp_var2;
  global_4 = temp_var4;
  file2_function22(2220, 4);
  break;
 case EVENT_ELEVEN:
  file2_function3(data);
  file2_function4();
```

```
  break;
 case EVENT_TWELVE:
  file2_function1(4);
  file2_function2(3);
  break;
 case EVENT_THIRTEEN:
  file2_function5();
  file2_function6();
  break;
 case EVENT_FOURTEEN:
 case EVENT_FIFTEEN:
  file2_function6();
  file2_function7();
  break;
 default:
  file2_function3(data);
  file2_function4();
  file2_function5();
  file2_function6();
  file2_function7();
  file2_function8();
  file2_function9();
  file2_function10();
  file2_function11(7);
  file2_function12(115);
  file2_function13(1);
  if (global_5 > 9)
  {
   file2_function14(5);
   file2_function15();
   file2_function16();
  }
  else if ((global_4) && (global_3 & 0x01))
  {
   file2_function17();
   file2_function18();
   file2_function19();
   file2_function20();
  }
  file2_function23(PARAM13);
  break;
 }
 return;
}
void announce_event1(enum event_type newevent)
{
 announce_event(newevent, 0);
 return;
}
```

7.4 FILE2.H

```
/*****************************************************************************
            Include Files
 *****************************************************************************/
/*****************************************************************************/
#ifndef FILE2_H
#define FILE2_H
#ifdef FILE2_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
/*****************************************************************************
            Public Constant Definitions
 *****************************************************************************/
#define PARAM1 0
#define PARAM2 1
#define PARAM3 10
#define PARAM4 20
#define PARAM5 30
#define PARAM6 40
#define PARAM7 50
#define PARAM8 60
#define PARAM9 70
#define PARAM10 80
#define PARAM11 90
#define PARAM12 100
#define PARAM13 110
#define PARAM14 120
#define PARAM15 130
#define   AUX_ON          0x01
#define   AUX_OFF         0x02
#define   AUX_RESET        0x03
#define   REQUEST_STATUS      0x04
#define   AUX_DIAG         0x05
#define   AUX_NULL_STATE      0x0F
#define   AUX_SW_VERSION      0x10
EXTERN_pfx unsigned char global_1;
EXTERN_pfx unsigned char global_2;
EXTERN_pfx unsigned char global_3;
EXTERN_pfx unsigned int global_4;
EXTERN_pfx unsigned char global_5;
EXTERN_pfx unsigned char global_6;
EXTERN_pfx unsigned char global_7;
EXTERN_pfx unsigned char global_8;
EXTERN_pfx unsigned char global_9;
EXTERN_pfx unsigned char global_10;
EXTERN_pfx unsigned char global_11;
EXTERN_pfx unsigned char global_12;
EXTERN_pfx unsigned char global_13;
EXTERN_pfx unsigned char global_14;
EXTERN_pfx unsigned char global_15;
EXTERN_pfx unsigned char global_16;
EXTERN_pfx unsigned char global_17;
```

```
EXTERN_pfx unsigned char global_18;
EXTERN_pfx unsigned char global_19;
EXTERN_pfx unsigned char global_20;
EXTERN_pfx unsigned char global_21;
EXTERN_pfx unsigned char sample_array[25];
EXTERN_pfx unsigned char sample_array2[25];
EXTERN_pfx void file2_function1(unsigned char dummy_param);
EXTERN_pfx void file2_function2(unsigned char dummy_param);
EXTERN_pfx void file2_function3(unsigned char dummy_param);
EXTERN_pfx void file2_function4(void);
EXTERN_pfx unsigned char file2_function5(void);
EXTERN_pfx void file2_function6(void);
EXTERN_pfx void file2_function7(void);
EXTERN_pfx void file2_function8(void);
EXTERN_pfx void file2_function9(void);
EXTERN_pfx void file2_function10(void);
EXTERN_pfx unsigned char file2_function11(unsigned char dummy_param);
EXTERN_pfx unsigned char file2_function12(unsigned char dummy_param);
EXTERN_pfx unsigned int file2_function13(unsigned char dummy_param);
EXTERN_pfx unsigned int file2_function14(unsigned char dummy_param);
EXTERN_pfx void file2_function15(void);
EXTERN_pfx void file2_function16(void);
EXTERN_pfx void file2_function17(void);
EXTERN_pfx void file2_function18(void);
EXTERN_pfx void file2_function19(void);
EXTERN_pfx void file2_function20(void);
EXTERN_pfx void file2_function21(unsigned char msg_size, unsigned char msg[]);
EXTERN_pfx void file2_function22(unsigned int aux_comm_timeout, unsigned char retry_txmsg);
EXTERN_pfx void file2_function23(unsigned char dummy_param);

#undef EXTERN_pfx
```

7.5 FILE2.C

```
#define FILE2_C
/**** Include Files ***/
#include  ”commnlib.h”
#include  ”file1.h”
#include  ”file2.h”
#include  ”file3.h”
#ifdef PWR_ANTENNA_SUPPORT
#include  ”antenna.h”
#endif
#define SCP_SPEED_MSG  0x08
#define SCP_OPT     0x01
#define DSP_OPT     0x02
#define TAPE_OPT    0x04
#define RDS_OPT     0x08
#define CLOCK_OPT   0x10
#define PHONE_OPT   0x20
#define CDDJ_OPT    0x40
#define sample_DISABLED  0x01
#define MAX_SPEED_DETECT    241
#define MIN_SPEED_DETECT    16
```

```
#define MAX_COMPENSATION_LEVEL20
#define sample_PLUS   0x20
#define AUX_INIT_WAIT    0
#define AUX_READY      1
#define AUX_DIAG_MODE    2
#define AUX_RESET_WAIT    3
#define AUX_SHUTDOWN_WAIT  4
#define AUX_SHUTDOWN_FAIL  5
#define AUX_SHUTDOWN    6
#define   USER1  0
#define   USER2  1
#define   pr_STORE_PENDING   2
#define   pr_STORE_NOT_PENDING 3
#define INT_MASK   0x01
/* make sure static is handled by compiler. Does it show up in link map? */
static unsigned char radio_options;
static unsigned char sample_volume;
static unsigned char sample_volume_compensation;
static unsigned char sample_comp_level;
static unsigned char sample_flags;
static unsigned char button_table[4];
static unsigned char button_time[4];
static unsigned char button_matrix_flags;
static unsigned char user;
static unsigned char presetx;
unsigned char speed_int_register;
unsigned char rx_scp_speed;
unsigned char sample_array3[10];

/*****************************************************************************
             CODE - Private Functions
 *****************************************************************************/
void pr_write_eeprom_preset(unsigned  char user1, unsigned char presetx1, unsigned char
band1);
unsigned char inspect_radio_options(void);
unsigned char inspect_scp_msg_pending_flags(void);

/*****************************************************************************
             CODE - Public Functions
 *****************************************************************************/
void file2_function1(unsigned  char dummy_param)
{
 global_4 = 0x77 - dummy_param;
 if (TESTBIT(global_3, 0x02))
 {
  CLRBIT(global_15, 0x01);
 }
}
void file2_function2(unsigned  char dummy_param)
{
 global_1 = 0x77 - dummy_param;
 if (TESTBIT(global_3, 0x02))
 {
  SETBIT(global_17, 0x01);
 }
```

```
}
void file2_function3(unsigned char dummy_param)
{
 global_3 = 255 - (2 * dummy_param);
 if (TESTBIT(global_5, 0x02))
 {
  SETBIT(global_19, 0x01);
 }
}
void file2_function4(void)
{
 global_4 = 255;
 if (TESTBIT(global_5, 0x02))
 {
  SETBIT(global_20, 0x01);
 }
}
unsigned char file2_function5(void)
{
 global_5 = 255;
 if (TESTBIT(global_6, 0x02))
 {
  CLRBIT(global_17, 0x01);
 }
 for (presetx = 0; presetx < 6; presetx++)
 {
  global_1 = sample_array[presetx];
  /*---------------------------*/
  global_5 = sample_array[presetx];
  if (((global_5 == USER1) || (user == USER2)) &&
 (global_15 <= 2))
  {
   pr_write_eeprom_preset(user,  presetx, global_15);
   return (pr_STORE_PENDING);/* still need to continue check */
  }
  /*---------------------------*/
  if (global_15 <= 2)
  {
   pr_write_eeprom_preset(user,  presetx, global_15);
   return (pr_STORE_PENDING);/* still need to continue check */
  }
  else              /* invalid/empty */
   /* global_5 = global_4; } } /*else never found any valid data to store */
   return (pr_STORE_NOT_PENDING);
 }
 void pr_write_eeprom_preset(unsigned char user1, unsigned char presetx1, unsigned char
band1)
 {
  if (user1 > 1)
  {
   SETBIT(global_9, 0x01);
   global_8 = presetx1;
  }
  else
  {
```

```
   CLRBIT(global_19, 0x20);
   global_7 = band1;
  }
 }
 void file2_function6(void)
 {
  copy_array(&sample_array[0], &sample_array3[0], sizeof sample_array);
 }
 void file2_function7(void)
 {
  copy_array(&sample_array[0], &sample_array3[0], sizeof sample_array);
 }
 void file2_function8(void)
 {
  unsigned char local_temp;
   local_temp = check_if_array_same(&sample_array[0], &sample_array2[0], sizeof
sample_array);
  if (local_temp == 0)
  {
   SETBIT(global_9, 0x04);
  }
 }
/* sample while loop */
 void file2_function9(void)
 {
  unsigned char local_temp;
  unsigned char delay_count;
   local_temp = check_if_array_same(&sample_array[0], &sample_array2[0], sizeof
sample_array);
  if (local_temp == 0)
  {
   SETBIT(global_7, 0x40);
  }
   SETBIT(global_5, 0x04);

  delay_count = 18;/* Wait for transmit to end.  (g) */
  while (TESTBIT(global_5, 0x04) && --delay_count);/* dummy wait */
  if (delay_count == 0)
  {
   return;
  }
  global_19 = 1;
 }
 void file2_function10(void)
 {
  unsigned char local_temp;
  unsigned char temp_count;
  unsigned int ee_crc_temp;
   local_temp = check_if_array_same(&sample_array[0], &sample_array2[0], sizeof
sample_array);
   temp_count = inspect_radio_options();
   ee_crc_temp = temp_count + 1000;
  if (local_temp == 0)
  {
   global_4 = ee_crc_temp;
```

```
  SETBIT(global_7, 0x40);
 }
}
unsigned char file2_function11(unsigned char dummy_param)
{
 unsigned char local_one;
  local_one = 0;
 if (dummy_param < 25)
 {
  local_one = 1;
 }
 else if ((dummy_param > 55) && (dummy_param < 75))
 {
  local_one = 0;
 }
 return (local_one);
}
unsigned char file2_function12(unsigned char dummy_param)
{
 unsigned char local_two;
 unsigned char local_three;
  local_two = (dummy_param + 55) - 10;
  local_three = 0x55;
  CLRBIT(local_three, 0x01);
 if (local_two < 25)
 {
  SETBIT(local_three, 0x02);
 }
 else if ((local_two > 55) && (local_two < 75))
 {
  SETBIT(local_three, (0x04 | 0x20));
 }
 else if (local_two == 45)
 {
  SETBIT(local_three, 0x08);
 }
 if ((TESTBIT(local_three, 0x04)) || (TESTBIT(local_three, 0x02)))
 {
  SETBIT(local_three, 0x01);
 }
 return (local_three);
}
unsigned int file2_function13(unsigned char dummy_param)
{
 unsigned int local_one;
 switch (dummy_param)
 {
  case PARAM1:
  local_one = 1000;
  dummy_param++;
  break;
  case PARAM2:
  local_one = 1200;
  break;
  case PARAM3:
```

```
   local_one = 2400;
   break;
   default:
   local_one = 7000;
   dummy_param++;
   break;
  }
   global_9 = dummy_param;

  return (local_one);
 }
 unsigned char inspect_radio_options(void)
 {
  return (radio_options);
 }
 unsigned char inspect_scp_msg_pending_flags(void)
 {
  return (global_3);
 }
/* Function with divides, long and math. Function returns char when it is supposed
to return int. See if value promoted correctly. */
 unsigned int file2_function14(unsigned char dummy_param)
 {
  unsigned char numerator, speed, speed_last;
  unsigned int temp;
   numerator = 0;
   speed = 0;
   speed_last = 0;
   temp = 0;
  if (TESTBIT(global_1, sample_DISABLED))
    return (0);
  if (inspect_radio_options() & SCP_OPT)
  {
   if (inspect_scp_msg_pending_flags() & SCP_SPEED_MSG)
   {
 temp = rx_scp_speed / 128;
 CLRBIT(global_3, SCP_SPEED_MSG);
 if (temp > MAX_SPEED_DETECT)
  temp = MAX_SPEED_DETECT;
 speed_last = temp;
   }
  }
  else
  {
   if (global_2 == 0)
   {
 CLRBIT(global_3, 0x01);
 if (global_1)
 {
  global_3 = global_1 - 1;
 }
   }
   if (global_1)
 --global_1;
   if ((!TESTBIT(speed_int_register, INT_MASK)) && (dummy_param > 5))
```

```
  {
temp = (unsigned long) 221814 /
 (unsigned long) (global_3 - global_1);
if (temp > MAX_SPEED_DETECT)/* Limit */
 temp = MAX_SPEED_DETECT;
speed_last = temp;
  }
 }
 speed = speed_last;
 /* No compensation if < */
 if (speed < MIN_SPEED_DETECT)
  speed = MIN_SPEED_DETECT;
 speed = speed - MIN_SPEED_DETECT;
 /* calculates compensation based on user set level */
 switch (sample_comp_level)
 {
 case 7:
  numerator = 93;
  break;
 case 6:
  numerator = 78;
  break;
 case 5:
  numerator = 66;
  break;
 case 4:
  numerator = 51;
  break;
 case 3:
  numerator = 42;
  break;
 case 2:
  numerator = 34;
  break;
 case 1:
  numerator = 23;
  break;
 default:
  numerator = 0;
 }
 sample_volume = ((numerator * speed) / 256);
 if (sample_volume > MAX_COMPENSATION_LEVEL)
  sample_volume = MAX_COMPENSATION_LEVEL;
 /* This part filters the calculated compensation for output */
 if (sample_volume < sample_volume_compensation)
 {
  if (!TESTBIT(sample_flags, sample_PLUS))
--sample_volume_compensation;
  else
CLRBIT(sample_flags, sample_PLUS);
 }
 if (sample_volume > sample_volume_compensation)
 {
  if (TESTBIT(sample_flags, sample_PLUS))
++sample_volume_compensation;
```

```
   else
  SETBIT(sample_flags, sample_PLUS);
   }
   return (speed);
 }
 void file2_function15(void)
 {
  global_3 = (global_2 * 5);
  global_4 = global_5;
 }
 void file2_function16(void)
 {
  global_5 = ((global_2 * 25) + 55) - 45;
  global_7 = global_5 & 0x01;
 }
/* FOR loop implementation, initialization, optimization */
 void file2_function17(void)
 {
  unsigned char i;
  /* clear all button RAM */
  for (i = 0; i < 4; i++)
  {
   button_table[i] = 0;
   button_time[i] = 0;
  }
   button_matrix_flags = 0;
 }
/* table search */
 void file2_function18(void)
 {
  unsigned char table_index;
   table_index = 0;
  while (table_index < 15)
  {
   if (sample_array2[table_index] == 0)
  break;
   else
  table_index++;
  }
   global_8 = table_index;
 }
 void file2_function19(void)
 {
  unsigned char index;
  while (global_4 > 0)
  {
   if (sample_array2[1] == 1)
   {
  global_4--;
  /* move remaining buttons up in queue */
  for (index = 0; index < global_4; index++)
  {
   sample_array2[index] = sample_array2[index + 1];
  }
   }
```

```
   else
  break;                                                /* stop at first failure */
  }
 }
/* check if constant gets pulled out of for loop in optimization */
 void file2_function20(void)
 {
  unsigned char index;
  /* clear button_pending_queue */
  for (index = 0; index < sizeof sample_array2; index++)
  {
   sample_array2[index] = 5 + global_5;
  }
   global_4 = (100 - 5) * 2;

  global_10 = 5 + global_3;
 }
/* typical state machine manager and array passing, array manipulation */
 void file2_function21(unsigned char msg_size, unsigned char msg[])
 {
  switch (msg[0])
  {
   case AUX_OFF:
   case AUX_ON:
   case AUX_RESET:
   case AUX_DIAG:
   case AUX_NULL_STATE:
   if (msg_size == 1)
   {
 global_7 = msg[0];
   }
   if (global_7 == AUX_RESET)
   {
 file2_function2(global_3);
 ;                                                      /* null statement on purpose */
   }
   else if ((global_8 == AUX_READY) && ((global_7 != AUX_ON) || (global_5 != 5)))
   {
 /* announce_event1(EVENT_TWO) */ ;
 /* announce_event1(EVENT_ONE) */ ;/* do not call events that use 21 */
   }
   else if ((global_8 == AUX_DIAG_MODE) && (global_7 == AUX_DIAG))
   {
 file2_function3(4);
   }
   else if ((global_8 == AUX_SHUTDOWN) ||
     (global_8 == AUX_SHUTDOWN_FAIL))/* error condition */
   {
 file2_function4();
 SETBIT(global_1, 0x02);
   }
   break;
  case AUX_SW_VERSION:
   if (msg_size == 4)
   {
```

```
 sample_array[0] = msg[1];
 sample_array[1] = msg[2];
 sample_array[2] = msg[3];
   }
   break;
 default:
   sample_array[4] = 0;
   CLRBIT(global_1, 0x80);
   break;
 }                                                    /* end switch(msg[0]) */
 return;
}
void file2_function22(unsigned int aux_comm_timeout, unsigned char retry_txmsg)
{
 if (aux_comm_timeout != 0)
 {
  if ((--global_4) == 0)
  {
if (global_8 == AUX_INIT_WAIT)
  /* announce_event1(EVENT_TWO) */ ;
else if (global_8 == AUX_DIAG_MODE)
  /* announce_event1(EVENT_THREE) */ ;
if (global_8 != AUX_SHUTDOWN_WAIT)
  /* announce_event(EVENT_TWO,3) */ ;
else
  /* announce_event(EVENT_THREE,7) */ ;
  }
  else
  {
switch (retry_txmsg)
{
 case AUX_ON:
 case AUX_OFF:
 case AUX_DIAG:
 SETBIT(global_9, 0x04);
 break;
 case REQUEST_STATUS:
 CLRBIT(global_10, 0x40);
 break;
}
global_9 = global_1;
global_1 = global_2;
global_3 = global_5;
  }
 }
  return;
}
/* else and if do the same thing. See if compiler optimizes */
 void file2_function23(unsigned char dummy_param)
 {
  if (dummy_param == 7)
  {
   fill_byte_array(sample_array,  sizeof sample_array, 55);
  }
  else
```

```
  {
   fill_byte_array(sample_array,  sizeof sample_array, 55);
  }
 }
```

7.6 FILE3.H

```
/*****************************************************************************
 *
 * Module:       utility.h
 *****************************************************************************/
#ifndef FILE3_H
#define FILE3_H
#ifdef FILE3_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
/*****************************************************************************
            Constant definitions
 *****************************************************************************/
/* return values for function  check_if_arrays_same */
#define ARRAYS_DIFFERENT   2
#define ARRAYS_SAME     1
/* defines used in convert_check */
#defineLOWERCASE_A                                              0x61
#define LOWERCASE_Z                                         0x7A
#define LOWER_TO_UPPERCASE0x20
#define RADIO_TEXT_DOLLAR      0xAB
#define DOLLAR_SIGN          98
#define  GENERATOR_POLYNOMIAL  0x1081
/*****************************************************************************
            GLOBAL Variable definitions
 *****************************************************************************/
/* hex to bcd conversion*/
EXTERN_pfx unsigned char convert_value[3];

/*****************************************************************************
            Public Function Prototypes
 *****************************************************************************/
/* return values are  ARRAYS_SAME and ARRAYS_DIFFERENT */
EXTERN_pfx unsigned char check_if_array_same(unsigned  char *first_array,
                                                    unsigned char *second_array,
                                                    unsigned char array_size);/*
compare two arrays */

/* copies one array to another */
EXTERN_pfx void copy_array(unsigned  char *array_to_be_copied_from,
                       unsigned char *array_to_be_copied_into,
                       unsigned char array_size);    /* copy one array into another */

/* fills array with supplied value */
EXTERN_pfx void fill_byte_array(unsigned  char *array_start_ptr,
                                              unsigned char array_size,
```

```
                                                    unsigned char fill_value);

/* fills array with ASCII 0x20 i.e, BLANK */
EXTERN_pfx void blank_selected_buffer(unsigned  char *array_to_be_blanked,
                                                      unsigned char array_length);

/* checks if the character is lower case converts it to upper case and checks for RBDS
  dollar which is located different than the ascii dollar. Blanks out all other chars
  if less then SPACE_CHAR(0x20) or Greater than alpha_Z (upper case Z)  */
EXTERN_pfx void blank_selected_buffer(unsigned  char *array_to_be_blanked, unsigned char
array_length);

/* convert check is used to check if all characters in an array are displayable
 * accroding to the RBDS characters which is similar to ASCII characters except
 * for some characters. Order of check conversions done
 * 1. If characters between lowercase a to Z converts to upper case
 * 2. Check for dollar sign as out of range of check
 * 3. Blank characters less then (SPACE_CHAR ox20) or greater than (alpha_Z) as
 *  none displayable character */
EXTERN_pfx void convert_check(unsigned  char *array_to_be_checked, unsigned char
array_length);

/* converts a unsigned integer to bcd and places it in an
array of 3 bytes byte 0 being the most significant and byte 2 the least significant
 Return value: Global:
 convert_value[3]   The output is a global array which is cleared
                 each time the function is run so the value must
                 be consumed as soon as the function is run.
      convert_value[0]   most significant byte BCD
      convert_value[1]   next significant byte BCD
      convert_value[2]   least significant byte BCD */
EXTERN_pfx void convert_hex_2_bcd(unsigned  char value_to_convert);
EXTERN_pfx unsigned int generate_crc(unsigned char *address, unsigned char number_of_bytes,
                                                      unsigned int current_crc);

#undef  EXTERN_pfx
```

7.7 FILE3.C

```
/*****************************************************************************
 *
 * Module:       utility.c
 *****************************************************************************/
#define FILE3_C
/*****************************************************************************
            Include Files
 *****************************************************************************/
#include "commnlib.h"
#include "file3.h"
#include "file4.h"
unsigned char sample_array4[5];

/*****************************************************************************
            CODE - Public Functions
```

```
 ****************************************************************************/
/****************************************************************************
 *
 * Function:     copy_array*
 ****************************************************************************/
void copy_array(unsigned char *array_to_be_copied_from, unsigned char
*array_to_be_copied_into, unsigned char array_size)
{
 unsigned char i;

 for (i = 0; i < array_size; i++)
 {
  array_to_be_copied_into[i] = array_to_be_copied_from[i];
 }
}                                                    /* end of function copy_array */
/****************************************************************************
 *
 * Function:     check_if_array_same*
 ****************************************************************************/
unsigned char check_if_array_same(unsigned char *first_array, unsigned char *second_array,
unsigned char array_size)
{
 unsigned char i;

 for (i = 0; i < array_size; i++)
 {
  if (first_array[i] != second_array[i])
  {
   return (ARRAYS_DIFFERENT);
  }
 }
 return (ARRAYS_SAME);
}                                                    /* end function */
/****************************************************************************
 *
 * Function:     fill_byte_array*
 ****************************************************************************/
/* usage array name (ie., start address) as first argument and sizeof as second */
void fill_byte_array(unsigned char *array_start_ptr, unsigned char array_size, unsigned
char fill_value)
{
 while (array_size--)
  *array_start_ptr++ = fill_value;
}                                                    /* end of function fill_byte_array */
/****************************************************************************/
/*
 * Function:    blank_selected_buffer
 *
****************************************************************************/
void blank_selected_buffer(unsigned char *array_to_be_blanked, unsigned char array_length)
{
 fill_byte_array(&array_to_be_blanked[0], array_length, SPACE_CHAR);
}
/****************************************************************************/
/*
```

```
 * Function:    convert_hex_2_bcd *
*****************************************************************************/
void convert_hex_2_bcd(unsigned char value_to_convert)
{
 unsigned char value_left;

 fill_byte_array(&sample_array4[0], 3, 0x00);
 if (value_to_convert <= 9)
  sample_array4[2] = value_to_convert;
 else
 {
  sample_array4[2] = value_to_convert % 0x0A;
  if (value_to_convert <= 99)
   sample_array4[1] = value_to_convert / 0x0A;
  else
  {
   value_left = value_to_convert / 0x0A;
   sample_array4[1] = value_left % 0x0A;
   sample_array4[0] = value_left / 0x0A;
  }
 }
}
/* intensive calculation. Look for processing cycles and ROM - RAM */
unsigned int generate_crc(unsigned char *address, unsigned char number_of_bytes,
                      unsigned int current_crc)
{
 unsigned char *bound_address;
 unsigned char pc;
 unsigned char q;

 bound_address = address + number_of_bytes;
 while (address < bound_address)
 {
  pc = *address++;
  q = (current_crc ^ pc) & 0x0F;/* lower nibble */
  current_crc = (current_crc >> 4) ^ (q * GENERATOR_POLYNOMIAL);
  q = (current_crc ^ (pc >> 4)) & 0x0F;/* high nibble */
  current_crc = (current_crc >> 4) ^ (q * GENERATOR_POLYNOMIAL);
 }
 return (current_crc);
}
void convert_check(unsigned char *array_to_be_checked, unsigned char array_length)
{
 unsigned char i;

 for (i = 0; i < array_length; i++)
 {
  if ((array_to_be_checked[i] >= LOWERCASE_A) && (array_to_be_checked[i] <= LOWERCASE_Z))
   array_to_be_checked[i] = array_to_be_checked[i] - LOWER_TO_UPPERCASE;
  else if (array_to_be_checked[i] == RADIO_TEXT_DOLLAR)
   array_to_be_checked[i] = DOLLAR_SIGN;
  else if ((array_to_be_checked[i] > LOWERCASE_Z) ||
    (array_to_be_checked[i] < SPACE_CHAR))
   array_to_be_checked[i] = SPACE_CHAR;
 }
```

7.8 FILE4.H

```
#ifndef FILE4_H
#define FILE4_H
#ifdef FILE4_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
/*****************************************************************************/
/*
 * FILE CHAR_DEF.H: Character definitions for use with display load messages
 */
/*****************************************************************************/
/*
 * Constant Definitions
 */
/* Lower message window -- ASCII Codes */
/* Numbers 0 through 9 */
#define ZERO  48
#define ONE   49
#define TWO   50
#define THREE 51
#define FOUR  52
#define FIVE  53
#define SIX   54
#define SEVEN 55
#define EIGHT 56
#define NINE  57
/* Letter A through Z (All caps) */
#define alpha_A   65
#define alpha_B   66
#define alpha_C   67
#define alpha_D   68
#define alpha_E   69
#define alpha_F   70
#define alpha_G   71
#define alpha_H   72
#define alpha_I   73
#define alpha_J   74
#define alpha_K   75
#define alpha_L   76
#define alpha_M   77
#define alpha_N   78
#define alpha_O   79
#define alpha_P   80
#define alpha_Q   81
#define alpha_R   82
#define alpha_S   83
#define alpha_T   84
#define alpha_U   85
#define alpha_V   86
#define alpha_W   87
#define alpha_X   88
#define alpha_Y   89
```

```
#define alpha_Z    90
/* Definitions of special symbols for use in message window */
#define CUST_CTR_BAR     5
#define CTR_BAR_LR_HALF_BAR 6
#define CTR_BAR_RT_HALF_BAR 7
#define CTR_BAR_LT_HALF_BAR 8
#define LT_RT_HALF_BARS   9
#define CTR_N_LT_BAR_DASH  10
#define CTR_N_RT_BAR_DASH  11
#define ALPHA_I_DASH     12
#define LT_BAR_SDASH_LT 13
#define LT_BAR_LDASH   14
#define RT_BAR_SDASH_LT 15
#define RT_BAR_LDASH   16
#define CTR_BAR_SDASH_LT 17
#define CTR_BAR_SDASH_RT 18
#define HALF_R_BAR    19
#define L_AND_HALF_R_BAR 20
#define HALF_L_BAR    21
#define DBL_BAR     22
#define UP_ARROW     23
#define DOWN_ARROW    24
#define SDASH_RT     25
#define END_BAR_STAR  26
#define END_BAR     27
/* segments,  segments b,c, and d */
#define L_BAR      28
#define R_BAR      29
#define SDASH_LT     30
#define BAR_STAR     31
#define SPACE_CHAR    32
#define EXCLAM_PT    33
#define DBL_QUOTE    34
#define NUMBER_HASH   35
#define DOLLAR_CHAR   36
#define PERCENT_CHAR  37
#define AMPERSAND    38
#define SNGL_RT_QUOTE  39
#define LT_PAREN     40
#define RT_PAREN     41
#define STAR       42
#define PLUS_CHAR    43
#define COMMA_CHAR    44
#define LDASH      45
#define PERIOD      46
#define FRONT_DIAG    47
#define COLON      58
#define SEMICOLON    59
#define LT_ARROW     60
#define EQUAL_CHAR    61
#define RT_ARROW     62
#define QUESTION_MARK  63
#define CIRCLE_A_CHAR  64
#define LT_SQ_BRACKET  91
#define BACK_DIAG    92
```

```
#define RT_SQ_BRACKET  93
#define MIDDLE_BAR    94
#define UND_SCORE    95
#define PARRELLEL_CHAR 96
#define SMALL_o     97
#define DOLLAR_SIGN   98
#define SMALL_c     99
#define CTR_BAR     124
/* characters defined for phone*/
/* all characters where previously defined therefore the characters for phone
  are mapped out twice ph char is the character received = denotes to the
  char in the table mapped out to and the equ denotes the character already
  existing so can be changed if any new custom characters required*/
#define PH_UND_SCORE   100
#define PH_BAR_STAR   101
#define PH_END_BAR_STAR 102
#define PH_END_BAR    103
EXTERN_pfx void audio_process(void);

#undef EXTERN_pfx
```

7.9 FILE4.C

```
#define FILE4_C
/*****************************************************************************
            Include Files
 *****************************************************************************/
#include ”commnlib.h”
#include ”file4.h”
#include ”file2.h”
#include ”file3.h”
const unsigned char lower_disp_msg_table[128][12] = {
alpha_S, alpha_E, alpha_E, alpha_K, SPACE_CHAR, alpha_U, alpha_P, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_E, alpha_E, alpha_K, SPACE_CHAR, alpha_D, alpha_O, alpha_W, alpha_N,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_R, alpha_A, alpha_D, alpha_I, alpha_O, SPACE_CHAR, alpha_S,
alpha_C, alpha_A, alpha_N, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_U, alpha_T, alpha_O, alpha_S, alpha_E,
alpha_T, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_U, alpha_T, alpha_O,
alpha_S, alpha_E, alpha_T, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_U, alpha_T,
alpha_O, alpha_S, alpha_E, alpha_T, SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR,alpha_A, alpha_U,
alpha_T, alpha_O, alpha_L, alpha_O, alpha_A, alpha_D, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR,alpha_S,
alpha_C, alpha_A, alpha_N, SPACE_CHAR, alpha_A, alpha_U, alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_M, alpha_E, alpha_T, alpha_A, alpha_L, SPACE_CHAR, alpha_T, alpha_A, alpha_P, alpha_E,
SPACE_CHAR, SPACE_CHAR,alpha_E, alpha_J, alpha_E, alpha_C, alpha_T, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_E, alpha_J, alpha_E, alpha_C, alpha_T, SPACE_CHAR,
alpha_E, alpha_R, alpha_R, alpha_O, alpha_R, SPACE_CHAR,alpha_L, alpha_O, alpha_A, alpha_D, SPACE_CHAR,
alpha_E, alpha_R, alpha_R, alpha_O, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_R, alpha_E, alpha_E, alpha_L,
SPACE_CHAR, alpha_E, alpha_R, alpha_R, alpha_O, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O,
SPACE_CHAR, alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_S, alpha_P, alpha_K, alpha_R, SPACE_CHAR, alpha_S, alpha_H, alpha_O, alpha_R, alpha_T,
SPACE_CHAR, SPACE_CHAR,alpha_P, alpha_H, alpha_O, alpha_N, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_F, alpha_A, alpha_S,
alpha_T, SPACE_CHAR, alpha_F, alpha_W, alpha_D, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_F, alpha_A,
alpha_S, alpha_T, SPACE_CHAR, alpha_R, alpha_E, alpha_V, SPACE_CHAR,alpha_S, alpha_H, alpha_U, alpha_F,
alpha_F, alpha_L, alpha_E, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_H, alpha_U,
alpha_F, alpha_F, alpha_L, alpha_E, SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR,alpha_S, alpha_H,
alpha_U, alpha_F, alpha_F, alpha_L, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_F, alpha_O, alpha_C, alpha_U, alpha_S, SPACE_CHAR, alpha_E, alpha_R, alpha_R, alpha_O,
alpha_R, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_H, alpha_O, alpha_T, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_R, alpha_O, alpha_M,
SPACE_CHAR, alpha_E, alpha_R, alpha_R, alpha_O, alpha_R,alpha_C, alpha_O, alpha_M, alpha_M, SPACE_CHAR,
alpha_E, alpha_R, alpha_R, alpha_O, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR, alpha_M,
```

```
alpha_A, alpha_G, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_E, alpha_M,
alpha_P, alpha_T, alpha_Y, SPACE_CHAR, alpha_M, alpha_A, alpha_G, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
alpha_N, alpha_O, SPACE_CHAR, alpha_C, alpha_D, alpha_D, alpha_J, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR, alpha_R, alpha_E, alpha_M, alpha_O, alpha_T, alpha_E,
SPACE_CHAR, alpha_C, alpha_D,alpha_N, alpha_O, SPACE_CHAR, alpha_C, alpha_D, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_R, alpha_A, alpha_F, alpha_F,
alpha_I, alpha_C, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_R, alpha_A,
alpha_F, alpha_F, alpha_I, alpha_C, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_R,
alpha_A, alpha_F, alpha_F, alpha_I, alpha_C, SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR,alpha_T,
alpha_E, alpha_X, alpha_T, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_T, alpha_E, alpha_X, alpha_T, SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_F, alpha_O, alpha_R, alpha_M, alpha_A, alpha_T, SPACE_CHAR,
alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_F, alpha_O, alpha_R, alpha_M, alpha_A, alpha_T,
SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_U, alpha_T, alpha_O,
SPACE_CHAR, alpha_C, alpha_L, alpha_K, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR,alpha_A, alpha_U, alpha_T,
alpha_O, SPACE_CHAR, alpha_C, alpha_L, alpha_K, SPACE_CHAR, alpha_O, alpha_F, alpha_F,alpha_S, alpha_E,
alpha_L, alpha_E, alpha_C, alpha_T, SPACE_CHAR, alpha_H, alpha_O, alpha_U, alpha_R, SPACE_CHAR,alpha_S,
alpha_E, alpha_L, alpha_E, alpha_C, alpha_T, SPACE_CHAR, alpha_M, alpha_I, alpha_N, alpha_S,
SPACE_CHAR,alpha_A, alpha_U, alpha_T, alpha_O, SPACE_CHAR, alpha_C, alpha_L, alpha_K, SPACE_CHAR, alpha_S,
alpha_E, alpha_T,alpha_R, alpha_D, alpha_S, SPACE_CHAR, alpha_O, alpha_N, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_R, alpha_D, alpha_S, SPACE_CHAR, alpha_O, alpha_F,
alpha_F, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_A, SPACE_CHAR, alpha_R,
alpha_E, alpha_P, alpha_O, alpha_R, alpha_T, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_J, alpha_A, alpha_Z,
alpha_Z, SPACE_CHAR, alpha_C, alpha_L, alpha_U, alpha_B, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_H,
alpha_A, alpha_L, alpha_L, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_H, alpha_U, alpha_R, alpha_C, alpha_H, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_T, alpha_A, alpha_D, alpha_I, alpha_U,
alpha_M, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_E, alpha_W, alpha_S,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_D,
alpha_S, alpha_P, SPACE_CHAR, alpha_O, alpha_F, alpha_F, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_O, alpha_C, alpha_C, alpha_U, alpha_P, alpha_A, alpha_N, alpha_C, alpha_Y, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_D, alpha_R, alpha_I, alpha_V, alpha_E, alpha_R, SPACE_CHAR, alpha_S, alpha_E,
alpha_A, alpha_T, SPACE_CHAR,alpha_N, alpha_O, alpha_T, SPACE_CHAR, alpha_F, alpha_O, alpha_U, alpha_N,
alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_R, alpha_E, alpha_A, alpha_R, SPACE_CHAR, alpha_S,
alpha_E, alpha_A, alpha_T, alpha_S, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_L, alpha_L, SPACE_CHAR, alpha_S,
alpha_E, alpha_A, alpha_T, alpha_S, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_A, alpha_V, alpha_E,
alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_F,
alpha_A, alpha_U, alpha_L, alpha_T, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_P, alpha_E, alpha_A, alpha_K, alpha_E, alpha_R, SPACE_CHAR, alpha_L,
alpha_F, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_P, alpha_E, alpha_A, alpha_K, alpha_E, alpha_R, SPACE_CHAR,
alpha_R, alpha_F, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_P, alpha_E, alpha_A, alpha_K, alpha_E, alpha_R,
SPACE_CHAR, alpha_R, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_P, alpha_E, alpha_A, alpha_K, alpha_E,
alpha_R, SPACE_CHAR, alpha_L, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_A, alpha_N, alpha_T, alpha_E, alpha_N,
alpha_N, alpha_A, SPACE_CHAR, alpha_I, alpha_I, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_U, alpha_B, alpha_W,
alpha_O, alpha_O, alpha_F, alpha_E, alpha_R, SPACE_CHAR, alpha_I, SPACE_CHAR,alpha_S, alpha_U, alpha_B,
alpha_W, alpha_O, alpha_O, alpha_F, alpha_E, alpha_R, SPACE_CHAR, alpha_I, alpha_I,alpha_S, alpha_E,
alpha_L, alpha_F, SPACE_CHAR, alpha_T, alpha_E, alpha_S, alpha_T, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_S, alpha_E, alpha_L, alpha_F, SPACE_CHAR, alpha_P, alpha_A, alpha_S, alpha_S, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_E, alpha_L, alpha_F, SPACE_CHAR, alpha_F, alpha_A, alpha_I, alpha_L,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR, alpha_D, alpha_T, alpha_C, alpha_S,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_D, alpha_T, alpha_C, alpha_S, SPACE_CHAR,
alpha_F, alpha_O, alpha_U, alpha_N, alpha_D, SPACE_CHAR, SPACE_CHAR,alpha_D, alpha_T, alpha_C, alpha_S,
SPACE_CHAR, alpha_C, alpha_L, alpha_E, alpha_A, alpha_R, SPACE_CHAR, SPACE_CHAR,alpha_U, TWO, ZERO, ZERO,
THREE, SPACE_CHAR, alpha_C, alpha_D, alpha_D, alpha_J, SPACE_CHAR, SPACE_CHAR,alpha_U, TWO, ZERO, ZERO,
FIVE, SPACE_CHAR, alpha_R, alpha_I, alpha_C, alpha_P, SPACE_CHAR, SPACE_CHAR,alpha_U, TWO, ZERO, ZERO,
EIGHT, SPACE_CHAR, alpha_P, alpha_H, alpha_O, alpha_N, alpha_E, SPACE_CHAR,alpha_U, ONE, ZERO, FOUR, ONE,
SPACE_CHAR, alpha_S, alpha_C, alpha_P, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_U, ONE, ONE, EIGHT, ZERO,
SPACE_CHAR, alpha_S, alpha_C, alpha_P, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_U, ONE, TWO, TWO, TWO,
SPACE_CHAR, alpha_S, alpha_C, alpha_P, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_B, ONE, THREE, FOUR, TWO,
SPACE_CHAR, alpha_E, alpha_C, alpha_U, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_B, TWO, FOUR, ZERO, ONE,
SPACE_CHAR, alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR,alpha_B, TWO, FOUR, ZERO, TWO,
SPACE_CHAR, alpha_C, alpha_D, alpha_D, alpha_J, SPACE_CHAR, SPACE_CHAR,alpha_B, TWO, FOUR, ZERO, THREE,
SPACE_CHAR, alpha_C, alpha_D, alpha_D, alpha_J, SPACE_CHAR, SPACE_CHAR,alpha_B, TWO, FOUR, ZERO, FOUR,
SPACE_CHAR, alpha_S, alpha_W, alpha_C, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_B, TWO, FOUR, ZERO, FIVE,
SPACE_CHAR, alpha_C, alpha_D, SPACE_CHAR, alpha_H, alpha_O, alpha_T,alpha_B, TWO, FOUR, ZERO, SIX,
SPACE_CHAR, alpha_C, alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_I, alpha_G,
alpha_N, alpha_A, alpha_L, SPACE_CHAR, alpha_T, alpha_E, alpha_S, alpha_T, SPACE_CHAR,alpha_D, alpha_I,
alpha_S, alpha_P, alpha_L, alpha_A, alpha_Y, SPACE_CHAR, alpha_T, alpha_E, alpha_S, alpha_T,alpha_T,
alpha_A, alpha_P, alpha_E, SPACE_CHAR, alpha_T, alpha_E, alpha_S, alpha_T, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, alpha_F, alpha_A, alpha_I, alpha_L, SPACE_CHAR,
```

```
SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, alpha_P, alpha_A, alpha_S, alpha_S,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_O, alpha_F, alpha_T, SPACE_CHAR, alpha_L, alpha_E,
alpha_V, alpha_E, alpha_L, alpha_S, SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, ONE,
SPACE_CHAR, alpha_F, alpha_F, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E,
SPACE_CHAR, TWO, SPACE_CHAR, alpha_F, alpha_F, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_A,
alpha_P, alpha_E, SPACE_CHAR, ONE, SPACE_CHAR, alpha_R, alpha_E, alpha_W, SPACE_CHAR, SPACE_CHAR,alpha_T,
alpha_A, alpha_P, alpha_E, SPACE_CHAR, TWO, SPACE_CHAR, alpha_R, alpha_E, alpha_W, SPACE_CHAR,
SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_S, alpha_E, alpha_E, alpha_K, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_T, alpha_R, alpha_K, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_S, alpha_C,
alpha_A, alpha_N, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR,
alpha_R, alpha_O, alpha_M, SPACE_CHAR, alpha_D, alpha_I, alpha_S, alpha_K, SPACE_CHAR,alpha_C, alpha_D,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, alpha_F, alpha_F, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, alpha_R, alpha_E, alpha_V,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, alpha_S, alpha_E,
alpha_E, alpha_K, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
alpha_T, alpha_R, alpha_K, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, alpha_S, alpha_C, alpha_A, alpha_N, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_T,
alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, alpha_P, alpha_L, alpha_A, alpha_Y,
SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, alpha_S, alpha_E, alpha_E, alpha_K,
SPACE_CHAR, SPACE_CHAR,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
alpha_S, alpha_E, alpha_E, alpha_K,alpha_T, alpha_A, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
alpha_S, alpha_C, alpha_A, alpha_N, SPACE_CHAR,alpha_A, alpha_U, alpha_D, alpha_I, alpha_O, SPACE_CHAR,
alpha_M, alpha_U, alpha_T, alpha_E, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_P, alpha_E, alpha_E, alpha_D,
SPACE_CHAR, alpha_V, alpha_O, alpha_L, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR,
alpha_P, alpha_H, alpha_O, alpha_N, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S,
alpha_E, alpha_E, alpha_K, SPACE_CHAR, alpha_T, alpha_R, alpha_A, alpha_F, alpha_F, alpha_I,
alpha_C,alpha_D, alpha_I, alpha_A, alpha_G, alpha_N, alpha_O, alpha_S, alpha_T, alpha_I, alpha_C, alpha_S,
SPACE_CHAR,alpha_S, alpha_C, alpha_A, alpha_N, SPACE_CHAR, alpha_T, alpha_R, alpha_A, alpha_F, alpha_F,
alpha_I, alpha_C,alpha_A, alpha_L, alpha_E, alpha_R, alpha_T, SPACE_CHAR, alpha_R, alpha_E, alpha_P,
alpha_O, alpha_R, alpha_T,alpha_S, alpha_H, alpha_O, alpha_W, SPACE_CHAR, alpha_N, alpha_O, alpha_N,
alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_H, alpha_O, alpha_W, SPACE_CHAR, alpha_N,
alpha_A, alpha_M, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_H, alpha_O, alpha_W,
SPACE_CHAR, alpha_T, alpha_Y, alpha_P, alpha_E, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_S, alpha_H,
alpha_O, alpha_W, SPACE_CHAR, alpha_T, alpha_E, alpha_X, alpha_T, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR, alpha_P, alpha_O, alpha_R, alpha_T, alpha_A, alpha_B, alpha_L,
alpha_E, SPACE_CHAR,alpha_E, alpha_N, alpha_D, SPACE_CHAR, alpha_O, alpha_F, SPACE_CHAR, alpha_D, alpha_I,
alpha_S, alpha_C, SPACE_CHAR,alpha_S, alpha_E, alpha_A, alpha_R, alpha_C, alpha_H, SPACE_CHAR, alpha_E,
alpha_R, alpha_R, alpha_O, alpha_R,alpha_B, alpha_A, alpha_D, SPACE_CHAR, alpha_D, alpha_I, alpha_S,
alpha_C, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_E, alpha_R,
alpha_R, alpha_O, alpha_R, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR,
alpha_T, alpha_R, alpha_K, SPACE_CHAR, alpha_F, alpha_O, alpha_U, alpha_N, alpha_D,SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR,alpha_C, alpha_D, SPACE_CHAR, alpha_F, alpha_A, alpha_I, alpha_L, alpha_U, alpha_R,
alpha_E, SPACE_CHAR, SPACE_CHAR,alpha_N, alpha_O, SPACE_CHAR, alpha_R, alpha_D, alpha_S, SPACE_CHAR,
SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR, SPACE_CHAR
};
const unsigned char loudness_offset[] =
{0, 0, 0, 0, 0, 0, 15, 15, 30, 30, 45, 45,
 60, 60, 60, 60, 75, 75, 75, 75, 75, 75, 75, 75,
 90, 105, 105, 120, 120, 135, 135, 150, 165,
180, 180};
const unsigned char bass_index[] =
{18, 20, 22, 24, 26, 28, 30, 32, 34, 35, 36, 37, 38, 39, 40,
 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 37, 38, 39, 40,
 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 35, 36, 37, 38,
 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 33, 34, 35, 36,
 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 31, 32, 33, 34,
 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 29, 30, 31, 32,
 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 27, 28, 29, 30,
 6, 8, 10, 12, 14, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28,
 6, 8, 10, 12, 14, 16, 18, 20, 21, 22, 23, 24, 25, 25, 26,
 4, 6, 8, 10, 12, 14, 16, 18, 19, 20, 21, 22, 23, 23, 24,
 4, 6, 8, 10, 12, 14, 16, 18, 19, 19, 20, 20, 21, 21, 22,
 4, 5, 6, 8, 10, 12, 14, 16, 17, 18, 19, 20, 20, 21, 22,
```

```
4, 5, 6, 8, 10, 12, 14, 16, 16, 17, 17, 18, 18, 19, 20};
const unsigned char treble_index[] =
{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28};
const unsigned int balance_table[] =
{2047, 2047, 2047, 2047, 2047, 2047, 2047, 2047, 2047, 2047, 2047,
1773, 1773, 1536, 1330, 997, 648, 364, 0};
const unsigned int fade_table[] =
{0, 7, 27, 100, 205, 364, 561, 864, 997, 1330, 1536, 1536,
1773, 2047, 2047, 2047, 2047, 2047, 2047};
const unsigned char lower_win_even_digit_table[14] = {
 42,                                                  /* seg a */
 3,                                                   /* seg b */
 7,                                                   /* seg c */
 12,                                                  /* seg d */
 8,                                                   /* seg e */
 4,                                                   /* seg f */
 6,                                                   /* seg g */
 1,                                                   /* seg h */
 0,                                                   /* seg j */
 2,                                                   /* seg k */
 5,                                                   /* seg m */
 10,                                                  /* seg n */
 11,                                                  /* seg p */
 9                                                    /* seg r */
};
const unsigned char lower_win_odd_digit_table[14] = {
 26,                                                  /* seg a */
 22,                                                  /* seg b */
 18,                                                  /* seg c */
 13,                                                  /* seg d */
 17,                                                  /* seg e */
 21,                                                  /* seg f */
 19,                                                  /* seg g */
 24,                                                  /* seg h */
 25,                                                  /* seg j */
 23,                                                  /* seg k */
 20,                                                  /* seg m */
 15,                                                  /* seg n */
 14,                                                  /* seg p */
 16                                                   /* seg r */
};
struct btbf
{
 unsigned char bass;
 unsigned char treble;
 unsigned char balance;
 unsigned char fade;
};
struct btbf *audio;
struct btbf btbf1;
struct btbf btbf2;

/* structure and table operation */
void audio_process(void)
{
```

```
 btbf2.bass = bass_index[15];
 btbf2.treble = treble_index[4];
 btbf2.balance = balance_table[0];
 btbf2.fade = fade_table[10];
 if (btbf2.bass < 15)
 {
  /* Update bass, then send the new bass value to the Audio Processing IC. */
  ++btbf2.bass;
 }
 if (global_1 > 127)
  global_1 = 127;/* bounds check first */
 copy_array(&(lower_disp_msg_table[global_1][0]),  &sample_array[0], 12);
 if (global_5 > 127)
  global_5 = 127;/* bounds check first */
 copy_array(&(lower_disp_msg_table[global_5][0]),  &sample_array2[0], 12);
 if (global_21 > 127)
  global_21 = 127;/* bounds check first */
 copy_array(&(lower_disp_msg_table[global_21][0]),  &sample_array[0], 12);
 global_1 = 5;
 global_4 = balance_table[global_1] + fade_table[global_1];
 global_7 = bass_index[global_1 + 2];
 global_8 = treble_index[global_1 + 3];
 copy_array(&bass_index[0], &sample_array[0], sizeof sample_array);
 global_9 = loudness_offset[5] + 5;
 if (global_9 > 55)
 {
  global_15 = lower_win_even_digit_table[6];
 }
 else
 {
  global_17 = lower_win_odd_digit_table[12];
 }
 copy_array(&(lower_disp_msg_table[global_5][0]),  &sample_array2[0], 12);
 copy_array(&(lower_disp_msg_table[global_5][0]),  &sample_array[0], 12);
}
```

7.10 FILE5.H

```
#ifndef FILE5_H
#define FILE1_H
#ifdef FILE5_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
#define cbench_CASE_1   1
#define cbench_CASE_2   2
#define cbench_CASE_3   3
#define cbench_CASE_4   4
#define cbench_CASE_5   5
#define cbench_CASE_6   6
#define cbench_CASE_7   7
#define cbench_CASE_8   8
#define cbench_CASE_9   9
```

```
#define cbench_CASE_10  10
#define cbench_CASE_11  11
#define cbench_CASE_12  12
EXTERN_pfx void file5_function1(void);
EXTERN_pfx void file5_function2(void);
EXTERN_pfx void file5_function3(void);
EXTERN_pfx void file5_function4(void);
EXTERN_pfx void file5_function5(void);
EXTERN_pfx void file5_function6(void);

#undef EXTERN_pfx
```

7.11 FILE5.C

```
/* switch statement samples */
#define FILE5_C
/*****************************************************************************
            Include Files
 *****************************************************************************/
#include "commnlib.h"
#include "file2.h"
#include "file5.h"
/* look for optimization in first two cases. Cases in sequence also. See if
  switch table used */
void file5_function1(void)
{
 switch (global_1)
 {
  case cbench_CASE_1:
  file2_function1(1);
  break;
 case cbench_CASE_2:
  file2_function1(1);
  break;
 case cbench_CASE_3:
  file2_function2(5);
  break;
 case cbench_CASE_4:
  file2_function3(5);
  break;
 case cbench_CASE_5:
  file2_function4();
  break;
 default:
  file2_function4();
  break;
 }
}
/* cases out of order. Verify how much ROM compared to function1 */
void file5_function2(void)
{
 switch (global_2)
 {
  case cbench_CASE_2:
```

```
  file2_function1(1);
  break;
 case cbench_CASE_1:
  file2_function1(1);
  break;
 case cbench_CASE_4:
  file2_function3(5);
  break;
 case cbench_CASE_3:
  file2_function2(5);
  break;
 case cbench_CASE_5:
  file2_function4();
  break;
 default:
  file2_function4();
  break;
 }
}
/* several cases. Reverse order */
void file5_function3(void)
{
 switch (global_7)
 {
  case cbench_CASE_12:
  file2_function1(10);
  break;
 case cbench_CASE_11:
  file2_function1(1);
  break;
 case cbench_CASE_10:
  file2_function2(3);
  file2_function1(1);
  file2_function3(10);
  break;
 case cbench_CASE_9:
  file2_function2(5);
  break;
 case cbench_CASE_8:
  file2_function3(5);
  break;
 case cbench_CASE_5:
  file2_function4();
  break;
 default:
  file2_function4();
  break;
 }
}
/* just two cases here. Check if it is more efficient than if */
void file5_function4(void)
{
 switch (global_17)
 {
  case cbench_CASE_12:
```

```
  file2_function1(10);
  break;
 default:
  file2_function4();
  break;
 }
}
/* 3 cases. Check if it is efficient to use switch or if */
void file5_function5(void)
{
 switch (global_19)
 {
  case cbench_CASE_1:
  file2_function1(10);
  break;
 case cbench_CASE_2:
  if ((global_9) && (global_1 & 0x01))
  {

   file2_function4();
  }
  break;
 default:
  file2_function4();
  break;
 }
}
/* 12 cases. Look for switch efficiency */
void file5_function6(void)
{
 switch (global_4)
 {
  case cbench_CASE_1:
  file2_function1(1);
  break;
 case cbench_CASE_2:
  file2_function1(1);
  break;
 case cbench_CASE_3:
  file2_function2(5);
  break;
 case cbench_CASE_4:
  file2_function3(5);
  break;
 case cbench_CASE_5:
  file2_function4();
  break;
 case cbench_CASE_6:
  file2_function6();
  break;
 case cbench_CASE_7:
  file2_function7();
  break;
 case cbench_CASE_8:
  file2_function8();
```

```
  break;
 case cbench_CASE_9:
  file2_function9();
  break;
 case cbench_CASE_10:
  file2_function10();
  break;
 case cbench_CASE_11:
  file2_function8();
  break;
 case cbench_CASE_12:
  file2_function9();
  break;
 default:
  file2_function10();
  break;
 }
```

7.12 FILE6.H

```
/*****************************************************************************
            Include Files
*****************************************************************************/
#ifndef FILE6_H
#define FILE6_H
#ifdef FILE6_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
/*****************************************************************************
            Public Constant Definitions
 *****************************************************************************/
EXTERN_pfx unsigned char cbench_dat[4];
EXTERN_pfx unsigned char cbench_syn[2];
EXTERN_pfx unsigned char cbench_af_flags;
EXTERN_pfx unsigned char cbench_sync_flags;
EXTERN_pfx unsigned char cbench_bit_ctr;
EXTERN_pfx unsigned char bad_blk_ctr;
EXTERN_pfx unsigned char next_block;
EXTERN_pfx unsigned char max_bad_blocks;
EXTERN_pfx unsigned char cbench_ls1;
EXTERN_pfx unsigned char cbench_ms1;
EXTERN_pfx unsigned char cbench_ls2;
EXTERN_pfx unsigned char cbench_ms2;
EXTERN_pfx unsigned char cbench_x;
EXTERN_pfx unsigned char proceed_with_cbench_interrupt;
EXTERN_pfx unsigned char mbs_station_found;
EXTERN_pfx unsigned char cbench_nxt_rw_data;
EXTERN_pfx void cbench_block_sync(void);

#undef  EXTERN_pfx
```

7.13 FILE6.C

```
/* check for matrix multiplication */
/*****************************************************************************/
#define FILE6_C
/**** Include Files ***/
#include "file6.h"
#include "commnlib.h"
#define bit_0 0x01
#define bit_1 0x02
#define bit_2 0x04
#define bit_3 0x08
#define bit_4 0x10
#define bit_5 0x20
#define bit_6 0x40
#define bit_7 0x80
/**** Constant Definitions ***/
/**** Structure Definitions ***/
/**** Public Variables */
/* look for matrix multiplication ROM - RAM and cycles See if 16 bit math used */
void cbench_block_sync(void)
{
 proceed_with_cbench_interrupt = 0;/* int to do not contiune with interrupt */
 cbench_ms2 = cbench_ms1;
 cbench_ls2 = cbench_ls1;
 cbench_ms1 = cbench_dat[1];
 cbench_ls1 = cbench_dat[2];
 cbench_dat[0] <<= 1;/* rotate datastream through buffer */
 if (cbench_dat[1] & bit_7)
  cbench_dat[0] |= 1;
 cbench_dat[1] <<= 1;
 if (cbench_dat[2] & bit_7)
  cbench_dat[1] |= 1;
 cbench_dat[2] <<= 1;
 if (cbench_dat[3] & bit_7)
  cbench_dat[2] |= 1;
 cbench_dat[3] <<= 1;
 if (TESTBIT(cbench_nxt_rw_data, 0x01))
  cbench_dat[3] |= 1;/* put new cbench bit into 26 bit buffer */
 CLRBIT(cbench_nxt_rw_data, 0x01);
 cbench_bit_ctr++;/* increment bit counter */
 if ((TESTBIT(cbench_sync_flags, 0x01)) || (TESTBIT(cbench_sync_flags, 0x04)))
 {
  if (cbench_bit_ctr >= 25)
  {
   proceed_with_cbench_interrupt = 1;
  }
 }
 else
 {
  if (cbench_bit_ctr >= 26)
  {
   proceed_with_cbench_interrupt = 1;
  }
 }
```

```
if (proceed_with_cbench_interrupt)
{
 /* 10 bit checkword */
 cbench_x = cbench_dat[0] & 0x03;/* upper 2 bits of checkword */
 cbench_syn[1] = cbench_dat[1];/* lower 8 bits of checkword */
 if (cbench_dat[3] & bit_0)
 {
  cbench_syn[1] ^= 0x1b;
  cbench_x ^= 0x03;
 }                                                    /* parity check matrix multiplication */
 if (cbench_dat[3] & bit_1)
 {
  cbench_syn[1] ^= 0x8f;
  cbench_x ^= 0x03;
 }
 if (cbench_dat[3] & bit_2)
 {
  cbench_syn[1] ^= 0xa7;
  cbench_x ^= 0x02;
 }
 if (cbench_dat[3] & bit_4)
 {
  cbench_syn[1] ^= 0xee;
  cbench_x ^= 0x01;
 }
 if (cbench_dat[3] & bit_5)
 {
  cbench_syn[1] ^= 0xdc;
  cbench_x ^= 0x03;
 }
 if (cbench_dat[3] & bit_6)
 {
  cbench_syn[1] ^= 0x01;
  cbench_x ^= 0x02;
 }
 if (cbench_dat[3] & bit_7)
 {
  cbench_syn[1] ^= 0xbb;
  cbench_x ^= 0x01;
 }
 if (cbench_dat[2] & bit_0)
 {
  cbench_syn[1] ^= 0x76;
  cbench_x ^= 0x03;
 }
 if (cbench_dat[2] & bit_1)
 {
  cbench_syn[1] ^= 0x55;
  cbench_x ^= 0x03;
 }
 if (cbench_dat[2] & bit_2)
 {
  cbench_syn[1] ^= 0x13;
  cbench_x ^= 0x03;
 }
```

```
  if (cbench_dat[2] & bit_3)
  {
   cbench_syn[1] ^= 0x9f;
   cbench_x ^= 0x03;
  }
  if (cbench_dat[2] & bit_4)
  {
   cbench_syn[1] ^= 0x87;
   cbench_x ^= 0x02;
  }
  if (cbench_dat[2] & bit_6)
  {
   cbench_syn[1] ^= 0x6e;
   cbench_x ^= 0x01;
  }
  if (cbench_dat[2] & bit_7)
  {
   cbench_syn[1] ^= 0xdc;
   cbench_x ^= 0x02;
  }
  cbench_syn[0] = cbench_x;
  if (cbench_dat[3] & bit_3)
   cbench_syn[1] ^= 0xf7;
  if (cbench_dat[2] & bit_5)
   cbench_syn[1] ^= 0xb7;
 }                                                  /* end if cbench_bit_ctr >= 26 */
```

7.14 FILE7.C

```
/* switch statement samples */
#define FILE7_C
/*****************************************************************************
            Include Files
 *****************************************************************************/
#include ”commnlib.h”
#include ”file1.h”
#include ”file2.h”
#include ”file3.h”
#include ”file4.h”
#include ”file5.h”
#include ”file6.h”
#include ”file8.h”
void main(void)
{
 unsigned char main_temp;
 unsigned int main_temp2;

 /* init local variables */
 main_temp = 5;
 main_temp2 = 200;
 /* init all globals from file filet2.h */
 global_1 = 1;
 global_2 = 2;
 global_3 = 3;
```

```
global_4 = 4;
global_5 = 0;
global_6 = 0;
global_7 = 0;
global_8 = 0;
global_9 = 0;
global_10 = 0;
global_11 = 0;
global_12 = 5;
global_13 = 5;
global_14 = 5;
global_15 = 5;
global_16 = 5;
global_17 = 6;
global_18 = 0xff;
global_19 = 0xf0;
global_20 = 0xf1;
global_21 = 0x50;
/* init file6 variables */
cbench_dat[0] = 0xFE;
cbench_dat[1] = 0x54;
cbench_dat[2] = 0xEE;
cbench_dat[3] = 0x11;
cbench_syn[0] = 0x80;
cbench_syn[1] = 0x43;
cbench_af_flags = 0;
cbench_sync_flags = 0x78;
cbench_bit_ctr = 25;/* force syndrome calculation */
bad_blk_ctr = 3;
next_block = 1;
max_bad_blocks = 8;
cbench_ls1 = 77;
cbench_ms1 = 0x45;
cbench_ls2 = 0x32;
cbench_ms2 = 0x31;
cbench_x = 1;
proceed_with_cbench_interrupt = 1;
mbs_station_found = 1;
cbench_nxt_rw_data = 1;
/* use functions in file 3.c */
blank_selected_buffer(&sample_array[0], sizeof sample_array);
fill_byte_array(&sample_array2[0], sizeof sample_array2, 0x50);
copy_array(&sample_array2[0], &sample_array[0], sizeof sample_array2);
announce_event(EVENT_ONE, 0);
announce_event(EVENT_TWO, 6);
announce_event(EVENT_THREE, 25);
announce_event(EVENT_FOUR, 13);
announce_event(EVENT_FIVE, 50);
announce_event(EVENT_SIX, 125);
announce_event(EVENT_SEVEN, 250);
announce_event(EVENT_EIGHT, 5);
announce_event(EVENT_NINE, 12);
announce_event(EVENT_TEN, 25);
announce_event1(EVENT_ELEVEN);
announce_event1(EVENT_TWELVE);
```

```
announce_event(EVENT_THIRTEEN, 7);
announce_event(EVENT_FOURTEEN, 25);
announce_event(EVENT_FIFTEEN, 25);
announce_event(EVENT_SIXTEEN, 25);
cbench_block_sync();
global_3 = file2_function5();
file2_function8();
file2_function9();
file2_function10();
if (file2_function11(2) == 1)/* look for optimization with return value of function */
{
 global_19 = file2_function12(5);
}
file2_function1(15);
file2_function2(34);
file2_function3(3);
file2_function4();
main_temp = check_if_array_same(&sample_array[0], &sample_array2[0], 25);
if (main_temp == ARRAYS_DIFFERENT)
{
 audio_process();
}
else
{
 convert_hex_2_bcd(45);/* result in sample_array4 */
}
convert_hex_2_bcd(5);
audio_process();
main_temp2 = generate_crc(&sample_array[0], sizeof sample_array, main_temp2);
sample_array2[0] = DOLLAR_SIGN;
sample_array2[5] = LOWERCASE_A;
sample_array2[9] = LOWERCASE_Z;
convert_check(&sample_array2[0], sizeof sample_array2);
global_1 = cbench_CASE_5;
file5_function1();
global_2 = cbench_CASE_3;
file5_function2();
global_7 = cbench_CASE_10;
file5_function3();
file5_function4();/* uses global_17 */
global_19 = cbench_CASE_2;
file5_function5();
global_4 = cbench_CASE_9;
file5_function6();
file8_function1();
file8_function2();
file8_function3();
file8_function4();
file8_function5();
file8_function6();
cbench_dat[0] = 0x0E;
cbench_dat[1] = 0x04;
cbench_dat[2] = 0x1E;
cbench_dat[3] = 0x21;
cbench_block_sync();
```

```
 cbench_block_sync();
 cbench_block_sync();
 cbench_block_sync();
}
```

7.15 FILE8.H

```
/*****************************************************************************
             Include Files
*****************************************************************************/
#ifndef FILE8_H
#define FILE8_H
#ifdef FILE8_C
#define EXTERN_pfx
#else
#define EXTERN_pfx extern
#endif
/*****************************************************************************
             Public Constant Definitions
 *****************************************************************************/
EXTERN_pfx void file8_function1(void);
EXTERN_pfx void file8_function2(void);
EXTERN_pfx void file8_function3(void);
EXTERN_pfx void file8_function4(void);
EXTERN_pfx void file8_function5(void);
EXTERN_pfx void file8_function6(void);

#undef  EXTERN_pfx
```

7.16 FILE8.C

```
/* switch statement samples */
#define file8_C
/*****************************************************************************
              Include Files
 *****************************************************************************/
#include ”commnlib.h”
#include ”file2.h”
#include ”file5.h”
#include ”file8.h”
/* This file has if equivalent of switches in file 5. Compare ROM in both */
 /* look for optimization in first two cases. Cases in sequence also. See if switch table used
*/
void file8_function1(void)
{
 if (global_1 == cbench_CASE_1)
  file2_function1(1);
 else if (global_1 == cbench_CASE_2)
  file2_function1(1);
 else if (global_1 == cbench_CASE_3)
  file2_function2(5);
 else if (global_1 == cbench_CASE_4)
  file2_function3(5);
 else if (global_1 == cbench_CASE_5)
```

```
  file2_function4();
 else
 {
  file2_function4();
 }
}
/* cases out of order. Verify how much ROM compared to function1 */
void file8_function2(void)
{
 if (global_2 == cbench_CASE_2)
  file2_function1(1);
 else if (global_2 == cbench_CASE_1)
  file2_function1(1);
 else if (global_2 == cbench_CASE_4)
  file2_function3(5);
 else if (global_2 == cbench_CASE_3)
  file2_function2(5);
 else if (global_2 == cbench_CASE_5)
  file2_function4();
 else
 {
  file2_function4();
 }
}
/* several cases. Reverse order */
void file8_function3(void)
{
 if (global_7 == cbench_CASE_12)
  file2_function1(10);
 else if (global_7 == cbench_CASE_11)
  file2_function1(1);
 else if (global_7 == cbench_CASE_10)
 {
  file2_function2(3);
  file2_function1(1);
  file2_function3(10);
 }
 else if (global_7 == cbench_CASE_9)
  file2_function2(5);
 else if (global_7 == cbench_CASE_8)
  file2_function3(5);
 else if (global_7 == cbench_CASE_5)
  file2_function4();
 else
  file2_function4();
}
/* just two cases here. Check if it is more efficient than if */
void file8_function4(void)
{
 if (global_17 == cbench_CASE_12)
  file2_function1(10);
 else
 {
  file2_function4();
 }
```

```
}
/* 3 cases. Check if it is efficient to use switch or if */
void file8_function5(void)
{
 if (global_19 == cbench_CASE_1)
  file2_function1(10);
 else if (global_19 == cbench_CASE_2)
 {
  if ((global_9) && (global_1 & 0x01))
  {
   file2_function4();
  }
 }
 else
  file2_function4();
}
/* 12 cases. Look for switch efficiency */
void file8_function6(void)
{
 if (global_4 == cbench_CASE_1)
  file2_function1(1);
 else if (global_4 == cbench_CASE_2)
  file2_function1(1);
 else if (global_4 == cbench_CASE_3)
  file2_function2(5);
 else if (global_4 == cbench_CASE_4)
  file2_function3(5);
 else if (global_4 == cbench_CASE_5)
  file2_function4();
 else if (global_4 == cbench_CASE_6)
  file2_function6();
 else if (global_4 == cbench_CASE_7)
  file2_function7();
 else if (global_4 == cbench_CASE_8)
  file2_function8();
 else if (global_4 == cbench_CASE_9)
  file2_function9();
 else if (global_4 == cbench_CASE_10)
  file2_function10();
 else if (global_4 == cbench_CASE_11)
  file2_function8();
 else if (global_4 == cbench_CASE_12)
  file2_function9();
 else
  file2_function10();
}
```

7.17 FILE9.C

```
/* switch statement samples */
#define file9_C
/*****************************************************************************
             Include Files
 *****************************************************************************/
```

```
#include ”commnlib.h”
#include ”file2.h”
#include ”file5.h”
/* signed arithmetic in this file */
signed char local_1;
signed char local_2;
signed int local_3;
signed int local_4;
void file9_function1( void );
unsigned int file9_function2( void );
unsigned char file9_function3( unsigned int ref_time, unsigned int desired_interval );
void file9_function1( void )
{
 local_1 = 55 - 100;
 local_2 = ( 9 - 24 ) * 2;
 local_3 = 2000 - 5000;
 local_4 = ( 3 - 7 ) / 5;
}
unsigned int file9_function2( void )
{
 return( local_3 );           /*round down*/
}
unsigned char file9_function3( unsigned int ref_time, unsigned int desired_interval )
{
 if(( unsigned int )( local_3 - ref_time ) >= desired_interval )
  return( 1 );
 return( 0 );
}
```

"THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS."