

AN1055

M6805 16-Bit Support Macros

If your microcontroller (MCU) application requires a small amount of program memory and not much raw computing power, the M6805 MCU Family is a most logical choice, given their low cost. But do not cross the M6805 Family off your selection list just because you need some 16-bit indexing and/or 16-bit operations, such as the higher cost M68HC11 Family provides. While the 8-bit X index register of the M6805 Family cannot access the entire M6805 12/13-bit address space and its single 8-bit accumulator cannot directly do 16-bit operations, advanced software techniques can be employed to work around the limitations of the M6805 Family. This application note gives specific details and examples of these techniques.

The code samples given here are available in source code form on the Motorola FREEWARE Bulletin Board Service (BBS), by telephoning 512/891-FREE (512/891-3733). The FREEWARE line operates continuously (except for maintenance) at 300-2400 baud, 8 data bits, 1 stop bit, and no parity. Sample test files are also included. Download the archive file, MACROS05.ARC, to get all the files. All files are suitable for use with the Motorola Development Systems M6805 Portable Assembler for MS-DOS, known as PASM05. Other assemblers may also be supported, but consult the BBS for details.

The techniques used here involve a combination of macros and RAM-based subroutines which use instruction modification. Macros allow programming on a higher level of thought with less chance for introducing errors.

Instruction modification is a technique of altering an instruction just prior to its execution. The modification requires that the instruction be in RAM and can involve the opcode and/or the operand portion of the instruction. By determining the exact instruction/operand needed in advance of execution, greater efficiency in execution speed and code size can be obtained. There is a significant risk in using the instruction modification technique because if used improperly, the program will either fail to work properly at best, or crash at worst. When the technique is used, great care must be exercised to ensure correct operation in all possible cases.

To illustrate the instruction modification technique, consider the following instruction at location \$0050 in RAM memory.

```
0050 C6 04FF LDA $4FF
```

The LDA instruction consists of three bytes: an opcode byte (\$C6) and two bytes that hold the extended address (\$04FF) of the operand. When executed, the A accumulator will be loaded with the contents of location \$04FF. Now consider what happens when the following instructions at location \$870 are executed and how the previous memory locations are changed.

```
0870 A6 05 LDA #$05
0872 B7 51 STA $51
0874 A6 2C LDA #$2C
0872 B7 52 STA $52
```

These instructions store \$05 into location \$0051 and \$2C into location \$0052, which is the operand address of the opcode byte at location \$0050. This has the effect of changing the instruction at location \$0050 to the following.

```
0050 C6 052C LDA $52C
```

When location \$0050 is executed, the A accumulator will now be loaded with the contents of location \$052C, i.e., the instruction at location \$0050 has been modified! Note that the original source listing will only show "LDA \$4FF", as the instruction is only changed at execution time, not assembly time.

When instruction modification is used in a ROM-based system, the RAM code must be initialized (from the ROM) before being used. This can be as simple as a few LOAD/STORE instructions for a small routine, or a MOVE BLOCK routine may be required for larger routines.

As with all things of value, there is a price to be paid. The price for using these macros is rather small, namely, 23 bytes (16 for RAM subroutines and 7 for local storage) of direct addressing memory, i.e., in the range of \$0-\$00FF. The macros have a small size/execution speed penalty associated with them that varies from zero to 50 percent, depending on the frequency and type of macros used. An estimated typical cost for an entire program with moderate macro usage would be in the 5 to 20 percent range. But this is small cost to pay for error-free code generation in areas of the program where speed is not critical. By judicious usage/choice of macros, the cost can be held closer to the 5 percent range.

Part of this cost is associated with the structured code technique of preserving registers and another part is involved with setting up the proper condition codes. The rest of the cost is inherent in the fact that M6805 MCUs must do extra work in software to simulate the hardware capabilities built in M68HC11 MCUs. Because these macros have been fine-tuned for size and speed efficiencies, the overhead cost of register preservation is typically 4 bytes and the overhead cost of condition code setting is typically 2 to 4 bytes, per macro invocation. It can be as high as 200 percent, as is the case of the MOV.B macro for extended addressing operands (16 versus 8 bytes if no register preservation/condition code initialization is done) or as low as zero percent for direct addressing operands.

The DREG macros have almost no overhead associated with them since the DREG is implemented using the already existing A and X registers. The overhead price for DREG macros is only 2 bytes of local storage (TEMP\$, TEST\$), because the RAM subroutines are only needed for XREG and YREG support. The Add and Subtract DREG macros (ADDD, SUBD) have the most overhead because they must save and restore a working register (A), but even this is rather minimal. If only DREG macros are used, it is estimated that the code size could expand from zero to 5 percent over straight assembly language, depending on how many and which type of DREG macros are used.

The XREG and YREG macros have the most overhead, since they are implemented using RAM memory locations. The Load and Store via XREG/YREG macros (LDAXY, STAXY) have the most overhead when nonzero offset values are used (26 bytes versus 2 bytes for zero offset, or 1300 percent). Thus, nonzero offset usages should be avoided unless absolutely necessary. There is also some inefficiency associated with internally maintaining two copies of each register, but it actually helps in the overall implementation.

The Increment, Decrement, and Move macros (INC.B, INC.W, DEC.B, DEC.W, MOV.B, MOVE.W, MOVE) have zero to high overhead. The DEC.B and INC.B macros have zero overhead when used with direct addressing operands, while the INC.W and DEC.W macros have high overhead due to setting the proper condition code based on the resultant 16-bit value.

To use these 16-bit macros, here is a quick summary of the steps involved for use on an IBM-PC with the Development Systems PASM05 macro assembler, which should already have been installed per the instructions supplied with the product. The sample MS-DOS commands are shown in upper case for clarity only (except as noted), as MS-DOS accepts either upper or lower case.

1. Create a new directory for your project and change directory to it as shown below.

```
C>MKDIR PROJ
C>CD PROJ
```

2. Download the archive file, MACROS05.ARC, from the Motorola FREeware Bulletin Board to your project directory and then de-archive it as shown below. See the FREeware bulletin file, archive.bul, for de-archiving information.

```
C>PKARC MACROS05.ARC
```

3. Read the READ.ME file first, and then read the Notes header in the MACROS05.MAC and RAMSBR.INI files. In the MACROS05.MAC file, study the individual macro headers of the macros you intend to use, especially the examples shown.

4. Write your source code using the example shown in the Notes header of the MACROS05.MAC file, as illustrated in Listing 1, lines 1030-1460. If the two ORG statements, lines 1130 and 1180, are not valid for your application's memory map, change them to the appropriate values. Notice especially how the initialization of the RAM subroutines is accomplished by the MOVE macro in line 1200 using the ROM code generated by the INCLUDE statement in line 1400. Or you can expand the BBS EXAMPLE.S file into your source file by first making a copy of it and then editing the copy as shown below (PE is the IBM Personal Editor command, but you can use any editor you feel comfortable with).

```
C>COPY EXAMPLE.S MYFILE.S
C>PE MYFILE.S
```

5. Invoke PASM05 to assemble your new source file as shown below. Because the options for PASM05 are case sensitive, be sure to enter them just as shown.

```
C>PASM05 -eq -l MYFILE.LST MYFILE.S
```

This produces listing file MYFILE.LST and a COFF relocatable object file MYFILE.O.

6. To produce an absolute S-record object file suitable for programming an EPROM, the COFF relocatable object file must first be linked to an absolute object file (MYFILE.OUT) and then converted to an S-record file (MYFILE.MX), which many commercial EPROM programmers, such as Data I/O, recognize. Enter the commands shown below to create the S-record file.

```
C>PLD -o MYFILE.OUT MYFILE.O
C>UBUILDS MYFILE.OUT
```

Steps 5 and 6 can be simplified by copying and editing the BBS batch file, MAKE1.BAT, in a manner similar to that shown in Step 4, except here we want the new batch file, MAKE.BAT, to process MYFILE instead of TEST1. The resultant MAKE.BAT file should be similar to the text shown below. The two DEL commands at the end are optional, as these files are no longer needed.

```
pasm05 -eq -l myfile.lst myfile.s
pld -o myfile.out myfile.o
ubuilds myfile.out
del myfile.o
del myfile.out
```

To accomplish Steps 5 and 6, invoke the batch file MAKE.BAT by simply typing its name as shown below. MS-DOS will execute each line of the file as if you had typed it on the keyboard.

```
C>MAKE
```

7. Consult your EPROM programmer's user manual for how to load the resultant S-record object file (MYFILE.MX) and program your chosen device.

Because of the length of the source listings involved and because these listings are intended to be self-explanatory, the rest of this discussion will only deal with salient points which might need clarification for the reader. Line numbers have been added at the beginning of each line for identification purposes in this Application Note, i.e., they are not present in the actual source file.

These macros have been written so that any syntax error will result in falling to the end of the macro where the FAIL directive will force a "Macro syntax error detected!" message. Proper usage results in exiting the macro early via the MEXIT directive and thus avoids the FAIL directive at the end.

The file header of **Listing 1** (lines 20-1800) gives specific details of the macros supported and their general operation and restrictions. Line 1810 defines where a seven byte block of low memory will be allocated for support of these macros (see lines 20430-20490). Line 1830 disables the output listing to avoid repetitious output, while line 19630 re-enables the listing.

A lot of conditionals (IFxx) involve testing to see if direct addressing can be used, as it is more efficient (one less byte and one less execution cycle per instruction) than extended addressing, as illustrated by lines 2090-2210. Also, conditionals check to see if shorter instruction forms can be used, like "CLRA" instead of "LDA #0", as in these same lines. For this same reason, it is most efficient to place the RAM subroutines, and thus the XREG and YREG, into direct addressing space (\$0000-\$00BF). Remember that the M6805 uses a fixed stack area in direct addressing space of \$00C0-\$00FF.

The COMPARE macros (CPD, CPXR, CPYR) use a shortened form when immediate addressing is specified and either

of the halves is zero, i.e., no "CMP #0" instruction is generated. Lines 3480-3660 are typical of this technique. As always, testing for zero is most efficient in computer architecture.

The 16-BIT INDEXING macros (LDAXY, STAXY) are most efficient when used with a zero offset, as typified in lines 3980-4060, but will function with any sized offset (lines 4070-4210). Nonzero offsets must first be added to the indexing register (XREG or YREG), the operation performed via the appropriate RAM subroutine, and lastly, the indexing register must be restored to its original value.

Lines 19640-20410 comprise the RAM subroutines which are the keystone of the macros using the XREG and YREG pseudo 16-bit index registers. It is here that the XREG and YREG pseudo-registers are defined and stored as the second and third bytes of extended addressing LDA and STA instructions using EQU directives (lines 19910-19920, 20060-20070, 20210-20220, and 20360-20370). Instructions that store values to XREG or YREG will thus modify the instruction's effective address, hence the name instruction modification. These RAM subroutines must be initialized before they can be used, and so a mirror image of this code with altered labels is provided in the RAMSBR.INI file (as seen in **Listing 2**). This file can then be simply INCLUDED in the ROM section of the user's code and copied to RAM via the MOVE macro, as discussed in lines 19660-19760 of **Listing 1**.

Listing 2 is the RAM subroutine initialization file and is essentially a copy of the RAM subroutines defined in lines 19640-20410 of **Listing 1**, but the labels have all been prefixed with a period (.). This allows using the MOVE macro to copy this RAM initialization code from a ROM to RAM (see lines 370-410).

Lines 1070-1090 verify that the number of RAM subroutine initialization bytes is identical.

Freescale Semiconductor, Inc.

Listing 1 - MACROS05.MAC File

```
00010 PAGE
00020 *****
00030 * macros05.mac 1.0
00040 * -----
00050 * Module Name: MACROS05 - M6805 Macros
00060 * -----
00070 *
00080 * Description:
00090 * This file contains macros and subroutines to support pseudo-registers
00100 * on the 6805 that simulate registers and addressing modes available on
00110 * the 68HC11. It is suitable for "black box" operation, i.e., the
00120 * macros may be used without knowledge of how they work. A list of the
00130 * supported macros follows. Consult the individual macro headers for
00140 * usage details and see the "Notes" below.
00150 *
00160 * LDD Load DREG
00170 * STD Store DREG
00180 * ADDD Add DREG
00190 * SUBD Add DREG
00200 * CPD Compare DREG
00210 * LDAXY Load A via 16-bit pseudo-register (XREG or YREG)
00220 * STAXY Store A via 16-bit pseudo-register (XREG or YREG)
00230 * LDXR Load XREG
00240 * STXR Store XREG
00250 * INCXR Increment XREG
00260 * DECXR Decrement XREG
00270 * CPXR Compare XREG
00280 * LDYR Load YREG
00290 * STYR Store YREG
00300 * INCYR Increment YREG
00310 * DECYR Decrement YREG
00320 * CPYR Compare YREG
00330 * DEC.B Decrement byte
00340 * DEC.W Decrement word
00350 * INC.B Increment byte
00360 * INC.W Increment word
00370 * MOV.B Move byte
00380 * MOV.W Move word
00390 * MOVE Move block of memory
00400 *
00410 * General Information:
00420 * The following pseudo-registers are supported.
00430 * DREG = pseudo 16-bit accumulator (A,X registers, A is MS half)
00440 * XREG = pseudo 16-bit index register
00450 * YREG = pseudo 16-bit index register
00460 *
00470 * The following terms are used.
00480 * # = specifies immediate addressing mode
00490 * <address> = address/value operand (absolute or immediate)
00500 * <offset> = unsigned 16-bit offset for indexed addressing
00510 *
00520 * Notes:
00530 * 1. Motorola reserves the right to make changes to this file.
00540 * Although this file has been carefully reviewed and is believed
00550 * to be reliable, Motorola does not assume any liability arising
00560 * out of its use. This code may be freely used and/or modified at
00570 * no cost or obligation by the user.
00580 * 2. This file was made for use with the Motorola Development Systems
00590 * MC6805 Portable Assembler/Linker for MS-DOS, known as PASM05 and
00600 * PLD, as released on 82HCVBASM B* and 82HCVBLNK B*. Consult the
00610 * PASM and PLD reference manuals, part numbers M68HASM/D1 and
00620 * M68HLINK/D1 for more details.
```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```

00630 *      3. These macros were made for ABSOLUTE assemblies only, i.e., for
00640 *      use with ORG directives. While most of the macro concepts will
00650 *      work in relocatable assemblies (BSCT, DSCT, PSCT, ASCT, XDEF,
00660 *      and XREF), errors will be generated because PASM limits the use
00670 *      of external symbols in expressions and because the value of an
00680 *      expression must be known at assembly time for IFxx directives to
00690 *      assemble the proper code. The first restriction is a result of
00700 *      limitations in the COFF object file format. If it is desired to
00710 *      have these macros work with relocatable assemblies, modifica-
00720 *      tions similar to below should be made, but be forewarned of the
00730 *      increased inefficiencies in size and speed. Consider the
00740 *      following code to change the LDD macro so that an XREF parameter,
00750 *      \1, can be loaded as an immediate value.
00760 *              LDA      \.8
00770 *              LDX      \.8+1
00780 *              BRA      \.9
00790 *              \.8     FDB      \1
00800 *              \.9     EQU      *
00810 *
00820 *      4. In order to efficiently support both LOAD and STORE operations
00830 *      for the pseudo 16-bit index registers, there are actually two
00840 *      such "registers", i.e., one for LOAD and one for STORE as
00850 *      defined below. These routines maintain both "registers" with
00860 *      the same value, and so the programmer may think of them as one
00870 *      register.
00880 *      XREG1$ = 16-bit XREG for LOAD operations
00890 *      XREG2$ = 16-bit XREG for STORE operations
00900 *      YREG1$ = 16-bit YREG for LOAD operations
00910 *      YREG2$ = 16-bit YREG for STORE operations
00920 *
00930 *      5. These macros can be used to create in-line code (speed
00940 *      efficient) or they be placed in a subroutine (byte efficient).
00950 *
00960 *      6. Instruction modified code is used here and is denoted by use
00970 *      of the unique string "0-0" (RAM subroutines).
00980 *
00990 *      7. Some macros use temporary storage locations (TEMPA$, TESTA$,
01000 *      etc.), so these macros should not be used in any interrupt
01010 *      routine in order to avoid corrupted values!
01020 *
01030 *      8. The user must ensure that the code is appropriately placed in
01040 *      the target M6805's memory map, i.e., the RAM subroutines MUST
01050 *      be located in RAM but must not overlap the stack area ($00C0-
01060 *      00FF) unless it can be GUARANTEED there is no conflict! See
01070 *      LO$MEM below to set low memory data storage area!
01080 *
01090 *      9. To use this file, either use an INCLUDE statement or just
01100 *      merge this file into your source code file. Consult your
01110 *      assembler's user's manual for the details specific to your
01120 *      situation. When using a ROM controlled system, the MOVE
01130 *      macro should be used to copy the RAM subroutines from ROM to
01140 *      RAM (see the comments after where RAMSBR$ is defined below
01150 *      and note the INCLUDE statement for the RAMSBR.INI file).
01160 *      Reference the code segment example below for usage ideas
01170 *      (shown in PASM05 for MS-DOS syntax).
01180 *
01190 *      ORG      $50
01200 *      TOTAL   RMB      2
01210 *      RTABLE  RMB      5
01220 *      INCLUDE MACROS05.MAC
01230 *
01240 *      ORG      $400
01250 *      RESET   RSP
01260 *      MOVE    #, .RAMSBR$, #, RAMSBR$, #, RAMSZ$ Init ram.
01270 *      START  MOV.W   #, 0, TOTAL
01280 *      LDD     COST
01290 *      ADDD   #, 1000
01300 *      SUBD   #, ADJUST
01310 *      ADDD   TOTAL
01320 *      STDD   TOTAL

```

Freescale Semiconductor, Inc.

```

01270 *          CPD      #,1500
01280 *          BEQ      MATCH
01290 *          *
01300 *          LDXR     #,0
01310 *          LDYR     #,0
01320 *          LOOP   LDAXY   TABLE,XREG
01330 *          STAXY   RTABLE,YREG
01340 *          INCXR
01350 *          INCYR
01360 *          CPY      #,5
01370 *          BNE     LOOP
01380 *          .
01390 *          .
01400 *          INCLUDE RAMSBR.INI
01410 *          TABLE FCB     1,2,3,4,5
01420 *          ADJUST FDB     150
01430 *          COST   FDB     859
01440 *          .
01450 *          .
01460 *          END
01470 *
01480 * 10. To assemble, use the following sample invocation lines:
01490 *      pasm05 -eq -l filename.lst filename.s      (debug= expanded)
01500 *      or
01510 *      pasm05 -bf -l filename.lst filename.s      (black box)
01520 * 11. Notations used by PASM05 are as follows:
01530 *      OPERATORS: Special two character operators used are...
01540 *          1.Logical AND          !.
01550 *          2.Shift Right (0 fill on left)  !>
01560 *      MACROS:      Special notations used are...
01570 *          1.Parameters are positionally named using \0,
01580 *             \1, \2, etc.
01590 *          2.Labels within macros are designated via \.a,
01600 *             where "a" is an alphanumeric character. The
01610 *             assembler generates a unique label to avoid
01620 *             multiply defined label problems.
01630 * 12. Some macros access 16-bit values as LS byte then MS byte in order
01640 *      to be more efficient for condition code (CC) setting. This is
01650 *      the reverse order that the 68HC11 would access the two byte
01660 *      halves. This difference would only be a concern in accessing
01670 *      hardware registers, as normal RAM makes no difference. Those
01680 *      macros with this difference have an entry in their Notes section.
01690 * 13. The latest version of this file is maintained on the Motorola
01700 *      FREEMEMO Bulletin Board, 512/891-FREE (512/891-3733). It operates
01710 *      continuously (except for maintenance) at 1200-2400 baud, 8 bits,
01720 *      no parity. Sample test files for PASM05 are also included.
01730 *      Download the archive file, MACROS05.ARC, to get all the files.
01740 *
01750 * *****
01760 * REVISION HISTORY (add new changes to top):
01770 *
01780 * 05/16/90 P.S. Gilmour
01790 * 1. Created for Application Note AN1055.
01800 * *****
01810 LO$MEM EQU $00C0-7 Low memory ($0000-00FF) storage (7 bytes)
01820

```

Freescale Semiconductor, Inc.

```
01830          OPT      NOL
01840 *****
01850 * LDD      = load  DREG
01860 *      LDD      [#,<address>
01870 *
01880 * Examples:
01890 *      1. "LDD  #,START"      puts the value of symbol 'START' into
01900 *                                the DREG (=A,X).
01910 *      2. "LDD  START"        puts the contents of location 'START'
01920 *                                and 'START'+1 into the DREG (=A,X).
01930 *
01940 * Register Usage:
01950 *      A,X = loaded with new value (DREG).
01960 *      CC = reflects MS half (=A).
01970 *      All other registers preserved.
01980 *
01990 * Notes:
02000 *      1. Byte access order is LS, then MS (reversed from 68HC11).
02010 *
02020 LDD      MACR
02030 IFEQ      NARG-1
02040         LDX      (\0)+1
02050         LDA      (\0)
02060         MEXIT
02070 ENDC
02080 IFEQ      NARG-2
02090         IFC      '\0', '#'
02100         IFEQ      (\1)!.$FF
02110         CLRX
02120         ENDC
02130         IFNE      (\1)!.$FF
02140         LDX      #(\1)!.$FF
02150         ENDC
02160         IFEQ      (\1)!>8
02170         CLRA
02180         ENDC
02190         IFNE      (\1)!>8
02200         LDA      #(\1)!>8
02210         ENDC
02220         MEXIT
02230 ENDC
02240 ENDC
02250         FAIL      Macro syntax error detected!
02260         ENDM
02270
02280 *****
02290 * STD      = store DREG
02300 *      STD      <address>
02310 *
02320 * Examples:
02330 *      1. "STD  START"        stores the DREG (=A,X) into locations
02340 *                                'START' and 'START'+1.
02350 *
02360 * Register Usage:
02370 *      CC = reflects MS half (=A).
02380 *      All other registers preserved.
02390 *
02400 * Notes:
02410 *      1. Byte access order is LS, then MS (reversed from 68HC11).
02420 *
02430 STD      MACR
02440         STX      (\0)+1
02450         STA      (\0)
02460         ENDM
02470
```

Freescale Semiconductor, Inc.

```
02480 *****
02490 * ADDD = add DREG
02500 * ADDD [#,<address>
02510 *
02520 * Examples:
02530 * 1. "ADDD #,START" adds the value of symbol 'START' to the
02540 * DREG (=A,X).
02550 * 2. "ADDD START" adds the contents of location 'START'
02560 * and 'START'+1 to the DREG (=A,X).
02570 *
02580 * Register Usage:
02590 * A,X = contains new value (DREG).
02600 * CC = reflects MS half (=A).
02610 * All other registers preserved.
02620 *
02630 ADDD MACR
02640 IFEQ NARG-1
02650 STA TEMPAS
02660 TXA
02670 ADD (\0)+1
02680 TAX
02690 LDA TEMPAS
02700 ADC (\0)
02710 MEXIT
02720 ENDC
02730 IFEQ NARG-2
02740 IFC '\0', '#'
02750 STA TEMPAS
02760 TXA
02770 ADD #(\1)!.$FF
02780 TAX
02790 LDA TEMPAS
02800 ADC #(\1)!>8
02810 MEXIT
02820 ENDC
02830 ENDC
02840 FAIL Macro syntax error detected!
02850 ENDM
02860 *****
02870 *****
02880 * SUBD = add DREG
02890 * SUBD [#,<address>
02900 *
02910 * Examples:
02920 * 1. "SUBD #,START" subtracts the value of symbol 'START' from
02930 * the DREG (=A,X).
02940 * 2. "SUBD START" subtracts the contents of location 'START'
02950 * and 'START'+1 from the DREG (=A,X).
02960 *
02970 * Register Usage:
02980 * A,X = contains new value (DREG).
02990 * CC = reflects MS half (=A).
03000 * All other registers preserved.
03010 *
03020 SUBD MACR
03030 IFEQ NARG-1
03040 STA TEMPAS
03050 TXA
03060 SUB (\0)+1
03070 TAX
03080 LDA TEMPAS
03090 SBC (\0)
03100 MEXIT
03110 ENDC
```


Freescale Semiconductor, Inc.

```

03120 IFEQ NARG-2
03130 IFC '\0', '#'
03140 STA TEMPAS
03150 TXA
03160 SUB #(\1)!.$FF
03170 TAX
03180 LDA TEMPAS
03190 SBC #(\1)!>8
03200 MEXIT
03210 ENDC
03220 ENDC
03230 FAIL Macro syntax error detected!
03240 ENDM
03250
03260 *****
03270 * CPD = compare DREG
03280 * CPD [#,<address>
03290 *
03300 * Examples:
03310 * 1. "CPD #,BLOCKSZ" compares the value of symbol 'BLOCKSZ'
03320 * with the DREG (=A,X).
03330 * 2. "CPD START" compares the contents of location
03340 * 'START' and 'START'+1 with the DREG.
03350 *
03360 * Register Usage:
03370 * CC = reflects DREG comparison (Z-bit only).
03380 * All other registers preserved.
03390 *
03400 CPD MACR
03410 IFEQ NARG-1
03420 CPX (\0)+1
03430 BNE \.0
03440 CMP (\0)
03450 \.0 EQU *
03460 MEXIT
03470 ENDC
03480 IFEQ NARG-2
03490 IFC '\0', '#'
03500 IFEQ (\1)!.$FF
03510 TSTX
03520 ENDC
03530 IFNE (\1)!.$FF
03540 CPX #(\1)!.$FF
03550 ENDC
03560 BNE \.0
03570 IFEQ (\1)!>8
03580 TSTA
03590 ENDC
03600 IFNE (\1)!>8
03610 CMP #(\1)!>8
03620 ENDC
03630 \.0 EQU *
03640 MEXIT
03650 ENDC
03660 ENDC
03670 FAIL Macro syntax error detected!
03680 ENDM
03690

```

Freescale Semiconductor, Inc.

```

03700 *****
03710 * LDAXY = load A via 16-bit pseudo-register (XREG or YREG)
03720 *     LDAXY   <offset>,XREG
03730 *     LDAXY   <offset>,YREG
03740 *
03750 * Examples:
03760 * 1. "LDAXY   0,XREG"      loads the contents of the memory location
03770 *                          specified by XREG+0 into the A accumulator.
03780 * 2. "LDAXY   ,XREG"      loads the contents of the memory location
03790 *                          specified by XREG+0 into the accumulator.
03800 * 3. "LDAXY   TBL,XREG"    loads the contents of the memory location
03810 *                          specified by XREG+'TBL' into the A accum-
03820 *                          ulator.
03830 * 4. Above examples can be repeated with substituting YREG for XREG.
03840 *
03850 * Register Usage:
03860 *   A      = loaded with new value.
03870 *   DREG   = destroyed.
03880 *   CC     = reflects value loaded.
03890 *   All other registers preserved.
03900 *
03910 LDAXY   MACR
03920   IFNC  '\1','XREG'
03930   IFNC  '\1','YREG'
03940   FAIL  Macro syntax error detected!
03950   MEXIT
03960   ENDC
03970 ENDC
03980   IFC  '\0',''
03990   JSR  LDA\1      Default offset= 0
04000   MEXIT
04010 ENDC
04020   IFNC  '\0',''
04030   IFEQ  \0
04040   JSR  LDA\1      Offset= 0
04050   MEXIT
04060 ENDC
04070   IFNE  \0
04080   LDA  \11$+1     Set nREG= offset + nREG
04090   ADD  #(\0)!.$FF
04100   STA  \11$+1
04110   LDA  \11$
04120   ADC  #(\0)!>8
04130   STA  \11$
04140   JSR  LDA\1      Offset= 0
04150   STA  TEMPAS$
04160   LDA  \12$      Restore nREG
04170   STA  \11$
04180   LDA  \12$+1
04190   STA  \11$+1
04200   LDA  TEMPAS$
04210   MEXIT
04220 ENDC
04230 ENDC
04240   FAIL  Macro syntax error detected!
04250   ENDM
04260

```

Freescale Semiconductor, Inc.

```
04270 *****
04280 * STAXY = store A via 16-bit pseudo-register (XREG or YREG)
04290 *     STAXY <offset>,XREG
04300 *     STAXY <offset>,YREG
04310 *
04320 * Examples:
04330 * 1. "STAXY 0,XREG"           stores the accumulator (=A) into the memory
04340 *                             location specified by XREG+0.
04350 * 2. "STAXY ,XREG"           stores the accumulator (=A) into the memory
04360 *                             location specified by XREG+0.
04370 * 3. "STAXY TBL,XREG"       stores the accumulator (=A) into the memory
04380 *                             specified by XREG+'TBL'.
04390 * 4. Above examples can be repeated with substituting YREG for XREG.
04400 *
04410 * Register Usage:
04420 *   CC = reflects value stored.
04430 *   All other registers preserved.
04440 *
04450 STAXY MACR
04460     IFNC  '\1', 'XREG'
04470     IFNC  '\1', 'YREG'
04480         FAIL  Macro syntax error detected!
04490         MEXIT
04500     ENDC
04510 ENDC
04520 IFC  '\0',''
04530     JSR   STA\1           Default offset= 0
04540     MEXIT
04550 ENDC
04560 IFNC  '\0',''
04570     IFEQ  \0
04580     JSR   STA\1           Offset= 0
04590     MEXIT
04600 ENDC
04610 IFNE  \0
04620     STA   TEMPAS$
04630     LDA   \12$+1           Set nREG= offset + nREG
04640     ADD   #(\0)!.$FF
04650     STA   \12$+1
04660     LDA   \12$
04670     ADC   #(\0)!>8
04680     STA   \12$
04690     LDA   TEMPAS$
04700     JSR   STA\1           Offset= 0
04710     LDA   \11$           Restore nREG
04720     STA   \12$
04730     LDA   \11$+1
04740     STA   \12$+1
04750     LDA   TEMPAS$
04760     MEXIT
04770 ENDC
04780 ENDC
04790     FAIL  Macro syntax error detected!
04800     ENDM
04810
```

Freescale Semiconductor, Inc.

```

04820 *****
04830 * LDXR = load XREG
04840 *   LDXR  [#,<address>
04850 *
04860 * Examples:
04870 *   1. "LDXR #,START"           puts the value of symbol 'START' into the
04880 *                                XREG.
04890 *   2. "LDXR START"            puts the contents of location 'START' and
04900 *                                'START'+1 into the XREG.
04910 *
04920 * Register Usage:
04930 *   CC = reflects MS half (=A).
04940 *   All other registers preserved.
04950 *
04960 * Notes:
04970 *   1. Byte access order is LS, then MS (reversed from 68HC11).
04980 *
04990 LDXR  MACR
05000     IFEQ      NARG-1
05010             STA      TEMPAS
05020             LDA      (\0)+1
05030             STA      XREG1$+1
05040             STA      XREG2$+1
05050             LDA      (\0)
05060             STA      XREG1$
05070             STA      XREG2$
05080     IFEQ      XREG1$!. $FF00
05090             LDA      TEMPAS
05100             TST      XREG1$
05110     ENDC
05120     IFNE      XREG1$!. $FF00
05130             STA      TESTAS
05140             LDA      TEMPAS
05150             TST      TESTAS
05160     ENDC
05170     MEXIT
05180     ENDC
05190     IFEQ      NARG-2
05200     IFC       '\0', '#'
05210             IFEQ      XREG1$!. $FF00           ! XREG in low memory?
05220             IFEQ      \1                       ! #0 value?
05230             CLR      XREG1$+1
05240             CLR      XREG2$+1
05250             CLR      XREG1$
05260             CLR      XREG2$
05270     MEXIT
05280     ENDC
05290     IFNE      \1                               ! not #0 value?
05300             STA      TEMPAS
05310     IFEQ      (\1)!. $FF
05320             CLR      XREG1$+1
05330             CLR      XREG2$+1
05340     ENDC
05350     IFNE      (\1)!. $FF
05360             LDA      #(\1)!. $FF
05370             STA      XREG1$+1
05380             STA      XREG2$+1
05390     ENDC
05400     IFEQ      (\1)!>8
05410             CLR      XREG1$
05420             CLR      XREG2$
05430     ENDC

```

Freescale Semiconductor, Inc.

```
05440     IFNE  (\1)!>8
05450     LDA   #(\1)!>8
05460     STA   XREG1$
05470     STA   XREG2$
05480     ENDC
05490     LDA   TEMPAS
05500     TST   XREG1$
05510     MEXIT
05520     ENDC
05530     ENDC
05540     IFNE  XREG1$!.$FF0      ! XREG in high memory?
05550     IFEQ  \1              ! #0 value?
05560     STA   TEMPAS
05570     CLRA
05580     STA   XREG1$+1
05590     STA   XREG2$+1
05600     STA   XREG1$
05610     STA   XREG2$
05620     CLR  TESTAS
05630     LDA   TEMPAS
05640     TST   TESTAS
05650     MEXIT
05660     ENDC
05670     IFNE  \1              ! not #0 value?
05680     STA   TEMPAS
05690     IFEQ  (\1)!.$FF
05700     CLRA
05710     ENDC
05720     IFNE  (\1)!.$FF
05730     LDA   #(\1)!.$FF
05740     ENDC
05750     STA   XREG1$+1
05760     STA   XREG2$+1
05770     IFEQ  (\1)!>8
05780     CLRA
05790     ENDC
05800     IFNE  (\1)!>8
05810     LDA   #(\1)!>8
05820     ENDC
05830     STA   XREG1$
05840     STA   XREG2$
05850     STA   TESTAS
05860     LDA   TEMPAS
05870     TST   TESTAS
05880     MEXIT
05890     ENDC
05900     ENDC
05910     ENDC
05920     ENDC
05930     FAIL  Macro syntax error detected!
05940     ENDM
05950
```

Freescale Semiconductor, Inc.

```
05960 *****
05970 * STXR = store XREG
05980 *   STXR <address>
05990 *
06000 * Examples:
06010 *   1. "STXR START"                stores the XREG into locations 'START' and
06020 *                                   'START'+1.
06030 *
06040 * Register Usage:
06050 *   CC = reflects MS half (=A).
06060 *   All other registers preserved.
06070 *
06080 * Notes:
06090 *   1. Byte access order is LS, then MS (reversed from 68HC11).
06100 *
06110 STXR MACR
06120     STA     TEMPAS
06130     LDA     XREG1$+1
06140     STA     (\0)+1
06150     LDA     XREG1$
06160     STA     (\0)
06170     IFEQ   XREG1$!. $FF00
06180     LDA     TEMPAS
06190     TST    XREG1$
06200     ENDC
06210     IFNE   XREG1$!. $FF00
06220     STA     TESTAS
06230     LDA     TEMPAS
06240     TST    TESTAS
06250     ENDC
06260     ENDM
06270
06280 *****
06290 * INCRX = increment XREG
06300 *   INCRX [[#,]<address>]
06310 *
06320 * Examples:
06330 *   1. "INCRX"                      adds one (1) to the XREG.
06340 *   2. "INCRX #,START"              adds the value of symbol 'START' to the
06350 *                                   XREG.
06360 *   3. "INCRX START"                adds the contents of location 'START' and
06370 *                                   'START'+1 to the XREG.
06380 *   4. "INCRX ! comment"            adds one (1) to the XREG (comment present!).
06390 *
06400 * Register Usage:
06410 *   CC = reflects value incremented (Z-bit only).
06420 *   All other registers preserved.
06430 *
06440 * Notes:
06450 *   1. Explicit comment character (!) MUST be used when comment field is
06460 *      present to prevent confusion with parameters!
06470 *   2. Assumes XREG1$ = XREG2$.
06480 *   3. When parameters are present, this macro becomes "ADD to XREG".
06490 *
06500 INCRX   MACR
06510     IFEQ   NARG
06520     IFEQ   XREG1$!. $FF00
06530     INC    XREG1$+1
06540     INC    XREG2$+1
06550     BNE    \.0
06560     INC    XREG1$
06570     INC    XREG2$
06580 \.0    EQU    *
06590     MEXIT
06600     ENDC
```

Freescale Semiconductor, Inc.

```
06610      IFNE  XREG1$!.$FF00
06620          STA  TEMPAS
06630          LDA  XREG1$+1
06640          ADD  #1
06650          STA  XREG1$+1
06660          STA  XREG2$+1
06670          LDA  XREG1$
06680          ADC  #0
06690          STA  XREG1$
06700          STA  XREG2$
06710          ORA  XREG1$+1
06720          STA  TESTAS
06730          LDA  TEMPAS
06740          TST  TESTAS
06750          MEXIT
06760      ENDC
06770      ENDC
06780      IFEQ  NARG-1
06790          STA  TEMPAS
06800          LDA  XREG1$+1
06810          ADD  (\0)+1
06820          STA  XREG1$+1
06830          STA  XREG2$+1
06840          LDA  XREG1$
06850          ADC  \0
06860          STA  XREG1$
06870          STA  XREG2$
06880          ORA  XREG1$+1
06890          STA  TESTAS
06900          LDA  TEMPAS
06910          TST  TESTAS
06920          MEXIT
06930      ENDC
06940      IFEQ  NARG-2
06950          IFC  '\0', '#'
06960          STA  TEMPAS
06970          LDA  XREG1$+1
06980          ADD  #(\1)!.$FF
06990          STA  XREG1$+1
07000          STA  XREG2$+1
07010          LDA  XREG1$
07020          ADC  #(\1)!>8
07030          STA  XREG1$
07040          STA  XREG2$
07050          ORA  XREG1$+1
07060          STA  TESTAS
07070          LDA  TEMPAS
07080          TST  TESTAS
07090          MEXIT
07100      ENDC
07110      ENDC
07120          FAIL  Macro syntax error detected!
07130          ENDM
07140
```

Freescale Semiconductor, Inc.

```
07150 *****
07160 * DECKR = decrement XREG
07170 *   DECKR [[#,]<address>]
07180 *
07190 * Examples:
07200 *   1. "DECKR"                subtracts one (1) from the XREG.
07210 *   2. "DECKR #,START"      subtracts the value of symbol 'START' from
07220 *                             the XREG.
07230 *   3. "DECKR START"         subtracts the contents of location 'START'
07240 *                             and 'START'+1 from the XREG.
07250 *   4. "DECKR ! comment"    subtract one from the XREG (comment present!).
07260 *
07270 * Register Usage:
07280 *   CC = reflects value decremented (Z-bit only).
07290 *   All other registers preserved.
07300 *
07310 * Notes:
07320 *   1. Explicit comment character (!) MUST be used when comment field is
07330 *       present!
07340 *   2. Assumes XREG1$ = XREG2$.
07350 *   3. When parameters are present, this macro becomes "SUBTRACT from XREG".
07360 *
07370 DECKR MACR
07380   IFEQ   NARG
07390       STA   TEMPAS
07400       LDA   XREG1$+1
07410       SUB   #1
07420       STA   XREG1$+1
07430       STA   XREG2$+1
07440       LDA   XREG1$
07450       SBC   #0
07460       STA   XREG1$
07470       STA   XREG2$
07480       ORA   XREG1$+1
07490       STA   TESTAS
07500       LDA   TEMPAS
07510       TST   TESTAS
07520       MEXIT
07530 ENDC
07540   IFEQ   NARG-1
07550       STA   TEMPAS
07560       LDA   XREG1$+1
07570       SUB   (\0)+1
07580       STA   XREG1$+1
07590       STA   XREG2$+1
07600       LDA   XREG1$
07610       SBC   \0
07620       STA   XREG1$
07630       STA   XREG2$
07640       ORA   XREG1$+1
07650       STA   TESTAS
07660       LDA   TEMPAS
07670       TST   TESTAS
07680       MEXIT
07690 ENDC
```


Freescale Semiconductor, Inc.

```

07700 IFEQ NARG-2
07710 IFC '\0', '#'
07720 STA TEMPAS
07730 LDA XREG1$+1
07740 SUB #(\1)!.$FF
07750 STA XREG1$+1
07760 STA XREG2$+1
07770 LDA XREG1$
07780 SBC #(\1)!>8
07790 STA XREG1$
07800 STA XREG2$
07810 ORA XREG1$+1
07820 STA TESTAS
07830 LDA TEMPAS
07840 TST TESTAS
07850 MEXIT
07860 ENDC
07870 ENDC
07880 FAIL Macro syntax error detected!
07890 ENDM
07900
07910 *****
07920 * CPXR = compare XREG
07930 * CPXR [#,<address>
07940 *
07950 * Examples:
07960 * 1. "CPXR #,BLOCKSZ" compares the value of symbol 'BLOCKSZ'
07970 * with the XREG.
07980 * 2. "CPXR START" compares the contents of location
07990 * 'START' and 'START'+1 with the XREG.
08000 *
08010 * Register Usage:
08020 * CC = reflects XREG comparison (Z-bit only).
08030 * All other registers preserved.
08040 *
08050 CPXR MACR
08060 IFEQ NARG-1
08070 STA TEMPAS
08080 BSET 0,TESTAS Preset for .NE. condition!
08090 LDA XREG1$+1
08100 CMP (\0)+1
08110 BNE \.0 Branch if LS half is .NE.
08120 LDA XREG1$
08130 CMP (\0)
08140 BNE \.0 Branch if MS half is .NE.
08150 CLR TESTAS Set for .EQ. condition!
08160 \.0 LDA TEMPAS
08170 TST TESTAS Set proper Z-bit (.EQ. or .NE.)!
08180 MEXIT
08190 ENDC

```

Freescale Semiconductor, Inc.

```

08200   IFEQ   NARG-2
08210     IFC   '\0','#'
08220     IFEQ   \1
08230       IFEQ   XREG1$!.$FF00
08240         TST   XREG1$+1
08250         BNE   \.0           Branch if LS half is .NE.
08260         TST   XREG1$
08270   \.0   EQU   *
08280         MEXIT
08290       ENDC
08300     IFNE   XREG1$!.$FF00
08310       STA   TEMPAS$
08320       BSET   0,TESTAS$       Preset for .NE. condition!
08330       LDA   XREG1$+1
08340       BNE   \.0           Branch if MS half is .NE.
08350       LDA   XREG1$
08360       BNE   \.0           Branch if MS half is .NE.
08370       CLR   TESTAS$         Set for .EQ. condition!
08380   \.0   LDA   TEMPAS$
08390       TST   TESTAS$         Set proper Z-bit (.EQ. or .NE.!)
08400       MEXIT
08410     ENDC
08420   ENDC
08430     STA   TEMPAS$
08440     BSET   0,TESTAS$       Preset for .NE. condition!
08450     LDA   XREG1$+1
08460     IFNE   (\1)!.$FF
08470     CMP   #(\1)!.$FF
08480     ENDC
08490     BNE   \.0           Branch if LS half is .NE.
08500     LDA   XREG1$
08510     IFNE   (\1)!>8
08520     CMP   #(\1)!>8
08530     ENDC
08540     BNE   \.0           Branch if MS half is .NE.
08550     CLR   TESTAS$         Set for .EQ. condition!
08560   \.0   LDA   TEMPAS$
08570     TST   TESTAS$         Set proper Z-bit (.EQ. or .NE.!)
08580     MEXIT
08590   ENDC
08600 ENDC
08610     FAIL   Macro syntax error detected!
08620     ENDM
08630

```

Freescale Semiconductor, Inc.

```

08640 *****
08650 * LDYR = load YREG
08660 * LDYR [#,<address>
08670 *
08680 * Examples:
08690 * 1. "LDYR #,START" puts the value of symbol 'START' into the
08700 * YREG
08710 * 2. "LDYR START" puts the contents of location 'START' and
08720 * 'START'+1 into the YREG.
08730 *
08740 * Register Usage:
08750 * CC = reflects MS half.
08760 * All other registers preserved.
08770 *
08780 LDYR MACR
08790 IFEQ NARG-1
08800 STA TEMPAS
08810 LDA (\0)
08820 STA YREG1$
08830 STA YREG2$
08840 LDA (\0)+1
08850 STA YREG1$+1
08860 STA YREG2$+1
08870 IFEQ YREG1$!.$FFF0
08880 LDA TEMPAS
08890 TST YREG1$
08900 ENDC
08910 IFNE YREG1$!.$FFF0
08920 STA TESTAS
08930 LDA TEMPAS
08940 TST TESTAS
08950 ENDC
08960 MEXIT
08970 ENDC
08980 IFEQ NARG-2
08990 IFC '\0', '#'
09000 IFEQ YREG1$!.$FFF0 ! YREG in low memory?
09010 IFEQ \1 ! #0 value?
09020 CLR YREG1$+1
09030 CLR YREG2$+1
09040 CLR YREG1$
09050 CLR YREG2$
09060 MEXIT
09070 ENDC
09080 IFNE \1 ! not #0 value?
09090 STA TEMPAS
09100 IFEQ (\1)!.$FFF
09110 CLR YREG1$+1
09120 CLR YREG2$+1
09130 ENDC
09140 IFNE (\1)!.$FFF
09150 LDA #(1)!.$FFF
09160 STA YREG1$+1
09170 STA YREG2$+1
09180 ENDC
09190 IFEQ (\1)!>8
09200 CLR YREG1$
09210 CLR YREG2$
09220 ENDC
09230 IFNE (\1)!>8
09240 LDA #(\1)!>8
09250 STA YREG1$
09260 STA YREG2$
09270 ENDC

```

Freescale Semiconductor, Inc.

```

09280          LDA      TEMPAS
09290          TST      YREG1$
09300          MEXIT
09310          ENDC
09320          ENDC
09330          IFNE     YREG1$!.$FF00      ! YREG in high memory?
09340          IFEQ     \1                  ! #0 value?
09350          STA      TEMPAS
09360          CLRA
09370          STA      YREG1$+1
09380          STA      YREG2$+1
09390          STA      YREG1$
09400          STA      YREG2$
09410          CLR     TESTAS
09420          LDA      TEMPAS
09430          TST      TESTAS
09440          MEXIT
09450          ENDC
09460          IFNE     \1                  ! not #0 value?
09470          STA      TEMPAS
09480          IFEQ     (\1)!.$FF
09490          CLRA
09500          ENDC
09510          IFNE     (\1)!.$FF
09520          LDA      #(\1)!.$FF
09530          ENDC
09540          STA      YREG1$+1
09550          STA      YREG2$+1
09560          IFEQ     (\1)!>8
09570          CLRA
09580          ENDC
09590          IFNE     (\1)!>8
09600          LDA      #(\1)!>8
09610          ENDC
09620          STA      YREG1$
09630          STA      YREG2$
09640          STA      TESTAS
09650          LDA      TEMPAS
09660          TST      TESTAS
09670          MEXIT
09680          ENDC
09690          ENDC
09700          ENDC
09710          ENDC
09720          FAIL    Macro syntax error detected!
09730          ENDM
09740
09750          *****
09760          *      STYR = store YREG
09770          *      STYR <address>
09780          *
09790          *      Examples:
09800          *      1. "STYR START"          stores the YREG into locations 'START' and
09810          *                               'START'+1.
09820          *
09830          *      Register Usage:
09840          *      CC = reflects MS half.
09850          *      All other registers preserved.
09860          *
09870          STYR    MACR
09880          STA      TEMPAS
09890          LDA      YREG1$
09900          STA      (\0)
09910          LDA      YREG1$+1
09920          STA      (\0)+1

```

Freescale Semiconductor, Inc.

```

09930   IFEQ   YREG1$!. $FF00
09940           LDA   TEMPAS
09950           TST   YREG1$
09960   ENDC
09970   IFNE   YREG1$!. $FF00
09980           STA   TESTAS
09990           LDA   TEMPAS
10000           TST   TESTAS
10010   ENDC
10020   ENDM
10030
10040 *****
10050 * INCYR = increment YREG
10060 *   INCYR [[#,<value>]
10070 *
10080 * Examples:
10090 *   1. "INCYR"           adds one (1) to the YREG.
10100 *   2. "INCYR #,START"  adds the value of symbol 'START' to the
10110 *                       YREG.
10120 *   3. "INCYR START"    adds the contents of location 'START' and
10130 *                       'START'+1 to the YREG.
10140 *   4. "INCYR ! comment" adds one (1) to the YREG (comment present!).
10150 *
10160 * Register Usage:
10170 *   CC = reflects value incremented (Z-bit only).
10180 *   All other registers preserved.
10190 *
10200 * Notes:
10210 *   1. Explicit comment character (!) MUST be used when comment field is
10220 *      present!
10230 *   2. Assumes YREG1$ = YREG2$.
10240 *   3. When parameters are present, this macro becomes "ADD to YREG".
10250 *
10260 INCYR MACR
10270   IFEQ   NARG
10280           IFEQ   YREG1$!. $FF00
10290           INC   YREG1$+1
10300           INC   YREG2$+1
10310           BNE   \.0
10320           INC   YREG1$
10330           INC   YREG2$
10340 \.0     EQU   *
10350           MEXIT
10360   ENDC
10370   IFNE   YREG1$!. $FF00
10380           STA   TEMPAS
10390           LDA   YREG1$+1
10400           ADD   #1
10410           STA   YREG1$+1
10420           STA   YREG2$+1
10430           LDA   YREG1$
10440           ADC   #0
10450           STA   YREG1$
10460           STA   YREG2$
10470           ORA   YREG1$+1
10480           STA   TESTAS
10490           LDA   TEMPAS
10500           TST   TESTAS
10510           MEXIT
10520   ENDC
10530 ENDC

```

Freescale Semiconductor, Inc.

```

10540 IFEQ      NARG-1
10550      STA      TEMPAS
10560      LDA      YREG1$+1
10570      ADD      (\0)+1
10580      STA      YREG1$+1
10590      STA      YREG2$+1
10600      LDA      YREG1$
10610      ADC      \0
10620      STA      YREG1$
10630      STA      YREG2$
10640      ORA      YREG1$+1
10650      STA      TESTAS
10660      LDA      TEMPAS
10670      TST      TESTAS
10680      MEXIT
10690 ENDC
10700 IFEQ      NARG-2
10710      IFC      '\0', '#'
10720      STA      TEMPAS
10730      LDA      YREG1$+1
10740      ADD      #(\1)!.$FF
10750      STA      YREG1$+1
10760      STA      YREG2$+1
10770      LDA      YREG1$
10780      ADC      #(\1)!>8
10790      STA      YREG1$
10800      STA      YREG2$
10810      ORA      XREG1$+1
10820      STA      TESTAS
10830      LDA      TEMPAS
10840      TST      TESTAS
10850      MEXIT
10860 ENDC
10870 ENDC
10880      FAIL      Macro syntax error detected!
10890      ENDM
10900
10910 *****
10920 * DECYR = decrement YREG
10930 *   DECYR    [[#,]<value>]
10940 *
10950 * Examples:
10960 *   1. "DECYR"                subtracts one (1) from the YREG.
10970 *   2. "DECYR #,START"       subtracts the value of symbol 'START' from
10980 *                             the YREG.
10990 *   3. "DECYR START"         subtracts the contents of location 'START'
11000 *                             and 'START'+1 from the YREG.
11010 *   4. "DECYR ! comment"     subtracts one from the YREG (comment present!).
11020 *
11030 * Register Usage:
11040 *   CC = reflects value decremented (Z-bit only).
11050 *   All other registers preserved.
11060 *
11070 * Notes:
11080 *   1. Explicit comment character (!) MUST be used when comment field is
11090 *       present!
11100 *   2. Assumes YREG1$ = YREG2$.
11110 *   3. When parameters are present, this macro becomes "SUBTRACT from YREG".
11120 *

```

Freescale Semiconductor, Inc.

```
11130  DECYR  MACR
11140  IFEQ   NARG
11150      STA   TEMPAS
11160      LDA   YREG1$+1
11170      SUB   #1
11180      STA   YREG1$+1
11190      STA   YREG2$+1
11200      LDA   YREG1$
11210      SBC   #0
11220      STA   YREG1$
11230      STA   YREG2$
11240      ORA   YREG1$+1
11250      STA   TESTAS
11260      LDA   TEMPAS
11270      TST   TESTAS
11280      MEXIT
11290  ENDC
11300  IFEQ   NARG-1
11310      STA   TEMPAS
11320      LDA   YREG1$+1
11330      SUB   (\0)+1
11340      STA   YREG1$+1
11350      STA   YREG2$+1
11360      LDA   YREG1$
11370      SBC   \0
11380      STA   YREG1$
11390      STA   YREG2$
11400      ORA   YREG1$+1
11410      STA   TESTAS
11420      LDA   TEMPAS
11430      TST   TESTAS
11440      MEXIT
11450  ENDC
11460  IFEQ   NARG-2
11470      IFC   '\0', '#'
11480      STA   TEMPAS
11490      LDA   YREG1$+1
11500      SUB   #(\1)!.$FF
11510      STA   YREG1$+1
11520      STA   YREG2$+1
11530      LDA   YREG1$
11540      SBC   #(\1)!>8
11550      STA   YREG1$
11560      STA   YREG2$
11570      ORA   YREG1$+1
11580      STA   TESTAS
11590      LDA   TEMPAS
11600      TST   TESTAS
11610      MEXIT
11620  ENDC
11630  ENDC
11640      FAIL   Macro syntax error detected!
11650  ENDM
11660
```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```

11670 *****
11680 * CPYR = compare YREG
11690 *   CPYR  [#,<address>
11700 *
11710 * Examples:
11720 *   1. "CPYR #,BLOCKSZ"      compares the value of symbol 'BLOCKSZ'
11730 *                           with the YREG.
11740 *   2. "CPYR START"         compares the contents of location
11750 *                           'START' and 'START'+1 with the YREG.
11760 *
11770 * Register Usage:
11780 *   CC = reflects YREG comparison (Z-bit only).
11790 *   All other registers preserved.
11800 *
11810 CPYR  MACR
11820   IFEQ  NARG-1
11830     STA  TEMPAS
11840     BSET 0,TESTAS  Preset for .NE. condition!
11850     LDA  YREG1$+1
11860     CMP  (\0)+1
11870     BNE  \.0      Branch if LS half is .NE.
11880     LDA  YREG1$
11890     CMP  (\0)
11900     BNE  \.0      Branch if MS half is .NE.
11910     CLR  TESTAS  Set for .EQ. condition!
11920 \.0     LDA  TEMPAS
11930     TST  TESTAS  Set proper Z-bit (.EQ. or .NE.)!
11940     MEXIT
11950   ENDC
11960   IFEQ  NARG-2
11970     IFC  '\0', '#'
11980     IFEQ \1
11990     IFEQ YREG1$!. $FF00
12000     TST  YREG1$+1
12010     BNE  \.0      Branch if LS half is .NE.
12020     TST  YREG1$
12030 \.0     EQU  *
12040     MEXIT
12050   ENDC
12060   IFNE  YREG1$!. $FF00
12070     STA  TEMPAS
12080     BSET 0,TESTAS  Preset for .NE. condition!
12090     LDA  YREG1$+1
12100     BNE  \.0      Branch if MS half is .NE.
12110     LDA  YREG1$
12120     BNE  \.0      Branch if MS half is .NE.
12130     CLR  TESTAS  Set for .EQ. condition!
12140 \.0     LDA  TEMPAS
12150     TST  TESTAS  Set proper Z-bit (.EQ. or .NE.)!
12160     MEXIT
12170   ENDC
12180   ENDC
12190     STA  TEMPAS
12200     BSET 0,TESTAS  Preset for .NE. condition!
12210     LDA  YREG1$+1
12220   IFNE  (\1)!. $FF
12230     CMP  #(\1)!. $FF
12240   ENDC
12250     BNE  \.0      Branch if LS half is .NE.
12260     LDA  YREG1$
12270   IFNE  (\1)!>8
12280     CMP  #(\1)!>8
12290   ENDC

```


Freescale Semiconductor, Inc.

```

12300         BNE     \.0           Branch if MS half is .NE.
12310         CLR     TESTA$       Set for .EQ. condition!
12320     \.0   LDA     TEMPAS$
12330         TST     TESTA$       Set proper Z-bit (.EQ. or .NE.)!
12340         MEXIT
12350         ENDC
12360     ENDC
12370         FAIL     Macro syntax error detected!
12380         ENDM
12390
12400 *****
12410 * DEC.B = decrement byte
12420 * DEC.B [[#,<value>,<address>]
12430 *
12440 * where:
12450 * <value> = value to decrement the contents of the <address>
12460 *          location by; immediate ("#," present) or absolute
12470 *          addressing ("#," not present). If only <address> is
12480 *          specified, a default immediate value of one is used.
12490 *
12500 * Examples:
12510 * 1. "DEC.B START"             subtracts one from the contents of
12520 *                               location 'START'.
12530 * 2. "DEC.B #,5,START"        subtracts five from the contents of
12540 *                               location 'START'.
12550 * 3. "DEC.B CNT,START"        subtracts the contents of location 'CNT'
12560 *                               from the contents of location 'START'.
12570 *
12580 * Register Usage:
12590 *   CC = reflects value decremented (N and Z-bits).
12600 *   All other registers preserved.
12610 *
12620 * Notes:
12630 * 1.<address> may be direct or extended!
12640 * 2.This macro essentially performs a "SUB n" function.
12650 *
12660     DEC.B     MACR
12670     IFEQ     NARG-1
12680     IFEQ     (\0)!. $FF00
12690     DEC     \0
12700     MEXIT
12710     ENDC
12720     STA     TEMPAS$
12730     LDA     \0
12740     SUB     #1
12750     STA     \0
12760     STA     TESTAS$
12770     LDA     TEMPAS$
12780     TST     TESTAS$
12790     MEXIT
12800     ENDC
12810     IFEQ     NARG-2
12820     STA     TEMPAS$
12830     LDA     \1
12840     SUB     \0
12850     STA     \1
12860     IFEQ     (\1)!. $FF00
12870     LDA     TEMPAS$
12880     TST     \1
12890     MEXIT
12900     ENDC

```

Freescale Semiconductor, Inc.

```

12910     IFNE     (\1)!. $FF00
12920     STA     TESTA$
12930     LDA     TEMPAS
12940     TST     TESTAS
12950     MEXIT
12960     ENDC
12970     ENDC
12980     IFEQ     NARG-3
12990     IFC     '\0', '#'
13000     STA     TEMPAS
13010     LDA     \2
13020     SUB     #\1
13030     STA     \2
13040     IFEQ     (\2)!. $FF00
13050     LDA     TEMPAS
13060     TST     \2
13070     MEXIT
13080     ENDC
13090     IFNE     (\2)!. $FF00
13100     STA     TESTA$
13110     LDA     TEMPAS
13120     TST     TESTAS
13130     MEXIT
13140     ENDC
13150     ENDC
13160     ENDC
13170     FAIL     Macro syntax error detected!
13180     ENDM
13190
13200     *****
13210     * DEC.W = decrement word
13220     *     DEC.W [[#,]<value>,<address>
13230     *
13240     * where:
13250     *     <value> = value to decrement the contents of the <address> and
13260     *                 <address>+1 locations by; immediate ("#", present) or
13270     *                 absolute addressing ("#", not present). If only
13280     *                 <address> is specified, a default immediate value of
13290     *                 one is used.
13300     *
13310     * Examples:
13320     *     1. "DEC.W START"           subtracts one from the contents of loca-
13330     *                                     tions 'START' and 'START'+1.
13340     *     2. "DEC.W CNT, START"     subtracts the contents of locations 'CNT'
13350     *                                     and 'CNT'+1 from the contents of locations
13360     *                                     'START' and 'START'+1.
13370     *     3. "DEC.W #, 5, START"    subtracts five from the contents of loca-
13380     *                                     tions 'START' and 'START'+1.
13390     *
13400     * Register Usage:
13410     *     CC = reflects value incremented (Z-bit only).
13420     *     All other registers preserved.
13430     *
13440     * Notes:
13450     *     1. <address> may be direct or extended!
13460     *     2. This macro essentially performs a "SUB n" function.
13470     *

```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```
13480 DEC.W MACR
13490     IFEQ  NARG-1
13500         STA  TEMPAS
13510         LDA  (\0)+1
13520         SUB  #1
13530         STA  (\0)+1
13540         LDA  \0
13550         SBC  #0
13560         STA  \0
13570         ORA  (\0)+1
13580         STA  TESTAS
13590         LDA  TEMPAS
13600         TST  TESTAS
13610     MEXIT
13620     ENDC
13630     IFEQ  NARG-2
13640         STA  TEMPAS
13650         LDA  (\1)+1
13660         SUB  (\0)+1
13670         STA  (\1)+1
13680         LDA  \1
13690         SBC  \0
13700         STA  \1
13710         ORA  (\1)+1
13720         STA  TESTAS
13730         LDA  TEMPAS
13740         TST  TESTAS
13750     MEXIT
13760     ENDC
13770     IFEQ  NARG-3
13780         IFC  '\0', '#'
13790         STA  TEMPAS
13800         LDA  (\2)+1
13810         SUB  #(\1)!.$FF
13820         STA  (\2)+1
13830         LDA  \2
13840         SBC  #(\1)!>8
13850         STA  \2
13860         ORA  (\2)+1
13870         STA  TESTAS
13880         LDA  TEMPAS
13890         TST  TESTAS
13900     MEXIT
13910     ENDC
13920     ENDC
13930     FAIL  Macro syntax error detected!
13940     ENDM
13950
```

Freescale Semiconductor, Inc.

```
13960 *****
13970 * INC.B = increment byte
13980 *   INC.B [[#,<value>,<address>]
13990 *
14000 * where:
14010 *   <value> = value to decrement the contents of the <address>
14020 *           location by; immediate ("#", present) or absolute
14030 *           addressing ("#", not present). If only <address> is
14040 *           specified, a default immediate value of one is used.
14050 *
14060 * Examples:
14070 *   1. "INC.B  START"           adds one to the contents of location
14080 *                               'START'.
14090 *   2. "INC.B  #,5,START"      adds five to the contents of location
14100 *                               'START'.
14110 *   3. "INC.B  CNT,START"      adds the contents of location 'CNT' to
14120 *                               the contents of location 'START'.
14130 *
14140 * Register Usage:
14150 *   CC= reflects value incremented (N and Z-bits).
14160 *   All other registers preserved.
14170 *
14180 * Notes:
14190 *   1. <address> may be direct or extended!
14200 *   2. This macro essentially performs an "ADD n" function.
14210 *
14220 INC.B MACR
14230     IFEQ  NARG-1
14240     IFEQ  (\0)!. $FF00
14250     INC   \0
14260     MEXIT
14270     ENDC
14280     STA   TEMPAS
14290     LDA   \0
14300     ADD   #1
14310     STA   \0
14320     STA   TESTAS
14330     LDA   TEMPAS
14340     TST   TESTAS
14350     MEXIT
14360     ENDC
14370     IFEQ  NARG-2
14380     STA   TEMPAS
14390     LDA   \1
14400     ADD   \0
14410     STA   \1
14420     IFEQ  (\1)!. $FF00
14430     LDA   TEMPAS
14440     TST   \1
14450     MEXIT
14460     ENDC
14470     IFNE  (\1)!. $FF00
14480     STA   TESTAS
14490     LDA   TEMPAS
14500     TST   TESTAS
14510     MEXIT
14520     ENDC
14530 ENDC
```

Freescale Semiconductor, Inc.

```

14540 IFEQ      NARG-3
14550 IFC      '\0', '#'
14560 STA      TEMPAS$
14570 LDA      \2
14580 ADD      #1
14590 STA      \2
14600 IFEQ      (\2)!.$FF00
14610 LDA      TEMPAS$
14620 TST      \2
14630 MEXIT
14640 ENDC
14650 IFNE      (\2)!.$FF00
14660 STA      TESTAS$
14670 LDA      TEMPAS$
14680 TST      TESTAS$
14690 MEXIT
14700 ENDC
14710 ENDC
14720 ENDC
14730 FAIL      Macro syntax error detected!
14740 ENDM
14750
14760 *****
14770 * INC.W = increment word
14780 * INC.W    [[#,]<value>,<address>
14790 *
14800 * where:
14810 * <value> = value to increment the contents of the <address> and
14820 *          <address>+1 locations by; immediate ("#," present)
14830 *          or absolute addressing ("#," not present). If only
14840 *          <address> is specified, a default immediate value of
14850 *          one is used.
14860 *
14870 * Examples:
14880 * 1. "INC.W START"          adds one to the contents of locations
14890 *                          'START' and 'START'+1.
14900 * 2. "INC.W CNT,START"     adds the value of 'CNT' to the contents
14910 *                          of locations 'START' and 'START'+1.
14920 * 3. "INC.W #,5,START"     adds five to the contents of locations
14930 *                          'START' and 'START'+1.
14940 *
14950 * Register Usage:
14960 * CC = reflects value incremented (Z-bit only).
14970 * All other registers preserved.
14980 *
14990 * Notes:
15000 * 1.<address> may be direct or extended!
15010 * 2.This macro essentially performs an "ADD n" function.
15020 *
15030 INC.W     MACR
15040 IFEQ      NARG-1
15050 IFEQ      (\0)!.$FF00
15060 INC      (\0)+1
15070 BNE      \.0
15080 INC      \0
15090 \.0     EQU      *
15100 MEXIT
15110 ENDC

```

```

15120     IFNE      (\0)!. $FF00
15130             STA      TEMPAS
15140             LDA      (\0)+1
15150             ADD      #1
15160             STA      (\0)+1
15170             LDA      \0
15180             ADC      #0
15190             STA      \0
15200             ORA      (\0)+1
15210             STA      TESTAS
15220             LDA      TEMPAS
15230             TST      TESTAS
15240             MEXIT
15250     ENDC
15260     ENDC
15270     IFEQ      NARG-2
15280             STA      TEMPAS
15290             LDA      (\1)+1
15300             ADD      (\0)+1
15310             STA      (\1)+1
15320             LDA      \1
15330             ADC      \0
15340             STA      \1
15350             ORA      (\1)+1
15360             STA      TESTAS
15370             LDA      TEMPAS
15380             TST      TESTAS
15390             MEXIT
15400     ENDC
15410     IFEQ      NARG-3
15420             IFC      '\0', '#'
15430             STA      TEMPAS
15440             LDA      (\2)+1
15450             ADD      #(\1)!. $FF
15460             STA      (\2)+1
15470             LDA      \2
15480             ADC      #(\1)!>8
15490             STA      \2
15500             ORA      (\2)+1
15510             STA      TESTAS
15520             LDA      TEMPAS
15530             TST      TESTAS
15540             MEXIT
15550     ENDC
15560     ENDC
15570             FAIL      Macro syntax error detected!
15580             ENDM
15590

```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```
15600 *****
15610 * MOV.B = move byte
15620 *   MOV.B   [#,<byte>,<address>
15630 *
15640 * where:
15650 *   <byte>  = byte value to move to the <address> location, using
15660 *             immediate ("#," present) or absolute addressing ("#,"
15670 *             not present).
15680 *
15690 * Examples:
15700 *   1. "MOV.B CNT,TMP"   puts the contents of location 'CNT'
15710 *                       into location 'TMP'.
15720 *   2. "MOV.B #,5,START" puts 5 into location 'START'.
15730 *
15740 * Register Usage:
15750 *   CC = reflects value moved.
15760 *   All other registers preserved.
15770 *
15780 MOV.B MACR
15790   IFEQ   NARG-2
15800       STA   TEMPAS
15810       LDA   \0
15820       STA   \1
15830   IFEQ   (\1)!.$FF00
15840       LDA   TEMPAS
15850       TST   \1
15860       MEXIT
15870   ENDC
15880   IFNE   (\1)!.$FF00
15890       IFEQ   (\0)!.$FF00
15900       LDA   TEMPAS
15910       TST   \0
15920       MEXIT
15930   ENDC
15940   IFNE   (\0)!.$FF00
15950       STA   TESTAS
15960       LDA   TEMPAS
15970       TST   TESTAS
15980       MEXIT
15990   ENDC
16000   ENDC
16010   ENDC
16020   IFEQ   NARG-3
16030   IFC   '\0','#'
16040       IFEQ   (\2)!.$FF00
16050       IFEQ   \1
16060       CLR   \2
16070       MEXIT
16080   ENDC
16090   ENDC
16100       STA   TEMPAS
16110       IFEQ   \1
16120       CLRA
16130   ENDC
16140       IFNE   \1
16150       LDA   #1
16160   ENDC
16170       STA   \2
16180       IFEQ   (\2)!.$FF00
16190       LDA   TEMPAS
16200       TST   \2
16210       MEXIT
16220   ENDC
```

Freescale Semiconductor, Inc.

```

16230      IFNE  (\2)!.$FF00
16240      STA   TESTA$
16250      LDA   TEMPAS$
16260      TST   TESTAS$
16270      MEXIT
16280      ENDC
16290      ENDC
16300      ENDC
16310      FAIL  Macro syntax error detected!
16320      ENDM
16330
16340      *****
16350      * MOV.W = move word
16360      *      MOV.W [#,<word>,<address>
16370      *
16380      * where:
16390      *      <word> = word (16-bit) value to move to the <address> and
16400      *                  <address>+1 locations, using immediate ("#," present)
16410      *                  or absolute addressing ("#," not present).
16420      *
16430      * Examples:
16440      *      1. "MOV.W #,5,START"      puts $0005 into location 'START' and
16450      *                  'START'+1.
16460      *      2. "MOV.W #,CNT,TMP"     puts the value of symbol 'CNT' into
16470      *                  locations 'TMP' and 'TMP'+1.
16480      *      3. "MOV.W CNT,TMP"       copies the contents of location 'CNT'
16490      *                  and 'CNT'+1 into locations 'TMP' and
16500      *                  'TMP'+1.
16510      *
16520      * Register Usage:
16530      *      CC = reflects MS half of value moved.
16540      *      All other registers preserved.
16550      *
16560      MOV.W  MACR
16570      IFEQ   NARG-2
16580      STA   TEMPAS$
16590      LDA   (\0)+1
16600      STA   (\1)+1
16610      LDA   \0
16620      STA   \1
16630      IFEQ   (\1)!.$FF00
16640      LDA   TEMPAS$
16650      TST   \1
16660      MEXIT
16670      ENDC
16680      IFNE   (\1)!.$FF00
16690      IFEQ   (\0)!.$FF00
16700      LDA   TEMPAS$
16710      TST   \0
16720      MEXIT
16730      ENDC
16740      IFNE   (\0)!.$FF00
16750      STA   TESTAS$
16760      LDA   TEMPAS$
16770      TST   TESTAS$
16780      MEXIT
16790      ENDC
16800      ENDC
16810      ENDC

```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```
16820 IFEQ NARG-3
16830 IFC '\0', '#'
16840 IFEQ ((\2)+1)!.$FF00
16850 IFEQ \1
16860 CLR \2
16870 CLR \2+1
16880 MEXIT
16890 ENDC
16900 ENDC
16910 STA TEMPAS
16920 IFEQ (\1)!.$00FF
16930 CLRA
16940 ENDC
16950 IFNE (\1)!.$00FF
16960 LDA #(\1)!.$00FF
16970 ENDC
16980 STA (\2)+1
16990 IFEQ (\1)!>8
17000 IFNE (\1)!.$00FF
17010 CLRA
17020 ENDC
17030 ENDC
17040 IFNE (\1)!>8
17050 LDA #(\1)!>8
17060 ENDC
17070 STA \2
17080 IFEQ (\2)!.$FF00
17090 LDA TEMPAS
17100 TST \2
17110 MEXIT
17120 ENDC
17130 IFNE (\2)!.$FF00
17140 STA TESTAS
17150 LDA TEMPAS
17160 TST TESTAS
17170 MEXIT
17180 ENDC
17190 ENDC
17200 ENDC
17210 FAIL Macro syntax error detected!
17220 ENDM
17230
```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```

17240 *****
17250 * MOVE = move block of memory
17260 *   MOVE  [#], <source>, [#], <destination>, [#], <length>
17270 *
17280 * where:
17290 *   <source>      is the address of the source memory block.
17300 *   <destination> is the address of the destination block.
17310 *   <length>      is the length of the block to move, in bytes.
17320 *                 Maximum of 65,536 bytes can be moved.
17330 *   #             is optional character to denote immediate
17340 *                 addressing for the next parameter
17350 * Examples:
17360 *   1. "MOVE #, ROM, #, RAM, #, CNT moves the block of memory starting at
17370 *       location 'ROM' for 'CNT' bytes, to
17380 *       location 'RAM'.
17390 *   2. "MOVE #, ABC, #, XYZ, , CNT moves the block of memory starting at
17400 *       location 'ABC' for the number of bytes
17410 *       in locations 'CNT' and 'CNT'+1, to
17420 *       location 'XYZ'.
17430 *
17440 * Register Usage:
17450 *   CC = unknown.
17460 *   All other registers preserved.
17470 *
17480 * Subr. used:
17490 *   LDAXREG, STAXREG
17500 *
17510 * Macros used:
17520 *   None, because this macro was written to be as efficient as
17530 *   possible.
17540 *
17550 * Notes:
17560 *   1. If all immediate addressing operands (#) and the move count is
17570 *       <= 256, then a special 'short form' is generated which DOES NOT
17580 *       contain any subroutine calls!
17590 *   2. Depending on the exact parameters passed, not all registers,
17600 *       subroutines and/or macros may be used.
17610 *   3. This macro takes advantage of the fact that there are in fact
17620 *       two XREGs, one for LOAD (XREG1$) and one for STORE (XREG2$).
17630 *   4. The INCXR macro cannot be used here, because it assumes that
17640 *       XREG1$ = XREG2$.
17650 * -----
17660 MOVE  MACR
17670       IFNE  NARG-6
17680       FAIL  ** 'move' macro requires six arguments!
17690       ENDC
17700       IFC  '\4', '#'           ! If all immediate operands (#) and move
17710       IFC  '\2', '#'           ! count <=256, use short form!
17720       IFC  '\0', '#'
17730       IFLE  5-256             ! No subr. calls!
17740       STA  TEMPAS$
17750       STX  TEMPX$
17760       LDX  #(\5)
17770 \.0   LDA  (\1)-1,x
17780       STA  (\3)-1,x
17790       DEX
17800       BNE  \.0
17810       LDA  TEMPAS$
17820       LDX  TEMPX$
17830       MEXIT
17840       ENDC
17850       ENDC
17860       ENDC
17870       ENDC
17880

```

Freescale Semiconductor, Inc.

```

17890      STA      TEMPX$
17900      STX      TEMPX$
17910      LDA      XREG1$
17920      STA      TEMPXR$
17930      LDA      XREG1$+1
17940      STA      TEMPXR$+1
17950      IFC      '\0', '#'          ! immediate type 'from' address?
17960      LDA      #(\1)!.$FF
17970      STA      XREG1$+1          ! Set XREG1$ = 'from' address
17980      LDA      #(\1)!>8
17990      STA      XREG1$
18000      ENDC
18010      IFNC      '\0', '#'          ! not immediate type 'from' address?
18020      LDA      (\1)+1
18030      STA      XREG1$+1          ! Set XREG1$ = 'from' address
18040      LDA      (\1)
18050      STA      XREG1$
18060      ENDC
18070      IFC      '\2', '#'          ! immediate type 'to' address?
18080      LDA      #(\3)!.$FF
18090      STA      XREG2$+1          ! Set XREG2$ = 'to' address
18100      LDA      #(\3)!>8
18110      STA      XREG2$
18120      ENDC
18130      IFNC      '2', '#'          ! not immediate type 'to' address?
18140      LDA      (\3)+1
18150      STA      XREG2$+1          ! Set XREG2$ = 'to' address
18160      LDA      (\3)
18170      STA      XREG2$
18180      ENDC
18190
18200      IFC      '\4', '#'          ! immediate type length?
18210      IFLE      '\5-256          ! yes: 8-bit size= use X reg.
18220      LDX      #(\5)
18230      \.0     JSR      LDAXREG
18240      JSR      STAXREG
18250      IFEQ      XREG1$!.$FF00
18260      INC      XREG1$+1
18270      BNE      \.1
18280      INC      XREG1$
18290      \.1     INC      XREG2$+1
18300      BNE      \.2
18310      INC      XREG2$
18320      \.2     EQU      *
18330      ENDC
18340      IFNE      XREG1$!.$FF00
18350      LDA      XREG1$+1
18360      ADD      #1
18370      STA      XREG1$+1
18380      LDA      XREG1$
18390      ADC      #0
18400      STA      XREG1$
18410      LDA      XREG2$+1
18420      ADD      #1
18430      STA      XREG2$+1
18440      LDA      XREG2$
18450      ADC      #0
18460      STA      XREG2$
18470      ENDC
18480      DEX
18490      BNE      \.0
18500      LDA      TEMPXR$
18510      STA      XREG1$
18520      STA      XREG2$
18530      LDA      TEMPXR$+1

```

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

```

18540      STA  XREG1$+1
18550      STA  XREG2$+1
18560      LDA  TEMPAS$
18570      LDX  TEMPX$
18580      MEXIT
18590      ENDC
18600      *
18610      IFGT  \5-256      !   no: 16-bit size= use 'length'
18620      LDA  #(\5)!.$00FF
18630      STA  LENGTH$+1
18640      LDA  #(\5)!>8
18650      STA  LENGTH$
18660  \.0   JSR  LDAXREG
18670      JSR  STAXREG
18680      IFEQ  XREG1$!.$FF00
18690      INC  XREG1$+1
18700      BNE  \.1
18710      INC  XREG1$
18720  \.1   INC  XREG2$+1
18730      BNE  \.2
18740      INC  XREG2$
18750  \.2   EQU  *
18760      ENDC
18770      IFNE  XREG1$!.$FF00
18780      LDA  XREG1$+1
18790      ADD  #1
18800      STA  XREG1$+1
18810      LDA  XREG1$
18820      ADC  #0
18830      STA  XREG1$
18840      LDA  XREG2$+1
18850      ADD  #1
18860      STA  XREG2$+1
18870      LDA  XREG2$
18880      ADC  #0
18890      STA  XREG2$
18900      ENDC
18910      LDA  LENGTH$+1
18920      SUB  #1
18930      STA  LENGTH$+1
18940      LDA  LENGTH$
18950      SBC  #0
18960      STA  LENGTH$
18970      ORA  LENGTH$+1
18980      BNE  \.0
18990      LDA  TEMPXR$
19000      STA  XREG1$
19010      STA  XREG2$
19020      LDA  TEMPXR$+1
19030      STA  XREG1$+1
19040      STA  XREG2$+1
19050      LDA  TEMPAS$
19060      LDX  TEMPX$
19070      MEXIT
19080      ENDC
19090      ENDC
19100

```

Freescale Semiconductor, Inc.

```

19110 IFNC      '\4', '#'      ! nonimmediate type length
19120         LDA      (\5)!.$00FF
19130         STA      LENGTH$+1
19140         LDA      (\5)!>8
19150         STA      LENGTH$
19160 \.0     JSR      LDAXREG
19170         JSR      STAXREG
19180         IFEQ     XREG1$!.$FFF0
19190         INC      XREG1$+1
19200         BNE      \.1
19210         INC      XREG1$
19220 \.1     INC      XREG2$+1
19230         BNE      \.2
19240         INC      XREG2$
19250 \.2     EQU      *
19260         ENDC
19270         IFNE     XREG1$!.$FFF0
19280         LDA      XREG1$+1
19290         ADD      #1
19300         STA      XREG1$+1
19310         LDA      XREG1$
19320         ADC      #0
19330         STA      XREG1$
19340         LDA      XREG2$+1
19350         ADD      #1
19360         STA      XREG2$+1
19370         LDA      XREG2$
19380         ADC      #0
19390         STA      XREG2$
19400         ENDC
19410         LDA      LENGTH$+1
19420         SUB      #1
19430         STA      LENGTH$+1
19440         LDA      LENGTH$
19450         SBC      #0
19460         STA      LENGTH$
19470         ORA      LENGTH$+1
19480         BNE      \.0
19490         LDA      TEMPXR$
19500         STA      XREG1$
19510         STA      XREG2$
19520         LDA      TEMPXR$+1
19530         STA      XREG1$+1
19540         STA      XREG2$+1
19550         LDA      TEMPAS$
19560         LDX      TEMPX$
19570         MEXIT
19580         ENDC
19590         FAIL     Macro syntax error detected!
19600         ENDM
19610
19620

```

Freescale Semiconductor, Inc.

```
19630          OPT    L
19640 RAMSBR$ EQU    *          Start of RAM based subroutines!
19650 *****
19660 ** The following RAM subroutines MUST BE INITIALIZED from ROM upon **
19670 ** startup (from 'RAMSBR$' for 'RAMSZ$' number of bytes). If changes **
19680 ** are to be made to the RAM subroutines, make them here. Then copy **
19690 ** the source below to the ROM area and insert a '.' in front of all **
19700 ** the labels (leading '.' will be used to denote ROM). This has **
19710 ** already been done for you in the RAMSBR.INI file. Just include **
19720 ** this file into your ROM data area and add the following line in **
19730 ** your RESET routine to initialize the RAM subroutines from the ROM. **
19740 **          MOVE #, .RAMSBR, #, RAMSBR, #, **
19750 ** It is more efficient if the RAM subroutines are placed in DIRECT **
19760 ** addressing memory, i.e., $0000-$00FF, but it is not required. **
19770 *****
19780
19790 *-- start of RAM subroutines -----*
19800 *****
19810 * LDAXREG = load A via XREG subr.
19820 *
19830 * Register Usage:
19840 *   CC = reflects value loaded.
19850 *   All other registers preserved.
19860 *
19870 * NOTE:
19880 *   1. Instruction modified code here must be located in RAM!
19890 *
19900 LDAXREG EQU    *
19910     LDA    0-0+$FFFF
19920 XREG1$ EQU    *-2          Pseudo XREG #1
19930     RTS
19940
19950 *****
19960 * STAXREG = store A via XREG subr.
19970 *
19980 * Register Usage:
19990 *   CC = reflects value stored.
20000 *   All other registers preserved.
20010 *
20020 * NOTE:
20030 *   1. Instruction modified code here must be located in RAM!
20040 *
20050 STAXREG EQU    *
20060     STA    0-0+$FFFF
20070 XREG2$ EQU    *-2          Pseudo XREG #2
20080     RTS
20090
20100 *****
20110 * LDAYREG = load A via YREG subr.
20120 *
20130 * Register Usage:
20140 *   CC = reflects value loaded.
20150 *   All other registers preserved.
20160 *
20170 * NOTE:
20180 *   1. Instruction modified code here must be located in RAM!
20190 *
20200 LDAYREG EQU    *
20210     LDA    0-0+$FFFF
20220 YREG1$ EQU    *-2          Pseudo YREG #1
20230     RTS
20240
```

Freescale Semiconductor, Inc.

```
20250 *****
20260 * STAYREG = store A via YREG subr.
20270 *
20280 * Register Usage:
20290 *   CC = reflects value stored.
20300 *   All other registers preserved.
20310 *
20320 * NOTE:
20330 *   1. Instruction modified code here must be located in RAM!
20340 *
20350 STAYREG EQU *
20360 STA 0-0+$FFFF
20370 YREG2$ EQU *-2 Pseudo YREG #2
20380 RTS
20390 *-- end of RAM subroutines ----- *
20400
20410 RAMSZ$ EQU *-RAMBR$ Size of ram subroutines (in bytes).
20420
20430 ORG LO$MEM
20440 * NOTE: TEMPAS$ and TESTAS$ must always be in low memory $0000-00FF.
20450 TEMPAS$ RMB 1 Temporary storage for A accumulator.
20460 TEMPX$ RMB 1 Temporary storage for X register.
20470 TEMPXR$ RMB 2 Temporary storage for XREG register.
20480 TESTAS$ RMB 1 Temporary operand storage for setting CC bits.
20490 LENGTH$ RMB 2 Temporary operand length.
20500
20510 *****
```

Freescale Semiconductor, Inc.

Listing 2 — RAMSBR.INI File

```
00010
00020 *****
00030 * ramsbr.ini 1.0
00040 * -----
00050 * Module Name:      RAMSBR - RAM Subroutine Initialization
00060 * -----
00070 *
00080 * Description:
00090 * This file contains the initialization code for the RAM subroutine
00100 * area needed to support the MACROS05.MAC file. It MUST be placed in
00110 * the ROM data area and then copied to RAM for proper operation.
00120 * Consult the MACROS05.MAC file for more details.
00130 *
00140 *****
00150 *
00160 * Notes:
00170 * 1. Motorola reserves the right to make changes to this file.
00180 * Although this file has been carefully reviewed and is
00190 * believed to be reliable, Motorola does not assume any
00200 * liability arising out of its use. This code may be freely
00210 * used and/or modified at no cost or obligation by the user.
00220 * 2. The latest version of this file is maintained on the Motorola
00230 * FREEMWARE Bulletin Board, 512/891-FREE (512/891-3733). It operates
00240 * continuously (except for maintenance) at 1200-2400 baud, 8 bits,
00250 * no parity. Sample test files for PASM05 are also included.
00260 * Download the archive file, MACROS05.ARC, to get all the files.
00270 *
00280 *****
00290 * REVISION HISTORY (add new changes to top):
00300 *
00310 * 05/16/90 P.S. Gilmour
00320 * 1. Original entry generated from MACROS05.MAC version 1.0.
00330 *****
00340
00350 .RAMSBR$ EQU * Start of RAM based subroutines!
00360 *****
00370 ** The following RAM subroutines MUST BE INITIALIZED from ROM upon **
00380 ** startup (from 'RAMSBR$' for 'RAMSZ$' number of bytes). If changes **
00390 ** are to be made to the RAM subroutines, make them in the MACROS05.MAC **
00400 ** file and then copy the source here (ROM area) and insert a '.' in **
00410 ** front of all the labels (leading '.' will be used to denote ROM). **
00420 *****
00430
00440 *-- start of RAM subroutines ----- *
00450 *****
00460 * LDAXREG = load A via XREG subr.
00470 *
00480 * Register Usage:
00490 * CC = reflects value loaded.
00500 * All other registers preserved.
00510 *
00520 * NOTE:
00530 * 1. Instruction modified code here must be located in RAM!
00540 *
00550 .LDAXREG EQU *
00560 LDA 0-0+$FFFF
00570 .XREG1$ EQU *-2 Pseudo XREG #1
00580 RTS
00590
```


Freescale Semiconductor, Inc.

```
00600 *****
00610 * STASX = store A via XREG subr.
00620 *
00630 * Register Usage:
00640 *   CC = reflects value stored.
00650 *   All other registers preserved.
00660 *
00670 * NOTE:
00680 *   1. Instruction modified code here must be located in RAM!
00690 *
00700 .STASX EQU *
00710     STA 0-0+$FFFF
00720 .XREG2$ EQU *-2      Pseudo XREG #2
00730     RTS
00740
00750 *****
00760 * LDAYREG = load A via YREG subr.
00770 *
00780 * Register Usage:
00790 *   CC = reflects value loaded.
00800 *   All other registers preserved.
00810 *
00820 * NOTE:
00830 *   1. Instruction modified code here must be located in RAM!
00840 *
00850 .LDAYREG EQU *
00860     LDA 0-0+$FFFF
00870 .YREG1$ EQU *-2      Pseudo YREG #1
00880     RTS
00890
00900 *****
00910 * STASY = store A via YREG subr.
00920 *
00930 * Register Usage:
00940 *   CC = reflects value stored.
00950 *   All other registers preserved.
00960 *
00970 * NOTE:
00980 *   1. Instruction modified code here must be located in RAM!
00990 *
01000 .STASY EQU *
01010     STA 0-0+$FFFF
01020 .YREG2$ EQU *-2      Pseudo YREG #2
01030     RTS
01040 *-- end of RAM subroutines -----*
01050
01060 .RAMSZ$ EQU *-.RAMSBR$   Size of ram subroutines (in bytes).
01070     FNE RAMSZ$-.RAMSZ$
01080     FAIL Size mismatch between RAM/ROM subroutine areas!
01090     ENDC
```

Freescale Semiconductor, Inc.

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



AN1055/D



**For More Information On This Product,
Go to: www.freescale.com**