MOTOROLA
Semiconductor Products Sector
Application Note

AN432

128-Kbyte Addressing with the M68HC11

**By Ross Mitchell
MCU Applications Engineering
Motorola Ltd.
East Kilbride, Scotland**

Overview

The maximum direct addressing capability of the M68HC11 device is 64 Kbytes, but this can be insufficient for some applications.

This application note describes two methods of memory paging that allow the MCU to fully address a single 1 megabit EPROM (128 Kbytes) by manipulation of the address lines.

The two methods illustrate the concept of paging and the inherent compromises. The technique may be expanded to allow addressing of several EPROM, RAM, or EEPROM memories, or several smaller memories by using both address lines and chip enables.

Digital DNA
from Motorola

Paging Scheme

The M68HC11 8-bit MCU is capable of addressing up to 64 Kbytes of contiguous address space. Addressing greater than 64 Kbytes requires that a section of the memory be replaced with another block of memory at the same address range. This technique of swapping memory is known as paging and is simply a method of overlaying blocks of data over each other such that only one of the blocks or pages is visible to the CPU at a given time.

In a system requiring more than 64 Kbytes of user code and tables, it is possible to use the port lines to extend the memory addressing range of the M68HC11 device. This has certain restrictions, but these can be minimized by careful consideration of the user code implementation.

There are two basic configurations:

- Method A uses only software plus a single port line to control the high address bit A16.

- Method B is a combination of a small amount of hardware and software controlling the top three address bits — A14, A15, and A16.

In the examples that follow, the MC68HC11G5 device is used to demonstrate the paging techniques, since this device has a non-multiplexed data and address bus. Any M68HC11 device may be used in a similar way.

Method A has the advantage of no additional hardware and very few limitations in the software. The user code main loop can be up to 64 Kbytes long and remain in the same page, but this is at the expense of longer interrupt latency. The vector table and a small amount of code must be present in both pages of memory to allow correct swapping of the pages.

Method B has the advantage of not affecting the interrupt latency and has just one copy of the vector table. The maximum length of the user code main loop in this example is 48 Kbytes with a further five paged areas of 16 Kbytes for subroutines and tables.

AN432

Method A — Software Technique

Address A16 of the EPROM is directly controlled by port D(5) of the M68HC11 as shown in **Figure 1**. This port is automatically configured to be in the input state following reset. It is vital that the state of the port line controlling address A16 is known following reset and so there is a 10-kΩ pullup resistor on this port line to force the A16 address bit to a logic high state following reset. This port bit is then made an output during the setup code execution, but care must be taken in ensuring that the data register is written to a logic 1 before the data direction register is written with a 1 to make the port line output a high state.

This port bit allows the M68HC11 to access the 128-Kbyte EPROM as two memories of 64 Kbytes each, which are paged by changing the state of the address A16 line on the EPROM. It is important to make sure that the port timing enables the port line to change state, at least the setup and hold time, before the address strobe (E clock rising edge on the MC68HC11G5), otherwise, there could be problems with address timing.

Figure 2 shows a schematic representation of the paging technique for this method where there are two separate 64-Kbyte pages of memory which may be addressed only individually.

This paging scheme means that code cannot directly jump from one 64-Kbyte page to another without running some common area of code during the page switch. This may be accomplished in two basic ways:

- The user code could build a routine in RAM, which is common to both pages, since it is internal and, therefore, unaffected by the port D(5) line.

- The user code could have the same location in both pages devoted to a page change routine.

The example software listing in **Appendix A — Software Paging Scheme** uses the latter approach.

Interrupt Routines   The change of page routine stores the current page before setting or clearing the port D(5) line and then has a jump command which must be at exactly the same address in both pages of memory. This is because the setting or clearing of the port D(5) line will immediately change the page of memory but the program counter will increment normally. Thus a change from page 0 to page 1 will result in the BSET PORTD command from page 0 followed by the JMP 0,X instruction from page 1 (the new page). To enable a jump to work, the X index register has been loaded with the address of the routine to be run in the new page. **Figure 3** shows the execution of code to perform a change of page from page 1 to page 0.

Returning from the interrupt routine requires the return-from-interrupt (RTI) command to be replaced with an RTI routine that checks the RAM location containing the memory page number prior to the interrupt routine execution. The routine then either performs an RTI command immediately, if it is to remain in the same page, or otherwise changes the state of the port D(5) line and then performs an RTI command in the correct page. Note that as with the JMP 0,X command, the RTI must be at the same address in both pages. It is important that the I bit in the condition code register (CCR) (interrupt inhibit) is set during this time for the example code to run correctly. Otherwise, the return page may be altered. This limitation can be overcome by using the stack to maintain a copy of the last page prior to the current interrupt.

The latency for an interrupt routine in a different page from the currently running user code is increased by 21 cycles on entering the interrupt routine and 18 cycles on leaving the interrupt routine. Any interrupt code that could not tolerate any such latency could be repeated in both pages of memory.

Other Routines   Jumping from one page to another may be done at any time by using the same change of page routine, but there is no need to store the current page in RAM, and so these two lines of code become redundant. In the example, the change page routine could be started at the BCLR or BSET command and save four cycles. This would, therefore, reduce the page change delay to 17 cycles. Note that it is not possible to perform a JSR command to move into the other page with the method shown in the example, since the RTS would not return to the original page. However, a modification to the return-from-interrupt routine would allow an equivalent function for a return from subroutine. In this case, the stack should be used to maintain the correct return page or the I bit in the CCR should be set to prevent interrupts.

Important Conditions   The state of the port line controlling address A16 after reset is very important. In the example, port D(5) is used which is an input after reset and has a pullup resistor to force a logic high on A16. If an output-only port line was used, then it could be reset such that A16 is a logic 0 (no pullup resistor required), which has an important consequence. The initialization routine which sets up the ports must be in the default page dictated by the state of address A16 following reset; otherwise, the user code may not be able to correctly configure the ports and hence be unable to manipulate address A16. Similarly, a bidirectional port line could have a pull-down resistor to determine the address A16 line after reset with the same implications.

The assembler generates two blocks of code with identical address ranges used by the user code. This could not be programmed directly into an EPROM since the second page would simply attempt to overwrite the first page. The code must, therefore, be split into two blocks and programmed into the correct half of the EPROM. Some linkers may be capable of performing this function automatically. **Figure 2** illustrates the expansion of the pages into the 128-Kbyte EPROM memory.

The RAM and registers, and internal EEPROM if available and enabled, will all appear in the memory map in preference to external memory, so care must be taken to avoid these addresses or move the RAM or registers away to different addresses by writing to the INIT register.
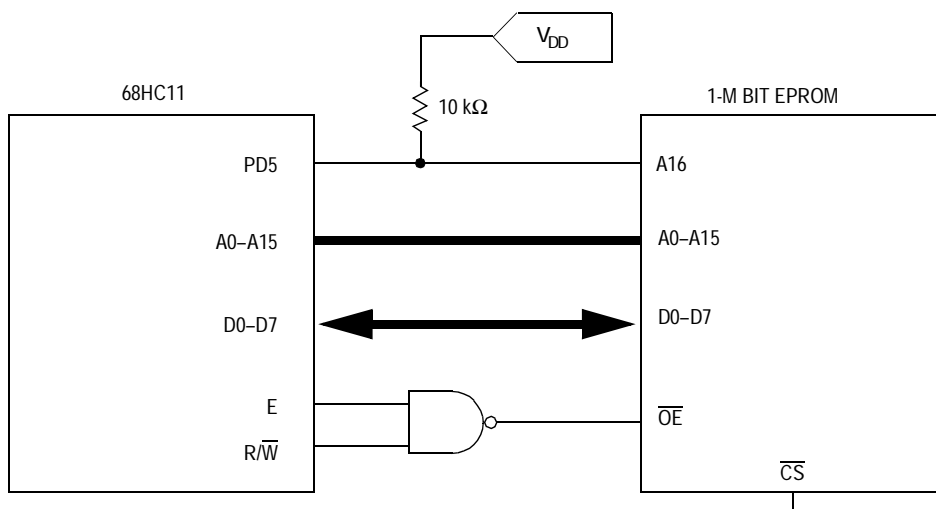
AN432

Figure 1. Software Paging Schematic Diagram
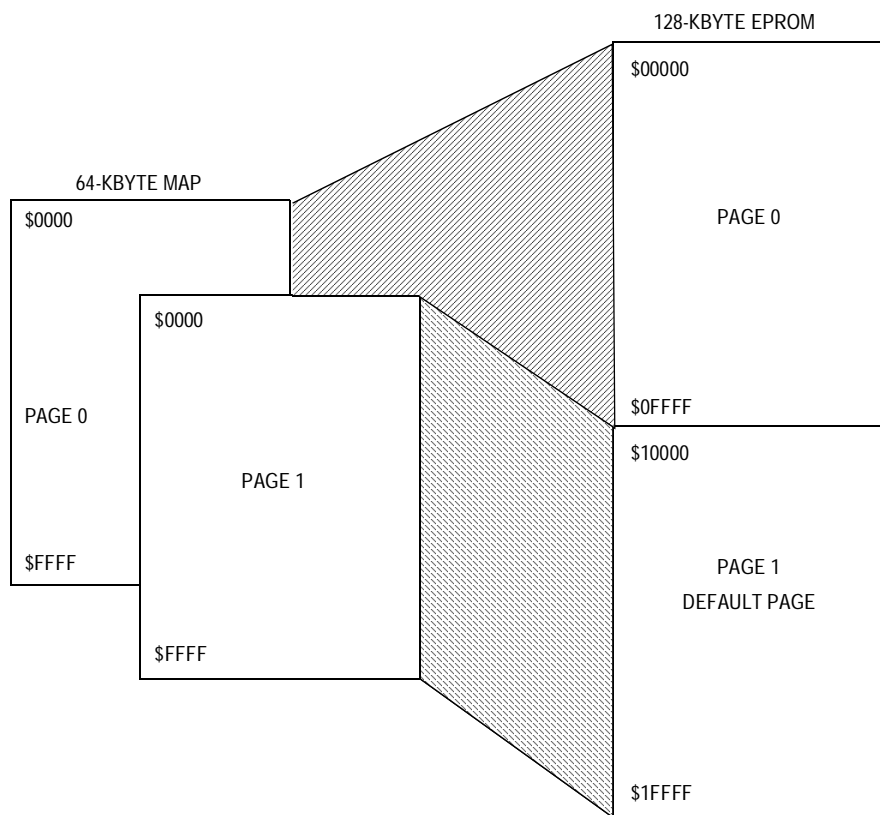
Figure 2. Software Paging Representation

AN432

128-KBYTE EPROM


```
                                              $00000
  TOGGLE PORT A-4
  JUMP TO CHANGE PAGE ROUTINE
                                              PAGE 0

  CHANGE TO PAGE 1                            $0F800
  JMP, X



  VECTORS                                     $0FFFF

                                              $10000
  TOGGLE PORT A-3
  JUMP TO CHANGE PAGE ROUTINE                 PAGE 1

  CHANGE TO PAGE 0                            $1F800
  JMP, X

                                              DEFAULT PAGE



  VECTORS

                                              $1FFFF
```

1 — Jump to change page routine
2 — Page changes to page 0
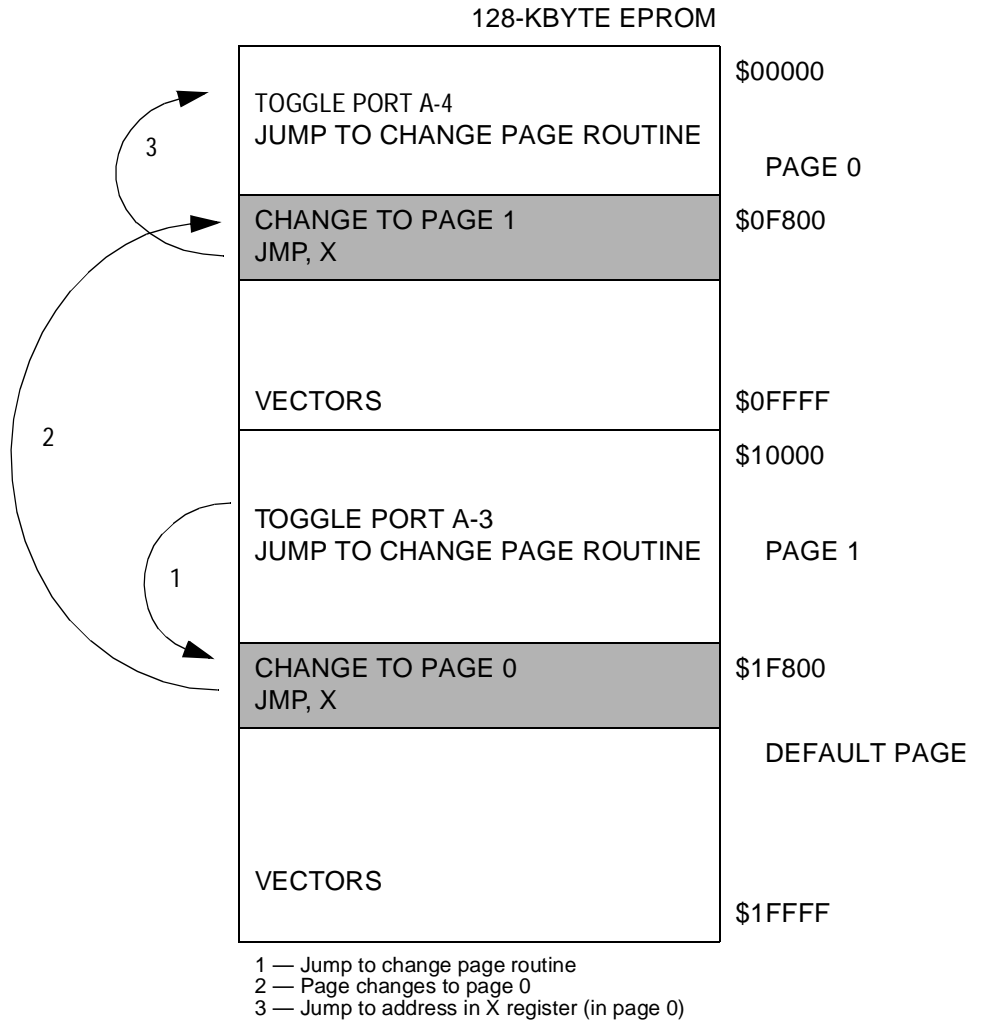3 — Jump to address in X register (in page 0)

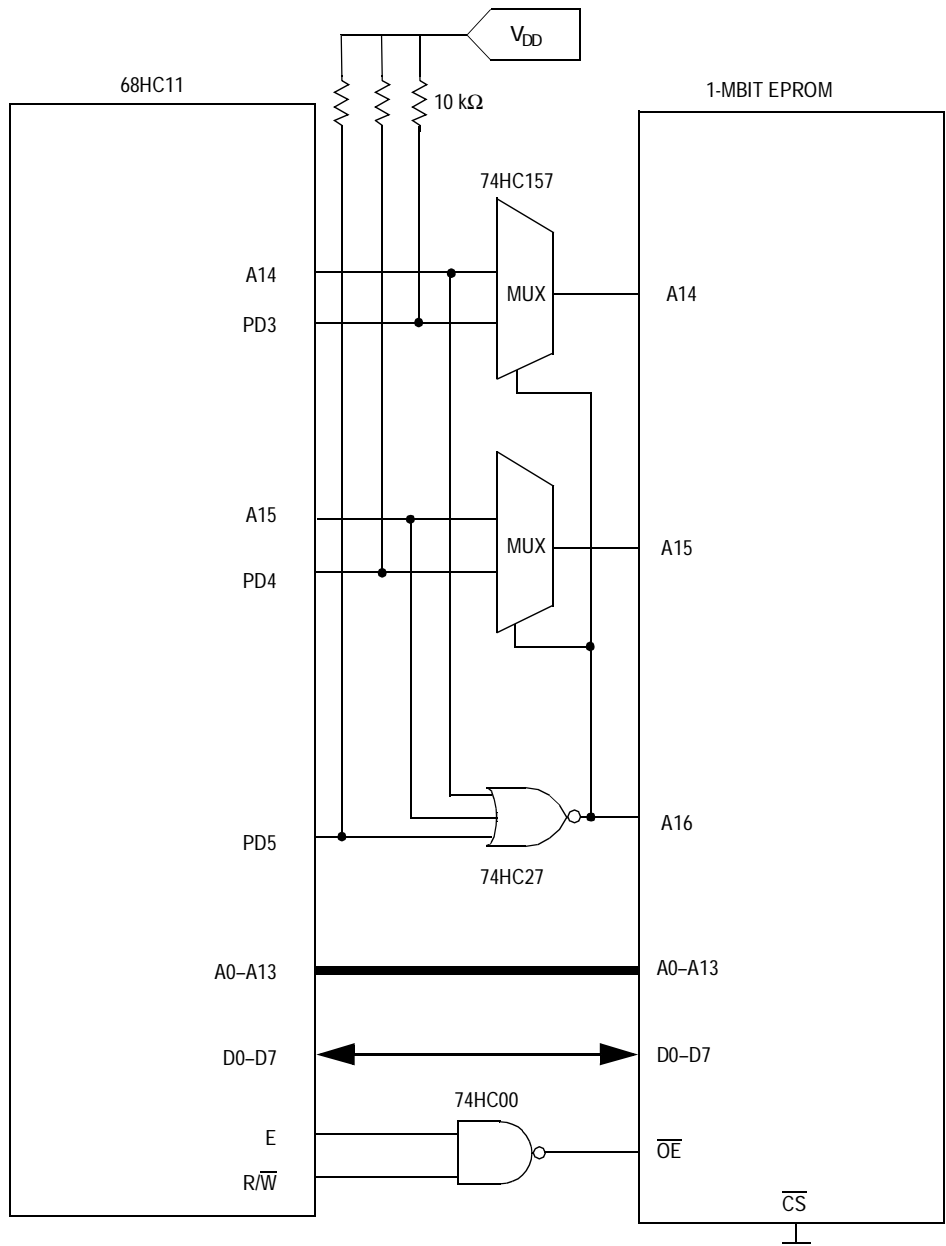**Figure 3. Flow of Program Changing from Page 1 to Page 0**

Figure 4. Hardware and Software Paging Schematic Diagram

Method B — Combined Hardware and Software Technique

The basic approach to this method is the same as in **Method A — Software Technique** except that hardware replaces some of the software. A port line together with M68HC11 addresses A14 and A15 are NORed to control the address A16 line of the EPROM. This signal is also used to select between the port line and address line for A15 and A15 (see **Figure 4**). The hardware between the port lines controlling the A14 and A15 addresses enables 64 Kbytes of user code to be addressed at all times with 48 Kbytes common to all the pages and then selecting one of five 16-Kbyte pages of EPROM memory.

In the example, port D(3) and address A14 are connected to the input of a 2-channel multiplexer such that port D(5), address A14, and address A14 control which of these two signals reaches the A14 pin of the EPROM. If addresses A14 or A15 are logic 1, the NOR gate outputs a logic 0 state, ensuring the A16 pin of the EPROM is a logic 0. In this case, address A14 controls the A14 pin of the EPROM and similarly A15 and port D(4) are selected such that address A15 controls the A15 pin of the EPROM. Thus the main 48-Kbyte portion of the EPROM memory may be addressed at all times at addresses $4000 up to $FFFF. With port D(5) and address A14 and A15 all at logic 0 (address range $0000 to $3FFF), the port lines port D(3) and port D(4) are selected in place of address lines A14 and A15. Page 0 is always selected whenever port D(5) is a logic 1. This makes it possible to have one of the five pages of 16 Kbytes paged into the 64-K addressing range of the HC11 while always maintaining the main 48 Kbytes of user code in the memory map.

There are few restrictions on the user code since the hardware provides the switching logic. Code can be made to run from one paged area to another by jumping to an intermediate routine in the main page. Port D is configured to be in the input state following reset which results in the main page plus page 0 of the paged memory in the 64-Kbyte address map since the port D lines each have a pullup resistor to maintain a logic high state after reset. A simple change memory map routine can then bring in the desired page at any time. **Appendix B — Hardware and Software Paging Scheme** shows the assembly code for a program that toggles different port pins in each of the five pages controlled from a

AN432

main routine in the main page. **Figure 5** shows the five overlaid pages expanded to a 128-K map with the flow of the program demonstrating a change from page 0 to page 1 by running the change page subroutine shown in bold type.

Implementation in C Language

The demonstration code was originally written in assembly language, but it may also be implemented in C, as shown in **Appendix C — C Language Routines for Method B**. The change of page routines were written in C with the first part an example of using in-line code and the second part calling a function. The short example shows the assembly code on the left, generated by the C code on the right. This is very similar to the assembly code example in **Appendix B — Hardware and Software Paging Scheme**, and so it is possible to extend the memory addressing beyond 64-Kbytes with the C language just as with assembly language.

Interrupt Conditions

The interrupt routines have normal latency when they reside in the main 48-Kbytes page since this is always visible to the CPU. The 25-cycle delay for changing pages may cause problems for interrupt routines in a paged area of memory.

Important Conditions

There are few special conditions for this method. The vectors must point to the main page of memory where the page changing routine must also reside. Routines in a paged area can only move to another page via the main 48-Kbyte page unless the technique in method A is utilized (for example, page change routine duplicated at identical addresses in both pages).

As with method A, the RAM and registers and internal EEPROM, if available and enabled, will all appear in the memory map in preference to external memory, so care must be taken to avoid these addresses or move the RAM or registers away to different addresses.

The assembler generates five blocks of code with identical address ranges used by the user code plus the main 48-Kbyte section. This could not be programmed directly into an EPROM since the second and subsequent pages would simply attempt to overwrite the first page. The code must, therefore, be split into blocks and programmed into the correct part of the EPROM. Some linkers may be capable of performing this function automatically.

Figure 6 illustrates the expansion of the pages into a single 128-Kbyte EPROM memory.

Customization   Clearly, the size of the paged areas may be made to suit the application with, for example, a 32-Kbyte main page and three 32 Kbytes of paged memory simply by not implementing control over the A14 address of the EPROM and not including port D(3) control. Similarly, by adding another port line to control address A13, the main program can be 56 Kbytes with nine pages of eight Kbytes each.

TOGGLE PORT A-3
AND RETURN TO MAIN PROGRAM

$00000

PAGE 0

$04000

SET UP PORT D FOR PAGE CONTROL

JSR TO PAGE 0 CHANGE SUBROUTINE
JSR TO PAGE 0

JSR TO PAGE 1 CHANGE SUBROUTINE
JSR TO PAGE 1

MAIN PAGE

JSR TO PAGE 2 CHANGE SUBROUTINE
JSR TO PAGE 2

ETC.

CHANGE TO PAGE 0 AND RETURN
CHANGE TO PAGE 1 AND RETURN
CHANGE TO PAGE 2 AND RETURN

ETC.

VECTORS

TOGGLE PORT A-4
AND RETURN TO MAIN PROGRAM

$10000

PAGE 1

TOGGLE PORT A-5
AND RETURN TO MAIN
PROGRAM

$14000

PAGE 2

TOGGLE PORT A-6
AND RETURN TO MAIN
PROGRAM

$18000

PAGE 3

TOGGLE PORT A-7
AND RETURN TO MAIN
PROGRAM

$1C000

PAGE 4

$1FFFF

1 — Return to page 0
2 — Jump to page 1 routine
3 — Return from page 1 to main page

Figure 5. Illustration of Changing from Page 0 to Page 1

AN432

EXPANDED
MEMORY
ADDRESS

64-KBYTE CPU
ADDRESS

$0000

PAGE 0

$3FFF
$4000

PAGE 1

PAGE 2

PAGE 3

PAGE 4
EACH PAGE
= 16 KBYTES

MAIN PAGE
48 KBYTES

$FFFF

PAGE 0

$00000

$03FFF
$04000

MAIN PAGE

$0FFFF
$10000

PAGE 1

$13FFF
$14000

PAGE 2

$17FFF
$18000

PAGE 3

$1BFFF
$1C000

PAGE 4

$1FFFF
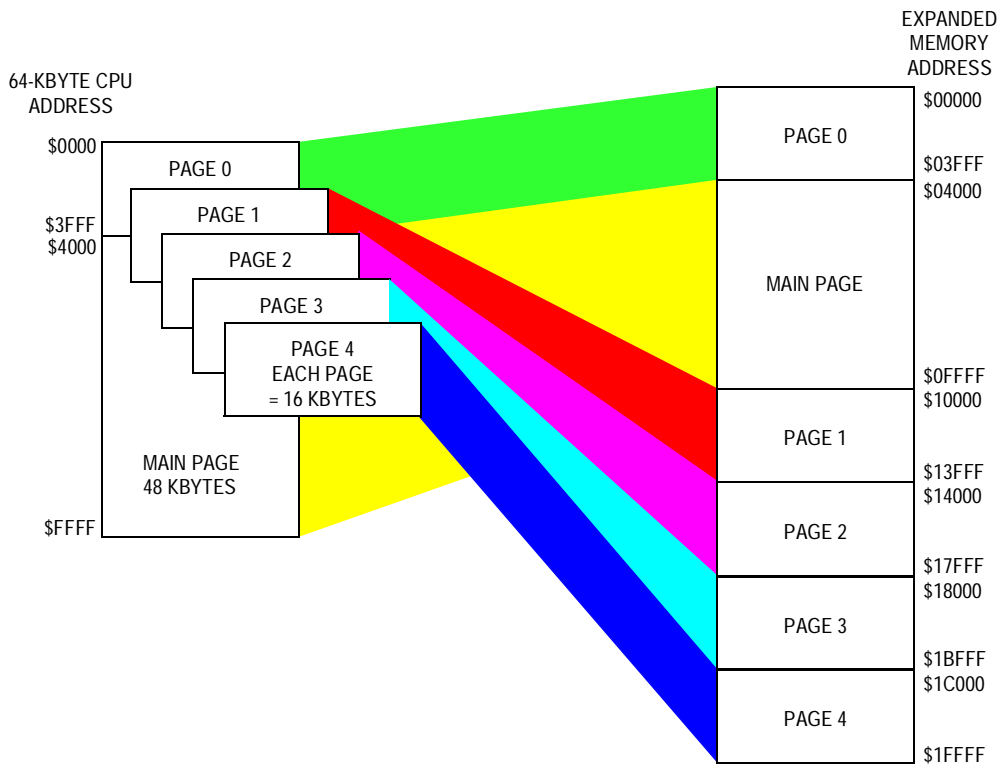
Figure 6. Hardware and Software Paging Representation

METHOD A

$0000

PAGE 1

PAGE 1

$FFFF

METHOD B

$0000

PAGE 0

PAGE 1

PAGE 2

$3FFF

PAGE 3

$4000

PAGE 4

MAIN PAGE

$FFFF

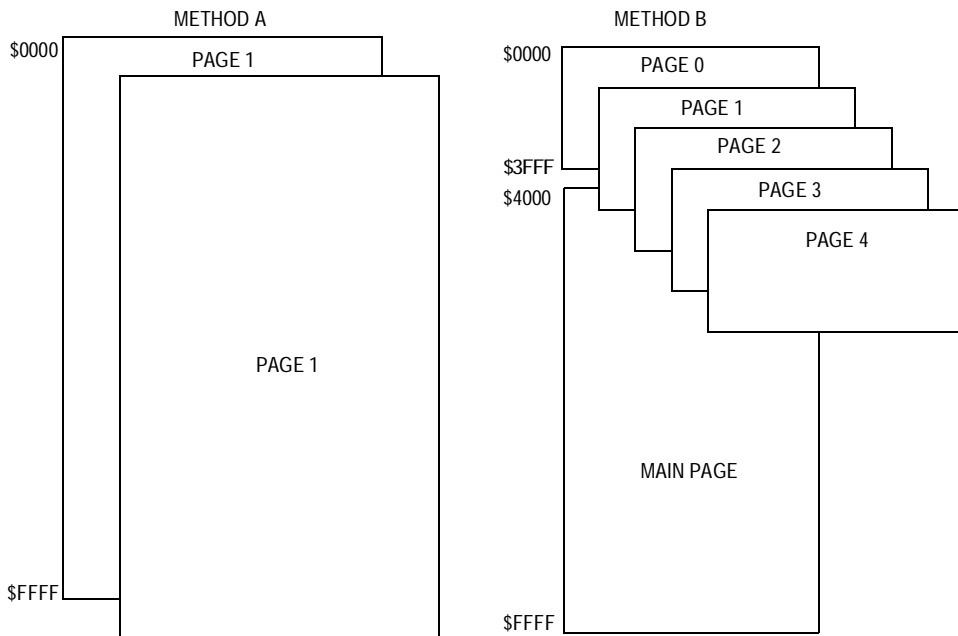**Figure 7. Comparison of Paging Schemes**

AN432

In General

In both methods, the registers may be moved to more appropriate addresses. If the usage of RAM is not critical, the registers may be moved to address $0000 by writing $00 to the INIT register immediately after reset. For the MC68HC11G5, this means losing 128 bytes of RAM, but results in a clean memory map above $1FF. In the examples, the registers and RAM remain at the default addresses and so care must be taken not to have user code from address $0000 to $01FF and $1000 to $107F for the MC68HC11G5. Note that the MC68HC11E9 and MC68HC11A8 have slightly different RAM and register address ranges plus the internal EEPROM which should be disabled if not used.

Figure 7 demonstrates the differences between the paging techniques by showing the overlap of the pages. The number and size of the pages can easily be modified by small changes to the page change routines and hardware.

Beyond 128 Kbytes

Both techniques may be scaled up with several port lines controlling address lines beyond address A15 with the addition of further change page routines and enhancing the return-from-interrupt routine to allow a return to a specific page in method A or the addition of further multiplexing logic in method B.

In Conclusion

The two methods described in detail are the basis for many other ways of controlling paging on a single large EPROM memory device or several smaller EPROMs. It is a simple matter to scale up or modify the techniques to suit a particular application or EPROM.

The software approach is the cheapest and allows for a main program of up to the full size of the EPROM while the combined hardware and software approach has a maximum main program size of 48 Kbytes (in this example) and no additional interrupt latency.

AN432

MOTOROLA

Appendix A — Software Paging Scheme

```
  1                              **** EXTENDA.ASC ***************************************************************
  2                              *
  3                              * TESTS EXTENDED MEMORY CONTROL
  4                              *
  5                              * For a single 1M bit (128K byte) EPROM split into 2 x 64K byte pages
  6                              * A16 is connected to Port D(5) which then selects which half of
  7                              * the EPROM is being accessed. PD5 = 1 after reset since it is in
  8                              * the input state with a pull-up resistor to Vdd.
  9                              *
 10                              * This code is written for the 68HC11G5 MCU but can be easily modified
 11                              * to run on any 68HC11 device. The 68HC11G5 has a non-multiplexed
 12                              * address and data bus in expanded mode.
 13                              *
 14                              *
 15                              *
 16                              *
 17                              ******************************************************************************
 18                              *
 19  00000000          PORTA   EQU             $00
 20  00000001          DDRA    EQU             $01
 21  00000004          PORTB   EQU             $04
 22  00000006          PORTC   EQU             $06
 23  00000007          DDRC    EQU             $07
 24  00000008          PORTD   EQU             $08
 25  00000009          DDRD    EQU             $09
 26  00000024          TMSK2   EQU             $24
 27  00000025          TFLG2   EQU             $25
 28  00000040          RTII    EQU             $40
 29  00000040          RTIF    EQU             $40
 30  00000026          PACTL   EQU             $26
 31  00000080          DDRA7   EQU             $80
 32  00001000          REGS    EQU             $1000
 33                              *
 34                              ******************************************************************************
 35                              *
 36                              *         RAM definitions (from $0000 to $01FF)
 37                              *
 38                              ******************************************************************************
 39                                        ORG     $0000
 40  00000000          PAGE              RMB     1               page number prior to interrupt
 41  00000001          TIME              RMB     2               counter value for real time interrupt routine
 42                              *
 43  00000020          NPAGE             EQU     $20             PORT D-5 page control line
 44  00000200          ROMBASE           EQU     $0200           Avoid RAM (from $0 to $1FF)
 45  0000f800          CHANGE            EQU     $F800
 46  0000ffcc          VECTORS           EQU     $FFCC
 47                              *
 48
 49                              ******************************************************************************
 50                              *         START OF MAIN PROGRAM
 51                              *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*******************
 52                              *
 53                              *         page 0 (1st half of EPROM)
 54                              *
 55                              *
 56                              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 57                                        org     ROMBASE
 58                              ******************************************************************************
 59                              *
 60                              *         Redirect reset vector to page 1
 61                              *
 62                              ******************************************************************************
 63  00000200 ce0200   RESET0            LDX     #RESET
 64  00000203 7ef800            JMP     CHGPAGE0
 65
 66                              ******************************************************************************
 67                              *
 68                              *         2nd half of page 0 loop running in page 1
 69                              *
 70                              ******************************************************************************
 71  00000206 181c0010 LOOPP0            BSET    PORTA,Y,#$10    Toggle bit 4
 72  0000020a 181d0010          BCLR    PORTA,Y,#$10
 73  0000020e ce0216            LDX     #LOOPP1         get return address in page 1
 74  00000211 7ef800            JMP     CHGPAGE0        jump to change page routine
 75                              *
 76                              ******************************************************************************
 77                              *
 78                              *         Real time interrupt service routine
 79                              *
 80                              ******************************************************************************
```

AN432

```
 81  00000214 181e254001 RTISRV          BRSET   TFLG2,Y,#RTIF,RTISERV   return if not correct interrupt source
 82  00000219 3b                          RTI                             This is an RTI because interrupt vector
 83                        *                                              only points here when in page 1
 84                        *
 85                        *
 86  0000021a              RTISERV
 87  0000021a 8640                         LDAA    #%01000000     page 0 interrupt starts here
 88  0000021c 18a725                       STAA    TFLG2,Y        clear RTI flag
 89  0000021f 9602                         LDAA    TIME+1         get the time counter
 90  00000221 4c                           INCA                   increment counter
 91  00000222 b71004                       STAA    PORTB+REGS     store time in port B
 92  00000225 de01                         LDX     TIME
 93  00000227 08                           INX
 94  00000228 fd01                         STX     TIME           and copy back into RAM
 95  0000022a 7ef80a                       JMP     RETRTI0        jump to RTI routine
 96                        *
 97
 98                        *****************************************************************************************
 99                        *
100                        * CHANGE PAGE ROUTINE
101                        *
102                        * This code must be executed with the I-bit set to prevent interrupts
103                        * during the change if it is a jump for an interrupt routine.
104                        * Otherwise PAGE could be updated and then another interrupt could
105                        * occur before the PAGE was changed causing the first interrupt
106                        * routine to return to the wrong page
107                        * The PAGE variable is not required for a normal jump and so it does
108                        * not require the I-bit to be set (only the BSET is important).
109                        *
110                        * This code is repeated for the same position in both pages
111                        *****************************************************************************************
112                        *               jump routine
113                        *               ORG     CHANGE         Address for this routine is fixed
114                        *                       cycles
115  0000f800             CHGPAGE0
116  0000f800 8600                         LDAA    #0      2      set current page number = 0
117  0000f802 9700                         STAA    PAGE    2      store page page number
118  0000f804 181c0820                     BSET    PORTD,Y,#NPAGE  8  change page by setting PD-5
119  0000f808 6e00                         JMP     0,X     3      This code is the same in both pages
120                        *
121                        *****************************************************************************************
122                        *       return from interrupt routine running in page 0
123                        *
124                        *
125                        *       check if interrupt occurred while code was running in page 1
126                        *       and return to page 1 before the RTI command is performed
127                        *
128                        *****************************************************************************************
129                        *                       cycles
130  0000f80a                              RETRTI0
131  0000f80a 9600                         LDAA    PAGE    2      get page the interrupt occured in
132  0000f80c 8101                         CMPA    #1      2      is it page 1
133  0000f80e 2701                         BEQ     RTIPAGE0  3    if yes then change page
134  0000f810 3b                           RTI               12   otherwise, return from interrupt
135  0000f811                              RTIPAGE0
136  0000f811 181c0820                     BSET    PORTD,Y,#NPAGE  8   change page and return from interrupt
137  0000f815 3b                           RTI               12   This codes is the same in both pages
138                        *
139
140                        *****************************************************************************************
141                        *               VECTORS
142                        *****************************************************************************************
143                        *
144                                        ORG     VECTORS
145  0000ffcc 0200                         FDB     RESET0         EVENT 2
146  0000ffce 0200                         FDB     RESET0         EVENT 1
147  0000ffd0 0200                         FDB     RESET0         TIMER OVERFLOW 2
148  0000ffd2 0200                         FDB     RESET0         INPUT CAPTURE 6 / OUTPUT COMPARE 7
149  0000ffd4 0200                         FDB     RESET0         INPUT CAPTURE 5 / OUTPUT COMPARE 6
150  0000ffd6 0200                         FDB     RESET0         SCI
151  0000ffd8 0200                         FDB     RESET0         SPI
152  0000ffda 0200                         FDB     RESET0         PULSE ACC INPUT
153  0000ffdc 0200                         FDB     RESET0         PULSE ACC OVERFLOW
154  0000ffde 0200                         FDB     RESET0         TIMER OVERFLOW 1
155  0000ffe0 0200                         FDB     RESET0         INPUT CAPTURE 4 / OUTPUT COMPARE 5
156  0000ffe2 0200                         FDB     RESET0         OUTPUT COMPARE 4
157  0000ffe4 0200                         FDB     RSEET0         OUTPUT COMPARE 3
158  0000ffe6 0200                         FDB     RESET0         OUTPUT COMPARE 2
159  0000ffe8 0200                         FDB     RESET0         OUTPUT COMPARE 1
160  0000ffea 0200                         FDB     RESET0         INPUT CAPTURE 3
161  0000ffec 0200                         FDB     RESET0         INPUT CAPTURE 2
162  0000ffee 0200                         FDB     RESET0         INPUT CAPTURE 1
163  0000fff0 0214                         FDB     RTISRV         REAL TIME INTERRUPT
164  0000fff2 0200                         FDB     RESET0         IRQ
165  0000fff4 0200                         FDB     RESET0         XIRQ
166  0000fff6 0200                         FDB     RESET0         SWI
167  0000fff8 0200                         FDB     RESET0         ILLEGAL OPCODE
```

AN432

```
168  0000fffa 0200                              FDB     RESET0       COP
169  0000fffc 0200                              FDB     RESET0       CLOCK MONITOR
170  0000fffe 0200                              FDB     RESET0       RESET
171          ****************************************************************************************
172
173          *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
174          *
175          *                page 1 (2nd half of EPROM)
176          *
177          *
178          *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
179          ****************************************************************************************
180          *
181          *               MAIN ROUTINE NOT UNDER INTERRUPT CONTROL
182          *
183          ****************************************************************************************
184
185                                              ORG     ROMBASE
186  00000200 8e01ff            RESET           LDS     #$01FF
187  00000203 bd021b                            JSR     SETUP        initialize RTI interrupt and DDRs
188  00000206 86ff              LOOP1           LDAA    #$FF
189  00000208 181c0008          LOOP            BSET    PORTA,Y,#$08  Toggle bit 3
190  0000020c 181d0008                          BCLR    PORTA,Y,#$08
191  00000210 ce0206                            LDX     #LOOPP0      set up jump to other page
192  00000213 7ef800                            JMP     CHGPAGE1     go to other page
193  00000216                   LOOPP1
194  00000216 4a                                DECA                 return point from other page
195  00000217 26ef                              BNE     LOOP         toggle port A
196  00000219 20eb                              BRA     LOOP1        start loop again
197          *
198          ****************************************************************************************
199          *               INITIALIZATION ROUTINE
200          ****************************************************************************************
201          *
202  0000021b 0f                SETUP           SEI
203  0000021c 18ce1000                          LDY     #$1000       Register address offset
204  00000220 86ff                              LDAA    #$FF
205  00000222 b71001                            STAA    DDRA+REGS    make port A all outputs
206  00000225 b71008                            STAA    PORTD+REGS   make sure port D-5 is written a 1
207  00000228 b71009                            STAA    DDRD+REGS    and only then make all outputs
208  0000022b 8640                              LDAA    #%01000000
209  0000022d b71025                            STAA    TFLG2+REGS   clear RTI flag
210  00000230 b71024                            STAA    TMSK2+REGS   enable RTI interrupt
211  00000233 0e                                CLI
212  00000234 39                                RTS
213          ****************************************************************************************
214          *
215          *               Redirect to the Real time interrupt service routine
216          *               Page 1 routine for service routine located in page 0
217          ****************************************************************************************
218          *
219  00000235 181e254001        INTRI           BRSET   TFLG2,Y,#RTIF,GOODINT
220  0000023a 3b                                RTI                  return if not correct interrupt source
221          *                                                       This is an RTI because interrupt vector
222          *                                                       only points here when in page 1
223          *
224  0000023b                   GOODINT                      cycles
225  0000023b ce021a                            LDX     #RTISERV  3  get the interrupt entry point in page 0
226  00000233 73f800                            JMP     CHPAGE!   3  jump to change page routine
227          *
228
229          ****************************************************************************************
230          *
231          *       CHANGE PAGE ROUTINE
232          *
233          * This code must be executed with the I-bit set to prevent interrupts
234          * during the change if it is a jump for an interrupt routine.
235          * Otherwise PAGE could be updated and then another interrupt could
236          * occur before the PAGE was changed causing the first interrupt
237          * routine to return to the wrong page.
238          * The PAGE variable is not required for a normal jump and so it does
239          * not require the I-bit to be set (only the BCLR is important).
240
241          * This code is repeated for the same position in both pages
242          ****************************************************************************************
243          * jump routine
244          * ORG            CHANGE                Address for this routine is fixed
245          *                              cycles
246  0000f800                   CHGPAGE1
247  0000f800 8601                              LDAA    #$1       2  set current page number = 1
248  0000f802 9700                              STAA    PAGE      2  store page page number
249  0000f804 181d0820                          BCLR    PORTD,Y,#NPAGE 8  Change page by clearing PD-5
250  0000f808 6e00                              JMP     0,X       3  This code is the same in both pages
251          *
```

```
252                     ************************************************************************************
253                     *      return from interrupt routine running in page 0
254                     *
255                     *
256                     *      check if interrupt occurred while code was running in page 1
257                     *      and return to page 0 before the RTI command is performed
258                     *
259                     ************************************************************************************
260                     *                                     cycles
261  0000f80a           RETRTI1
262  0000f80a 9600                      LDAA   PAGE      2       get page the interrupt occured in
263  0000f80c 8100                      CMPA   #0        2       is it page 0
264  0000f80e 2701                      BEQ    RTIPAGE1 3        if yes then change page
265  0000f810 3b                        RTI              12      otherwise, return from interrupt
266  0000f811           RTIPAGE1
267  0000f811 181d0820                  BCLR   PORTD,Y,#NPAGE  8  change page and return from interrupt
268  0000f815 3b                        RTI              12      This codes is the same in both pages
269                     *
270
271                     ************************************************************************************
272                     *                  VECTORS
273                     *
274                     ************************************************************************************
275                                      ORG    VECTORS
276  0000ffcc 0200                       FDB    RESET        EVENT 2
277  0000ffce 0200                       FDB    RESET        EVENT 1
278  0000ffd0 0200                       FDB    RESET        TIMER OVERFLOW 2
279  0000ffd2 0200                       FDB    RESET        INPUT CAPTURE 6 / OUTPUT COMPARE 7
280  0000ffd4 0200                       FDB    RESET        INPUT CAPTURE 5 / OUTPUT COMPARE 6
281  0000ffd6 0200                       FDB    RESET        SCI
282  0000ffd8 0200                       FDB    RESET        SPI
283  0000ffda 0200                       FDB    RESET        PULSE ACC INPUT
284  0000ffdc 0200                       FDB    RESET        PULSE ACC OVERFLOW
285  0000ffde 0200                       FDB    RESET        TIMER OVERFLOW 1
286  0000ffe0 0200                       FDB    RESET        INPUT CAPTURE 4 / OUTPUT COMPARE 5
287  0000ffe2 0200                       FDB    RESET        OUTPUT COMPARE 4
288  0000ffe4 0200                       FDB    RESET        OUTPUT COMPARE 3
289  0000ffe6 0200                       FDB    RESET        OUTPUT COMPARE 2
290  0000ffe8 0200                       FDB    RESET        OUTPUT COMPARE 1
291  0000ffea 0200                       FDB    RESTE        INPUT CAPTURE 3
292  0000ffec 0200                       FDB    RESET        INPUT CAPTURE 2
293  0000ffee 0200                       FDB    RESET        INPUT CAPTURE 1
294  0000fff0 0235                       FDB    INTRI        REAL TIME INTRRUPT
295  0000fff2 0200                       FDB    RESET        IRQ
296  0000fff4 0200                       FDB    RESET        XIRQ
297  0000fff6 0200                       FDB    RESET        SWI
298  0000fff8 0200                       FDB    RESET        ILLEGAL OPCODE
299  0000fffa 0200                       FDB    RESET        COP
300  0000fffc 0200                       FDB    RESET        CLOCK MONITOR
301  0000fffe 0200                                           RESET
302                     ************************************************************************************
303                                      END
```

Appendix B — Hardware and Software Paging Scheme

```
 1              ***************EXTENDB.ASC ****************************************************************
 2              *
 3              * TESTS EXTENDED MEMORY CONTROL
 4              *
 5              * for a single 1M bit (128K byte) EEPROM split into 48KB + 5 x 16KB
 6              * $4000 - $FFFF        48K      COMMON PAGE
 7              * $0200 - $3FFF        16K      PAGES 0,1,2,3,4
 8              *
 9              * A multiplexer is used to switch between address and port D lines
10              * controlled by PD5 and A16 is controlled by /(PD5+A14+A15)
11              * This ensures that Address A16 is a logic 1 whenever A14 or A15 are
12              * high and that all three lines must be low for the paged memory between
13              * addresses $00000 and $0FFFF.
14              *
15              *
16              *              SOURCE CODE                            EPROM
17              *              ADDRESS                                ADDRESS
18              *              0000        ----------------------    00000
19              *                                      PAGE 0
20              *              4000        ----------------------    04000
21              *
22              *                                      MAIN PAGE
23              *
24              *
25              *              0000        ----------------------    10000
26              *                                      PAGE 1
27              *              0000        ----------------------    14000
28              *                                      PAGE 2
29              *              0000        ----------------------    18000
30              *                                      PAGE 3
31              *              0000        ----------------------    1C000
32              *                                      PAGE 4
33              *              3FFF        |---------------------    1FFFF
34              *
35              *
36              *
37              *
38              *        +-__
39              * A14    |       ""-
40              * ------|A
41              *        |
42              *        |MUX      ---------------------+
43              * PD3    |                              |
44              * ------|B         |                   |            --------------------
45              *        |    _     |                   |
46              *        +-""|      |                   |
47              *            |       |                   |
48              *        +----------  +                  |
49              *                    |                   +---------  A14
50              *        +-__        |                   |
51              *        |       ""-  |                   |
52              * A15    |                |                   |
53              * ------|A             |                   |
54              *        |              |                   |
55              *        |MUX      --------------------------------  A15
56              * PD4    |              |
57              * ------|B         |    |
58              *        |    _     |    |                               1M BIT
59              *        +-""|      |    |                               EPROM
60              *            |       |    |
61              *        +-----------+------+
62              *         _____        |
63              * PD5    \      \      \    |
64              * --------\              |
65              * A14             \    |                   |
66              * --------|  NOR  >O-----------+--------------  A16
67              * A15     /      /             |
68              * --------/      /              |
69              *         /_____/               |
70              *                               |
71              *                               |
72              *    PD3, PD4 and PD5 = 1 AFTER RESET     --------------------
73              *    SINCE PULL-UP RESISTORS FORCE HIGH STATE WITH PORT D AS INPUTS
74              *    WHICH DEFAULTS TO MAIN PROGRAM PLUS PAGE 0
75              *
76              *
77              *
78              *
79              *
80              ****************************************************************************************
```

AN432

```
81                        *
82   00000000   PORTA                EQU     $00
83   00000001   DDRA                 EQU     $01          68HC11G5 only
84   00000004   PORTB                EQU     $04
85   00000006   PORTC                EQU     $06
86   00000007   DDRC                 EQU     $07
87   00000008   PORTD                EQU     $08
88   00000009   DDRD                 EQU     $09
89   00000024   TMSK2                EQU     $24
90   00000025   TFLG2                EQU     $25
91   00000040   RTII                 EQU     $40
92   00000040   RTIF                 EQU     $40
93   00000026   PACTL                EQU     $26
94   00000080   DDRA7                EQU     $80          68HC11E9 only
95   00001000   REGS                 EQU     $1000
96                        *
97                        **********************************************************************************
98                        *
99                        *            RAM definitions
100                       *
101                       **********************************************************************************
102                       *           ORG     $0000
103  00000000   TIME                 RMB     2            Real time interrupt routine counter
104                       *
105  00000200   ROMBASE0             EQU     $0200        Avoid RAM (from $0 to $1FF)
106  00000400   ROMBASE1             EQU     $0400
107  0000ffcc   VECTORS              EQU     $FFCC
108                       *
109                       **********************************************************************************
110                       *      PAGE 0 = $00000 - $04FFF (A16=0,A15=0,A14=0) => PAGE0=%00100000
111                       *      MAIN  = $04000 - $0FFFF (A16=0)              => START=%001XX000
112                       *      PAGE 1 = $10000 - $13FFF (A16=1,A15=0,A14=0) => PAGE1=%00000000
113                       *      PAGE 2 = $14000 - $17FFF (A16=1,A15=0,A14=1) => PAGE2=%00001000
114                       *      PAGE 3 = $18000 - $1BFFF (A16=1,A15=1,A14=0) => PAGE3=%00010000
115                       *      PAGE 4 = $1C000 - $1FFFF (A16=1,A15=1,A14=1) => PAGE4=%00011000
116                       *
117                       *      PAGEn is added to %xx000xxx to give the state of port
118                       *      D(3), D(4) and D(5).
119                       *
120  00000000   START                EQU     $00
121  00000020   PAGE0                EQU     $20
122  00000000   PAGE1                EQU     $00
123  00000008   PAGE2                EQU     $08
124  00000010   PAGE3                EQU     $10
125  00000018   PAGE4                EQU     $18
126                       *
127                       **********************************************************************************
128                       *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
129                       *
130                       *            page 0 (1st half of EPROM)
131                       *
132                       *
133                       *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
134                       *                    org     ROMBASE0
135  00000200 181c0008 LOOPP0          BSET    PORTA,Y,#$08
136  00000204 181d0008                 BCLR    PORTA,Y,#$08    Toggle Port A-3
137  00000208 7e4014                   JMP     MAIN0           return to main page
138                       *
139                       *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
140                       *            START OF MAIN PROGRAM
141                       **********************************************************************************
142                       *
143                       *            MAIN ROUTINE NOT UNDER INTERRUPT CONTROL
144                       *
145                       **********************************************************************************
146                       *
147                                    ORG     ROMBASE1
148  00004000 8e01ff  RESET           LDS     #$01FF
149  00004003 bd204e                  JSR     SETUP           initialize RTI interrupt and DDRs
150  00004006 181c0840 LOOP           BSET    PORTD,Y,#$40
151  0000400a 181d0840                BCLR    PORTD,Y,#$40    main routine toggles port D-2
152  0000400e bd4062                  JSR     CHGAGE0         select page 0
153  00004011 7e0200                  JMP     LOOPP0          Toggle Port A-3
154  00004014 bd406d  MAIN0           JSR     CHGPAGE1        select page 1
155  00004017 bd0200                  JSR     LOOPP1          Toggle Port A-4
156  0000401a bd4078                  JRS     CHGPAGE2        select page 2
157  0000401d bd0200                  JSR     LOOPP2          Toggle Port A-5
158  00004020 bd4083                  JSR     CHGPAGE3        select page 3
159  00004023 7e0200                  JMP     LOOPP3          Toggle Port A-6
160  00004026 bd408e  MAIN3           JSR     CHGPAGE4        select page 4
161  00004029 7e0200                  JMP     LOOPP4          Toggle Port A-7
162  0000402c 20d8   MAIN4            BRA     LOOP            start loop again
163                       *
164                       **********************************************************************************
165                       *            INITIALIZATION ROUTINE
166                       **********************************************************************************
167                       *
```

AN432

```
168   0000402e 0f        SETUP               SEI
169   0000402f 18ce1000                      LDY   #$1000           Register address offset
170   00004033 86ff                          LDAA  #$FF
171   00004035 b71001                        STAA  DDRA+REGS        make port A all outputs (68HC11G5)
172   00004038 b71009                        STAA  DDRD+REGS        make port D all outputs
173   0000403b 7f0000                        CLR   TIME
174   0000403e 7f0001                        CLR   TIME+1
175   00004041 4f                            CLRA
176   00004042 b71000                        STAA  PORTA+REGS
177   00004045 b71008                        STAA  PORTD+REGS
178   00004048 8640                          LDAA  #%01000000
179   0000404a b71025                        STAA  TFLG2+REGS       clear the RTI flag
180   0000404d b71024                        STAA  TMSK2+REGS       enable RTI interrupt
181   00004050 0e                            CLI
182   00004051 39                            RTS
183                      *
184                      ********************************************************************************
185                      *
186                      *          Real time interrupt service routine
187                      *
188                      ********************************************************************************
189   00004052 8640      RTISRV              LDAA  #%01000000
190   00004054 b71025                        STAA  TFLG2+REGS       clear TRI flag
191   00004057 9601                          LDAA  TIME+1
192   00004059 b71004                        STAA  PORTB+REGS       store counter in port B
193   0000405c de00                          LDX   TIME             get time counter
194   0000405e 08                            INX                    increment counter
195   0000405f df00                          STX   TIME             save counter value in RAM
196   00004061 3b                            RTI                    Return from interrupt
197                      *
198                      ********************************************************************************
199                      * CHANGE PAGE
200                      * acc B (bits 3-5) contains the 1's complement of new page number address
201                      *
202                      *          SOURCE CODE                      EPROM
203                      *          ADDRESS                          ADDRESS
204                      *          0000     |---------------------| 00000
205                      *                   |        PAGE 0        |
206                      *          4000     |---------------------| 04000
207                      *                   |                     |
208                      *                   |      MAIN PAGE       |
209                      *                   |                     |
210                      *                   |                     |
211                      *          0000     |---------------------| 10000
212                      *                   |        PAGE 1        |
213                      *          0000     |---------------------| 14000
214                      *                   |        PAGE 2        |
215                      *          0000     |---------------------| 18000
216                      *                   |        PAGE 3        |
217                      *          0000     |---------------------| 1C000
218                      *                   |        PAGE 4        |
219                      *          3FFF     |---------------------| 1FFFF
220                      *
221                      * PAGE 0 = $00000 - $03FFF (A16=0,A15=0,A14=0) => PAGE0=%00100000
222                      * MAIN   = $04000 - $0FFFF (A16=0)             => START=%001XX000
223                      * PAGE 1 = $10000 - $13FFF (A16=1,A15=0,A14=0) => PAGE1=%00000000
224                      * PAGE 2 = $14000 - $17FFF (A16=1,A15=0,A14=1) => PAGE2=%00001000
225                      * PAGE 3 = $18000 - $1BFFF (A16=1,A15=1,A14=0) => PAGE3=%00010000
226                      * PAGE 4 = $1C000 - $1FFFF (A16=1,A15=1,A14=1) => PAGE4=%00011000
227                      *
228                      ********************************************************************************
229                      *
230   00004062           CHGPAGE0
231   00004062 b61008                        LDAA  PORTD+REGS       get port D data
232   00004065 84c7                          ANDA  #%11000111       make middle 3 bits low state
233   00004067 8b20                          ADDA  #PAGE0           add PAGE descriptor to this
234   00004069 b71008                        STAA  PORTD+REGS       write back to port D
235   0000406c 39                            RTS                    (only bits 3, 4 and 5 are changed)
236                      *
237   0000406d           CHGPAGE1
238   0000406d b61008                        LDAA  PORTD+REGS       get port D data
239   00004070 84C7                          ANDA  #%11000111       make middle 3 bits low state
240   00004072 8b00                          ADDA  #PAGE1           add PAGE descriptor to this
241   00004074 b71008                        STAA  PORTD+REGS       write back to port D
242   00004077 39                            RTS                    (only bits 3, 4 and 5 are changed)
243                      *
244   00004078           CHGPAGE2
245   00004078 b61008                        LDAA  PORTD+REGS       get port D data
246   0000407b 84c7                          ANDA  #%11000111       make middle 3 bits low state
247   0000407d 8b08                          ADDA  #PAGE2           add PAGE descriptor to this
248   0000407f b71008                        STAA  PORTD+REGS       write back to port D
249   00004082 39                            RTS                    (only bits 3,4 and 5 are changed)
250                      *
251   00004083           CHGPAGE3
252   00004083 b61008                        LDAA  PORTD+REGS       get port D data
253   00004086 84c7                          ANDA  #%11000111       make middle 3 bits low state
254   00004088 8b10                          ADDA  #PAGE3           add PAGE descriptor to this
```

```
255  0000408a b71008                         STAA   PORTD+REGS    write back to port D
256  0000408d 39                             RTS                  (only bits 3, 4 and 5 are changed)
257              *
258  0000408e          CHGPAGE4
259  0000408e b61008                         LDAA   PORTD+REGS    get port D data
260  00004091 84c7                           ANDA   #%11000111    make middle 3 bits low state
261  00004093 8b18                           ADDA   #PAGE4        add PAGE descriptor to this
262  00004095 b71008                         STAA   PORTD+REGS    write back to port D
263  00004098 39                             RTS                  (only bits 3, 4 and 5 are changed)
264              *
265              ********************************************************************************************
266              *                    VECTORS
267              ********************************************************************************************
268              *
269                                           ORG    VECTORS
270  0000ffcc 4000                           FDB    RESET         EVENT 2
271  0000ffce 4000                           FDB    RESET         EVENT 1
272  0000ffd0 4000                           FDB    RESET         TIMER OVERFLOW 2
273  0000ffd2 4000                           FDB    RESET         INPUT CAPTURE 6 / OUTPUT COMPARE 7
274  0000ffd4 4000                           FDB    RESET         INPUT CAPTURE 5 / OUTPUT COMPARE 6
275  0000ffd6 4000                           FDB    RESET         SCI
276  0000ffd8 4000                           FDB    RESET         SPI
277  0000ffda 4000                           FDB    RESET         PULSE ACC INPUT
278  0000ffdc 4000                           FDB    RESET         PULSE ACC OVERFLOW
279  0000ffde 4000                           FDB    RESET         TIMER OVERFLOW 1
280  0000ffe0 4000                           FDB    RESET         INPUT CAPTURE 4 / OUTPUT COMPARE 5
281  0000ffe2 4000                           FDB    RESET         OUTPUT COMPARE 4
282  0000ffe4 4000                           FDB    RESET         OUTPUT COMPARE 3
283  0000ffe6 4000                           FDB    RESET         OUTPUT COMPARE 2
284  0000ffe8 4000                           FDB    RESET         OUTPUT COMPARE 1
285  0000ffea 4000                           FDB    RESET         INPUT CAPTURE 3
286  0000ffec 4000                           FDB    RESET         INPUT CAPTURE 2
287  0000ffee 4000                           FDB    RESET         INPUT CAPTURE 1
288  0000fff0 4052                           FDB    RTISRV        REAL TIME INTRRUPT
289  0000fff2 4000                           FDB    RESET         IRQ
290  0000fff4 4000                           FDB    RESET         XIRQ
291  0000fff6 4000                           FDB    RESET         SWI
292  0000fff8 4000                           FDB    RESET         ILLEGAL OPCODE
293  0000fffa 4000                           FDB    RESET         COP
294  0000fffc 4000                           FDB    RESET         CLOCK MONITOR
295  0000fffe 4000                           FDB    RESET         RESET
296              *
297              ********************************************************************************************
298              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
299              *
300              *                    page 1 (2nd half of EPROM)
301              *
302              *
303              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
304                                           org    ROMBASE0
305  00000200 181c0010 LOOPP1                 BSET   PORTA,Y,#$10
306  00000204 181d0010                        BCLR   PORTA,Y,#$10  Toggle Port A-4
307  00000208 39                              RTS
308              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
309              *
310              *                    page 2 (2nd half of EPROM)
311              *
312              *
313              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
314                                           org    ROMBASE0
315  00000200 181c0020 LOOPP2                 BSET   PORTA,Y,#$20
316  00000204 181d0020                        BCLR   PORTA,Y,#$20  Toggle Port A-5
317  00000208 39                              RTS
318              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
319              *
320              *                    page 3 (2nd half of EPROM)
321              *
322              *
323              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
324                                           org    ROMBASE0
325  00000200 181c0040 LOOPP3                 BSET   PORTA,Y,#$40
326  00000204 181d0040                        BCLR   PORTA,Y,#$40  Toggle Port A-6
327  00000208 7e4026                          JMP    MAIN3         return to main page
328              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
329              *
330              *                    page 4 (2nd half of EPROM)
331              *
332              *
333              *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
334                                           org    ROMBASE0
335  00000200 181c0080 LOOPP4                 BSET   PORTA,Y,#$80
336  00000204 181d0080                        BLCR   PORTA,Y,#$80  Toggle Port A-7
337  00000208 7e402c                          JMP    MAIN4         return to main page
338              ********************************************************************************************
339                                           END
```

Appendix C — C Language Routines for Method B

```
              /* CHGPAGE.C
                 C coded extended memory control for 68HC11
 *
 *            */
 *
 *
 *****************************************************************************
 /*      HC11 structure - I/O registers for MC68HC11 */
 struct HC11IO {
 unsigned     char      PORTA;                                          /* Port A - 3 input only, 5 output only */
 unsigned     char      Reserved;                                       /* Motorola's unknown register */
 unsigned     char      PIOC;                                           /* Parallel I/O control */
 unsigned     char      PORTC;                                          /* Port C */
 unsigned     char      PORTB;                                          /* Port B - Output only */
 unsigned     char      PORTCL;                                         /* Alternate port C latch */
 unsigned     char      Reserved1;                                      /* Motorola's unknown register 2 */
 unsigned     char      DDRC;                                           /* Data direction for port C */
 unsigned     char      PORTD;                                          /* Port D */
 unsigned     char      DDRD;                                           /* Data direction for port D */
 unsigned     char      PORTE;                                          /* Port E */
 };
 /*           End of structure HC11IO */
 *****************************************************************************
 *            #define regbase (*(struct HC11IO *) 0x1000)
 *            typedef unsigned char byte;
 *
 *            /* Some arbitrary user defined values */
 *            #define page0 0x20
 *            #define page1 0x00
 *            #define page2 0x08
 *            #define pagemask 0xc7
 *
 *            /* Macro to generate in line code */
 *            #define chgpage (a) regbase.PORTD = (regbase.PORTD & pagemask) + a
 *
 *            /* Function prototype */
 *            void func_chgpage (byte p);
 *
 *            /* Externally defined functions in separate pages */
 *            extern void func_in_page0();                              /* Dummy function in page 0 */
 *            extern void func_in_page2();                              /* Dummy function in page 2 */
 *
 *------------------------compiled assembly code ---------------------------C source code----------------------------
 *                                                             main()
   6 0000                  main:          fbegin
 *                                                             {
 *                                                             chgpage(page2);
                                                               /* Change page using inline code */
   8 0000    f61008                       ldab           $1008
   9 0003    c4c7                         andb           $199
  10 0005    cb08                         addb           #8
  11 0007    f71008                       stab           $1008
 *                                                             func_in_page2();
                                                               /* Call function in page 2 */
  13 000a    >bd0000                      jsr            func_in_page2
 *
 *                                                             func_chgpage(page0);
                                                               /* Change page using function call */
  15 000d    cc0020                       ldd            #32
  16 0010    8d04                         bsr            func_chgpage
 *                                                             func_in_page0();
                                                               /* Call function in page 0 */
  18 0012    >bd0000                      jsr            func_in-page0
 *                                                             }
  20 0015    39                           rts
  21 0016                                 fend
 *                                                             void func_chgpage(p)
 *                                                             byte p;
  24 0016                  func_chgpage:                 fbegin
  25 0016    37                           pshb
 *                                                             {
 *                                                             chgpage(p);
  27 0017    f61008                       ldab           $1008
  28 001a    c4c7                         andb           #199
  29 001c    30                           tsx
  30 001d    eb00                         addb           0,x
  31 001f    f71008                       stab           $1008
 *                                                             }
  33 0022    31                           ins
  34 0023    39                           rts
  35 0024                                 fend
  36                       import         func_in_page2
  37                       import         func_in_page0
  30                       end
```

Application Note

How to reach us:
   USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447

   JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569

   ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

   Technical Information Center: 1-800-521-6274

   HOME PAGE: http://www.motorola.com/semiconductors/

MOTOROLA

AN432/D