Order this document
as AN432/D

MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

AN432

128K byte 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 64K bytes, 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 (128K bytes) 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.

PAGING SCHEME

The M68HC11 8-bit MCU is capable of addressing up to 64K bytes of contiguous address space. Addressing greater than 64K bytes 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 64K bytes 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 minimised 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 3 address bits A14, A15 and A16.

In the examples below, 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 64K bytes 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 48K bytes with a further 5 paged areas of 16K bytes for subroutines and tables.

© MOTOROLA LTD., 1990

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 10KΩ pull-up 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 set-up code execution but care must be taken in ensuring that the data register is written to a logic one before the data direction register is written with a one to make the port line output a high state.

This port bit allows the M68HC11 to access the 128K byte EPROM as two memories of 64K bytes each which are paged by changing the state of the address A16 line on the EPROM. It 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 64K byte pages of memory which may only be addressed individually.

This paging scheme means that code cannot directly jump from one 64K page to another without running some common area of code during the page switch. This may be accomplished in 2 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) or have the same location in both pages devoted to a page change routine. The example software listing in appendix A 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 RTI command to be replaced with a return from interrupt 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 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 4 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 pull-up 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 zero (no pull-up resistor required) which has an important consequence. The initialisation 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 128K byte 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.

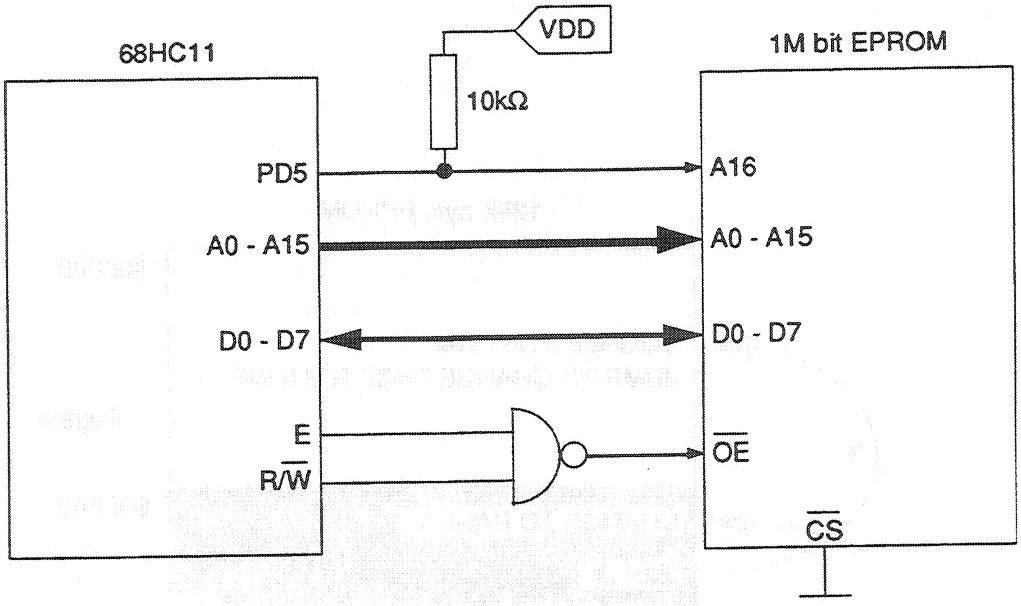


Figure 1. Software Paging Schematic Diagram

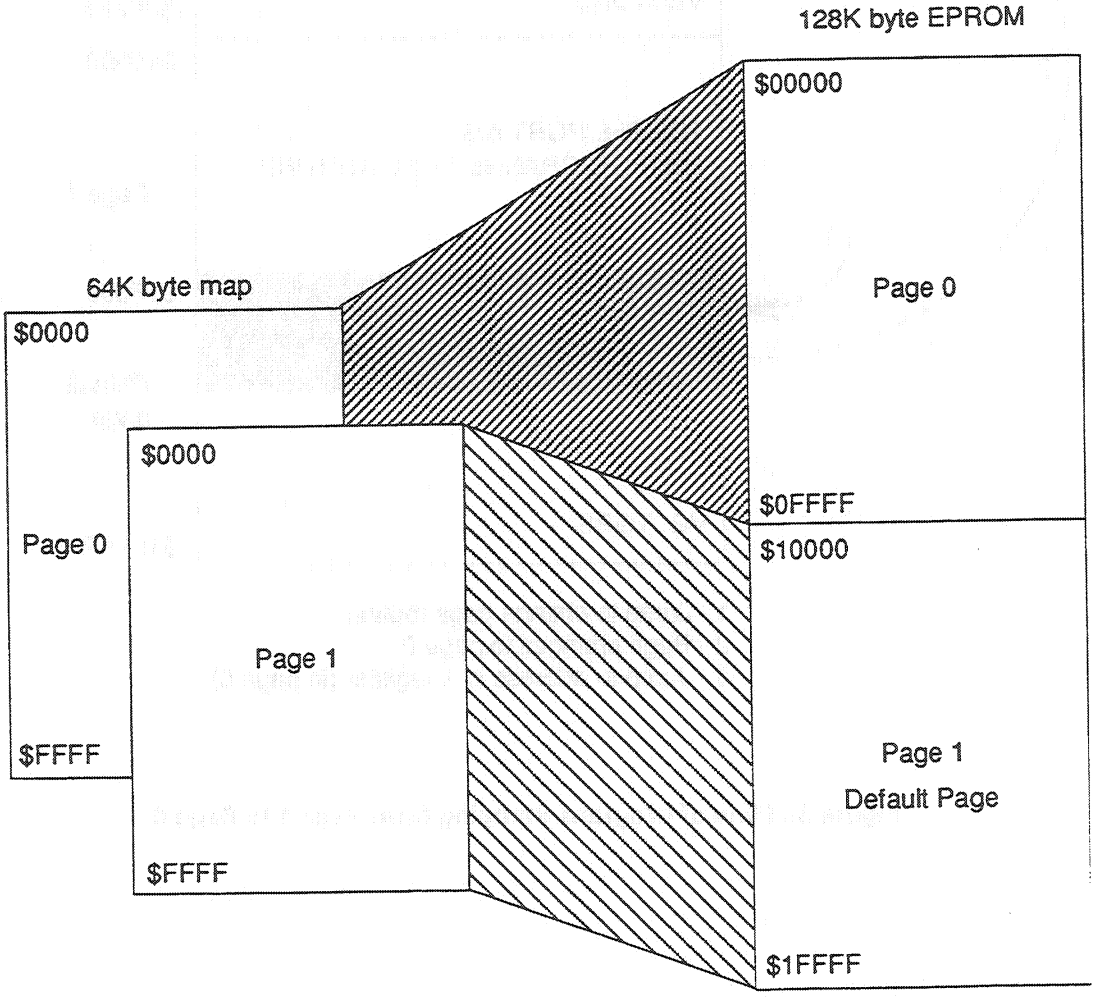


Figure 2. Software Paging Representation

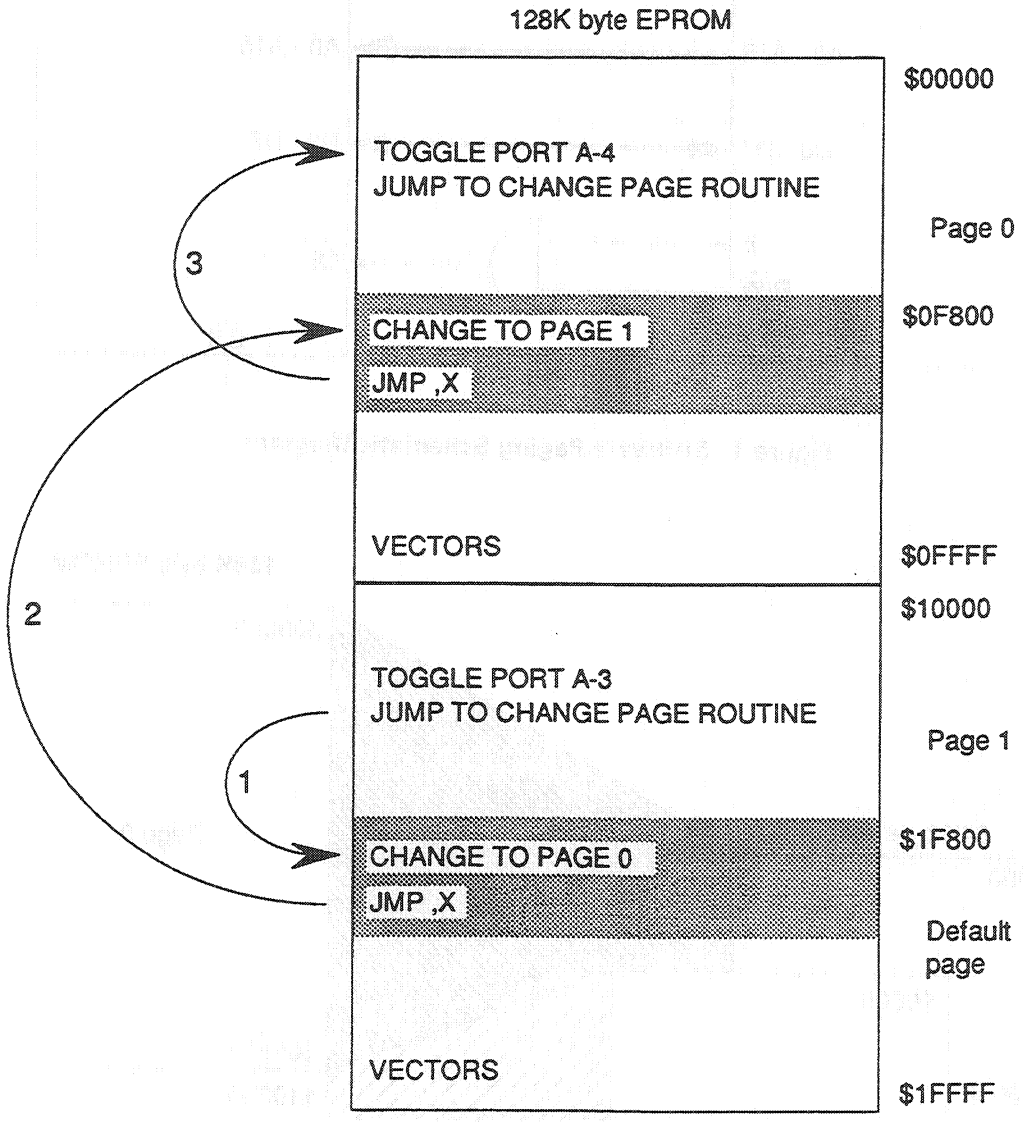


Figure 3. Flow of program changing from Page 1 to Page 0

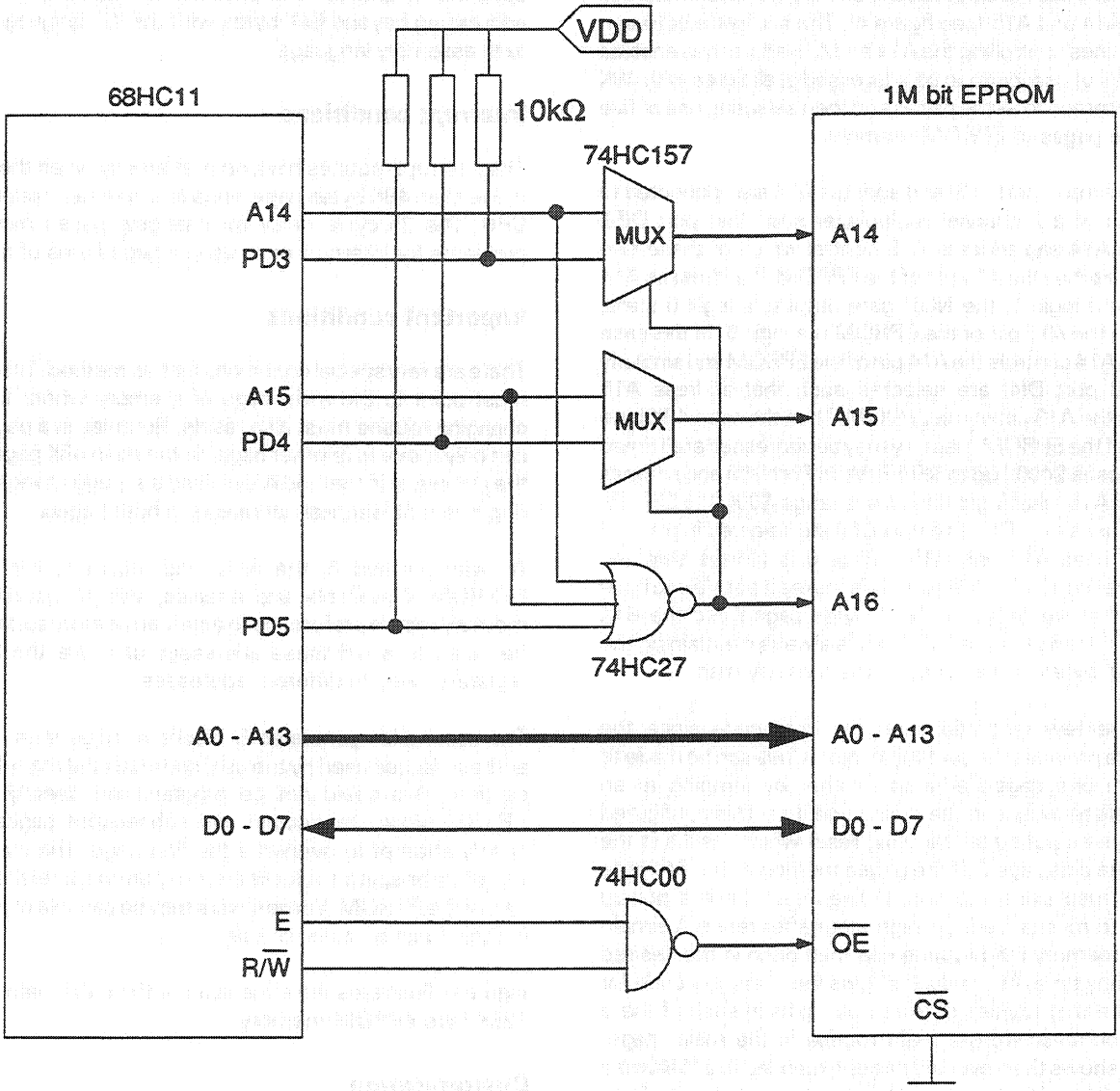


Figure 4. Hardware and Software Paging Schematic Diagram

METHOD B – COMBINED HARDWARE AND SOFTWARE TECHNIQUE

The basic approach to this method is the same as above except that hardware replaces some of the software. A port line together with M68HC11 addresses A14 and A15 are NOR'd to control the address A16 line of the EPROM. This signal is also used to select between the port line and address line for A14 and A15 (see figure 4). The hardware between the port lines controlling the A14 and A15 addresses enables 64K bytes of user code to be addressed at all times with 48K bytes common to all the pages and then selecting one of five 16K byte 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 A15 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 48K byte 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 K bytes paged into the 64K addressing range of the HC11 while always maintaining the main 48K bytes 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 64K byte address map since the port D lines each have a pull-up 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 shows the assembly code for a program that toggles different port pins in each of the 5 pages controlled from a main routine in the main page. Figure 5 shows the 5 overlaid pages expanded to a 128K 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. 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 and so it is possible to extend the memory addressing beyond 64K bytes with the 'C' language just as with assembly language.

Interrupt conditions

The interrupt routines have normal latency when they reside in the main 48K bytes 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 48K page unless the technique in method A is utilised (i.e. page change routine duplicated at identical addresses in both pages).

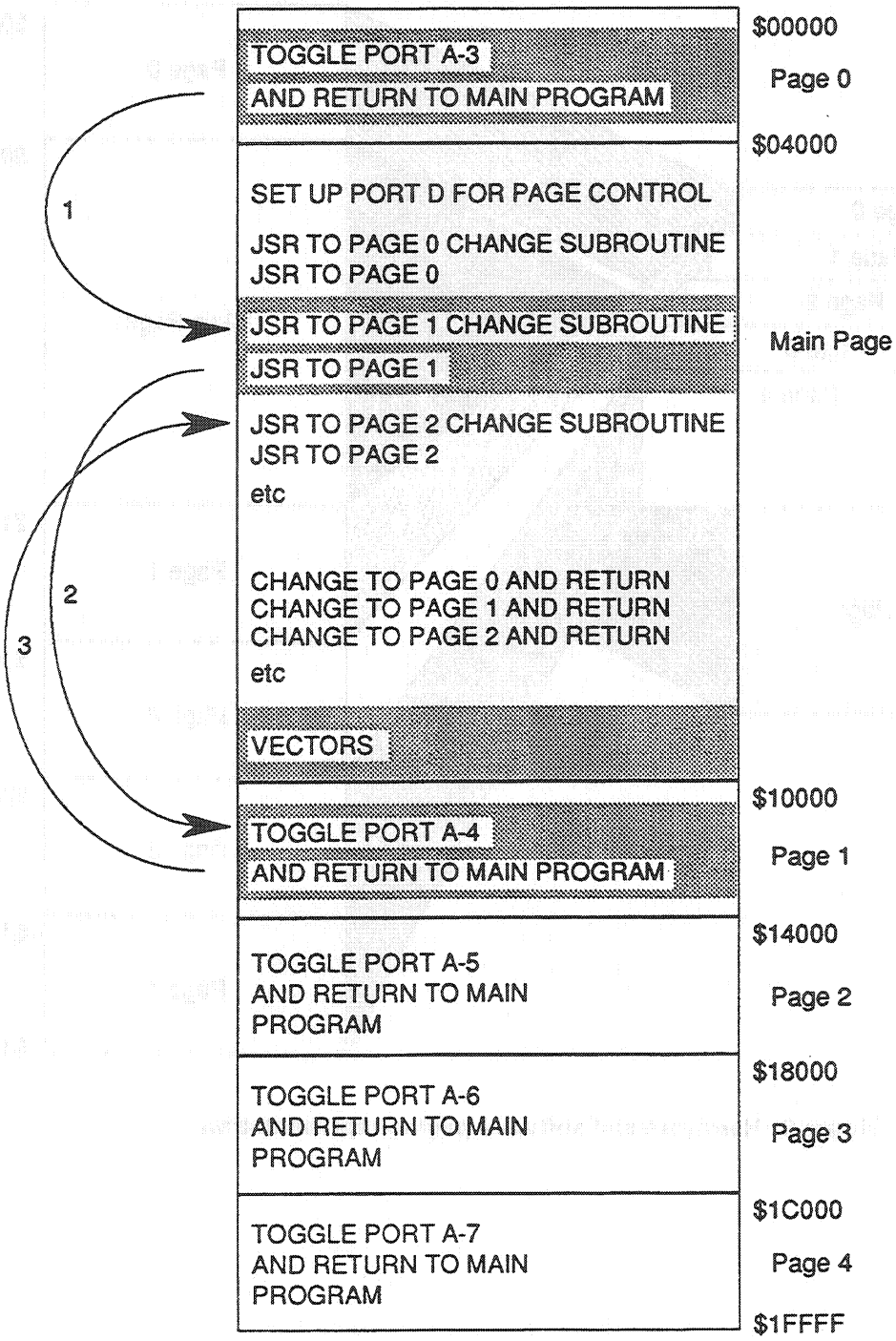
As with method A, the RAM and registers, and internal EEPROM if available and enabled, will all appear in the memory map in preference to external memory so care must be taken to avoid these addresses or move the RAM or registers away to different addresses.

The assembler generates 5 blocks of code with identical address ranges used by the user code plus the main 48K byte 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 128K byte EPROM memory.

Customisation

Clearly the size of the paged areas may be made to suit the application with for example a 32K byte main page and three 32K bytes 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 56K bytes with 9 pages of 8K bytes each.



1 - Return from 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

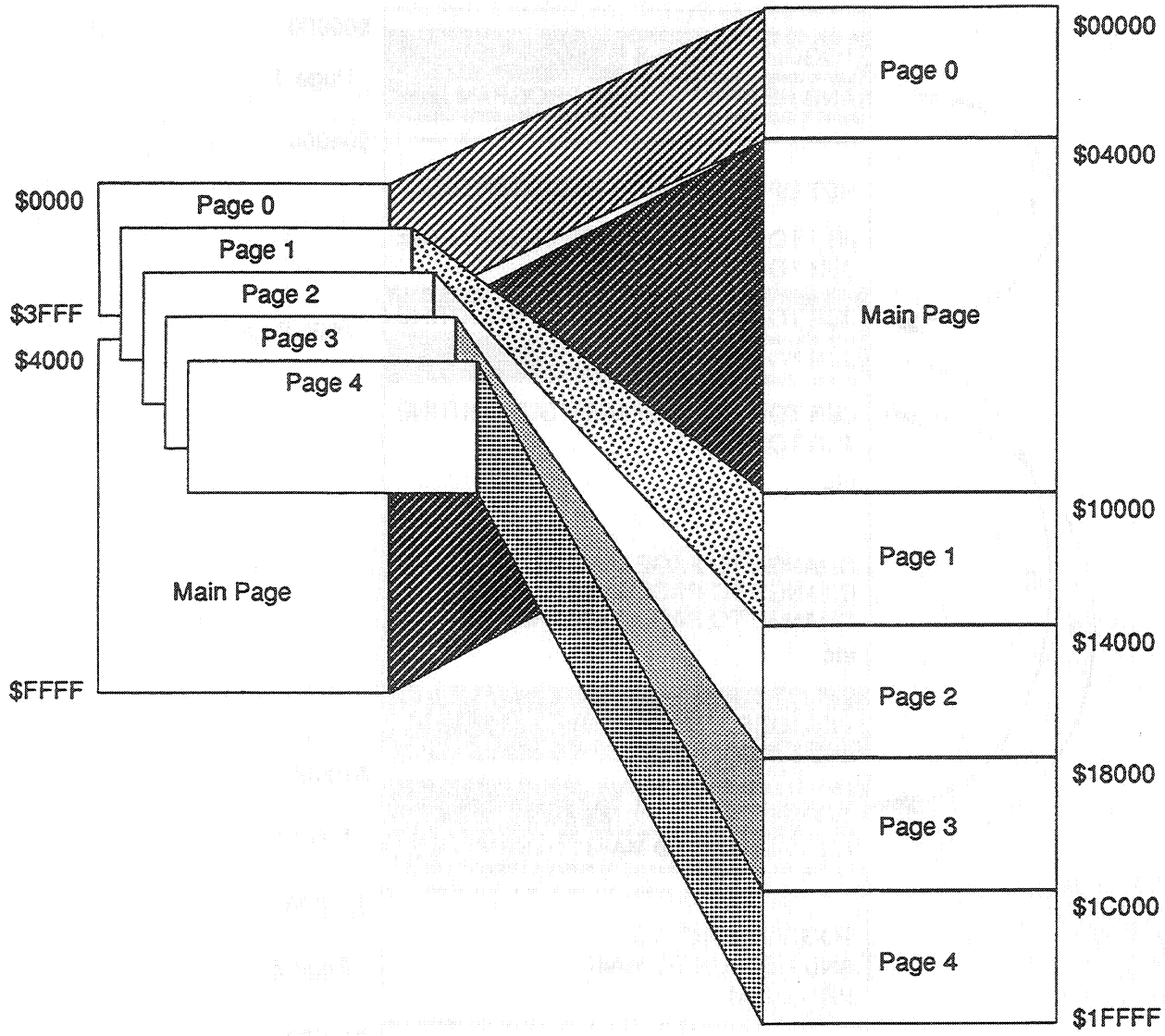


Figure 6. Hardware and software paging representation

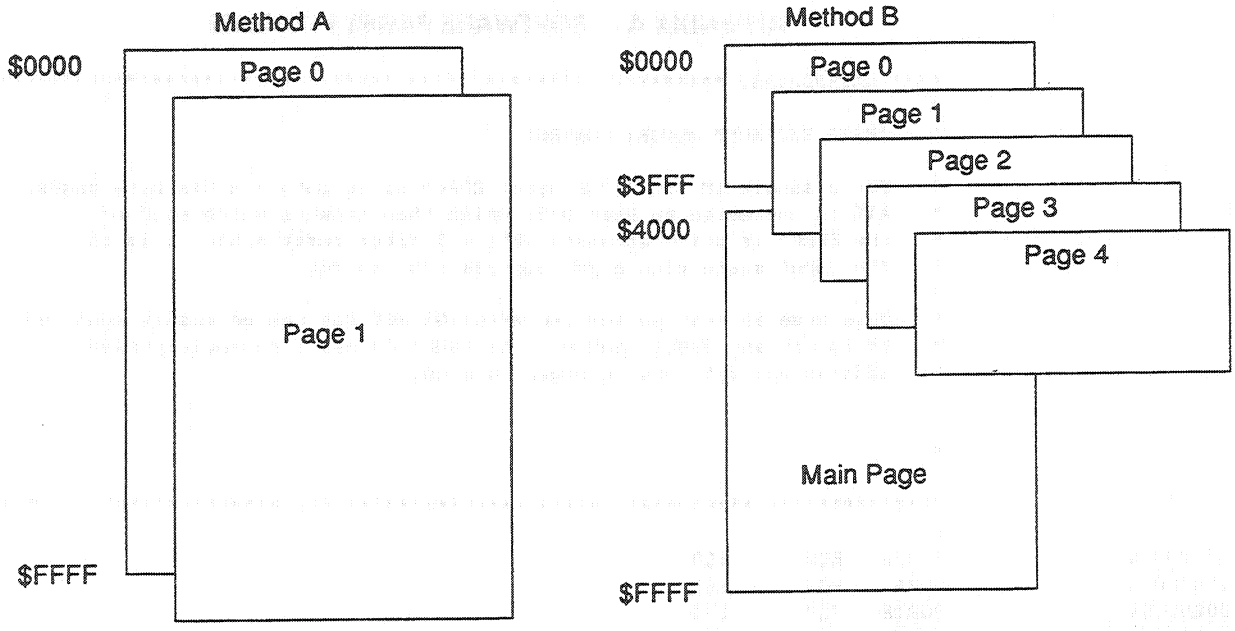


Figure 7. Comparison of paging schemes

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 128K bytes

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 48K bytes (in this example) and no additional interrupt latency.

APPENDIX A - SOFTWARE PAGING SCHEME

```
                        **** EXTENDA.ASC ************************************************************
                        *
                        *   TESTS EXTENDED MEMORY CONTROL
                        *
                        *   For a single 1M bit (128K byte) EPROM split into 2 x 64K byte pages.
                        *   A16 is connected to Port D(5) which then selects which half of
                        *   the EPROM is being accessed. PD5 = 1 after reset since it is in
                        *   the input state with a pull-up resistor to Vdd.
                        *
                        *   This code is written for the 68HC11G5 MCU but can be easily modified
                        *   to run on any 68HC11 device. The 68HC11G5 has a non-multiplexed
                        *   address and data bus in expanded mode.
                        *
                        *
                        *
                        *
                        ********************************************************************************
                        *
00000000                PORTA   EQU     $00
00000001                DDRA    EQU     $01
00000004                PORTB   EQU     $04
00000006                PORTC   EQU     $06
00000007                DDRC    EQU     $07
00000008                PORTD   EQU     $08
00000009                DDRD    EQU     $09
00000024                TMSK2   EQU     $24
00000025                TFLG2   EQU     $25
00000040                RTII    EQU     $40
00000040                RTIF    EQU     $40
00000026                PACTL   EQU     $26
00000080                DDRA7   EQU     $80
00001000                REGS    EQU     $1000
                        *
                        ********************************************************************************
                        *
                        *       RAM definitions (from $0000 to $01FF)
                        *
                        ********************************************************************************
                                ORG     $0000
00000000                PAGE    RMB     1       page number prior to interrupt
00000001                TIME    RMB     2       counter value for real time interrupt routine
                        *
00000020                NPAGE   EQU     $20     PORT D-5 page control line
00000200                ROMBASE EQU     $0200   Avoid RAM (from $0 to $1FF)
0000f800                CHANGE  EQU     $F800
0000ffcc                VECTORS EQU     $FFCC
                        *


                        ********************************************************************************
                        *       START OF MAIN PROGRAM
                        *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        *
                        *       page 0 (1st half of EPROM)
                        *
                        *
                        *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              org       ROMBASE
                        ********************************************************************************
                        *
                        *       Redirect reset vector to page 1
                        *
                        ********************************************************************************
```

```
00000200  ce0200      RESET0 LDX       #RESET
00000203  7ef800             JMP       CHGPAGE0

                       ********************************************************************************
                       *
                       *       2nd half of page 0 loop running in page 1
                       *
                       ********************************************************************************
00000206  181c0010   LOOPP0 BSET      PORTA,Y,#$10       Toggle bit 4
0000020a  181d0010          BCLR      PORTA,Y,#$10
0000020e  ce0216            LDX       #LOOPP1            get return address in page 1
00000211  7ef800            JMP       CHGPAGE0           jump to change page routine
                       *
                       ********************************************************************************
                       *
                       *       Real time interrupt service routine
                       *
                       ********************************************************************************
00000214  181e254001 RTISRV BRSET     TFLG2,Y,#RTIF,RTISERV
00000219  3b                RTI                          return if not correct interrupt source
                       *                                 This is an RTI because interrupt vector
                       *                                 only points here when in page 1
                       *
0000021a             RTISERV
0000021a  8640              LDAA      #%01000000         page 0 interrupt starts here
0000021c  18a725            STAA      TFLG2,Y            clear RTI flag
0000021f  9602              LDAA      TIME+1             get the time counter
00000221  4c                INCA                         increment counter
00000222  b71004            STAA      PORTB+REGS         store time in port B
00000225  de01              LDX       TIME
00000227  08                INX
00000228  df01              STX       TIME               and copy back into RAM
0000022a  7ef80a            JMP       RETRTI0            jump to RTI routine
                       *

                       ********************************************************************************
                       *
                       *   CHANGE PAGE ROUTINE
                       *
                       *   This code must be executed with the I-bit set to prevent interrupts
                       *   during the change if it is a jump for an interrupt routine.
                       *   Otherwise PAGE could be updated and then another interrupt could
                       *   occur before the PAGE was changed causing the first interrupt
                       *   routine to return to the wrong page.
                       *   The PAGE variable is not required for a normal jump and so it does
                       *   not require the I-bit to be set (only the BSET is important).
                       *
                       *   This code is repeated for the same position in both pages
                       ********************************************************************************
                       *       jump routine
                              ORG    CHANGE                Address for this routine is fixed
                       *                            cycles
0000f800             CHGPAGE0
0000f800  8600              LDAA    #0              2     set current page number = 0
0000f802  9700              STAA    PAGE            2     store page page number
0000f804  181c0820          BSET    PORTD,Y,#NPAGE  8     change page by setting PD-5
0000f808  6e00              JMP     0,X             3     This code is the same in both pages
                       *
                       ********************************************************************************
                       *       return from interrupt routine running in page 0
                       *
                       *
                       *       check if interrupt occurred while code was running in page 1
                       *       and return to page 1 before the RTI command is performed
                       *
                       ********************************************************************************
```

```
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     RESET0            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 INTRRUPT
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
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     #$01FF
187 00000203  bd021b               JSR     SETUP             initialise RTI interrupt and DDRs
188 00000206  86ff           LOOP1 LDAA    #$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                          *
```

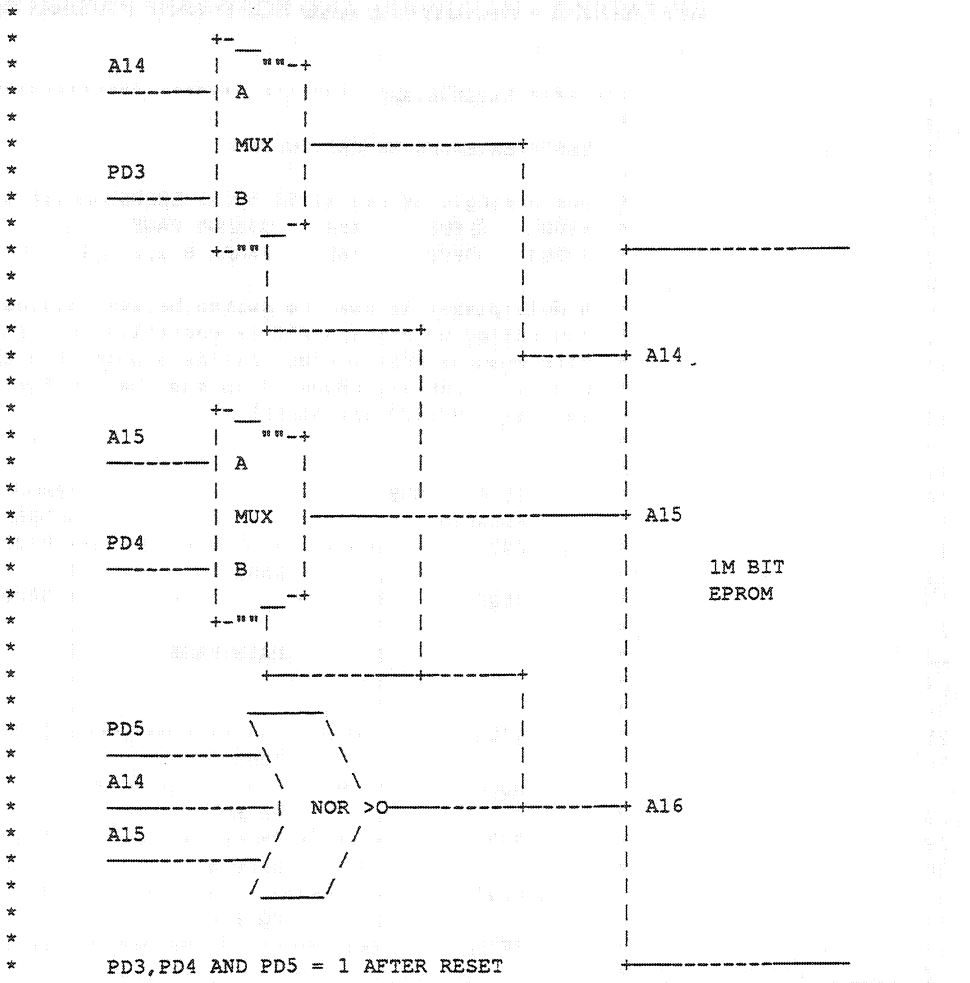
```
198                     ***************************************************************************
199                     *       INITIALISATION ROUTINE
200                     ***************************************************************************
201                     *
202 0000021b  0f        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 INTRTI 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 0000023e  7ef800           JMP    CHGPAGE1     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     RESET               INPUT CAPTURE 3
292 0000ffec  0200          FDB     RESET               INPUT CAPTURE 2
293 0000ffee  0200          FDB     RESET               INPUT CAPTURE 1
294 0000fff0  0235          FDB     INTRTI              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          FDB     RESET               RESET
302                     ********************************************************************
303                         END
```

APPENDIX B – HARDWARE AND SOFTWARE PAGING SCHEME

```
******** EXTENDB.ASC ************************************************
*
*  TESTS EXTENDED MEMORY CONTROL
*
*  for a single 1M bit (128K byte) EEPROM split into 48KB + 5 x 16KB
*  $4000 - $FFFF     48K    COMMON PAGE
*  $0200 - $3FFF     16K    PAGES 0,1,2,3,4
*
*  A multiplexer is used to switch between address and port D lines
*  controlled by PD5 and A16 is controlled by  /(PD5+A14+A15)
*  This ensures that Address A16 is a logic 1 whenever A14 or A15 are
*  high and that all three lines must be low for the paged memory between
*  addresses $00000 and $0FFFF.
*
*
*       SOURCE CODE                          EPROM
*       ADDRESS                              ADDRESS
*       0000       +-------------------------+ 00000
*                  |       PAGE 0            |
*       4000       + - - - - - - - - - - - - + 04000
*                  |                         |
*                  |       MAIN PAGE         |
*                  |                         |
*                  |                         |
*       0000       +-------------------------+ 10000
*                  |       PAGE 1            |
*       0000       +-------------------------+ 14000
*                  |       PAGE 2            |
*       0000       +-------------------------+ 18000
*                  |       PAGE 3            |
*       0000       +-------------------------+ 1C000
*                  |       PAGE 4            |
*       3FFF       +-------------------------+ 1FFFF
*
```

(Continued overleaf)


A14
A
MUX
PD3
B
A15
A
MUX
PD4
B
PD5
A14
NOR >O
A15
A14
A14
A15
A16
1M BIT
EPROM
PD3,PD4 AND PD5 = 1 AFTER RESET
SINCE PULL-UP RESISTORS FORCE HIGH STATE WITH PORT D AS INPUTS
WHICH DEFAULTS TO MAIN PROGRAM PLUS PAGE 0

```
                        *
00000000                PORTA       EQU     $00
00000001                DDRA        EQU     $01             68HC11G5 only
00000004                PORTB       EQU     $04
00000006                PORTC       EQU     $06
00000007                DDRC        EQU     $07
00000008                PORTD       EQU     $08
00000009                DDRD        EQU     $09
00000024                TMSK2       EQU     $24
00000025                TFLG2       EQU     $25
00000040                RTII        EQU     $40
00000040                RTIF        EQU     $40
00000026                PACTL       EQU     $26
00000080                DDRA7       EQU     $80             68HC11E9 only
00001000                REGS        EQU     $1000
                        *
                        ********************************************************************************
                        *
                        *       RAM definitions
                        *
                        ********************************************************************************
                              ORG $0000
00000000                TIME        RMB   2                   Real time interrupt routine counter
                        *
00000200                ROMBASE0    EQU   $0200               Avoid RAM (from $0 to $1FF)
00004000                ROMBASE1    EQU   $4000
0000ffcc                VECTORS     EQU   $FFCC
                        *
                        ********************************************************************************
                        *       PAGE 0 = $00000 - $03FFF  (A16=0,A15=0,A14=0) => PAGE0=%00100000
                        *       MAIN   = $04000 - $0FFFF  (A16=0)             => START=%001XX000
                        *       PAGE 1 = $10000 - $13FFF  (A16=1,A15=0,A14=0) => PAGE1=%00000000
                        *       PAGE 2 = $14000 - $17FFF  (A16=1,A15=0,A14=1) => PAGE2=%00001000
                        *       PAGE 3 = $18000 - $1BFFF  (A16=1,A15=1,A14=0) => PAGE3=%00010000
                        *       PAGE 4 = $1C000 - $1FFFF  (A16=1,A15=1,A14=1) => PAGE4=%00011000
                        *
                        *       PAGEn is added to %xx000xxx to give the state of port
                        *       D(3), D(4) and D(5).
                        *
00000000                START   EQU     $00
00000020                PAGE0   EQU     $20
00000000                PAGE1   EQU     $00
00000008                PAGE2   EQU     $08
00000010                PAGE3   EQU     $10
00000018                PAGE4   EQU     $18
                        *
                        ********************************************************************************
```

```
                        *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        *
                        *       page 0 (1st half of EPROM)
                        *
                        *
                        *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                               org       ROMBASE0
00000200  181c0008    LOOPP0 BSET      PORTA,Y,#$08
00000204  181d0008           BCLR      PORTA,Y,#$08       Toggle Port A-3
00000208  7e4014             JMP       MAIN0              return to main page
                        *
                        *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        *        START OF MAIN PROGRAM
                        ***************************************************************************
                        *
                        *       MAIN ROUTINE NOT UNDER INTERRUPT CONTROL
                        *
                        ***************************************************************************
                        *
                               ORG       ROMBASE1
00004000  8e01ff      RESET  LDS       #$01FF
00004003  bd402e             JSR       SETUP              initialise RTI interrupt and DDRs
00004006  181c0840    LOOP   BSET      PORTD,Y,#$40
0000400a  181d0840           BCLR      PORTD,Y,#$40       main routine toggles port D-2
0000400e  bd4062             JSR       CHGPAGE0           select page 0
00004011  7e0200             JMP       LOOPP0             Toggle Port A-3
00004014  bd406d      MAIN0  JSR       CHGPAGE1           select page 1
00004017  bd0200             JSR       LOOPP1             Toggle Port A-4
0000401a  bd4078             JSR       CHGPAGE2           select page 2
0000401d  bd0200             JSR       LOOPP2             Toggle Port A-5
00004020  bd4083             JSR       CHGPAGE3           select page 3
00004023  7e0200             JMP       LOOPP3             Toggle Port A-6
00004026  bd408e      MAIN3  JSR       CHGPAGE4           select page 4
00004029  7e0200             JMP       LOOPP4             Toggle Port A-7
0000402c  20d8        MAIN4  BRA       LOOP               start loop again
                        *
                        ***************************************************************************
                        *       INITIALISATION ROUTINE
                        ***************************************************************************
                        *
0000402e  0f          SETUP  SEI
0000402f  18ce1000           LDY     #$1000               Register address offset
00004033  86ff               LDAA    #$FF
00004035  b71001             STAA    DDRA+REGS            make port A all outputs (68HC11G5)
00004038  b71009             STAA    DDRD+REGS            make port D all outputs
0000403b  7f0000             CLR     TIME
0000403e  7f0001             CLR     TIME+1
00004041  4f                 CLRA
00004042  b71000             STAA    PORTA+REGS
00004045  b71008             STAA    PORTD+REGS
00004048  8640               LDAA    #%01000000
0000404a  b71025             STAA    TFLG2+REGS           clear the RTI flag
0000404d  b71024             STAA    TMSK2+REGS           enable RTI interrupt
00004050  0e                 CLI
00004051  39                 RTS
                        *
```

```
                        ********************************************************************************
                        *
                        *       Real time interrupt service routine
                        *
                        ********************************************************************************
00004052  8640    RTISRV LDAA     #%01000000
00004054  b71025         STAA     TFLG2+REGS            clear RTI flag
00004057  9601           LDAA     TIME+1
00004059  b71004         STAA     PORTB+REGS            store counter in port B
0000405c  de00           LDX      TIME                  get time counter
0000405e  08             INX                            increment counter
0000405f  df00           STX      TIME                  save counter value in RAM
00004061  3b             RTI                            Return from interrupt
                        *
                        ********************************************************************************
                        *  CHANGE PAGE
                        *  acc B (bits 3-5) contains the 1's complement of new page number address
                        *
                        *  SOURCE CODE                    EPROM
                        *  ADDRESS                        ADDRESS
                        *  0000      +--------------------+ 00000
                        *            |       PAGE 0       |
                        *  4000      + - - - - - - - - - - + 04000
                        *            |                    |
                        *            |       MAIN PAGE    |
                        *            |                    |
                        *            |                    |
                        *  0000      +--------------------+ 10000
                        *            |       PAGE 1       |
                        *  0000      +--------------------+ 14000
                        *            |       PAGE 2       |
                        *  0000      +--------------------+ 18000
                        *            |       PAGE 3       |
                        *  0000      +--------------------+ 1C000
                        *            |       PAGE 4       |
                        *  3FFF      +--------------------+ 1FFFF
                        *
                        *  PAGE 0 = $00000 - $03FFF  (A16=0,A15=0,A14=0) => PAGE0=%00100000
                        *  MAIN   = $04000 - $0FFFF  (A16=0)             => START=%001XX000
                        *  PAGE 1 = $10000 - $13FFF  (A16=1,A15=0,A14=0) => PAGE1=%00000000
                        *  PAGE 2 = $14000 - $17FFF  (A16=1,A15=0,A14=1) => PAGE2=%00001000
                        *  PAGE 3 = $18000 - $1BFFF  (A16=1,A15=1,A14=0) => PAGE3=%00010000
                        *  PAGE 4 = $1C000 - $1FFFF  (A16=1,A15=1,A14=1) => PAGE4=%00011000
                        *
                        ********************************************************************************
```

```
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
```

```
0000ffe8  4000              FDB     RESET               OUTPUT COMPARE 1
0000ffea  4000              FDB     RESET               INPUT CAPTURE 3
0000ffec  4000              FDB     RESET               INPUT CAPTURE 2
0000ffee  4000              FDB     RESET               INPUT CAPTURE 1
0000fff0  4052              FDB     RTISRV              REAL TIME INTRRUPT
0000fff2  4000              FDB     RESET               IRQ
0000fff4  4000              FDB     RESET               XIRQ
0000fff6  4000              FDB     RESET               SWI
0000fff8  4000              FDB     RESET               ILLEGAL OPCODE
0000fffa  4000              FDB     RESET               COP
0000fffc  4000              FDB     RESET               CLOCK MONITOR
0000fffe  4000              FDB     RESET               RESET
                    *
                    ********************************************************************************
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                    *
                    *       page 1 (2nd half of EPROM)
                    *
                    *
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                          org     ROMBASE0
00000200  181c0010  LOOPP1 BSET   PORTA,Y,#$10
00000204  181d0010        BCLR    PORTA,Y,#$10        Toggle Port A-4 .
00000208  39              RTS
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                    *
                    *       page 2 (2nd half of EPROM)
                    *
                    *
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                          org     ROMBASE0
00000200  181c0020  LOOPP2 BSET   PORTA,Y,#$20
00000204  181d0020        BCLR    PORTA,Y,#$20        Toggle Port A-5
00000208  39              RTS
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                    *
                    *       page 3 (2nd half of EPROM)
                    *
                    *
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                          org     ROMBASE0
00000200  181c0040  LOOPP3 BSET   PORTA,Y,#$40
00000204  181d0040        BCLR    PORTA,Y,#$40        Toggle Port A-6
00000208  7e4026          JMP     MAIN3               return to main page
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                    *
                    *       page 4 (2nd half of EPROM)
                    *
                    *
                    *+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
                          org     ROMBASE0
00000200  181c0080  LOOPP4 BSET   PORTA,Y,#$80
00000204  181d0080        BCLR    PORTA,Y,#$80        Toggle Port A-7
00000208  7e402c          JMP     MAIN4               return to main page
                    ********************************************************************************
                          END
```

```
*       /* 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
 38               end
```

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not authorized for use as components in life support devices or systems intended for surgical implant into the body or intended to support or sustain life. Buyer agrees to notify Motorola of any such intended end use whereupon Motorola shall determine availability and suitability of its products for the use intended. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

Literature Distribution Centres:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
ASIA PACIFIC: Motorola Semiconductors (H.K.) Ltd.; Silicon Harbour Center, No. 2, Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawaku, Tokyo 141, Japan.

MOTOROLA
Printed in Great Britain by Tavistock Press (Bedford) Ltd. 4500 7/90