



TECHNICAL UPDATE

MC68HC705J2

Technical Update contains updates to documented information appearing in other Motorola technical documents as well as new information not covered elsewhere.

We are confident that your Motorola product will satisfy your design needs. This Technical Update and the accompanying manuals and reference documentation are designed to be helpful, informative, and easy to use.

Should your application generate a question or a problem not covered in the current documentation, please call your local Motorola distributor or sales office. Technical experts at these locations are eager to help you make the best use of your Motorola product. As appropriate, these experts will coordinate with their counterparts in the factory to answer your questions or solve your problems. To obtain the latest document, call your local Motorola sales office.


Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

TABLE OF CONTENTS

Modules

COP (Computer Operating Properly) Timeout Test Code	3
---	---

MC68HC705J2 Part Specific

Mask Option Register (MOR) — MOR Programming Using External 8K EPROM	5
DC Electrical Characteristics	5
Bootloader ROM — Bootloader Programming Clarification	6
Bootloader Code	7

TECHNICAL UPDATE

Module

COP (Computer Operating Properly) Timeout Test Code COPOCOPRT2

Revision History

Date	Revision	Description
3/26/93	1.00	Initial release. Includes tracker HC705K1.005.
4/25/95	2.00	Includes tracker HC705K1.005 Rev. 2.

Reference Document: Not Applicable

Tracker Number: HC705K1.005

Revision: 2.00

The program below tests the timeout on the COP module COP0COPRT2. The HC705K1 was used to verify operation. The program can be used on any HC05 as long as the part has the COP0COPRT2 module. Memory and reset vectors may need to be changed to work properly with a particular MCU.

*

7K1_COP.ASM – COP Test on the HC705K1

*

* Program Description

*

* This program is a simple routine that tests the COP timeout on the HC705K1 *MCU. The HC705K1 was programmed with the M68HC705KICS board. The part then was *tested for COP resets on a protoboard. If the COP is working correctly, port A *will toggle on approximately one-half second intervals.

*

```
*****
PORTA equ $00
DDRA equ $04
MOR equ $17

ORG MOR
DB $01 ;enable COP

ORG $200

START lda #$FF ;make port A all output
      sta DDRA ;

      com $E0 ;complement RAM mem $E0
      lda $E0 ;ACCA <- ($E0)
      sta PORTA ;port A <- (ACCA)

DONE NOP ;branch into an infinite loop
      BRA DONE ;waiting for a COP timeout

ORG $03FE ;define reset vector
DW START
```

Part Specific**Mask Option Register (MOR) — MOR Programming
Using External 8K EPROM****Revision History**

Date	Revision	Description
3/26/93	1.00	Initial release. Includes trackers HC705J2.001, HC705J2.006, HC705J2.011, and HC705.012.

Subject: MC68HC705J2/D Information, Page 8-4**Tracker Number: HC705J2.001****Revision: 1.00**

To avoid unintentionally enabling any of the options in the mask option register (MOR), the user should ensure that location \$0F00 of the 8-K external EPROM (2764) is programmed with either \$00 or the appropriate value for the options to be enabled. This is necessary since the erased state of an 8-K external EPROM is \$FF, whereas the erased state of the MOR is \$00.

DC Electrical Characteristics**Reference Document: MC68HC705J2/D, Page 10-3****Tracker Number: HC705J2.011****Revision: 2.00**

This applies to mask set 1D86B.

Some low-frequency applications put the part into a pseudo stop mode simply by pulling the OSC1 line low. Current as high as 3 mA has been observed if this pseudo stop mode occurs while an EPROM location is being read.

This pseudo stop mode technique can be used with good results if code is executing out of RAM while pulling OSC1 low. Current values will approach STOP I_{DD} values, but are still not guaranteed to match.

Motorola does not guarantee the STOP I_{DD} values for any cases other than stop mode. Motorola cannot guarantee that there will not be some other source of high current drain when this psuedo stop mode technique is used rather than the recommended and specified stop mode.

Bootloader ROM — Bootloader Programming Clarification

Reference Document: MC68HC705J2/D, Page 8-1

Tracker Number: HC705J2.006

Revision: 1.00

Section 8.1, paragraph two currently reads:

In MC68HC705J2 native mode, the bootloader copies to the 2-Kbyte space located at EPROM addresses \$0700-\$0EFF. In MC68HC05J1 emulation mode, the bootloader copies to the 1-Kbyte space located at EPROM addresses \$0300-\$06FF. The addresses of the copied code must correspond to the internal addresses to which the code is copied. The bootloader ignores all other addresses.

However, Section 8.1, paragraph two should read:

In MC68HC705J2 native mode, the bootloader copies to the 2-Kbyte space located at EPROM addresses \$0700-\$0EFF, the MOR byte at location \$0F00, and the user vector addresses \$0FF0-\$0FFF.

In MC68HC05J1 emulation mode, the bootloader copies to the 1-Kbyte space located at EPROM addresses \$0300-\$06FF, the MOR byte at location \$0700, and the user vector addresses \$07F0-\$07FF. The addresses of the copied code must correspond to the internal addresses to which the code is copied. The bootloader ignores all other addresses.

Bootloader Code

Reference Document: Not applicable

Tracker Number: HC705J2.012 Revision: 1.00

The following boot code can be found in these three mask sets:

- 0D86B
- 1D86B
- 2D86B

```

*****
*
*
*      68HC705J2 EPROM BOOTLOADER PROGRAM      *
*      =====*
*
*      This version:- 5/30/90 - REV 2      *
*      Timing delays based on 2 Mhz bus      *
*      REV 1 was never used      *
*      REV 3 - fixed drain stress, made prog      *
*      time 4 ms. NCN 3/26/91      *
*****
*
* I/O DEFINITIONS
*
PORTA EQU $00 PORT A DATA
PORTB EQU $01 PORT B DATA
DDRA EQU $04 PORT A DDR
DDRB EQU $05 PORT B DDR
*
* EPROM CONTROL REGISTER
*
PROG EQU $1C EPROM CONTROL
*
* MEMORY MAP DEFINITIONS
*
RAM EQU $90 BEGINNING OF RAM

```

```

BOOTST EQU  $0F01  START OF BOOTSTRAP ROM AREA
BOOTV  EQU  $0FE0  START OF BOOTSTRAP VECTOR AREA
VECTOR EQU  $0FF0  START OF USER VECTOR AREA
*
* RAM VARIABLES
*
      ORG  RAM
*
RAMSUB RMB  1      LOCATION OF RAM SUBROUTINE
ADDR  RMB  3      EXTENDED ADDRESS FOR RAM SUBROUTINE
J1J2  RMB  1      J1J2=0 -> J2 MAP, =1 -> J1 MAP
TEMP  RMB  1      TEMPORARY RAM LOCATION
SAVA  RMB  1      TEMPORARY LOCATION FOR ACC. A
SAVE  RMB  1      ANOTHER TEMPORARY LOCATION FOR ACC.
*
* PORT A DEFINITIONS
*
DATAIN EQU  PORTA  ROM DATA INPUT PORT
*
* PORT B DEFINITIONS
*
SYNC  EQU  0      PB0, SYNC INPUT
MODE1 EQU  2      PB2
MODE2 EQU  3      PB3
RST   EQU  4      COUNTER RESET
CLK   EQU  5      COUNTER CLK'
VFYLED EQU  3      BIT 3 DRIVES 'VERIFY' LED
PRGLED EQU  2      BIT 2 DRIVES 'PROGRAMMING' LED

```

page

```

* MISCELLANEOUS DEFINITIONS
*
EPGM  EQU  0      PROG BIT0; - Vpp CONTROL BIT
ERASED EQU  $00   VALUE OF AN ERASED EPROM BYTE
INSTAT EQU  %00111100 INITIAL PORT B LED STATUS
LAT   EQU  2      PROG BIT2; - EPROM ADDRESS LATCH BIT
TEST  EQU  1      PORTB BIT1; - '1' GO BOOT,'0'GO $C0 (RAM)

```


TSTREG EQU \$1F TEST REGISTER

*

*

* INITIAL REGISTER VALUES

- * FCB %00000000 PORT A :- ROM DATA INPUT
- * FCB % 101000 PORT B :- COUNTER IN RESET
- * FCB %00000000 PORT A DDR :- ALL INPUTS
- * FCB % 111100 PORT B DDR :- PB0,1 ARE USED AS INPUT

*

* RAM AREA IS INITIALISED AS FOLLOWS;

*

* LOCATION:- INSTRUCTION:-

*

- * RAMSUB \$90 \$C7 STA EPROM0
- * ADDR \$91 \$00
- * ADDR+1 \$92 \$00
- * \$93 \$81 RTS

*

*

* J1J2 \$94 0 MEMORY MAP

*

*

page

ORG BOOTST

*

TABLE FCB \$C7 'STA EXTENDED' INSTRUCTION

FCB \$00 ADDRESS \$0000

FCB \$00 .

FCB \$81 'RTS' INSTRUCTION

FCB 0 J2 MAP

*

START EQU *

*

* CHECK PORT B, BIT 1 TO SEE IF USER WISHES TO JUMP TO

* RAM OR JUMP INTO THE BOOTLOADER PROGRAM.

*

*

BRSET TEST,PORTB,BOOT

JMP RAM GO TO RAM PROGRAM AT \$0090

*

* SET UP PORTS, RAM SUBROUTINE, AND RAM VARIABLES

*

*

BOOT EQU *

*

* INITIALISE RAM SUBROUTINE AND VARIABLES

*

LDX #\$4

MOVE LDA TABLE,X GET A BYTE FROM THE TABLE

STA RAM,X MOVE IT INTO RAM

DECX POINT TO THE NEXT BYTE TO BE MOVED

BPL MOVE KEEP MOVING UNTIL ALL ARE IN PLACE

*

BRCLR 5,PORTB,J2 IF PB5=0, J2 MAP

INC J1J2

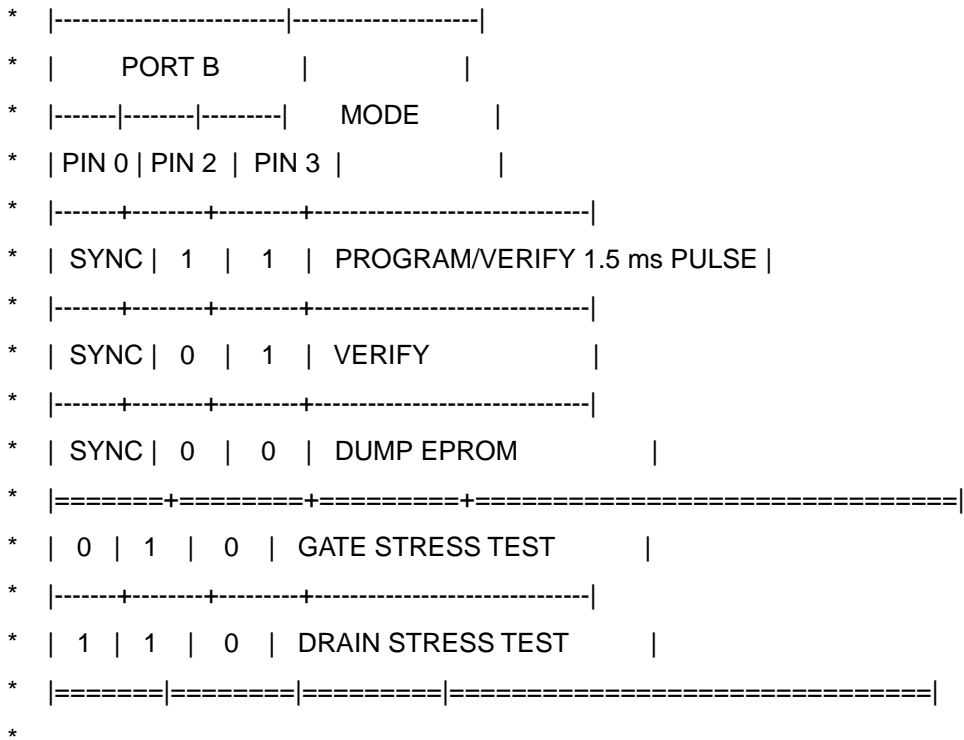
J2 BRSET MODE1,PORTB,MAYPRG

```

BRSET  MODE2,PORTB,VERIFY
*
* DUMP EPROM CONTENTS
*
DUMPE  BSR  INIT
      BSR  ISJ1J2      IS IT A J1 OR J2 MAP
      COM  DDRA        MAKE PORT A ALL OUTPUT
      DEC  RAMSUB      PUT `LDA EXTENDED' IN RAMSUB ($C6)
DLOOP  JSR  RAMSUB    GET DATA STARTING AT $02FF
      STA  PORTA      PUT DATA ON PORT A
      JSR  NXTADR     BUMP ADDRESS
      BNE  DLOOP      EXIT IF END ADDRESS
*
      WAIT
*
MAYPRG BRSET  MODE2,PORTB,PRGVERF
      BRSET  0,PORTB,DTEST
*
*DO THE GATE STRESS TEST
*
GTEST  LDA  #$40  EPTST=1, TS1:TS0=0:0
NOCOM  STA  PROG  ENABLE GATE STRESS TEST
      TXA      WRITE $FF DATA (MAINLY FOR DRAIN STRESS)
      BSR  ZAP
      WAIT
*
* DO THE DRAIN STRESS TEST
*
DTEST  LDA  #$48  EPTST=1, TS1:TS0=0:1
      BRA  NOCOM
page
*****
*
* THE BOOTLOADER PROGRAM HAS 3 MODES OF OPERATION:
*
* I. PROGRAM/VERIFY - PERFORMS 1 NORMAL PROGRAM CYCLES FOLLOWED BY A

```

- * VERIFY CYCLE WHICH HANGS IF THE EPROM IS NOT
- * CORRECTLY PROGRAMED.
- *
- * II. VERIFY - PERFORMS ONLY A VERIFY CYCLE WHICH HANGS IF THE EPROM
- * IS NOT CORRECTLY PROGRAMMED.
- *
- * III. DUMP EPROM - DUMPS THE EPROM CONTENTS OF THE 705J1 TO PORT A
- *
- * WHEN COMING OUT OF RESET INTO THE BOOTLOADER PROGRAM (ASSUMING THAT
- * PORT B PIN 1 ALLOWS YOU TO ENTER THE BOOTLOADER) THE STATE OF
- * PORT B PINS 3 AND 2 DETERMINES WHICH MODE OF OPERATION THE
- * PROGRAM WILL ENTER.
- *
- * THE GATE AND DRAIN STRESS TESTS CAN ALSO BE INVOKED THROUGH THE BOOTLOADER.
- *



*

* INCREMENT COUNTER BY \$700

*

```

INC700 LDA  #7
        BRA  INCX00

```

*

* INCREMENT COUNTER BY \$300

*

```

INC300 LDA  #$3
        BSET 3,ADDR      MAKE INTERNAL ADDRESS $0800
INCX00 STA  SAVE        SAVE ACC.
INC100 CLRA
        BCLR RST,PORTB   REMOVE RESET FROM COUNTER
BUMP JSR  NXTADR
        LDA  SAVA        RECOVER ACCUMULATOR
INCFE DECA
        BNE  BUMP
        DEC  SAVE
        BNE  INC100
        RTS

```

*

* ADVANCE COUNTER

*

```

ADCNT BCLR CLK,PORTB    PULSE COUNTER
        BRSET SYNC,PORTB,*  WAIT FOR SYNC PULSE
LDX  DATAIN          GET DATA
STX  TEMP              SAVE DATA
BSET CLK,PORTB
RTS
page

```

* INITIALIZE PORTB AND ITS DDR

INIT EQU *

```
LDA    #%00111100
INITV STA    DDRB          PB2-5 OUTPUT
INIT1 LDA    #INSTAT      GET INITIAL STATE FOR PORTB
      STA    PORTB
      RTS
```

*

* PROGRAM THE EPROM WITH THE CONTENTS OF THE EXTERNAL ROM

*

```
PRGVERF BSR  INIT
          BCLR  PRGLED,PORTB      LIGHT 'PROGRAMMING' LED
```

*

```
          BSR  ISJ1J2
```

*

```
PRGLOP BSR  PRGSUB          PROGRAM ONE EPROM BYTE
          BSR  NXTADR        POINT TO NEXT ADDRESS
          BNE  PRGLOP        KEEP PROGRAMMING UNTIL DONE
          CLR  ADDR          RESET HIGH ORDER ADDR TO $02
          BSR  INIT1
          BRA  VERIFY1
```

*

* ISJ1J2 DETERMINES IF A J1 OR A J2 IS BEING PROGRAMMED AND
 * BUMPS THE COUNTER ACCORDINGLY. FOR J1, INTERNAL ADDRESS IS
 * \$B00 WHEN COUNTER IS \$300.

```
ISJ1J2 TST  J1J2
          BNE  DOJ1          IF J1J2 = 1, J1 MAP
          BSR  INC700        BUMP COUNTER TO ADDRESS $0700
          RTS
DOJ1 BSR  INC300          BUMP COUNTER BY $0300
          RTS
```

*

* VERIFY THE EPROM CONTENTS AGAINST EXTERNAL MEMORY.
 * (ASSUMES 'RAMSUB' CONTAINS \$C7)

*

```
VERIFY LDA  #%00111000    KEEP 'PROG' PIN AS INPUT
          BSR  INITV
VERIFY1 INC  RAMSUB        CHANGE 'STA' TO 'EOR' EXTENDED ($C8).
```

*

```
          BSR  ISJ1J2
```

*

```
CHECK LDA TEMP GET DATA
      JSR RAMSUB COMPARE TO AN EPROM BYTE
      BNE * HANG IF THEY DON'T MATCH
      BSR NXTADR POINT TO NEXT ADDRESS TO BE COMPARED
      BNE CHECK KEEP CHECKING BYTES UNTIL EPROM END
*
DONE BCLR VFYLED,PORTB INDICATE EPROM VERIFIED AS CORRECT
      WAIT HANG
page
```


RTS RETURN AFTER WANTED DELAY

page

*

* NXTADR SUBROUTINE

*

* COMPUTES NEXT EPROM ADDRESS TO BE PROGRAMMED, VERIFIED, OR DUMPED.

* INCREMENTS THE COUNTER ACCORDINGLY.

* UPDATES RAMSUB ALSO. SKIPS THE RAM, BOOTSTRAP AND UNUSED AREAS.

* RETURNS WITH Z=1 IF THE COMPUTED ADDRESS IS = \$0800, MEANING THAT

* A PASS THROUGH THE MEMORY MAP HAS BEEN COMPLETED. OTHERWISE Z=0.

*

NXTADR STA SAVA

 BSR ADCNT INCREMENT COUNTER AND GET DATA

 INC ADDR+1 INCREMENT LS. ADDRESS BYTE

 BNE CMOR CHECK FOR MOR ADDRESS (\$F00)

 INC ADDR INCREMENT MS. ADDRESS

*

* LOOK OUT FOR HAVING GONE THROUGH THE ENTIRE MEMORY MAP.

*

CMOR LDA ADDR READ MS. ADDRESS

 CMP #\$10 WAS THAT THE END OF MEMORY (\$0FFF)?

 BEQ GOBACK1 EXIT WITH Z=1 IF THE END WAS REACHED.

*

* LOOK OUT FOR HAVING ACCESSED THE LAST LOCATION IN THE MAIN BLOCK

*

 CMP #\$0F WAS THAT THE END OF THE MAIN BLOCK

 BNE GOBACK BRANCH IF STILL WITHIN THE MAIN BLOCK

 LDA ADDR+1 CHECK FOR LS. ADDRESS = \$01

 CMP #1

 BNE GOBACK

*

* SKIP OVER THE BOOTSTRAP AREAS

*

LA LDA #239

LX JSR ADCNT

```

DECA
BNE LX
*
INMAIN LDA  #$F0      FORCE LS. ADDRESS BYTE TO $F0
      STA  ADDR+1
GOBACK LDA  TEMP      GET DATA BYTE
      LDX  #1          CLEAR Z BIT
GOBACK1 RTS
*****
*
      FCB  0,0,0
      ORG  $0FEE
*
RESET FDB  START      RESET VECTOR
*
      END

```