

AN1228

Interfacing the HC05 MCU to the MC145051 A/D Converter

By Mark Glenewinkel
CSIC Applications
Austin, Texas

INTRODUCTION

This application note describes the interface between Motorola's HC05 Family of microcontrollers and Motorola's MC145051(5051) analog to digital converter (ADC). The 5051 is a 10-bit, 11 channel, serial interface ADC. The microcontroller unit (MCU) interface must be able to "talk" to the 5051 using a serial communication link. One of the most popular hardware modules available in the HC05 Family is the serial peripheral interface (SPI). This application note provides the hardware and software design to link the SPI module on the MC68HC705C8 MCU to the 5051.

Not all HC05 Family members have SPI modules. An HC05 MCU without an SPI must interface to the 5051 using a software driver. This method "bit bangs" a port of the MCU to communicate with the 5051. Although not as efficient as the hardware SPI method, it provides MCUs without an SPI a means to retrieve data from the 5051. This application note will utilize the MC68HC705K1 MCU to demonstrate the software driver routine.

MC145051 ADC

Overview

The MC145051 is a ratiometric 10-bit ADC providing 11 analog channels of conversion with an internal sample-and-hold. The 5051 has an internal resistor capacitor (RC) clock oscillator to run its internal digital circuitry. The maximum conversion time for the 5051 is 44 μ s with a maximum sample rate of 21.4 ksamples/s. If faster conversion time is needed, a 5050 can be used that is the same as the 5051 except it requires an external clock. With a 2.1 MHz clock, the 5050 provides a 21 μ s conversion time and a maximum sample rate of 38 ksamples/s. The 5051 operates with a single voltage supply between 4.5 and 5.5 volts. A serial interface is used to receive the channel address to convert and transmit converted values to the outside world.



Successive Approximation

The 5051 utilizes successive approximation to convert the analog input signal to a digital value. This technique consists of comparing the unknown analog input to a known analog voltage created by a digital to analog converter (DAC). The digital number given to the DAC is the number that will eventually be the result of the ADC's output. This process of "guessing" the analog input voltage is similar to weighing with a balance. If you had 3 weights consisting of 1/2, 1/4, and 1/8 of a gram, you could measure something up to 1 gram within +/-1/16 gram of the weight. One side of the scale would hold the unknown and the other side would contain various weights "guessing" at the weighted value of the unknown.

Consider how a 3-bit A/D converter would convert an unknown signal. Figure 1 shows the block diagram of a very simple 3-bit A/D converter. A digital number is fed into the DAC and the DAC converts this to an analog voltage for the comparator to use. If the input analog voltage is larger than the DAC's output voltage, a "1" is the result of the comparison. If the input analog voltage is smaller than the DAC's output, a "0" is the result of the comparison. The result of the comparison is fed back into the successive approximation register. The control logic adds a smaller digitally "weighted" value to the DAC to "guess" at the input analog voltage. This sequence continues until the smallest digital "weight" is used to guess at the input voltage. Figure 2 illustrates this process with a graph that depicts the testing of a signal of magnitude between 3/8 and 4/8 of the full scale analog reference voltage. After the guesswork is done, the binary answer of 011 is written to an output register for further processing.

In this example, the input voltage does not change over the entire conversion process. We have assumed that the signal does not change and there is no noise to change the input voltage. In most cases a sample-and-hold circuit is used to sample a voltage signal and hold it for a specific length of time until the conversion process is complete.

The accuracy, linearity, and speed of the successive approximating A/D converter are dependent on the properties of the DAC and the comparator. The settling time of the DAC and the speed of the comparator determine the speed of the conversion process. Likewise, if the conversion demands more resolution, the time to convert will be lengthened. The DAC's nonlinearity will result in nonlinearities within the ADC. All of these factors affect the digital output result of the ADC.

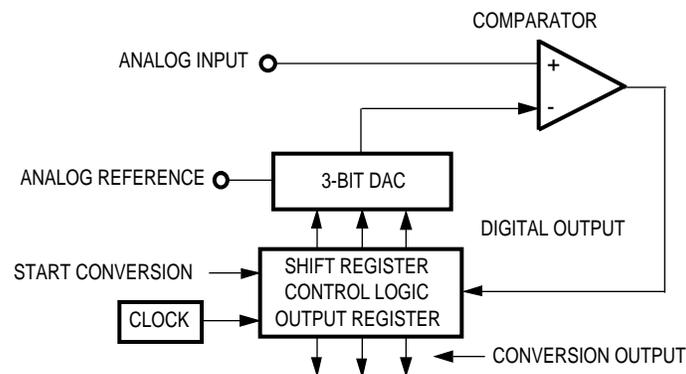


Figure 1. Simple 3-Bit A/D Converter

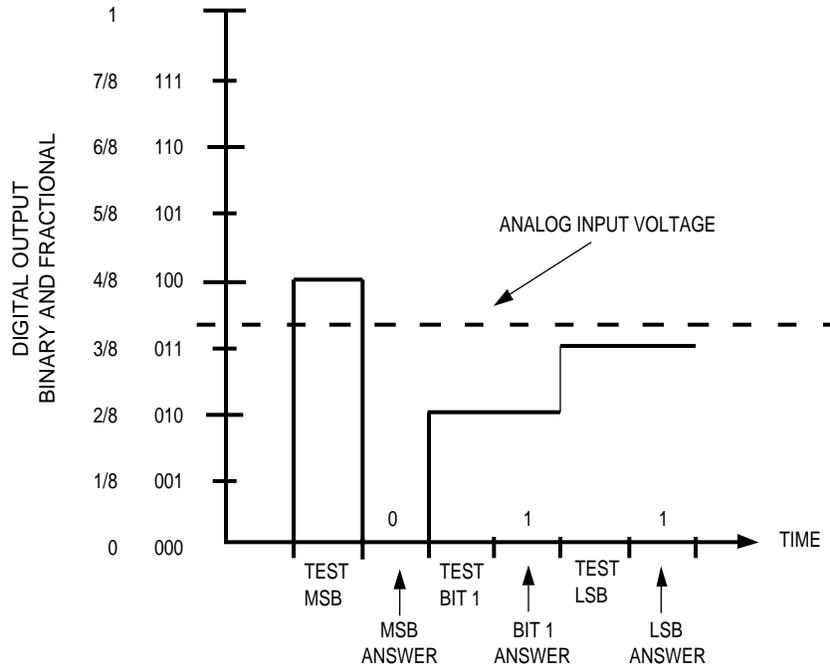


Figure 2. 3-Bit A/D Weighing Sequence

Inside the 5051

As stated earlier, the 5051 will convert one of the 11 analog inputs into a 10-bit digital representation of the analog signal. The 10-bit digital value is transmitted to the outside world via a serial bus. Figure 3 shows a block diagram of the 5051. The sequence of starting a conversion, converting the voltage, and transmitting the result is listed below:

1. The \overline{CS} signal is driven low to initialize the serial port that a 4-bit address is going to be received and the previous 10-bit digital result will be transmitted.
2. After the 4-bit mux address is received in the Mux Address Register, one of the analog inputs is selected from the Analog Multiplexer. This signal is sent to the Sample-and-Hold to start the 10-bit conversion process.
3. While the 4-bit address is received, the 10-bit previously converted value is sent out on the D_{OUT} pin.
4. The internal clock drives the Digital Control circuitry, which in turn manipulates the Successive Approximation Register until the 10-bit conversion is complete.
5. Once the conversion is complete, the final value of the Successive Approximation Register is written to the Data Register. The 10-bit result will stay here until it is queued to be sent out on the D_{OUT} pin on the next serial transmission. Also, the 5051 will signal the ending of a conversion by driving the end-of-conversion (EOC) pin high. In some transmission scenarios, the \overline{CS} pin must be negated high before another transmission and conversion can occur.

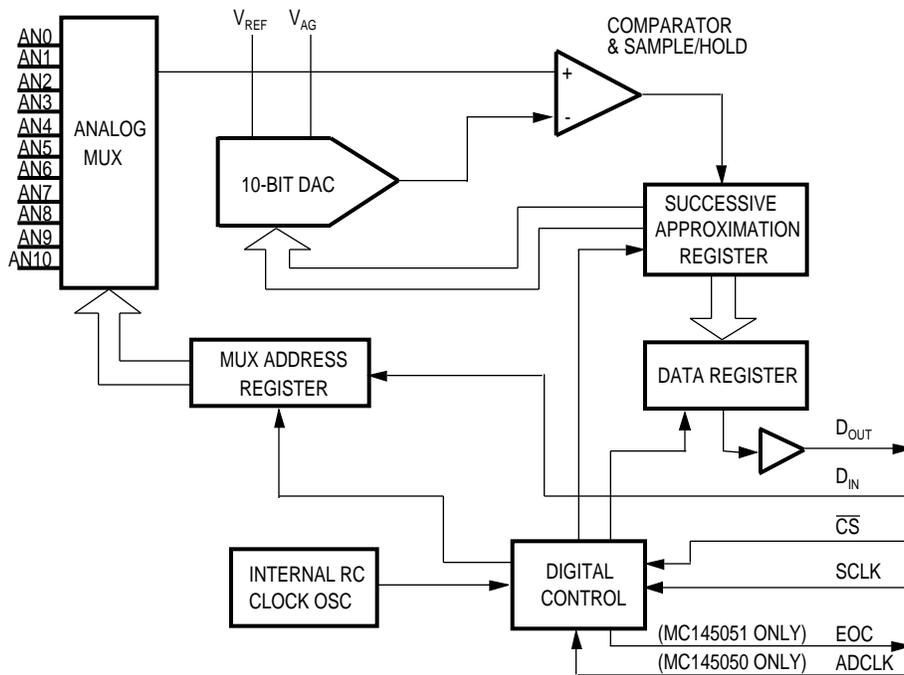


Figure 3. MC145051 Block Diagram

Analog Interface

The analog input consists of the converter's high and low voltage reference pins and all 11 analog input pins. The analog specs are listed in the following table:

Table 1. Analog Specifications

Symbol	Parameter	Min	Max
V_{REF}	DC reference voltage	$V_{AG} + 4.0$	$V_{DD} + 0.1$
V_{AG}	Analog ground	$V_{SS} - 0.1$	$V_{REF} - 4.0$
V_{AI}	Analog input voltage	V_{AG}	V_{REF}

The 5051 will take the voltage it samples off its analog input pin and convert it to a number equivalent to the ratio of the input voltage and the difference between the V_{REF} and V_{AG} . This number is the converter's digital representation of the sampled voltage input. Figure 4 illustrates this ratio and describes an equation that predicts the ADC's conversion value. For example, if $V_{AI} = 2.34$ volts, then the 10-bit representation of that voltage is 479 or \$1DF.

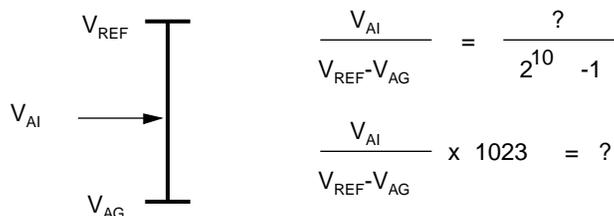


Figure 4. A/D Conversion Ratio

Digital Interface

The digital interface to the 5051 is composed of a serial data port that synchronously transceives data. Each digital pin's function is explained below.

$\overline{\text{CS}}$ Active-Low Chip Select

When asserted low, this pin initializes the chip to start performing A/D conversions. While high, the D_{OUT} pin is forced to a high-impedance state and the D_{IN} pin is disabled.

D_{OUT} Serial Data Out

This pin serves as the serial output data of the A/D conversion result. After $\overline{\text{CS}}$ is asserted low, D_{OUT} is driven with the most significant bit of the previous 10-bit A/D result. The value of D_{OUT} changes to the second most significant bit after the falling edge of the serial clock (SCLK). After 10 bits of transmission, D_{OUT} is driven low. The A/D result is always driven out of D_{OUT} most significant bit (MSB) first.

D_{IN} Serial Data In

This pin serves as the input data line that receives the 4-bit address of the serial stream. The address is shifted on the rising edge of SCLK with the MSB being the first bit received. After all four bits have been received, the D_{IN} pin is ignored.

SCLK Serial Data Clock

This pin is an input that drives the serial transmission lines. It drives the data shift registers so that the next mux address is received and the previous conversion is driven out.

EOC End-of-Conversion Output

This pin is driven low on the 10th falling edge of SCLK. A low-to-high transition on EOC occurs after the A/D conversion is complete.

The 5051 is capable of various bit stream formats. The timing diagram used in this application note is shown in Figure 5. The 5051 will wait patiently until its $\overline{\text{CS}}$ pin is asserted low. This signifies that a serial clock will be driving the SCLK pin to transfer the next A/D channel address to be converted. At the same time, the 5051 will be driving out the converted value of the previous conversion. After the 10-bit address is driven out of D_{OUT} , the $\overline{\text{CS}}$ pin will be driven high to signify the end of the transmission process.

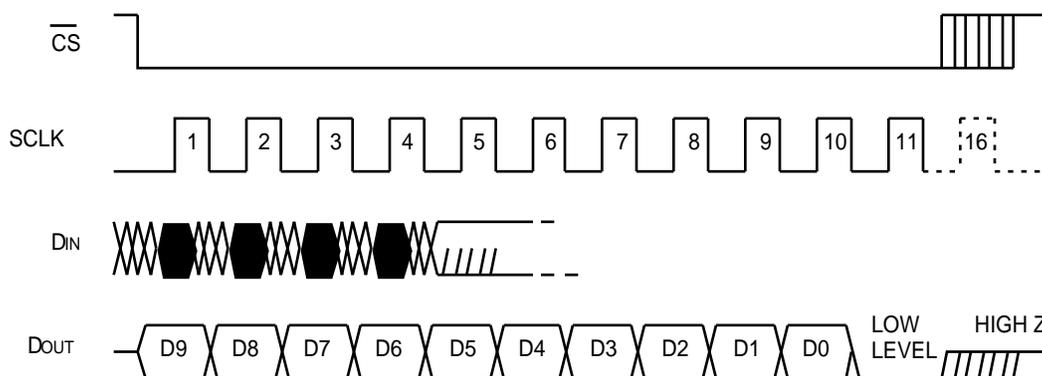


Figure 5. MC145051 Timing Diagram

DESCRIPTION OF THE HC705C8 INTERFACE

Hardware

The MC68HC705C8 is one of the most popular members of the HC05 Family of 8-bit MCUs. It has the serial peripheral interface (SPI) that will be used to interface to the 5051. The SPI is, in essence, an 8-bit serial shift register that can be manipulated by software instructions. The SPI can be programmed with different clock polarities and clock phases to correctly communicate with a number of devices. The SPI can also be configured to act as a master or a slave. Each signal of the SPI is explained below. For more detail on the SPI, consult *MC68HC705C8 Technical Data*, Rev. 1 (MC68HC705C8/D).

SCK Serial Data Clock

The SCK signal is used to synchronize the movement of data in and out of the SPI module. This pin is an output or an input dependent on whether the SPI is configured as a master or a slave. Data is shifted on one side of the clock edge and sampled on the other. The SCK signal can be configured to accommodate different serial peripheral bus structures.

MOSI Master Output, Slave Input

When the SPI is configured as a master, this pin is used as an output to shift the 8-bit serial data out with the most significant bit first. The pin is used as a slave data input when the SPI is configured as a slave.

MISO Master Input, Slave Output

If the SPI is configured as a master, this pin is utilized as an input. When the SPI is in slave mode, the pin is used as an output.

\overline{SS} Slave Select

When the SPI is a slave, this pin enables the SPI for an incoming transfer. As a master, this pin should be tied high.

To correctly interface to the 5051, the SPI is configured as a master with the timing diagram shown in Figure 6. This configuration enables the SCK to drive out data with the MOSI pin on the rising edge and receive data with the MISO pin on the falling edge.

The schematic used for this interface is shown in Appendix A. The HC705C8 is clocked by a 4 MHz crystal circuit. This provides the MCU with a 2 MHz internal bus frequency and a 500 ns bus period or instruction cycle. The MC34064 is used as a low voltage inhibitor circuit. This 3-pin, T0-92 device ensures that the reset pin is pulled low if the operating voltage to the MCU falls below 4.6 volts.

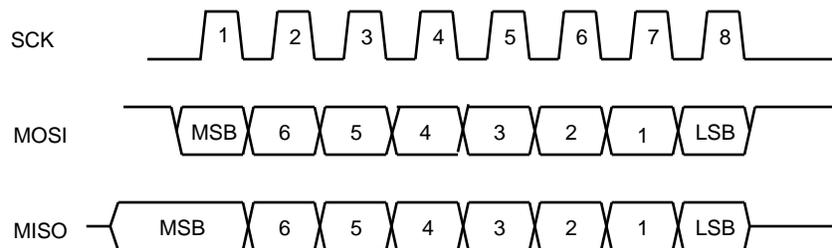


Figure 6. SPI Timing Diagram

The SPI lines are connected to the appropriate pins on the 5051. The MOSI pin drives data out of the HC705C8 and into the D_{IN} pin of the 5051. The D_{OUT} pin drives data out of the 5051 pin into the MISO pin of the HC705C8. Since the SPI is configured as a master, the SCK pin is driving the SCLK pin of the 5051 and the \overline{SS} pin is tied high.

The HC705C8 is programmed to utilize the SPI to read the 5051. Channel AN0 of the 5051 is used to read the voltage created by the 10 k potentiometer between the V_{REF} and the V_{AG} levels. A 0.22 μ F capacitor is used between the V_{REF} and V_{AG} pins to filter out high frequency noise. This capacitor should be mounted as close to the 5051 as possible. After the HC705C8 receives the data from the 5051, it is driven out onto Ports B and C of the HC705C8.

The circuit given in Appendix A minimizes the noise often found in emulated systems. Instead of programming the HC705C8, the M68HC05EVM can be used to emulate the HC705C8. This evaluation module will not give as accurate an A/D reading as the circuit in Appendix A, but allows more flexibility in code development than using a programmed HC705C8.

Software

The flowchart for the SPI-driven 5051 is shown in Appendix B, and the actual HC05 assembly code is given in Appendix C. This code was written for a programmed HC705C8. Extra lines of code were added so that the routine would perform in a standalone application.

For the SPI to "talk" to the 5051, the SPI must be configured to match up with the 5051 timing diagram, as shown earlier in Figure 5. Also, two SPI transmissions must be sent to form a 16-bit transfer. Before any transmissions can start, the \overline{CS} pin must be asserted low. This initializes the 5051 and tells it that a new mux address will be sent to it to start the conversion process. The first transfer sends the A/D channel number to the 5051, and the 5051 sends the upper 8 bits of the previously converted value. These 8 bits are written to the MSB of the 16-bit result register and to Port B. The second transfer sends the A/D channel to the 5051 but the 5051 ignores it because it is not needed. The 5051 sends the HC705C8 the 2 least significant bits from the previously converted value. These 2 bits are the 2 most significant bits in the received SPI data. This byte is written to the LSB of the 16-bit result register and to Port C. After both transmissions are done, \overline{CS} is negated high. Port B and Port C now have the 10-bit A/D value of the previous conversion. This output value on Port B and Port C is illustrated in Figure 7. The routine will now sit in an infinite loop waiting for a reset.

The following example is provided to test the software routine. Follow these steps after programming the HC705C8 with the code in Appendix C and constructing the schematic in Appendix A.

1. Set the potentiometer to a reading of 2.20 volts.
2. If $V_{REF}-V_{AG}$ is exactly 5.00 volts, the A/D should convert to a reading of 450 or \$1C2. (See **Figure 4 A/D Conversion Ratio.**)
3. Power on the circuit.
4. The A/D value will be outputted on Port B and Port C. This value is the previously converted value. Since there was no previous conversion, the data will be garbage.
5. Pull the \overline{RESET} pin low and then high. The routine will run again, and the previous value of the AN0 channel conversion is represented on Port B and Port C. The value for Port B should be \$70 and Port C should be \$80. The result might differ by a least significant bit (LSB). (See Figure 7.)

This routine is the simplest example to test and learn the interface from the HC705C8 to the 5051. Notice that the mux address must be in the high nibble of the byte before it is written to the SPI data register. Also, since this routine was hard-coded, the A/D channel was already known and written into memory. The code

can be easily adapted as a subroutine, which requires that the channel to be converted is an input to the subroutine. If the application requires that successive A/D conversions are made, make sure that the 5051 has enough time to convert the present channel before initializing another conversion. If needed, the 5051 provides the EOC pin. During a conversion process, the pin is held low. After conversion is complete, the pin is driven high. Another port pin on the HC705C8 might be used to read the EOC pin.

10-BIT RESULT = \$1C2 = 01,1100,0010%
 PORT B = \$70 = 0111,0000%
 PORT C = \$80 = 1000,0000%

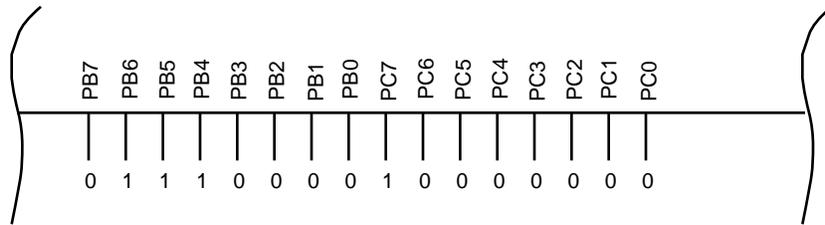


Figure 7. A/D Value on Port B and Port C

DESCRIPTION OF THE HC705K1 INTERFACE

Hardware

With only 16 pins, the HC705K1 is one of the smallest members of the HC05 Family. It has a total of 504 bytes of erasable programmable read only memory (EPROM) and includes 10 I/O pins. The schematic for the HC705K1 to 5051 interface is shown in Appendix D. With this interface, the HC705KICS development board was used to write and test the code. The circuitry surrounding the 5051 is the same as in the HC705C8 design. The only changes are the serial pins of the 5051. These pins are connected to the emulation header of the HC705KICS board. This emulation header has the exact pin-out of the HC705K1. The pins used to drive the 5051 on the HC705K1 are as follows:

- Port A, Bit 0 —This I/O pin (\overline{CS}) is configured as an output to drive the \overline{CS} pin on the 5051.
- Port A, Bit 1 —This I/O pin (SER_CLK) is configured as an output to drive the serial clock of the serial transmission bus.
- Port A, Bit 2 —This I/O pin (SER_OUT) is configured as an output to drive the serial data out and into the D_{IN} pin of the 5051.
- Port A, Bit 3 —This I/O pin (SER_IN) is configured as an input to receive data driven from pin D_{OUT} of the 5051.

The emulation test circuit may also be configured as a standalone design. For further information on programming the HC705K1, consult the *MC68HC705K1 Technical Data*, Rev. 1 (MC68HC705K1/D) and the HC705KICS development board documentation.

Software

The flowchart for the bit-banged-driven 5051 is shown in Appendix E, and the actual HC05 assembly code is given in Appendix F. Bit-banging is the process of toggling I/O pins with software instructions to emulate a certain piece of hardware peripheral. This bit-banged routine was written especially for the 5051. It is not a full featured representation of the HC705C8 SPI module. Enhancements to the routine were not included in order to maximize the efficiency of the code.

As stated in the preceding hardware section, I/O pins have been used to send out the correct serial transmission protocol to the 5051. The HC05 CPU provides special instructions to specifically manipulate single I/O pins. The 5051 serial stream shown in Figure 5 will be re-created by four I/O pins on the HC705K1.

The best way to describe the code is to list each segment of the code and explain its purpose to bit-bang the 5051. PA2 is shorthand for Port A, bit 2.

Equivalents

PA0 = $\overline{\text{CS}}$

PA1 = SER_CLK

PA2 = SER_OUT

PA3 = SER_IN

Initialize Port A

$\overline{\text{CS}} = 1 \Rightarrow$ output

SER_CLK = 0 \Rightarrow output

SER_OUT = 0 \Rightarrow output

SER_IN = 0 \Rightarrow input

Begin A/D Acquisition

The $\overline{\text{CS}}$ pin is driven low to start the serial transmission.

The CHANNEL ram byte is read. The address is in the low nibble of the byte. The 16-bit RESULT registers are cleared and a copy of CHANNEL is stored in TMP_CHN for future use. When emulating, make sure that location \$E2 is initialized with the A/D channel \$00.

Initialize Loop1

Set the index register to 4.

Read the serial input pin — Start of Loop 1

The branch-if-clear instruction is used to read SER_IN. The purpose of this is to transfer the logic state on the SER_IN pin to the carry bit (C). No branch is taken. The next line of code is always executed.

Two rotate left instructions rotate the C bit into the 16-bit RESULT register composed of RESULT and RESULT+1. The first bit read on SER_IN is the MSB of the previous A/D result from the 5051.

Write the serial output pin

The TMP_CHN is rotated left. Bit four of TMP_CHN is read. If it is high, a "1" is written to SER_OUT. If it is low, a "0" is written to SER_OUT. This first transmitted bit is the MSB of the 4-bit A/D channel address.

Clock the serial clock pin

The SER_CLK pin is written high and then written low.

Is Loop 1 done?

The index register is decremented and checked to see if it is 0. If IX is not 0, the code is executed at the start of Loop 1. This loop continues until four transmissions are completed.

Initialize Loop 2

Set the index register to 6.

Read the serial input pin — Start of Loop 2

This is the same code that was executed at the start of Loop 1 above. Notice that Loop 2 does not transmit any more bits on SER_OUT. This is because the 5051 ignores the last 6 transmitted bits because it has already received the 4 address bits it needs.

Clock the serial clock pin

The SER_CLK pin is written high and then written low.

Is Loop 2 done?

The index register is decremented and checked to see if it is 0. If IX is not 0, the code is executed at the start of Loop 2. This loop continues until six transmissions are completed.

Negate \overline{CS}

A "1" is written to the \overline{CS} . This completes the serial transmission to the 5051.

Since this code was written for emulation on the M68HC705KICS board, it is easy to experiment with different applications. The code can be easily adapted to fit any custom application that needs 10-bit A/D data.

LAYOUT CONSIDERATIONS

There are many things to consider when laying out mixed signal designs such as the 5051 and the HC05 MCU. The accuracy of the 5051 may be greatly affected if proper layout design is not followed. Listed below are some things to check to ensure the accuracy of your A/D converter. For more in depth study of ADC layout issues, please consult *Reducing A/D Errors in Microcontroller Applications (AN1058/D)*.

- Physically separate critical analog circuits from the digital circuits of the MCU. If possible, split your board in half to separate analog and digital circuits. Each half will have its own power and ground system.
- Do not let analog input line traces cross digital traces. If this has to happen, make sure they cross at right angles to each other.
- Use power or ground traces to isolate the analog-input pins from the digital pins.
- Bypass the power supplies to the proper ground at the 5051 power pins with quality ceramic capacitors. Keep the bypass capacitors lead lengths as short as possible.
- To bypass low frequency power supply noise, use tantalum or aluminum electrolytic capacitors of 5 to 20 μ F. These should be placed near the point the power supplies enter the board.

REFERENCES / FURTHER READING

Analog-Digital Conversion Handbook, Third Edition, New York: Prentice-Hall, 1986.

MC145050/51 Technical Data Sheet, (MC145050/D), Motorola, 1993.

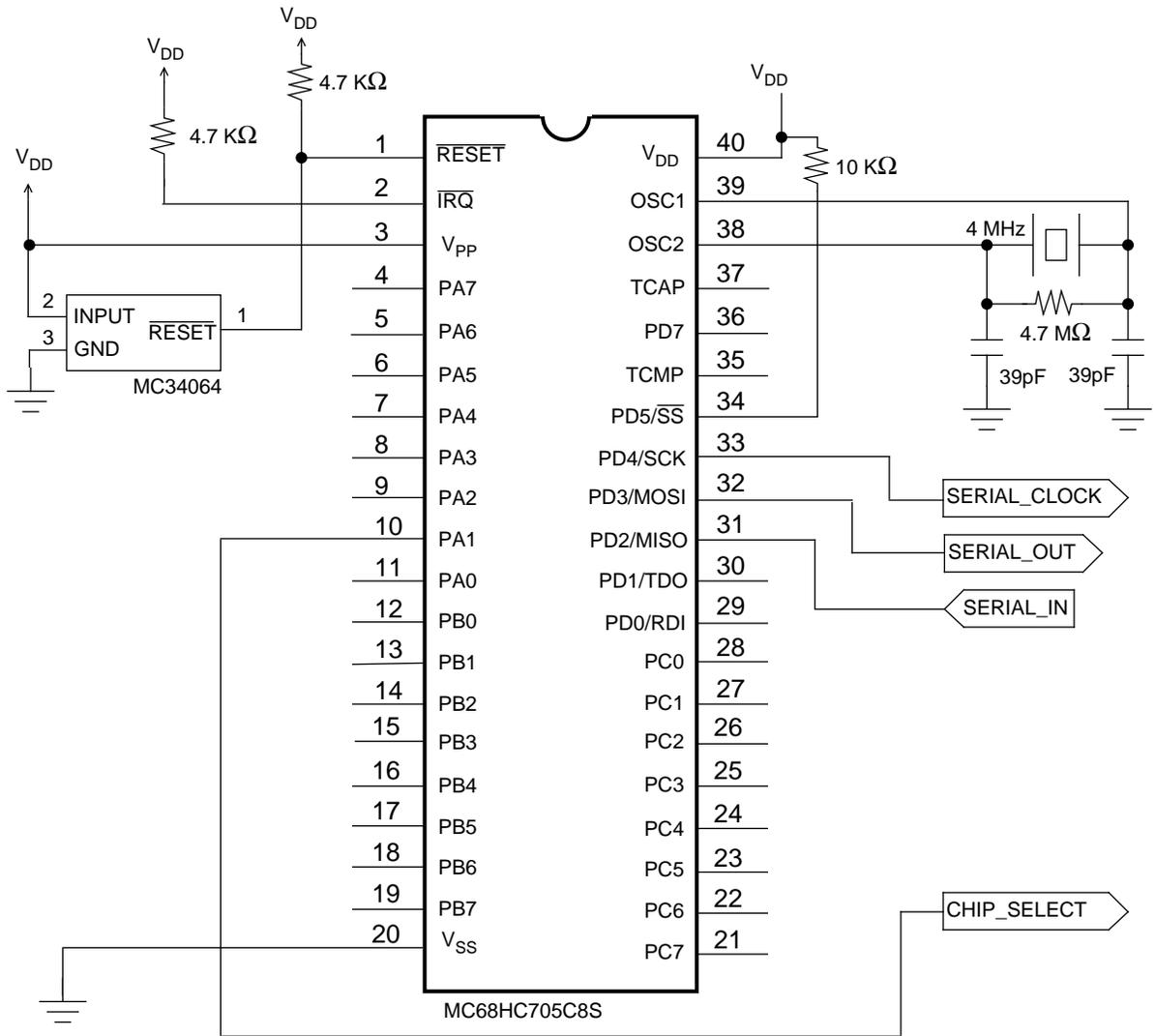
MC68HC05 Applications Guide, (M68HC05AG/AD), Motorola, 1989.

MC68HC705C8 Technical Data, (MC68HC705C8/D), Motorola, 1990.

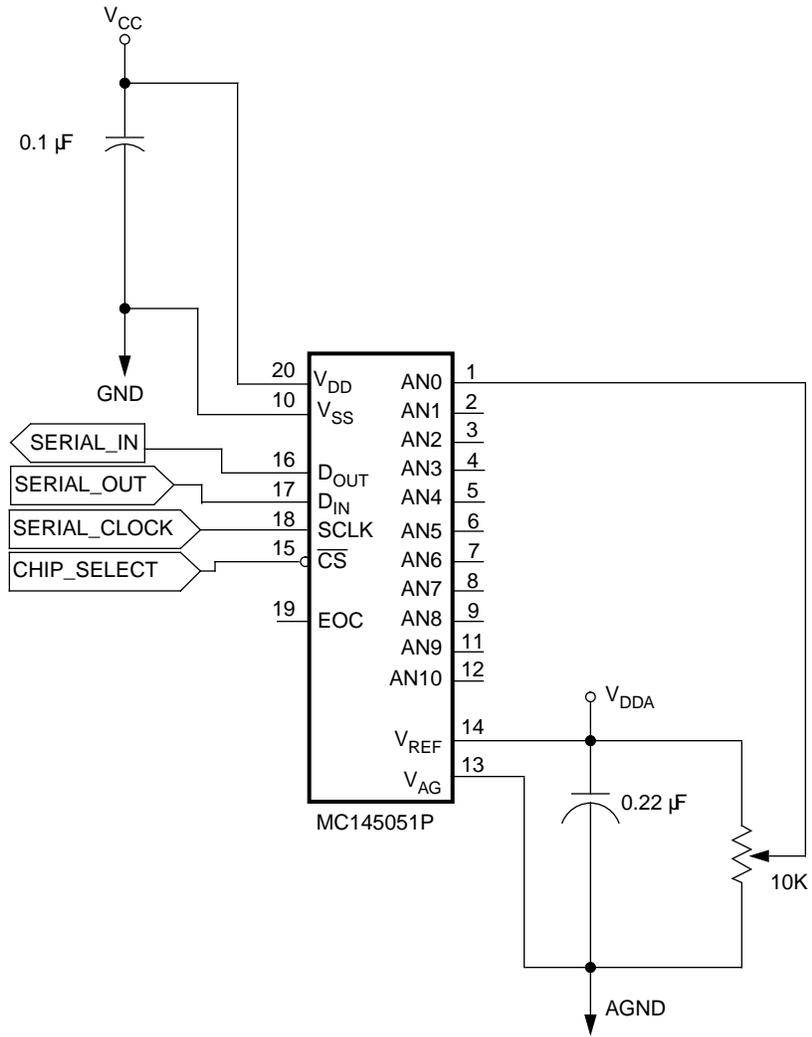
MC68HC705K1 Technical Data, (MC68HC705K1/D), Motorola, 1993.

Reducing A/D Errors in Microcontroller Applications, (AN1058/D), Motorola, 1990.

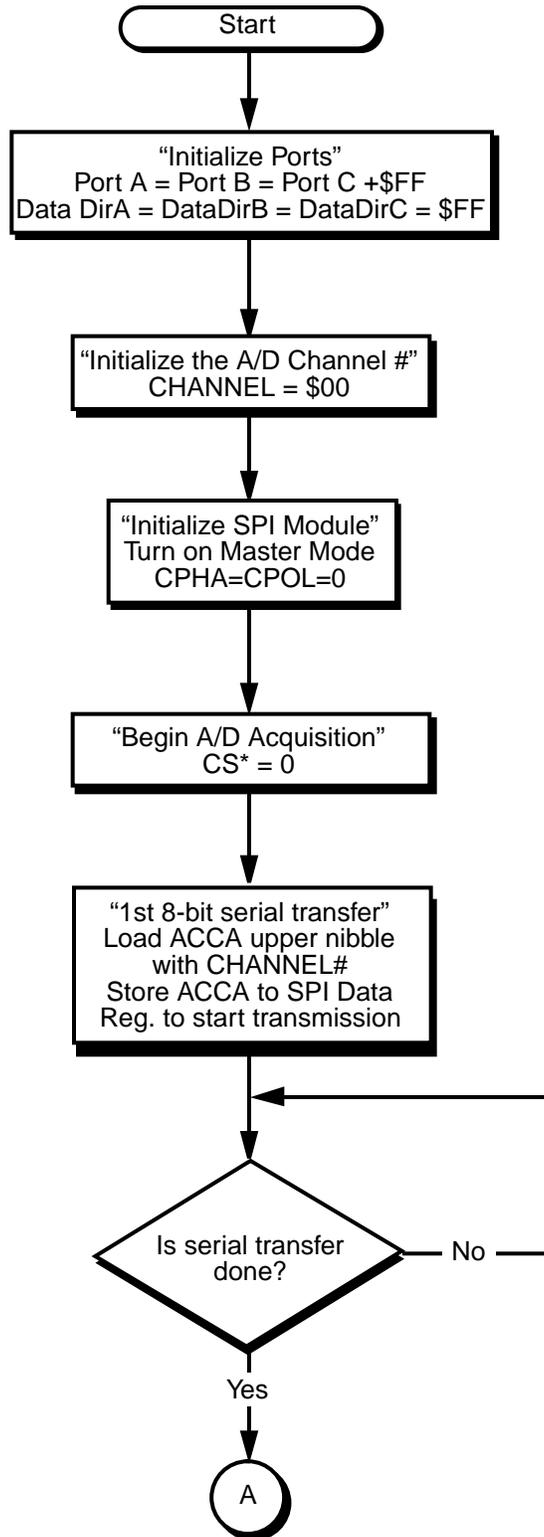
Appendix A HC705C8/5051 Schematic

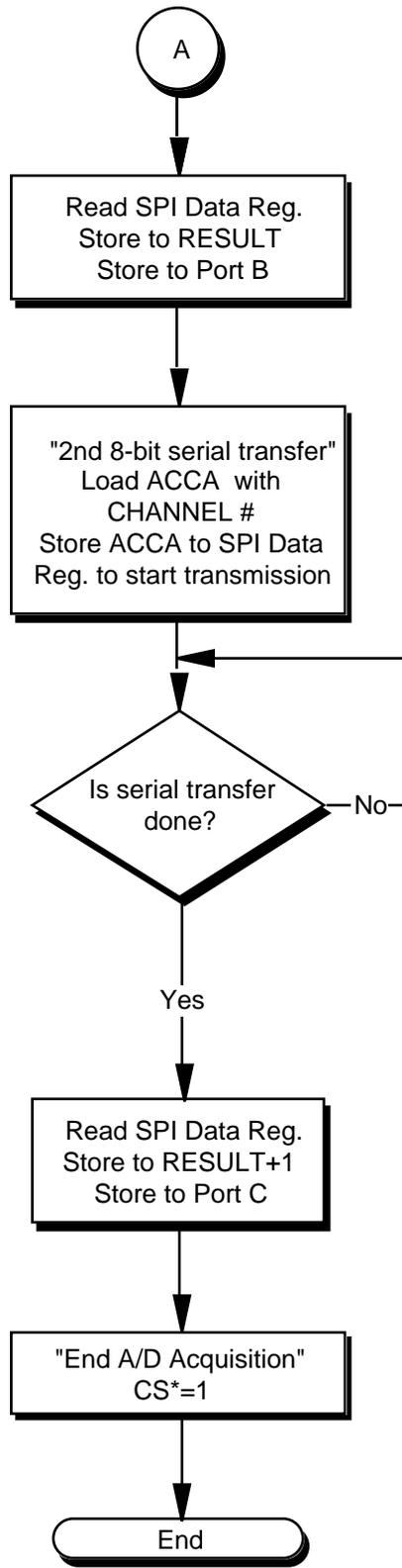


Appendix A HC705C8/5051 Schematic (continued)



Appendix B HC705C8/5051 Flowchart





Appendix C HC705C8/5051 Assembly Code

```
*****
*
* Program Name: C8_5051.ASM ( 705C8 to 145051 interface )
* Revision: 1.00
* Date: October 7, 1993
*
* Written By: Mark Glenewinkel
*             Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
*             *****
*             *           Revision History           *
*             *****
*
*             Rev      1.00      10/07/93           M.R. Glenewinkel
*             Initial Release
*
*****
*
* Program Description:
*
*       This software routine provides a way for MCUs with an
*       SPI module on chip to interface to the Motorola MC145051
*       10 bit, 11 channel analog to digital converter.
*
*       This program specifically uses the MC68HC705C8 MCU
*       to test the code. The HC705C8 "talks" to the 5051 with
*       the appropriate serial data transfer from its SPI module.
*
*       For more information, please consult Motorola
*       Application Note AN1228/D.
*
*****

***      Equates for 705C8      ***

PORTA    equ      $00           ;port A data reg
DDRA     equ      $04           ;data dir reg A
PORTB    equ      $01           ;port B data reg
DDRB     equ      $05           ;data dir reg B
PORTC    equ      $02           ;port C data reg
DDRC     equ      $06           ;data dir reg C
SPCR     equ      $0A           ;spi ctrl reg
SPSR     equ      $0B           ;spi status reg
SPDR     equ      $0C           ;spi data reg

CS       equ      1             ;bit # for chip select

***      RAM storage variables      ***

        org      $50           ;start of static RAM
RESULT   rmb      2             ;2 bytes needed for 10 bit result
```

```

CHANNEL rmb      1                ;a/d channel #

***      Start of program          ***

      org      $1000                ;start of program

START   lda      #$FF
      sta      PORTA                ;port A = $FF
      sta      DDRA                 ;port A all outputs
      sta      PORTB                ;port B = $FF
      sta      DDRB                 ;port B all outputs
      sta      PORTC                ;port C = $FF
      sta      DDRC                 ;port C all outputs

      lda      #$00                ;CHANNEL = AN0
      sta      CHANNEL

*       Initialize SPI module      *

      lda      #$50                ;turn on spi, mstr mode
      sta      SPCR                 ;cpha=cpol=0

*       Send out 16 bit frame      *

      bclr     CS,PORTA             ;CS* is low
      lda      CHANNEL              ;load ACCA with CHANNEL

*       Send out address, receive most significant byte
WAIT1   brclr   7,SPSR,WAIT1        ;wait until SPIF flag is set
      lda      SPDR                 ;load ACCA with MSB of ADC result
      sta      RESULT              ;store this to MSB of RESULT
      sta      PORTB                ;store the 8 MSBs to Port B

*       Start another SPI transmission to receive the
*       2 least significant bits

WAIT2   brclr   7,SPSR,WAIT2        ;wait until SPIF flag is set
      lda      SPDR                 ;load ACCA with LSB of ADC result
      sta      RESULT+1            ;store this to LSB of RESULT
      sta      PORTC                ;store the 2 LSBs to Port C

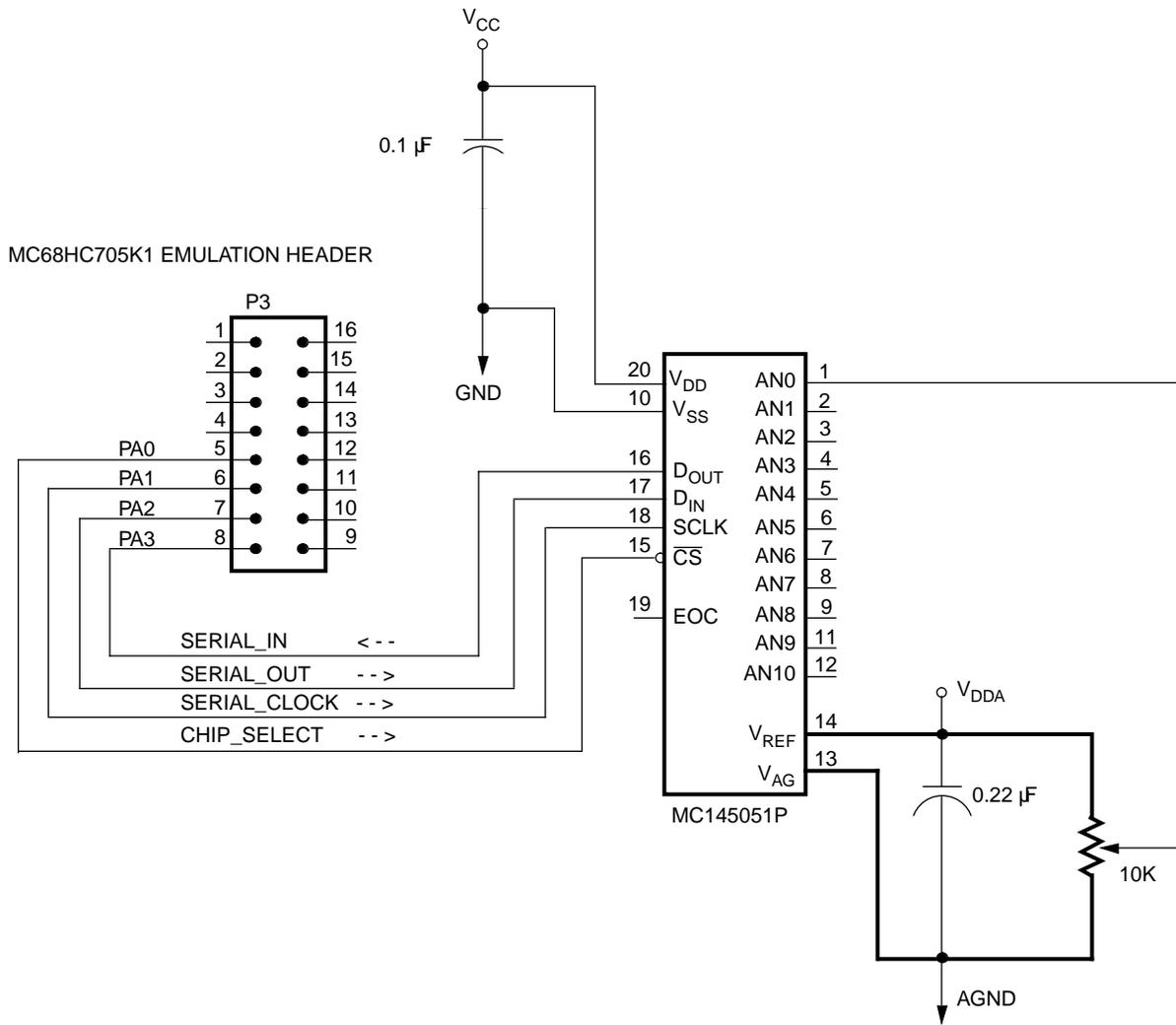
      bset     CS,PORTA             ;CS* is high, end 16 bit frame

*       Wait for ever until reset
FOR     bra     FOR                 ;branch to itself

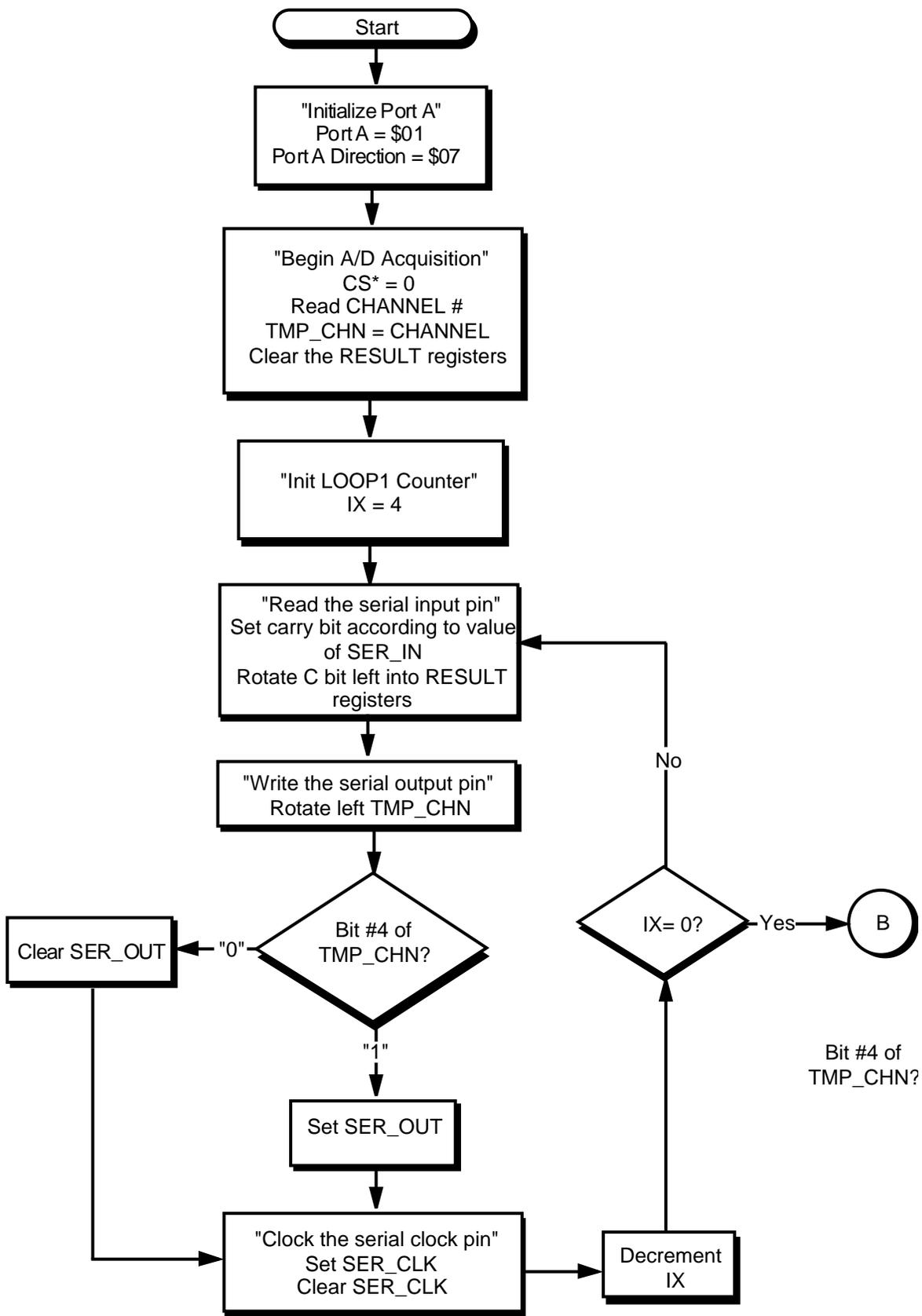
      org      $1FFE                ;define reset vector
      dw      START

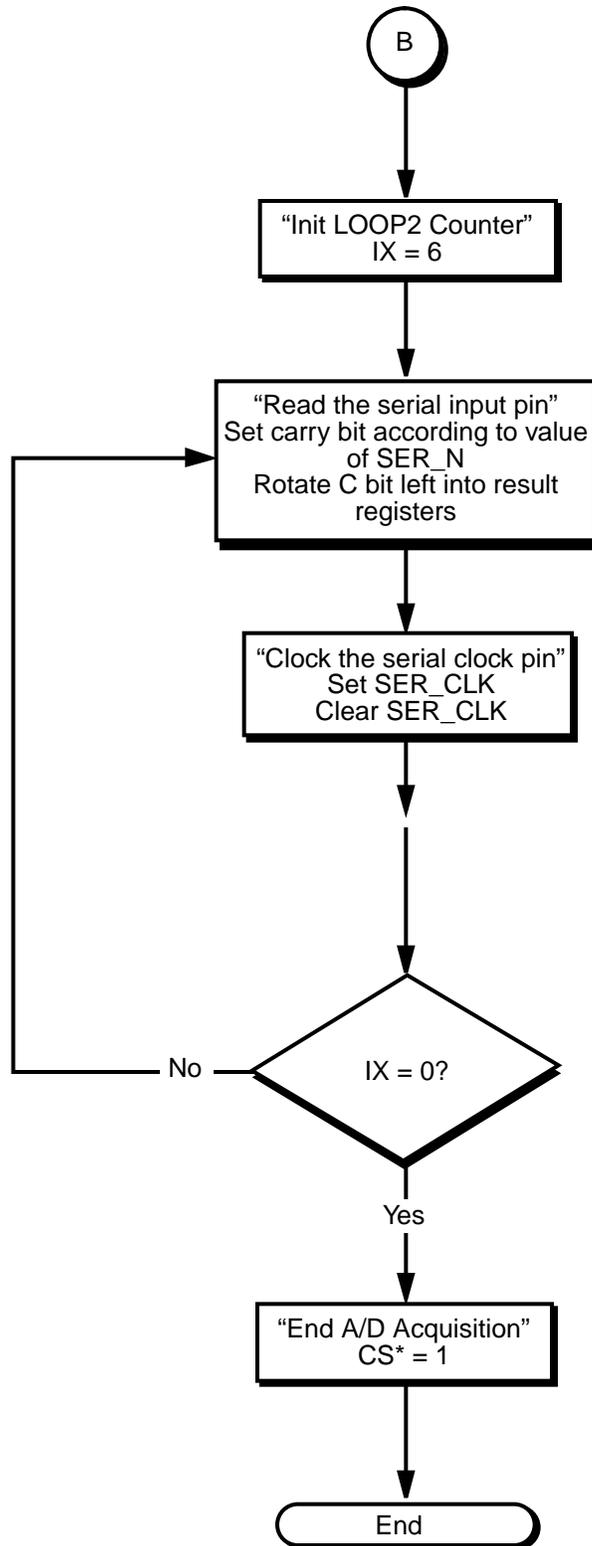
```

Appendix D HC705K1/5051 Schematic



Appendix E HC705K1/5051 Flowchart





Appendix F HC705K1/5051 Assembly Code

```
*****
*
* Program Name: K1_5051.ASM ( 705K1 to 145051 interface )
* Revision: 1.00
* Date: September 22, 1993
*
* Written By: Mark Glenewinkel
*           Motorola CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM05
*
*           *****
*           *           Revision History           *
*           *****
*
*           Rev      1.00      09/22/93           M.R. Glenewinkel
*           Initial Release
*
*****
*
* Program Description:
*
*           This software routine provides a way for MCUs with no
*           SPI module on chip to interface to the Motorola MC145051
*           10 bit, 11 channel analog to digital converter.
*
*           This program specifically uses the MC68HC705K1 MCU to
*           test the code. The HC705K1 "bit bangs" the 5051 with the
*           appropriate serial data transfer the 5051 understands.
*
*           For more information, please consult Motorola
*           Application Note AN1228/D.
*
*****

***      Equates for 705K1      ***

PORTA  equ    $00           ;port A data reg
PORTB  equ    $01           ;port B data reg
DDRA   equ    $04           ;data dir reg A
DDRB   equ    $05           ;data dir reg B

CS      equ    0           ;bit # for chip select
SER_CLK equ    1           ;bit # for serial clock
SER_OUT equ    2           ;bit # for serial data out
SER_IN  equ    3           ;bit # for serial data in
```

```

***      RAM storage variables      ***

        org      $E0                ;start of static RAM
RESULT  rmb      2                  ;2 bytes needed for 10 bit result
CHANNEL rmb      1                  ;a/d channel #
TMP_CHN rmb      1                  ;temp a/d channel for routine

***      Start of program          ***

        org      $200               ;start of user eprom

***      Initialization            *
START   lda      #$01               ;init port A
        sta      PORTA
        lda      #$07               ;init i/o of port A
        sta      DDRA

***      Init CS to low to start a/d
        bclr     CS,PORTA           ;CS* line is low

        lda      CHANNEL           ;load ACCA with a/d channel #
                                           ;(for emulation, init $E2=$00)
        sta      TMP_CHN           ;store ACCA to temp channel
        clr      RESULT+1          ;clear result regs
        clr      RESULT
        ldx      #4T               ;init counter for LOOP1

***      Read the serial input pin
LOOP1   brclr   SER_IN,PORTA,L1_1    ;carry bit = serial in
L1_1    rol     RESULT+1           ;rotate left result
        rol     RESULT

***      Write the serial output pin
        rol     TMP_CHN           ;rotate left TMP_CHN
        brclr   4,TMP_CHN,L1_2      ;if tmp_chn bit4 = 0, goto L1_2
        bset    SER_OUT,PORTA      ;ser_out = 1
        bra     L1_3               ;goto L1_3
L1_2    bclr    SER_OUT,PORTA      ;ser_out = 0

***      Clock the serial clock pin
L1_3    bset    SER_CLK,PORTA      ;ser_clk = 1
        bclr    SER_CLK,PORTA      ;ser_clk = 0

        decx                    ;decrease counter loop
        bne     LOOP1             ;is LOOP1 finished?

        ldx     #6T               ;init counter for LOOP2

```

```

***      Read the serial input pin
LOOP2    brclr   SER_IN,PORTA,L2 ;carry bit = serial in
L2       rol    RESULT+1        ;rotate left result
         rol    RESULT

***      Clock the serial clock pin
         bset   SER_CLK,PORTA   ;ser_clk = 1
         bclr  SER_CLK,PORTA   ;ser_clk = 0

         decx          ;decrease counter loop
         bne    LOOP2     ;is LOOP2 finished?

***      CS* high to finish serial transfer
         bset   CS,PORTA     ;CS* line is high

FOR      BRA    FOR         ;branch forever

         org   $03FE        ;reset vector
         dw   START

```

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.

How to reach us:

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE (602) 244-6609

INTERNET: <http://Design-NET.com>

USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298



MOTOROLA

AN1228/D

