



MC104P Software and Examples

PC104 Single board computer

©2005 Calmotion LLC, All rights reserved

**Calmotion LLC
9909 Topanga Canyon Blvd. #322
Chatsworth, CA 91311
www.calmotion.com**

- 1 -

Introduction

This manual is intended to provide a description of the software requirements and operation of the Calmotion MC104P controller. The manual will review the libraries used to initialize and implement the MC104P. The manual will also review how to use the Calmotion libraries to access the PC104 bus, VGA cards supporting the GD542x chipset and the Calmotion DC104 drive.

Requirements

All of the libraries used as examples in this manual were created using the C18 library builder from Microchip. The compiler and linker used for your project must be compatible with this library builder. In addition, the MPLAB® ICD2 programmer/debugger from Microchip will be required to program and download projects.

The Calmotion video library is designed to be used with a Calmotion MC104P CPU board and a PC104 VGA video card using a GD542x VGA chip. If needed, please refer to the Calmotion website www.calmotion.com for links to companies that offer GD542x based VGA cards. The user is encouraged to run the color sample project discussed later in this manual prior to using a video card in a new project. In addition to confirming the operation of the video card, the program will generate color combinations that the developer can choose from avoiding trial and error methods.

ICD2 Quick Start Instructions

1. Please refer to the “MPLAB® ICD2 In-Circuit Debugger User’s Guide” for detailed instructions on how to properly connect to a target device. Pay special attention to the power supply sequence description in Chapter 2.
2. Select Configure and then Configuration Bits in the main menu. The following configuration bits must be set as follows:
 - a. CONFIG1H: EC oscillator, port function on RA6
 - b. CONFIG3L: Wait states for table reads and table writes are determined by the WAIT1:WAIT0 bits. 8-bit external bus mode. 20 bit address bus. Extended Microcontroller mode.
 - c. CONFIG3H: MCLR# pin enabled; RG5 input pin disabled
3. Confirm that the target processor has been set for the PIC18F8722. Select Configure and then Select Device in the main menu.
4. Once your project has been loaded, make sure that the Large Code model has been selected. Project, Build Options, Project, MPLAB C18 tab, Categories, Memory Model,

Example of Use

In order to operate the video library, the `calmotion_video.lib` and `calmotion_MC104P.lib` need to be added to the project prior to use. The following is a “Hello World” example using these two libraries.

```
#include <p18f8722.h>
#include <stdio.h>
#include "pc104.h"

void main (void)
{
    init_MC104P();           // configure MC104P inputs, outputs, analog and pc104 bus
    init_cirrus_video();    // initialize the VGA video card with the Cirrus Logic
    cls();                  // clear the video screen of text and colors
    locate(1,1);           // set the cursor position to upper left of the screen
    color(0x7,0x0);        // set the modal color of the text to white on black
    printf("Hello world");  // display a message
    while(1)
    {}
}
```

Sample program instructions:

1. Copy the sample program files from the Sample program directory on the Calmotion CD to your hard drive. Be sure to copy all of the files in the directory.
2. Start the MPLAB IDE.
3. Select from the menu: ***File/Open Workspace...***
4. Find the *mcw* file on your hard drive copied from the Calmotion CD, i.e. ***color_sample.mcw***.
5. Once the workspace opens, you may receive an error that some of the libraries cannot be found. If this is the case, delete the libraries from the project window

Figure 1, Example program with Calmotion Library.

- on the MPLAB IDE by using the DEL key.
6. Add the two libraries, *calmotion_MC104P.lib* and *calmotion_video.lib*, copied from the Calmotion CD in the previous step. See Figure 2. The `p18f8722.h` and linker paths on the project may not be consistent with your particular file structure. If the compiler complains that it cannot find these files, add the files to your project or change the path in the General tab of the Build Options within the project.
 7. Build the project and program the MC104P card through the ICD2 connector.

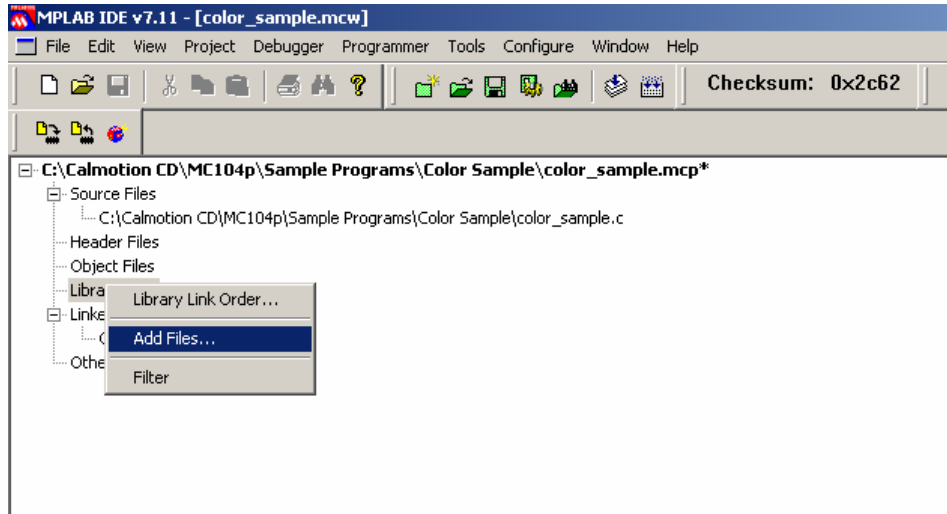


Figure 2, Adding libraries to the project window

Video library and stdout

The standard C compiler output can be changed from within the program. By default, the standard output for the Microchip C18 compiler and the PIC18F8722 is the serial port. (USART) As an example, the function printf() by default will use the serial port to output characters. Be aware when using the MC104P video library that the standard output device is changed when the initialization function for the video card is called. The init_cirrus_video() function will set the standard output to the video card by executing in C "stdout = _H_USER;" In order to return to the default output device, the standard output device can be changed back by executing in C "stdout = _H_USART;". Upon execution, all functions that use the standard output function will now use the serial port as the output device.

init_cirrus_video

Function: Sets VGA foreground and background color attributes used with the printf() function.

Include: pc104.h

Prototype: void init_cirrus_video(void);

Arguments: None

Remarks: None

Return Value: None

locate

Function: Sets VGA foreground and background color attribute to be used with the printf() function.

Include: pc104.h

Prototype: void locate(unsigned int row, unsigned int column);

Arguments: *row*
This is the value of the row to place the video cursor at. The range of the value is 0 and 25. When the row value specified is 0, the cursor's row position will not be modified. When the row value is 1, the cursor will be placed on the top line of the video screen. When the value is 25, the cursor will be placed on the bottom line of the screen.

column
This is the value of the column to place the video cursor at. The range of the value is 0 and 80. When the column value specified is 0, the cursor's column position will not be modified. When the column value is 1, the cursor will be placed on the far left of the video screen. When the value is 80, the cursor will be placed on at the far right of the screen.

Remarks: Rows go top to bottom on the video screen. Columns go left to right on the video screen.

Return Value: None

Function Definitions for the `calmotion_MC104P.lib`

The following are the detailed descriptions of the functions available with the Calmotion PC104 library:

`init_MC104P`

Function: Initializes the MC104P hardware peripherals including the PC104 bus requirements. Sets the ports for the digital inputs, digital outputs, analog inputs, and LEDs.

Include: `pc104.h`

Prototype: `void init_MC104P(void);`

Arguments: None

Remarks: None

Return Value: None

`inmemb`

Function: This function reads memory from the PC104 bus. The address passed to this function is indexed from the start of the MC104P PC104 memory. A passed address of `0x30000h` will point to `0x30000h` of PC104 memory. The PC104 memory address range accessible by the MC104P is from `0x30000h` to `0xFFFFFh`.

Include: `pc104.h`

Prototype: `unsigned char inmemb(unsigned long mem_add);`

Arguments: `mem_add`
Address of the byte to be read from the PC104 memory space.

Remarks: None

Return Value: The byte read from the PC104 memory address passed to the function.

Code Example: `scroll_char = inmemb(0xB8000);`

inportb

Function: This function will read from the ports on the PC104 bus. The address that is passed to this function is indexed from the start of the MC104P PC104 I/O. A passed address of 0x0000h will point to 0x0000h of PC104 I/O. The addressable range of PC104 I/O is 0x0000h to 0xFFFFh.

Include: pc104.h

Prototype: `unsigned char inportb(unsigned int io_add);`

Arguments: *io_add*
Address of the byte to be read from the PC104 I/O ports.

Remarks: None

Return Value: The byte read from the PC104 I/O from the address passed to the function.

Code Example: `ack = inportb(0x7FF);`

outmemb

Function: This function writes to the PC104 memory bus. The address passed to this function is indexed from the start of the MC104P PC104 memory. A passed address of 0x30000h will point to 0x30000h of PC104 memory. The addressable range of PC104 memory is 0x30000h to 0xFFFFFh.

Include: pc104.h

Prototype: `void outmemb(unsigned long mem_add, unsigned char mem_val);`

Arguments: *mem_add*
Address of the byte to be written to the PC104 memory space.
mem_val
Byte value to be written to the PC104 memory pointed to by *mem_add*.

Remarks: None

Return Value: None

Code Example: `outmemb(0xB8000, 0);`

outportb

Function: This function will write a byte to an I/O port on the PC104 bus. The address passed to this function is indexed from the start of the MC104P PC104 I/O. A passed address of 0x0000h will point to 0x0000h of PC104 I/O. The addressable range of PC104 I/O is 0x0000h to 0xFFFFh.

Include: pc104.h

Prototype: void outportb(unsigned int io_add,
unsigned char io_val);

Arguments: *io_add*
Address of the byte to be written to the PC104 memory space.
io_val
Byte value to be written to the PC104 memory pointed to by mem_add.

Remarks: None

Return Value: None

Code Example: outportb(0x10000, red);

Function Definitions for the calmotion_utility.lib

The following are the detailed descriptions of the functions available with the Calmotion Software Utility library:

init_utility

Function: Initializes the Com 2 UART to be compatible with the MC104P software utility for either the Windows® or Palm® OS versions. The interrupt level is also passed to this function. The user must ensure that the data_xfer() is placed in the appropriate interrupt handler set by priority.

Include: pc104.h

Prototype: void init_MC104P(unsigned char priority);

Arguments: *priority*
Priority level of the data_xfer() function
0 = low priority
1 = high priority

Remarks: None

Return Value: None

Code Example: init_utility(0);

data_xfer

Function: Handles the data transfer to the MC104P utility. This function should be placed in an interrupt routine with the priority level set in the init_utility function. The user should check to see if the UART2 interrupt flag has been set prior to executing this function. The data_xfer function will clear the interrupt from within the function.

Include: pc104.h

Prototype: void data_xfer(void);

Arguments: None

Remarks: None

Return Value: None

Code Example:

```
if(PIR3bits.RC2IF == 1) //located within ISR
    {data_xfer();}
```

MC104P_utility

Function: Provides general housekeeping and error recovery for COM2. MC104P_utility should be placed in a portion of the user's code that executes approximately every 2mS or less for best performance. If the period is longer, the utility will continue to function. However, there may be some performance loss if the integrity of the RS-232 signals is not robust.

Include: pc104.h

Prototype: void MC104P_utility(void);

Arguments: None

Remarks: None

Return Value: None

Code example:

```
MC104P_utility(); //located within a
//periodic loop
```

Sample program “color sample.c”

This sample program will configure a VGA card connected to the MC104P. It will then display a color grid with all the corresponding background and foreground values available for selection.

```
/* Copyright 2005 Calmotion LLC

   Date: July 21, 2005
   Version: 1.0

   Purpose: Show a color chart for foreground and background possibilities.
*/

#include <p18f8722.h>
#include <stdio.h>
#include "pc104.h"

void main (void)
{
char row, column, foreground, background;
  init_MC104P(); // initialize the MC104P ports
  init_cirrus_video(); // initialize the video card with the cirrus chip set
  cls(); // clear the screen
  color(7,0);
  locate(1,1);
  printf("First\n");
  printf("hex #\n");
  printf("is the\n");
  printf("fore-\n");
  printf("ground\n");
  printf("color\n");
  printf("\n");
  printf("Second\n");
  printf("hex #\n");
  printf("is the\n");
  printf("back-\n");
  printf("ground\n");
  printf("color\n");
  printf("\n");

  row=19;
  column=2;
  foreground=0;
  background=0;
  for(background=0; background<16; background++) // cycle through all the background colors
  {
    for(foreground=0; foreground<16; foreground++) // cycle through all the foreground colors
    {
      row++;
      if(row>25)
        {column = column + 7; // past the last row, reset to top of
          //screen, move column over 7 spaces
          row=1;
        }
      locate(row, column); // set the cursor position
      color(foreground,background); // set the foreground and background
      printf(" ");
      locate(row, column); // set the cursor position
      // display foreground and background hex numbers on the screen
      printf("0x%x 0x%x",foreground,background);
    }
  }
  while(1)
  {}
}
```

Sample program “hello world.c”

This sample program will configure the VGA card connected to the MC104P and display a “Hello World” message in white on black on the screen.

```
/* Copyright 2005 Calmotion LLC

   Date: July 21, 2005
   Version: 1.0

   Purpose: Demonstrate the MC104P use of the library calmotion_video.lib.
            The program will set the text colors and print
            a Hello World message.
*/

#include <p18f8722.h>
#include <stdio.h>
#include "pcl04.h"

void main (void)
{
    init_MC104P();           // initialize the MC104P ports
    init_cirrus_video();    // initialize the video card with the cirrus chip set
    cls();                  // clear the screen
    locate(1, 1);          // set the cursor position
    color(7,0);             // set the foreground to white and background to black
    printf("Hello World"); // display Hello World statement on screen
    while(1)
        {}
}
```

Sample program “IRQ sample.c”

This sample program will configure the MC104P and its Utility software. The four IRQ's and high-speed inputs will also be configured. When an interrupt from the PC104 IRQ4, IRQ5, IRQ6 or IRQ7 is handled, this sample program will increment a memory location and set the LEDs on the MC104P board. The incremented memory location and the LED are for debugging/confirmation purposes and are not necessary.

```
/* Copyright 2005 Calmotion LLC

Date: August 4, 2005
Version: 1.0

Purpose: Handle IRQ4, IRQ5, IRQ6 OR IRQ7 or if the interrupts tied to inputs
using jumpers JP1, JP2, JP3 or JP4.

For debug help, the MC104P utility software support has been added. This utility will
use COMM 2 port connect to the Windows MC104P or Palm MC104P software utility.
When an interrupt occurs, a memory location will be incremented to show the interrupt was
handled. Also, the MC104P LEDS 1 and 2 will show which interrupt occurred in binary.

*/

#include <p18f8722.h>
#include <stdlib.h>
#include <stdio.h>
#include "pc104.h"

void high_isr(void);
void low_isr (void);
int rand( void );

#pragma code high_vector=0x08
void interrupt_at_high_vector(void)
{
    _asm GOTO high_isr _endasm
}

#pragma code

#pragma interruptlow high_isr

void high_isr (void)
{
    unsigned char IRQ_count;

    // IRQ4 or High speed input RC0 Service routine
    if(INTCONbits.INT0IF == 1)
    {
        INTCONbits.INT0IE = 0; // disable int
        INTCONbits.INT0IF = 0; // clear int flag
        INTCONbits.INT0IE = 1; // enable int

        // debug help:
        // increment the memory location 0x1 for MC104P Utility Software
        // LED2=0 LED1 = 0
        IRQ_count=inmemb(0x100001);
        IRQ_count++;
        outmemb(0x100001,IRQ_count);
        PORTGbits.RG3 = 0; // clear LED2 to show INT0 was received
        PORTGbits.RG4 = 0; // clear LED1 to show INT0 was received
    }
}
```

Figure 3, IRQ and high speed input example

```

// IRQ5 or High speed input RC1 Service routine
if(INTCON3bits.INT1IF == 1)
{
    INTCON3bits.INT1IE = 0;        // disable int
    INTCON3bits.INT1IF = 0;        // clear int flag
    INTCON3bits.INT1IE = 1;        // enable int

    // debug help:
    // increment the memory location 0x1 for MC104P Utility Software
    // LED2=0 LED1 = 1
    IRQ_count=inmemb(0x100002);
    IRQ_count++;
    outmemb(0x100002,IRQ_count);
    PORTGbits.RG3 = 0;            // clear LED2 to show INT1 was received
    PORTGbits.RG4 = 1;            // set LED1 to show INT1 was received
}

// IRQ6 or High speed input RC2 Service routine
if(INTCON3bits.INT2IF == 1)
{
    INTCON3bits.INT2IE = 0;        // disable int
    INTCON3bits.INT2IF = 0;        // clear int flag
    INTCON3bits.INT2IE = 1;        // enable int

    // debug help:
    // increment the memory location 0x1 for MC104P Utility Software
    // LED2=0 LED1 = 1
    IRQ_count=inmemb(0x100003);
    IRQ_count++;
    outmemb(0x100003,IRQ_count);
    PORTGbits.RG3 = 1;            // set LED2 to show INT2 was received
    PORTGbits.RG4 = 0;            // clear LED1 to show INT2 was received
}

// IRQ7 or High speed input RC3 Service routine
if(INTCON3bits.INT3IF == 1)
{
    INTCON3bits.INT3IE = 0;        // disable int
    INTCON3bits.INT3IF = 0;        // clear int flag
    INTCON3bits.INT3IE = 1;        // enable int

    // debug help:
    // increment the memory location 0x1 for MC104P Utility Software
    // LED2=0 LED1 = 1
    IRQ_count=inmemb(0x100004);
    IRQ_count++;
    outmemb(0x100004,IRQ_count);
    PORTGbits.RG3 = 1;            // set LED2 to show INT3 was received
    PORTGbits.RG4 = 1;            // set LED1 to show INT3 was received
}
}

#pragma code low_vector=0x18
void interrupt_at_low_vector(void)
{
    _asm GOTO low_isr _endasm
}

#pragma code

#pragma interruptlow low_isr

void low_isr (void)
{
    if(PIR3bits.RC2IF == 1)
    {data_xfer();}
}

```

Figure 4, IRQ example continued...

```

void main(void)
{
unsigned int turn_on_off_dc104;

    init_utility();           // initialize the MC104P Software Utility monitor
    init_MC104P();           // initialize the MC104P hardware
    RCON |= 0x80;            // enable priority levels on interrupts
    INTCON |= 0xc0;          // enable gobal ints, peripheral
    INTCONbits.INT0IE = 1;   // enable INT0
    INTCON3bits.INT1IE = 1;   // enable INT1
    INTCON3bits.INT2IE = 1;   // enable INT2
    INTCON3bits.INT3IE = 1;   // enable INT3

    TRISGbits.TRISG4 = 0 ;    //make LED an output
    TRISGbits.TRISG3 = 0 ;    //make LED an output

    while(1)
    {
        MC104P_utility();     // Periodic monitoring of MC104P software utility operation
                                // Should run every 2mS or less for optimal performance
                                //(run windows/palm software and connect to COMM 2)
    }
}

```

Figure 5, IRQ example continued...