HC9S08JM60 USB Device Demonstration Suite

This package contains a set of demonstration programs for the HC9S08JM60.

These programs are tested on a DEMOJM Rev. A development board in conjunction with a Windows XP system.

These demonstrations are provided as complete Metrowerks Code Warrior projects with full source code.

The code is designed to show how to use various aspects of USB Device mode in conjunction with the HC9S08JM60.

Basic Directory structure

The directory tree groups files by functionality and by dependency. Subfolders named after a developers environment contain files specific to the particular environment. Subfolders named after an MCU hold files specific to that MCU. Folders named after a project hold files specific to that project. For example the folder "/usb-peripheral/project/CodeWarrior/hcs908jm60/cdc" contains files specific to usb peripheral functionality, CodeWarrior developers environment, HC9S08JM60 MCU and the cdc demo project.

/usb-common
Contains files shared by multiple projects independent of host or device functionality. Still these files may be specific to a micro controller. Such files are placed to a subfolder named after the specific micro controller.

/usb-peripheral
Contains usb peripheral specific project and source files.

/usb-peripheral/ projects
Contains developer environment and MCU specific files.

/usb-peripheral/ pc-side/hid_led_demo
Contains a Visual Studio 7 Project for accessing an embedded device through the HID class over USB. Using the application four LEDs on the board can be controlled, and status of two switches read. Note: to be able to recompile the project some header files and libraries are needed from the Windows XP Driver Development Kit available from Microsoft. These are: hid.lib, hidclass.lib, hidparse.lib, hidpi.h, hidsdi.h, hidusage.h.

/usb-peripheral/src
Contains the USB source code that is developer environment independent.

Source Code

The source code is contained in the following files:

/ usb-common/hc9s08jmxx

hcc_types.h	Common type definitions.
target.c	Hardware (board) specific routines. Mainly related to
target.h	initialization.

/ usb-common/hc9s08jmxx/uart-drv

uart.c	Simple SCI driver.
uart.h	

/ usb-common/terminal

terminal.c	Simple terminal implementation. It is able to execute
terminal.h	commands specified during initialization.

/usb-common/terminal/utils

utils.c	Basic utilities mainly related to string conversion.
utilsl.h	

/usb-common/cdc_drv

usb_cdc.c	CDC layer for the USB driver. Provides UART like API
usb_cdc.h	for an application.

/usb-peripheral/src/hc9s08/usb-drv

usb.c	USB device driver source.
usb.h	
usb_config.h	USB driver compile time configuration parameters. This includes another file from the current project based on conditional compiling directives.

/usb-peripheral/src/hc9s08/hid-demo

hid.c	HID device layer for the USB driver.
hid.h	
hid_generic.c	Generic HID demo. Allows control of on board LEDs and
hid_generic.h	read status of on board switches.
hid_kbd.c	HID keyboard demo.
hid_kbd.h	
hid_mouse.c	HID mouse demo.
hid_mouse.h	
hid_usb_config.c	USB configuration for the three HID demos.
hid_usb_config.h	
ints.c	Interrupt and exception handlers.
hid_main.c	Main entry point. Will select which demo to execute.

/usb-peripheral/src/hc9s08/cdc-demo

HC9S08JMxx.inf	Driver descriptions file for windows. When the device is first connected and windows looks for a driver specify the folder holding this file. Windows will identify the device as an USB based serial port.

/usb-peripheral/src/hc9s08/cdc-demo/cdc2serial

cdc_main.c	Main loop. Contains CDC to UART bridge functionality.
cdcs_usb_config.c	USB configuration for the CDC demo.
cdcs_usb_config.h	

/usb-peripheral/src/hc9s08/cdc-demo/terminal

cdc_main.c	Main loop. Contains CDC terminal (simple shell) functionality.
cdct_usb_config.c	USB configuration for the CDC demo.
cdct_usb_config.h	

HC9S08JM60 Demonstration Projects

HID Class Project

The HID class project is effectively three demonstrations – depending on which switches are pressed at startup the demonstration is selected – you must restart the device to make a new selection:

HID Keyboard (no switches pressed at startup)

This simple demonstration enables the device to behave as a keyboard – if switch 1 is pressed (PTG0) a page_up is sent to the host and if switch 2 (PTG1) is pressed a page_down is sent. It is a simple matter to modify this loop to take a different input source and generate different key values. Four LEDs on the board work as lock key indicators (Num Lock, Caps Lock, etc…)

Note: you can run this while your standard keyboard is connected to your PC – this will effectively just provide an extra key source to the host.

HID Mouse (switch 1 (PTG0) pressed at startup)

This demonstration turns the device in to a standard mouse. The demo will move the mouse cursor left to right and back while connected to the PC.

Note: you can run this while your standard mouse is connected to your PC – this will effectively just provide an extra mouse movement source to the host.

HID Generic Device (switch 2 (PTG1) pressed at startup)

This demonstration is for use with the VC++ project described below. The device configures itself as a vendor specific HID device and the VC++ application is able to read data from and write data to the device.

The code has two reports – an input report and an output report.

If the device receives an output report from the host application then it sets the state of four LEDs to reflect the value sent.

If the device receives an input report then it reads the settings of switches one and two and returns the combined value to the host.

Note: All operations are initiated by the host software – the VC++ application.

CDC Class Projects

These demos will turn the device into a USB to serial device a device that emulates a serial port for the host. This emulated serial port is called hereafter virtual COM port (short VCP) and is implemented using the Communication Device Class. The VCP on the embedded side is connected to an API which forms a virtual serial line (VSL). The system acts if the VCP and the VSL would be connected by a virtual cable. Any data sent to the VCP on the PC is available for reception on the VSL and vice versa. This way a PC application can easy connect an application on the embedded side. Communication properties defined on the PC side can be read on the embedded side but will not affect real communication in any way.

Installation (Windows XP)

Build the CDC demo application project and download to the HC9S08JM60. When the application is started the PC shall recognize the new device.

When the device is first recognized by the PC, Windows will ask for a driver. Please specify the folder that contains HC9S08JMxx.inf, and windows will install the necessary driver files. After the installation is done, a new serial port can be seen in the device manager under "Ports (COM & LTP)". The COM port assigned to the device can be changed here by right clicking the device and selecting properties.

Using the CDC terminal demo

In this demo the VSL is used by a simple shell application. The shell inputs characters from the VSL and sends its output to the VSL. Is can recognize and execute some simple commands.
After the installation is done, start HyperTerminal or any other terminal client application. Configure it to use the virtual COM port assigned to the HC9S08JM60 (e.g. COM4). Disable handshake mechanisms and local echo in the terminal client. Select line coding properties you like.
Issue “help” in hyper terminal and the shell will list all commands with a brief description it knows.

Using the cdc2serial demo

In this demo the VSL is connected to the serial port of the HC9S08JM60. Line coding settings of the serial line will match the settings of the VCP on the PC. Note: hardware handshaking is not supported.
After the installation is done, start HyperTerminal (or any other terminal client) application. Configure it to use the COM port assigned to the HC9S08JM60 (e.g. COM4). Disable hardware handshake and set line properties you wish to use.
Start TerminalWindow.exe part of the DEMOJM_Toolkit. Configure the properties of the serial line to match the setting of HyperTerminal. Press the “Open Serial Port” button.

Any characters typed in HyperTerminal will show up on the screen of TerminalWindow and vice versa.

Microsoft VC++ Version 7.0 Generic HID Class Project

To be able to build this project you need to have the Microsoft DDK installed.
The following files are required from the DDK.

hid.lib
hidclass.lib
hidparse.lib
hidpi.h
hidsdi.h
hidusage.h

By default the installation contains a pre-built executable to make testing possible without having the DDK.
Once the Generic HID demonstration is running on the target and it is connected to the host PC you can use the GUI to control the device.

The GUI of the application has four buttons to control the four LEDs on the board, and two check boxes to reflect the state of SW1 and SW2. Any status change will take effect immediately.

Note: if the device is not connected to the PC when the PC side demo is started it will give an error sound and exit. Nothing will be seen on the screen.