uIP port to the HCS12NE architecture

Introduction

This document describes the uIP port to Freescale Semiconductor MC9S12NE family of microcontrollers, more specifically to the MC9S12NE64 MCU which has an on-chip Ethernet media access controller (EMAC) with integrated 10/100 Mbps Ethernet physical transceiver (EPHY) allowing it to connect to the worldwide web. An Adapt9S12NE64 evaluation board manufactured by Technological Arts was used to deploy and test this port. (http://www.technologicalarts.ca)

The port is written in C, compiled and assembled with the GNU development chain for the 68HC11 & 68HC12 (http://www.gnu-m68hc11.org) using the Windows based EmbeddedGNU IDE (http://www.geocities.com/englere_geo). Adapting this project to use another C compiler should be straightforward.

Description of the uIP HCS12NE port

The port is made out of these files and folder:

Name	Content
critical.h	Definitions of macroinstructions used to control CPU interrupt recognition.
datatypes.h	Definitions of data types used in this project.
ethernet.h	Ethernet driver has one compile option to be set here.
ethernet.c	Ethernet driver module are made of four function calls and eight interrupt service routines. It can be compiled separately as it does not use any uIP definition.
ethernet_stats.c	Same Ethernet driver module with added link statistics and and extra function to access the gathered statistics.
main.c	uIP main event control loop with support routines to handle uIP timing and event logging.
mc9s12ne_regs.h	Definitions of most registers of the MC9S12NE64 MCU family.
mc9s12ne_vectors.s	The project interrupt vectors table to be assembled.
netlog.h netlog.c	A simple application demonstration which routes uIP logging output from SCI channel 1 (at 57600 bauds) to TCP port 23 (telnet) on the fly.

timer.h	
timer.c	Module to offer a delay routine and countdown timers (up to 254) with a resolution of 0.1s (100ms). It uses TIMER output compare 4 (OC4) interrupt as the tick clock.
uip.h	
uip.c	The uIP TCP/IP stack code.
uip_arch.h	
uip_arch.c	Functions to implement the IP check sum and 32-bit additions for the HCS12 architecture.
uip_arp.h	
uip_arp.c	Implementation of the ARP Address Resolution Protocol.
uipopt.h	Configuration options for uIP to be compiled.
uip-ne64.mak	uIP project make file created by EmbeddedGNU.
uip-ne64.prj	uIP project file.
9s12ne64-0.mem	Hardware profile file used by EmbeddedGNU to define project options and MCU memory map for 3 FIFO buffers of 128 bytes. (bufmap = 0). UIP_BUFSIZE is defined as 128 and 7680 bytes of RAM are left for the application.
9s12ne64-1.mem	Hardware profile file used by EmbeddedGNU to define project options and MCU memory map for 3 FIFO buffers of 256 bytes. (bufmap = 1). UIP_BUFSIZE is defined as 256 and 7168 bytes of RAM are left for the application.
9s12ne64-2.mem	Hardware profile file used by EmbeddedGNU to define project options and MCU memory map for 3 FIFO buffers of 512 bytes. (bufmap = 2). UIP_BUFSIZE is defined as 512 and 6144 bytes of RAM are left for the application.
9s12ne64-3.mem	Hardware profile file used by EmbeddedGNU to define project options and MCU memory map for 3 FIFO buffers of 1024 bytes. (bufmap = 3). UIP_BUFSIZE is defined as 1024 and 4096 bytes of RAM are left for the application.
9s12ne64-4.mem	Hardware profile file used by EmbeddedGNU to define project options and MCU memory map for 3 FIFO buffers of 1536 bytes. (bufmap = 4). UIP_BUFSIZE is defined as 1500 and 2084 bytes of RAM are left for the application.
memory.x	Memory layout file created by EmbeddedGNU to be used by GNU linker.
\ uip-0.9	Folder contains all the other files part of the uIP release 0.9

RAM memory issues

The MC9S12NE64 comes equipped with 8K bytes of RAM and a portion is used by the EMAC module. Register (BUFCFG) configures part of the system RAM into FIFO buffers using five different maps. Each of these maps assigns a fixed amount of RAM to the FIFO buffers. RAM for user's application is dependent on the EMAC module chosen map.

We have provided five hardware profile files which define available RAM to the user's application. Each of these files must be used with the specified buffer map.

UIP_BUFSIZE must be adjusted manually via the uipopt.h file for the uIP buffer to fit into each of the FIFO buffers.

The buffer map number (0..4) is one of the parameter that must be specified on the ethernet_open() call.

Driver operating modes

Five modes are offered: FULL100, HALF100, FULL10, HALF10 and AUTONEG, which are full and half duplex at 100 Mbps, full and half duplex at 10 Mbps, and Auto-Negotiation which selects 100 Mpbs or 10 Mbps at half duplex depending on the link hardware.

The Ethernet communication mode is the second parameter that must be specified on the ethernet_open() call.

License issues

Pierre Morency <pmorency@globetrotter.net>
2004-07-20