# ECET330 Lab 1 Part 1 Procedures

# Title: Introduction to Memory Map

# I. OBJECTIVES

1. To become familiar with address decoding
2. To become familiar with HCS12 memory map

## II. PARTS LIST

###### Equipment: IBM PC, or compatible with Windows 2000 or higher

# INTRODUCTION

For the CPU to process information, the data must be stored in RAM or ROM. ROM is a type of memory that does not lose its contents when the power is turned off. Different types of ROM are PROM, EPROM, EEPROM, Flash EPROM, and mask ROM. The CPU is connected to RAM and ROM using the address bus, data bus, and control bus.

RAM: Stands for random access memory; refers to memory that the microprocessor can read from and write to. Program variables and data as well as stack data storage are saved in RAM of the microcontroller.

ROM: Stands for read-only memory. The microcontroller can read from ROM, but it can’t write to or modify it.

EEPROM: One type of ROM is EEPROM, which is Electrically Erasable Programmable ROM. Variables that must remain when the power is turned off are stored in EEPROM.

Flash memory: A special type of memory that works like both RAM and ROM. You can write information to flash memory, like you can with RAM, but that information isn’t erased when the power is off, like it is with RAM. All program code, constants such as messages and lookup tables, and any other information that does not change, are saved in flash memory of the microcontroller.

The programs for an embedded system must be in read-only memory, and thus we must know something about the hardware upon which our software is installed. For example, we need to know the memory location where the code or data should reside. Figure 1 shows the block diagram of HCS12 microcontroller.

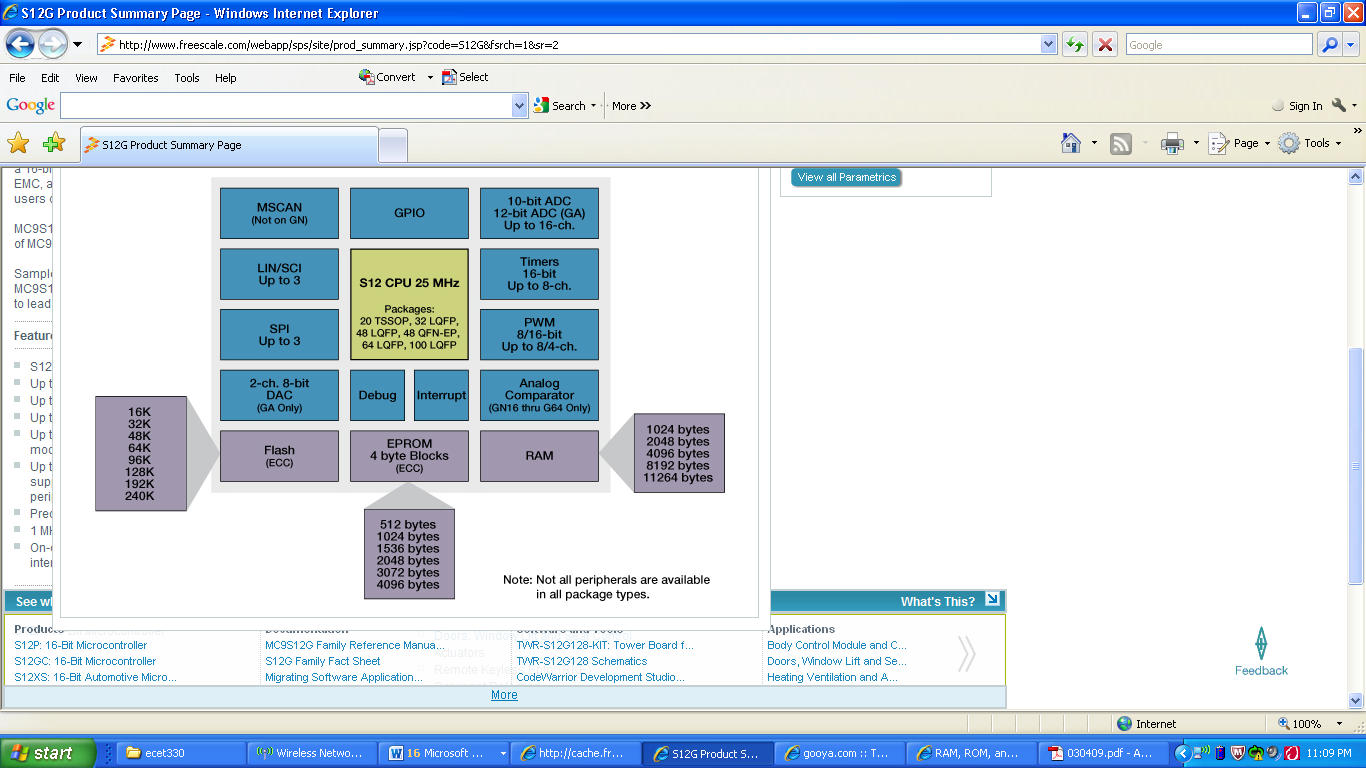


Figure 1: Block Diagram of HCS12 Microcontroller

This block diagram shows that there are three types of memory in HCS12 called Flash, EPROM, and RAM. Note that the code in Flash and EEPROM will remain after the power is lost. The variables which will be stored in RAM would be lost after the power is lost.

These memory units are connected to the CPU of the microcontroller using internal address, data, and control buses. The CPU uses its address lines to select a given location of one of these memory units. Some of these address lines will be directed to decoder circuits (not shown in the block diagram), which would select the given memory unit.

**Memory Map**: A diagram showing which address region is used for which type of memory is called a memory map. Each microcontroller has a reference manual which would define the memory map of that particular microcontroller.

## PROCEDURE

# Figure 1 below shows a given CPU connected to two 8Kx8 SRAM units, one 16Kx8 SRAM unit, and a 32Kx8 SRAM unit. Address lines A13, A14, and A15 are used to select one of the memory units. Looking at the figure below, complete the Table 1 that represents the memory map of Figure 1.

Address Range

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Memory Unit** | **A15** | **A14** | **A13** | **Lower Limit** | **Upper Limit** | **Size in Bytes** |
| *RAM1* | 0 | 0 | 0 | $0000 | $1fff | 8K |
| *RAM2* |  |  |  |  |  |  |
| *RAM3* |  |  |  |  |  |  |
| *RAM4* |  |  |  |  |  |  |

Table 1: Memory Address Ranges for the Circuit Shown in Figure 1



2

Figure 1: Memory Decoding Example

1. Assume that the content of all locations of the memory units in Figure 1 is $ff. Assume that the CPU places $0000 on the address lines, $02 on the data lines, and activates the WR signal. How does the memory system of Figure 1 react to these values?

1. Explain which memory unit in Figure 1 is selected, what happens if the value on the address lines is $4500, and what occurs when the RD signal is activated.
2. Given Table 2, how much on-chip RAM memory do we have for the MC9S12G128 microcontroller?

|  |  |  |
| --- | --- | --- |
| **Memory Unit** | **Lower Limit** | **Upper Limit** |
| *Registers* | $0000 | $03FF |
| *EEPROM* | $0400 | $13FF |
| *RAM* | $2000 | $3FFF |
| *Flash ROM* | $4000 | $FFFF |