Embedding it better...

µTasker Document

MCF5207, MCF5208

Table of Contents

1. Introduction

The µTasker project for the Coldfire V2 started with the availability of the M5223X. This device is a true single chip processor with embedded Ethernet FEC and PHY. The M5207 and M5208 are older devices with internal FEC (requiring external PHY) and do not have internal FLASH. They do reside over 16kByte internal SRAM, 8kBytes configurable cache and have a dual-bank SDR/DDR SDRAM controller to allow large external memory resources.

The M5207 and M5208 can operate up to 166.67MHz and its external memory interface up to 83.33MHz

A quick comparison between the M5223X and these reveals the following main differences:

- The internal FEC requires an external PHY to complete the Ethernet interface
- 8k-byte unified cache
- No ADC
- External bus interface supporting SDRAM and the ability to boot from an external device; up to 32 data bus width and 24 bit address bus and up to 6 chip select lines
- Enhanced 16 channel DMA controller
- No 4-channel general purpose timer
- No PWM module
- No CAN module

The following document concentrates on the most important details for understanding how to use the M5282, using the M5223X project as comparison. It does not attempt to replace the user's guide but instead tries to keep things as simple as possible – identifying and elaborating on the most important practical points.

The device name M5208 is used generally for all of the following devices:

- M5208 – with Fast Ethernet controller in 160 QFP or 196 MAPBGA package
- M5207 – without Fast Ethernet Controller in 144 LQFP or 144 MAPBGA

2. Clock Module

The M520X has a different clock module to the M5223X. The PLL configuration is thus different. One reason being that it generates two different clocks; one for the core and one for the bus. The core clock can operate up to 166Mz but the bus clock only to half of this frequency.

The crystal frequency should always be 16.000MHz or 16.6667MHz.

The PLL frequency is also determined at reset depending on configuration settings although some settings can be modified by software later, including a dithering mode for the PLL.

3. Processor Modes

The M5208 operates always in MASTER MODE. This means that the external bus is always active (generally no port sharing).

The Chip Configuration Module (CCM) is responsible for the detailed configuration.

If the control input RCON is pulled high (‘1’) at reset the chip will configured with default settings

When RCON is pulled low ‘0’ the configuration is defined by several data lines on the data bus. These are driven by external hardware to the required state.

D1, D2, D6 and D7 determine the PLL configuration. The allowed crystal frequencies are 16.000MHz and 16.667MHz

Data lines read at reset when RCON is asserted	Crystal frequency	PLL setting
D1	-	‘1’ = 166MHz (peripheral clock 88MHz) ‘0’ = 88MHz (peripheral clock 44MHz
D2	‘0’ oscillator input ‘1’ = crystal input	- -
D6	‘0’ Limp mode disabled ‘1’ Limp mode is enabled	- PLL is disabled and the CPU is clocked by a fraction of the input clock frequency
D7	‘1’ = 16.667MHz ‘0’ = 16.000MHz	-

D4 and D3 control the CS0 line to be asserted as 8-bit, 16-bit or 32-bit port

D4	D3	Boot width
0	0	32-bit width
0	1	16-bit width
1	0	8-bit width
1	1	32-bit width

When D5 is read as '1' the port drive strength is set to full drive strength. '0' results in partial strength.

When D9 is read as '1' the bus signals A23 and A22 are configured as CS4 and CS5 respectively, rather than having their address line use.

The configuration lines need to be controlled by external hardware during reset.

When simulating the M5208 using the µTasker simulator the mode can be controlled in `app_hw_m5223x.h`. By setting the define `CHIP_CONFIGURATION` accordingly.

```
#define CHIP_CONFIGURATION RCON_ASSERTED  // read configuration from ports
```

or

```
#define CHIP_CONFIGURATION         0      // default - single-chip mode
```

The initial input state can furthermore be defined by setting:

```
#define FORCE_D9_D1    (0x95)
```

This corresponds to a configuration: '10010101':

- CS[4:5]
- 16MHz crystal
- Limp mode disabled
- High drive strength
- 16-bit boot port size
- Crystal oscillator mode
- 166.667MHz / 83.333MHz (CPU/peripheral) speeds

The values of `PODR` and `PFDR` (PLL configuration registers) after initialization can be checked in the simulator to verify that the settings correspond to the desired mode. In addition, the default settings of various other registers are conditional on the configuration.

Eg. To check the value of PODR in the µTasker simulator do the following:

1) Search for the register name in the file M5223x.h. The define is as follows:

```
#define PODR  *(unsigned short *)(CLOCK_MODULE_ADD + 0x0)
                                        // PLL Output Divider Register
```

This shows that the register belongs to the CLOCK_MODULE_ADD block. It is 16 bits wide and located at an offset of 0x0 from the start of the block.

2) Now search for the definition of CLOCK_MODULE_ADD. It will be found twice, once for hardware use and once for simulator use:

```
#define CLOCK_MODULE_ADD    (0xfc090000)

#define CLOCK_MODULE_ADD    ((unsigned char *)(&ucM5223X.SimCMR))
```

3) Select the struct ucM5223X.SimCMR and drag it into the simulator's watch window.

4) In the watch windows the struct can be expanded and the register contents displayed. Note that this can only be performed when the simulator has been paused.

All other registers can be viewed in an analog manner.

4. Stack and Interrupt Vectors

The stack is positioned in internal SRAM. This is 16k in size and so generous for the processor stack use. The Interrupt vectors are placed at the bottom of this SRAM (reducing stack place to 15k, minus any other variable placed in it).

The SRAM is location at 0x80000000 and so the initial stack pointer is set to 0x80004000.

5. Boot-ROM and SDRAM

The width of the bus when booting (using CS0) is configured by the state of D[4:3] at reset. The M5208EVB uses a 16 bit FLASH as boot-ROM. Since the M5208 has no internal FLASH/ROM memory it has to have initialization code in the external Boot-ROM (FLASH).

One of the first configurations by the initialization SW is to configure the SDRAM. This is located at 0x40000000 and, as soon as it is configured for use, initialized variables are copied to it and the BSS (zeroed variable) are also set in it.

The HEAP is also located in the SDRAM. Note that accesses in the SRAM are faster than in SDRAM and so some critical variables or programs can be located in SDRAM as long as the remaining stack space is adequate for program execution. As long as it is not necessary to have large Ethernet buffers their location in SRAM (rather than SDRAM) can have some performance advantages.

Due to the differences in memory usage in comparison to M522XX µTasker projects the following adjustments are valid in this configuration:

- Stack size monitoring is from the last SRAM location to the stack pointer (rather than from the last HEAP location).

Since FLASH is not very fast (eg. 70ns) it has to be accessed with a number of wait states. This means that running program directly from FLASH doesn't allow the program to be executed at the speed that the processor could achieve. SDRAM, on the other hand, can operate at speeds close to the processor's maximum clock rate. For this reason the program in FLASH will typically be copied to SDRAM for subsequent execution. This means that the program is compiled for its final execution location (in SDRAM) but is actually downloaded to a specified area of the FLASH - *it cannot actually operate from this area unless it is especially compiled as position-independent code, which is generally not the case* –and copied from there to its final location before the controlling program (boot-loader) passes control to this program. This means that the boot-ROM needs to initialise the SDRAM, copy the code to it and finally jump to the start address of the program contained in that code.

The application code itself can be compiled as if it starts at the first SDRAM address (or at an offset in case the boot loader code needs to retain its own variables) and doesn't need to perform any further initialization of memory. Its flow can be as follows:

1) First address in SDRAM is the stack pointer initialization value (this value can be copied by the boot loader to ensure that the SP corresponds to that required by the application)
2) The second address in SDRAM is the program counter's initialization value (this can be used by the boot loader to jump to the first program instruction)
3) The application doesn't need to configure any hardware (chip selects, PLL, RAMBAR, interrupt vector base address, etc.) unless it specifically requires a different configuration to that defined be the boot loader.
4) The application should initialize the BSS section to ensure that zero initialized variables start with the correct value (BSS variables do not need to occupy FLASH). Other variables do not need to be initialised since they are located within program code space and are automatically initialized by the code copy.

Note that, for maximum performance, a 32 SDRAM should be used since this enables fetches of 32 bit instructions with a single external bus access. When 16 bit SDRAMs are used two reads are required to fetch a single 32 bit instruction.

6. Interrupt Controller

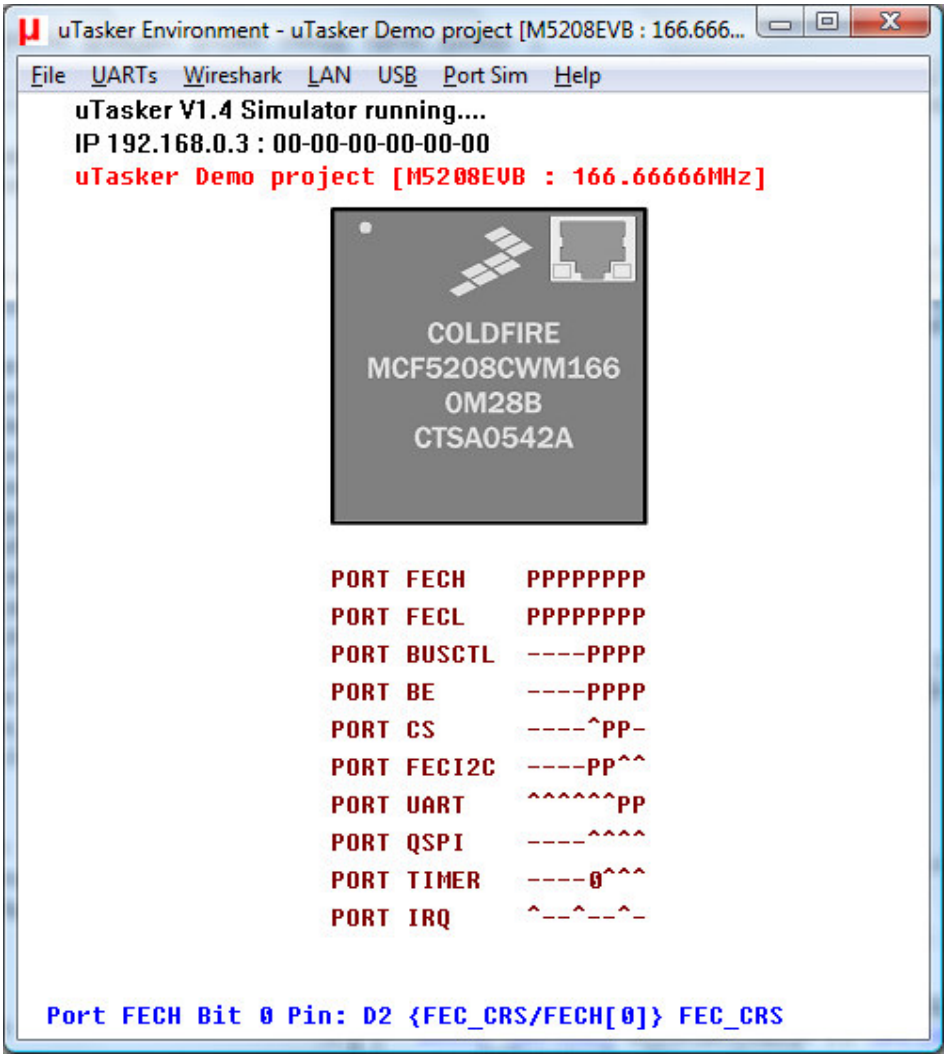
There are some differences in the interrupt controller in comparison to the M522XX.

The interrupt numbers are different and so a separate configuration for the M520X is required.

7. Ports

The M5208 has a large external memory interface (32 data lines, 24 address lines and various chip select and memory control lines). These are dedicated signals since the device operates exclusively in MASTER mode.It has a limited number of ports which are shared with peripherals.

The µTasker simulator shows the ports, including their peripheral function and state as it is operating, making verification of their configuration possible. By hovering the mouse over the port of interest its possible configurations and actual function are displayed as well as the pin number where the connection can be found on the package.



µTasker simulator displaying the M5208 and its ports during operation

8. Peripheral Power Management

9. FEC

The M5208EVB uses the National DP83848 as PHY. The PHY has the address 0x01 and reads the identifier code 0x20005c90 (whereby the last 4 bits are the chip's revision number).

When the Ethernet interface is enabled ports FECH and FECL are configured for peripheral use as FEC MII signals. FECI2C bits 2 and 3 are configured as MII management lines. Since the PHY requires 167ms stabilization time after power is applied, the FEC and PHY initialization is delayed by 200ms.

10. Cache

Open.

11. Conclusion

This document is in progress and has not been officially released.

Modifications:
- V0.01 25.5.2010 – provisional version for the documentation page for development monitoring