Application Note 174

Changing clocks on the ARM1136JF-S Core Module

Released on: 21 March, 2007

ARM®

DAI0174A

Application Note 174
Changing clocks on the ARM1136JF-S Core Module

Release Information

The following changes have been made to this application note.

Table 1 Change history

Date	Issue	Change
March 2007	A	First release

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

1 Introduction

Many customers want to increase the performance of ARM development boards to be nearer their final ASIC. For example, the ARM1136JF-S core was designed to run at 330-550MHz, but the test chip on our boards runs at a default 240MHz. Please note that we cannot guarantee the maximum operating frequency of our boards, there will be some variation between boards, and you may only be able to achieve a relatively small increase, for example to 285MHz.

Some customers want to reduce clock speeds for benchmarking or to prototype their hardware in an FPGA. The minimum AHB bus frequency is 25MHz, due to the delay locked loop (DLL) used in the FPGA.

Unfortunately, the CM1136JF-S user guide does not include all the details you need to change the clocks. This application note describes the clock circuits, gives step-by-step instructions how to change the frequencies, and includes two programs to set and measure the clocks. The calculator program assumes you have an Integrator/CP baseboard; it does not work with an Integrator/AP or no baseboard.

1.1 Overview

There are three clocks locked together by programmable divide ratios: core CLK, internal bus HCLKI, and external bus HCLKE. They each have a maximum operating frequency that will vary for different boards and test chips. Finding the maximum operating frequencies is an iterative process: select some frequencies to test, work out the register settings you need, set the registers, run a test program to check the board is stable at the frequencies chosen, then repeat with increasing frequencies.

For best performance you should try and get the highest core frequency CLK, since this is also the speed of the L1 cache, and caches have a large impact on performance. External bus frequency HCLKE is usually less important unless your application is doing a lot of DMA or I/O accesses.

HCLKI should be run as fast as possible, at CLK / 2, to avoid AHB components on the test chip adding unnecessary delays. This assumes you do not want to access the SRAM on the test chip, which may require a slower speed.

When you have found the best operating frequencies for your board, you should run a 'soak test' for several hours to confirm that the board is stable with these settings. You should do this with the ambient temperature a few degrees higher than normal room temperature. This is because temperature and voltage variations cause silicon timing to change. We recommend using the Dhrystone program provided in the RVDS Examples directory to test stable operation. You should also use the Cached Dhrystone example, since memory cells (in this case the L1 cache) tend to push the silicon design rules more than processors.

2 Hardware

2.1 Core Module

Figure 1 shows the clock signals on the CM1136JF-S Core Module. The clock circuit inside the ARM1136JF-S test chip is described in more detail later.

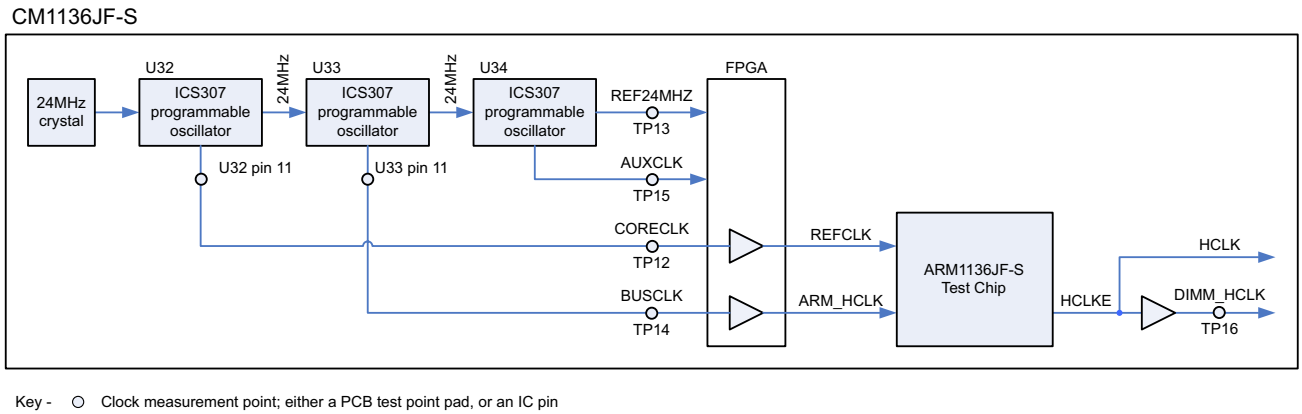

Figure 1 Clock signals on the CM1136 Core Module

The clock output from the test chip, HCLKE, is used as the timing reference for most of the Core Module. It is used for the local AHB bus, SDRAM, SSRAM, PLD and peripherals in the FPGA. If the Core Module is fitted on top of a CP baseboard then it is also used for the AHB system bus between boards.

HCLKE is fed into the CM1136 FPGA which uses a delay-locked loop (DLL) to distribute the clock internally. The DLL has a minimum rated operating frequency of 25MHz. This lower limit may be a problem if you are using a CP baseboard (so it applies to the system bus) and are prototyping hardware in a Logic Tile FPGA.

2.2 Test Chip

Figure 2 on page 5 shows the clock circuit inside the ARM1136JF-S test chip. The default clock frequencies, divider and multiplexer settings for a CM1136 fitted on top of an Integrator/CP are shown.

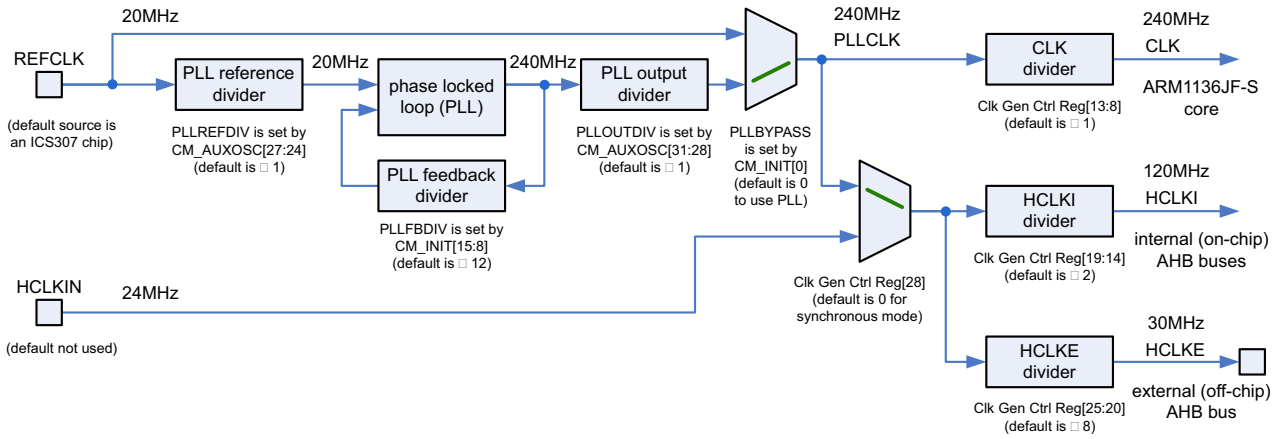

Figure 2 Clock circuit inside the ARM1136 test chip

The test chip clock circuit has three parts: an analogue Phase Locked Loop (PLL) surrounded by 3 dividers that can generate arbitrary frequencies from the incoming REFCLK signal; multiplexers to bypass the PLL or select an alternative clock source for asynchronous mode; and 3 dividers for the different clock domains on and off the test chip.

2.3 Default Frequencies

Table 2 lists default clock frequencies for the CM1136 when used with the CP and AP baseboards, and on its own. These defaults apply to boards with the latest FPGA image, Rev B build 3 / Rev D build 7. There should be a label on the CM1136 FPGA stating the version it was shipped with.

The 'Rev B' image is used when the CM1136 is fitted on top of an AP baseboard or without a baseboard; and the 'Rev D' image applies to CM1136 with a CP baseboard. Note that the Rev D image has been developed further since the AP is now obsolete.

Table 2 Default clock frequencies for the CM1136

Signal	Description	Measure at	CM1136 + CP	CM1136 + AP or stand-alone
REFCLK	Input to test chip PLL	U32 pin 11	20MHz	180MHz
ARM_HCLK	Input to test chip for asynchronous mode	U33 pin 11	24MHz	16MHz
CLK	ARM1136 core clock	Run test program	240MHz	180MHz
HCLKI	Internal (on-chip) AHB clock	Cannot measure	120MHz	90MHz
HCLKE	External (off-chip) AHB clock	Test point TP16	30MHz	36MHz
AUXCLK	Auxiliary clock (CLCD pixel clock for CP)	Test point TP15	25MHz	~97MHz

2.4 Asynchronous Mode

The ARM1136 bus interface ports can operate either synchronously or asynchronously to the core clock, enabling the choice of core and bus clock frequencies. To keep things simple we will not discuss how to use asynchronous mode here. You might want to use it on your ASIC design if:

- the different ARM1136 ports are connected to AHB buses with asynchronous frequencies
- it is difficult to synchronise the core clock with the AHB clock across the whole chip
- the AHB clock is fixed at say 133MHz for the SDRAM and you are trying to run the core as fast as possible, say 633MHz
- the AHB clock is fixed and you have multiple masters each with a different core frequency
- the AHB clock is fixed and you want to reduce the core clock to say 10MHz for power saving.

The additional 1-cycle latency of using an asynchronous bridge is small compared to the 5 or 6-cycle latency of a normal access. For more details, see figure 8-2 in the ARM1136 Technical Reference Manual, which compares the performance lost through synchronization penalty with the performance lost through reducing the core frequency to be an integer multiple of the bus frequency.

2.5 Constraints

Here is a list of constraints for the clock control registers. The registers are summarized in a table at the end of this document.

- CM_OSC, CM_AUXOSC and CM_INIT are locked from accidental changes by the CM_LOCK register. Write 0x0000A05F to CM_LOCK to unlock them, and any other value to lock them.
- Changes to the clock dividers for CLK, HCLKI or HCLKE take place immediately. If you are using the ARM1136JF-S in asynchronous mode (CPU asynchronous to bus ports), you must make sure changes to these dividers propagate before selecting the asynchronous ratio.
- HCLKI must be an integer multiple of HCLKE. In ARM1136JF-S synchronous mode (the default), CLK must be an integer multiple of HCLKI greater than 1.
- Changes to PLLREFDIV, PLLFBDIV and PLLOUTDIV take place after a soft reset. That is, when you press the CM1136 reset button S1, or when you set the RESET bit (bit 3) in the CM_CTRL register. Reads from the CM_AUXOSC register (PLLOUTDIV and PLLREFDIV control bits) will not show the updated values until a soft reset has occurred.
- When the Core Module is mounted on an AP baseboard or standalone, writes to CM_OSC take effect immediately. When the Core Module is mounted on an Integrator/CP baseboard (which uses a different FPGA image) you must perform a soft reset for changes in CM_OSC to take effect.
- When mounted on a CP baseboard there is an undocumented feature to change clocks without a soft reset. Setting bit 26 of the CM_OSC register forces the CM_INIT, CM_OSC and CM_AUX registers to immediately update the hardware. To prevent the test chip PLL and FPGA DLL losing lock you must keep the frequency changes small. The DLL is more sensitive, its input can accept a change of up to 1ns period (where period = 1 / frequency) without losing lock, for example changing HCLKE from 30 to 30.9MHz.

- All divider ratios are (register value + 1). For example, a PLLOUTDIV value of 0 means divide by 1, a value of 1 means divide by 2.
- You must preserve the original values of bits you are not writing to (read-modify-write the register).

2.6 Clock control registers

Table 2 shows the Core Module and test chip clock registers that control clocks. You must preserve the original values of bits that you are not writing to.

Table 3 Clock control registers

Register	Address	Bits	Parameter	Description
CM_OSC[a]	0x10000008	26	UPDATE	Set to 1 update all ICS307 chips with new register settings. If you change HCLKE in steps greater than 1ns period the FPGA's delay-locked loop (DLL) may lose lock
CM_OSC[a]	0x10000008	22:15	BUS_VDW	Generates ARM_HCLK using ICS307 chip U33. Only needed in async mode
CM_OSC[a]	0x10000008	14:12	BUS_OD	Generates ARM_HCLK using ICS307 chip U33. Only needed in async mode
CM_OSC[a]	0x10000008	7:0	PLL_VDW	Generates REFCLK using ICS307 chip U32. The REFCLK frequency in MHz is PLL_VDW + 8. REFCLK can be in the range 16 to 180MHz
CM_CTRL	0x1000000C	3	RESET	Set to 1 to do a soft reset and use the new clock register values. Same effect as pressing the RESET button S1
CM_CTRL	0x1000000C	2	REMAP	Set to 1 to map SDRAM at address 0x0 instead of flash (the default)
CM_LOCK	0x10000014	15:0	LOCKVAL	Set to 0xA05F to unlock the protected control registers CM_INIT, CM_OSC and CM_AUXOSC
CM_AUXOSC[a]	0x1000000C	31:28	PLLOUTDIV	Test chip PLL output divider. Maximum value is 0x7. The divide ratio is PLLOUTDIV + 1
CM_AUXOSC[a]	0x1000000C	27:24	PLLREFDIV	Test chip PLL reference input divider. The divide ratio is PLLREFDIV + 1
CM_INIT[a]	0x10000024	15:8	PLLFBDIV	Test chip PLL feedback loop divider. Maximum value is 0x3F. The divide ratio is PLLFBDIV + 1
CM_INIT[a]	0x10000024	0	PLLBYPASS	Set to 0 to use the test chip PLL, or 1 to bypass the PLL
ClkGenCtrlReg[b]	0x3F200080	31:26	Various	Set to 0x0 for ARM1136 ports in sync mode and PLL not bypassed

Table 3 Clock control registers (continued)

Register	Address	Bits	Parameter	Description
ClkGenCtrlReg[b]	0x3F200080	25:20	HCLKEDIV[c]	Divides the PLL output (or REFCLK if PLL bypassed, or ARM_HCLK in async mode) to generate HCLKE. Maximum value is 0x3E. The divide ratio is HCLKE_DIV + 1
ClkGenCtrlReg[b]	0x3F200080	19:14	HCLKEDIV[c]	Divides the PLL output (or REFCLK if PLL bypassed, or ARM_HCLK in async mode) to generate HCLKI. Maximum value is 0x3E. The divide ratio is HCLKI_DIV + 1
ClkGenCtrlReg[b]	0x3F200080	13:8	CLKDIV[d]	Divides the PLL output (or REFCLK if PLL bypassed) to make the ARM1136 core clock. Maximum value is 0x3E. The divide ratio is CLK_DIV + 1

a. Locked from accidental changes by the CM_LOCK register.
b. Clock Generator Control Register. The change takes place immediately; other register changes take place on a soft reset.
c. The HCLKI frequency must be an integer multiple of the HCLKE frequency.
d. In synchronous mode, the core frequency CLK must be an integer multiple greater than 1 of HCLKI.

3 Measuring the Frequency

If you manually set the registers you should measure the resulting frequencies to be sure you have set them correctly. You can use an oscilloscope to measure the input to the test chip (REFCLK at U32 pin 11) and the output (HCLKE at test point TP16).

Figure 3 shows the test points for clocks on the CM1136. It also shows the connectors and LEDs mentioned elsewhere.

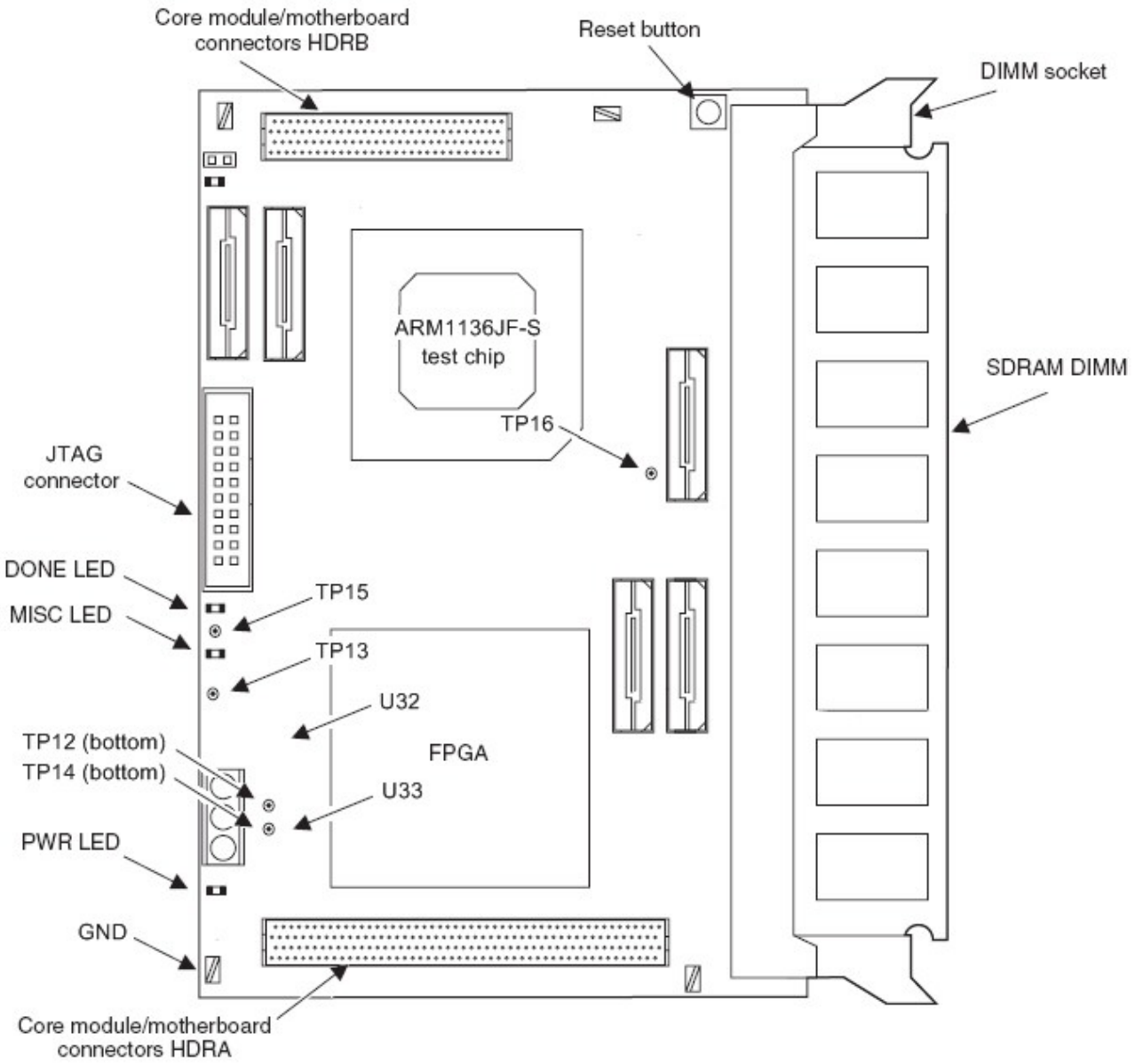

Figure 3 CM1136 test points for clocks

A utility is provided with this application note so you can measure the ARM1136 core frequency. The utility comprises two stand-alone programs called InitTCM.axf and Speedtest.axf. Tightly Coupled Memory (TCM) on the ARM1136 test chip runs at the core frequency with zero wait states; the utility measures how long it takes for the processor to run around a loop in TCM. The steps to use the utility are:

1. Use your debugger to load and run InitTCM.axf, then stop execution. This program initializes the TCM. It only needs to be run once per debug session, unless the TCM settings are subsequently changed by another program.

2. Use your debugger to load and run Speedtest.axf, then stop execution. This program was linked to be loaded by the debugger into TCM at address 0x80000000. Speedtest executes 1,000,000 loops that each take 20 core clock cycles on an ARM11. It uses a hardware timer clocked at 24MHz to measure how long it takes. The final count value is left in register R3.

3. Use the value in R3 to calculate the core frequency. For example:

   R3 = 0x001E8486 = 2,000,006 counts

   1 count = 1 / 24MHz = 41.67ns

   Total program execution time = 2,000,006 x 41.67ns = 83.33ms

   Core frequency = total cycles / total time= 20,000,000 / 83.33ms = 240MHz

4 Instructions to set the clocks

You will need:

- An Integrator/CM1136JF-S Core Module with a test chip with a working PLL and the latest FPGA version: Check that the test chip is not marked "0344" (for details please see our FAQ 'Problems with CM1136JF-S test chip internal PLL and SRAMs' `http://www.arm.com/support/faqdev/5303.html`). Check that the label on the FPGA says "Rev D build 7" (Rev D is for the CM1136 with a CP baseboard; ignore the "Rev B" build which applies to CM1136 with an AP or no baseboard).
- An Integrator/CP baseboard and its 12V DC power supply. These instructions do not work for CM1136 without a baseboard or on an Integrator/AP baseboard.
- Debugger software and JTAG run control hardware that supports ARM 'semihosting'. These instructions assume you have a RealView ICE (RVI) and RealView Debugger (RVD) version 3.0, although earlier versions are similar.

The steps are:

1. To assemble the CM1136 on top of CP: Check that the SDRAM DIMM is connected to the CM1136. Check the holes on the HDRA and HDRB connectors on CM1136 have no obstructions and that the pins on the HDRA and HDRB connectors on Integrator/CP are not bent. On a level surface, carefully line up the HDRA and HDRB connectors then push down evenly.
2. Connect RVI to the JTAG connector on the CM1136.
3. Connect the supplied 12V DC power supply to the Integrator/CP baseboard. Turn on the mains power and the red STANDBY LED D12 should light on the CP.
4. Press the POWER button **S1** (near the STANDBY LED) to power up the boards. On the CP you should see the 3V3 and 5V LEDs lit and the alphanumeric LED display should say 'CP'. On the CM1136 you should see the PWR, DONE and MISC LEDs lit. When the boot monitor runs it configures the SDRAM.
5. 5.Run RVD and connect to the ARM1136 core:
   a. In the RVDEBUG window select **Target → Connect to Target…**
   b. In the Connection Control window that pops up, right-click on RealView-ICE and select **Configure…**
   c. In the RVConfig window that pops up, click on the **Auto Configure Scan Chain** button. It should detect two devices, ETMBUF and ARM1136JF-S.
   d. Select **File → Save** then **File → Exit**.
   e. In the Connection Control window, select the RealView-ICE line, click on the **Open Target Access** button, then select the line that appears underneath RealView-ICE and click on the **Connect** button.
   Note that you do not have to do steps b, c, d the next time you connect to this target.
6. In the RVD window (labeled **RVDEBUG**) locate the memory pane and enter address 0x10000000 to display the Core Module control registers.
7. The calculator program CM1136_clocks.axf calculates the register values needed for a given ARM core frequency, then sets the registers to change the frequency. Load and run the CM1136_clocks.axf program in RVD: select **Target → Load Image** then **Debug → Run**.

8. The program prompts you to enter a target ARM1136 core frequency in the range 100 to 340MHz. You can enter a number such as 233 or 233.33 but do not enter 'MHz'. The program will calculate the new register settings and prompt you to accept them. Enter 'Y' and the program will write the new values and initiate a soft reset to make the values take effect.

   a. If you are increasing the core frequency:

      After the soft reset, flash is mapped to the bottom of memory instead of SDRAM. Use the RVD memory pane to write 0x4 to address 0x1000000C. This 're-maps' SDRAM to address zero so that you can load and run programs normally again.

      If you cannot write the value to memory, or cannot load a program with RVD, then the core frequency you chose is too fast. Disconnect RVD (select **Target → Disconnect**), power-down the system by pressing the POWER button **S1**, then go back to step 4 and enter a lower target frequency.

   b. b)If you are reducing the core frequency:

      It may not be possible to reduce the core frequency to the value you entered in one step, while keeping the HCLKE frequency within its normal operating range. If the program tells you this is the case, run the program again and enter a core frequency that is not as low, and repeat until you get to the frequency you want.

      As it finishes, the program may prompt you to write the HCLKE divider value into the test chip register at address 0x3F200080. The program cannot do this itself because it must occur after the soft reset, which stops the program.

      After the soft reset, flash is mapped to the bottom of memory instead of SDRAM. Use the RVD memory pane to write 0x4 to address 0x1000000C. This 're-maps' SDRAM to address zero so that you can load and run programs normally again.

9. You can measure the clock frequencies to confirm that your changes have taken effect as planned:

   You can measure REFCLK at U32 pin 11 on the CM1136.

   You can measure HCLKE at test point TP16 on the CM1136.

   You can measure the ARM1136 core frequency using the Speedtest program.

10. Now run a program to check the CM1136 is stable in this configuration. While searching for the maximum operating frequency you only need to run the test program for a minute to see if it crashes. You can use the Dhrystone program in the RVDS Examples directory for this; with the default CM1136 settings enter a value of 600,000 runs to give about 1 minute.

11. If the test program crashes then the target core frequency you chose is too fast. Disconnect RVD (select **Target → Disconnect**), power-down the system by pressing the POWER button **S1**, then go back to step 4 and enter a lower target frequency.

12. If the test program does not crash and you have not yet found the highest stable frequency, then go back to step 7 and enter a higher target frequency.

13. When you have found the highest stable frequency for your CM1136, run a 'soak test' for several hours to ensure the hardware is stable for software development. Ideally you should do this with a higher ambient temperature than normal room temperature, in which case you may need to use a frequency 1 or 2MHz slower. Run the Dhrystone program to test the core and external bus. Then run the Cached Dhrystone program to test the core and cache. If these programs do not crash then the hardware is stable at the core frequency you entered. For each program, time how long it takes for 600,000 runs and extrapolate for the time you want to run your soak test.

14. In development you can use the calculator program to initialize the CM1136 to the operating frequency you need.

5 Software

The calculator program is fairly straightforward: it prompts you to enter a target core frequency, it searches all combinations to find a core frequency closest to the target, then it searches for a suitable external bus frequency, and finally it sets the registers and initiates a soft reset.

If you do not wish to use the program then replace step 7 above with the following method:

1. Using the test chip diagram above, determine the REFCLK frequency and divider values that you want to use. See for constraints on the range of values.

2. Unlock the protected registers by writing 0x0000A05F to the CM_LOCK register at address 0x10000014.

3. Enter the new register values you want to change, see Table 2 for the bit fields and addresses. You must preserve the original values of bits you are not writing to.

   The first time you try this just write to one register, for example change the value of CM_OSC at address 0x10000008 from the default 0x010B100C to 0x010B100B. This should reduce the REFCLK frequency input to the test chip from 20MHz to 19MHz, which you can measure at U32 pin 11.

   Note that changes to the Clock Control Generator Register (CLKDIV, HCLKIDIV, HCLKEDIV) take place immediately, while changes to the ICS307 chips and the test chip PLL take place after a soft reset. You must update the registers in the correct order so that clock frequencies remain within operational values. For example, increasing REFCLK to increase the ARM1136 core frequency also increases HCLKI and HCLKE. If HCLKE exceeds 38MHz the external memory may fail, so you must select a larger HCLKEDIV value before doing the soft reset to update REFCLK.

4. Generate a soft reset by pressing the RESET button S1 on the Core Module, or write a '1' to the RESET bit, which is bit 3 of the CM_CTRL register (just write 0x8 to address 0x1000000C). You will see the MISC LED turn off. RVD should trap the reset vector and stop execution at address 0x0.

