

HOWTO: Overclock C2Q (Quads) and C2D (Duals) - Guide v1.6.1

Edited on 09-May-2008: Guide is now version 1.6.1 – minor corrects made and slight re-ordering of sections.

Before you continue, I wrote this guide with the newbie in mind, so please don't reply criticizing it for being too simplistic -- it's this way by design. Also know that the steps for overclocking apply to all chips: quads, duals, single-cores, or triple-core processors. You can use the basics taught in this guide with any modern machine.

I wrote the guide originally using a Q6600/Asus P5B-Deluxe, but recently sold that machine and upgraded to an X3360/DFI LT P35-T2R. I didn't want to change the first half of the guide, so it's still based on the Q6600/Asus board. The newly written section about finding a minimum stable CPU and MB vcore section is written based on my actual experience finding stable settings for this newer machine. **Again, the steps for overclocking are pretty independent of this subtle change.** Finally, I take no responsibility for what you do with the information in this guide. Overclock your hardware at your own risk.

Overlocking Basics

Before you start, read your motherboard manual. Know how to reset your BIOS in the event that you are too aggressive in your CPU settings and it doesn't complete a POST (Power On Self Test, that beep when you first turn the machine on and it starts up means you passed the POST). Some motherboards reset automatically if you switch off the power supply for 30 seconds or so. Others require you to move a jumper to reset them.

The basic formula you need to know for CPU speed is:

Code :

```
. CPU Speed = CPUM x FSB  
. where CPUM is the CPU Multiplier, and FSB is the front side  
  bus.
```

Example: The Q6600 runs at a factory setting of 2.40 GHz. That's the product of a 9x multiplier and a 266 MHz FSB (quad pumped it's 1066 MHz but we're not quad pumping these numbers). So CPU Speed = 9 x 266 which is 2,394 MHz or 2.40 GHz.

Below is a list of Intel chips. Most of them, including the Q6600, have a "locked" multiplier – meaning it can't go above a certain value (9x in this case). The only way to increase the CPU speed beyond the stock value is by raising the FSB. Other "Extreme" chips like the QX9650 or X6850 have "unlocked" multipliers; you can raise their multipliers above the stock value. These chips are denoted from the standard stock by the letter "X" in their model number.

For reference, here are all Intel offerings as of May/'08:

Quote :

Quads:

QX9775: $8.0 \times 400 = 3.20$ GHz

QX9775: $8.0 \times 400 = 3.20$ GHz

Q9300: $7.5 \times 333 = 2.50$ GHz

Q9450: $8.0 \times 333 = 2.67$ GHz

Q9550: $8.5 \times 333 = 2.83$ GHz

QX9650/QX6850: $9.0 \times 333 = 3.00$ GHz

Q6600: $9.0 \times 266 = 2.40$ GHz

QX/Q6700: $10.0 \times 266 = 2.67$ GHz

QX6800: $11.0 \times 266 = 2.93$ GHz

Duals:

E6540/50: $7.0 \times 333 = 2.33$ GHz

E6750: $8.0 \times 333 = 2.67$ GHz

E8190/8200: $8.0 \times 333 = 2.67$ GHz

E6850/8400: $9.0 \times 333 = 3.00$ GHz

E8500: $9.5 \times 333 = 3.16$ GHz

E8300: $8 \times 333 = 2.83$ GHz

E6400/20: $8.0 \times 266 = 2.13$ GHz

E6600: $9.0 \times 266 = 2.40$ GHz

E7200: $9.5 \times 266 = 2.53$ GHz

E6700: $10.0 \times 266 = 2.67$ GHz

X6800: $11.0 \times 266 = 2.93$ GHz

E2140: $8.0 \times 200 = 1.60$ GHz

E2160/E4300: $9.0 \times 200 = 1.80$ GHz

E6300/20: $7.0 \times 200 = 1.86$ GHz

E2180/E4400: $10.0 \times 200 = 2.00$ GHz

E2200/E4500: $11.0 \times 200 = 2.20$ GHz

E2220/E4600: $12.0 \times 200 = 2.40$ GHz

E4700: $13.0 \times 200 = 2.60$ GHz

X7800: $13.0 \times 200 = 2.60$ GHz

X7900: $14.0 \times 200 = 2.80$ GHz

My first quad system was based on a Q6600 at $9 \times 333 = 3.00$ GHz (25 % over factory). I found the max it can go when cooled with air is $9 \times 370 = 3.33$ GHz (39 % over factory), but it just ran too hot for me. Every chip is different... you might be an unlucky owner of a chip that just doesn't overclock very high at all.

Overclocking is more complicated than just adjusting two settings in the BIOS, because as you increase the FSB, you'll also need to increase the core voltage (vcore) which is the actual juice going to the processor. As well, you may have to increase the other voltages on the board like: memory, FSB, NB, SB, ICH chipset. There are also parameters controlling your memory that may need tweaking as well. Don't worry about them for now. The board can manage these automatically which is what you should do initially. When you finally decide on an overclock number, you'll want to go back and minimize your voltages to minimize your heat production. We'll get into this later. For now, you want to verify you can successfully POST, and verify that your system can run stable at the settings you've selected.

Pre-Overclocking Checklist

Before you think about overclocking your system, you'll need to be sure you're using quality parts that can handle the increased stresses.

1. Motherboard

I decided not to maintain a list of motherboards that are known to be good overclockers; keeping the list updated would be too time consuming. I only mention this because if you're using some generic MB you got free with the purchase of your CPU, you're probably not going to be able to overclock it.

2. Cooling

Cooling is very important since you're asking the system to produce more heat than it's designed to produce. A quad core chip will produce twice the heat of a dual core chip, so if you're using the Intel Stock HSF, you'll probably want to upgrade to something better. Again, I don't wanna maintain a list. I can tell you that I am using a Thermalright Ultra-120 Extreme and am very happy with it.

[Here](#) is a more recent list of HS's that have actually been reviewed and ranked based on performance.

Finally, there is a section at the end of the guide entitled, "Temperature Management" which I would strongly suggest you at least have a look at since it contains some good info. For example, for under \$5 you can probably shave off ~10-15 % of your NB (North Bridge) load temps simply by adding a small fan to the heatsink even if it was never designed to have one (attach it with a zip tie):



3. Memory

You will need memory that can keep up with your overclocked system. Again, I'm not going to keep a list. You'll see RAM listed with timings and speeds that I'll decode for you using the following examples:

Quote :

DDR2-800 (PC2-6400) 4-4-4-12
DDR2-1066 (PC2-8500) 5-5-5-15

- The first part is self-explanatory (DDR2

memory).

- The number after it is the data transfer rate. Simply divide it by 2 to get the maximum FSB speed for which the module is rated. Example: $800/2 = 400$ MHz. Therefore, DDR2-800 can work on systems with a FSB of up to 400 MHz (anything more and you're lucky).
- The PC2-XXXX is designation denoting theoretical bandwidth in MB/s. Some memory manufactures use this instead of the DDR2-xxx designation. You can calculate it for any FSB you want by simply taking the FSB and multiplying by 16 (rounded in some cases). Example using a 400 MHz FSB: $400 \times 16 = 6400$. So you'd need at least PC2-6400 to run on a FSB of 400 MHz.

The numbers after that are the main timings (clock cycles). In general, the lower these numbers are, the faster the memory. For more on memory timings, see [this page](#).

Quote :

DDR3-1333 (PC3-10666) 9-9-9-24
DDR3-1600 (PC3-12800) 7-7-7-20

- The first part is self-explanatory (DDR3 memory).
- The number after it is the data transfer rate. Simply divide it by 4 to get the maximum FSB speed for which the module is rated. Example: $1600/4 = 400$ MHz. Therefore, DDR3-1600 can work on systems with a FSB of up to 400 MHz (anything

more and you're lucky).

- The PC3-XXXXX is designation denoting theoretical bandwidth in MB/s. Some memory manufactures use this instead of the DDR3-xxxx designation. You can calculate it for any FSB you want by simply taking the FSB and multiplying by 32 (rounded in some cases). Example using a 400 MHz FSB: $400 \times 32 = 12800$. So you'd need at least PC3-12800 to run on FSB of 400 MHz.

4. Power Supply

There are really two major factors to consider when selecting a power supply:

- 1) Quality of the PSU
- 2) Power output

I don't have the expertise to write up this selection of the guide, so I'll point you to this nice list written by [perkam](#) to use as a guide. More recently, [TH.com](#) wrote another article you can check out on the topic.

There is a great article on power consumption over at [TH.com](#) that I suggest you read at your leisure. I distilled out some highlights to underscore how much power systems really use:

Code :

Component	Best Case	Worst Case
Power Supply	5-15 W	40-60 W
Motherboard	10-15 W	30-50 W
Processor	12-30 W	60-120 W
RAM	5-15 W	30-50 W
HDD	3-5 W (2.5")	10-15 W (3.5")
GFX Card	3-10 W (on MB)	25-180 W (PCI Express)
Total	38-90 W	195-475 W

So you can see that depending on the hardware specs, your system power requirements can approach 500 W.

There are also a number of good online power supply calculators you can use. Find them with google as always. [Here](#) is one such example.

5. Required Software

Here are few utilities you'll need to continue, all are freeware.

General System Info

[CPU-Z](#) is a great app to display your current settings including vcore, FSB, multiplier, RAM settings, etc. This one is a must-have.

CPU Stress Testing

[Prime 95 v25.6](#) is a great app for stress testing. It is very efficient at generating CPU loads equally across all your cores. There are few other apps that will stress a system as hard as p95. Alternatively, if you're using a 64-bit o/s you can download the [64-bit version of prime95 v25.6](#).

I like to use version 25.x over the current "production" version 24.x because it [version 25.x] automatically stresses all your cores without having to load up two different instances of the app like you had to do with [orthos](#).

System Monitoring

There are several options for processor core temp and system temp monitoring. For a discussion of what is different between the apps I am about to list, see [this thread](#). PLEASE READ THAT THREAD BEFORE ASKING QUESTIONS ABOUT WHY SOME OF THESE READ DIFFERENT TEMPS!

These first two will give you just the core temperatures (not system temps, voltages, etc.):

[Core Temp](#) (freeware)

[Real Temp](#) (freeware)

The next three will give you core temps plus many other temps, voltages, fan RPMs, etc.:

[HWMonitor](#) (freeware)

[Speedfan](#) (freeware)

[Everest Ultimate](#) (shareware \$\$\$)

Memory Testing (optional but can help rule-out bad memory)

[Memtest86+](#) is a great piece of software that will test your memory. Download the bootable ISO and allow it to test your system for 6-24 h. I ran it on my machine for a little over 6h 17m with no errors:

```
Memtest86+ v2.01 | Pass 63% #####
Intel Core 2 3000 MHz | Test 31% #####
L1 Cache: 64K 49181 MB/s | Test #7 [Random number sequence]
L2 Cache: 4096K 20979 MB/s | Testing: 128K - 2048M 4095M
Memory : 4095M 4043 MB/s | Pattern: d68d6456
Chipset : Intel P965/G965 - FSB : 333 MHz - Type : DDR-II
Settings: RAM : 333 MHz (DDR666) / CAS : 4-4-4-12 / Dual Channel

WallTime  Cached  RsvdMem  MemMap  Cache  ECC  Test  Pass  Errors  ECC  Errs
-----
6:17:58  4095M    512K  e820-Std  on  off  Std    8    0

Pass complete, no errors, press Esc to exit
```

Note that you can't just run the memory tests without first setting up your memory in your BIOS even though they might be auto configured. This usually entails the user to specify the timings, voltage, and fsb:dram divider *prior* to running the tests since the memory might be stable at one given set of conditions (maybe 5-5-5-15 @ 667 MHz is stable), not unstable under another (maybe 5-5-5-15 @ 1066 MHz gives errors, but you only



tested the 667 MHz level)!

Do NOT trust the temperatures that your motherboard's free temp utility reads. "PC Probe 2" that comes with Asus boards really sucks because it's not measuring your core temps. They are what you really care about.

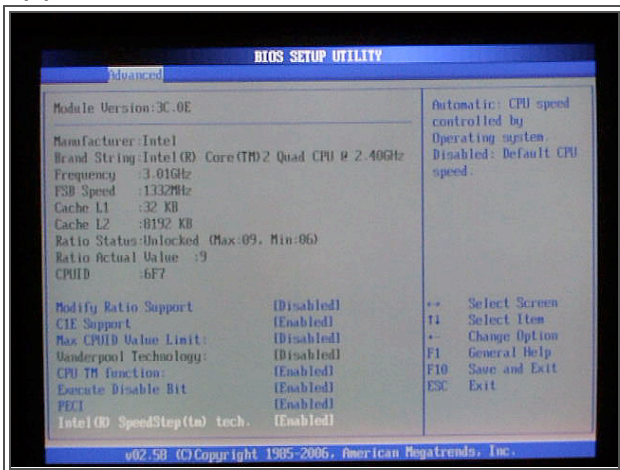
There are other temp monitoring programs out there. These are what I recommend.... I'll only mention one other by name with the advice that you do NOT use it: TAT (thermal analysis tool). It's made by Intel and I don't care what anyone else out there thinks: it was NOT designed to read the coretemps of a C2D/C2Q chip. It was written for Pentium M chips. Yes, it will display temps, and yes, sometimes they match up with the values Coretemp/HwMonitor display, but I have found that TAT often reports temps higher than the real values. How do I know this? Read [this](#) thread and pay attention to unclweb's instructions to use crystalcpuid to directly read your DTS (digital temperature sensor) and calculate your core temp yourself if you don't believe me.

BIOS Settings

Let's look for some settings in your BIOS. Not all boards are the same. The following terms/pics are taken from my old P5B-Deluxe; other manufactures will likely have their own names for these settings. You're on your own to figure them out (shouldn't be that tough). It goes without saying that your board will have these organized differently as well.

Lemme apologize upfront for the poor quality images below. I have no idea how to effectively photograph a computer monitor. I just used a cheap p&s camera with the lights off. You can still read them.

First thing you want to do is change a few settings, I'll take them in order as they appear in the BIOS:



Modify Ratio Support - disabled, but you can if you want to select a different multiplier. For the Q6600, 9x is the highest as I said. If you enable this, you can select a lower one if you want, some people think a lower multiplier and a higher FSB is better. For example: 9x333 = 3.01 GHz and so does 8x375. In my experience, doing this can lead to faster synthetic benchmark scores, but most real-world applications usually don't go appreciably faster.

C1E - Intel's so-called enhanced halt state. Read Anandtech's blurb about it [here](#) for more.

Disable initially, enable later on and see if the system remains stable. **This is a power savings option.**

Max CPUID value limit - disable unless you're running an older O/S like Windows NT.

Vanderpool - disable unless you're running VMWare or virtualPC; this option enables additional extensions within the processor that yields added acceleration when running

multiple O/S's on the same machine through virtual machines.

CPU TM function – enable. Option affects CPU protection/throttle management to help you when you don't realize you're pushing your chip too hard.

Execute Disable Bit - enable. XP has a setting to help with virus protection and requires this set to enable.

PECI – This stands for Platform Environment Control Interface - disable or enable. This affects how your DTS (Digital Thermal Sensors) report the core temps of your CPU. I have mine enabled and have read several posts now that suggest having it enabled does indeed give more accurate core temps. I can't say if you want it on or off in your system.

According the [Asus P5B-Deluxe FAQ](#), this setting toggles between two temp modes.

Note: if you're using a real core temperature monitoring application such as coretemp (mentioned and linked above), this setting has no effect that I can see.

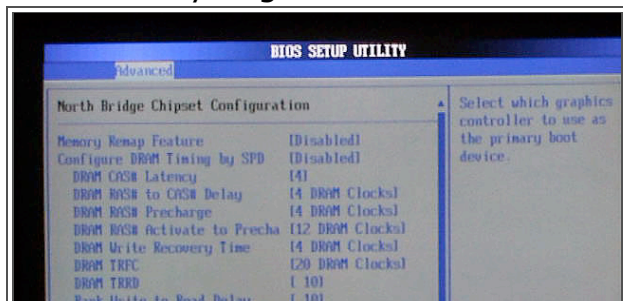
SpeedStep - Automatically lowers the multiplier from its max. (9x for the Q6600) to 6x when the machine is idle. The result is less power consumption and heat production. It goes back up to 9x when you start to get a CPU load. Disable initially, enable later on and see if the system remains stable. **This is a power savings option.**

Why do you care about power savings? Increased power consumption translates into increased heat production. As well, power costs money and unless generated from a nuclear power plant, creates carbon dioxide gas. It's true that energy savings will only matter when the machine is idle, but odds are your machine will spend most of its time at idle unless your run an app like fold@home or seti@home etc. Let's assume for the sake of discussion that enabling these saves you 10 cents / day. A few pennies per day will add up over time. Using the dime-per-day as an example for a machine running every day is roughly a savings of \$35 per year – not too shabby.

Tomshardware.com's [power savings article](#) reported a savings of 12 full watts by enabling speedstep on their test system.

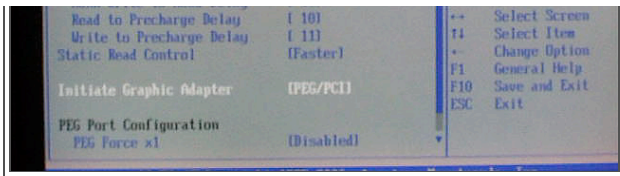
Second thing you'll want to do is dial in the manufacture's specs for your specific memory. Also take care not to exceed the design specs for your memory initially. We want to minimize the number of variables to deal with on a first time overclocking. In other words, if your machine isn't stable, you want to be sure it's due to the CPU settings, NOT the memory timings.

Where can you get the manufacture's specs? Try their website or the product packaging.



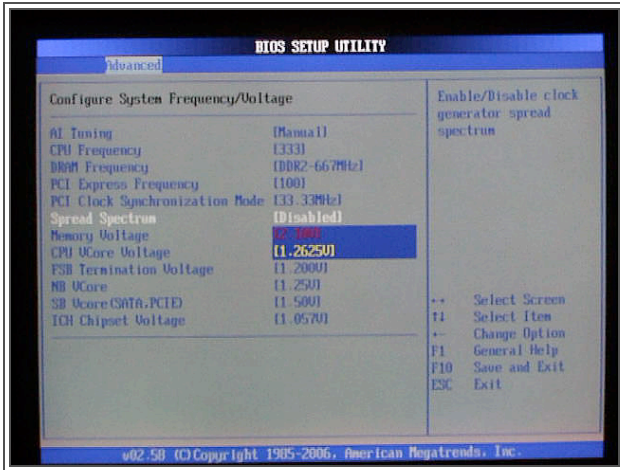
Enter in the first four timings (4-4-4-12 in my case) and don't mess with the default or auto values for the "sub timings" at this time. You can do that after you get a stable overclock.

The only other setting worth mentioning here is the so-called "memory remap feature." If you are running with more than 3 gigs of memory,



and you want to actually have the BIOS/OS see it, you'll need to enable this. Also enable this if you're running a 64-bit operating system.

Next, find the section where you can control the nuts and bolts of your system. On my P5B-Del I had to switch the AI tuning to "manual" mode to see these options:



CPU Frequency - This is the FSB in MHz. Set it to whatever you're planning to multiply by 9x (333 in my case).

DRAM Frequency - This the speed your RAM will run. Make sure you don't exceed the amount for which your specific RAM is rated.

Most good boards will offer several fsb:dram dividers. Some common ones are listed below. Assuming that you're using a 333 MHz FSB the ratios are:

Code :

- . FSB : DRAM
- . 1:1 = 333 MHz : 667 MHz
- . 4:5 = 333 MHz : 833 MHz
- . 2:3 = 333 MHz : 1,000 MHz
- . 5:8 = 333 MHz : 1,066 MHz
- . 3:5 = 333 MHz : 1,111 MHz
- . 1:2 = 333 MHz : 1,333 MHz

Now, if you're running @ a 400 MHz FSB, the ratios become:

Code :

- . FSB : DRAM
- . 1:1 = 400 MHz : 800 MHz
- . 4:5 = 400 MHz : 1,000 MHz
- . 2:3 = 400 MHz : 1,200 MHz
- . 5:8 = 400 MHz : 1,280 MHz
- . 3:5 = 400 MHz : 1,333 MHz
- . 1:2 = 400 MHz : 1,600 MHz

You can calculate these yourself with this formula:

Code :

- . DRAM Final Clockrate = (2 x FSB)/Divider

Example, 2/3 divider @ 400 MHz FSB: $(2 \times 400 \text{ MHz}) / (2/3) = 1,200 \text{ MHz}$

Running in 1:1 mode is termed, "synchronous mode." If you use a higher frequency, you're running in so-called "asynchronous mode" which offers marginal speed advantages at the price of more heat and power consumption on a C2D/C2D Quad-based system for most users. Depending on your chipset, running in an asynchronous mode may require more vcores to some of your motherboard components such as the NB, IHC, and/or FSB Termination (more on these later).

PCI Express Frequency – Set this to 100 MHz. If you don't, I believe the PCIe bus speed will increase proportionally with your FSB which is something you DON'T want to do to your expensive video board.

PCI Clock Synchronization - Use 33.33 MHz here. Again, if you leave the setting on auto, the PCI clock will creep up proportionally with your FSB which can damage cards you may have there aren't designed to run at higher frequencies.

Spread Spectrum - disable.

Memory Voltage - Read the specs for your memory. My DIMMS can use up to 2.2v. You can damage your memory if you overvolt it.

CPU VCore – **THIS IS KEY! This single BIOS setting will have the largest effect on your processor's operating temperatures!** Again, read on to the section entitled, "Stress Testing and Minimizing Your Vcores."

It needs to be enough to run stable, but not too much or else you're just wasting power and creating a ton of heat. This is particularly true with multicore processors!

In case you're wondering what Intel recommends for your processor, find your chip on [Intel's Processor Finder](#). The Q6600 is between 0.85 – 1.5V.

In my experience, a setting of "auto" ALWAYS over-estimates, but for your first boot, just leave it on auto. The next section of this guide covers stress testing whose goal is to verify stability and to minimize your vcore. For example, once you verify that you can run stable for several hours of stress testing, you'll want to come back and minimize this voltage until you become unstable again. Then simply add a little back. As you can see, my system runs stable @ 9x333 using 1.2625v.

[quote="Related topic"]Why do you care? Heat (power) increases with the square of voltage. It increases in a linear fashion with frequency. What does that mean? It means

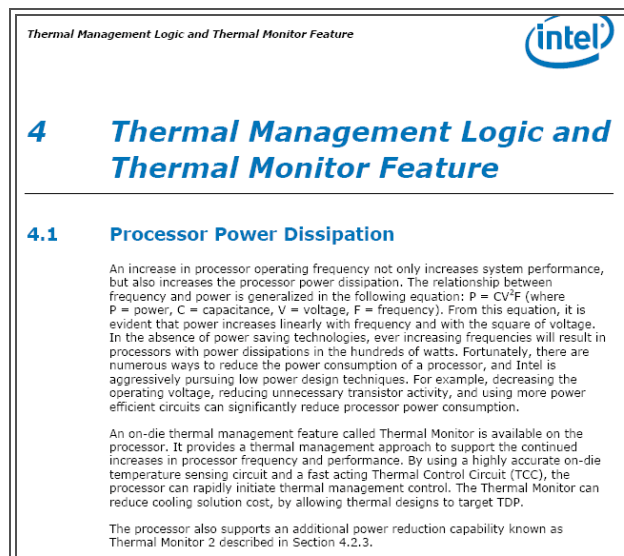
that as your FSB goes up, so does your heat, but as your vcore goes up, your heat goes up exponentially.

Quote :

An increase in processor operating frequency not only increases system performance, but also increases the processor power dissipation. The relationship between frequency and power is generalized in the following equation: $P = CFV^2$ (where P = power, C = capacitance, V = voltage, F = frequency). From this equation, it is evident that power increases linearly with frequency and with the square of voltage.

I quoted the above statement from an Intel document. It has been removed from intel.com and used to reside at the following link: [Missing Intel Document](#). I managed to find a copy of the pdf file in one of my backup sets. Knuspar from guru3d kindly agreed to host it [here](#).

The title of the document is, "Intel® Core™2 Extreme Quad-Core Processor QX6700Δ and Intel® Core™2 Quad Processor Q6000 Δ Sequence Thermal and Mechanical Design Guidelines." It's dated Jan 2007 and has an official Intel Document Number of 315594-002. I took a screenshot of section 4.1 on page 31 (where the above quote came from):



To illustrate, consider this analysis of two difference vcore settings and the temps they produce on my Q6600 @ stock settings (9x266=2.40 GHz) as well as running overlocked (9x333=3.00 GHz). The two voltages I used were 1.1375V and 1.2625V set in the BIOS that correspond to the two clock levels of 2.40 GHz and 3.00 GHz respectively. In case you're wondering, these translated into 1.112V and 1.232V in Windows as read by CPUZ.

Prime95 ran for 30 minutes and the temperatures were averaged over the last 10 minutes of those runs (well after they stabilized). Room temp was 75-76 °F. Notice that the difference in voltage is ONLY 0.120 V or 120 mV,

but this seemingly small difference brought the load temps up by an average of 6-7 °C per core!

Code :

```
. Run1 (9x266 @ 1.112 V), Average temps (°C): 51,52,50,50
. Run2 (9x266 @ 1.232 V), Average temps (°C): 57,58,57,57
. Differences (°C): +6, +6, +7, +7
```

Now if I add a faster FSB, they increased further:

Code :

```
. Run3 (9x333 @ 1.232 V), Average temps (°C): 61,61,60,60  
. Differences from lowest voltage (°C): +10, +9, +10, +10  
. Differences from same voltage (°C): +4, +3, +3, +3
```

The same thing holds true for speed in a car: $\text{energy} = 0.5mv^2$ where m is mass and v is velocity. This is the basis of the old expression, "speed kills." You generate way more energy driving 75 MPH than you do driving 55 MPH since energy and velocity have an exponential relationship. Take a 5,500 lb SUV as an example; its energy nearly doubles as a result of that mere 20 MPH increase.

Energy @ 55 MPH = 754 kJ

Energy @ 75 MPH = 1,402 kJ[/quote]

The last four voltages are also required to make a stable system. Leave them on auto for now. On my system, I lowered my chipset temps by about 4 °C by lowering them to the values you see in the pic.

As I mentioned earlier, if you're using high memory dividers (a.k.a. running your memory in asynchronous mode), you might have to manually tweak your NBvcore and your ICH vcore to get the memory to run stable. For example, my Q6600/P5B-Deluxe system required me to up the NB vcore by +2 steps and the ICH vcore had to be set to the maximum value or else I couldn't run my PC1066 memory at the higher dividers.

My X3360/LT P35-T2R system on the other hand, didn't require nearly that much extra to run in the 5:6 divider.

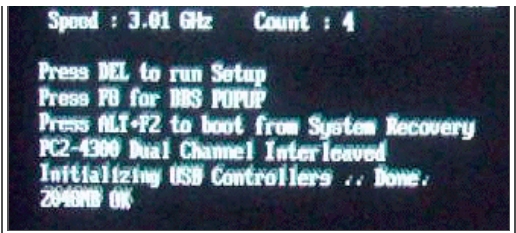
In general, the P35 chipset is better than the P965 in this regard. I have read that the X38/X48 are on par or slightly superior to the P35.

Okay, save your settings and hopefully your machine will complete the POST.

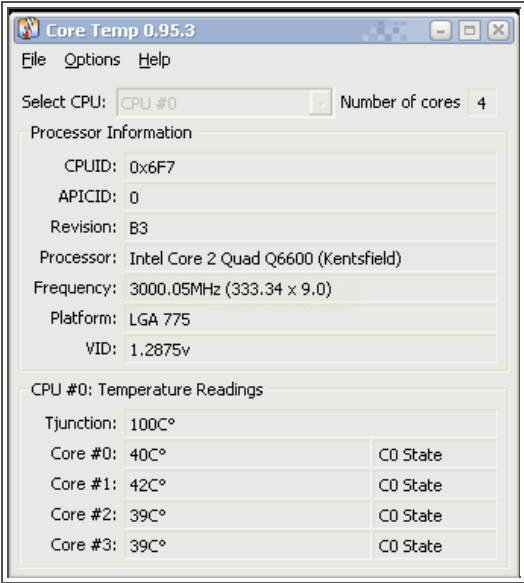


If it doesn't, and assuming you set your voltages to Auto, some common reasons are:

- Memory voltage too low
- Memory timings too aggressive
- FSB too aggressive



If you complete the POST, and make it into windows without a blue screen or reboot that's a good sign. Now on to the testing. Now that you're in Windows, load up CoreTemp or HWMonitor and have a look at your **core temps** when idle.



They should be well under 50 °C unless it's REALLY hot in your room, see the end of this document for more on how ambient temps affect your CPU load temps. There are a number of things you can do to bring down your idle and load temps. Again, see the end of this guide for some suggestions.

See [this guide](#) for more information on temperatures and temp sensors for Intel processors.

The formula for reading core temp from the DTS (digital thermal diode) is:

Code :

```
. Core Temp = tjmax - DTS
. Where DTS is the number the DTS is reporting, and tjmax is a
  constant (which differs with processor model and sometimes
  within a processor model based on its stepping)
```

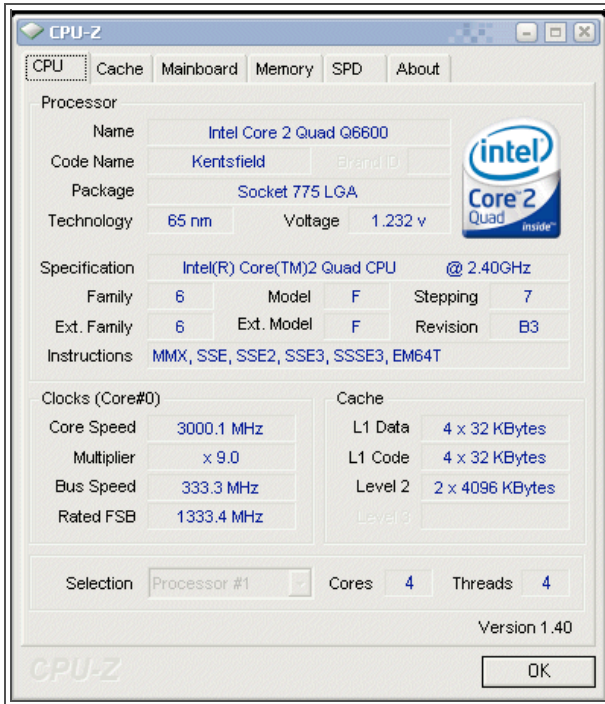
Note: **There is no official communication from Intel** as to the magnitude of tjmax for desktop/server C2D/C2Q chips! This makes calculating the "real" core temp tough since people are just guessing.

For example, a Q6600 (G0) stepping may have a tjmax of either 95 or 105 (again, these are people's best guesses). If tjmax is 105, then Core Temp = 105 - DTS. THIS DOESN'T MEAN THAT THE LIMIT FOR THE CHIP IS 105 °C! In this example, let's say the DTS value is 50. Therefore, Coretemp = 105-50 = 55 °C. If tjmax is 95, the math becomes 95-50 = 45 °C. Don't worry about doing this calculation; all the temp monitoring software will do it for you. I only mention it so you can understand what's going on.

I like to keep my core temps under 65 °C. I may be using a conservative number here, but I don't want to replace my chip anytime soon. If you don't care about the longevity of your chip, you can likely use higher numbers. I have read about people running their chips right up to the factory shutdown/auto throttle down temp. It's your chip, do what

you want.

Load up CPU-Z to see what your vcore is at idle.



You'll notice that the vcore in CPU-Z is different from the value you selected in your BIOS. This is normal and true for all boards. You'll also notice it drops again when your machine enters a load state: again, this is normal and known as vdroop; some boards/chipsets do it worse than others. If you read at the end of the guide, some boards can be modified to eliminate or greatly reduce vdroop.

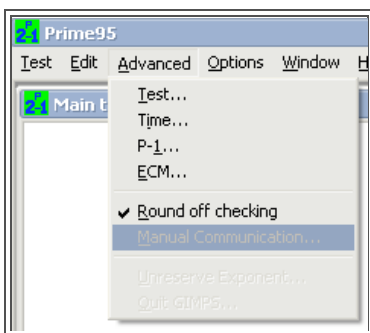
Stress Testing and Minimizing Your Vcores

The goal of stress testing is two fold:

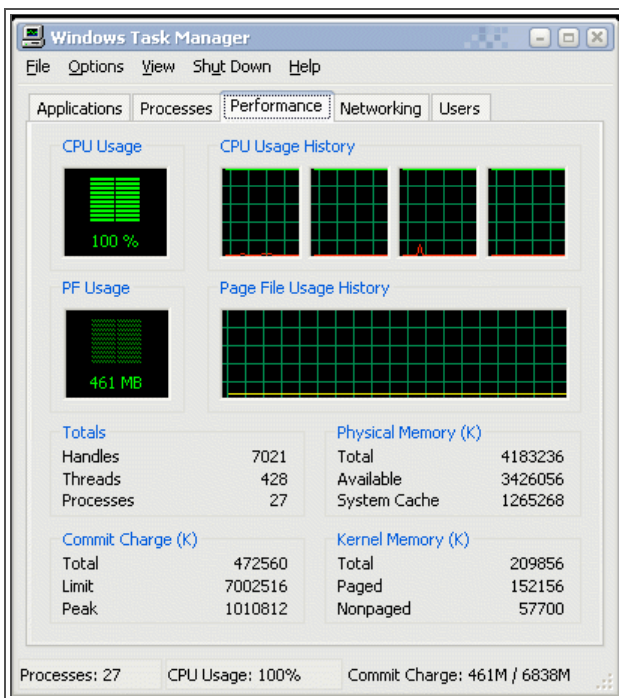
- 1) To arrive at a stress test stable system (>24 hours with no prime95 errors).
- 2) To minimize your vcores and thus minimize heat product both on your CPU but also on your NB/SB and other MB components.

Prime95 will run and every now and then it will check the values it's calculating using your processor to its internal standards since its torture testing using known values. Assuming you enable error checking, you'll be notified if your values differ indicating an instability. This is why it is **IMPERATIVE that you enable error checking within Prime95**; again, if you don't enable it, you **WILL NOT** be notified of errors!

Do so simply by going to the "Advanced" menu and enabling "Round off Checking." If the system isn't stable, it will report an error and stop stressing the core that gave the error.

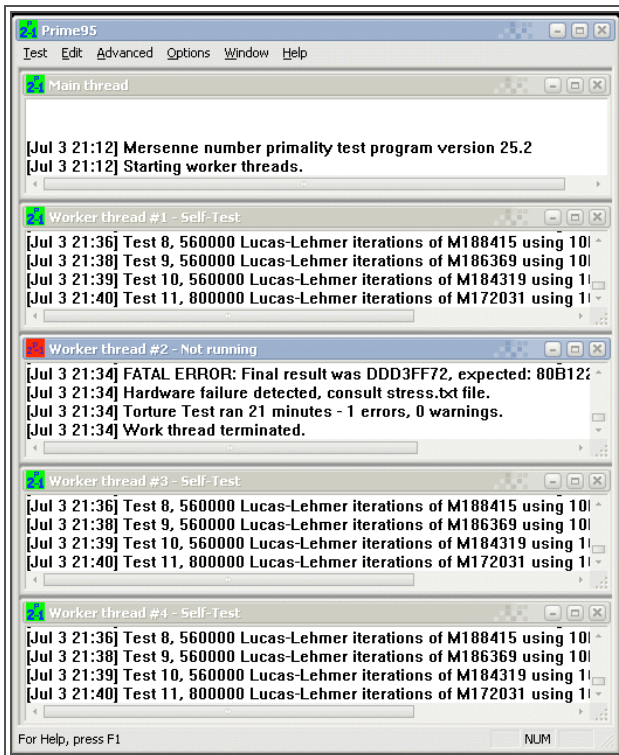


Now that you picked your operating condition (i.e. 9x333 or 8.5x400, etc.) let's stabilize the system through stressing it with prime95. Just so you get an idea what to look for, Coretemp as well as Prime95 (double-check that you enabled round off error checking) and run the Torture Test>Large FFTs. You'll wanna keep an eye on your system temps to make sure they don't exceed the redline so the chip doesn't get throttled (assuming you have thermal management enabled in your BIOS). All your cores should get stressed equally (look in the task manager to verify):

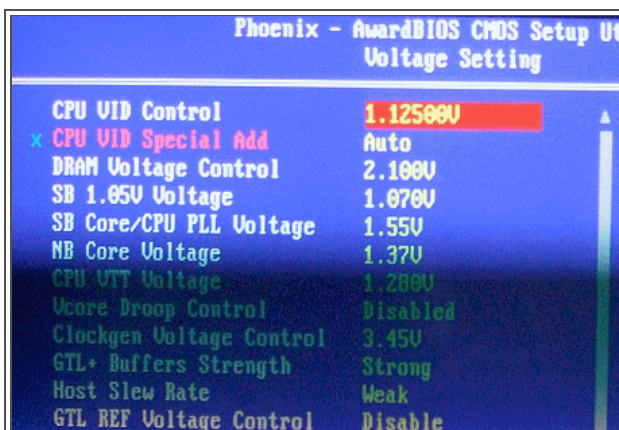


For your reference, here's what an error from within prime95 looks like:
 When/if you get an error (and you will), you'll need to either back off on the operating conditions (FSB or multiplier) or add some voltage to your vcores. Therein lies the challenge. Since you have four different vcores to select from, how do you know which one or which ones to adjust?

It's now time to minimize your vcore settings. Reboot and go into the BIOS' section where you can control your CPU and MB voltages. Remember, different motherboard will call these variables different terms. The pic below is right out of my BIOS so you can see what DFI calls them, and what they mean:



CPU VID Control – The processor vcore, I'm not sure why DFI calls it "CPU VID Control" but whatever. From here on out, I'm going to call it Vcc since technically, the term VID is an entirely different concept (see [this document](#), page 14 for more if you have an interest).
 DRAM – The memory vcore.
 SBCore – Southbridge vcore (might be called ICH in your board).
 NBCore – Northbridge vcore (might be called MCH in your board).
 VTT – Reference voltage (might be called FSB Termination voltage in your board). It's used to terminate data lines between the MCH and CPU. Some motherboards give the option for GLT reference controls. If you enable this you're adding three additional variables to the mix and making your life more complicated. Unless you're an extreme overclocker wanting to squeeze every single MHz out of your system, my advice is not to enable the GLT options. I'd also caution you not to enable this option since there is tons of misinformation out there about these undocumented features.



If you must, here a few links that might help you understand how it works and give you some starting points, but I won't be using them in this guide:

[Adjusting \[Advanced\] Gunning Transceiver Logic \(A/GTL+\) Voltage Levels for Increased Front Side Bus \(FSB\) Signaling Margins and Overclocking.](#)

x CPU GTL 1/3	REF Volt	95
x CPU GTL 0/2	REF Volt	95
x North Bridge GTL	REF Volt	95
CPU Core	Voltage	1.200
DRAM	Voltage	2.080
NB Core	Voltage	1.400

DFI UT P35-T2R: Tweakers Rejoice!

Good thread (kinda long) but good info.

There are several approaches you can use to arrive at a stable, minimized set of vcores. I recommend that you start with lower vcore values and work your way up. Lower values will fail much faster than higher values thus making the process a bit quicker for you.

To start with, select a set of vcores that are kinda low and see if you can POST. How do you know where to start? Use trial and error at this point unless you know someone else's settings to use as starting points. When in doubt, I'd recommend that you start near the bottom of the scale. Here are some rough guidelines for setting your VTT:

[color=green]1.2-1.3V[/color] - for a FSB of ~400 MHz.

[color=orange]1.4[/color]-[color=red]1.5V[/color] - for a FSB of ~420-440 MHz (exceed 1.4V at your own risk with a 45nm chip)!

[color=red]1.6V[/color] - for a FSB of ~440-475 MHz - use at your own risk with a 45nm chip!

You should be aware that newer 45nm fab chips are MUCH less tolerant toward high VTT than their 65nm predecessors. Anantech published their experience [frying a QX9650](#) with high VTT's as an example.

Vcc - Initially, set within 200-400 mV of where the auto setting used (remember that you need a little more in the BIOS compared to what CPU-Z told you). Remember to consult [Intel's processor finder](#) to know where the upper-end of safety is for your processor (I believe the figures there correspond to the values CPU-Z is displaying, not what you set in the BIOS.).

DRAM - What ever the RAM manufacture recommends is a good starting point. Unless you're really overdriving them, they shouldn't need more.

SBCore - I've always used the lowest setting, but I typically don't push my systems that hard (20-25 %). You're on your own here.

NBCore - Start off low, 1.33 or 1.37 and see if you need more. Also, a little bit can go a long way. My system is unstable @ 1.330V here but stable @ 1.370V which is a difference of only 40 mV (0.04V).

Here are the levels my Q6600 @ 9x333 uses to run stable:

Code :

```
. Memory Voltage=2.100V
. CPU VCore=1.2625V
. FSB Termination=1.200V
. NB Vcore=1.25V
. SB Vcore=1.50V
. ICH Chipset=1.057V
```

Here are the levels my X3360 @ 8.5x400 uses to run stable:

Code :

```
. Vcc=1.12500V
. SB 1.05V=1.070V
. NB Core=1.370V
. SB Core/CPU PLL=1.550V
. CPU VTT=1.310V
```

I show those only to give you an idea, not all hardware is the same, and really, those values are personal to my chip, RAM (and RAM settings), MB, etc.!

Once you select a baseline set, that will complete a POST, you'll want to start a more vigorous evaluation by changing the MB vcores **one-at-a-time** moving forward. If you change too many variables at once, you'll never be able to arrive at the stable settings. Confused? Don't be, just read on and after you see the examples, I think the process will seem clearer to you.

The basic process is to try different Vcc values keeping the other vcores constant. Run p95 at a given Vcc and record what happens after an arbitrary time point (10 to 15 min is good to start with). If Vcc level is stable for 15 min of p95, reboot and lower it a little and repeat. The goal is to find the minimum level that gives errors, then increase it until it's stable, then extend that time out to say 2-4 h. If it's still stable, further extend it to 10-14 h. You can probably call it "stable" if you can run p95 for 24 h. If a setting fails after 4 h, increase it one notch or so and repeat until it's stable out to 24 h. You can then come back knowing this Vcc and try to lower one of the other vcores repeating the process. Yes, it's time consuming and yes, it's tedious, and yes, that's a **** of rebooting, but it works.

The key to this process is keeping a detailed record to help you achieve a stable system and troubleshoot which vcore to change – p95 errors are NOT always the fault of a low Vcc! Without these data, you'll have a tough time. So what do you keep track of here?

- 1) The MB vcores you're using
- 2) The Vcc values you're testing
- 3) Which core failed (prime95 tells you) and how long it took to fail
- 4) Any observations or comments you want to record for yourself

Here is an example minimizing vcores using my X3360/P35-based system. The data presented aren't fabricated to help illustrate the method; rather, they are the real data I used to arrive at the stable system.

Hardware specs for your reference:

Quote :

X3360 running @ 8.5x400, DFI LT P35-T2R (BIOS 3/17/2008), Ultra-120 Extreme, Corsair TWIN2X4096-8500C5DF 2x2 GB @5-5-5-15 running @ 960 MHz (5:6), 620HX power supply.

Before we dig into the examples, know that to really *really* do this right, you'd need to do several runs at the various levels; doing it just once as I am is the quick 'n dirty approach and can cause you to draw an incorrect conclusion or two as you will see.

On to it: in my first try, I set up my MB vcores and began testing Vcc starting low (I chose 1.12500V somewhat arbitrarily).

Keeping the motherboard vcores constant, I varied the Vcc starting out low and working up high. You may or may not get a stable system on your first set of iterations (probably not actually). If you do, you'll probably want to repeat keeping your stable Vcc but optimizing (minimizing) for one of the other vcores such as NB or VTT, etc.

Code :

```
. Overclocking log, Iteration Set 1
. Comments: Initial try
. DRAM      2.100V
. SBCore    1.55V
. NBCore    1.37V
. VTT       1.200V
. Vcc/Prime95 success or failure
. 1.12500V   Failed on core 3 ~ 5 min
. 1.13750V   Failed on core 0 ~ 28 min
. 1.15000V   Failed on core 2 ~1 h 18 min
. 1.16250V   Failed on core 1 ~ 4 h 4 min
```

Looking at the data, we see there that multiple cores have failed as I increased the Vcc. That's suggestive of one of the *other* voltages lacking and thus needing to be increased. There are two likely causes for my instability: NBCore and VTT. In my next Iteration set (below), I chose to raise the NBCore several notches keeping the rest of the MB vcores constant.

For discussion's sake, let's say the same core failed repeatedly. This scenario is *likely* caused by a low Vcc (although it doesn't have to be). For you quad core users, cores 0/1

and cores 2/3 should be treated the same, so if you get some core 0 and core 1 failures, treat them like a single core failure as you consider this analysis.

So, I increased the NBCore a few notches and tried a few higher Vcc settings just to see if it was enough:

Code :

```
. Overclocking log, Iteration Set 2
. Comments: Added some NBCore
. DRAM      2.100V
. SBCore    1.55V
. [b]NBCore  1.41V[/b]
. VTT       1.200V
. Vcc/Prime95 success or failure
. 1.16250V   Failed on core 2 ~2 min
. 1.17500V   Failed on core 1 ~3 min
```

Again, I got two quick failures across the entire chip. Ideally, you might want to collect more data points, but I took a hunch that 1.45V should be plenty for 8.5x400, and next added some VTT keeping the newer, higher NBCore constant – remember to only change one of them per iteration set!

Code :

```
. Overclocking log, Iteration Set 3
. Comments: Added some VTT and kept the higher NBCore
. DRAM      2.100V
. SBCore    1.55V
. NBCore    1.41V
. [b]VTT     1.310V[/b]
. Vcc/Prime95 success or failure
. 1.17500V   STABLE 15 min
. 1.16250V   STABLE 15 min
. 1.15000V   STABLE 15 min
```

Now, with the higher VTT, I didn't get a single failure for at least 15 min at the three Vcc values I ran. I concluded that the VTT gave me the stability. To test this hypothesis, I kept the higher VTT, but lowered the NBCore back to 1.37 and repeated in the 4th iteration:

Code :

```
. Overclocking log, Iteration Set 4
. Comments: Kept the VTT, lowered the NBCore
. DRAM      2.100V
```

```
. SBCore      1.55V
. [b]NBCore    1.37V[/b]
. VTT         1.310V
. Vcc/Prime95 success or failure
. 1.15000V    STABLE 2 h
. 1.13750V    STABLE 30 min
. 1.12500V    STABLE 1 h
. 1.07500V    crashed p95 (n=2)
. 1.09375V    crashed p95 (n=1)
. 1.10625V    BSoD after 1+h
. 1.11875V    STABLE 11 h
. 1.11250V    Failed on core 0 ~ 1 h 8 min
```

Now I got some stable runs. After evaluating the data, I was able to nail down both my NB and VTT in only 3 iteration sets, arriving at what I thought was the stable Vcc in the 4th (I was later wrong).

It's a little easier to visualize if you sort the Vcc from low to high. If you keep your log in a spreadsheet, you can easily sort them, here are the same data sorted by Vcc:

Code :

```
. Overclocking log, Iteration Set 4
. Comments: Kept the VTT, lowered the NBCore
. DRAM       2.100V
. SBCore     1.55V
. [b]NBCore   1.37V[/b]
. VTT        1.310V
. Vcc/Prime95 success or failure
. 1.07500V   crashed p95-program exited (n=2)
. 1.09375V   crashed p95-program exited (n=1)
. 1.10625V   BSoD after 1 h
. 1.11250V   Failed on core 0 ~ 1 h 8 min
. 1.11875V   STABLE 11 h
. 1.12500V   STABLE 1 h
. 1.13750V   STABLE 30 min
. 1.15000V   STABLE 2 h
```

It would seem as though 1.11875V was the winner. I could have stopped right here and repeated extending the time out to 24+ h with these settings, but I elected to further optimize and targeted the VTT since I thought I could do better having jumped from 1.20 to 1.31 and skipping 5 sub levels in the process. This time through, I held the Vcc constant and varied, VTT:

Code :

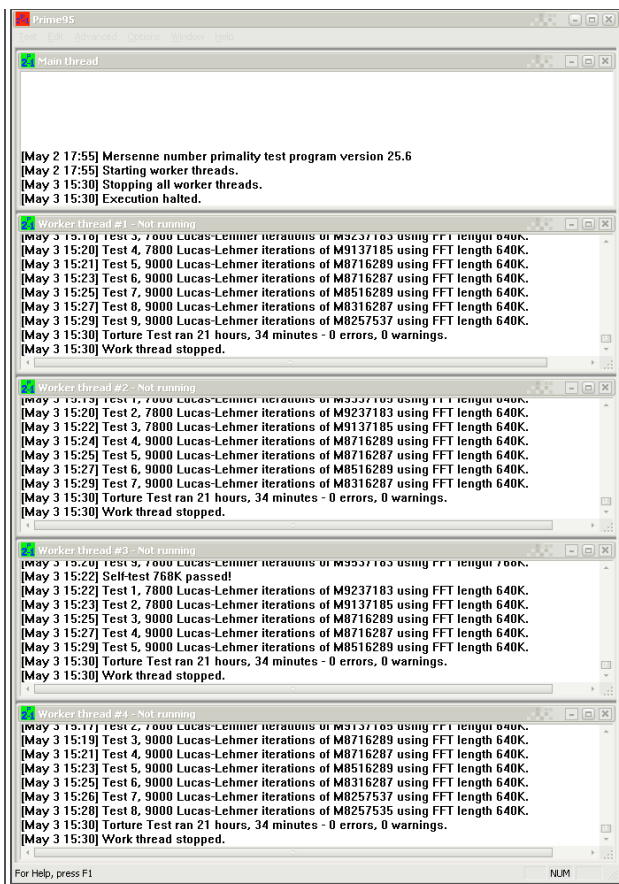
```
. Overclocking log, Iteration Set 5
```

```
. Comments: 1.11875V seemed stable, minimizing VTT
. DRAM      2.100V
. SBCore    1.55V
. NBCore    1.37V
. [b]Vcc    1.11875V[/b]
. VTT/Prime95 success or failure
. 1.250V    Failed on core 0 ~ 2 h
. 1.260V    Failed on core 2 ~ 1 h 20 min
. 1.280V    Failed on core 0 ~ 18 h 22 min
. 1.310V    Failed on core 1 ~ 1 h 20 min
```

This one is a little puzzling since the 3rd run (VTT=1.280V) lasted for over 18 h, yet the 4th run with a *higher* VTT died in under 1-1/2 h. My thinking was that VTT wasn't the problem, and that I had been misled on the Vcc. I was also getting a little anxious for this to be finished and I broke my own cardinal rule for the next iteration set by upping two variables at once: Vcc to 1.12500V and VTT to 1.310V.

Code :

```
. Overclocking log, Iteration Set 6
. Comments: 1.11875V seemed flaky, so upped the Vcc and kept
  the higher VTT.
. DRAM      2.100V
. SBCore    1.55V
. NBCore    1.37V
. [b]VTT    1.310V[/b]
. Vcc/Prime95 success or failure
. [b]1.125000V[/b]    STABLE 21 h 34 min
```



Okay! So maybe it was the Vcc after all since it ran for over 21-1/2 h before I stopped it. You could argue that there's no difference between 18-1/2 h and 21-1/2 h and you would have a valid argument. This underscores the need to collect multiple data point per level as I mentioned in the beginning of this section (I told you it was quick 'n dirty)!

Finally, I set out to essentially repeat my Iteration Set 5 minimizing the VTT with the slightly higher Vcc.

Code :

```
. Overclocking log, Iteration Set 7
. Comments: 1.12500V seemed stable, minimizing VTT
. DRAM      2.100V
. SBCore    1.55V
. NBCore    1.37V
. Vcc       1.12500V
. VTT/Prime95 success or failure
. 1.250V    Failed on core 0 ~ 1 h 3 min
. 1.280V    Failed on core 1 ~ 1 h 0 min
. 1.310V    STABLE 34 h 41 min
```

Apparently VTT needs to be 1.310V on this system. In any case, those examples should serve to illustrate the method you need to use to attack the task.

To summarize, using a stepwise approach and documenting your runs, you should be able to arrive at a stable system (assuming your hardware can operate at the level you choice). It probably goes without saying that you will need to repeat this process if change your operating conditions (multiplier and FSB).

Temperature Management

An overclocked quad system is often limited by the amount of heat it's producing, and

the ability of the heat sink and fans to dissipate it. If you're getting high temps, there are a number of things you can do to help. Most of them are hardware related but the first is the single most important non-hardware change you can make:

- Minimize your vcores first (described in the guide above)!
- Ensure good contact between the CPU and Heat sink is a must for efficient heat transfer. A major bang-for-the-buck modification in this regard is lapping the surfaces that transfer heat (the base of your heat sink and the top of your CPU). This involves gently moving the surface along wet/dry sand paper in increasing grits on a flat surface such as a piece of glass. I did both the base of my Ultra-120 Extreme and the IHS (Internal Heat Spreader) on my Q6600 and saw some pretty dramatic decreases in load temps.

It should be noted that lapping your HS and/or CPU will void the warranty.

Comparing my stock HS/CPU to my lapped HS/CPU, on average lapping lowered the coolest core by 7 °C and the hottest core by 10 °C. To read more about lapping your heat sink and CPU see these two threads; I have results and pictures of the process:

[Lapping Q6600 IHS](#)

[Lapping the Ultra-120 Extreme](#)

That said my X3360 did not need to be lapped. I'm not sure if Intel is doing this with all their 45nm chips or just the Xeons, but it came from the factory very flat. When I run prime95, the heat spread between cores is 2-3 °C.

- If your NB chipset runs too hot, consider adding a small fan. I put a silent 40x40x10mm fan on my NB HS via a zip tie which lowered my NB temps by ~7 °C on load. Pretty amazing effect for \$3 fan and free zip tie 😊

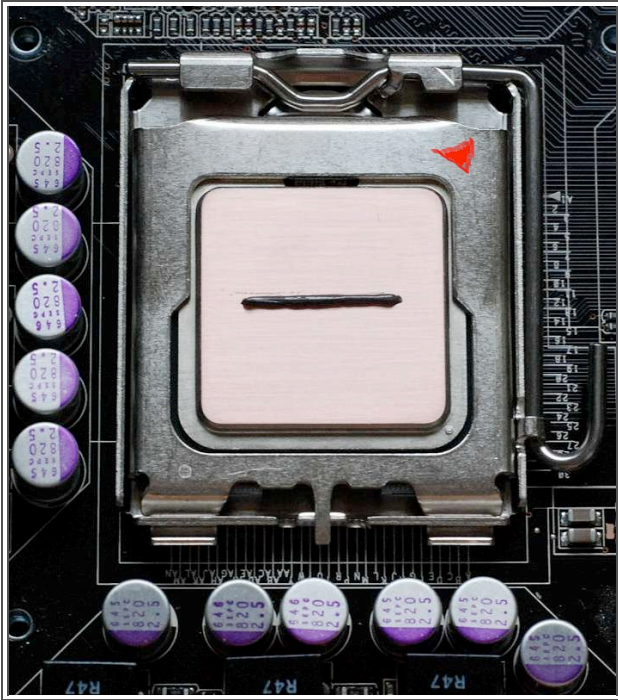


- Consider an upgrade to a more efficient heat sink (like the few mentioned in the beginning of the guide). Remember that a quad core chip will produce about 2x the heat compared to a dual core chip. You really do need to consider using an extreme HS if you plan to overclock a quad.
- Consider an upgrade to the cooling fan on the heat sink to something that has more flow. Most of the larger HS's will use a 120mm fan. Some have the option for two fans. I think the fastest 120mm fan you can use is around 1600 RPMs. If you have a slower one, you might consider upgrading.

- Reseat your heat sink and make sure you're using a quality TIM (thermal interface material) such as AS5. Consider rotating the HS 90 degrees if it is designed to do so. I seem to get better contact with my Ultra-120 Extreme when it's orientated

“North/South” than when it’s orientated “East/West.”

- Re-evaluate the way you’re applying the TIM/don’t use too much or make sure you’re using enough. Thermal pastes aren’t all created equally. Some are reported to be better than others. I have always used Arctic Silver 5 on my CPUs (and AS3 and AS1 before that). You can find all sorts of posts out there showing one to be better than another. I’ll leave it up to you to pick one. Again, I like AS5. Here is a shot of my q6600 installed in the MB with AS5 right before I added the HS. It shows the right amount in my opinion given a lapped HS and CPU (which is a thicker line than I used before); the red triangle I drew shows where that tag is on the CPU, remember that on quad core chips, the dies are placed in a different located relative to a dual core, see the [instructions](#) on AS5's website for more on this.



- Use good cable management inside your case. Use twist ties or tie downs to bunch cables and keep them out of the way of airflow.
- Make sure you have adequate airflow inside the case and make sure you’re using a well ventilated case. People often overlook this, but it’s important. Not all cases are designed for good airflow. I have an Antec P182 which is a great design. Make sure you have several exhaust fans and at least one intake fan. 120mm fans move more air than smaller 80mm fans do and also run much more quietly.

You can see that my CPU load temps will increase/decrease as the ambient temperature fluctuates. Have a look at the following thread for details:

[Effect of room temp on CPU load temps](#)

Controlling vdroop

Remember the vdroop you saw earlier? If you have a P5B-Del, you can use a pencil to modify your board to minimize or fully remove this idle-to-load vdroop. Read the following thread if you want to do that:

[Get more vcore under load: vdroop pencil mod \(pics\)](#)

That's it for the guide. I hope you got some good info out of it and are able to successfully o/c your system as a result!