Energy efficiency and code size

for ordinary embedded applications in C language

August 2008
www.imsystech.com

Abstract

C language benchmarks devised by Texas Instruments to demonstrate the superior efficiency of their MSP430 microcontroller family have been used here to determine how well Imsys IM3000 processor measures up to its most energy efficient and code efficient competitors.

1. Summary

The Imsys architecture has proven its efficiency for some important special things like Java interpretation, data compression/decompression, and precision time&frequency control. It has a rich ISA, stack oriented architecture, microprogrammed control unit, narrow datapath, shallow pipeline, and no cache – the opposite of a RISC in all aspects. Ordinary C code, without optimization, should not be expected to be its forte.

Yet when testing the MSP430 benchmark programs from TI on the IM3000, the results show that its code density outshines the MSP430 and the competitors TI has chosen to compare with, and that its energy efficiency is in fact better than that of MSP430. It is also several times faster.

2. The C benchmarks

In order to demonstrate the efficiency of their microcontroller family MSP430, Texas Instruments has devised a number of benchmark programs in C language, including Dhrystone and Whetstone, FIR filter, and a suite of small tests of various sequences common in modern microcontroller applications. They have published [1] the results for two slightly different members of the family and also for eight processors from their competitors. Compilers from the same independent compiler vendor (IAR) have been used for all, in order to minimize differences due to compiler efficiency.

Code size and cycle count are compared. The cycles counted are machine cycles; in most cases here these are the same as memory cycles. These numbers should indicate energy consumption and execution time, but the actual calculation of those important parameters from the cycle count is left to the reader. To that end, the clock division factors are given for the processors (there are more than one clock cycle per machine cycle for some of the processors).

2.1. Using the cycle count to calculate energy consumption and execution time per benchmark

Execution times for a given clock frequency can be calculated directly (cycle count * cycle time). Since the energy consumed per cycle in a CMOS processor is essentially independent of clock frequency, energy consumption per benchmark can be calculated if power consumption at some frequency can be obtained from a datasheet. These calculations are meaningful only if there is no cache memory involved (which is often the case for one of the processors, ARM7).

Furthermore, since a datasheet is needed, it is not sufficient to know the architecture. Most of the competitors that TI has chosen are microcontroller families or generic architectures rather than particular devices. Energy efficiency and speed have here therefore been calculated from cycle count only for the two MSP430 devices and for IM3000.

TI has chosen the code, but otherwise the method used produces results that are rather indisputable. Using cycle count avoids interpretation of datasheets. The cycle count results are also more generally useful than measured results for certain devices; together with datasheets they can be used for estimation of energy efficiency and speed for many more different microcontroller devices, as long as these share the internal processor core architecture with one of the example processors.

2.2. Test conditions – voltage and clock frequency

For a CMOS processor of a given design, the following is normally true:

- performance is proportional to clock frequency
- power consumption is proportional to supply voltage and clock frequency
- maximum clock frequency is roughly proportional to supply voltage.

Normally one nominal supply voltage is specified, together with a tolerance, which is often close to +/-10%. The specified maximum clock frequency is safe to use for any supply voltage within the range, i.e. it is the frequency that the processor must tolerate at the minimum voltage limit. Typical power consumption is specified for this maximum clock frequency, but with the nominal supply voltage. IM3000 is specified in this manner. Its specified maximum clock frequency can be used with a supply voltage that is 10% below the nominal.

The MSP430 devices have a wider voltage range, 1.8-3.6V. They are therefore specified differently, allowing the customer to make a trade-off between speed and power consumption. Allowable clock frequency is specified as a function of supply voltage. The upper limit there is 8MHz but that frequency cannot safely be used, since it would not allow any tolerance at all for the voltage, which would have to be exactly 3.6V.

In different application notes about power supply for MSP430, TI uses 3.0V, 3.3V, and 3.45V. These nominal voltages would require tolerances within +/- 20%, 9%, and 4%, respectively, due to the specified limit of 3.6V.

IM3000 is specified with 10% voltage tolerance. For the comparison we therefore choose 3.3V (close enough, and not favoring IM3000) as one nominal supply voltage for MSP430, with a tolerance of plus/minus 0.3V. The lower voltage limit is then 3.0V, and this is the voltage for which the maximum clock frequency should be calculated.

From the frequency/voltage diagram in the datasheets, the maximum clock frequency at 3.0V is found to be 6.7MHz. (The diagrams for the two MSP430 devices in the study are slightly different, but only at the very low voltages, and that difference has no influence here.)

2.3. Energy consumption for MSP430

Typical consumption is for MSP430 given in the datsheets [2], [3], as current drawn at a clock frequency of 1MHz. The intention is that the customer should multiply this current consumption by the frequency, in MHz, that is actually used, in order to know how much current is needed from the supply.

This is not what we are primarily interested in; instead we want to know how much total energy the processor consumes for each benchmark execution. Therefore we convert the given data (uA at 1MHz) to energy consumed per clock cycle (uWs per cycle), by multiplying by voltage and dividing by 1MHz.

The typical consumption is for MSP430 given at two different supply voltages, 3V and 2.2V. However, to be fair we must use the power consumption at the nominal voltage 3.3V. This is calculated as (3.3/3.0) times the specified value for 3.0V.

Nom. supply voltage	**3.3V**
Multiplier for nominal voltage	3.3/3 = 1.1
MSP430FG4619	
Current@1MHz	0.6mA@3V
At nominal supply voltage:	
Current@1MHz	1.1*0.6=0.66mA
Power@1MHz	3.3*0.66= 2.18mW
Energy/cycle	**0.00218 uWs**
MSP430F149	
Current@1MHz	0.42mA@3V
At nominal supply voltage:	
Current@1MHz	1.1*0.42=0.462mA
Power@1MHz	3.3*0.462=1.52mW
Energy/cycle	**0.00152 uWs**

The following two cases can thus be used for comparing execution time and energy efficiency with other processors:

- MSP430FG4619 at 3.3V, 6.7MHz (149ns cycle), consuming 0.00218 uWs/cycle
- MSP430F149 at 3.3V, 6.7MHz (149ns cycle), consuming 0.00152 uWs/cycle

2.4. Energy consumption for IM3000

The IM3000, when executing standard ISA instructions (using microinstructions in ROM), consumes 19.2mA from 1.8V when clocked at 167MHz. Each microinstruction needs 2 clock cycles and therefore consumes 19200*1.8/83333333 = 0.000415 uWs. The number of microinstructions (uinstr) per ISA instruction can be very different, but the average number has been calculated for each benchmark and is shown here. This is multiplied by the number of "cycles" (ISA instructions) executed by the benchmark, to get the number of microinstructions, which is then multiplied by 0.000415 uWs.

The tables in Appendix A show that IM3000, although being a much more complex processor - e.g. capable of executing Java bytecodes natively and performing cryptographic routines faster than much bigger processors - in most cases consumes less energy for a given sequence in C language than the low-power specialist MSP430 (which in turn consumes less than the other processors). The weaker efficiency on the Matrix and Switch benchmarks reflects missing optimization in the Imsys C compiler. (A new, improved compiler exists in beta version, but has not been used here.)

The code sizes, i.e. number of bytes used, for all the processors (except for PIC18, for which data was incomplete in the TI report) have been combined for all benchmarks and normalized to the result for IM3000. The diagram below shows the inverted values, which are then the relative code density, here shown as a percentage of the IM3000 code density.

The two diagrams at the bottom show the comparison of energy efficiency (normalized inverted energy consumption) and speed (normalized inverted execution time) between the MSP430 devices and IM3000.

2.5. Combining results by using geometric average

A processor that is good on one type of processing isn't necessarily just as good on another. Therefore it is valuable to combine the results of a suite of different benchmarks, covering different types of processing that are as relevant as possible for the application at hand. However, different benchmarks usually have different size and require different amount of work, and this is not related to their importance. An arithmetic sum or average is then not a good way to combine the result, since it will give greater weight to the bigger benchmark programs.

A geometric mean value, of N results, is equal to the Nth root of the product of all N results. This is a much better way of combining benchmark results, since all results get the same importance. If, for example, one result is improved by 50%, the change will be the same regardless of which one of the results was changed

2.6. Note about internal vs. external memory

IM3000 is normally used with complex software and uses external SDRAM main memory (usually 8-32MB but 2-4096MB is possible) for programs and data, while the studied MSP430 microcontrollers have on-chip memory – but only 60/120KB flash and 2/4 KB SRAM. Thus, to compensate for this, one could argue that the consumption in the SDRAM should be included for the IM3000. However, if the two architectures, in a hypothetical IC design, were to compete for the same application, then either the software would be small enough for the MSP430 device and the IM3000 processor would then also use internal memory (since its code would be even smaller than that for MSP430), or the software would be large and the internal memory of MSP430 would need to be replaced by SDRAM interface logic – which may in fact consume more than the internal flash it mainly replaces. Note also that the program memory activity per benchmark for IM3000 is the lowest of all the processors, leading to low memory energy consumption. This can be seen in the ISA instruction count column in the tables in Appendix A, considering that the width of the IM3000 SDRAM interface is only 8 bits, which is also the full size of most of the instructions fetched from the SDRAM.

References

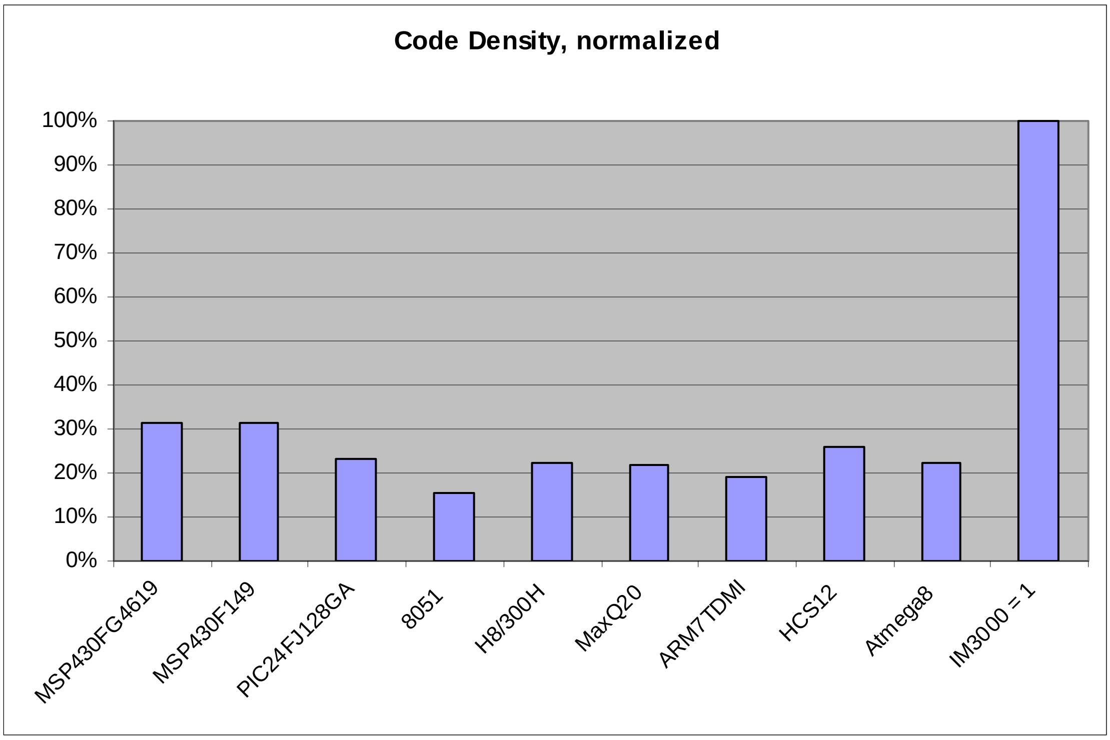
[1] Texas Instruments, "MSP430 Competitive Benchmarking", SLAA205B – June 2005 – Revised July 2006. Available on the Internet.

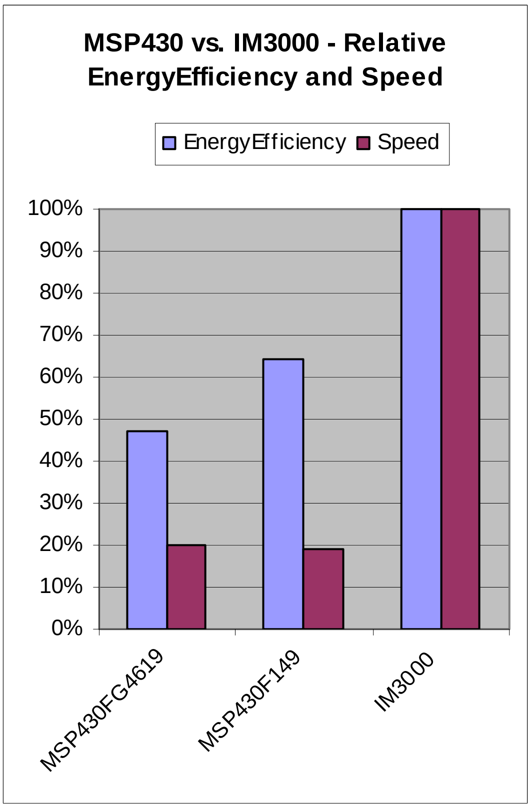
[2] Texas Instruments, Datasheet "MSP430xG461x Mixed Signal Microcontroller", SLAS508D – April 2005 – Revised January 2007. Available on the Internet.

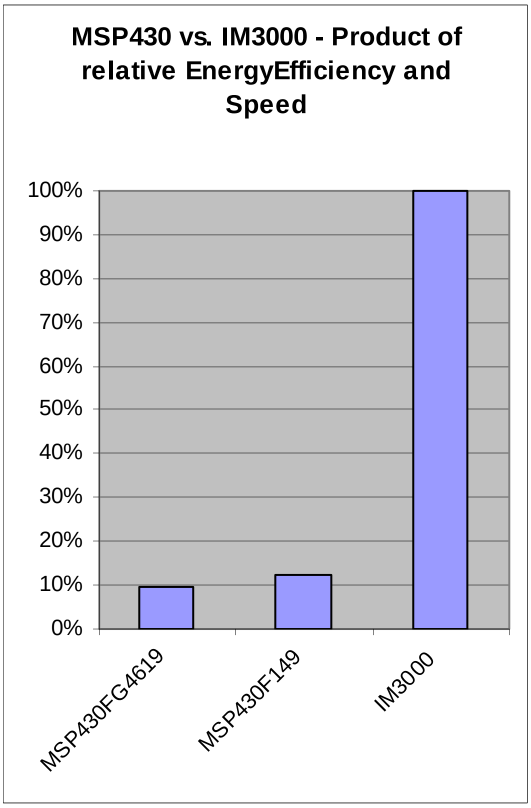
[3] Texas Instruments, Datasheet "MSP430x47x3, MSP430x47x4 Mixed Signal Microcontroller", SLAS545A – May 2007 – Revised December 2007. Available on the Internet.

[4] Imsys Technologies, "Data Book, IM3000 Family Microcontrollers" ", STO-DEV7026-CB. Available on the Internet (www.imsystech.com).

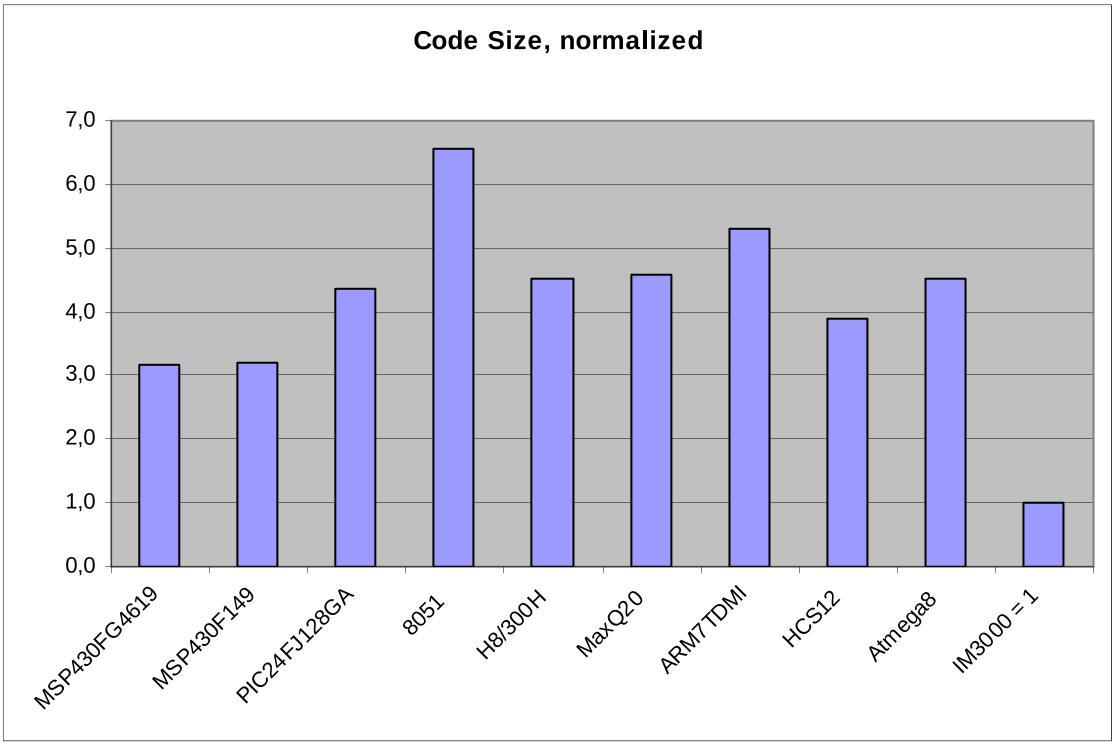
Diagrams – Version 1 (High is better)

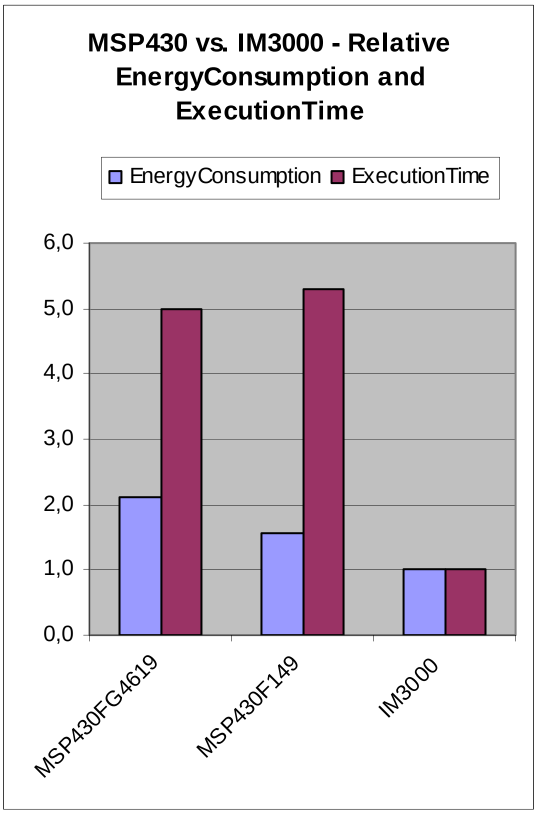


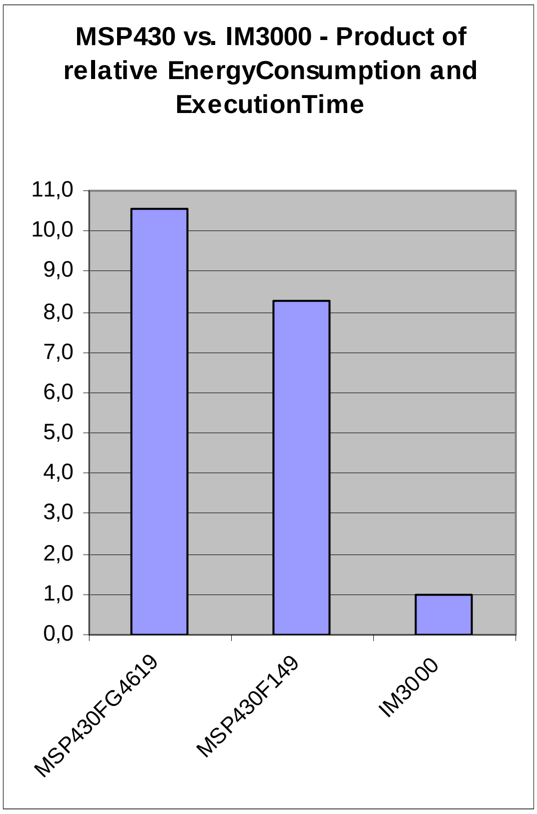




Diagrams – Version 2 (Low is better)


Code Size, normalized
MSP430FG4619
MSP430F149
PIC24FJ128GA
H8/300H
MaxQ20
ARM7TDMI
HCS12
Atmega8
IM3000 = 1


MSP430 vs. IM3000 - Relative EnergyConsumption and ExecutionTime
EnergyConsumption
ExecutionTime
MSP430FG4619
MSP430F149
IM3000


MSP430 vs. IM3000 - Product of relative EnergyConsumption and ExecutionTime
MSP430FG4619
MSP430F149
IM3000