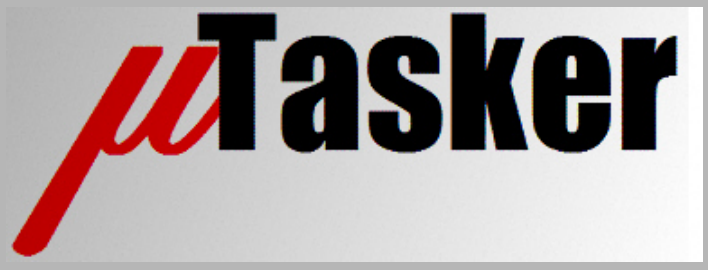
Embedding it better...



µTasker Document

µTasker Benchmark Measurements

Table of Contents

1. Introduction

µTasker is an operating system designed especially for embedded applications where a tight control over resources is desired along with a high level of user comfort to produce efficient and highly deterministic code.

The operating system is integrated with TCP/IP stack and important embedded Internet services along-side device drivers and system specific project resources.

µTasker and its environment are essentially not hardware specific and can thus be moved between processor platforms with great ease and efficiency.

However the µTasker project setups are very hardware specific since they offer an optimal pre-defined (or a choice of pre-defined) configurations, taking it out of the league of “board support packages (BSP)” to a complete “project support package (PSP)”, a feature enabling projects to be greatly accelerated.

This document discusses some performance comparisons with various events and settings.

2. M52235 EVB – Power consumption test - V1.2.008

The power consumption of the M52235EVB was measured from 3V3 when the software was operating at 40MHz with and without 100MHz LAN connection.

Current from 3V3 (LEDs disabled)	***Without LAN connection***	***With 100M LAN connection***
Normal operation	280mA	330mA
With LOW_POWER task	260mA	310mA

Note: Reference current when the processor is held in reset = 80mA

Conclusion

It can be seen that low power support saves current but the saving is minimum on the M52235EVB. It has to be further investigated whether the low power mode (using the stop instruction) puts the processor in the lowest power state or whether other settings may improve the figure.

3. M52235 EVB – UDP reaction test – Project version V1.2.008

UDP frames were sent to the evaluation board and echoed back by the software. The time from the Ethernet RX frame interrupt to the call back routine, the time required for a copy of the received data (using `uMemcpy()`) to a backup buffer, and the time from the UDP transmission routine call until activating the transmit buffer were measured. The measurement was repeated for various project configurations and the influence of the memory requirements also recorded.

The memory copy step is not required for an echo to be achieved but was tested to monitor general `uMemcpy()` performance.

The tests were performed at 40MHz. The delays are inversely proportional to the PLL clock used. The M5223X is specified to 60MHz although its PLL can operate beyond 100MHz under normal conditions (although not officially specified for this).

	512 byte UDP echo	***1024 byte UDP echo***	***Program size***	***RAM requirements***
Basic	408/181/586 = 1'175µs	708/359/1'060 = 2'127µs	Reference	Reference
No UDP CS	84/181/265 = 530µs	86/364/443 = 893µs	Same as ref.	Same as ref.
DMA for uMemcpy and uMemset	408/88/492 = 988µs	730/175/880 = 1'785µs	+208 bytes	-20 bytes
Loop code in SRAM	393/164/559 = 1'116µs	680/333/1'010 = 2'023µs	+848 bytes	+473 bytes
Loop code in SRAM and DMA	393/88/470 = 951µs	685/175/853 = 1'713µs	+800 bytes	+364 bytes*
Loop code in SRAM and DMA No UDP CS	84/88/170 = 342µs	87/175/256 = 518µs	+800 bytes	+364 bytes

Notes:

- *Rx interrupt->Callback / uMemcpy() / Call to transmission = Total*
- *The ping reaction time from reception interrupt to transmit buffer activation has approximately 220µs in all cases*
- *The basic configuration is without any additional features activated but with UDP checksum calculation*
- *All times are in µs*
- *Active low power support gives identical reaction times*
- ** The uMemcpy is either implemented as DMA or in SRAM and so adding DMA decreases the RAM use in this case*
- *The following routines were operating in SRAM when the option was set – IP Check sum calculation, uMemcpy, uMemset, uMemcmp, uStrlen, uStrcmp, uStrcpy*

Conclusion

It is seen that activating DMA support for uMemcpy() results in useful performance improvements. Since the same routine is also used during the frame preparation, it also saves during this portion of the echo test. It has no RAM price and only a small additional code size of around 200 bytes.

The results suggest that the SRAM routines run about 10% faster than when run directly from FLASH. The reaction time in the test was improved by about 5% overall but the price paid was about 800 bytes of extra code, plus about 400 bytes of SRAM for the routines themselves (several routines were operating in SRAM and so this figure is in fact much less when only the uMemcpy() and IP check sum routines are compared).

The most critical code segment in the test is obviously the IP check sum over the UDP data. By deactivating the calculation, the reception frame and also the transmission frame are processed much faster. DMA support cannot improve this calculation, whereas operation from SRAM results in only about a 6% advantage. This suggests that investment in an improved calculation algorithm may result in best additional performance increases.

4. K70F120 Tower Kit – UDP reaction test – Project version V1.4.1

This test is a repetition of the same test performed with the M52235EVB but on the Kinetis K70 operating at 120MHz. The main reason for repeating the test is to demonstrate the effect of the check sum offloading that is possible with the Ethernet controller in this newer device.

The tests were made with 1024 bytes payload in the UDP frames. The comparison between M52235 is based on the assumption that its processing times can be reduced by 1/3 to account for the fact that the Kinetis is running 3x faster than the original clock rate (the M52235 cannot practically run at such a clock rate but the figures make comparisons simpler). Since the M52235 cannot perform UDP checksum calculations in HW, the Kinetis figures, with disabled UDP check sum, are compared with the check sum enabled but offloading active.

	1024 byte UDP echo – M52235 (120MHz interpolated)	***1024 byte UDP echo – Kinetis K70 (120MHz)***
Basic	236/120/353 = 709µs	130/61/187 = 378µs
No UDP CS*	29/121/148 = 298µs	13.6/58/77 = 148µs
No UDP CS* ***Memory copy with DMA***	29/58/85 = 173µs	12.8/34.4/48.8 = 96µs

**For M52235 tests the UDP checksum was disabled / For the Kinetis the UDP checksum was active but HW checksum offloading was enabled for both reception and transmission*

Conclusion

This comparison shows that the Kinetis K70 executes the code, without any acceleration techniques, about twice as fast from Flash as the M52235, when its operation is interpolated to be running a 120MHz.

A direct comparison with the M52235 running at 60MHz (its fastest speed) shows that the Kinetis' execution is about 4x that of the Coldfire running from Flash.

As was seen with the original tests with the M52235, the UDP checksum calculation is the main software load involved. The use of DMA for memory buffer copies improves efficiency but is still outweighed by the calculation overhead. This overhead could only be reduced in M52235 tests by disabling the checksum operation (which is an option for UDP but not so for most other TCP/IP protocols).

Also in the case of the Kinetis tests it is clear that the use of DMA for memory buffer copies is useful and reduces the copy time by almost half. More interesting and important are the results of the use of the checksum offloading in offered by its Ethernet hardware since this achieves equivalent savings in software overhead as disabling the UDP checksum did for the Coldfire:
- the processing of the reception was reduced from 130µs to 13.6µs just be enabling this

(960% improvement of efficiency and similar to the improvement obtained by completely disabling the checksum operation in the Coldfire)

- the processing of the transmission was reduced from 187µs to 77µs, whereby this task includes a copy of the user's data to the Ethernet transmit buffer. Again, however, a similar saving is seen as was obtained by disabling UDP checksum in the Coldfire case.

When comparing just the performance with UDP checksum active it shows that the Kinetis can process the test frame of 1024 bytes payload in **96µs** (this includes an additional 1024 byte memory copy in the application to add extra overhead or around 30µs) whereas the M52235 would need **1'142µs** to do the same at 60MHz (or interpolated to 120MHz still needs 571µs for the operation).

The use of checksum offloading and DMA for memory buffer transfers allows the Kinetis to reduce its processing time from **378µs** to **96µs**, whereby the checksum offloading is the predominant factor in this test case. The use of both checksum offloading and DMA memory buffer transfer is however seen to be of significance to obtain optimal performance.

Modifications:
- V1.0 19.11.2006 Original version with M52235 tests
- V1.1 29.4.1012 Added document header and UDP performance comparison with Kinetis