

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.

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.

<i>Current from 3V3 (LEDs disabled)</i>	<i>Without LAN connection</i>	<i>With 100M LAN connection</i>
<i>Normal operation</i>	280mA	330mA
<i>With LOW_POWER task</i>	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.

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, uMemcpy, 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 can not 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.