

IN4073 Embedded Real-Time Systems Speedup Report

Imara C.T.M. SPEEK 1506374

June 1, 2014

1 Speed-up report

Additional to the original QR report, this report discusses improvements made to speed up the original code. These modifications were done on the fpga part of the system. This report will briefly go through the profiling results and the modifications made.

The profiling results as presented in table 1 were made using the X32.us.clock and measured by sending a dummy packet to the rx buffer with the interrupts disabled. The timing measurements of the interrupts were taken with the interrupt service routines enabled. The third column of table 1 represents the timing measurements after the improvements were made.

Function	Original time in (μs)	New time (μs)	Speedup (μs)
time to get a character	14	-	-
time to decode	172	150	22
time to check sum	97	43	54
time to store data	827	607	220
time to send telemetry	620	565	145
time to check commflag	16	-	-
time to switch case SAFE.MODE	277	173	104
time to switch case MANUAL.MODE	187	-	-
time to switch case CALIBRATION.MODE	199	-	-
time to switch case YAW.CONTROL.MODE	346	-	-
time to switch case FULL.CONTROL.MODE	606	316	290
isr_rs232_tx	60	-	-
isr_rs232_rx	49	-	-
total control loop MANUAL.MODE	595	429	166
total control loop YAW.CONTROL_.MODE	754	588	166
total control loop FULLCONTROL.MODE	1014	648	366

Table 1: Profiling results (total control loops exclude the storing and sending of the data)

1.1 Modifications

The biggest improvements can be made in the storing of the data, the sending of the telemetry, calculating the checksum, but mostly within the control loop in manual, yaw-control and full-control mode.

To achieve speedup within the storing of the data and the sending of the telemetry I addressed the `cbWrite()` function that writes elements to a buffer and calculates the sum by passing a pointer to it. Calculating the sum afterwards for 32 byte elements provided a speed up for storing the data and sending the telemetry. The `decode()` function is also called every control loop and in turn calls the `cbGet()` which gets a character out of the buffer. By inlining `cbGet()` a speedup of $21\mu s$ is achieved for decoding. In the `check.sum()`, the checksum is calculated and compared to the received checksum. By extracting the assignments from the for loop and writing every single assignment a speedup of $54\mu s$ can be achieved. Inlining this further downsizes the function to $37\mu s$. Inlining and extracting the for loop in the `SAFE.MODE` switch case works in a similar manner.

However, most important was the speedup achieved in the full control mode. By leaving out the butterworth filter and rewriting the kallman filter to a faster macro I was able to save $290\mu s$. This caused the full control mode loop to fit in the response time of 0.7 ms , the refresh rate of the sensor interrupt routine.