Performance Comparison of a Proprietary CCSDS Stack Implementation between two 32-bit Embedded Controllers

Muhammad Owais, Syed Ashar Akhtar
Space and Upper Atmosphere Research Commission of Pakistan, SUPARCO
owa1983@gmail.com, Engineer.ashar@gmail.com

Abstract—CCSDS has published widely recognized data communication standard for space missions. This paper discusses a custom streamlined implementation of the CCSDS channel, frame and packet layers. Non essential features and extra fields are omitted/not implemented from the packets and frames. The stack is ported for two 32-bit controllers. One is ARM7 based STR710 from STMicroelectronics and other is 80386 based, 386Ex from Intel. Both are candidates for a space application protocol processor and therefore their performance data is required, to evaluate protocol stack performance thus assisting selection. Parameters compared are frame/packet generation time, parsing time, throughput and application memory footprint. Porting effort is also considered.

It is hard to select a controller when two different architectures and tool chains are involved but this comparison may act as a worthy parameter for decision making.

Keywords*: CCSDS Stack, 32-bit Embedded Controller, Execution time comparison*

1 INTRODUCTION

The Consultative Committee for Space Data Systems (CCSDS) is an international organization formed by space agencies in 1982. CCSDS develops common standards and provides a forum for exchange of ideas. CCSDS has been very successful in generating standards which are well known and used by almost every space related organization.

CCSDS provides comprehensive specification for space data protocols. These protocols [1] are layered just like ISO layered reference model as seen in figure 1. Each of these protocols and its transfer unit is discussed next.

1.1 Space Packet Protocol

Formerly called CCSDS source packet, this protocol corresponds to the network layer routing functions. The packet format is shown in figure 2. The CCSDS source packet [6] is a data structure generated by spacecraft on-board and ground end communication systems to share application data. These packets are variable length depending in the information conveyed. The packet consists of two parts, the header and the source data. The packet header is mandatory. There is a possibility of a secondary header just before the source data.

Its inclusion is up to the communicating nodes. The length of the packet is between 7 and 65542 bytes.

1.2 Space TM Data Link Protocol

The TM Data Link protocol or transfer frames are used to transport source packets and segments through the telemetry system to receiver network. The frame starts with a 4 byte synchronization marker which is a fixed pattern called ASM (Attached Synchronization Marker). The frame pattern is shown in figure 4.

1.3 Space TC Data Link Protocol

The tele-command transfer frame is the unit of encapsulation for tele-command packets sent from ground systems to spacecrafts. It provides link to link delivery for tele-command packets. The format of TC frame is given in Figure 3. It is important to note that the data link layer for TM and TC link is different but the network (packet) layer can use same transmission unit: space Packet.

2 IMPLEMENTATION SCOPE

The implementation scope for this paper is limited to the highlighted portion (blue colored, staircase shaped outline) in figure 1; including Space Packet at network layer, TM and TC Data Link protocols at data link layer and associated channel coding schemes. The software is developed in stages. The software is developed for the PC environment first, because PC as a target is easier to develop for compared to embedded processors. After implementation for PC platform, the code is ported for STR710 and 386Ex processors. It is to be noted that only TCs frames are re-transmitted when using reliable service COP standard, not TM frames. TM reliability is not a problem because it is generated continuously by spacecrafts. If a packet gets corrupted, then next packet generated will provide the values required. If no loss is acceptable then an application layer should provide the required service.

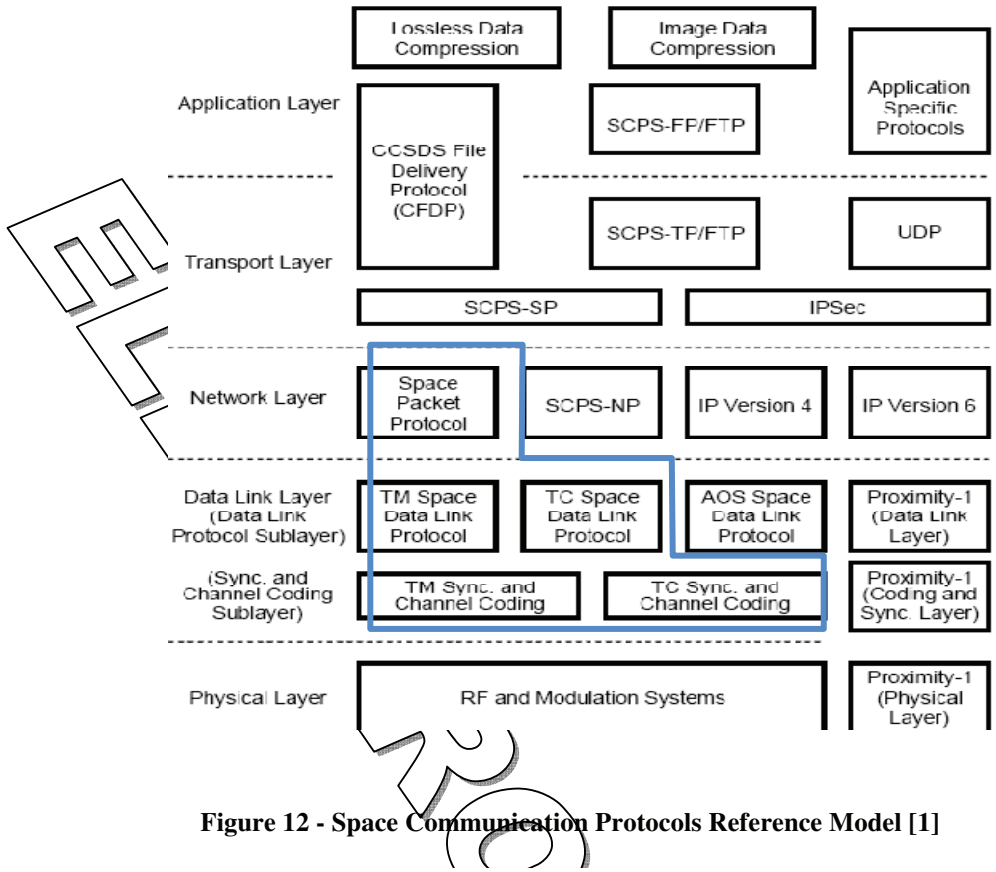

Figure 12 - Space Communication Protocols Reference Model [1]

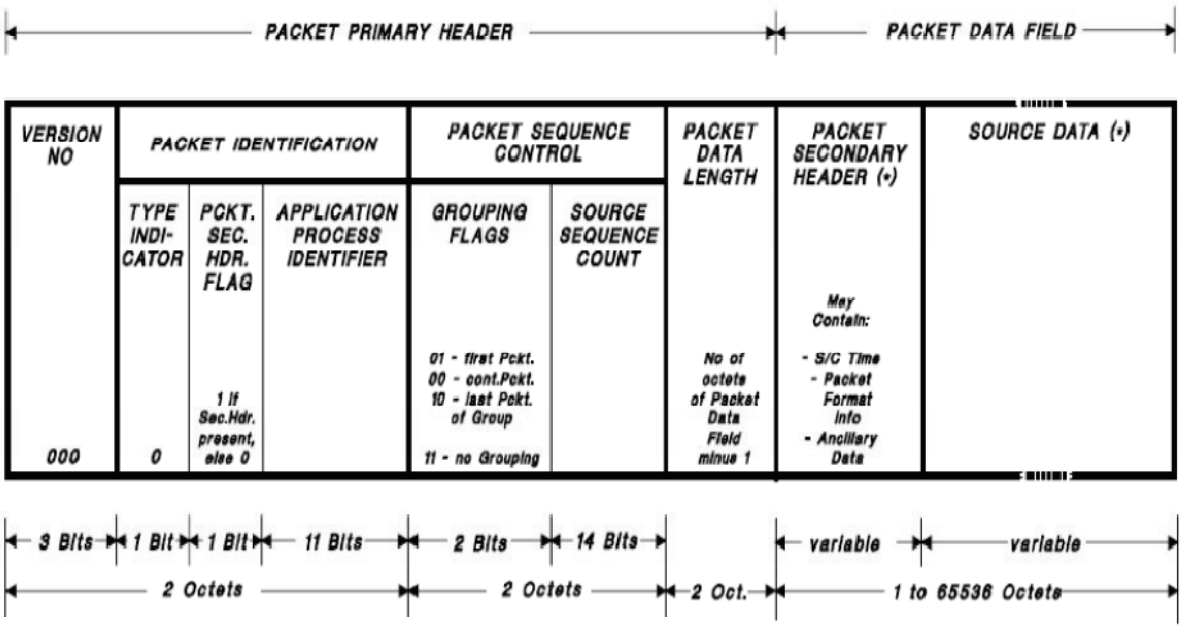

Figure 2 - Space Packet Protocol Format [6]

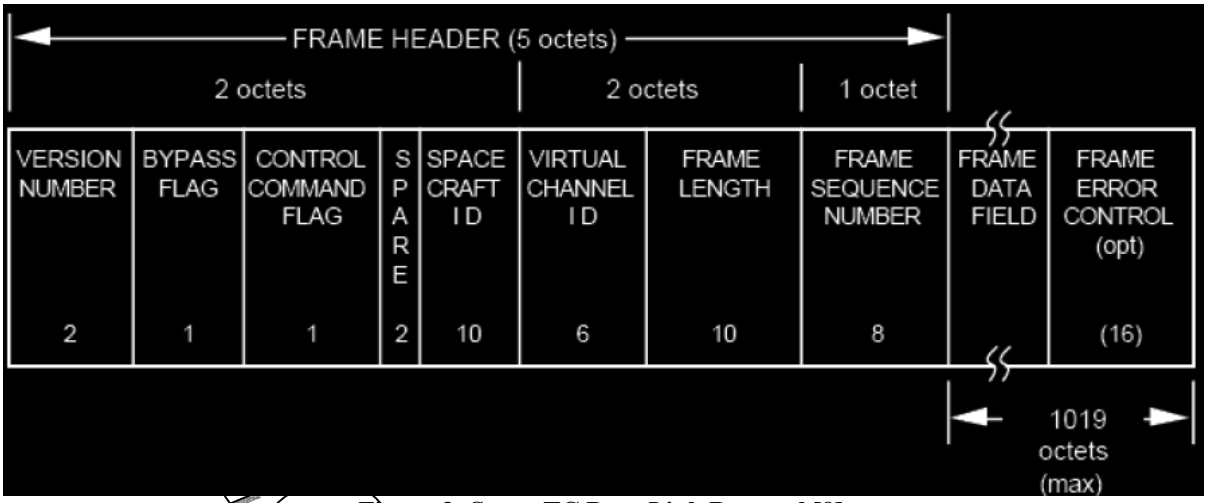

Figure 3- Space TC Data Link Protocol [9]

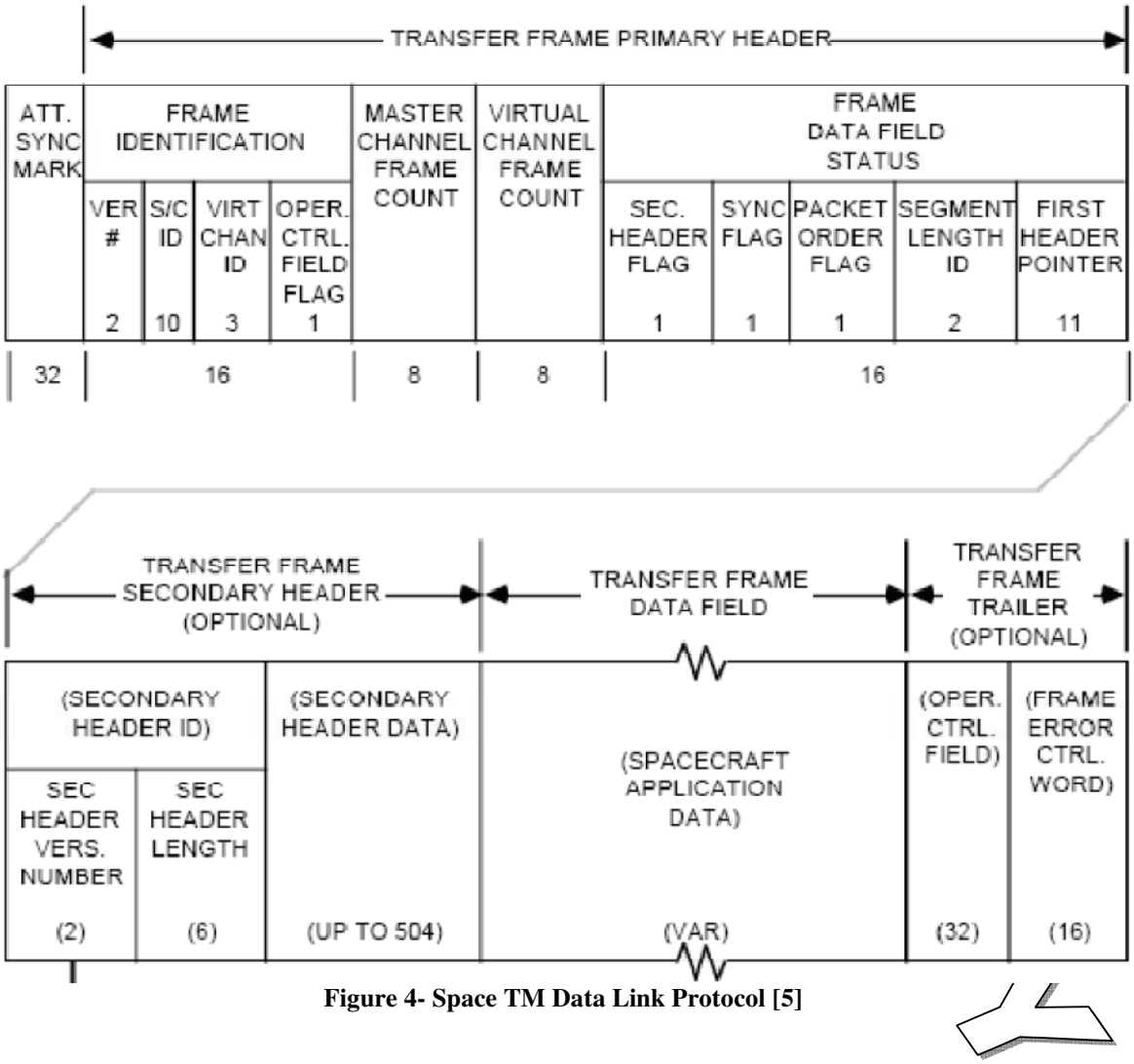

Figure 4- Space TM Data Link Protocol [5]

The software developed is not a complete implementation of CCSDS TM and TC standards. The limitations or deviations are,

1. Handling of one Virtual Channel in each direction
2. One packet per frame and one frame per Command Link Transfer Unit CLTU is used.
3. Large packets greater than a fixed size (280 bytes) are not handled.
4. Frame Extraction for TC receiver is different and uses liner array searching rather than detecting two false BCH code block failures; because searching an array of 16 characters for 16 byte fixed sequence is faster than doing BCH decoding of two 8 byte blocks.
5. No COP-1 support is implemented.

The respective channel coding and CRC schemes are implemented as well. The Channel coding scheme used for TC frames is BCH (56, 63) in error detecting mode. 16-bit CRC is used for TM frames. The CRC and BCH code is reused from another project [13]. RS coding is recommended for TM frames which is not implemented in current setup. There are two communication nodes in the experiments; one emulating a spacecraft computer / protocol processor and other is the grounds station software which sends commands and receives telemetry.
The performance of this stack is compared, when run on 386Ex and ARM7 based CPUs.

3 ARCHITECTURAL COMPARISON OF TARGET CONTROLLERS

The target processors for comparison are both 32-bit controllers. They have their similarities as well as differences. STR7 is based on ARM7TDMI RISC core and features a three stage instruction pipeline. 386Ex is a CISC architecture based on 386CX core and has a four stage pipeline. Both controllers have no on-chip cache for data or addresses (386Ex can be used in protected mode and has some address cache in this mode). There is abundance of embedded peripherals on each making embedded platform development easier.

3.1 STM STR710

There are many variants of this controller. The one which is used is STR710FZ2T6. It features 256KB internal Flash and 64KB internal RAM. It is capable of running at 59MIPS when executing code from RAM. External clock for core can be a maximum of 16MHz which can be multiplied or divided internally by a PLL Unit to reduce EMI. The clock input to the core and the peripherals is configurable using a system peripheral called clock control unit (PRCCU). The core clock is varied to get timing result at different speeds (Table 1 and 2). The maximum allowed clock for the core is 50MHz. The controller features fast interrupt controller with multiple vectors. There are abundant resources on the fabric. About 10 serial communication peripherals, 5 timers and a real time clock as well. A basic block diagram for the main components of STR7 is shown in figure 5 as taken from the datasheet.

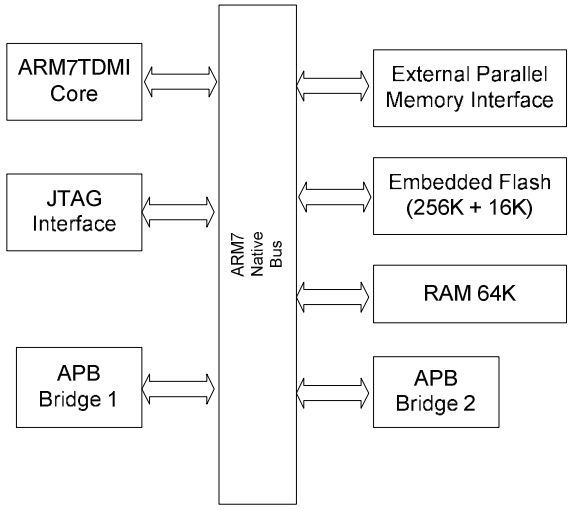

Figure 5- STR7 Simplified block diagram [15]

3.2 Intel 386Ex

The processor used in this experimental setup is Intel 386EXTC running at 33MHz @ 5.0V. It is a 32bit controller using x86 instruction set. It has CISC architecture and can execute some complex instructions. It is capable of running 11.4 MIPS. This variant of the processor features system and power management and built in peripheral including: Two 82C59A interrupt controllers; Timer, Counter (3 channels); Asynchronous SIO (2 channels); Synchronous SIO (1 channel); Watchdog timer and many parallel IOs [14]. The controller block diagram is shown in figure 6.

4 HARDWARE PLATFORM FOR TARGET CONTROLLERS

4.1 Hardware Platform for 386Ex

The 386Exhardware platform is a custom single board computer equipped with a single Intel 80386Ex. TC processor is running in Real Address Mode, with 1 MBit (128K x 8 Bit) boot EEPROM with typical read access time of 70ns and 4 MBit (256K x 16Bit) of SRAM with typical read access time of 15ns. There is no internal Flash or RAM present on the 386Ex. The crystal used is 66MHz which is scaled to half

inside the CPU by a divider. So effectively it is running at fixed 33MHz clock.

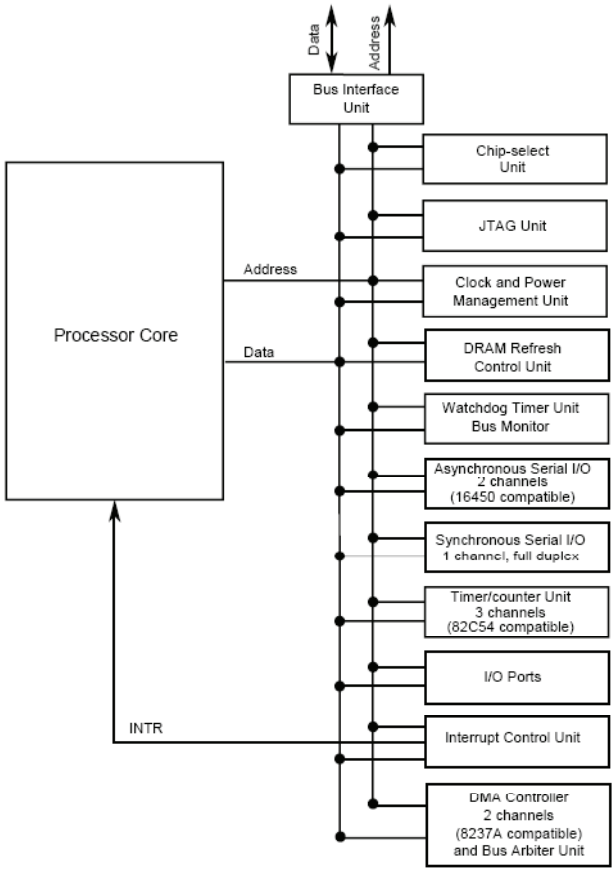

Figure 6- 386Ex Block Diagram [14]

The development tool chain used is Paradigm Pro 6 C/C++ with debug support. Results are acquired in two different scenarios. In first scenario the code was executing from SRAM using Paradigm Debug Kernel. In the second scenario code was loaded in EEPROM boot memory to execute without kernel. Both test scenarios used Real Address Mode with Large Memory model. Due to unwanted extra wait states in second scenario code showed better performance in execution from first scenario (SRAM) even under Debug Kernel. Thus, to obtain better performance from the processor, code should be executed from high-speed SRAM without presence of any Debug kernels. The results in table 1 and 2 are based on code execution from the external RAM.

4.2 Hardware Platform for STR7

The hardware platform used for STR7 is a COTS board. It is called STR710-EVAL [15]. It has external 16-bit Flash and RAM installed for use but it is chosen to use internal 64KB RAM only as the entire code and data fits in it. One UART is used for transfer of TC and TM frames. There are some LEDS and test points present which are used for indicating status of running code. The JTAG header is used to program and debug the board. No other board peripheral is used.

The development tool chain used is IAR Embedded Workbench for ARM version 5.50. The code executes in a flat memory model and there is no address translation required. The processor runs in ARM mode (also possible to use Thumb mode).

5 PERFORANCE MEASUREMENT RESULTS

The main performance metric in consideration is time required to complete packetization and processing tasks. Code density is not an issue at the moment and is not considered. The image size of binary for STR7 is about 32KB with additional 12KB for RAM. The code was first written for a PC. It was then ported to the STR7. After successful execution on STR7, it was then ported to 386Ex. The required changes were minimal and mostly involved changing device drivers for peripherals.

Execution Time is calculated on both platforms for two tasks.

1. Time required for preparing, filling and transmitting a TM packet inside a TM frame.
2. Time required for receiving, decoding and consuming a TC packet inside a TC frame.

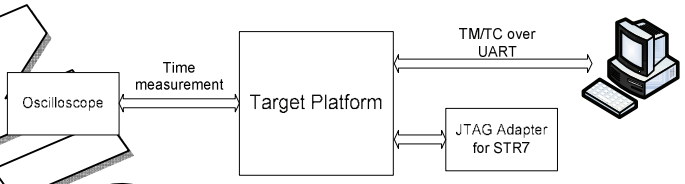

Figure 7 - Experimental Setup

The times are calculated by pulsing specific GPIO lines and using oscilloscope to measure the pulse duration. RTC and internal timers can be used also but it was decided to use GPIO lines as their drivers are very simple and have no execution overhead. Table 1 shows the execution speed results with different processor speeds on both controllers. STR was setup for 32MHz and 386Ex fixed at 33MHz. STR7 was found to be faster in both TM and TC processing. Figure 7 shows block diagram of experimental setup.

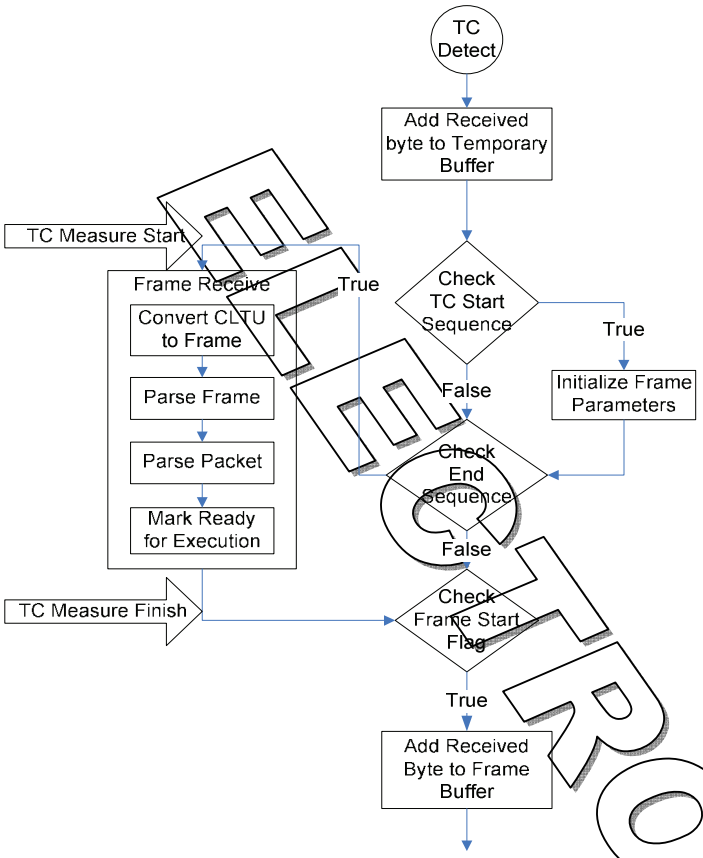

Figure 8- Flow chart for TC processing

Figure 8 shows the flow when TC times are measured. The exact points where measurement is done is highlighted by the big arrows. Start and end sequence parsing times are not included. The emphasis is on BCH decoding, packet and frame parsing and processing by application layer.

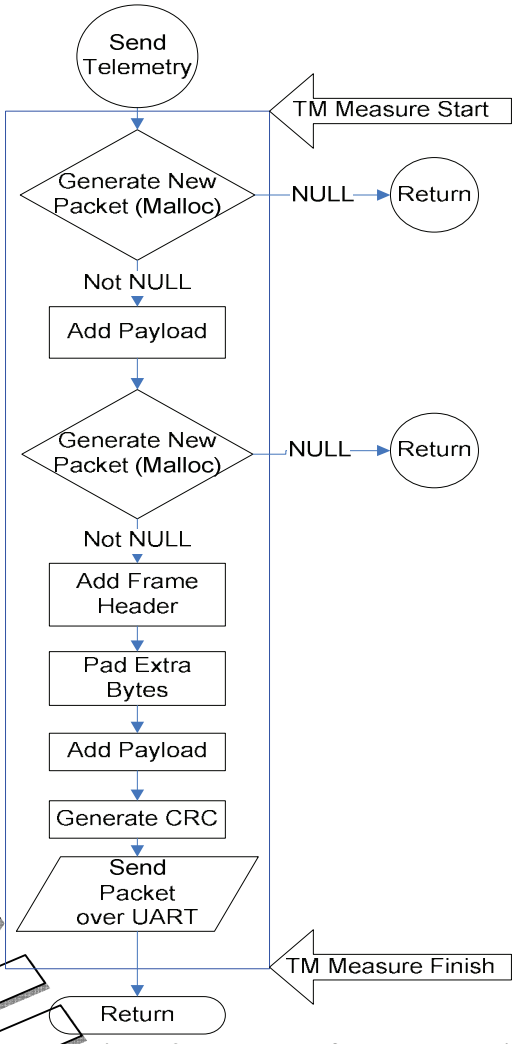

Figure 9- Flow chart for TM processing

Similarly figure 9 shows TM processing flow. For TM case the time for sending bytes over UART is subtracted from analysis as it should be almost same for a fixed baud rate.

Table 1 - Timing execution results with STR7@ 32MHz and 386Ex at 33MHz

Metric	STR710	386Ex RAM		STR Speed compared to 386Ex
Speed of core in MHz	32	33	MHz	-
TM Transmission + Processing Time	289	289.6	ms	-
TM Processing/Preparation time only	1.5	2.1	ms	1.40
TC Processing and Execution time	15.5	38.9	ms	2.51

Table 2 shows the test results when the STR7 was slowed down to 8MHz only with 386Ex at 33MHz. 386Ex is naturally faster. TC processing is seen to be 4 times slower on STR7 in this scenario. Table 3 and 4 are same experiment as in table 1 and 2 except the code for 386Ex was burnt to EEPROM and executed from there instead of RAM. The EEPROM device used is not very fast and incurs one extra wait state compared to SRAM. It is also an 8 bit parallel access device compared to 16 bit SRAM. Due to these two factors the performance is almost degraded by more than five times for 386Ex when STR runs at same clock rate. Figure 10 shows the summary of all gathered timing data.

Table 2- Timing execution results with STR7@ 8MHz and 386Ex at 33MHz

Metric	STR710	386Ex RAM		STR Speed compared to 386Ex
Speed of core in MHz	8	33	MHz	-
TM Transmission + Processing Time	300	289.6	ms	-
TM Processing/Preparation time only	12.5	2.1	ms	0.17
TC Processing and Execution time	60.2	38.9	ms	0.65

Table 3 - Timing execution results with STR7@ 8MHz and 386Ex at 33MHz

Metric	STR710	386Ex EEPROM		STR Speed compared to 386Ex
Speed of core in MHz	8	33	MHz	-
TM Transmission + Processing Time	300	296.2	ms	-
TM Processing/Preparation time only	12.5	8.7	ms	0.7
TC Processing and Execution time	60.2	82.96	ms	1.38

Table 4 - Timing execution results with STR7@ 32MHz and 386Ex at 33MHz

Metric	STR710	386Ex EEPROM		STR Speed compared to 386Ex
Speed of core in MHz	32	33	MHz	-
TM Transmission + Processing Time	289	296.2	ms	-
TM Processing/Preparation time only	1.5	8.7	ms	5.8
TC Processing and Execution time	15.5	82.96	ms	5.35

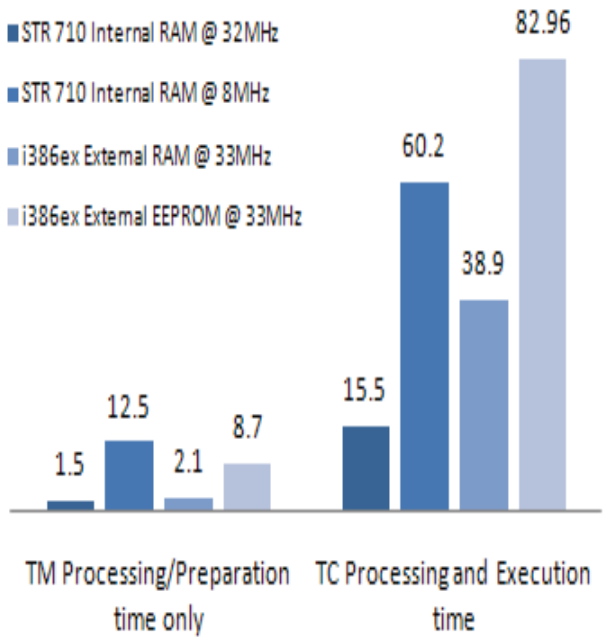

Figure 10 - Summary of execution times for both TM and TC Operations

6 CONCLUSION

Performance results are gathered for a custom CCSDS stack implemented in C/C++ for two different embedded platforms. It is important to know parameters like, execution times and memory footprints of software libraries when they are reused in other projects. The stack library developed here can be reused as a building block for a larger application targeting a specific mission. The results generated experimentally should be useful as they are parameterized in terms of execution speed. The effects of variations in the hardware platforms used are quantified experimentally. The variations studied include access times and bus width for memory and main clock speed for the embedded controllers.

Execution times of developed stack (under similar conditions) prove to be faster for STR710 compared to 386Ex possibly because of nature of code being more suitable for a RISC processor. Results can help in deciding if further optimization is needed or not. Also it can provide information in case a completely new solution is needed. It is also observed that for low power and resource constrained devices (as in space applications) time critical software could be executed from faster SRAM instead as compared to generally slower FLASH or EEPROM devices, to get maximum performance.

REFERENCES

[1] Overview of Space Communication Protocols, CCSDS Informational Report, 130.0-G-2, December 2007

[2] "Telemetry Channel Coding", CCSDS 101.0-B-6-S, October 2002

[3] "Packet Telemetry", CCSDS 102.0-B-5-S, November 2000

[4] "TM Synchronization and Channel Coding", CCSDS 131.0-B-1, September 2003

[5] "TM Space Data Link Protocol", CCSDS 132.0-B-1, September 2003

[6] "Space Packet Protocol", CCSDS 133.0-B-1, September 2003

[7] "Communication Operation Procedure-1", CCSDS 232.1-B-1

[8] "Telecommand Part 1: Channel Service", CCSDS 201.0-B-3-S, June 2000

[9] "Telecommand Part 2: Data Routing Service", CCSDS 202.0-B-3-S, June 2001

[10] "Telecommand Part 2.1: Command Operation Procedures", CCSDS 202.1-B-2-S, June 2001

[11] "Telecommand Part 3: Data Management Service", CCSDS 203.0-B-2-S, June 2001

[12] "TC Synchronization and Channel Coding", CCSDS 231.0-B-1, September 2003

[13] J.Rutter, T.Vladimirova and H.Tiggeler. "A CCSDS Software System for a Single-Chip On-Board Computer of a Small Satellite", *Proceedings of 15th AIAA/Utah Sate University Conference on Small Satellites*, Utah, USA, August 13-16 2001, SSC01-VI-4.

[14] "386Ex Processor User Manual", Intel Corporation

[15] "STR710-Eval Board User Manual", STMicroelectronics, September 2005

"STR71xF Series Controllers Data Sheet", STMicroelectronics, August 2006