TTEthernet Development and CFS Integration

DSG CFS / Open SW Multi-lateral TIM
NASA JSC, Houston, TX
11 – 13 Sept. 2017

Andrew Loveless (NASA/JSC)
andrew.loveless@nasa.gov
TTE Chip-IP

**Gen. 2 (Phoenix IP)**
- MAC layer only
- VL-based, no ports
- Proprietary loader
- ARINC 664-p7 like RC traffic
- Includes TTTTech and Honeywell IP
- Custom ASICs
- High-Integrity end systems and switches

**Gen. 3 (Pegasus IP)**
- UDP/IP layers in hardware
- Port-based with shared VLs
- ARINC 615A data loader
- ARINC 664-p7 RC traffic
- TTTTech Chip-IP and ASICs
- Plastic package with ceramic planned (QML-V)
- High-Integrity switches only
TTE Chip-IP

Gen. 2 (Phoenix IP)
- MAC layer only
- VL-based, no ports
- Proprietary loader
- ARINC 664-p7 like RC traffic
- Includes TTTech and Honeywell IP
- Custom ASICs
- High-Integrity end systems and switches

Gen. 3 (Pegasus IP)
- UDP/IP layers in hardware
- Port-based with shared VLs
- ARINC 615A data loader
- ARINC 664-p7 RC traffic
- TTTech Chip-IP and ASICs
- Plastic package with ceramic planned (QML-V)
- High-Integrity switches only

We Are Using This
A TTEthernet network is composed of end systems and switches.

The end system contains the host and network controller (NIC).

TTE Dataflow Concept

- User Partitions
- Platform Services
- Device Partitions
- Dataports
- Queues / Buffers

TTEthernet Network
Partitioned Traffic Flows

All links are bidirectional
TTE Dataflow Concept (ports)

- TTE dataports represent the end points for all communication.
- Ports are Sampling or Queuing, and use UDP, IPv4, or MAC layers.

**User Partitions**

- Partition A
- Partition B
- Partition C
- Partition D

**Platform Services**

- OS

**Device Partitions**

- TTE NIC
  - Partition A
  - Partition B
  - ... (Ports P1, P2, P3, P4)

**Dataports**

- Queues / Buffers

**Buffers**

- Buffers depend on port type

**Physical Network**

- TTEthernet Network
  - Partitioned Traffic Flows

**Ports are directional**

- TT means less buffering needed
TTE Dataflow Concept (VLs)

- Ports are connected to one another by Virtual Links (VLs).
- Virtual Links are logically point-to-point – one sender, N receivers.

Traffic is TT, RC, or static BE

VLs provide QoS
**TTE Dataflow Concept (VLs)**

- Traffic from different VLs is multiplexed over the same cabling.
- Logically the same as ARINC 429, except the “links” aren’t real.

![Diagram showing TTE Dataflow Concept]

**ARINC 429**

**TTE / ARINC 664**
 Ports are grouped in device partitions (8 per end system).
 Different partitions can be accessed simultaneously – no mutexes.
Device partitions can be mapped to the RTOS partitions. Thus partitioning is end to end – from app → network → app.
The **TTE** application enables TTE communication in CFS.
It includes pre-compiled drivers for supported targets and OSs.
It provides API to claim ports and send or receive messages.

The diagram shows the TTE application's integration into the TTE LIB and TTE APP. The TTE LIB provides services such as checksum, limit checker, file manager, scheduler, and SBN. The TTE APP connects to the inter-task software bus, which includes software bus, time services, executive services, event services, and table services. The TTE APP also integrates with C&DH, GN&C, cFS, and cFE applications. The diagram illustrates function calls and SB messaging between these components.

Receives commands and generates telemetry.
TTE CFS App (design)

Port Mapping Tables:
• Defines information needed to access local ports.
  – Types, direction, payload, etc.
• CFS apps claim ownership of ports via the API.

ES Config. Tables:
• Defines end system binary data resulting from scheduling process.

Commands:
• TTE_SEND_HK_MID
• TTE_CMD_MID
  – TTE_NOOP_CC
  – TTE_RESET_CC
  – TTE_SEND_DIAG_CC
• Commands for table config may be added in the future.

Main App Info:
• Sources: 6
• Headers: 13 + 8
• SLOC: 6,521

Telemetry:
• TTE_HK_TLM_MID
  – General state, status, counters, etc.
• TTE_DIAG_TLM_MID
  – Detailed port status information.

Library Info:
• Sources: 22
• Headers: 26
• SLOC: 12,161
• OSs: Linux, VxWorks
  – Xenomai is planned.

Driver Info:
• OSs: Linux, VxWorks
  – Xenomai is planned.
  – Arch.: x86, PowerPC

Prebuilt Drivers

Interrupt

SB Messages

Function Calls

Input / Output

HK, TO

IRQ Handler Task

Waits on semaphore

TTE Network Controller

TTE App

Main Task

Apps

Libraries and API

Library Info:

Main App Info:

Commands:

Telemetry:

Port Mapping Tables:

ES Config. Tables:

Drivers

CI

Input / Output

Interrupt

SB Messages

Function Calls
Port Mapping Tables:
- Defines information needed to access local ports.
  - Types, direction, payload, etc.
- CFS apps claim ownership of ports via the API.

ES Config. Tables:
- Defines end system binary data resulting from scheduling process.

Commands:
- TTE_SEND_HK_MID
- TTE_CMD_MID
  - TTE_NOOP_CC
  - TTE_RESET_CC
  - TTE_SEND_DIAG_CC
- Commands for table config may be added in the future.

Telemetry:
- TTE_HK_TLM_MID
  - General state, status, counters, etc.
- TTE_DIAG_TLM_MID
  - Detailed port status information.

Driver Info:
- OSs: Linux, VxWorks
  - Xenomai is planned.
- Arch.: x86, PowerPC

Main App Info:
- Sources: 6
- Headers: 13 + 8
- SLOC: 6,521

Library Info:
- Sources: 22
- Headers: 26
- SLOC: 12,161
- OSs: Linux, VxWorks
  - Xenomai is planned.

Prebuilt Drivers

Apps

Libraries and API

TTE App Main Task

SB Messages
Function Calls
Input / Output
Interrupt

IRQ Handler Task

Waits on semaphore

TTE Network Controller

CI

HK, TO

Andrew Loveless (NASA JSC/EV2)  DSG CFS / Open SW Multi-lateral TIM– 9/2017  Slide: 12/22
SCH_TT CFS App

- **SCH_TT** is a drop-in replacement for the standard **SCH** scheduler.
- Drives the FSW execution according to scheduled interrupts.
- Aligns the task scheduling with the TT message scheduling.
SCH_TT CFS App (design)

Local messages are aligned across the network = eliminates effect of drifting local clocks.

Main App Info:
- Sources: 7
- Headers: 18
- SLOC: 6,127

Message Tables:
- Defines contents of software bus messages to send during scheduled activities.

Schedule Tables:
- Defines activities to perform periodically.
- Organized into major and minor frames.

Uses API to register interrupts, claim ports, and send/receive messages.

Waits on semaphores to step through schedule table.

Commands:
- SCH_TT_SEND_HK_MID
- SCH_TT_CMD_MID
  - SCH_TT_NOOP_CC
  - SCH_TT_RESET_CC
  - SCH_TT_ENABLE_ENTRY_CC
  - SCH_TT_DISABLE_ENTRY_CC
  - SCH_TT_ENABLE_GROUP_CC
  - SCH_TT_DISABLE_GROUP_CC
  - SCH_TT_MODE_ENABLE_CC
  - SCH_TT_MODE_DISABLE_CC

Telemetry:
- SCH_TT_HK_TLM_MID
  - General state, minor frame, major frame, virtual time, etc.

Scheduled Messages:
- Wakeup and housekeeping request messages defined in message table.

SB Messages
- Function Calls
- Input / Output
- Interrupt

CFS Major Frame = TTE Communication Cycle
## Telemetry Example

### Slide: 15/22

#### DSG CFS / Open SW Multi-lateral TIM – 9/2017

[Image of telemetry example with a focus on data tables and diagrams.

**Data Table 1:**
- **Column Headers:** Destination, Port Type, Port Mode, Time Stamp, Event, MSG, REC, ERR, CMD, DRCP, Time Last SDT, Time Last REC
- **Data Entries:**
  - `(DSG_CFS/OPEN_SW_MULTI_LATERAL_TIM)`
  - `(9/2017)`

**Data Table 2:**
- **Column Headers:** Destination, Port Type, Port Mode, Time Stamp, Event, MSG, REC, ERR, CMD, DRCP, Time Last SDT, Time Last REC
- **Data Entries:**
  - `(DSG_CFS/OPEN_SW_MULTI_LATERAL_TIM)`
  - `(9/2017)`

**Diagram Overview:**
- **Legend:**
  - DSG CFS
  - Open SW Multi-lateral TIM
- **Key Elements:**
  - Telemetry multicasting
  - Synchronization
  - Data processing

**Additional Information:**
- **Authors:** Andrew Loveless (NASA JSC/EV2)
- **Presentation Details:**
  - **Title:** Telemetry Example
  - **Date:** 9/2017

[End of document]
Telemetry Example (cont.)
Telemetry Example (cont.)
App Availability

- Both planned to be open source, but are not yet.
- Both are fully documented.
  - Doxygen used for all sources.
- So far only tested on non-partitioned operating systems.
  - Linux, RTLinux, VxWorks.
- Currently only supports PowerPC and x86 targets.
Network and FSW Scheduling

Network Scheduling

Knowledge Needed:
1. Physical properties of devices / platforms
2. Physical interconnect between platforms
3. Message flows and timing properties
4. Interrupt types and timing requirements
5. Port to device partition mapping

TTE Network Description (XML):
1. Network properties – e.g. sync, links, planes
2. Device properties – e.g. dataports, partitions
3. Virtual Links – e.g. periodicity, payload sizes

TTE Toolchain (TTTech):
- Converts network description to device-specific configurations
- Converts device configurations to .hex and .bin image files

Flight Software Scheduling

Knowledge Needed:
1. Task assignment to platforms and partitions
2. Timing requirements for task execution
3. Message structure and packet formats

Undefined data exchange format

SysML Model

Project Database

CCDD
Command & Data Dictionary

TTE Toolchain (NASA):
- Converts .hex images to CFS end system config tables
- Converts device configs and network description to CFS port mapping tables.

TTE Toolchain (NASA): (CFS Tables for SCH_TT App)

CFS Tables for TTE App
Integrating with Other Buses

TTE Hardware Status

- TTTech developing space ASIC for switches and end systems.
  - Testing plastic ASIC on the boards now.
  - Ceramic samples scheduled for Feb. 2018.
  - Fully qualified ceramic (QML-V) planned for Q3/2018.

  - Will perform network-level testing.
  - Electrical testing will have already been done.

- TI is in final development of DP83561R rad-tolerant PHY in ceramic.
  - First samples available in Q1/2018.

- Focus now is on 100/1000Base-T.
  - Some plans for Base-CX down the line.
Questions?