Various Emerging Time-Triggered Protocols for Drive-by-Wire Applications

Syed Masud Mahmud, Ph.D.
Electrical and Computer Engg. Dept.
Wayne State University
Detroit MI 48202
smahmud@eng.wayne.edu

January 11, 2007

6th Annual Winter Workshop
U.S. Army Tank-Automotive RD&E Center, Warren, MI
Speaker’s Background

- Received Ph.D. in Electrical Engineering from the University of Washington, Seattle (1984).
- Has been working for Wayne State University since 1988.
- Published over 90 technical papers in referred journals and conference proceedings.
Speaker’s Current Areas of Interest

• Intelligent Vehicles,
• Intelligent Transportation Systems,
• In-Vehicle Networking,
• Time Triggered Protocols for Real-Time Applications,
• Collision Warning and Collision Avoidance systems,
• Security in Wireless Communications, and
• Embedded Systems.
Outline of the Talk

• Event-Triggered versus Time-Triggered Operations.
• Need for Time-Triggered Protocols.
• Time-Triggered Protocol (TTP).
• A Quick Overview of the CAN Protocol.
• Time-Triggered CAN (TTCAN) Protocol.
• The FlexRay Protocol.
• Comparison of TTP, TTCAN and FlexRay.
Event-Triggered Operations

TIME

Brake Message  Steering-Wheel Message  Brake Message  Steering-Wheel Message

Brake Event  Steering Wheel Event  Brake Event  Steering Wheel Event
Problems with Event-Triggered Communications

• Message latency is not deterministic.
• Though the maximum latency could be quantified, the *latency jitter* can’t be eliminated.
• Lower bus utilization to avoid high latencies.
• *Babbling idiot* problems due to faulty nodes. A higher priority faulty node can monopolize the bus.
Time-Triggered Operations

TIME

| Brake Message | Steering-Wheel Message | Vehicle Speed | Brake Message | Steering-Wheel Message | Vehicle Speed |

Emerging Time-Triggered Protocols
Benefits of Time-Triggered Communications

• *Deterministic* latency.
• *No latency jitter*.
• Very *high bus utilization*.
• *High system throughput* due to high bus utilization.
• A faulty node can’t monopolize the bus. Thus, there is *no Babbling Idiot* problem.
Need for Time Triggered Protocols
Hard Real-Time Distributed Control System

• Hard real-time systems such as x-by-wire systems need deterministic latencies.
• In an event-triggered communication, though the safety-critical messages can be prioritized to keep the latencies under a bound, the latency jitters of these messages can’t be eliminated.
Latency Jitter

Latency Jitter vs. Average Latency

Maximum Allowed Latency for Safety Critical Applications

Number of Nodes

Latency
Effect of Latency Jitters on Hard Real-Time Distributed Control System

• Control applications do not run well under latency jitters.
• Latency jitters produce **overshoots** in the output of a control application.
• If the latency jitter is too high, then the control system may even become **unstable**.
• Thus, Time-Triggered protocols are appropriate for hard real-time systems.
Time-Triggered Protocol (TTP)
A TTP/C Cluster

CNI: Communication Network Interface

Emerging Time-Triggered Protocols

© 2007 Syed Masud Mahmud
TTP/C Network Topology

Bus

Star

Starcoupler device for one channel

© 2007 Syed Masud Mahmud
TTP/C Network Topology (contd.)

Multi-Star
TTP/C Network Topology (contd.)

Star/Bus Combination
Overview of an Electronic Module

MEDL: Message Descriptor List
Structure of the Message Descriptor List (MEDL)

- Each TTP node includes a globally known MEDL that controls the cluster’s scheduling policy. (A detailed description of the MEDL is given later.)
Frame-Application Data-Message

Application data:
produced/consumed by the host

Messages:
data interpreted by the host (e.g. 12 bit sensor value)

Frame:
Bit stream transmitted on the channel

Protocol overhead:
frame type, explicit C-state

Protocol overhead:
frame CRC

TTP

Message 1

Message 2

Message 3

CRC
Media Access Scheme

Node # | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0
Round Slot # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0
Frame Ch0
Frame Ch1

Node Slot
TDMA Round
Cluster Cycle
TDMA Round

© 2007 Syed Masud Mahmud
Emerging Time-Triggered Protocols
Multiplexed Slot Assignment

![Diagram of multiplexed slot assignment](image)
Slot Timing

- A TDMA slot begins with the start of the *Transmission Phase (TP)*. The time at which TP begins is called the *Action Time (AT)*.
- The evaluation of received data and execution of protocol services are done at *Post Receive Phase (PRP)*.
Slot Timing

- The **Pre-Send Phase (PSP)** is necessary to load the schedule information of the next slot from the MEDL and to prepare the transmission/reception of data at the next AT. Thus, the start of PSP is the start of node slot.
- The **Idle Phase** is used as tolerance if extra time is need for PRP.
Slot Timing

- A node performs all housekeeping operations during the Inter-Frame Gap (IFG).
- During IFG there is no traffic on the bus.
Controller State (C-state)

• It describes the internal state of a TTP/C controller.
• Although the C-state is calculated locally by each controller, it represents a global view of the cluster and must agree with the calculated C-states of the other nodes.
• A C-state disagreement between a node and the majority of the other nodes will mark this node as faulty.
Controller State (C-state)

The C-state consists of the following items:

**Global Time** The global time of the transmission of the current node slot at the start of PSP.

**Round Slot Position** The current slot in the cluster cycle updated at the start of the node slot (begin of PSP).

**Cluster Mode** The current active cluster mode the controller is operating in, updated at the start of the node slot. (Examples of cluster modes for an aircraft are *on ground, take off, cruising* and *landing.*)
Controller State (C-state)

Deferred Pending Mode Changes (DMC)
If a mode change is requested for the start of the next cluster cycle, the pending request is saved in the C-state, updated at the end of the PRP after the request is received.

Membership Information
Consistent view of the activity of all nodes in the cluster. A node is in the membership after having correctly sent in its last slot. A node is outside the membership, if having not sent or sent incorrectly.
Frame Types and Formats

A frame consists of the following parts:

- Application data prepared by the host.
- Explicit C-state for agreement check between the nodes.
- Mode change requests.

**Explicit C-state in frame**

<table>
<thead>
<tr>
<th>Frame Type</th>
<th>Mode change request</th>
<th>C-state</th>
<th>Application data</th>
<th>CRC</th>
</tr>
</thead>
</table>

**Implicit C-state in frame**

<table>
<thead>
<tr>
<th>Schedule ID</th>
<th>Frame Type</th>
<th>Mode change request</th>
<th>Application data</th>
<th>C-state</th>
<th>CRC</th>
</tr>
</thead>
</table>

Data explicit in frame and included in the CRC

Data not in the frame, but included in the CRC
Frame Types and Formats

• Additional frame description data, e.g. a frame type identifier.
• CRC calculated over the frame for detection of errors.
Frame Types and Formats

- Implicit C-state calculated into the frame CRC, if no explicit C-state is transmitted.
- Unique cluster schedule ID to guarantee that only nodes with the same schedule can communicate with each other.

![Diagram of frame types and formats](image-url)
Cold Start Frame

This frame is used for the cluster startup and must contain the following information:

- Global time of sending (C-state time)
- Round slot position of the sender (must be located in the first TDMA round in the first cluster mode)
Cold Start Frame

- Frame type identifier, which indicates this frame as cold start frame
- CRC calculated over the frame.
Cold Start Frame

• Unique cluster schedule ID is also calculated into the frame’s CRC

• Other nodes may integrate on this frame by taking over the C-state time and the schedule position. The sender is identified by the round slot position and is set into the membership.
Explicit versus Implicit C-state

- Implicit C-states improves performance, because no bandwidth on the channel is spent for the C-state transmission. The saved bandwidth can be used to send more application data.

- On the other hand the logic for comparing the local C-state with the received implicit one is more complex at the receiver and may take a longer processing time.

- Explicit C-states are required for the integration process and cluster startup.
Startup of a Node

The startup of each node consists of three steps:

- Initialization of the host and the TTP/C controller.
- Listening on the channels for frames with explicit C-state for the duration of the listen timeout. If such a frame is received, the controller starts the integration to adopt the C-state.
Startup of a Node

The startup of each node consists of three steps:

- If no running cluster is detected, it checks the conditions for performing a cold start:
  - the host has updated its life-sign
  - the *Cold Start Allowed flag (CF)* is set in the MEDL
  - the maximum number of allowed cold starts for this node is not reached

If these conditions are fulfilled, the node sends a cold start frame, increases the *cold start counter* and processes the TDMA scheme.
Integration

• Integration is the process of a node to synchronize with a running cluster.

• The node listens on the channels to receive a frame with explicit C-state or a cold start frame and then integrates on it.

• The cluster schedule must provide at least a minimum of one frame per channel with explicit C-state every two TDMA rounds (minimum listen timeout).
Emerging Time-Triggered Protocols

Integration

Observed Channel: Channel on which traffic is detected first.

[Flowchart diagram with decision points and actions described in text]

Protocols
Big Bang

• A mechanism that ensures that in case of a startup collision between two cold start nodes no node will integrate on any of the collided frames.

• If *Propagation Delay > Cold Start Frame Duration* different nodes may integrate on different cold starters.

• To prevent this problem the first received cold start frame is rejected by all nodes and the listen timeout is restarted.
Big Bang (contd.)

- The cold start nodes will not detect any traffic during their TDMA round and start their startup timeouts after they get a *Communication Blackout*\(^1\) error.

- Because the difference between any startup timeouts is at least a slot duration and therefore longer than the maximum propagation delay, no further inconsistent collision will appear.

\(^1\)Communication Blackout: Error raised by the controller if no traffic was observed on any channel during one TDMA round
Startup

- Node A
- Node B
- Node C
- Node D
- Node E

long distance

long distance

propagation delay > cold start frame duration
Startup Collision Scenario – Without Big Bang

- Node A
  - Frame A
  - Frame A
  - Sending, ignore E

- Node B
  - Frame A
  - Frame A
  - No collision detected, ignore E
  - Sync on cold start A

- Node C
  - Frame A
  - Frame E
  - Detect collision, restart listen timeout
  - Correct behavior

- Node D
  - Frame E
  - Frame A
  - No collision detected, ignore A
  - Sync on cold start E

- Node E
  - Frame E
  - Frame A
  - Sending, ignore A

Transmission Phase > max prop. delay + frame duration
- Send cold start frame
- Receive frame
Startup Collision Scenario – Using Big Bang

Node A
- Frame A
- Frame A
- Frame A
- Frame A
Sending, ignore E
Perform cold start round
Comm. Blackout
Startup timeout A
Sending

Node B
- Frame A
- Frame E
- Frame A
- Frame E
Ignore E, reject correct A
Restart listen timeout
Listen timeout B
Sync on cold start A

Node C
- Frame A
- Frame E
- Frame A
- Frame E
Detect collision,
Restart listen timeout
Listen timeout C
Sync on next frame

Node D
- Frame E
- Frame A
- Frame E
- Frame A
Ignore A, reject correct E
Restart listen timeout
Listen timeout D
Sync on cold start A

Node E
- Frame E
- Frame A
- Frame E
- Frame A
Sending, ignore A
Perform cold start round
Comm. Blackout
Startup timeout E
Sync on cold start A
Host/Controller Life-Sign

• The host must provide the controller with a life-sign before each sending slot of the node.

• If the life-sign is not updated by the host, the controller does not send but remains synchronized with the cluster.

• An updated host life-sign is also a precondition for a controller to startup a cluster by sending a cold start frame.
Membership

• The *membership service* informs all nodes of a cluster about the operational state of each node within a latency of about one TDMA round.

• A node is operational at time *now*
  – If the host computer of the node has updated its life-sign within the last TDMA round and
  – the communication controller is operating and synchronized with the rest of the cluster.
Membership Vector

- The membership status of each *member node* is recorded in the membership vector.
- The membership vector is an array of flags with a length to equal to the number of member nodes in the cluster.
- The membership flag of a node indicates whether or not the node is alive.

<table>
<thead>
<tr>
<th>MEMB(A)</th>
<th>MEMB(B)</th>
<th>MEMB(C)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>-</td>
</tr>
</tbody>
</table>

Membership Flag

<table>
<thead>
<tr>
<th>Membership Vector</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMB(A) MEMB(B) MEMB(C)</td>
<td>- - -</td>
</tr>
</tbody>
</table>
Acknowledgment Scheme

• The TTP/C acknowledgment service is based on the membership service.
• Acknowledgment of the receipt of a frame by the successor of the sender is performed implicitly through the node membership in the C-state.
• During a sending slot a node sends two frames. If any of these two frames is received correctly by a receiver, the receiver considers the sender to be active.
Acknowledgment Scheme

• If the successor of node A, say node B, has received a correct frame from A, MEMB(A) is TRUE in B’s transmission.
• Thus A can use B’s frame for acknowledgment.
• During normal operation, A performs two CRC checks over B’s frame.
  – **Check 1a:** A sets MEMB(A) to TRUE and MEMB(B) to TRUE. It then performs the CRC check over the received frame, concatenated with its local C-state.
  – **Check 1b:** A sets MEMB(A) to FALSE and MEMB(B) to TRUE. Then it performs the CRC check over the received frame, concatenated with its local C-state.
Acknowledgment Scheme

• If the CRC check Ia is OK, node A assumes that the transmission was correct and remains in the membership.
• If the CRC check Ia is not OK, then at least one failure has occurred.
• If both checks (check Ia and check Ib) fail, it is assumed that a transient disturbance has corrupted B’s frames or B is not operational at all.
Acknowledgment Scheme

• If check Ia to fail and check Ib to pass, then it is not certain whether A or B is correct.

• A final decision of the cause is made after receiving the next frame from A’s second successor, say node C.
Acknowledgment Scheme

transmission of B

CRC la: 1 1 X

true

A acknowledged

CRC lb: 0 1 X

false

A failed OR B failed?

B failed, take C as successor

© 2007 Syed Masud Mahmud

Emerging Time-Triggered Protocols
Acknowledgment Scheme

• In the next slot, A performs two CRC checks over the frames received from C.
  – **Check Ila:** A sets MEMB(A) to TRUE and MEMB(B) to FALSE. It then performs the CRC check over the received frame from C.
  – **Check IIb:** A sets MEMB(A) to FALSE and MEMB(B) to TRUE. It then performs the CRC check over the received frame from C.
Acknowledgment Scheme
Message Descriptor List (MEDL)

- Each controller has an MEDL whose layout looks like the one shown below.
Schedule Parameters

**Cold Start Allowed Flag (CF)** If this flag is set the node can enter the cold start state.

**Cold Start Integration Allowed Flag (CIA)** The controller may only integrate on a cold start frame if this flag is set. Otherwise the controller must wait for a regular frame with explicit C-state to integrate.
Schedule Parameters (contd.)

Minimum Integration Count (MIC) A controller needs a frame with explicit C-state from another node to integrate into a cluster. To ensure that controllers do not integrate on faulty frames and re-transmit the faulty C-state in their own subsequent transmissions, a specific number of correct frames (only the first one has to have an explicit C-state) has to be received before a node can transmit actively.
Schedule Parameters (contd.)

Maximum Cold Start Entry This field specifies the maximum number of cold start attempts by the controller.

Maximum Acknowledgment Failure Count (MAFC) This field specifies the number of successive acknowledgment failures at which the controller freezes.
Schedule Parameters (contd.)

Communication Bit Rates Used to define the bit rates on the two channels of the cluster.

Startup Timeout Startup timeout for the node.

Listen Timeout Listen timeout for the node.

Flag Position in Membership Vector Own flag position in the membership vector.
Identification Section

Cluster Schedule Identification This field contains an identifier of the cluster design. The cluster designer (tool) has to ensure that the schedule IDs differ for different designs. Only nodes based on the same cluster design can participate in the same cluster.

Application Identification Unique ID used by the host application to verify the actual MEDL is compatible to it.
Round Slot Selection

Sending Slot Flag (SS) Marks that the node is allowed to send in this slot.

CNI Addresses of Frames Defines the address of the application data transmitted in the slot.

Slot Duration The slot duration entry contains the duration of the current node slot in units of macroticks.
Round Slot Selection

Frame Types Marks the type of the frames transmitted in the round slot (e.g. with implicit or explicit C-state).

Application Data Length of Frames Defines the length of the application data in the frames.

Clock Synchronization (ClkSyn) Flag If this flag is set, the clock synchronization algorithm is executed in this slot.
Round Slot Selection (contd.)

Synchronization Frame (SYF) Flag If the synchronization frame flag is set, the frames arriving in the slot are used by the clock synchronization algorithm: For these frames the deviation between the expected and the actual arrival time of the frame is measured and put onto the clock synchronization stack.

Mode Change Permissions (MCP) Permission for a mode change request to a specified cluster mode.

Flag Position in Membership Vector This entry contains the flag position of the sender in the membership vector.
Protocol States

Freeze (0) The execution of the protocol is halted until the controller is turned on by setting the Controller On (CO) flag.

Init (1) The controller enters the init state after it is switched on (CO flag is set by the host).

Listen (2) The controller enters the listen state when the initialization has been completed and tries to integrate on a received frame.

Cold Start (3) The controller enters the cold start state if it is allowed to (CF flag set and host alive).
Protocol States (contd.)

Active (4) The controller transmits frames.

Passive (5) The controller is synchronized but has not yet acquire a node slot or it has lost the slot acquirement due to a fault. Reasons for that are:

– The host is inactive (no life-sign update).
– A failed host (Host Error).
– Loss of membership due to acknowledgment algorithm decision.
– The node has just integrated but not yet sent.
– The node is permanently passive.
Protocol States (contd.)

**Await (6)** Upon request of the host, the controller is waiting for download or other implementation dependent asynchronous services instead of executing the TTP/C protocol.

**Test (7)** The controller is performing self tests upon a request of the host processor.

**Download (8)** The controller executes download, i.e., it retrieves a new MEDL or new application data for the host processor via the TTP/C bus. It does not execute the TTP/C protocol at that time.
## Summary of Protocol States

<table>
<thead>
<tr>
<th>No.</th>
<th>State</th>
<th>CO flag</th>
<th>Sending</th>
<th>Synchronized</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Freeze</td>
<td>Off</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Init</td>
<td>On</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Listen</td>
<td>On</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Cold Start</td>
<td>On</td>
<td>Cold start frames</td>
<td>Yes (^1)</td>
</tr>
<tr>
<td>4</td>
<td>Active</td>
<td>On</td>
<td>Normal frames</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Passive</td>
<td>On</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Await</td>
<td>On</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Test</td>
<td>On</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Download</td>
<td>On</td>
<td>Download</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

\(^1\) after having received one correct frame
Protocol State Transitions
### State Transition Events

<table>
<thead>
<tr>
<th>No.</th>
<th>From</th>
<th>To</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Init</td>
<td>Listen</td>
<td>Initialization complete</td>
</tr>
<tr>
<td>2</td>
<td>Listen</td>
<td>Passive</td>
<td>Valid frame containing explicit C-state received</td>
</tr>
<tr>
<td>3</td>
<td>Listen</td>
<td>Cold Start</td>
<td>Listen timeout expired, cold start allowed and host life-sign updated</td>
</tr>
<tr>
<td>4</td>
<td>Active/Passive</td>
<td>Freeze</td>
<td>Clique error, Communication blackout, Synchronization error, Membership error, periodic MEDL CRC check failed</td>
</tr>
<tr>
<td>5</td>
<td>Listen/Cold Start/Active/Passive/Init/Await/Test/Download</td>
<td>Freeze</td>
<td>CO flag turned off</td>
</tr>
<tr>
<td>6</td>
<td>Cold Start</td>
<td>Active</td>
<td>Controller in majority clique (after clique avoidance), at least 1 correct frame received (min. 2 controllers alive), host life-sign updated, no host error</td>
</tr>
</tbody>
</table>
### State Transition Events (contd.)

<table>
<thead>
<tr>
<th>#</th>
<th>State Transition</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Cold Start</td>
<td>Listen</td>
<td>Clique error, host life-sign not updated, max. cold start entries exceeded or traffic detected on observed channel during startup timeout</td>
</tr>
<tr>
<td>8</td>
<td>Active</td>
<td>Passive</td>
<td>Host life-sign not updated, host error, acknowledgment failed</td>
</tr>
<tr>
<td>9</td>
<td>Passive</td>
<td>Active</td>
<td>Node slot acquired, host alive</td>
</tr>
<tr>
<td>10</td>
<td>Freeze</td>
<td>Test</td>
<td>Self tests requested, CO flag set. This transition is optional and only valid if the controller supports self-tests</td>
</tr>
<tr>
<td>11</td>
<td>Init</td>
<td>Freeze</td>
<td>MEDL CRC check failed</td>
</tr>
<tr>
<td>12</td>
<td>Freeze</td>
<td>Await</td>
<td>Await state requested, CO flag set. This transition is optional and only valid if the controller has an await state used as preparation for download</td>
</tr>
</tbody>
</table>
State Transition Events (contd.)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Freeze</td>
<td>Init</td>
<td>CO flag set, transition to test and await state not requested by host</td>
</tr>
<tr>
<td>14</td>
<td>Test</td>
<td>Freeze</td>
<td>Tests completed</td>
</tr>
<tr>
<td>15</td>
<td>Listen/Await</td>
<td>Download</td>
<td>Frame indicating Download Mode. The transition from listen to the download state is optional and not recommended for safety-critical systems</td>
</tr>
<tr>
<td>16</td>
<td>Download</td>
<td>Freeze</td>
<td>Download completed</td>
</tr>
<tr>
<td>17</td>
<td>Cold Start</td>
<td>Passive</td>
<td>Controller in majority clique but host life-sign not updated or host error (see transition 6)</td>
</tr>
</tbody>
</table>
A Quick Overview of the CAN Protocol
CAN Protocol

- CAN is a serial protocol.
- It supports distributed real-time control with high level of data integrity.
Dominant and Recessive Bits

Bus Values

- 0: Dominant bit, 1: Recessive bit.
- During simultaneous transmission of dominant and recessive bits, the resulting bus value will be dominant.
CAN Message Frames

**Data Frame:** It carries data from a transmitter to the receivers.

**Remote Frame:** It is transmitted by a node to request the transmission of a Data frame with the same identifier.

**Error Frame:** It is transmitted by any node after detecting a bus error.

**Overload Frame:** It is used to provide additional delay between two data or remote frames.
Data Frame

There are seven fields in a Data frame.

- **Start of Frame (SOF):** It marks the beginning of a data or remote frame. It consists of only one dominant bit.
Arbitration Field of Data Frame

- **Arbitration Field**: This field consists of the identifier and the RTR bit.
- **Identifier**: There are 11 or 29 bits in the identifier.
Arbitration Field of Data Frame

- The Identifier performs the following operations:
  - Determines the type of a message.
  - Performs acceptance test of a message.
  - Arbitrates & determines the priority of the message.
RTR Bit

- RTR bit (Remote Transmission Request Bit): In Data frames the RTR bit must be dominant. In Remote frames it must be recessive.

<table>
<thead>
<tr>
<th>RTR bit</th>
<th>Type of Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>Data Frame</td>
</tr>
<tr>
<td>r</td>
<td>Remote Frame</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>Arbitration Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF</td>
</tr>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>RTR Bit</td>
</tr>
<tr>
<td>Control</td>
</tr>
</tbody>
</table>
```
Control Field

- Out of the 6 bits, 4 bits are used to indicate the length of the data and the other two bits are reserved for future expansion.
- Reserved bits must be sent as dominant bits.
Data Field

- The data field contains the data to be transmitted.
- The data field can contain 0 to 8 bytes.
CRC Field

- It contains a 15-bit CRC Sequence followed by a 1-bit (a recessive bit) CRC Delimiter.
Acknowledge Field

- The ACK field consists of two bits.
- One bit for ACK slot and another bit for ACK delimiter.
- The TRANSMITTING node sends two recessive bits through the ACK field.
Acknowledge Field

- The RECEIVING node sends a dominant bit through the ACK slot after receiving a valid message.
- Since the CRC and ACK delimiters are both recessive bits, the ACK slot is always surrounded by two recessive bits.
End of Frame

- Each data frame and remote frame is delimited by a flag sequence consisting of seven recessive bits.
- This is a contradiction to the Bit Stuffing technique.
Time Triggered CAN: TTCAN
Time Triggered CAN (TTCAN)

- TTCAN is a higher layer protocol above the standard CAN protocol.
- A message that needs a guaranteed latency can be transmitted at a specific time slot, without competing with other messages for the bus.
Time Triggered CAN (TTCAN)

- The activities of a TTCAN system are determined by the progression of a globally synchronized time.
• Message “a” is sent if the system clock reaches 3 and 6 while message “b” is sent at 5, and message “d” is sent at 9.
TTCAN Time Windows

- There are three time windows in a TTCAN system:
  - Exclusive Window.
  - Arbitrating Window.
  - Free Window
Exclusive Window

- It is used by a message that needs guaranteed latencies.
- A particular Exclusive Window is reserved for a particular message.
- Thus, the message does not have to compete for the bus.
Arbitrating Window

- It is used by messages that can tolerate latency jitters.
- Like the standard CAN system, various messages compete for the bus during this time window.
- The highest priority message (among the competing messages) gets the bus.
Free Window

- It is reserved for the future expansion of the system.
- Depending upon the need (during system expansion) some Free Windows can be converted to new Exclusive Windows and others can be converted to new Arbitrating Windows.
Master Node

- A node called the Master Node is responsible for maintaining global time.
- The Master Node periodically broadcasts its local time (global time for the entire system) through a message called the Reference Message.
Reference Message

- All nodes read the global time from the reference message and then synchronize their own local time with the global time.
Potential Master Nodes

• There are several potential master nodes in a TTCAN system.
• If the potential master nodes detect the absence of the Reference Message within a timeout period, then they understand that the Master Node is dead.
Potential Master Nodes

- The potential masters then compete for the bus to become the new master.
- The highest priority active potential master will become the new master of the system.
- The new master will then start broadcasting reference messages.
• The period between two consecutive reference messages is called the **basic cycle**.
The System Matrix of a TTCAN System

- Several basic cycles are connected to build a matrix cycle called the System Matrix.
- This allows to combine multiple sending patterns, e.g. sending every basic cycle, sending every second basic cycle, or sending only once within the whole system matrix.
# System Matrix

## Basic Cycle 0

<table>
<thead>
<tr>
<th>Reference Message</th>
<th>Message A</th>
<th>Arbitra. Window</th>
<th>Message B</th>
<th>Free Window</th>
</tr>
</thead>
</table>

## Basic Cycle 1

<table>
<thead>
<tr>
<th>Reference Message</th>
<th>Message A</th>
<th>Message B</th>
<th>Arbitra. Window</th>
<th>Free Window</th>
</tr>
</thead>
</table>

## Basic Cycle 2

<table>
<thead>
<tr>
<th>Reference Message</th>
<th>Message C</th>
<th>Free Window</th>
<th>Message D</th>
<th>Arbitra. Window</th>
</tr>
</thead>
</table>

## Basic Cycle 3

<table>
<thead>
<tr>
<th>Reference Message</th>
<th>Message C</th>
<th>Arbitra. Window</th>
<th>Message D</th>
<th>Free Window</th>
</tr>
</thead>
</table>
Retransmission of a Faulty Message is not allowed in a TTCAN System

• In case of a fault, retransmission of a message is not allowed through any windows (Exclusive or Arbitrating).
• This is due to the fact that the retransmitted message may cross over the boundary of the current window.
Retransmission of a Faulty Message...

• Thus, in a TTCAN system, if a safety critical message cannot go through its Exclusive Window, then it will be delayed significantly. (Note: In TTP the message always goes through two channels. Thus, TTP can tolerate single channel fault.)

• In TTCAN, the message that could not go will have to wait until its next Exclusive Window.
Length of a TTCAN Window

- A TTCAN window must be long enough to carry its message plus possible error flags.
Core Partners of the FlexRay Consortium

**FlexRay: Basic Characteristics**

- Synchronous and asynchronous frame transfer.
- Guaranteed frame latency.
- Prioritization of frames during asynchronous transfer.
- Multi-master clock synchronization.
- Error detection and signaling.
- Error containment on the physical layer.
- Scalable fault tolerance.
A Differential Voltage Link

Signal Voltage = \( \text{Volt}_{BP} - \text{Volt}_{BM} \)
**Electrical Signaling**

*Idle_LP*: The Bus is in Idle Low Power Mode. Both BP and BM pins are at 0v.

*Idle*: The Bus is in Idle Mode. Both BP and BM pins are biased to a certain voltage.
Point to Point Terminated Connection

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBus</td>
<td>Length of a point-to-point connection</td>
<td>24</td>
<td>m</td>
<td></td>
</tr>
</tbody>
</table>
Network Topology

- Single Channel Passive Bus
- Dual Channel Passive Bus
- Single Channel Passive Star
- Dual Channel Passive Star
- Single Channel Active Star
- Dual Channel Active Star
- Various Combinations of Bus and Star Topologies
A Single Channel Passive Bus

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$lBus$</td>
<td>Maximum electrical distance between two nodes in the system</td>
<td>24</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>$lStubDistance_{MN}$</td>
<td>Distance between two network splices</td>
<td>150</td>
<td>mm</td>
<td></td>
</tr>
<tr>
<td>$nStubNodes$</td>
<td>Number of stub nodes</td>
<td>22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Dual Channel Passive Bus
A Single Channel Passive Star

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPassiveStar(_N)+IPassiveStar(_M)</td>
<td>Maximum electrical distance between two nodes in the system</td>
<td>24</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>nStarNodes</td>
<td>Maximum number of nodes in a passive star network.</td>
<td>22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Single Channel Active Star

- Node 1
- Node 2
- Node 3

Node 2 is connected to the Active Star through IActiveStar₂.

Node 1 is connected to the Active Star through IActiveStar₁.

Node 3 is connected to the Active Star through IActiveStar₃.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>IActiveStarₜₙ</td>
<td>Length of a branch from node N to the star</td>
<td>24</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>nActiveBranches</td>
<td>Number of branches at an active star</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Single Channel Cascaded Active Stars

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n\text{StarPath}_{M,N}$</td>
<td>Number of stars on the signal path from any node $M$ to a node $N$ in network with active stars</td>
<td>0</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$\text{IStarStar}$</td>
<td>Electrical distance between two active stars</td>
<td>24</td>
<td></td>
<td>m</td>
</tr>
</tbody>
</table>
A Single Channel Hybrid Topology

- **Active Star 1**
  - Node 1
  - Node 2

- **Active Star 2**
  - Node 3
  - Node 4
  - Node 5

- **Passive Star Network**
  - Node 7
  - Node 8
  - Node 9

- **Linear Passive Bus**
  - Node 6
A Dual Channel Active Star Network
A Dual Channel Cascaded Active Star Network

Diagram showing the network topology with nodes and stars connected.
A Dual Channel Hybrid Network

Node A  Node B  Node C  Node D  Node E

Channel A

Emerging Time-Triggered Protocols
Node Architecture

Communication Data
Configuration Data & Status Information
Synchronization Signals
Configuration Data & Status Information
Control Signals
Status Signals
Control Data & Status Information
Bus Guardian
Bus Driver
Communication Controller
Power Supply
Host
Bus
Bus

Wayne State University
College of Engineering
Emerging Time-Triggered Protocols
© 2007 Syed Masud Mahmud
120
Host – Communication Controller Interface

• The host provides control and configuration information to the CC, and provides payload data that is transmitted during the communication cycle.

• The CC provides status information to the host and delivers payload data received from communication frames.
Bus Guardian

- The bus guardian (BG) has *a-priori* knowledge of the transmission times of the node.
- If the bus guardian detects any *mismatch* between the schedules of the communication controller and the bus guardian, it *signals an error* condition to the host and inhibits any further transmission attempts.
FlexRay Frame Format
• The frame ID defines the slot in which the frame should be transmitted. (*Performs Arbitration*)
• Each frame that may be transmitted in a cluster has a frame ID assigned to it.
• The frame ID ranges from 1 to 2047. The frame ID 0 is an invalid frame ID.
Payload Length (7 bits)

- The payload length indicates the number of 16-bit words in the payload segment.
- Thus, the payload segment can contain 0 to 127 words or 0 to 254 bytes.
Header CRC (11 bits)

- The Header CRC is computed over the Sync Frame Indicator, Startup Frame Indicator, Frame ID and Payload Length.
Cycle Count (6 bits)

- This field indicates the current cycle number.
- A FlexRay communication cycle is similar to the TTCAN basic cycle and TTP TDMA round.
Trailer Segment: Frame CRC

- The Frame CRC field is computed over the header segment and the payload segment of the frame.
Trailer Segment: Frame CRC

- The frame CRC has a Hamming distance of six for payload lengths up to and including 248 bytes.
- For payload lengths greater than 248 bytes the CRC has a Hamming distance of four.
Communication Cycle

One Communication Cycle

Static Segment
For Time-Triggered Messages

Dynamic Segment
For Event-Triggered Messages

Symbol Window
For Special Signals such as Wake Up

Network Idle Time

It allows time to do clock corrections and housekeeping operations at a node.
Communication Cycle

One Communication Cycle

Static Segment  Dynamic Segment  Symbol Window  Network Idle Time

Similar to the Exclusive Window of the TTCAN system

Similar to the Arbitrating Window of the TTCAN System
Timing Hierarchy

communication cycle level

0 1 2 3 4 5 ...

vCycleCounter

ccycleCountMax

gdCycle

macrotick level

0 1 2 3 4 5 ...

vMacroTick

gMacroPerCycle - 1

gdMacroTick

microtick level

0 1 2 3 4 5 ...

vMicroTick

pdMicroTick

Emerging Time-Triggered Protocols
Microtick

- **Microticks** are local time units of a node.
- The granularity of a node’s local time is a microtick.
- Microticks may have different durations in different nodes. *(For example, in one node a microtick may be 10 ns and in another node it may be 5 ns.)*
- The microtick is derived directly from the node’s oscillator clock tick, optionally making use of a prescaler.
Macrotick

• The duration of a *macrotick* is equal throughout all synchronized nodes in a cluster.
• Each local macrotick is an integer number of local microticks.
• The number of microticks per macrotick may vary within the same node due to rate corrections.
• The number of microticks per macrotick may also differ between nodes, and depends on the oscillator frequency and the prescaler.
Length of a Communication Cycle (Cycle)

- A *cycle* consists of an integer number of macroticks.
- The number of macroticks per cycle shall be identical in all nodes in a cluster, and remains the same from cycle to cycle.
- At any given time all nodes should have the same cycle number.
Timing Hierarchy in a Communication Cycle

An action point (AP) is a time instant at which a node performs a specific action, e.g. transmits a frame.
Structure of the Static Segment

- The static segment contains a number of static slots.
- For a given system, the number of static slots in the static segment is a global constant.
- All static slots are of same length in terms of macroticks.
Structure of the Static Segment

- Every node maintains two *slot counters*: one for Channel A and one for Channel B.
- The node *simultaneously increments* both counters at the end of each static slot.
Arbitration in the Static Segment

- **Static Slot “N”** is reserved for **Frame ID “N”**.
- If a node detects that its slot counters contain the same value as the Frame ID of one of its queued messages, then it sends that message.
- Static Slot “N” of the current cycle will *remain idle* if the node that contains Frame ID “N” has no message to send now.
Fault-Tolerant Transmission in the Static Segment

- If a node is connected to both channels, then the node can *simultaneously send* a time-triggered message through both channels.

- However, if a node is connected to only one channel (say Channel A), then Channel B will *remain idle* when the node sends a time-triggered message through Channel A.
Timing within the Static Segment

- Channel active
- Channel idle
- Frame ID 1
- Frame ID 2
- Microtick
- Slot counter
- Slot 1
- Slot 2
Static Slot Length

- Appropriate configuration of the static slot length ($gdStaticSlot$) must assure that the frame and the channel idle delimiter and any potential safety margin fit within the static slot under worst-case assumptions.
Structure of the Dynamic Segment

- The dynamic segment contains a number of minislots.
- For a given system, the number of minislots in the dynamic segment is a global constant.
• All minislots are of *same length* in terms of macroticks.

• In the dynamic segment the *duration* of a communication slot depends upon the frame length.
Arbitration in the Dynamic Segment

- Each node continues to maintain the two slot counters – one for each channel.
- The slot counters are *incremented independently* according to the arbitration scheme within the dynamic segment.
- The slot number of the first minislot in the dynamic segment is the last static slot number plus 1.
Timing within the Dynamic Segment

- Channel active
- Delimiter
- Channel idle

Frame ID m

- Minislot
- Minislot action point
- Dynamic slot transmission phase
- Dynamic slot idle phase

Slot counter m

Dynamic slot m+1 m+2 m+3

Slot counter incremented at the end of each dynamic slot

Frame ID m+3

- Minislot
- Minislot action point
- Dynamic slot transmission phase
- Dynamic slot idle phase

T
Timing within the symbol window.

Arbitration is not provided by the protocol for the symbol window. The arbitration has to be performed by means of a higher-level protocol.
Network Idle Time

- During the network idle time a node calculates and applies clock corrections.
- The network idle time also serves as a phase during which the node performs implementation specific housekeeping operations.
- The network idle time contains the remaining number of macroticks within the communication cycle not allocated to the static segment, dynamic segment, or symbol window.
Frame Encoding in the Static Segment

- The transmitting node generates a **Transmission Start Sequence** (TSS) that consists of a continuous LOW for a given period.
- The **Frame Start Sequence** (FSS) consists of one HIGH bit. The **Byte Start Sequence** (BSS) is used to provide timing information to the receiver. The BSS consists of one HIGH bit followed by one LOW bit.
- The BSS is followed by eight data bits.
- The **Frame End Sequence** (FES) marks the last byte of a frame. The FES consists of one LOW bit followed by one HIGH bit.
Frame Encoding in the Dynamic Segment

- The *Dynamic Trailing Sequence* (DTS) is used to indicate the exact point in time of the transmitter’s *minislot action point*.
- After sending the FES, the transmitter keeps the TxD line low until the next minislot action point.
- The transmitter then moves the line to the HIGH level. *(DTS → Variable number of LOW bits + one HIGH bit)*.
Sync Nodes and Sync Frames

- At least three nodes shall be configured to be sync nodes.
- Sync nodes must be connected to both channels.
- The sync frames shall be sent in the same static slot in each cycle.
- Non sync nodes must not transmit frames with the sync frame indicator set.
Rate Correction

• The Rate Correction determines by how many microticks the node’s cycle length should be increased or decreased.

• A node determines the length of a communication cycle by taking the difference between the arrival times of a pair of Sync Frames, with the same Frame ID, on consecutive cycles.
Rate Correction

• If there are two pairs for a given Sync Frame ID (one pair for channel A and another pair for channel B) the average of the differences is used.

• The process is repeated for all Sync Frames, and a fault-tolerant midpoint algorithm is executed for the correction.

Average Length = \( \frac{L_A + L_B}{2} \)
Offset Correction

- The offset correction value indicates how many microticks the node should shift the start of its cycle.
- Negative values mean the NIT should be shortened (making the next cycle start earlier). Positive values mean the NIT should be lengthened (making the next cycle start later).

Example: NIT has been shortened
Offset Correction

• A node determines the offset of a communication cycle by taking the difference (deviation) between the actual arrival times of a given Sync Frame ID and its expected arrival time.

• If a given Sync Frame ID has two deviation values (one for channel A and one for channel B) the smaller value is selected.

• The process is repeated for all Sync Frames, and a fault-tolerant midpoint algorithm is executed for the correction.
Fault-Tolerant Midpoint Algorithm

- The algorithm determines the value of a parameter, k, based on the number of values in the sorted list.

<table>
<thead>
<tr>
<th>Number of values</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 2</td>
<td>0</td>
</tr>
<tr>
<td>3 – 7</td>
<td>1</td>
</tr>
<tr>
<td>&gt; 7</td>
<td>2</td>
</tr>
</tbody>
</table>

FTM term deletion as a function of list size.
Fault-Tolerant Midpoint Algorithm

- The measured values are sorted and the k largest and the k smallest values are discarded.
- The largest and the smallest of the remaining values are averaged for the calculation of the midpoint value.

Algorithm for clock correction value calculation (k=2):

```
15
13
11
...
+ \rightarrow 17 / 2 = 8
6
8
5
```
Clock Correction

• The correction terms are used to adjust the number of microticks in each macrotick
Wake-Up Signal

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dWU_{01}$, $dWU_{02}$</td>
<td>Duration of Data_0 phase in WU (*)</td>
<td>4</td>
<td></td>
<td>µs</td>
</tr>
<tr>
<td>$dWU_{idle1}$, $dWU_{idle2}$</td>
<td>Duration of Idle phase in WU (*)</td>
<td>4</td>
<td></td>
<td>µs</td>
</tr>
<tr>
<td>$dWU$</td>
<td>Duration of valid wake up pattern</td>
<td>16</td>
<td>48</td>
<td>µs</td>
</tr>
</tbody>
</table>

(*) as seen by receiving BD.
Wakeup of two Channels in a Fault-Tolerant way by Coldstart Nodes
Startup

- Wakeup procedure has to be completed before **startup** can commence. The startup is performed on all channels synchronously.
- The action of initiating a startup process is called a **coldstart**. Only a limited number of nodes may initiate a startup, they are called the **coldstart nodes**.
Startup

- When a coldstart node enters startup, it listens to its attached channels and attempts to receive FlexRay frames.
- If no communication is received, the node commences a coldstart attempt.
- A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS).
- The CAS is succeeded by the first regular cycle (Cycle 0: a startup frame).
- From Cycle 0 on, the node transmits its startup frame.
Collision Avoidance Symbol (CAS)

- The Collision Avoidance Symbol (CAS) contains a long train of LOW bits.
Startup

• Since each coldstart node is allowed to perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path.

• This situation is resolved during the first four cycles after CAS transmission.

• As soon as a node that initiates a coldstart attempt receives a CAS symbol or a frame header during these four cycles, it reenters the listen state.

• Consequently, only one node remains in this path.
Startup

- In **Cycle 4**, other coldstart nodes begin to transmit their startup frames.
- The node that initiated the coldstart collects all startup frames from **Cycle 4** and **5** and performs clock correction.
- Then the node leaves startup and enters operation.
Path of the Integrating Coldstart Nodes

• When a coldstart node enters the startup, it listens to its attached channels and attempts to receive FlexRay frames.
• If communication is received, it tries to integrate to a transmitting coldstart node.
• It tries to receive a valid pair of startup frames.
• If these frame receptions have been successful, it performs clock correction in the following double-cycle.
Path of the Integrating Coldstart Nodes

• If clock correction does not signal any errors, it begins to transmit its startup frame; otherwise it reenters the listen state.

• If for the following three cycles the clock correction does not signal any errors and at least one other coldstart node is visible, the node leaves startup and enters operation.

• Thereby, it leaves startup at least one cycle after the node that initiated the coldstart.
Path of a Non-Coldstart Node

- When a non-coldstart node enters startup, it listens to its attached channels and tries to receive FlexRay frames.
- If communication is received, it tries to integrate to a transmitting coldstart node.
- It tries to receive a valid pair of startup frames to derive its schedule and clock correction.
Path of a Non-Coldstart Node

- In the following double-cycles, it tries to find at least two coldstart nodes that transmit startup frames.
- If this fails or if clock correction signals an error, the node aborts the integration attempt and tries anew.
Path of a Non-Coldstart Node

- After receiving valid startup frame pairs for two consecutive double-cycles from at least two coldstart nodes, the node leaves startup and enters operation.
- Thereby, it leaves startup at least two cycles after the node that initiated the coldstart.
- This means that all nodes of the cluster can leave startup at the end of cycle 7, just before entering cycle 8.
State Transitions for the Fault-Free Startup

Node A
- POC state cycle schedule
- coldstart collision resolution
- coldstart consistency check
- normal active

Node B
- POC state cycle schedule
- coldstart collision resolution
- coldstart consistency check
- normal active

Node C
- POC state cycle schedule
- coldstart collision resolution
- coldstart consistency check
- normal active

Channel
- CAS symbol
- startup frame of node A
- startup frame of node B
- frame of node C

Legend
- CAS symbol
- startup frame of node A
- startup frame of node B
- frame of node C
Comparison TTP, TTCAN, FlexRay

Dr. Markus Plankensteiner
markus.plankensteiner@tttech.com
TTTech Computertechnik AG
## Transmission Speed / Frame Length

<table>
<thead>
<tr>
<th>Protocol</th>
<th>TTP</th>
<th>TTCAN</th>
<th>FlexRay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission speed</td>
<td>25 Mbit/s; higher speeds possible</td>
<td>1 Mbit/s; higher speeds not possible due to CSMA/CD access</td>
<td>10 Mbit/s planned; higher speeds possible</td>
</tr>
<tr>
<td>Maximum frame length for application data</td>
<td>Variable for each node, up to 240 Bytes</td>
<td>Variable for each transmission, up to 8 Bytes</td>
<td>Up to 254 Bytes; for TDMA part all slots must be of equal size</td>
</tr>
</tbody>
</table>
## Clock Synchronization

<table>
<thead>
<tr>
<th>Protocol</th>
<th>TTP</th>
<th>TTCAN</th>
<th>FlexRay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock synchronization</td>
<td>Single fault-tolerant distributed formally verified offset correcting synchronization (with four or more nodes), optional rate correction to external reference time</td>
<td>Master-slave synchronization, cluster internal or driven by external events</td>
<td>Fault-tolerant distributed offset and rate correcting synchronization (formal verification unknown)</td>
</tr>
</tbody>
</table>
## Fault Tolerance / Faults Detected / Consistency Support

<table>
<thead>
<tr>
<th>Protocol</th>
<th>TTP</th>
<th>TTCAN</th>
<th>FlexRay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault hypothesis / Fault tolerance</td>
<td>Tolerates all single hardware faults, two redundant communication channels</td>
<td>No systematic fault tolerance, no redundant channels</td>
<td>Tolerates some single hardware faults, no fault-hypothesis</td>
</tr>
<tr>
<td>Faults detected</td>
<td>Bit errors on the bus, transmit and receive faults</td>
<td>Bit errors on the bus, transmit and receive faults</td>
<td>Bit errors on the bus</td>
</tr>
<tr>
<td>Consistency support</td>
<td>Acknowledgment, membership, clique avoidance</td>
<td>Acknowledgment</td>
<td>None</td>
</tr>
<tr>
<td>Protocol</td>
<td>TTP</td>
<td>TTCAN</td>
<td>FlexRay</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------</td>
<td>--------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Event handling strategy</td>
<td>Event channel on top of TDMA (CAN emulation)</td>
<td>TDMA plus arbitrating window for CSMA/CA</td>
<td>TDMA plus dynamic segment for mini-slotting</td>
</tr>
<tr>
<td>Availability</td>
<td>Chips since 1998</td>
<td>Chips as engineering samples</td>
<td>Chips according to specification 2.0 announced for 2005</td>
</tr>
</tbody>
</table>
TTP – Efficient Solution

Benefits
- High speed
- High data efficiency
- Flexible
- Good EMC

Properties
- 25 Mbit/s, higher speed possible
- Up to 90%
- Individual slot size
- Collision-free

Mbit/s Transmission Speed

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>100 Mbit/s</td>
</tr>
<tr>
<td>TTP</td>
<td>25 Mbit/s</td>
</tr>
<tr>
<td>FlexRay</td>
<td>10 Mbit/s</td>
</tr>
<tr>
<td>CAN</td>
<td>500 kbit/s</td>
</tr>
</tbody>
</table>

Efficiency

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 %</td>
</tr>
<tr>
<td></td>
<td>70 – 90 %</td>
</tr>
<tr>
<td></td>
<td>40 – 70 %</td>
</tr>
<tr>
<td></td>
<td>20 %</td>
</tr>
</tbody>
</table>
References

5. Overview of CAN, Motorola.
References


