

Goldilocks Serial Communication Protocol

B.Terry Compton
Delphi

Copyright © 2008 SAE International

ABSTRACT

Goldilocks is a proposed serial communication protocol designed to find the “sweet spot” in trading off overhead, complexity, throughput, cost, and real time performance. Simple off line analysis of schedulability is a feature.. While backward compatible with CAN at the applications level, Goldilocks promises to be performance and feature competitive with FlexRay with less complexity and cost.

INTRODUCTION

This paper is conceptual and does not document every feature of Goldilocks in great detail. An information report for J2824 will be submitted in the spring of 2008 to the SAE VADC committee containing significant additional detail. Goldilocks is not yet an approved standard of SAE.

Goldilocks is a protocol consistent with the priority arbitration which is characteristic of CAN. However, as that form of arbitration constrains bus speed, Goldilocks proposes to perform scheduling and simulation off-line, based on the principles of DMA (Deadline Monotonic Analysis), which is the state of the art in scheduling algorithms. DMA provides adaptability and flexibility to changing requirements. However, DMA requires pre-emption to reduce jitter to levels consistent with the deterministic requirements of automotive OEMs for hard real-time and safety-critical systems.

Pre-emption is a salient feature of Goldilocks. Once pre-emption has been introduced, long low priority messages are possible, since they can be pre-empted. This facilitates supporting instrumentation, multi-media streams, fast re-flash, etc. concurrent with closed loop control signals, provided there is sufficient bus bandwidth. Goldilocks proposes that after schedules have been prepared and simulated off-line to verify schedulability by an automated Goldilocks schedule and simulation tool, that each ECU be loaded with its schedule. For native Goldilocks controllers, there shall be a hardware maintained global time, and the Goldilocks controllers shall be responsible for ECU conformance to schedules, since the timing accuracy

required to implement the schedules would normally be beyond the capability of software. Goldilocks controllers shall also be capable of being configured to produce a real-time interrupt for the RTOS synchronized to global time, so that software tasks can be synchronized to the bus. Additionally, Goldilocks proposes to synchronize multiple buses for safety-critical applications. Each bus shall have its own Goldilocks communication controller for added redundancy vs. a single communications controller for dual synchronized buses, as in FlexRay. Additionally, a provision is made to synchronize multiple CPUs and buses when CPU redundancy is also required.

Goldilocks is not a single monolithic protocol, but a set of protocol features where controllers can be built to meet Goldilocks conformance classes. This allows a consistent bus philosophy to be applied to a continuum of technologies, providing scaleable solutions to OEM vehicle bus challenges. This approach supports more precise management and measurement of ISO 26262 ASIL levels, so important to achieve reliable systems in the face of today's electronic complexity in vehicles. Goldilocks provides protocol stack backwards compatibility to CAN by means of a new hardware abstraction layer, as native Goldilocks controllers have hardware differences with CAN. TTCAN complies with Goldilocks Conformance Class 1. Goldilocks proposes to minimize complexity, while maximizing performance, flexibility, adaptability to changing requirements, and reliability of the network. It is expected that the features of Goldilocks will increase quality while reducing costs to the OEMs.

PROPOSAL

At the recent meeting of the SAE VADC committee, Goldilocks was proposed. The committee's decision was to assign SAE J2824 to this proposed protocol, and assign with a requirement to provide an information report on this protocol by VADC, meeting in the Spring of 2008. More details about Goldilocks can be found in the minutes of this meeting.

<http://forums.sae.org/access/dispatch.cfm?EVSIS=docshare/showFolder/104893>  **Documents**
Document Sharing Hub

At the VADC meeting, the Goldilocks proposal suggested that Goldilocks might be able to initially use the CAN Physical Layer at a faster speed than CAN uses its Physical Layer. Some contributors to Goldilocks have encouraged the author to try very hard to adapt Goldilocks to an existing Physical Layer to avoid the significant investments in time and money required to implement and verify new Physical Layers. This paper will document how that advice has been followed while not constraining Goldilocks to existing Physical Layers.

PROTOCOL ARCHITECTURE

The design of Goldilocks is intended to make it possible for Goldilocks to adapt to many different Physical Layers, as the following block diagram illustrates. Note that the design is to have a standard API between the Goldilocks Controller and the Medium Access Control and CODEC. This design therefore does not require a standard MAC/CODEC and Physical Layer.

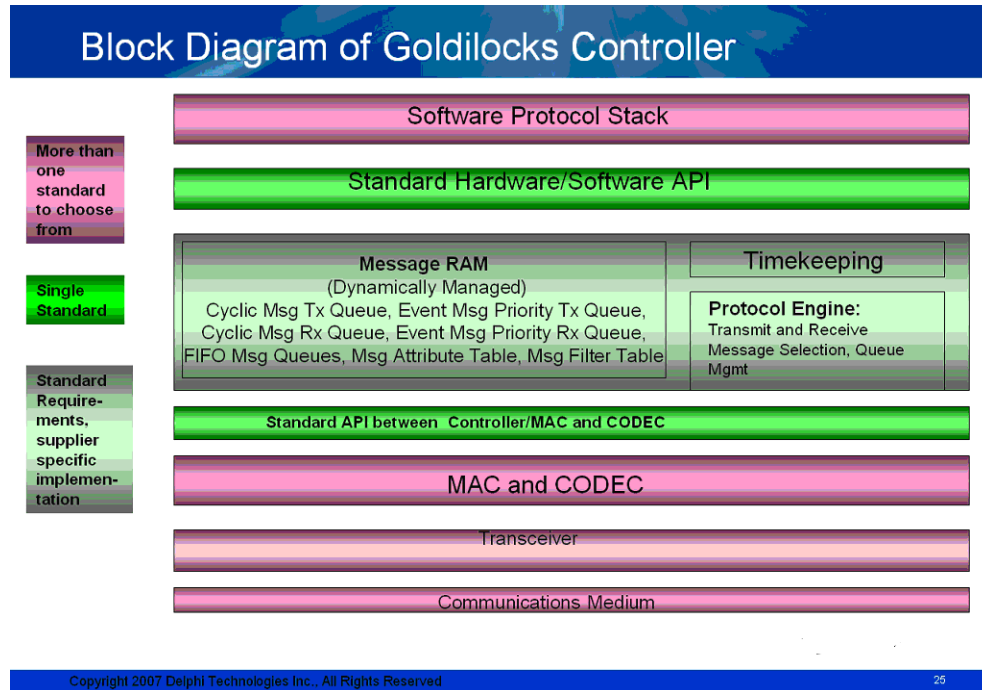


Figure 1 Block Diagram of Goldilocks Controller

Therefore, Goldilocks development is evaluating faster Physical Layers up to 100 mbps, but nothing has yet been chosen beyond the high-speed CAN Physical Layer.

In the VADC meeting, there was some discussion as to the speeds CAN can achieve at 40 meters. It is common in the literature for CAN to be advertised to run at 1 mbps for 40 meters. However, there were suggestions that this is not possible because of total system propagation delays. These delays come from the transceiver, length of the bus and other circuitry, and how these parameters affect the speed at which CAN arbitration can occur.

It is true that the limitation on CAN's bus speed derives from its method of arbitration where each node must write to the same bit at effectively the same time. In addition to transceiver and miscellaneous propagation delays, as the bus length increases, so does propagation delay, and hence with fast enough speeds, and long enough buses, nodes are unable to write to the same bit at temporally the same time. In practice today, CAN is generally limited to 500kbps at 40 meters, even

though a few OEMs are running CAN at 1mbps for special functions, such as ECU re-flash, where occasional errors are corrected with a reasonable number of retries and before bus off occurs. Additionally, it is probably true that for shorter buses, some CAN implementations are running at speeds in excess of 1 mbps, again for specialty purposes. The author is not aware of any vehicle CAN bus that runs continuously at greater than 500kbps.

FEATURES OF THE GOLDILOCKS PROTOCOL

SCHEDULING - Goldilocks proposes to use scheduling methods compatible with CAN (priority scheduling), but also proposes to avoid the use of methods which allow propagation delays to limit bus speed. Specifically, Goldilocks proposes to avoid arbitration by scheduling messages, so there is no bus contention. In addition, Goldilocks proposes to avoid the use of ACK, which also requires all nodes to write to the same bit position at the same time. CRCs shall be sufficient to detect most errors. Additional documentation on error detection and containment will be included in J2924. However, no active error frames will be used, as their

use destroys deterministic behavior, and a goal of Goldilocks is to produce deterministic bus behavior. Therefore, CAN does not fill all the requirements of Goldilocks. However, TTCAN does satisfy a subset of the design criteria of Goldilocks, as it does not require ACKs, and does not use error frames.

Goldilocks scheduling is done offline, and assigns priorities to messages according to the principles of DMA (Deadline Monotonic Analysis). You can Google this subject and find many pertinent documents and tutorials. These DMA principles are:

1. The longer the message, the lower the priority
2. The faster the cycle time of a message, the higher the priority
3. The greater the deadline (allowable jitter), the lower the priority

By analyzing these message attributes offline, a DMA-based Goldilocks scheduling and simulation tool will be able to assign priorities to all cyclic messages, and determine if the schedule is achievable at a specified baud rate, or conversely, what baud rate would be required to support the schedule.

EVENT MESSAGES - In a Goldilocks message database, event messages should also have a cycle time, which is the maximum burst rate at which the event must be serviced (not necessarily the rate at which it can occur). Event messages should also have a deadline. For offline schedulability analysis, event messages will be treated as cyclic messages, but at run time they will be handled according to the rules of event messages.

ECU SCHEDULE - Once a schedule and baud rate have been selected by the network designer, each ECU would be provided with its schedule. . These schedules would be stored in the ECU, and executed by the ECU at run time. For a full featured Goldilocks controller, the Goldilocks controller hardware would insure global time and conformance to the schedule. For preliminary Goldilocks systems using a TTCAN controller, the software would be responsible to insure the schedule was followed, though the TTCAN hardware does provide global time synchronization to facilitate conformance to schedule.

By following the schedule, and with no error frames, all bus contention would be avoided, and the Goldilocks application would be able to use the same Physical Layer as CAN and run at a much faster speed with greater determinism. TTCAN controllers support a sufficient subset of the features of Goldilocks to be considered a Conformance Class 1 Goldilocks controller.

BAUD RATES - While most third party tools only allow CAN baud rates up to 1mbps, Delphi has an internally developed CAN analysis tool which supports much higher baud rates. The processor used is a TriCore 179X running at 150 MHz. This tool has two CAN ports, each separately configurable. To find out how fast a CAN Physical Layer can run under the rules of Goldilocks, it was decided to provide an on-board loopback connection between these two ports for the SAINT2 tool, so that propagation delay would be only that of the on-board circuitry, transceiver, and just a few inches of loopback wire. The transceiver delay was measured with a scope at approximately 60 ns. The circuitry delay was measured at approximately 6 ns, resulting in a total delay between the two nodes of approximately 132 ns. That is, the propagation delay between the two on-board CAN controllers was about 132 ns. The transceiver was a TJA 1040.

A second SAINT2 was configured to be in "receive only" mode, so that it would not ACK nor send messages. This node was 40 meters from the first two internally connected nodes. At 5 ns delay per meter, if the second SAINT2 had been providing an ACK or sending messages, then the pertinent delay for that node would have been 200 more ns, or a total of 332 ns. With a target speed of 500ns per bit, such a delay would have been prohibitive. Some consider that the delay should be calculated for a round trip, and in that case, delays would be doubled. However, whether they are doubled or not does not change the conclusions of this paper.

All three CAN ports were configured to run at 1.973 mbps. Only one node sent messages. The messages were sent on the bus error-free in this configuration, documenting that without ACKs and arbitration, the CAN Physical Layer is capable of up to approximately 2 mbps communication. This same configuration did not work reliably at 2.3 mbps.

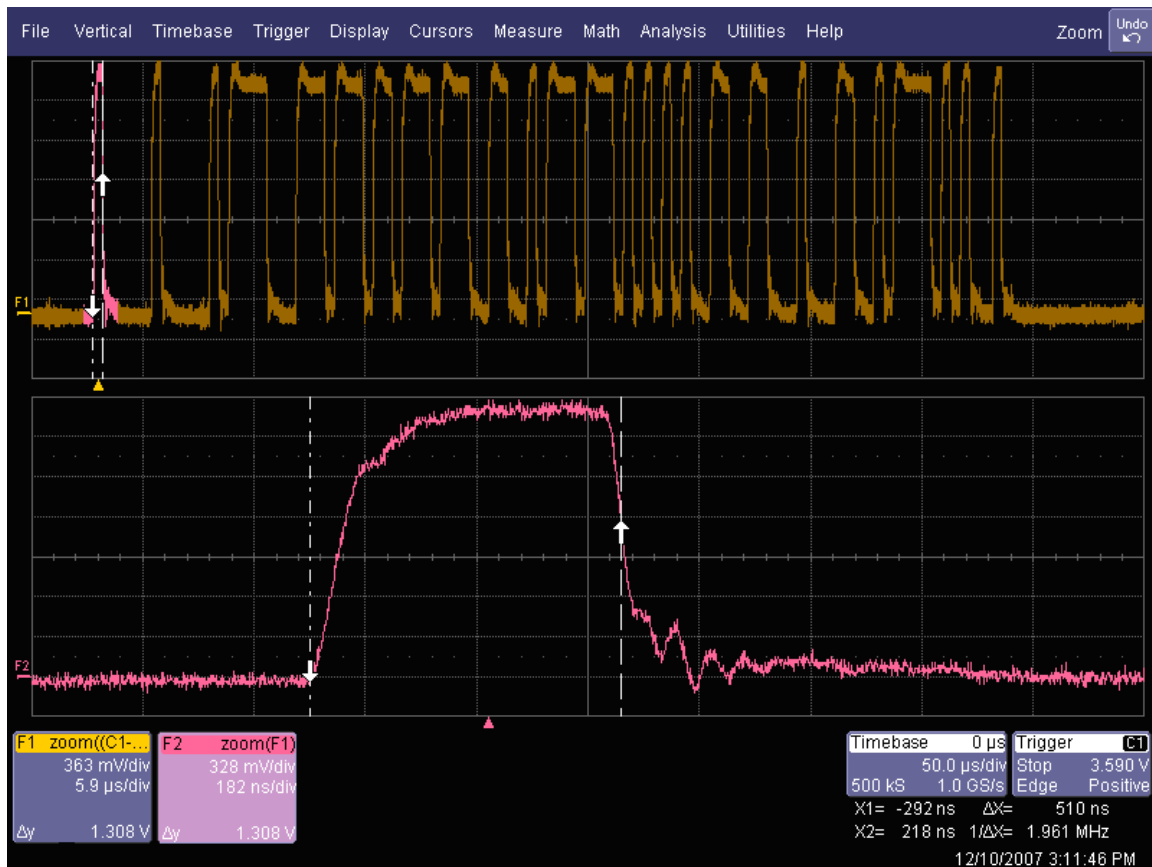


Figure 2 Single Bit Time For CAN At Near 2mbps and 40 meters

Two screen shots of the scope were taken and show the data being transmitted (Figures 2 and 3). Both screen shots show the entire message and zoom in on a single bit. In the first screen shot, the bit time is measured, showing the baud rate to be in the neighborhood of 2mbps. The second scope screen shot measures the differential voltage of the bit, and shows it to be 2.4 volts, above the required 2 volts for a CAN differential signal. The measurement was taken at about 35 meters from the transmitter. The rise and fall times of the signal appears to be adequate to support a little faster baud rate. EMI is not expected to change significantly, as the wave shaping of the TJA1040 appears to be independent of the frequency up to its maximum frequency, which has not been determined. However, the 2mbps frequency itself might present a problem, and if it does, a different baud rate/oscillator frequency may have to be used for a particular application.

It is not recommended to use CAN controllers at these baud rates. They present a number of problems, which however are all solved by using TTCAN controllers, also available on the Tricore 179x series, and other vendors micros, e.g. NEC. It is the authors understanding that Bosch also sells TTCAN FPGAs, external TTCAN controllers, and TTCAN VHDL code. TTCAN does not use error frames nor require ACKs. Hence, the 132 ns of delay for the ACK seen in the above configuration, is of no consequence in determining the maximum baud rate for TTCAN, and one would expect to achieve even higher baud rates with a TTCAN controller up to the limit of the TJA1040 transceiver capability. No TTCAN controller was available to the author at the time of the writing of this paper to determine how fast a TTCAN controller could run with a TJA1040 transceiver.