

Task and Message Scheduling for a FlexCAN-based Hybrid-Electric Vehicle Drivetrain Functional Unit

Juan Pimentel
Kettering University

Copyright © 2008 SAE International

ABSTRACT

A Task and Message Schedule for a FlexCAN-based Hybrid-Electric vehicle (HEV) functional unit is described. The resulting schedule is a component of an incremental message and task scheduling approach based on a time-driven message schedule and priority-driven task schedule. The HEV functional unit involves the combined control and monitoring functions of an internal combustion engine working in parallel with a permanent magnet synchronous motor. The control algorithm for the synchronous motor has been simulated using VHDL-AMS. The global message system is supported by FlexCAN and the task scheduler system is supported by a priority based OS (e.g., OSEK or AUTOSAR).

INTRODUCTION

The complete design of a distributed automotive embedded system that includes several applications can be complex – it involves multitude steps dealing with communications, control, and computing issues [1]. The design flow being adopted by many researchers and including AUTOSAR is the *Y-chart approach* [3], where the application description and the architecture description are initially separated, but joined together later in an explicit *mapping* phase. The main steps in the mapping phase include decomposing applications into functions, function allocation to software tasks, task allocation to ECUs (electronic control units), task priority assignment, task scheduling, signal mapping into messages, message priority assignment, and global message scheduling.

This paper details the task and message scheduling of a single functional unit, the drivetrain of a hybrid electric vehicle (HEV), in the context of the *mapping phase* described above. The task and message schedule is based on the approach described in [1] that uses an incremental technique for message scheduling used by FlexCAN [4,5], a CAN-based communication system for dependable automotive applications. The approach is augmented by performing task scheduling in the context of a priority based O.S. (e.g., OSEK or AUTOSAR). A

notable feature of the approach is simple synchronization of both task and message scheduling to the application in terms of *control system transactions* (CSTs).

The advantages of event-triggered protocols such as CAN have been widely reported, particularly its bandwidth usage and its flexibility, which allows incremental system design (e.g., new ECUs or tasks can be added without redesigning the system from scratch) [6-8]. However, their dynamic nature makes them difficult to analyze and to guarantee design properties. Thus verifying certain real-time and deterministic properties becomes problematic and challenging. This becomes a serious hindrance to their applicability in certain applications such as safety-critical systems [5].

In the past few years, a great deal of research has been carried out aimed at making CAN more reliable and more dependable [4,6,7,9,10]. From this point of view, FlexCAN is not so new. However, this protocol is particularly interesting, as it provides a holistic approach with satisfactory results, despite the fact that it is simpler than other solutions. However its simplicity is a good advantage, making its implementation simple, particularly using the AUTOSAR architecture [11].

Basically, FlexCAN partitions messages and schedules them offline in several *sub-cycles* (which, under reasonable conditions do not overlap), so that Tindell analysis [12] can be used inside each sub-cycle. Due to the limited number of messages involved in each sub-cycle, the overall deterministic behavior is better than the original CAN. By creating a schedule offline, complex timing constraints, such as irregular task arrival patterns and precedence constraints, can be handled by FlexCAN in a predictable manner that would be difficult to do online (i.e., during run time). Therefore, the online behavior of time-driven schedulers such as the one used by FlexCAN is very predictable [5]. The FlexCAN framework is thus highly compositional. There are other fault tolerant and dependable features of FlexCAN, such as node and channel replication, and bus guardians, but these features are not discussed in this paper [13,14].

In addition to scheduling CAN messages in a time driven fashion adopted from FlexCAN, the approach schedules tasks in a way that makes them synchronized to both the application and the corresponding messages and uses a priority based operating system (OS). This is done by defining a model where some tasks configure (but not schedule) messages after their execution, and some messages trigger the execution of certain tasks. In this way tasks and messages are interrelated to one another in a cause-effect relationship. The model adopted and the message and task scheduling policy ensure that tasks complete execution before the corresponding messages are scheduled for transmission on a global CAN bus network. Likewise, it is ensured that messages meet their real-time latency and bandwidth requirements using a specific schedulability analysis developed for FlexCAN..

The approach is intended to be used in the design of distributed automotive embedded systems in an iterative and incremental fashion that can be automated. The process begins with one or several functional units (i.e., applications such as drive-by-wire and powertrain), and continues with task and message scheduling, including the evaluation of the performance and resource utilization of the resulting system. In particular, this paper details the task and message schedule for a HEV powertrain functional unit. Additional functional units can then be incorporated in an incremental and flexible fashion until the performance or the resource utilization exceed pre-established values. The process can continue after additional resources (FlexCAN networks or ECUs) are added to the system. Thus the proposed approach is not only incremental, but is also scaleable and flexible so that it can deal with tasks and messages ranging from a few to several hundreds involving many in-vehicle applications, ECUs, and several CAN networks.

PAPER ORGANIZATION

In Section 2, FlexCAN and control system transactions are summarized. The HEV functional unit is detailed in Section 3. Section 4 describes the model used for task and message scheduling. The HEV task and message schedule is presented in Section 5. Finally the summary and conclusions are in Section 6.

FLEXCAN AND CONTROL SYSTEM TRANSACTIONS

The goal of the FlexCAN architecture is to provide additional deterministic and dependable capabilities to the CAN protocol without compromising its flexibility. This is accomplished by incorporating an additional layer on top of CAN. The main features of the FlexCAN architecture are: replicated components, support for time-domain composability, replica synchronization, replication management, and enforcement of fail-silent

behavior. FlexCAN has been described in a number of publications [4,5,13,14] and only a brief summary is provided next.

SUMMARY OF FLEXCAN

The FlexCAN architecture can incorporate several replicated channels and several replicated nodes (in addition to normally non-replicated ones) in a flexible fashion [6]. A node and all of its replicas is called an FTU (Fault-Tolerant Unit), but not all nodes in a network need to be replicated.

Time domain composability has been a major liability of CAN-based networks for safety-critical applications. There have been several proposals to overcome this limitation, such as TT-CAN and FTT-CAN, which basically adopt a time-triggered transmission scheme at a high level and still use CAN at the lower level.

FlexCAN also adopts the time-triggered paradigm by using reference messages. The interval between reference messages is called the *communication cycle*, and this interval is further divided into a number of *sub-cycles*. A small set of messages are scheduled in a time-triggered fashion on each sub-cycle, and the worst case message latency for all messages in every sub-cycle is calculated in a time frame relative to the beginning of the sub-cycle. Other messages in the application function are scheduled in the other sub-cycles. The communication cycles, together with their sub-cycles, not only help with time domain composability, but also help to synchronize replicated nodes and channels, and to enforce fail-silent behavior, particularly using bus guardians. Communication cycles are also useful to synchronize applications. FlexCAN is flexible in that it supports both periodic and a-periodic traffic. Multi-rate periodic messages (i.e., periodic messages with different periods) are also handled through appropriate message scheduling.

CONTROL SYSTEM TRANSACTIONS

Nearly all embedded systems involve control systems of one kind or another. Basically, a control system takes input (I) measurements from a physical process (P), performs a control function (C), and produces outputs (O) to effect changes on the physical process. From a software perspective, the fundamental software functions or tasks found within control systems constitute a *CST (control system transaction)*. A *CST* simply consists in performing a series of software tasks involving task *I*, for collecting inputs from a process, Task *C*, to implement a control function, and Task *O*, to send controller outputs back to the physical process. These software tasks have to be performed in the following precise order

$$P \rightarrow I \rightarrow C \rightarrow O \rightarrow P \quad (1)$$

which is known as the *CST Precedence*.  The various software tasks in a *CST* communicate with one

another through messages resulting in a *Task-message CST flow* to be detailed below.

Time synchronization between a TT global message schedule and end applications are crucial for most control systems. FlexCAN enables the synchronization of any CST event (e.g., the enqueueing, transmission, or reception of messages) to the communication cycle. A simple and effective synchronization method that is used in this paper is to synchronize the enqueueing of certain messages to the beginning of sub-cycles.

THE HEV DRIVETRAIN FUNCTIONAL UNIT

The HEV Drivetrain Functional unit consists of the control and monitoring functions of a conventional internal combustion engine (ICE) and an AC permanent magnet synchronous motor (PMSM) joined together by a planetary gear torque coupler, as shown in Fig. 1. The PMSM is controlled by a suitable motor controller through a power inverter that draws energy from a traction battery. In the past 20 years there have been steady advances in the control of AC motors, with techniques ranging from field oriented control (FOC), to direct torque control (DTC), to non-linear control.

To design a suitable task and message schedule for the control and monitoring software and associated communications, it is necessary to know some details of the control systems. There is a wealth of information on ICE controllers, so we do not provide any details here. However, in the following we will provide some details about a non-linear PMSM controller based on Lyapunov functions [15]. To simplify the analysis and design of control methods, three-phase AC machines are modeled in a rotating frame in terms of two perpendicular axes d and q . The equations describing the behavior of a PMSM in the so-called d - q frame are,

$$V_d = R i_d + d\Phi_d/dt - \omega \Phi_q \quad (2)$$

$$V_q = R i_q + d\Phi_q/dt + \omega \Phi_d \quad (3)$$

$$\Phi_d = L_d i_d + \Phi_m \quad (4)$$

$$\Phi_q = L_q i_q \quad (5)$$

where V_d , V_q are the stator voltages, i_d , i_q the stator currents, Φ_d and Φ_q are the flux linkages in the d - and q -axes respectively. Φ_m is the flux created by the rotor magnets, R is the stator resistance, and L_d , L_q are the stator d - and q -axis inductances.

The electric torque is,

$$T_e = 3 P [\Phi_m i_q - (L_q - L_d) i_d i_q] / 2 \quad (6)$$

where P is the number of pole pairs of the AC machine. The equation for the motor dynamics is,

$$T_e = T_L + B\omega_r + Jd\omega_r/dt \quad (7)$$

where B is the viscous friction coefficient, and J is the motor moment of inertia. The inverter frequency ω is related to the rotor speed ω_r by:

$$\omega = P \omega_r \quad (8)$$

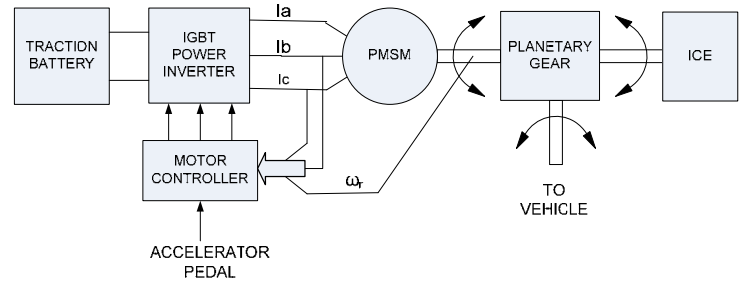


Figure 1. Constituent Components of the Monitoring and Control Functions of the HEV Functional Unit.

For vehicle applications, it is appropriate to control the torque developed by both the PMSM and the ICE. In the following we detail a non-linear torque control method obtained using a Lyapunov function that guarantees global stability [15]. The control equations are:

$$V_d = K_d L_d e_d + K_1 L_d \theta_d + R i_d - \omega L_q i_q \quad (9)$$

$$V_q = K_q L_q e_q + K_2 L_q \theta_q + R i_q + \omega L_d i_d + \omega \Phi_m \quad (10)$$

where K_d , K_q , K_1 , and K_2 are positive constant gains and,

$$e_d = i_d^* - i_d \quad (11)$$

$$e_q = i_q^* - i_q \quad (12)$$

are the tracking errors from the desired trajectory (i_d^* , i_q^*), and,

$$\theta_d = \int e_d(\tau) d\tau \quad (13)$$

$$\theta_q = \int e_q(\tau) d\tau \quad (14)$$

To verify the above control algorithms we have simulated the above torque control equations in the VHDL-AMS language using SystemVision with the motor parameters of Table 1. Fig. 2 shows the commanded and actual torque, motor speed, and one of the three-phase motor currents (I_a) for 2 seconds of simulation time. One can observe that this control algorithm results in fast torque dynamics and effective motor torque control.