

A Graphical Model for Interactive Distributed Control

Foreword—This document defines an architectural model of a Distributed Embedded System and a method of graphically representing the events, activities, and transactions that may be used for supporting interactive Distributed Control.

TABLE OF CONTENTS

1.	Scope	2
1.1	Background	2
1.2	Rationale	2
2.	References	2
2.1	Applicable Documents	2
2.1.1	SAE Publications	2
2.1.2	Other Publications	3
3.	Definitions.....	3
4.	Abbreviations/Acronyms.....	3
5.	General.....	3
5.1	Essential Building Blocks.....	3
5.2	Distributing a Function	4
5.3	The Event, Activity, and Transaction Diagrams.....	6
5.4	Model of a General Function.	9
5.5	Process Models.....	10
5.5.1	Input Process	10
5.5.2	Output Process.....	11
5.5.3	Control Process.....	13
5.5.4	Transmit Process	14
5.5.5	Receive Process.....	15
5.6	Interconnects	16
5.7	Diagramming the Distributed Function	17
6.	Summary and Conclusions	21
	Appendix A Timing Symbols and Definitions	23

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be reaffirmed, revised, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2007 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER:

Tel: 877-606-7323 (inside USA and Canada)
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
http://www.sae.org

SAE WEB ADDRESS:

- 1. Scope**—The demonstrated architectural model and associated graphical techniques defined herein were developed to provide a simple method of visualizing the general functional operation or behavior of a Distributed Embedded System with a strong emphasis on representing system time characteristics.

1.1 Background

- 1.1.1 WHAT IS INTERACTIVE DISTRIBUTED CONTROL?—Interactive Distributed Control, also identified as IDC, is an information and control transfer strategy that allows a network of microcomputer-based modules to support a collection of high-speed distributed functions.

- 1.1.2 WHY WAS THE SAE IDC COMMITTEE FORMED?—Three key motivations led to the formation of the Interactive Distributed Control Committee.

First—The timing analysis of whether high-speed distributed systems have the capacity to adequately handle all the required functions is critical.

Second—The design of an interactive distributed control system, the development of hardware and software, and the selection of a suitable network protocol can benefit from such analysis prior to system realization.

Third—The interest to establish a benchmark to determine the effectiveness of an IDC system.

As a result, the IDC Committee's main focus is to study the automotive application of high-speed multiplexing (small area networking) to handle distributed motion control.

- 1.1.3 WHY ARE HIGH-SPEED DATA TRANSFERS NEEDED?—It is quite possible that vehicle motion control processing may become more distributed. The timing of these motion control algorithms, like engine control, braking, traction control, speed control, and transmission control, is considered critical to proper operation.

- 1.1.4 WHAT PROBLEMS ARE ENCOUNTERED WHEN USING MULTIPLEX FOR IDC?—Two key IDC problems or questions need to be addressed:

- Is the network speed and bandwidth capable of supporting all of the intended high-speed distributed functions?
- What microcomputer processing power is required to support the module-related high-speed distributed functions?

Analyzing these problems is the primary focus of the IDC Committee and the basis for this document that examines the modeling and timing characteristics of a generalized Interactive Distributed Control system.

- 1.2 Scope**—The rationale to cancel SAE J2356, an older SAE Information Report, is to reuse a portion of the document for a new recommended practice. The intention is to redirect some of the SAE J2640 membership towards this new agenda once the SAE J2640 document is finished.

2. References

- 2.1 Applicable Publications**—The following publications form a part of the specification to the extent specified herein. Unless otherwise indicated the latest revision of SAE publications shall apply.

- 2.1.1 SAE PUBLICATIONS—Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE J1213-1—Glossary—Vehicle Networks for Multiplexing and Data Communications

SAE J1850—Class B Data Communications Network

SAE J1930—Electrical/Electronic Systems Diagnostic Terms, Definitions, Abbreviations, and Acronyms

SAE J1939—Heavy Truck High-Speed CAN

SAE J2056—Class C Multiplexing

SAE J2284—Automotive High-Speed CAN

2.2 Other Publications

Mullender, Sape, Distributed Systems - Second Edition. Addison-Wesley, 1993.

3. Definitions

3.1 Distributed Embedded System—A general form of a distributed system characterized by being confined to a small geographic area, interconnected using small area networking, and implemented using microcomputer-based modules, smart sensors, and smart actuators. Alternate term—Small Area Distributed System.

3.2 Distributed Function—A general function which is physically distributed across two or more system entities.

3.3 Interactive Distributed Control—An information and control transfer strategy that uses interaction between a network of microcomputer-based modules to support a collection of high-speed distributed functions.

4. Abbreviations/Acronyms

DES—Distributed Embedded System

IDC—Interactive Distributed Control

5. General—The intent is to develop a simple and generalized architectural model that includes the essential elements required to support a collection of distributed functions typically found within the boundaries of a Distributed Embedded System. Using these elements or building blocks, the model is further expanded such that the timing aspects of the distributed processes can be both defined and graphically depicted. Three time diagramming formats are presented.

5.1 Essential Building Blocks—A general distributed function, process, or system can be conceptually constructed using five architectural elements (Figure 1). Three elements common to any general system are the input, control, and output building blocks. The other two, transmit and receive, are unique only to distributed systems which have been physically partitioned into two or more entities (usually called nodes of a network).

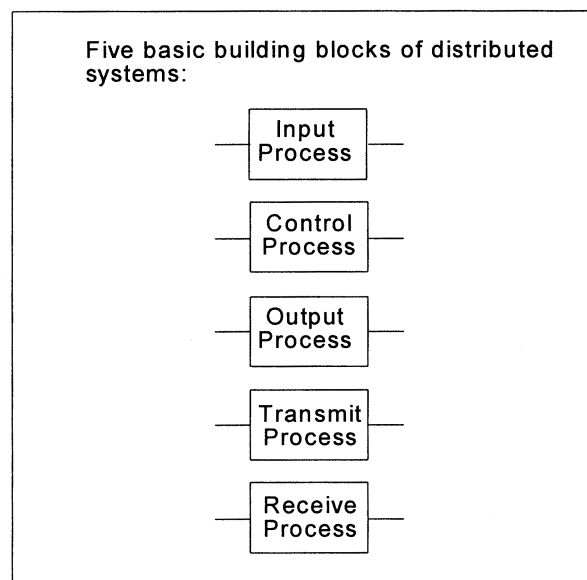


FIGURE 1—ELEMENTS OF THE DISTRIBUTED FUNCTION

- The **Input** element interconnects an input device or signal to a control element by providing a data representation of the input. In some cases, an input may be connected to more than one control.
- The **Control** element contains the algorithm for translating input mechanism abstractions into output mechanism abstractions and implements the desired customer and system behaviors for a given function or feature.
- The **Output** element accepts information in the form of a data structure from a control element and converts the information into system output responses which may be signals or actuated devices.
- The **Transmit** element accepts information in the form of a data structure from either an input or control element and converts the information into a data transfer activity on a network.
- The **Receive** element accepts a data transfer from a network and interconnects the received data structure to the correct control or output element.

5.2 Distributing a Function—Once the decision to distribute a system has been made, the selection of partitioning boundaries for each function within the system must be made. This development step will establish the module to which each architectural element belongs. The partitioning choice is usually made by the Distributed Embedded System designer and several factors may influence the physical boundary selection including:

- The type of implemented control architecture, especially whether system-level control is administered centrally or distributed.
- The type of implemented software architecture, especially whether a standardized software building block approach is used.
- The expected data rate at a potential boundary; usually partitioning at boundaries that produce lower data transfer rates is preferable to a selection at a high data range.
- The available microcomputer resources within a related electronic module and the location of the input or output.

Once the boundary is established, the two network connected process blocks, transmit and receive, are appropriately inserted into the overall process as shown in Figure 2. In general, the distributed function contains one of each of the five DES architectural elements.

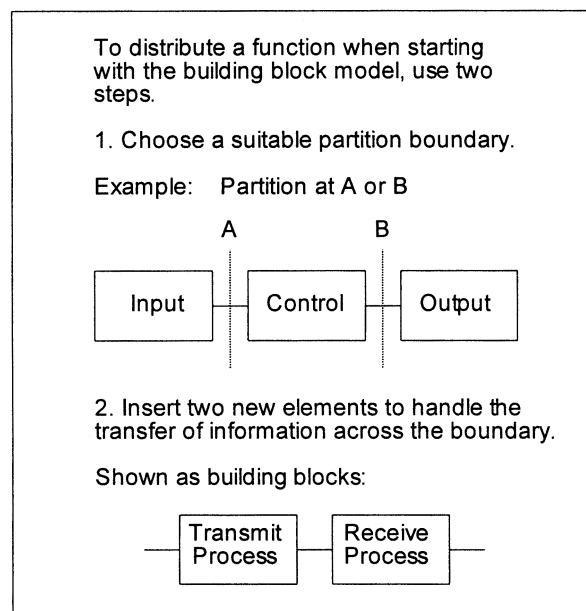


FIGURE 2—DISTRIBUTING A FUNCTION

Based on this architectural model, the partitioning selection appears to be quite simple, especially if each elemental block has already been prepackaged. However, when one moves to implement these concepts, the meaning and importance of partitioning becomes clear. The major realization is the fact that a partitioning boundary is always located within the software.

In the software world, these boundary locations are called interfaces. Each software interface will require a structural design to handle both process invocation and data transfer. Good software design principles encourage the creation of standard or logical interfaces at these partitioning locations. This disciplined approach provides a consistent, easier to manage, and potentially reusable software architecture.

The two examples, Figure 3 and Figure 4, show the common partitioning locations. While partitioning within an input, control, or output block is also achievable, this document does not consider this topic.

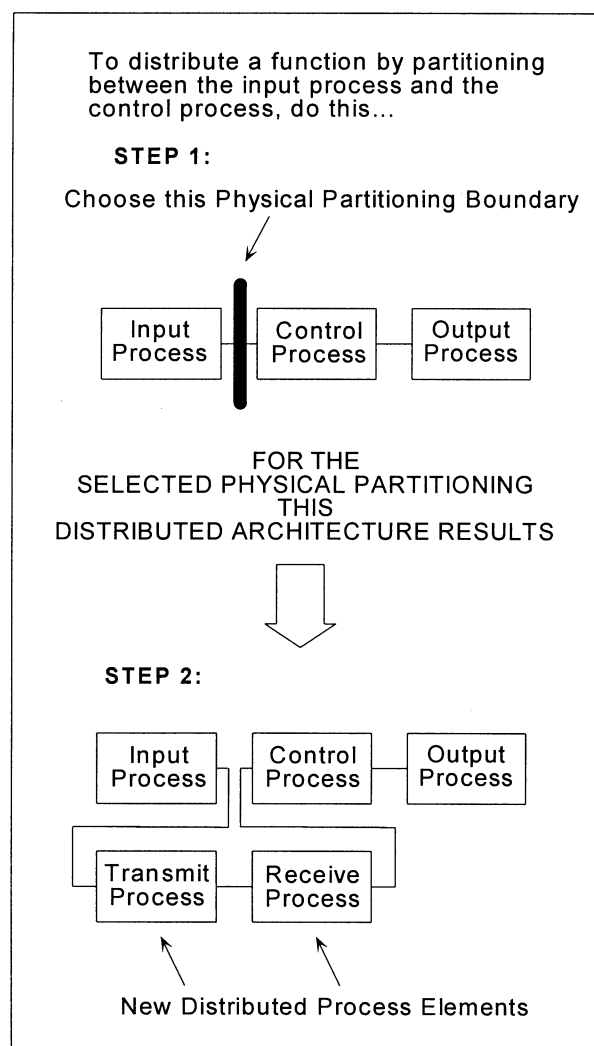


FIGURE 3—PARTITIONING EXAMPLE

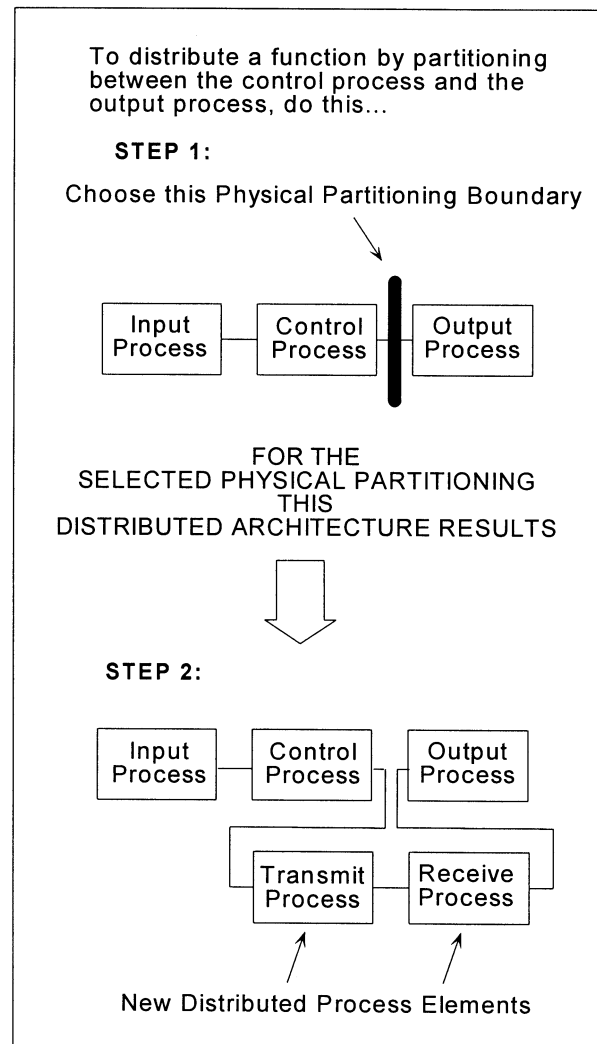


FIGURE 4—ANOTHER PARTITIONING EXAMPLE

Some distributed embedded system designers and experienced software developers recommend:

- Matching the overall system software architecture to the distributed function architecture.
- Partitioning using standardized software building blocks to reduce complexity.

5.3 The Event, Activity, and Transaction Diagrams—To help visualize the processes of a distributed embedded system, representations of the system's events, activities, and transactions may be used. Each representation models and shows the timing characteristic of a functional component of the system.

An **event** is an occurrence that is recognizable at a specific point in time, for example, a power-up event or a reset event. The event diagram is shown in Figure 5. An **activity** is a process, a collection of related operations all assigned to a designated group. For example, the initialization of a network is an activity. The activity diagram is shown in Figure 6. A transaction is a representation of an activity used to show the movement from one activity or event to the next. A series of transactions could be used to model a complete distributed function showing each activity across the entire system. The transaction diagram is shown in Figure 7.

Each of these diagram methods is related, as shown in Figure 8, and may be used to show specific timing.

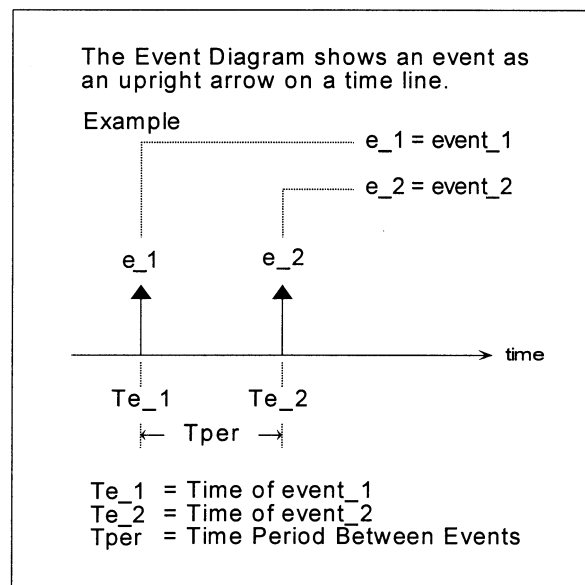


FIGURE 5—EVENT DIAGRAM

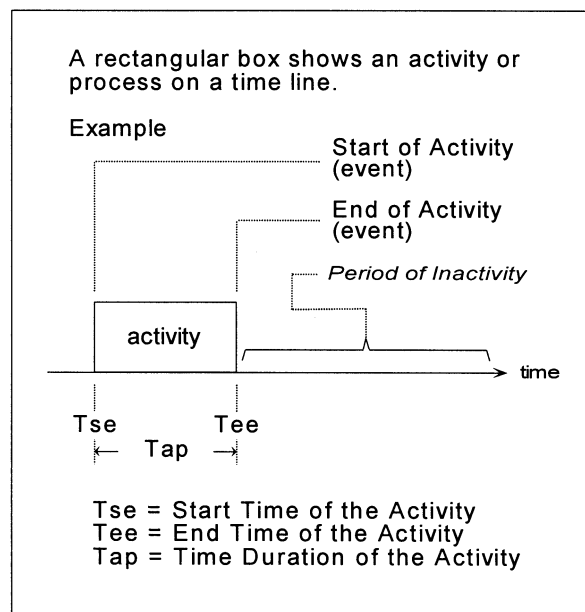


FIGURE 6—ACTIVITY DIAGRAM

The Transaction Diagram shows a process and its timing elements along a single horizontal time line.

Example

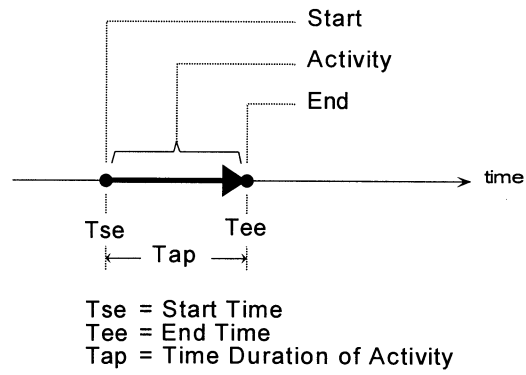


FIGURE 7—TRANSACTION DIAGRAM

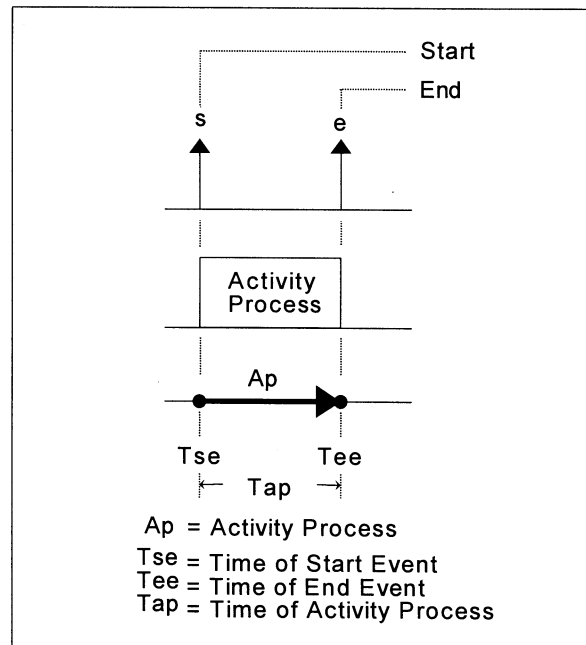


FIGURE 8—ALL DIAGRAM REPRESENTATIONS

5.4 Model of a General Function—The graphical depiction of a general function, a function which may or may not be distributed, can always be represented by the three common building blocks - input, control, and output. Although such a graphical representation is useful, the building block diagram is a physical representation of a function and lacks the ability to show the element of time.

The comparable activity diagram is a functional representation of the same familiar building block components. However, this simple transition, shown in Figure 9 using the same functional elements, now includes the added capacity to indicate time.

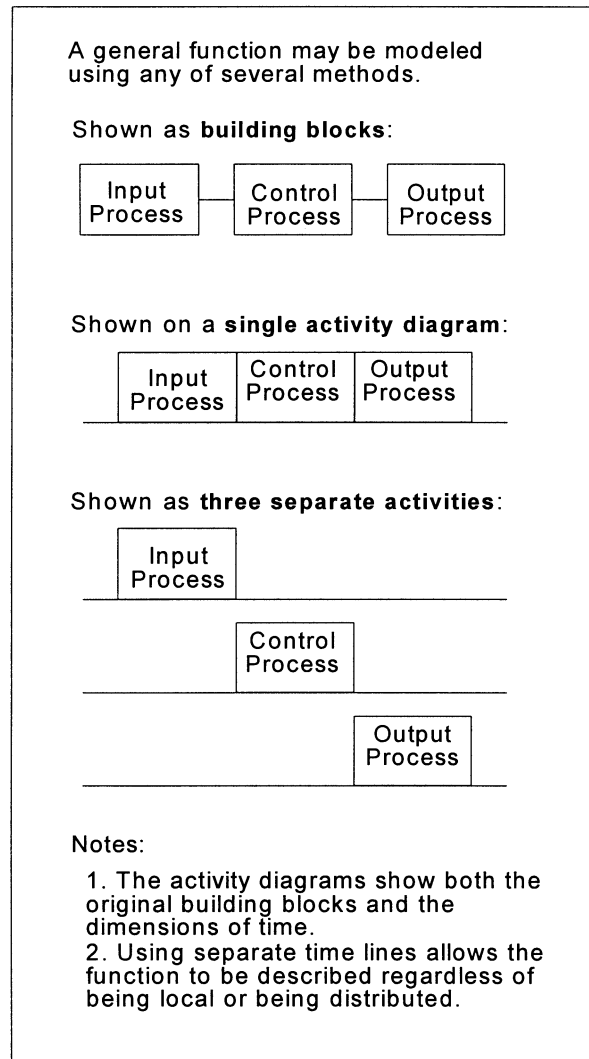


FIGURE 9—MODEL OF A GENERAL FUNCTION

The system designer may choose to show activities on the same time line or indicate each process with its own distinctive time line. Whether the general function is located within a single module or distributed across a group of modules, the choice of multiple separate activity diagrams allows functions to be characterized regardless of the selected physical partitioning. This technique of separating the system activities into individual constituents, which is effectively decomposing the system by function rather than by module, is called functional partitioning.