To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

RENESAS
Renesas Technology Corp.

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

Ho7055 Communication Manual

ADE-702-257

Cautions

PLEASE READ THE FOLLOWING CAREFULLY BEFORE YOU USE THIS PRODUCT.

1. If you use the enclosed software product and any related software products (hereafter referred to as "PRODUCT"), before exporting or taking such PRODUCT to other countries or states, you must comply with applicable export control laws and regulations of Japan and other countries with jurisdiction and the applicable states and provinces within Japan and such other countries.

2. Please be advised that Hitachi neither warrants nor grants licenses of any rights to the patents, copyrights, trademarks, or other intellectual property rights owned by Hitachi or any third party for the use of the PRODUCT, unless otherwise expressly granted to you by Hitachi in a contract or other document including without limitation any warranty or license included in the user's manual for the PRODUCT (hereinafter referred to as "CONTRACTS"). Please be further advised that Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with the use of the PRODUCT.

3. The PRODUCT, its specifications and/or its description in the user's manual are subject to change in the future without any prior notice. Confirm that you have received the latest standards and/or specification for the PRODUCT (including the user's manual) before you make your final design, purchase or use.

4. Please be advised that Hitachi will not have any liability whatsoever for damages, including indirect or consequential damages, arising out of your use of the PRODUCT (including the use based on the descriptions of the user's manual). Hitachi shall not be liable for any damages caused by any equipment or media used for delivery of the PRODUCT.

5. The PRODUCT is not designed for, and you may not use the PRODUCT for, applications that demand especially high quality and reliability, or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as equipment used for aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support. If you have any questions regarding whether or not your intended use of the PRODUCT is permitted by Hitachi,, please contact your local Hitachi's sales office.

6. At the time of designing or planning your system using the PRODUCT, please consider normally foreseeable failure rates or failure modes and employ sufficient systematic measures such as fail-safe systems so that the equipment incorporating the PRODUCT does not cause any accident or other consequential damage due to operation of the PRODUCT.

7. This manual and the PRODUCT are copyrighted by Hitachi. Under any circumstances, you may not copy, analyze, reverse engineer, and/or modify, in whole or in part, the PRODUCT, except to the extent expressly provided in the CONTRACTS.

8. You may not use or copy, in whole or in part, the user's manual for the PRODUCT without the prior written consent of Hitachi, except to the extent expressly provided in the CONTRACTS.

9. You may use the PRODUCT on just one (1) computer. You may not transfer, lease or otherwise assign the PRODUCT to any third party or parties, except to the extent expressly provided in the CONTRACTS.

10. Please contact your local Hitachi's sales office for any questions regarding the PRODUCT, any Hitachi semiconductor products or any related products.

Select the items you want to know from the following flowchart before reading this manual.

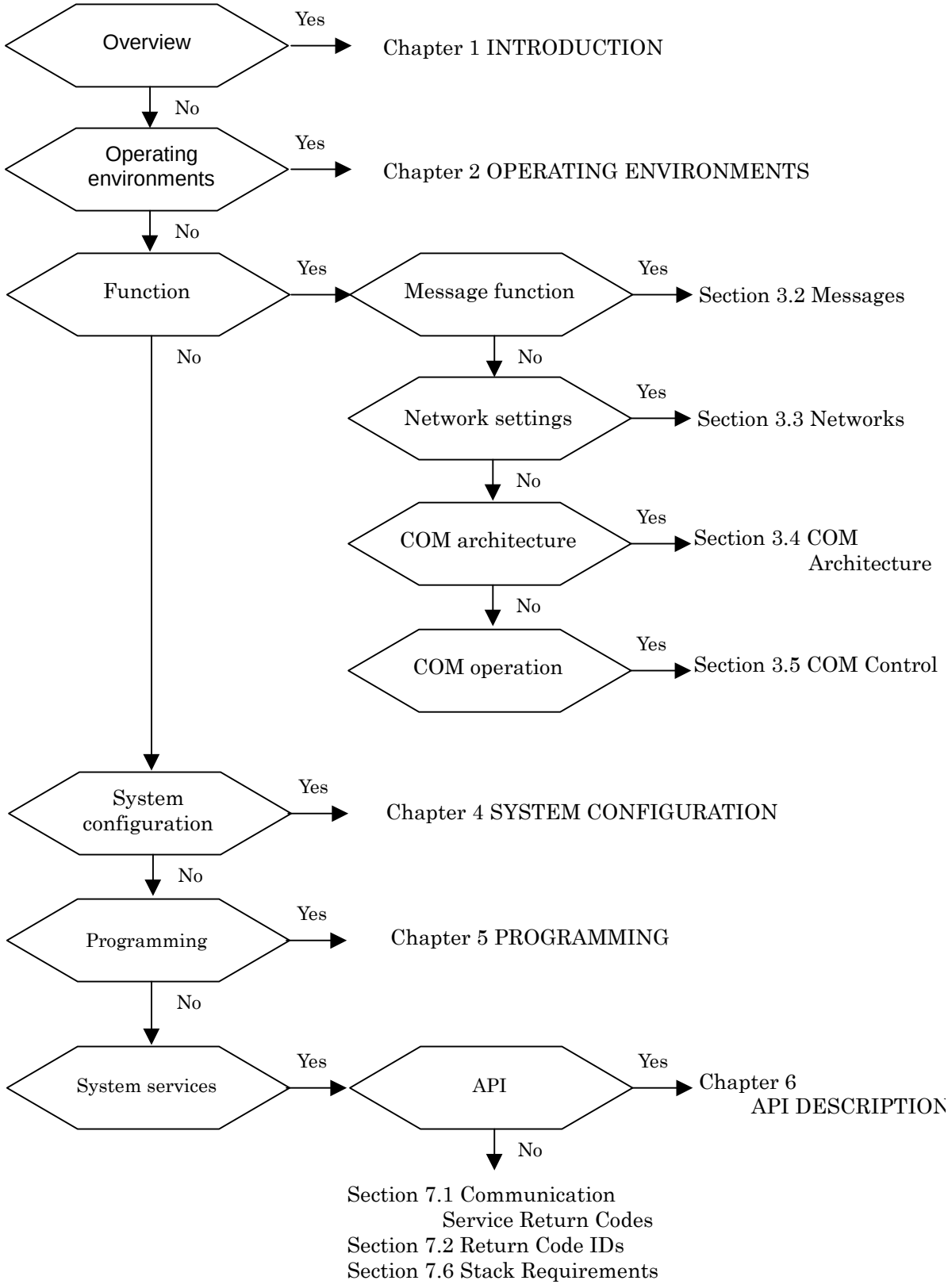

Table of Contents

1 INTRODUCTION

1.1 Document Overview

This document defines the operation of the Hitachi Vehicle Operating System Communication V2.1 (hereafter referred to as COM) which conforms to the OSEK/VDX (hereafter referred to as OSEK) open standard for communication, Specification Version 2.0a. **This document is described on the assumption that OSEK specification is understood.**
Read this document carefully and understand the contents of the following documents before using this product.

- "OSEK/VDX Operating System", Version 2.0 revision 1 (OSEK/VDX steering committee)
- "OSEK/VDX Communication", Version 2.0a (OSEK/VDX steering committee)
- "OSEK/VDX Communication", Version 2.1 revision 1 (OSEK/VDX steering committee)
- Release Notes of this product
- SuperH RISC engine C/C++ Compiler Package Manual
- Programming Manual and Hardware Manual of the target SH microprocessor

1.2 Definitions and Acronyms

API	Application Program Interface
CAN	Controller Area Network
CCC	Communication Conformance Class
COM	Communication for Hitachi Vehicle Operating System
DLL	Data Link Layer
ECU	Electronic Control Unit
HCAN	Hitachi Controller Area Network
HW	Hardware
NM	Network Management
OS	Operating System
OSEK	Open systems and the corresponding interfaces for automotive electronics
SW	Software
VDX	Vehicle Distributed eXecutive

1.3 COM Overview

The OSEK COM is one of SW components specified by the OSEK Standard. The other components are the operating system (OS) and the network management module (NM). All three components provide functionality common to automotive applications.

Figure 1.1 shows a system concept for automotive applications. In a car, there are a number of ECUs exchanging information over a CAN network. Each ECU is dedicated to the control of a certain mechanical component or subsystem in the car. For instance, ECU #1 may be the Engine Management Controller, ECU #2 the Gearbox Controller, etc. Higher level functionality in the car may be distributed over several ECUs.

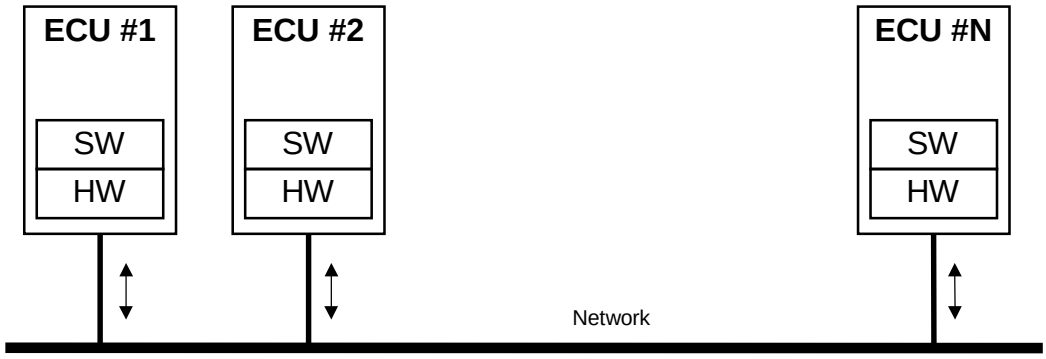

Figure 1.1 System Concept for Automotive Applications

The internal structure of an ECU is shown in Figure 1.2.

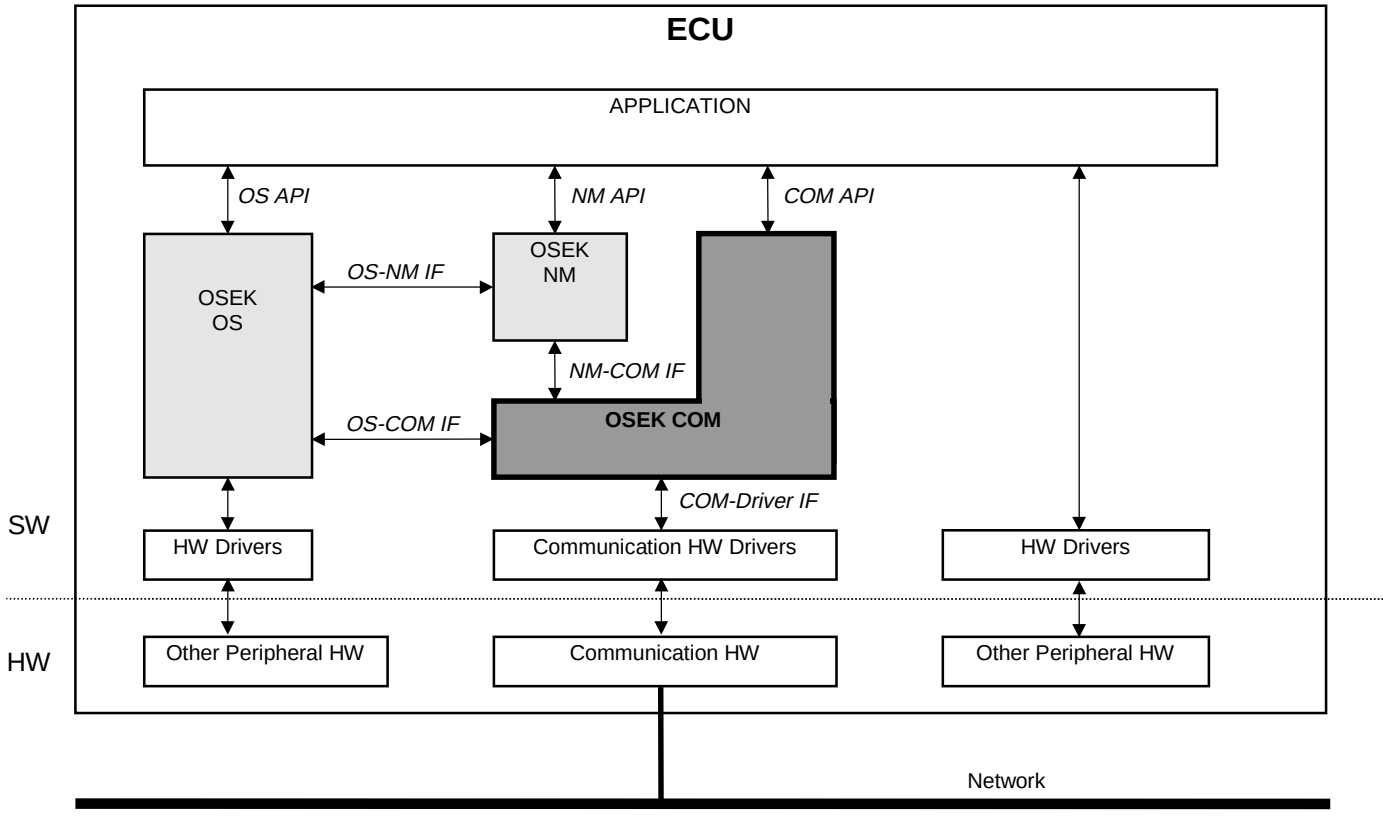

Figure 1.2 Internal SW Structure of an ECU

The COM controls the communication, both within the ECU, and between ECUs. It provides the application with services for sending and receiving messages. The API is uniform for services exchanging messages within a node and between nodes.

The COM component can be implemented with compatibility, which is reflected by its Communication Conformance Class (CCC). With this product, the CCC0 and CCC1 conformance classes are supported. CCC0 is the least resource demanding class and may be implemented without any OS support. CCC1 requires the OS. The CCCs are provided in order to allow the communication component capabilities to be adapted to the communication need of the application.

In order to gain access to the communication HW, the COM makes use of the device drivers provided by the COM-driver interface. This interface is not specified by the OSEK standard, and is thus be specific to each particular implementation. The CAN device is used in this product.

1.4 Features of This Product

The features of this product are shown below:

- The COM supports portability of application by providing a standardised application program interface that is defined according to the ANSI C standard.
- An OSEK COM implementation provides the following features for message exchange in applications.
  - **Task – Task communication**

- **ECU – ECU communication**
- **Different transmission concepts: Direct/Periodical/Mixed**
- **Communication deadline monitoring**
- **Notification by event setting and task activation**

Note: Periodical transmission and mixed transmission is provided by an original interface. Please refer to section 3.2.4 for details.

2 OPERATING ENVIRONMENTS

2.1 Conformity Version

The COM conforms to version 2.0a of the OSEK/VDX Communication specification and version 2.1 revision 1 of the OSEK/VDX Message transmission modes and deadline monitoring.

2.2 Conformance Class

The function corresponding to the conformance class of OSEK Communication specification is shown by ✔.

Table 2.1 Conformance Classes

Functions and Services		**Conformance Classes**	
		CCC0	**CCC1**
Transmission concept	Direct	✔	✔
	Periodical		✔
	Mixed		✔
Communication deadline monitoring			✔
Messages	Unqueued	✔	✔
Protocols	UUDT (undivided message)	✔	✔
Services	SendMessage	✔	✔
	ReceiveMessage	✔	✔
	GetMessageStatus	✔	✔
Notification	Task activation		✔
	Event setting		✔

CCC0
Provides the minimum services and functions to ECU-internal and inter-ECU communication. **The OS is not indispensable.**

CCC1
Provides periodical transmission, mixed transmission, deadline monitoring, and notification mechanism. **The OS is indispensable.**

2.3 Communication Bus

The COM is implemented for a communication bus of CAN (hereafter referred to as a bus). The COM uses HCAN as communication HW. **The COM does not operate on other communication HW and buses.**

2.4 Operating System

The COM can be used with the OS of Hitachi Vehicle Operating System. **The COM cannot be used by combining with other OSs.**

2.5 Notes

(1) The COM includes drivers for the HCAN devices that are built in the CPU. The drivers are not parts of the OSEK COM standard, but do not remove them because they are required for the COM operation.
(2) Refer to the help file of the COM configurator for details on COM configuration.
(3) Cannot be used in little endian. The COM must be used in big endian.

3 FUNCTIONS

3.1 Function Overview

The overview of the COM function is shown in Figure 3.1.

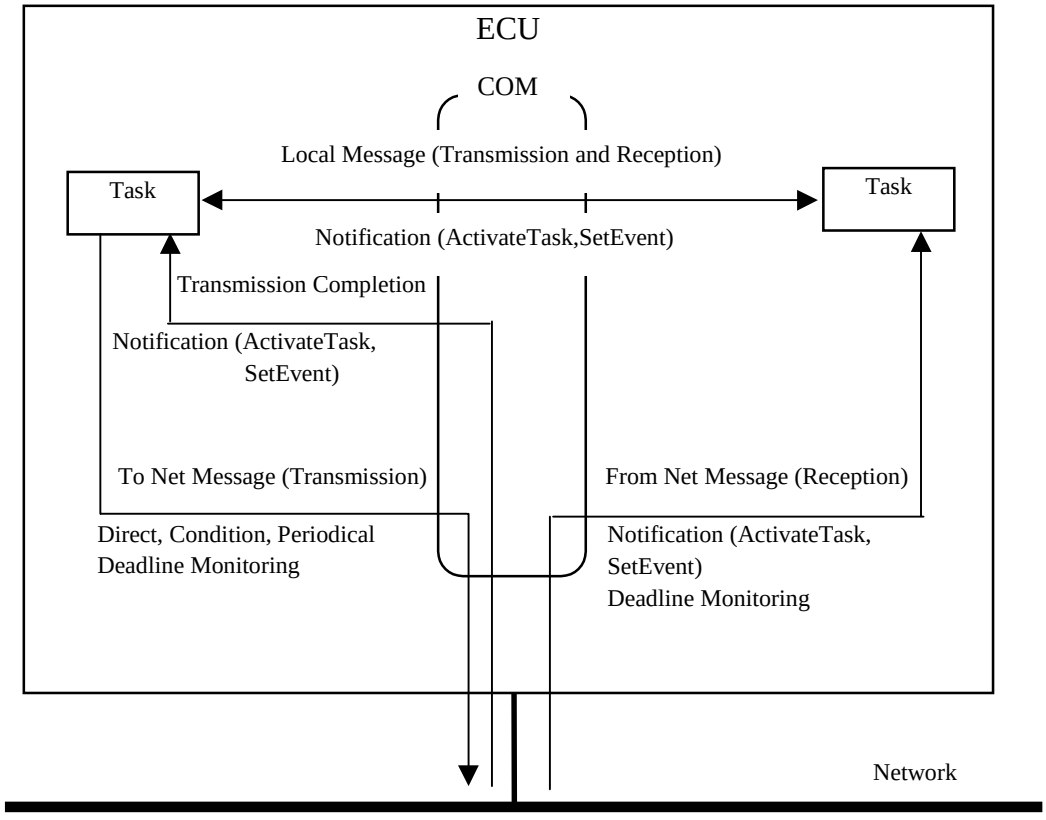

Figure 3.1 COM Function Overview

3.2 Messages

3.2.1 Message Types

There are three types of messages as follows:

- Local messages
- To Net messages
- From Net messages

3.2.1.1 Local

The Local messages are sent and received within a single ECU, without being transmitted over the network. They are used to exchange information between different parts of a single application.

3.2.1.2 To Net

The To Net messages are transmitted from the ECU to the network. They are received from the network by other ECUs. These messages may be received locally by the sending the ECU.

3.2.1.3 From Net

The From Net messages are received in the ECU from the network. They are transmitted from another ECU.

3.2.2 Message Operations

The following message operations are possible:

- Sending a message
- Receiving a message
- Reading the status of a message

3.2.2.1 Send

A message is transmitted via the SendMessage communication service. The transmission operation differs depending on the message types.

Local
For Local messages, transmission means copying message data from the application to a data area within the COM.

To Net
For To Net messages, message data is copied from the application, as for Local messages. In addition, the message data is sent to the network.
The above description is relevant for a message that is transmitted in *Direct* mode. Please see section 3.2.4 for details on other transmission modes.

3.2.2.2 Receive

Local
This service copies the message data from a data stored within COM message data area to the application.

From Net
When receiving a message from a bus, the message data domain in COM is immediately updated at the time. Message data is copied to application from the message data in COM at the time of ReceiveMessage communication service call. **Reception from a bus is asynchronously processed with ReceiveMessage communication service call.**

3.2.2.3 Get Status

The status of a message is obtained via the communication service GetMessageStatus. The operation is applicable to messages of all types. The status expresses the following conditions of the message:

- In use status: Locked (in use) or unlocked (not used)
- Data value status: Updated at least once or never updated

In Use
The status is locked if COM is using the message. The application cannot access the message when the status is locked. If the application calls a communication service, no action that alters the message will be performed and an error code indicating Locked will be returned. When the message can be used, the status is unlocked.

Data Value
At COM start-up, the status of all messages is Never Updated**.** As soon as a message has been sent from the application or received from the network, the status is changed into Updated at Least Once. Henceforth, it does not become Never Updated.

3.2.3 Notification to Application

This function can use only CCC1. Operations on messages are performed asynchronous to the application. A transmission to the network is requested by the application, but the transmission operation is not finished before the communication service returns to the application. In CCC1, the notification mechanism makes it possible for the COM to inform the application when the transmission has been completed. A reception from the network is performed entirely without intervention of the application. The COM receives a new message from the communication bus and stores it internally. These operations are:

Table 3.1 Notification Timing

Operation	**Timing**	**Applies to Message of Type**
Transmission of message	When issuing SendMessage	Local
	When transmission of the message to a bus is completed	To Net
Reception of message	When the reception of a message from bus is completed	From Net

There are two kinds of notification methods.

- COM sets an **event** when the operation is finished
- COM activates a **task** when the operation is finished

3.2.4 Transmission Modes

The COM offers the following transmission modes:

- **Direct**
- **Periodical (only CCC1)**
- **Mixed (only CCC1)**

Table 3.2 Message Transmission Modes

Direct	Message transmission is requested by each call by issuing SendMessage.
Periodical (only CCC1)	Message transmission is requested by a dedicated task activated by a cyclic alarm.
Mixed (only CCC1)	Message transmission is requested by a dedicated task activated by a cyclic alarm. In addition, message transmission is conditionally. The condition is user-defined.

3.2.4.1 Direct

A transmitted message is unconditionally transmitted by issuing SendMessage. Direct transmission is available for messages of types *Local* and *To Net*.

3.2.4.2 Periodical

When CCC1 is used, periodic transmission using alarm can be performed by combining with OS. A transmission message is transmitted periodically. In this mode, the communication service SendMessage does not initiate the transmission to the net. It only updates the message data.

The transmission to net is initiated by a periodically activated task. The task and the alarm must be defined by the user.

The periodic task initiates transmission through a special Send service that is not a formal part of the COM API. This service is named OC_SendMsgToNetPer. The user must add a call to this service in the periodic task. See chapter 6 for details.

Do not start periodical transmission until COM initialisation is complete. This means that the earliest point in time to start the alarm is from the MessageInit() function.

Periodical transmission is available for messages of type *To Net*.

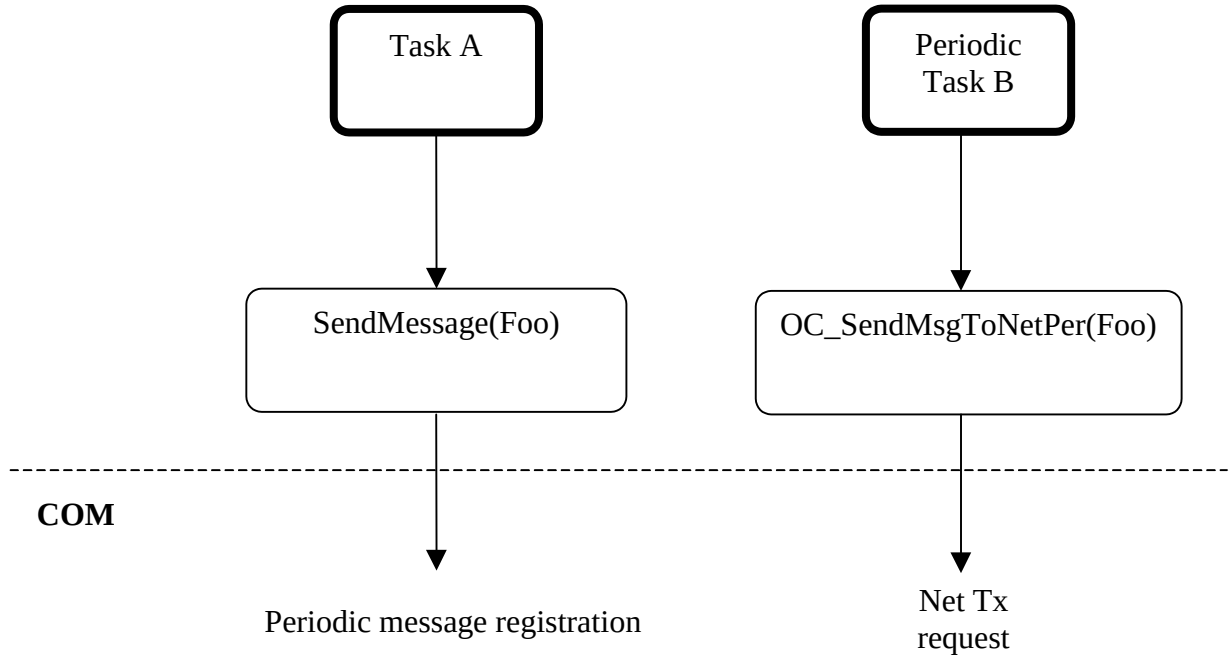

Figure 3.2 Periodical Transmission

The Task A runs independently from the periodic task B. The latter requests a transmission to the network, each time it is activated. The message named *Foo* is configured for periodical transmission mode.

3.2.4.3 Mixed

When CCC1 is used, mixed transmission using alarm can be performed by combining with OS. A transmission message is transmitted periodically. In addition, SendMessage conditionally initiates a transmission to the network. A condition is evaluated each time SendMessage is called. If the condition evaluates to “Send”, a transmission is initiated; otherwise not.

In this mode, the communication service SendMessage

1. Updates the message data
2. Calls the evaluation original function OC_MixedTxEval()
3. Requests a transmission to network if the evaluation function returned “Send”.

The evaluation performed in OC_MixedTxEval() is defined by application. The function must return a value that tells COM whether to send the message or not. See section 6 for details.

Transmissions to net are initiated by a periodically activated task. The task and the alarm must be defined by the user.

The periodic task initiates transmission through a special Send service that is not a formal part of the COM API. This service is named OC_SendMsgToNetPer. The user must add a call to this service in the periodic task.

Do not start periodical transmission until COM initialisation is complete. This means that the earliest point in time to start the alarm is from the MessageInit() function.

Mixed transmission is available for messages of type *To Net*.

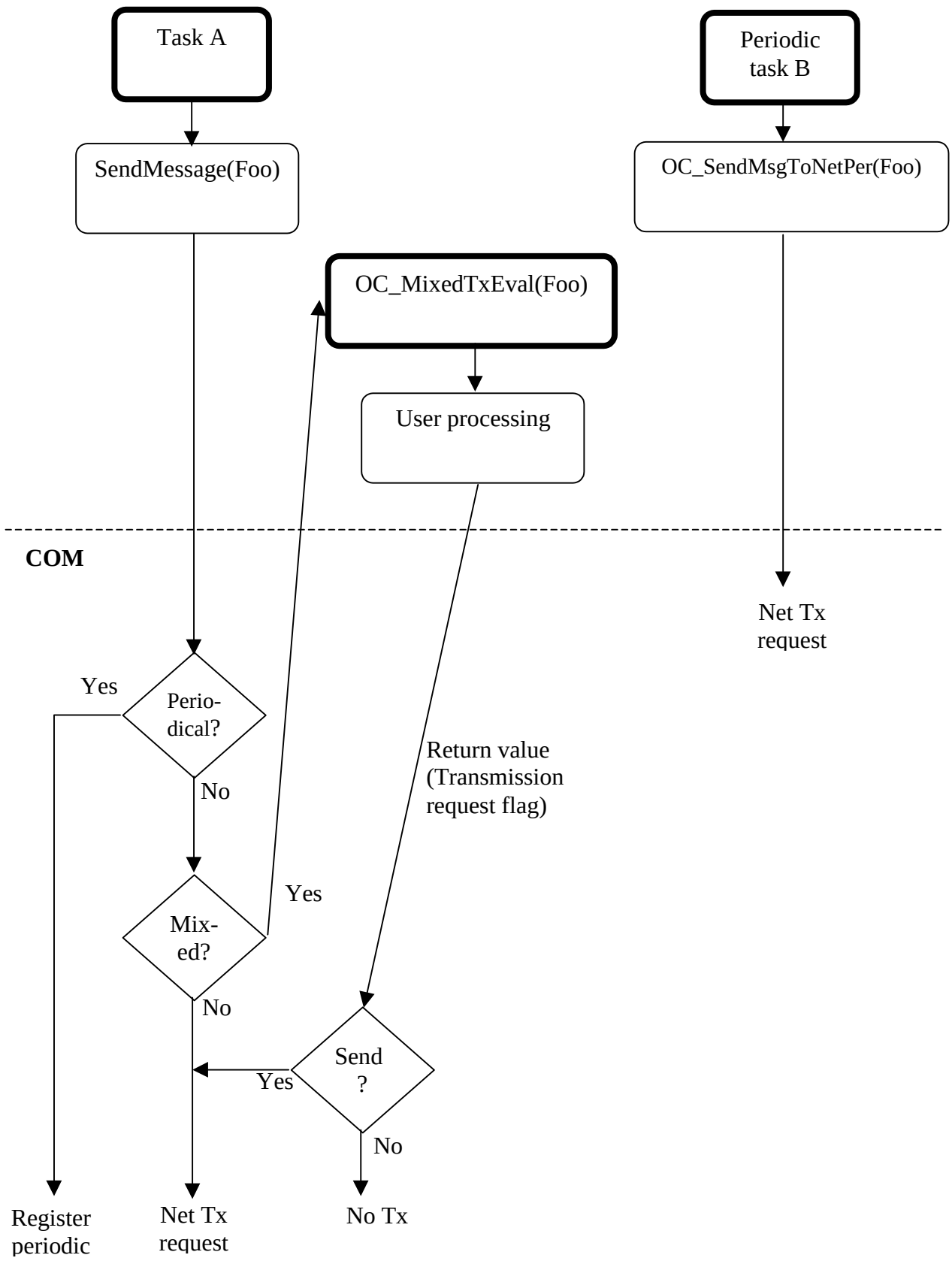

Figure 3.3 Mixed Transmission

Task A runs independently from the periodic task B. Task A calls SendMessage that conditionally requests a transmission to the network. Task B requests a transmission to the network, each time it is activated. The message named *Foo* is configured for mixed transmission mode.

3.2.5 Deadline Monitoring

When CCC1 is used, the deadline monitoring using alarm can be performed by combining with OS. Deadline monitoring means that a deadline is monitored. It can be applied regardless of whether the message is configured for transmission or reception. For transmission, this means that an initiated transmission to the net must be completed within a certain time frame. If not, an alarm expires and as a consequence, a task is activated or an event is set. For reception, a message must be received within a certain time frame. The start of this time frame is the point in time when the previous message was received. If the deadline is broken, an alarm expires and as a consequence, a task is activated or an event is set.

User must define the alarm, the task, and the event.

Deadline monitoring is applicable to messages of type *To Net* or *From Net*.

(1) Transmission

An alarm is started just before COM issues a transmission request to the network. Depending on the transmission mode, this is done in either of

- SendMessage
- OC_SendMsgToNetPer

The alarm is cancelled automatically when COM receives a transmission confirmation from the network.
If a confirmation does not occur within the alarm time, the alarm will expire.

Deadline monitoring may be used in both direct, periodical and mixed transmission modes.

(2) Reception

Start of the deadline monitoring of reception is possible by starting the alarm from a task or by receiving the message that had once specified deadline monitoring.
In reception monitoring, the alarm is never cancelled; only restarted automatically.

Do not start reception monitoring until COM initialisation is complete. This means that the earliest point in time to start the alarm is from the MessageInit() function.

The alarm is restarted each time COM receives a reception indication from the network.
If an indication does not occur within the alarm time, the alarm will expire.

3.2.6 Message Attributes

The following attributes must be configured for each message. All configurations are done with the COM Configurator. Please see the COM Configurator Help File for details.

- Message Name (The character string which starts with an alphabetic character or an underline, followed by zero or more alphabetic characters, underlines, or numbers. A maximum of 32 characters.)
- Data Length (A maximum of 8 bytes)
- Message Type
  - Local
  - To Net
  - From Net
- Notification Function
  - None
  - Event Mask (Event setting)
  - Task ID (Task ID for task activation or event setting)
- Transmission Mode
  - Direct
  - Periodic (only CCC1)
  - Mixed (only CCC1)
- Deadline Monitoring
  - Alarm ID
  - Expiration Time
- HCAN
  - Mailbox Number (0 to 15)
  - Bit Length of CAN ID (11 or 29 bits)
  - CAN ID

Please note the following.
(1) Do not set up multiple messages of the same name.
(2) Do not set up multiple same CAN IDs.

3.3 Networks

3.3.1 HCAN Related Parameters

Each net message has the following settings related to the HCAN:

- Mailbox
- CAN ID

(1) Mailbox

Data exchange between a message and the CAN net is done through a mailbox. This is a data area within the HCAN device that holds message data and control information. Each net message must use a mailbox.
The following mailboxes exist in an HCAN device

- Mailbox for either transmission or reception (mailboxes 1-15)
- Mailbox only for reception (mailbox 0)

Mailboxes 1-15
These mailboxes can be used for *To Net* , or *From Net*. Multiple messages must not be assigned to the same mailbox.

Mailbox 0
Only messages of type *From Net* may use this mailbox. It is intended to receive a group of messages. During reception processing, reception of two or more messages in the mailbox 0 causes as an error, and is not processed.

(2) CAN ID

Each message has a unique identifier (CAN ID) that is used to identify the message on the CAN bus. The CAN ID may consist of either 11 or 29 bits. 11 bits mean that 11 standard ID bits are used in the CAN bus frame. 29 bits mean that 18 extended and 11 standard ID bits are used.

3.3.2 Channel

COM can treat two separate HCAN buses by one ECU. HCAN has the following parameters. All configurations are done with the COM Configurator. Please see the Hardware Manual and the COM Configurator Help File for details.

- Mailbox 0
  - Bit length of CAN ID (11 or 29 bits)
  - CAN ID
  - LAFM (local acceptance filter mask for mailbox 0)
- Operation
  - Transmission priority (mailbox order or ID order)
  - Bit sample point (1 point or 3 points)
- Bit configuration
  - Baud-rate prescaler (BRP)
  - Time segment 1 (TSEG1)
  - Time segment 2 (TESG2)

  - Maximum bit synchronisation width (SJW)
- Interrupts
  - For each HCAN interrupt source: Enabled/disabled

Restriction of a setting of a bit configuration is shown below.
TSEG1 > TSEG2 >= SJW (SJW = 1 to 4)
3 + TSEG1 + TSEG 2= 8 to25 Time Quanta
TSEG2 > B'001 (BRP = B'000000)
TSEG2 > B'000 (BRP > B'000000)

Refer to the Hardware Manual for details.

3.4 COM Architecture

The COM is divided into two main parts that are termed packages.

- Message Package
- Net Package

The Message Package provides the COM API. The Net Package transmits and receives the HCAN network. *To Net* and *From Net* Messages require both packages.

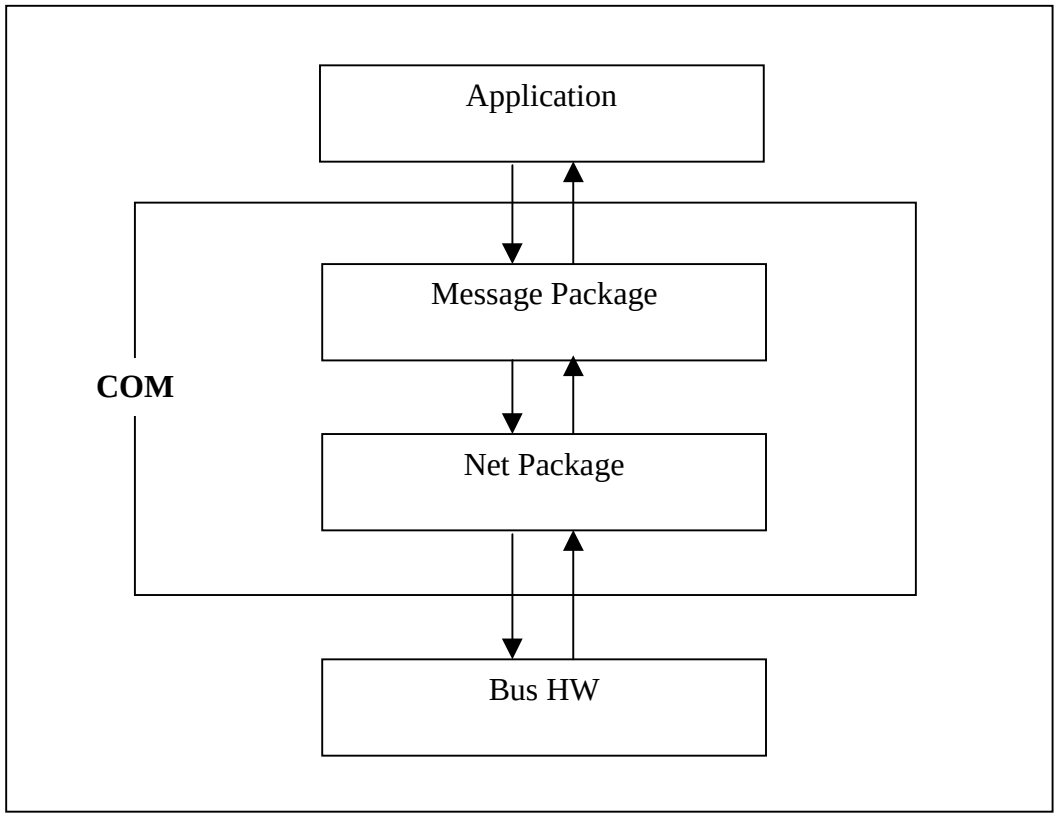

Figure 3.4 COM Architecture

3.5 COM Control

The processing procedure of COM is shown below.

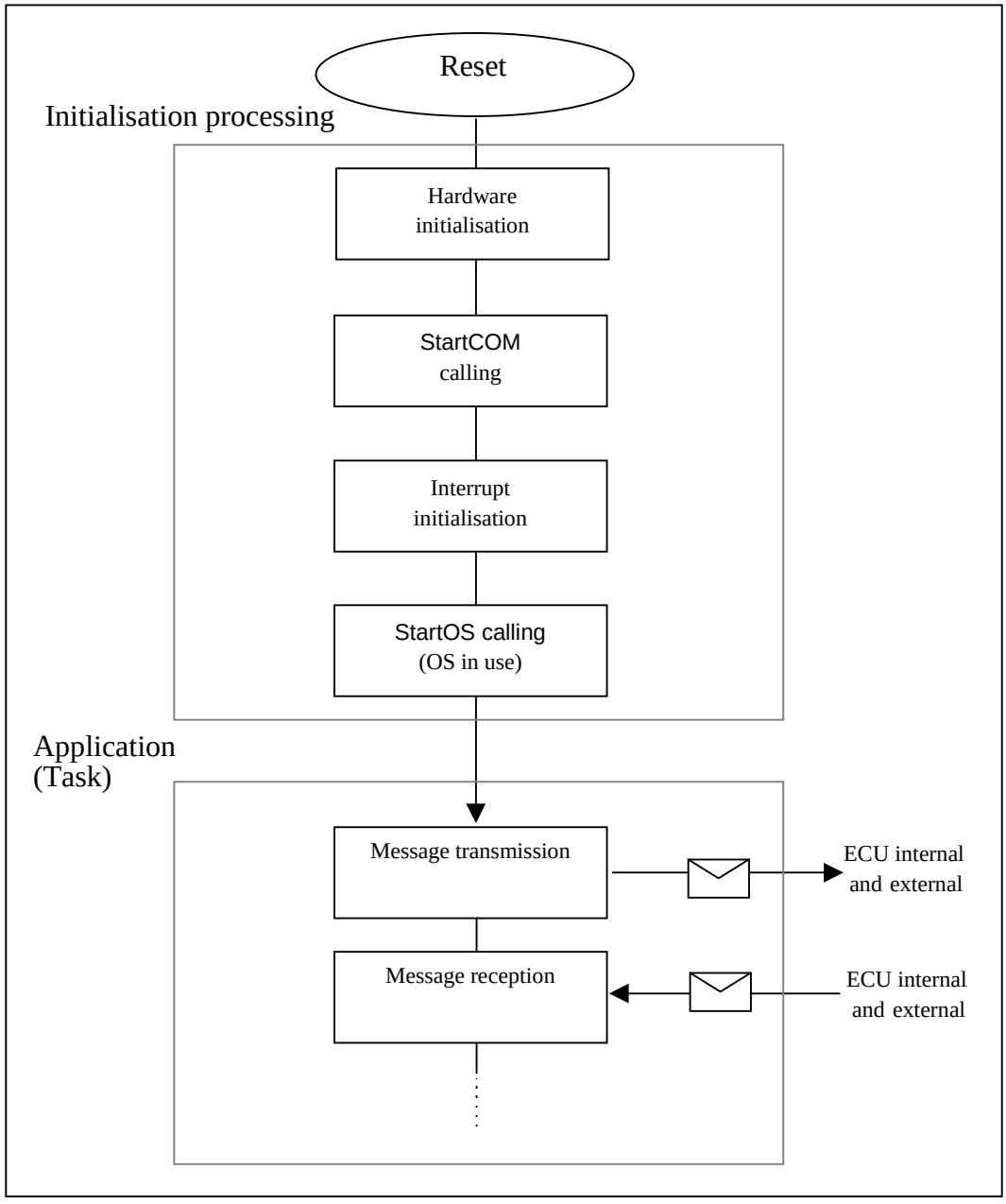

Figure 3.5 COM Processing Procedure

3.5.1 Start Up

The start-up method of COM is

1. HW Reset
2. HW initialisation with interrupts disabled
3. Call StartCOM communication service
4. Interrupt initialisation
5. Call StartOS system service, if OS is used

The application must provide the code for steps 2 and 4. This product provides those codes as sample (see section 4.2).
During start up, no messages may be transmitted or received.

3.5.2 Normal Operation

COM runs in normal mode as soon as start up is finished. Message transmission and reception can be performed in this state. The flow of transmission and reception is shown below.

3.5.2.1 Message Transmission

1. Application calls SendMessage.
2. SendMessage copies message data from application to data area in COM.
3. For a net message, SendMessage starts net transmission.
4. SendMessage returns.
5. The message is sent from the HCAN to the CAN bus.
6. When transmission is finished, a Slot Empty interrupt occurs.
7. The ISR performs a call to the message package and net package of COM. This transmission confirmation makes COM aware that the transmission operation is finished.
8. The ISR returns.

3.5.2.2 Message Reception in Application

1. Application calls ReceiveMessage.
2. ReceiveMessage copies message data from data area in COM to application.
3. ReceiveMessage returns.

3.5.2.3 Message Reception from Network

1. When message is received from the bus, a received-message interrupt occurs.
2. The ISR performs a call to the message package and net package of COM.
3. The COM copies message data from the HCAN to the data area in COM.
4. The ISR returns.

3.5.3 Shut Down

The COM cannot be shut down. When initialising COM, issue the StartCOM communication service again.

3.5.4 Error Handling

The following error mechanisms are provided:

- Error codes returned from communication service
- Error interrupts related to the HCAN device
- Exceptions

3.5.4.1 Error Codes

The COM can operate in one of the following two error modes.

- Standard Status Mode
- Extended Status Mode

The COM with extended status is used in the development and debugging of applications; the COM with standard status is used in fully debugged systems. Section 7.1 gives a summary of the error codes returned by COMMUNICATION services.

3.5.4.2 Error Interrupt

The HCAN module may generate interrupts as described below:

Table 3.3 Interrupts from the HCAN Module

Interrupt Cause	Default Setting	Configurator Setting
Received message	Available (reserved for COM)	Impossible
Unread mail	Available (reserved for COM, user code can be added)	Impossible
Transmission mailbox empty	Available (reserved for COM)	Impossible
Reset	Available (can be changed by the user)	Impossible
Overload frame	Unavailable	Possible
Bus-off	Unavailable	Possible
Error passive	Unavailable	Possible
Bus receive overload warning	Unavailable	Possible
Bus transmit overload warning	Unavailable	Possible
Remote frame request	Unavailable	Possible
Bus operation request	Unavailable	Possible

All interrupts that are not reserved by COM may be used by the application. The COM provides default handlers for all these interrupts. If required, it is possible to add codes. Refer to section 5.3 for details.

3.5.4.3 Exception

COM does not include any exception handling features.
If OS is not used, exceptions must be handled by the application.
If OS is used, exceptions will be handled by the OS. Please refer to the Operating System Manual for details.

4 SYSTEM CONFIGURATION

4.1 COM Application Building

Figure 4.1 shows the building process of a COM application.

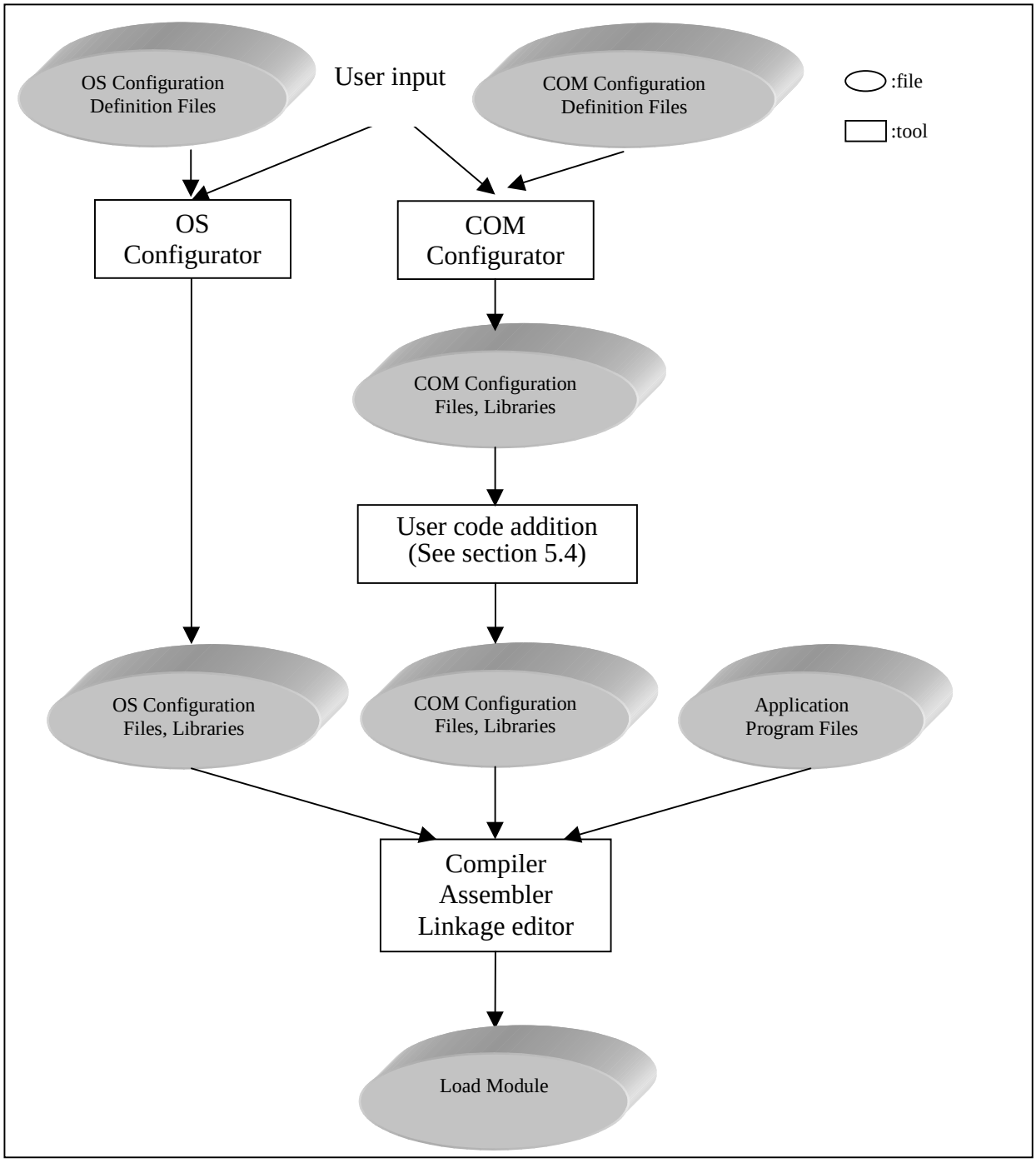

Figure 4.1 Building Process of a COM Application.

The configurator generates the configuration files and the library files determined by the user input or configuration definition file. Along with the application program files, these files are compiled, assembled and linked to generate the load module.

Refer to help files of OS and COM Configurator for information on how to generate configuration files. Please refer to the Operating System Manual on the details of OS configuration file.

4.2 COM Configuration Files

Configuration files are divided into the following file groups:

- **App**
  Sample application files.
- **Com**
  COM libraries, COM objects, COM configuration files and sample files generated by the COM Configurator.
- **Os**
  Sample OS configuration files generated by the OS Configurator.

The **Com** group may be further subdivided into:

- **Evb**
  Sample files related to the target HW set-up.
- **Hcan**
  HCAN driver files.
- **Mcs**
  Sample CPU initialisation and section initialisation files.
- **Msg**
  Message package files.
- **Net**
  Net package files.

If required, rewrite the following files. Note that they are overwritten whenever these files generate a COM configuration file by COM Configurator.

4.2.1 Evb

Table 4.1 Evb Group

File	Description
int_hndl.c	Interrupt handler
ma_cpu.c	Hardware initialisation processing
ma_cpu.h	Hardware initialisation header
ma_int.c	Interrupt initialisation
ma_int.h	Interrupt initialisation header
ma_io.c	Port setting
ma_io.h	Port setting header
ma_pfc.c	Pin function controller (PFC) setting
ma_pfc.h	Pin function controller (PFC) setting header
makeapp.h	Macro

4.2.2 Hcan

Table 4.2 Hcan Group

File	Description
ma_hcan1.c	HCAN channel 0 processing
ma_hcan1.h	HCAN channel 0 header
ma_hcan2.c	HCAN channel 1 processing
ma_hcan2.h	HCAN channel 1 header
mac.h	HCAN macro
macan1.h	HCAN channel 0 macro
macan2.h	HCAN channel 1 macro

4.2.3 Mcs

Table 4.3 Mcs Group

File	Description
dbsct.src	Section initialisation definition
hwsetup.src	Hardware initialisation
initsct.c	Section initialisation
intprg.src	Common interrupt program
resetprg.src	Reset processing
stacksct.src	Reset stack definition
vect.inc	Vector table include file
vecttbl.src	Vector table definition

4.2.4 Msg

Table 4.4 Msg Group

File	Description
occallb.c	Call function for message initialisation and mixed transmission
ocos.h	COM interrupt level setting

5 PROGRAMMING

The programming method of COM application is shown below.

5.1 COM Initialisation

For COM initialisation, call services and functions according to the procedure shown below. COM interrupt level must be maintained until COM initialisation is completed. If not using OS, StartOS system service call is not required.

```
/*--- HW Init ---*/
MA_Init_PFC();              /* Pin function controller (PFC) initialisation */
MA_Init_CPU();              /* Hardware initialisation */
MA_Init_IO();               /* Port initialisation */

/*--- COM Init ---*/
StartCOM();                 /* COM initialisation */
MA_Init_INT();              /* Interrupt initialisation */

/*--- OS Init ---*/
StartOS(application mode);  /* OS initialisation */
```

5.2 Include Files

The program which uses a message must include the ocmsg.h file.

5.3 Interrupts

5.3.1 HCAN Interrupts

There are four vectors in each HCAN as follows, and each interrupt is divided into groups. Each ISR must be defined in a vector table. In CCC0 with OS, register as category 1 interrupt. In CCC1, register as category 2 interrupt. In CCC1, add "_ISR" at the end of ISR name because HCAN interrupt handler is called via interrupt preamble of OS. HCAN ISR name is shown in Table 5.1.

- **ERS vector**
  - Error passive interrupt
  - Bus off interrupt
- **OVR vector**
  - Reset interrupt
  - Remote frame request interrupt
  - Bus transmit overload warning interrupt
  - Bus receive overload warning interrupt
  - Overload frame interrupt
  - Unread mail interrupt
  - Bus operation request interrupt
- **RM vector**
  - Received message interrupt
- **SLE vector**
  - Transmission mailbox empty interrupt

Table 5.1 HCAN ISR Name

vector	CCC0	CCC1
ERS	MA_IntHandler_ERS_HCAN*x*	MA_IntHandler_ERS_HCAN*x*_ISR
OVR	MA_IntHandler_OVR_HCAN*x*	MA_IntHandler_OVR_HCAN*x*_ISR
RM	MA_IntHandler_RM_HCAN*x*	MA_IntHandler_RM_HCAN*x*_ISR
SLE	MA_IntHandler_SLE_HCAN*x*	MA_IntHandler_SLE_HCAN*x*_ISR

Note: *x* is 1 (for HCAN channel 0) or 2 (for HCAN channel 1).

5.3.2 Interrupt Priority Levels

- COM Interrupt Level

The highest interrupt priority level in the program which uses communication service must be set as a COM interrupt level. When the OS is used by CCC0, the COM interrupt level must be set higher than the OS interrupt level. In CCC1, the OS interrupt level is set to the COM interrupt

level. Communication service must not be issued from a program higher than the COM interrupt level. Refer to ocos.h for the setting method of COM interrupt level.

- HCAN Interrupt Level

Both channels 0 and 1 of the HCAN module should be set to the same interrupt priority level. The HCAN interrupt level should be set to the same COM interrupt level. Refer to ma_int.c provided as a sample for the setting method.

5.4 User Code

The COM configuration file is generated by the COM Configurator. However, the user can add some user codes to the following processing:

- Initialisation processing
- Call-back functions
- HCAN interrupts that are not reserved by COM

5.4.1 Initialisation processing

MA_Init_PFC()
This function exists in **ma_pfc.c**. This function is called from main.c provided as a sample.

MA_Init_CPU()
This function exists in **ma_cpu.c**. This function is called from main.c provided as a sample.

MA_Init_IO()
This function exists in **ma_io.c**. This function is called from main.c provided as a sample.

MA_Init_INT()
This function exists in **ma_int.c**. This function is called from main.c provided as a sample.

5.4.2 Call-Back Functions

MessageInit()
This function exists in **occallb.c**. This function is called from the StartCOM service. Refer to section 6.1.2 for details.

OC_MixedTxEval()
This function exists in **occallb.c**. This function is called from within COM when a message is transmitted in the mixed transmission mode. Refer to section 6.2.2 for details.

5.4.3 HCAN ISR

The user can add codes that handle interrupts from the HCAN. For details on interrupt causes, refer to the Hardware Manual.

Table 5.2 User Code Placement for HCAN Interrupts.

Interrupt Cause	**Add User Code In**
Unread mail	int_hndl.c: MY_UnReadMailHandler1() (for HCAN channel 0) MY_UnReadMailHandler2() (for HCAN channel 1)
Reset	int_hndl.c: MY_PowerUpHandler1() (for HCAN channel 0) MY_PowerUpHandler2() (for HCAN channel 1)
Overload frame	macan1.h: MA_OVERLOAD_FRAME_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_OVERLOAD_FRAME_USERCODE2 macro (for HCAN channel 1)
Bus-off	macan1.h: MA_BUS_OFF_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_BUS_OFF_USERCODE2 macro (for HCAN channel 1)
Error passive	macan1.h: MA_ERROR_PASSIVE_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_ERROR_PASSIVE_USERCODE2 macro (for HCAN channel 1)
Bus receive overload warning	macan1.h: MA_REC_WARNING_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_REC_WARNING_USERCODE2 macro (for HCAN channel 1)
Bus transmit overload warning	macan1.h: MA_TEC_WARNING_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_TEC_WARNING_USERCODE2 macro (for HCAN channel 1)
Bus operation request	macan1.h: MA_WU_BUS_ACTIF_USERCODE1 macro (for HCAN channel 0) macan2.h: MA_WU_BUS_ACTIF_USERCODE2 macro (for HCAN channel 1)

5.5 Interface to OS

5.5.1 Interrupt

According to the following category rule, interrupt which uses COM (communication service) must be registered into OS Configurator.

Table 5.3 Category Rule

Interrupt	CCC0	CCC1
HCAN interrupt	Category 1	Category 2
Other interrupts	Category 1 or 2	Category 1 or 2

5.5.2 Alarm and Task

Cyclic alarm and task for a periodic and the mixed transmission modes must be prepared by user application. The task needs to be registered into OS Configurator.

5.6 Registers

The values of general registers R0-R7, FR0-FR11, FPUL, and FPSCR[1] cannot be guaranteed in the communication service. If you use these registers after a communication service call, they must be saved beforehand.

5.7 Stack

The COM never changes the stack. Therefore, add stack size of each service to the stack of the program which calls communication service. Moreover, add the interrupt stack size of each interrupt to the interrupt stack. Refer to section 7.6 for details on stack size.

[1] The FR0-FR11, FPUL, and FPSCR registers are only valid for the processor with Floating Point Unit (FPU).

5.8 Calling a Communication Service From an Assembler Routine

Communication services may be called from assembler routines. In this case, the application programmer must branch to the start address of each communication service by the JSR command. Follow the rules governing parameter area allocation. For the type of a parameter, refer to Table 7.4 Data Types, and C-language header file of Msg group of configuration files.

Table 5.4 Argument Convention

Register	Argument number
R4	First argument
R5	Second argument
R6	Third argument
R7	Fourth argument

General register R0 is used for the return value.

The values of general registers R0-R7, PR, FR0-FR11, FPUL, and FPSCR cannot be guaranteed before and after calling the communication service. If you use these registers, they must be saved before the call, and restored after the call.

5.9 Notes on Assembler Use

When the COM is initialised from an assembler routine, call each initialisation processing of a MA_Init_PFC() function, etc. by JSR. Although assembly language can describe Evb and Mcs groups of a configuration file, be sure to describe Hcan, Msg, and Net groups in the C language.

6 API DESCRIPTION

The interface of the communication service is described below. Refer to section 5.8 on an assembler interface.

6.1 Standard Interface

6.1.1 StartCOM

Syntax: StatusType StartCOM(void)

Parameter (In):
None
Parameter (Out):
None

Description:

COM is initialised. In the end, this service calls call-back function MessageInit().

Error Status:

Standard
No error: E_OK

Extended (Code added as extended status)
None

6.1.2 MessageInit

Syntax: StatusType MessageInit(void)

Parameter (In):
None
Parameter (Out):
None

Description:

COM calls this function automatically by StartCOM service. This function is provided by the occallb.c file. Rewrite if needed.

Error Status:

Standard

No error:	E_OK
Initialise failed:	E_COM_SYS_MSG_INIT_FAILED

Extended (Code added as extended status)
None

6.1.3 SendMessage

Syntax: StatusType SendMessage(OC_SymbolicNameT <Msg>, OC_DataRefT <Data>)

Parameter (In):
Msg Message name
Data The address of the area where transmitting data was stored
Parameter (Out):
None

Description:

The transmit data is copied from the data area specified by application to the message data area in COM. In the case of a network message, transmission is started for a network. In this case, this service returns before the completion of transmission. When the periodic message which can be used by periodic transmission or mixed transmission is transmitted, transmission to a network is not started. Transmission to a network is performed by calling OC_SendMsgToNetPer service from periodic task. Moreover, in mixed transmission, it is possible to start transmission to a network by conditioning.

Error Status:

Standard

No error:	E_OK
Message is locked:	E_COM_LOCKED
An event could not be set as notification:	E_COM_SYS_EVENT_SETTING_DENIED
A task could not be started as notification:	E_COM_SYS_TASK_ACTIVATION_DENIED
An alarm for deadline monitoring could not be started:	E_COM_SYS_ALARM_START_DENIED
Transmission was denied by the net package:	E_COM_SYS_NET_TX_DENIED

Extended (Code added as extended status)

Invalid message:	E_COM_ID

6.1.4 ReceiveMessage

Syntax: StatusType ReceiveMessage(OC_SymbolicNameT <Msg>,
OC_DataRefT <Data>)

Parameter (In):
Msg Message name
Data The area and address for received data
Parameter (Out):
Data Received data

Description:

Received data is copied from the message data area in COM to the data area specified by the application. This service does not wait for reception of message data. Message data received last which exists in COM is only copied.

Error Status:

Standard
No error: E_OK
After initialising message data, it has not updated: E_COM_NOMSG

Extended (Code added as extended status)
Invalid message: E_COM_ID

6.1.5 GetMessageStatus

Syntax: StatusType GetMessageStatus(OC_SymbolicNameT <Msg>)

Parameter (In):
Msg Message name
Parameter (Out):
None

Description:

Returns the status of the message.

Error Status:

Standard
No error: E_OK
Message is locked: E_COM_LOCKED
After initialising message data, it has not updated: E_COM_NOMSG

Extended (Code added as extended status)
Invalid message: E_COM_ID

6.2 Original Interface Functions

In addition to the communication services defined by the OSEK COM specification, the **original interface** is provided.

6.2.1 OC_SendMsgToNetPer

Syntax:

enum OC_ILOpResultE OC_SendMsgToNetPer(OC_SymbolicNameT <Msg>)

Parameter (In):
Msg Message name
Parameter (Out):
None

Description:

Transmission to the network of a periodic message is started. Only the message for periodical or mixed transmission can be used. This service must be called from task activated periodically. The SendMessage service must be issued before transmitting to a network periodically by this service. Message data transmitted by SendMessage communication service is transmitted periodically. When not issuing SendMessage service, this service does not return an error but transmits undefined data to a network.

Error Status:

Standard

No error:	OC_ILOpResOK
Message is locked:	OC_ILOpResMsgDsInUse
Alarm for deadline monitoring could not be started:	OC_ILOpResError
Transmission was denied by the net package:	OC_ILOpResNetTxDenied

Extended (Code added as extended status)

Invalid message:	OC_ILOpResError

6.2.2 OC_MixedTxEval

Syntax:

OC_CallbStatusT OC_MixedTxEval(OC_SymbolicNameT <pMsgH>)

Parameter (In):
pMsgH Message name
Parameter (Out):
None

Description:

The purpose is to evaluate a condition that tells COM whether to initiate or not a transmission to network. Only the message for mixed transmission can be used. This function is automatically called within COM at the time of the message transmission by SendMessage service. Since this function is provided by the occallb.c file, the application programmer is responsible for adding the code that evaluates the transmission condition. Since pMsgH is not an integer, a judgement by the “switch” statement cannot be performed. It can judge by the "if" statement. The evaluation must result in either of the following return values.

Error Status:

Standard
Initiate a transmission to net: OC_TxCondIsSendD
Do NOT initiate a transmission to net: OC_TxCondIsNoSendD

Extended (Code added as extended status)
None

7 APPENDIX

7.1 Communication Service Return Codes

Table 7.1 Communication Service Return Codes

Communication Service	Standard Error Status	Code Added in Extended Error Status
StartCOM	E_OK	---
MessageInit	E_OK E_COM_SYS_MSG_INIT_FAILED*	---
SendMessage	E_OK E_COM_LOCKED E_COM_SYS_EVENT_SETTING_DENIED* E_COM_SYS_TASK_ACTIVATION_DENIED* E_COM_SYS_ALARM_START_DENIED* E_COM_SYS_NET_TX_DENIED*	E_COM_ID
ReceiveMessage	E_OK E_COM_NOMSG	E_COM_ID
GetMessageStatus	E_OK E_COM_NOMSG E_COM_LOCKED	E_COM_ID
OC_SendMsgToNetPer*	OC_ILOpResOK * OC_ILOpResMsgDsInUse * OC_ILOpResError * OC_ILOpResNetTxDenied *	OC_ILOpResError *
OC_MixedTxEval*	OC_TxCondIsSendD * OC_TxCondIsNoSendD *	---

Note: *: Original function

7.2 Return Code IDs

Table 7.2 Return Code IDs

Return Code	Description	ID
E_OK	No error	0 (H'0)
E_COM_ID	Invalid message	1 (H'1)
E_COM_LIMIT	Reserved (not used)	2 (H'2)
E_COM_NOMSG	After initialising message data, it has not updated	3 (H'3)
E_COM_LOCKED	Message is locked	4 (H'4)
E_COM_SYS_NET_TX_DENIED*	Transmission was denied by the net package	64 (H'40)
E_COM_SYS_EVENT_SETTING_DENIED*	An event could not be set as notification	65 (H'41)
E_COM_SYS_TASK_ACTIVATION_DENIED*	A task could not be started as notification	66 (H'42)
E_COM_SYS_ALARM_START_DENIED*	An alarm for deadline monitoring could not be started	67 (H'43)
E_COM_SYS_UNEXPECTED_STATE*	Reserved (not used)	68 (H'44)
E_COM_SYS_MSG_INIT_FAILED*	Initialise failed	69 (H'45)
OC_ILOpResOK*	No error	256(H'100)
OC_ILOpResError*	Invalid message Alarm for deadline monitoring could not be started	257 (H'101)
OC_ILOpResMsgDsInUse*	Message is locked	258 (H'102)
OC_ILOpResNetTxDenied*	Transmission was denied by the net package	259 (H'103)
OC_ILOpResMsgUnexpectedState*	Reserved (not used)	260 (H'104)
OC_ILOpResToNetMsgNotSupported*	Reserved (not used)	261 (H'105)
OC_ILOpResFrNetMsgNotSupported*	Reserved (not used)	262 (H'106)
OC_TxCondIsSendD*	Initiate a transmission to net	1 (H'1)
OC_TxCondIsNoSendD*	Do NOT initiate a transmission to net	2 (H'2)

Note: *: Original function

7.3 Communication Service Calls

In calling a program, “Yes” shows communication service which can be issued. "---" means that operation is not guaranteed.

Table 7.3 Communication Service Calls

Communication service	Before COM initial-isation (Indispens-able) CCC0/ CCC1	CCC0	CCC1						
			Task	ISR	Error hook	PreTask hook	PostTask hook	Startup hook	Shutdown hook
StartCOM	Yes	---	---	---	---	---	---	---	---
SendMessage	---	Yes	Yes	Yes	---	---	---	---	---
ReceiveMessage	---	Yes	Yes	Yes	Yes	---	---	---	---
GetMessageStatus	---	Yes	Yes	Yes	Yes	---	---	---	---
OC_SendMsgToNetPer*	---	---	Yes	---	---	---	---	---	---

Note: *: Original function

7.4 Data Types

Table 7.4 Data Types

Name	Type(size)	Description
StatusType	unsigned long (4 bytes)	Return value
OC_SymbolicNameT	struct OC_MsgHandleS * (4 bytes)	Message name
OC_DataRefT	void * (4 bytes)	Address or pointer for message data
OC_ILOpResultE	enum (4 bytes)	Return value
OC_CallbStatusT	unsigned char (1 byte)	Return value

7.5 Maximum Parameters

Table 7.5 Maximum Parameters

Item	Limit
Max number of Nets per COM *	2
Max number of Local messages	1000
Max number of To Net messages per Net *	15
Max number of From Net messages per Net *	31
Max number of messages for mailbox 0 *	16
Max number of messages for mailboxes 1-15 *	1
Max number of messages per Net *	31

Note: *: They are based on hardware specification.

7.6 Stack Requirements

Table 7.6 Stack Requirements

Service, ISR			**Stack Usage [Byte]**
StartCOM			72 + stack used with MessageInit() function (default is 0)
SendMessage	Local message		32
	To Net message	Direct	104
		Periodical	32
		Mixed	104 + stack used with OC_MixedTxEval() function (default is 0)
ReceiveMessage			0
GetMessageStatus			0
OC_SendMsgToNetPer	To Net message	Periodical	88
ISR of Receive Message Interrupt from HCAN			192
ISR of Transmit Mailbox Empty Interrupt from HCAN			176

Ho7055 Communication Manual

Publication date: 1st Edition, May 2001
Copyright (c) Hitachi, Ltd., 1999. All rights reserved. Printed in Japan.