

NEC

User's Manual

DAFCAN

CAN Interface with Diagnosis Support

Hardware

Addendum to AFCAN User's Manual

Document No. U18569EE1V1UM00
Date Published December 2008

© NEC Corporation 2008
Printed in Germany

[MEMO]

NOTES FOR CMOS DEVICES

① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (MAX) and V_{IH} (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (MAX) and V_{IH} (MIN).

② HANDLING OF UNUSED INPUT PINS

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ PRECAUTION AGAINST ESD

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ STATUS BEFORE INITIALIZATION

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

⑤ INPUT OF SIGNAL DURING POWER OFF STATE

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

All other product, brand, or trade names used in this publication are the trademarks or registered trademarks of their respective trademark owners.

Product specifications are subject to change without notice. To ensure that you have the latest product data, please contact your local NEC Electronics sales office.

Legal Notes

- The information in this document is current as of December 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).
"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact NEC Electronics sales representative in advance to determine NEC Electronics 's willingness to support a given application.

- Notes:**
1. "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
 2. "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).
 3. SuperFlash[®] is a registered trademark of Silicon Storage Technology, Inc. in several countries including the United States and Japan. This product uses SuperFlash[®] technology licensed from Silicon Storage Technology, Inc.

*For further information,
please contact:*

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
<http://www.necel.com/>

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
<http://www.am.necel.com/>

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
<http://www.eu.necel.com/>

Hanover Office
Podbielski Strasse 166 B
30177 Hanover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52180
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiú, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
TEL: 010-8235-1155
<http://www.cn.necel.com/>

NEC Electronics Shanghai Ltd.
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai P.R. China P.C:200120
Tel: 021-5888-5400
<http://www.cn.necel.com/>

NEC Electronics Hong Kong Ltd.
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886-9318
<http://www.hk.necel.com/>

Seoul Branch
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
<http://www.sg.necel.com/>

G06.6-1A

INTRODUCTION

Readers This manual is intended for users who want to understand the functions of the DAFCAN and to design application systems using the functions.

Purpose This manual presents the hardware manual of DAFCAN.

Organization This system specification describes the following sections:

How to Read This Manual

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

To understand the overall functions of the DAFCAN macro:
Read this manual in the order of the CONTENTS.

Bars on the page borders show major revised points.

Legend

Symbols and notation are used as follows:

Weight in data notation : Left is high-order column, right is low order column

Active low notation : $\overline{\text{xxx}}$ (pin or signal name is over-scored) or
/xxx (slash before signal name)

Memory map address: : High order at high stage and low order at low stage

Note : Explanation of (Note) in the text

Caution : Item deserving extra attention

Remark : Supplementary explanation to the text

Numeric notation : Binary . . . xxxx or xxxB
Decimal . . . XXXX
Hexadecimal . . . xxxXH or 0x xxxx

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo) : $2^{10} = 1024$

M (mega) : $2^{20} = 1024^2 = 1,048,576$

G (giga) : $2^{30} = 1024^3 = 1,073,741,824$

TABLE OF CONTENTS

Chapter 1	INTRODUCTION	17
1.1	Overview of Functions	18
Chapter 2	ARCHITECTURE	19
2.1	CPU I/F	20
2.2	Global Macro Control	20
2.3	CAN Interrupt Generator	20
2.4	Message Control (MSG Ctrl)	20
2.5	Arbitration Logic	21
2.6	RXONLY_CH CAN machine	21
2.7	DIAG_CH CAN machine	21
2.8	Memory and Register Layout	22
Chapter 3	Macro Initialisation and Control	23
3.1	Global Macro Initialisation and Control	24
3.1.1	Global Macro Disabled (GOM=0)	24
3.1.2	Initialisations before Enabling the Global Macro	24
3.1.3	Global Macro Enabled	24
3.1.4	Switch-Off the Global Macro	24
3.1.5	Flowchart of the Initialisation	27
Chapter 4	Message Buffer Initialisation and Configuration	31
4.1	Message Buffer Initialisation	31
4.2	Message Buffer to CAN I/F Channel Assignment	32
4.3	Configuration before switching to Mirror Mode	34
4.4	Configuration before switching to DIAG Side	36
Chapter 5	DAFCAN Macro Initialisation and Control	37
5.1	CAN Bit Time Programming	37
5.1.1	Transitions for Operational Modes of DIAG_CH	38
5.1.2	Transition for Operational Modes of RXONLY_CH	40
Chapter 6	Macro Interrupts	41
Chapter 7	Message Reception and Transmission	44
7.1	Principal Reception Process	44
7.2	Reception History	44
7.3	Reception of Remote Frames	44
7.4	Message Transmission	44
Chapter 8	Operational Modes of RXONLY_CH	45
8.1	Receive-Only Mode	45
8.1.1	Processing received Frames in Receive-Only Mode of RXONLY_CH	47
8.2	Mirror Mode	49
8.2.1	Operations of the Mirror Mode Engine	49
8.3	Mirror Mode with TIF	51

TABLE OF CONTENTS

Chapter 9	Transitions for Buffer Assignment	52
Chapter 10	Register Description	54
10.1	Registers of Global Macro Control	54
10.1.1	CAN Transfer ID Reference Registers (CnTIDRmL, CnTIDRmH; m = 0 - 7)	57
10.1.2	CAN Transfer ID Mask Registers (CnTIDML, CnTIDMH)	62
10.2	Registers of the CAN Module	63
10.2.1	RXONLY_CH CAN Module Control Register (CnCTRL_R)	67
10.2.2	RXONLY_CH CAN Module Last Error Code Register (CnLEC_R)	68
10.2.3	RXONLY_CH CAN Module Error Counter Register (CnERC_R)	69
10.2.4	RXONLY_CH CAN Module Interrupt Enable Register (CnIE_R)	70
10.2.5	RXONLY_CH CAN Module Interrupt Status Register (CnINTS_R)	71
10.2.6	RXONLY_CH CAN Module Bit-Rate Prescaler Register (CnBRP_R)	72
10.2.7	RXONLY_CH CAN Module Bit Rate Register (CnBTR_R)	73
10.2.8	RXONLY_CH CAN Module Last In-Pointer Register (CnLIPT_R)	74
10.2.9	RXONLY_CH CAN Module Time Stamp Register (CnTS_R)	75
10.2.10	RXONLY_CH CAN Module Bus Selector (CnBSEL_R)	76
10.3	Registers of Message Buffers	77
10.3.1	CAN Message Configuration Register (CnMCONFm)	79
10.3.2	CAN Message Identifier Registers (CnMIDLm, CnMIDHm)	80
10.3.3	CAN Message Control Register (CnMCTRLm)	81

LIST OF FIGURES

Figure 1-1:	Diagnosis Concept using DAFCAN and 4 other CAN Channels	17
Figure 2-1:	Architecture of DIAG Macro	19
Figure 2-2:	Register Address Layout	22
Figure 3-1:	State Transitions of Global Macro Switching.....	23
Figure 3-2:	Shutdown Process (Normal Shutdown)	25
Figure 3-3:	Shutdown Process (Forced Shutdown).....	26
Figure 3-4:	DAFCAN Macro Initialisation.....	27
Figure 3-5:	Re-initialisation of the DIAG_CH CAN Module using 48 Buffers.....	28
Figure 3-6:	Re-initialisation of the DIAG_CH CAN Module using TX ID Filter Function	29
Figure 3-7:	Re-initialisation of the DIAG_CH CAN Module using first 32 Buffers only	30
Figure 4-1:	CPU Write Access to a Message Buffer.....	31
Figure 4-2:	Operation in Mirror or Receive-only Mode of RXONLY_CH.....	32
Figure 4-3:	Operation during INIT or STOP of RXONLY_CH.....	33
Figure 4-4:	Set up of Mirror Mode, Mirror Mode w/ TIF or Receive-Only Mode for RXONLY_CH	35
Figure 4-5:	Cancellation of Mirror Mode or Receive-Only Mode for RXONLY_CH	36
Figure 5-1:	Transitions for Operational Modes of the DIAG_CH CAN module.....	38
Figure 5-2:	Transitions for operational modes of the RXONLY_CH CAN module.....	40
Figure 6-1:	Schematic of the DAFCAN Macro Interrupt Generation (1/2)	42
Figure 8-1:	Reception Process of Receive-Only Mode.....	45
Figure 8-2:	Receive Operation for RXONLY_CH in Receive-Only Mode	48
Figure 8-3:	Mirror Mode without Application Communication on DIAG_CH	50
Figure 8-4:	Mirror Mode with interleaved Application Communication on DIAG_CH.....	51
Figure 9-1:	Changing the Assignment of the upper 16 Buffers.....	52

LIST OF FIGURES

LIST OF TABLES

Table 1-1:	Outline of DIAG Macro Functions.....	18
Table 4-1:	Minimum Configuration of Message Buffer even when unused in the Application.....	31
Table 4-2:	Message Buffer Assignment versus PS/OPMODE	33
Table 6-1:	List of all Macro Interrupt Sources.....	41
Table 8-1:	Differences in Reception Process between regular AFCAN and RXONLY_CH	46
Table 10-1:	Global Macro Registers (CPU read access).....	54
Table 10-2:	Global Macro Registers (CPU write access)	56
Table 10-3:	CAN Module Registers (CPU read access).....	63
Table 10-4:	CAN Module Registers (CPU write access)	65
Table 10-5:	CAN Message Buffer Registers (CPU read access)	77
Table 10-6:	CAN Message Buffer Registers (CPU write access).....	78

LIST OF TABLES

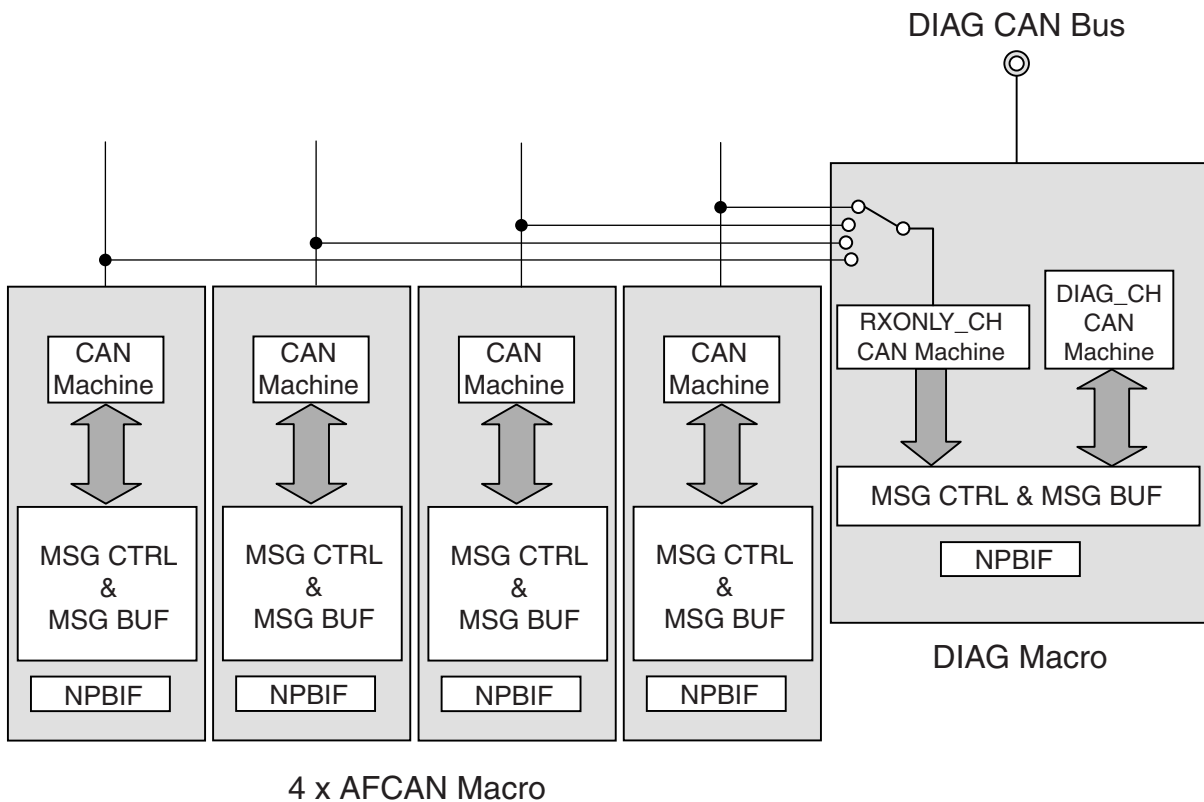
CHAPTER 1 INTRODUCTION

The DAFCAN (Diagnostic AFCAN) macro is in first place an ordinary single channel AFCAN macro with 48 message buffers. In difference to the single channel AFCAN macro the DAFCAN macro features two CAN channels. The channel, which can be operated as an ordinary CAN interface, is called diagnostic channel (DIAG_CH).

For specific diagnostic purposes of the application a second CAN interface is available. This second channel has only limited operational modes because its main purpose is to listen only to the data traffic of one of the other single channel AFCAN macros. This channel is called listen-only channel (RXONLY_CH). It provides the mirror function, which transfers the messages from the source CAN bus to the diagnostic channel from where these messages are automatically sent onto the CAN bus. Furthermore, it provides the mirror mode with the Transfer ID Filter function (w/ TIF). This mode can transmit particular ID only from the source CAN channel to destination channel automatically.

The mirror function as part of a diagnosis concept including other CAN channels is shown in Figure 1-1.

Figure 1-1: Diagnosis Concept using DAFCAN and 4 other CAN Channels



The RXONLY_CH and the DIAG_CH have independent control registers for power save (PSMODE) and operational (OPMODE) modes. The combination of both affects the message buffer allocation for the RXONLY_CH. When RXONLY_CH is in PSMODE == STOP or in OPMODE == INIT, the upper 16 message buffer are assigned to the DIAG_CH. This offers the opportunity to use the DAFCAN macro as an ordinary CAN channel with 48 message buffers when monitoring of other CAN channels is not necessary.

CHAPTER 1 INTRODUCTION

1.1 Overview of Functions

Table 1-1: Outline of DIAG Macro Functions

Feature	Details
Protocol	Active support of extended frame format (international standard ISO11898)
Channels	RXONLY_CH, DIAG_CH
Baud rate	Max. 1Mbit/s @ $f_{CAN} \geq 16$ MHz
Message Storage	RAM area with shared access (Accessing entities: CPU, RXONLY_CH CAN, DIAG_CH CAN) The DAFCAN macro features 48 message buffers. The upper 16 buffer are assigned to the RXONLY_CH depending on operational and power save mode.
Message Organisation	Each message buffer can be initialised either to be a transmit message buffer or a receive message buffer. A group of message buffers assigned as receive buffer can form a multi-buffer receive block. A group of message buffers assigned as transmit buffer can be used for automatic block transfer.
Masks (DIAG_CH only)	The DIAG_CH features 4 masks. Each mask can be assigned to each message buffer. There is no difference between global and local masks within one CAN module. Mirror mode does not provide any masks. All messages are copied to DIAG_CH. Mirror mode w/ TIF of the RXONLY_CH provides 8 reference transfer ID register (CnTIDRnL/H) and 1 mask transfer ID register (CnTIDML/H) for filtering. Only message matching the filter criteria are stored in the upper 16 message buffers. Only those messages stored here participate in the Mirror mode.
Application Support for Message Handling	The DIAG_CH provides a fast mechanism to find receive message buffers with updated content (Reception History List). The DIAG_CH provides a fast mechanism to find the transmit message buffers, from which a message frame has been sent (Transmission History List). The RXONLY_CH does not provide these history lists.
Remote Frame Support	Remote frame handling by message buffer defined for transmit
Time Stamp Functions Note 1 (DIAG_CH only)	Time stamp upon message reception Trigger for time stamp capture selectable (SOF or EOF detection in a CAN message frame) Append time stamp on transmission (specific bytes in the data field are replaced by the captured time stamp) Note 2
Diagnostics	The DAFCAN macro features a mirror function. Messages received by RXONLY-CH are automatically sent on the DIAG_CH. The RXONLY-CH input can be switched to any of the other CAN channels of the device. Readable error counters Valid protocol activity flag for verification of bus connection "Receive-only Mode" CAN protocol error type decoding "Self-test Mode" (DIAG_CH only)
Power Save Modes	Sleep Mode: Wake up from CAN bus Stop Mode: No wake up from CAN bus The RXONLY_CH CAN I/F module and the DIAG_CH CAN I/F module can be configured independently to all these power save modes respectively. The whole macro consumes minimum power when both CAN I/F modules are in such modes simultaneously.

Notes: 1. The architecture of the DAFCAN macro is prepared for the integration of a TTCAN Module, but not all DAFCAN macro derivatives are equipped with a TTCAN Module.

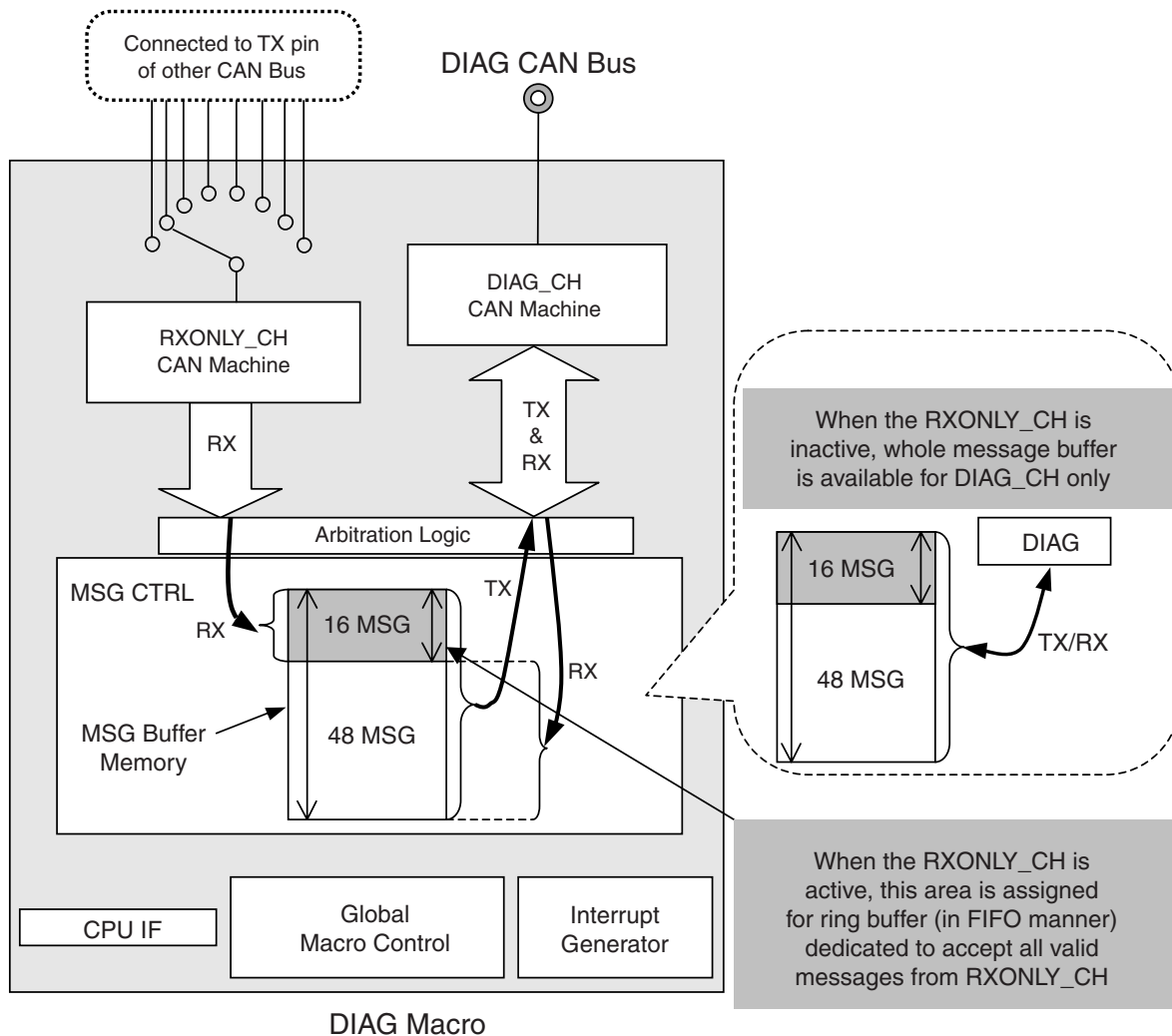
2. The 'Append time stamp on transmission' function is offered only in AFCAN/DAFCAN macro derivatives, which are equipped with the Advanced Time Stamp function.

CHAPTER 2 ARCHITECTURE

The DAFCAN macro is connected to a host CPU and provides two CAN bus interfaces. The DIAG_CH CAN I/F module is dedicated for a particular CAN bus, the diagnostic CAN bus. The RXONLY_CH CAN I/F module can be connected to several CAN buses via a programmable selector. The selector is located within the DAFCAN macro and provides connectivity for up to 8 sources.

The figure below shows an example of DAFCAN macro with 8 inputs for other CAN buses and 48 message buffers.

Figure 2-1: Architecture of DIAG Macro



The DAFCAN macro contains several sub-blocks that are described in the following chapters.

2.1 CPU I/F

The CPU I/F is the interface of the entire macro to the peripheral bus of the microcomputer system. The CPU I/F enables the CPU to access the control and status registers of all sub-blocks within the macro directly. The addresses of the control and status registers are mapped into the memory map of the microcomputer system.

The interrupt signals from the CAN Interrupt Generator to the Interrupt Controller of the microcomputer system are routed via the CPU I/F.

The macro clock f_{CAN} is connected to a peripheral macro clock of the clock generator of the microcomputer system.

2.2 Global Macro Control

The Global Macro Control sub-block contains the logic that enables or disables the entire macro. As well the macro clock f_{CANMOD} is set here.

2.3 CAN Interrupt Generator

In the CAN Interrupt Generator all interrupt request signals from the CAN I/F channels are collected. Appropriated signal shaping is applied to those interrupt request signals for a proper connection to the interrupt request signal inputs of the interrupt controller in the microcomputer system (i.e. pulse generation).

2.4 Message Control (MSG Ctrl)

The MSG Ctrl sub-block provides the memory for the message buffers and its control.

The upper 16 message buffers have no static assignment to RXONLY_CH or DIAG_CH. The assignment of the message buffers depends on the status of RXONLY_CH. The basic functions of these message buffers are:

When assigned to the RXONLY_CH:

- Receive all data frames from RXONLY_CH without any acceptance filtering when Mirror mode or Receive-Only mode is selected. In case of the Mirror mode w/ TIF, data/remote frames are filtered by mask register and reference ID registers.
- Receive all remote frames from RXONLY_CH without any acceptance filtering.
- Behave as ring buffer in FIFO manner. Storage of received messages is calculated with help of LIPT of RXONLY_CH.
- Automatically send all messages to DIAG_CH in FIFO manner. The TX-search state machine of the DIAG-CH takes care to send these message on the diagnostic bus. Any TRQ by a FIFO candidate needs to start a TX-search, which always starts at message buffer #0. The TX-search always must be applied to both areas, the DIAG_CH message buffers and the FIFO.

When assigned to the DIAG_CH:

- Send data frames to DIAG_CH
- Receive data frames from DIAG_CH
- Receive remote frames from DIAG_CH
- Send remote frames to DIAG_CH
- Each message buffer contains its own identifier. Therefore a message buffer can be reserved to receive a data frame unambiguously or to transmit only a particular message frame.
- A mask can be assigned to a message buffer in case the reception of a group of data frames is required.
- Furthermore, message buffers can be grouped to build multi-buffer receive blocks (MBRB).

In other words, the complete message buffer memory (48 buffers) are fully assigned to DIAG_CH with all related features of an ordinary AFCAN channel.

The lower message buffers (#0 to #31) are always assigned to the DIAG_CH.

2.5 Arbitration Logic

The Arbitration Logic sub-block controls the access of the RXONLY_CH and the DIAG_CH to the MSG Ctrl sub-block.

2.6 RXONLY_CH CAN machine

The RXONLY_CH CAN machine sub-block contains logic of the CAN protocol transfer layer, which is dedicated to monitor a CAN BUS chosen by the selector. It never transmits any dominant bit on the monitored bus. The RXONLY_CH is only connected to the monitored CAN channel via the RX-input terminal.

2.7 DIAG_CH CAN machine

The DIAG_CH CAN machine sub-block contains all logic for the CAN Protocol transfer layer. All protocol activities required for the reception of data frames, the transmission of message frames and the CAN protocol error management are executed automatically.

For transmission the DIAG_CH CAN machine uses always all 48 message buffers. However the storage of received messages depends on the state of the RXONLY_CH CAN machine. If the RXONLY_CH CAN machine is 'in power save mode' (PSMODE = STOP) or in initialisation mode (OPMODE = INIT), all message buffer are assigned to the DIAG-CH. When RXONLY_CH CAN machine is online or in sleep mode while the operating mode 'RXONLY' or 'Mirror Mode' is selected, only the lower 32 message buffer (#0 - #31) are used to store received messages from the DIAG_CH CAN machine.

Note: When preparing for mirror mode while the upper 16 message buffers are still assigned to DIAG_CH, the application needs to clear all pending TRQ bits of these buffers at first. Then, all RDY of these buffers need to be cleared as well. Then all these buffer need to be configured as receive message buffers before the initialisation state for the RXONLY_CH is released; the buffers are assigned to RXONLY_CH.

The acceptance filtering process, which decides whether a received data frame has to be stored into one of the assigned message buffers, is executed in the CAN module. The acceptance filtering process manages the storage into message buffers with and without an assigned mask, into multi-buffer receive blocks, and it resolves ambiguous storage situations.

CHAPTER 2 ARCHITECTURE

In case more than one of the assigned message buffers are defined as transmit message buffer, more than one transmit request may be pending when the CAN module is able to get access to the CAN bus. The CAN module automatically detects the pending transmit request with the highest priority, which is evaluated by specific rules, and processes the transmission.

2.8 Memory and Register Layout

Figure 2-2: Register Address Layout

07FH	CAN Module Register (for both RXONLY and DIAG)
040H	
03FH	Global Macro Control Register
000H	

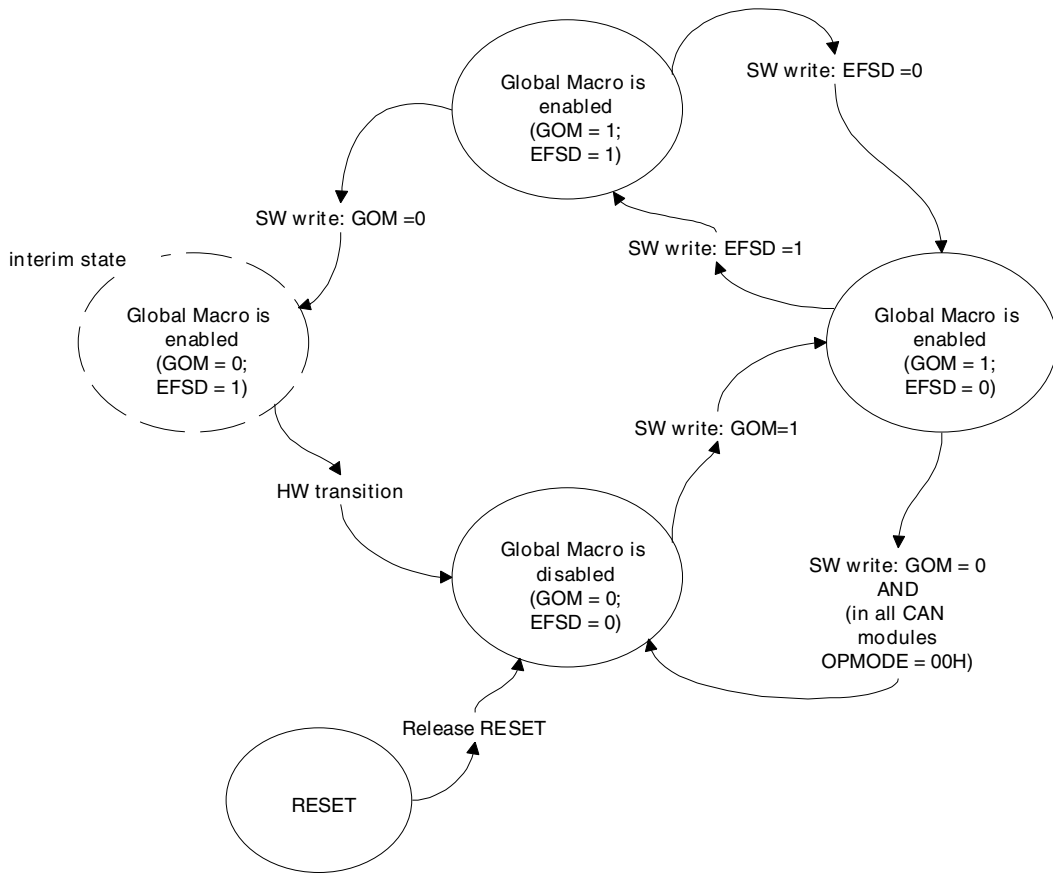
CHAPTER 3 MACRO INITIALISATION AND CONTROL

Macro initialisation is defined as the software processes, which are necessary to use the DAFCAN macro after a RESET (i.e. system reset generated with the RESET pin of the microcomputer system).

Global DAFCAN macro switch-off by software is performed by clearing the GOM bit (0).

After those 2 transitions the global macro is disabled. The CPU can check the status of the global macro by reading the GOM bit in the CnGMCTRL register.

Figure 3-1: State Transitions of Global Macro Switching



3.1 Global Macro Initialisation and Control

3.1.1 Global Macro Disabled (GOM=0)

The global macro is disabled after a RESET or a global DAFCAN macro switch-off by software. The CPU can check the status of the macro by reading the GOM bit in the CnGMCTRL register.

Characteristic of global macro disabled (GOM=0):

- CAN I/F channel output signal CANnTX is fixed to recessive level (logical 1)
- Read and write access to the global macro registers CnGMCTRL, CnGMCS, CnGMABT, CnGMABTD is possible.
- No read access and no write access to the message buffers.
- CAN modules are disabled.
- No write access to the CAN module register (CnMASK1L, CnMASK1H, CnMASK2L, CnMASK2H, CnMASK3L, CnMASK3H, CnMASK4L, CnMASK4H, CnCTRL, CnLEC, CnINFO, CnERC, CnIE, CnBRP, CnBTR, CnLIPT, CnRGPT, CnLOPT, CnTGPT, CnTS, CnCTRL_R, CnLEC_R, CnERC_R, CnIE_R, CnINTS_R, CnBRP_R, CnBTR_R, CnLIPT_R, CnBSEL_R). Accidental write accesses to those registers are ignored by the hardware.

3.1.2 Initialisations before Enabling the Global Macro

The application software has to determine all settings for the macro clock f_{CANMOD} in the CnGMCS register. The clock settings for the macro clock f_{CANMOD} must not be changed after the global macro is enabled.

3.1.3 Global Macro Enabled

The global macro is switched on by setting GOM bit (1) in CnGMCTRL register. Characteristics of global macro enabled (GOM=1):

- CAN I/F channels are enabled
- Both CAN modules are in INIT mode
- The CAN module register CnCTRL, CnLEC, CnINFO, CnERC, CnIE, CnINTS, CnBRP, CnBTR, CnTS, CnCTRL_R, CnLEC_R, CnERC_R, CnIE_R, CnINTS_R, CnBRP_R, CnBTR_R, CnBSEL_R contain their initial values upon switching on the global macro. The CAN module register CnLIPT, CnRGPT, CnLOPT, CnTGPT, CnLIPT_R, CnLOPT_R contain undefined values.
- Write access to the global macro register CnGMCS is ignored.
- Read and write access to all message buffers is possible.
- Read and write access to the CAN module registers (CnMASK1L, CnMASK1H, CnMASK2L, CnMASK2H, CnMASK3L, CnMASK3H, CnMASK4L, CnMASK4H, CnCTRL, CnLEC, CnERC, CnIE, CnINTS, CnBRP, CnBTR, CnTS, CnCTRL_R, CnLEC_R, CnIE_R, CnINTS_R, CnBRP_R, CnBTR_R, CnBSEL_R) is possible. Read access to the CAN module registers CnLIPT, CnRGPT, CnLOPT, CnTGPT, CnLIPT_R and is possible.
- The CAN module register CnTIDR0L/H, CnTIDR1L/H, CnTIDR2L/H, CnTIDR3L/H, CnTIDR4L/H, CnTIDR5L/H, CnTIDR6L/H, CnTIDR7L/H, CnTIDML/H contain their initial values upon switching on the global macro.

3.1.4 Switch-Off the Global Macro

The global macro has not be switched off while one or more CAN modules are processing a message frame. A sudden interruption of a message frame transmission or reception will cause an error in other nodes connected to the same CAN bus.

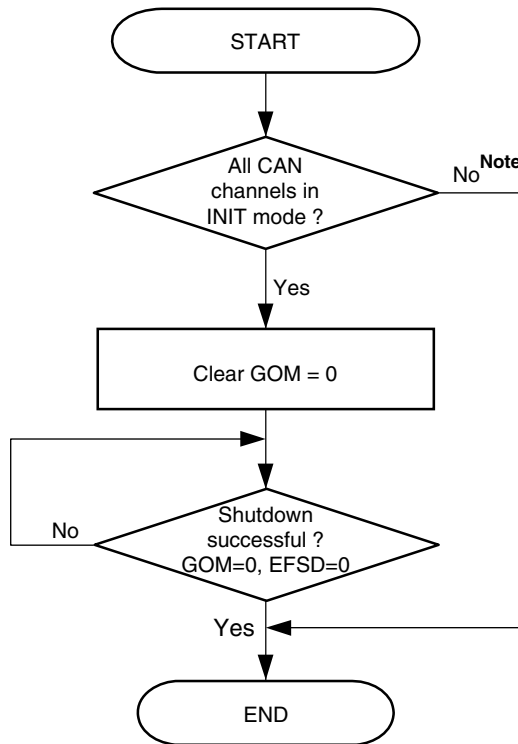
CHAPTER 3 MACRO INITIALISATION AND CONTROL

Before switching off the global macro the application software has to switch all CAN modules in INIT mode. Clearing the GOM bit in CnGMCTRL register will put a switch-off request to the global macro. The switch-off request is discarded in case one or more of the CAN modules are not properly set to INIT mode.

Some applications may require an immediate switch-off of the entire macro in an emergency, regardless of generating disturbance to the other nodes connected to the CAN bus. For such applications the EFSD bit in CnGMCTRL register can be set (1) to allow an immediate global macro switch-off without switching the CAN modules to a defined state (i.e. INIT mode) beforehand.

When using the 'forced shut down' function by setting EFSD bit (1) the subsequent access to the macro has to be the instruction to clear GOM bit (0). In case setting EFSD bit (1) is not directly followed by clearing GOM bit (0), the EFSD bit is cleared (0) automatically (i.e. the 'forced shut down' function is disabled). Also a read access to the CnGMCTRL register between setting of EFSD bit (1) and clearing GOM bit (0) will lead to an automatic clear (0) of EFSD bit.

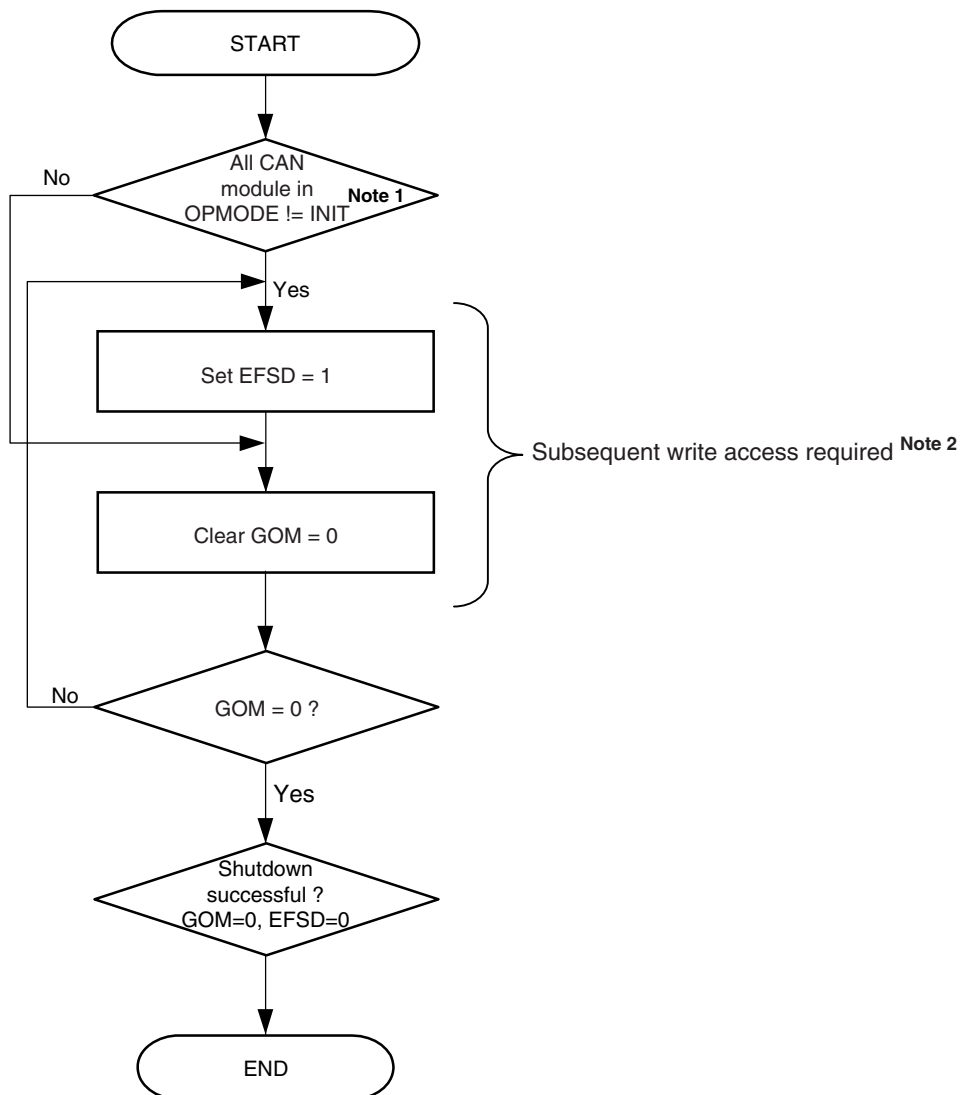
Figure 3-2: Shutdown Process (Normal Shutdown)



Note: Consider to use forced shutdown procedure.

CHAPTER 3 MACRO INITIALISATION AND CONTROL

Figure 3-3: Shutdown Process (Forced Shutdown)



Notes: 1. OPMODE:

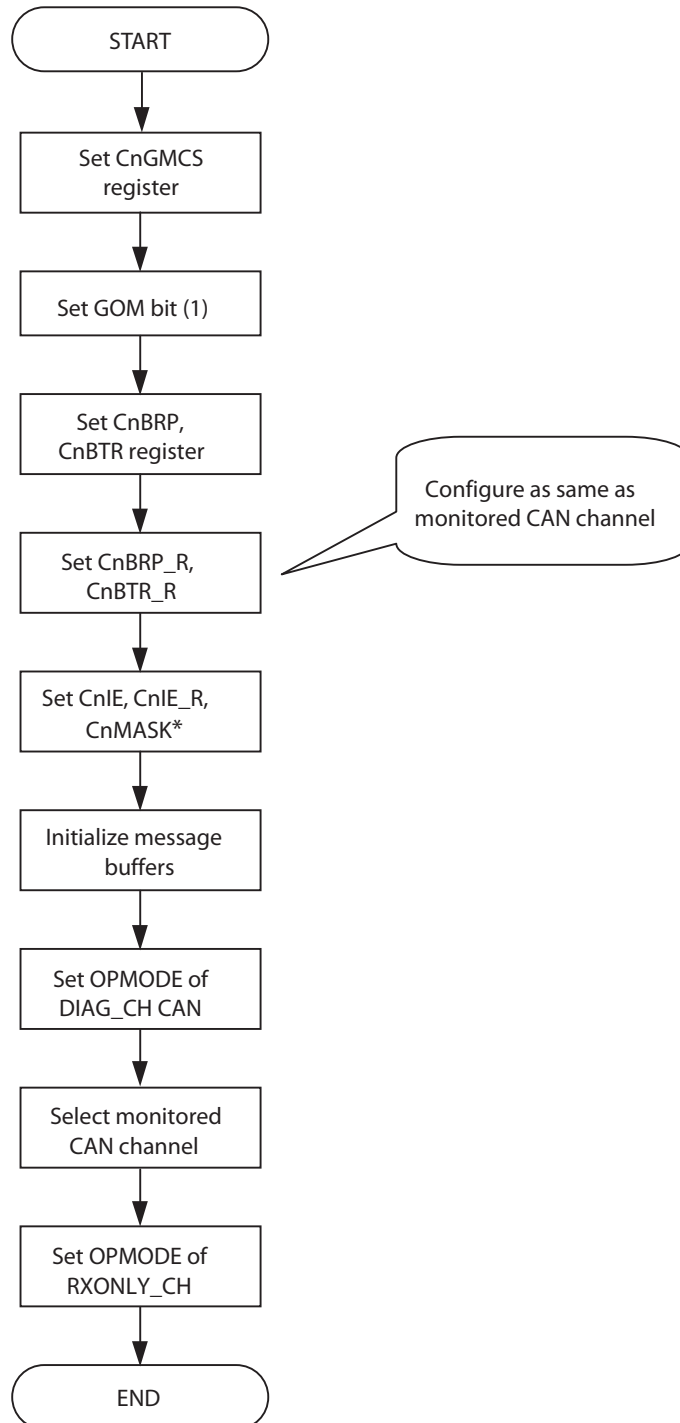
- Normal operation mode
- Normal operation mode with ABT
- Receive only mode
- Single shot mode
- Self test mode

- 2.** Ensure that no CPU read or write access to any register happen between EFSD=1 and GOM=0.
To guarantee that also all interrupts have to be disabled.

3.1.5 Flowchart of the Initialisation

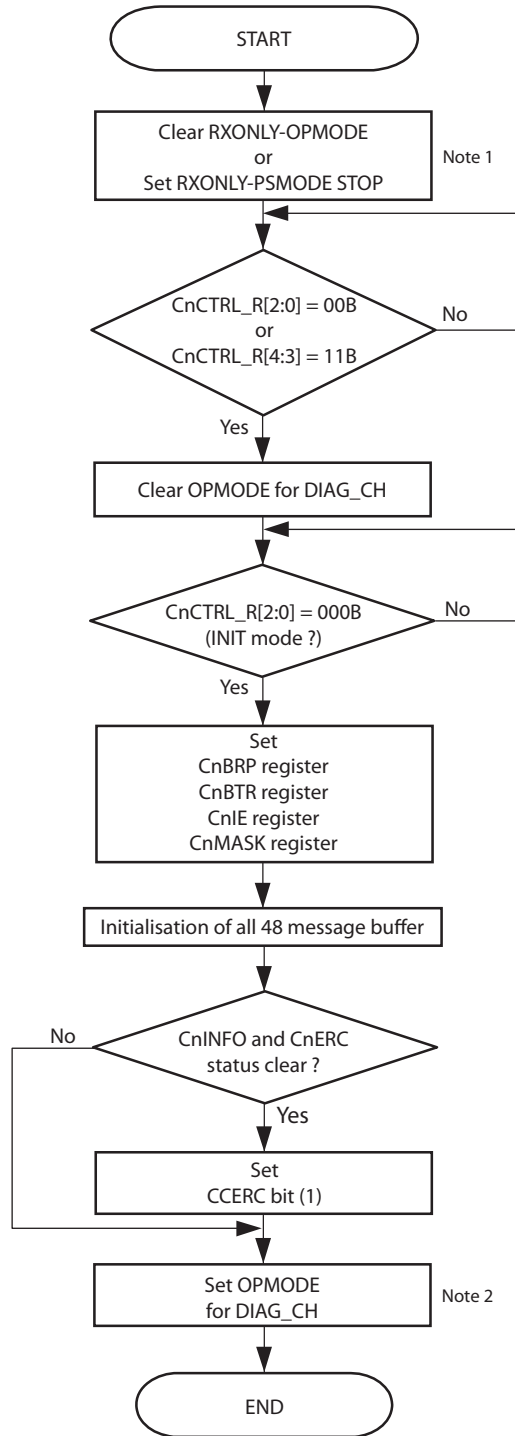
The following flowcharts describe the basic initialisation and in addition the re-initialisation of the CAN module. Note that the configuration for the baud rate of RXONLY_CH should have the same configuration as the monitored CAN channel.

Figure 3-4: DAFCAN Macro Initialisation



CHAPTER 3 MACRO INITIALISATION AND CONTROL

Figure 3-5: Re-initialisation of the DIAG_CH CAN Module using 48 Buffers



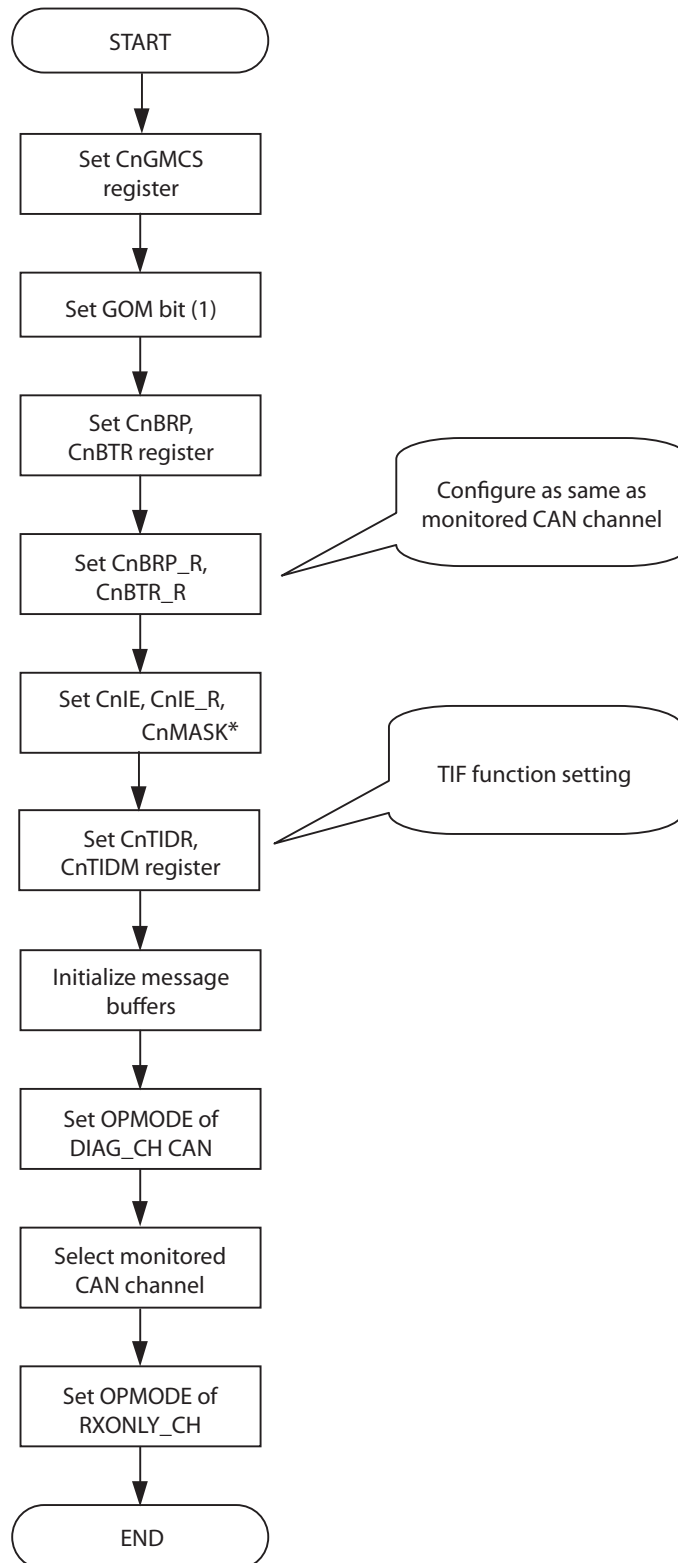
Notes: 1. PSMODE for RXONLY_CH - STOP mode

2. OPMODE:

- Normal operation mode
- Normal operation mode with ABT
- Receive only mode
- Single shot mode
- Self test mode

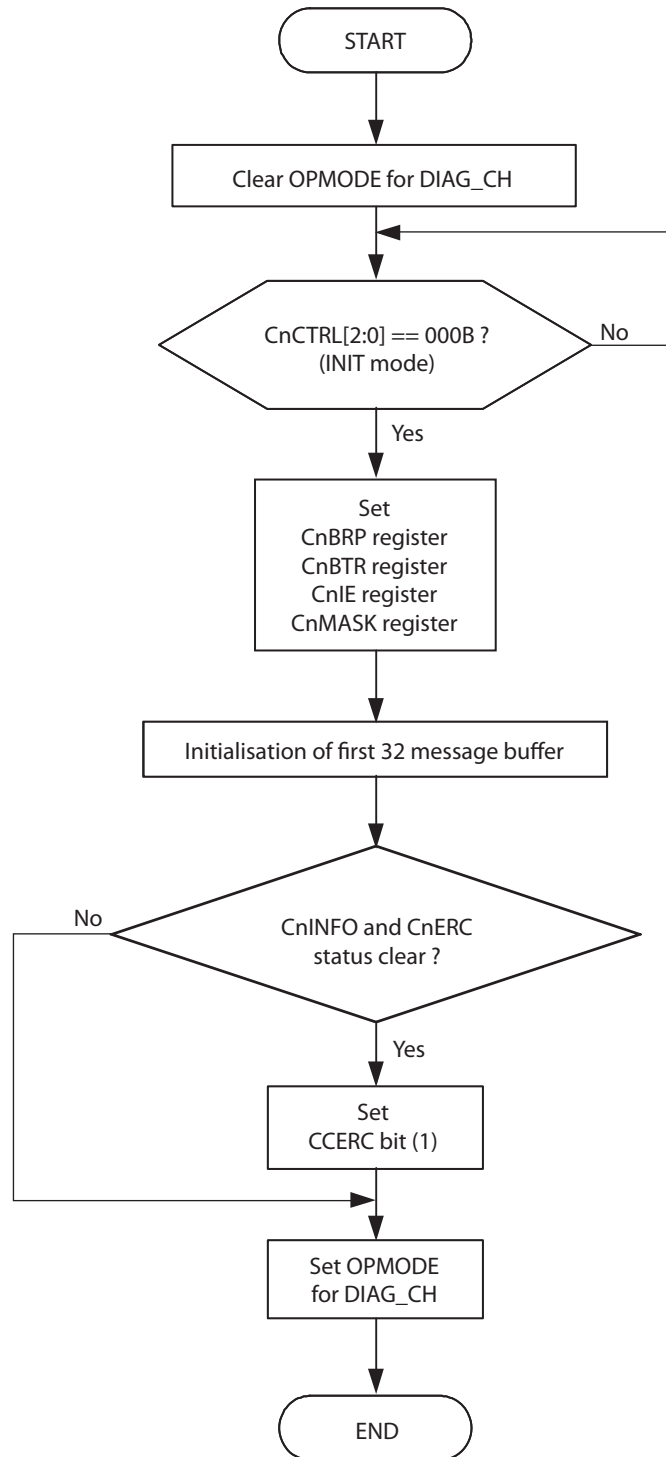
CHAPTER 3 MACRO INITIALISATION AND CONTROL

Figure 3-6: Re-initialisation of the DIAG_CH CAN Module using TX ID Filter Function



CHAPTER 3 MACRO INITIALISATION AND CONTROL

Figure 3-7: Re-initialisation of the DIAG_CH CAN Module using first 32 Buffers only



Remark: OPMODE:
-Normal operation mode
-Normal operation mode with ABT
-Receive only mode
-Single shot mode
-Self test mode

4.1 Message Buffer Initialisation

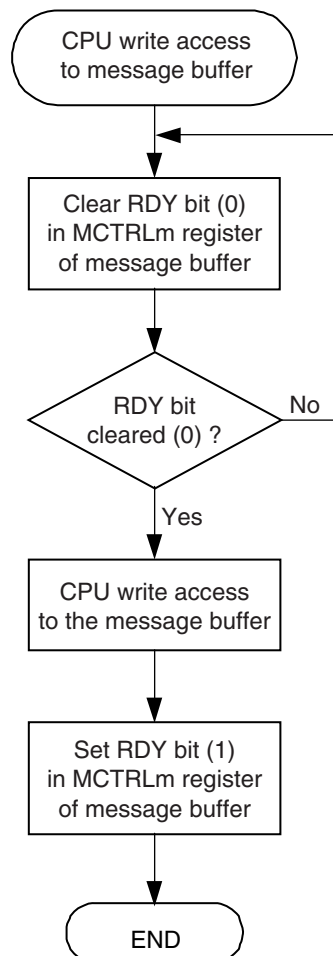
After a RESET or a global DAFCAN macro switch-off by software, the message buffers contain undefined values. A minimum initialisation for all message buffers, even for those not used in the application, is necessary before switching one or more CAN modules from INIT mode to one of the operational modes (refer to Figure 5-1, “Transitions for Operational Modes of the DIAG_CH CAN module,” on page 38 for details of the operational modes). The minimum initialisation has to include the proper configuration of the following bits and bit-strings in the message buffer registers MCTRLm and MCONFm to avoid unexpected behaviour of the macro in the operational modes.

Table 4-1: Minimum Configuration of Message Buffer even when unused in the Application

Bit	Register	Minimum initialisation value
RDY	CnMCTRLm	0B
TRQ	CnMCTRLm	0B
DN	CnMCTRLm	0B
MA0 to MA2	CnMCONFm	0x000B

The RDY bit in the MCTRLm register of each message buffer must be used as a semaphore to avoid access conflicts when writing to a message buffer.

Figure 4-1: CPU Write Access to a Message Buffer



CHAPTER 4 MESSAGE BUFFER INITIALISATION AND CONFIGURATION

While RDY bit is cleared (0) the message buffer is not accessed by the assigned CAN I/F channel (CAN module and/or TTCAN module) and/or bridge module (RXONLY_CH). Thus no message frame can be received in the message buffer or transmitted from the message buffer. Whenever the CPU wants to write into the message buffer, the RDY bit must be cleared (0).

Once the CPU has finished the write access to the message buffer it must set the RDY bit (1) to allow CAN protocol processing to the particular message buffer by the assigned CAN I/F channel.

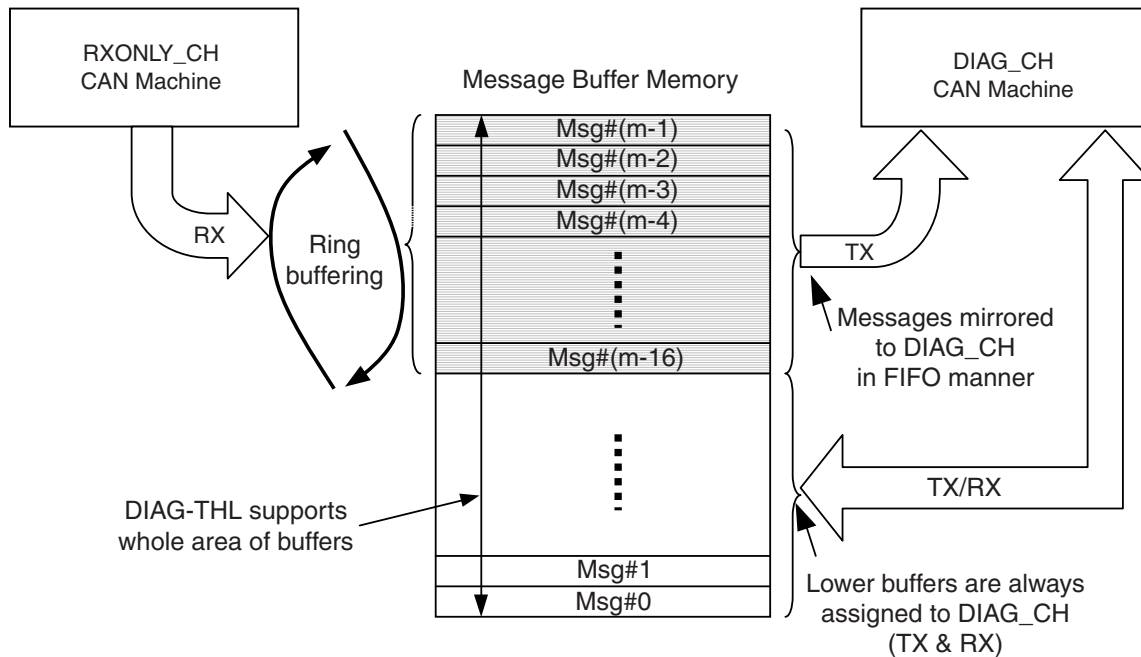
CPU write accesses to a message buffer with RDY bit set (1) are ignored.

For the RXONLY_CH the semaphore handling via RDY bit is identical to the rule mentioned above. However, the application is strongly advised to set all RDY bits (1) for the upper 16 message buffers in order to provide the maximum depth for the ring buffer in mirror mode. As well it is absolutely mandatory not to write to the RDY bit (clear or set the RDY bit) to any of the upper 16 message buffers while the RXONLY_CH is operated in mirror mode or mirror mode w/ TIF. This will terminate the mirror mode or mirror mode w/ TIF or cause an erroneous sequence of monitored messages on the diagnostic CAN bus.

4.2 Message Buffer to CAN I/F Channel Assignment

The area of upper 16 message buffers can be assigned to the DIAG_CH or RXONLY_CH while the rest of the message buffers is statically assigned to the DIAG_CH as shown in the figures below.

Figure 4-2: Operation in Mirror or Receive-only Mode of RXONLY_CH



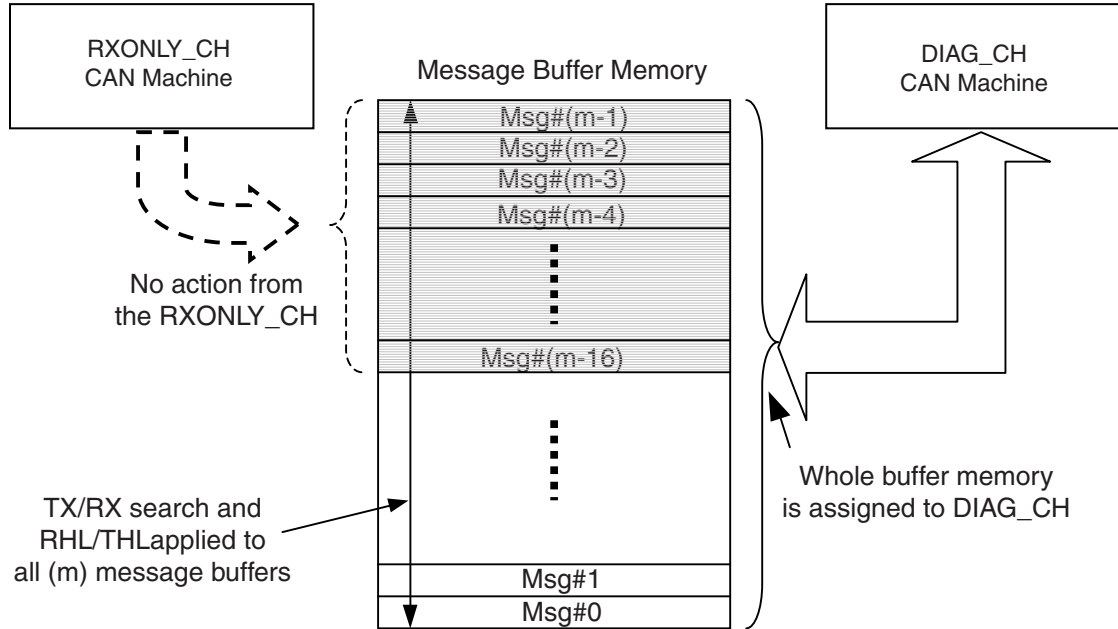
The assignment of the upper 16 message buffers depends on the OPMODE and PSMODE of RXONLY-CH CAN machine.

Basically, these buffers are assigned to the RXONLY_CH while the RXONLY_CH is operational mode (i.e. mirror mode / rec-only mode). And additionally, it must be assigned to the RXONLY_CH while the RXONLY_CH is in SLEEP mode because the wake-up event on the RXONLY_CH is unpredictable. In STOP mode these buffers are assigned to the DIAG_CH in order to provide a mode of operation where the DIAG_CH uses all 48 buffers while the RXONLY_CH uses minimum power. The user needs to con-

CHAPTER 4 MESSAGE BUFFER INITIALISATION AND CONFIGURATION

figure the upper16 message buffers before changing assignment (i.e. leaving initialisation or STOP mode of the RXONLY_CH).

Figure 4-3: Operation during INIT or STOP of RXONLY_CH



The assignment of message buffers versus operational and power-save modes is shown in the table below.

Table 4-2: Message Buffer Assignment versus PS/OPMODE

OPMODE/ PSMODE	MIRROR mode	MIRROR mode w/ TIF	Receive-only mode	INIT mode
NO PWR-SAVE	RXONLY	RXONLY	RXONLY	DIAG
SLEEP	RXONLY	RXONLY	RXONLY	
STOP	DIAG	DIAG	DIAG	

When the assignment of the upper 16 message buffers switches, the application should configure all buffers beforehand as follows.

4.3 Configuration before switching to Mirror Mode

For mirror mode operation all upper 16 message buffers shall be used as a ring buffer working in FIFO manner. Thus, the application has to prepare these buffers as TX-buffers beforehand as follows. Note that OWS setting is invalid (not necessary) because the non-overwriting of message buffers with DN = 1 is built in feature of operational modes of the RXONLY_CH.

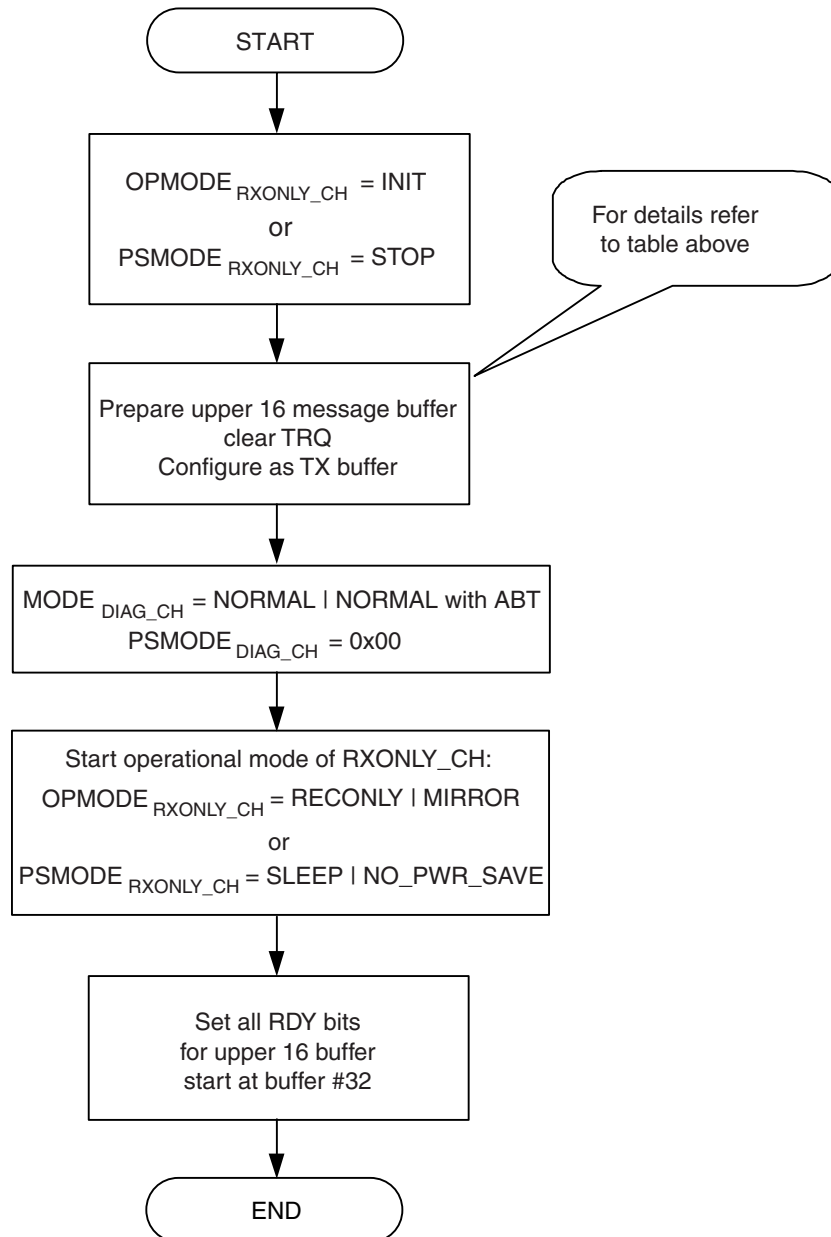
Bit name	Configuration
MOW	Clear
IE	Depends on application
DN	Clear
TRQ	Clear
MA[2:0]	001b
MT[2:0]	000b
RTR	Don't care
OWS	Don't care
IDE	Don't care
ID	Don't care
DLC	Don't care
MDATA	Don't care

The flow of operations in order to change the assignment of message buffers to a CAN-channel is shown in the figure below. At first the application needs to apply RDY = 0 for the upper 16 message buffer or set the DIAG-CH into INIT mode before the new configuration can be written to the buffers.

Once the new configuration is set, the DIAG_CH followed by RXONLY_CH is put into its designated operational and power save mode. Finally the upper 16 message buffer are re-enabled by setting the RDY bit (1).

CHAPTER 4 MESSAGE BUFFER INITIALISATION AND CONFIGURATION

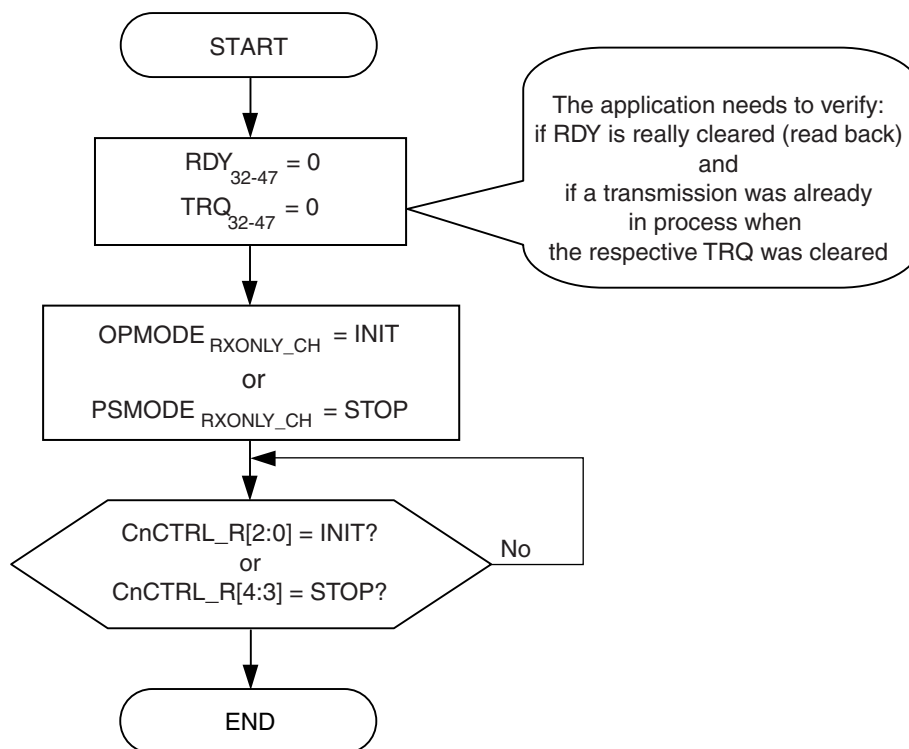
Figure 4-4: Set up of Mirror Mode, Mirror Mode w/ TIF or Receive-Only Mode for RXONLY_CH



4.4 Configuration before switching to DIAG Side

All upper 16 message buffers can be assigned to the DIAG_CH in order to use them as normal TX/RX message buffers by the application. It is not necessary to apply a particular configuration for these buffers before the assignment is linked again to the DIAG_CH. However, it is recommended that the application clears RDY and TRQ bits before assigning the buffers to the DIAG_CH or puts the DIAG_CH into OPMODE = INIT or PSMODE = STOP before changing the assignment. Then, at least clear the RDY bits for the upper 16 message buffers before putting the DIAG_CH to normal mode. This will ensure proper message handling during the transition.

Figure 4-5: Cancellation of Mirror Mode or Receive-Only Mode for RXONLY_CH



When clearing the TRQ in the upper message buffer a transmission from one of these buffers may still be in process. In order to ensure correct message handling (i.e. transmission history list), it is recommended to wait for the first transmission completion interrupt of the DIAG_CH before resuming the transition procedure.

After a RESET the registers CnGMCTRL, CnGMCS, CnGMABT, CnMASK1L, CnMASK1H, CnMASK2L, CnMASK2H, CnMASK3L, CnMASK3H, CnMASK4L, CnMASK4H, CnCTRL, CnLEC, CnINFO, CnERC, CnIE, CnINTS, CnBRP, CnBTR, CnTS, CnCTRL_R, CnLEC_R, CnERC_R, CnIE_R, CnINTS_R, CnBRP_R, CnBTR_R, CnLIPT_R, CnBSEL_R of the CAN module in the CAN I/F channels contain their initial value.

Before switching the CAN module into an operational mode the registers (CnMASK1L, CnMASK1H, CnMASK2L, CnMASK2H, CnMASK3L, CnMASK3H, CnMASK4L, CnMASK4H, CnCTRL, CnIE, CnBRP, CnBTR, CnTS, CnCTRL_R, CnIE_R, CnBRP_R, CnBTR_R) must be initialised according the requirements of the application. Further, the CnLEC and CnLEC_R registers must be set to its initial value (00H) and the VALID bit in the CnCTRL and CnCTRL_R registers must be cleared (0) by the user.

5.1 CAN Bit Time Programming

The DIAG_CH provides the registers CnBRP and CnBTR and the RXONLY_CH provides CnBRP_R and CnBTR_R. The write access to these register has to be allowed only when the particular CAN module is in INIT mode. During an operational mode of the CAN module only reading of these registers has to be granted while the writing is blocked by hardware.

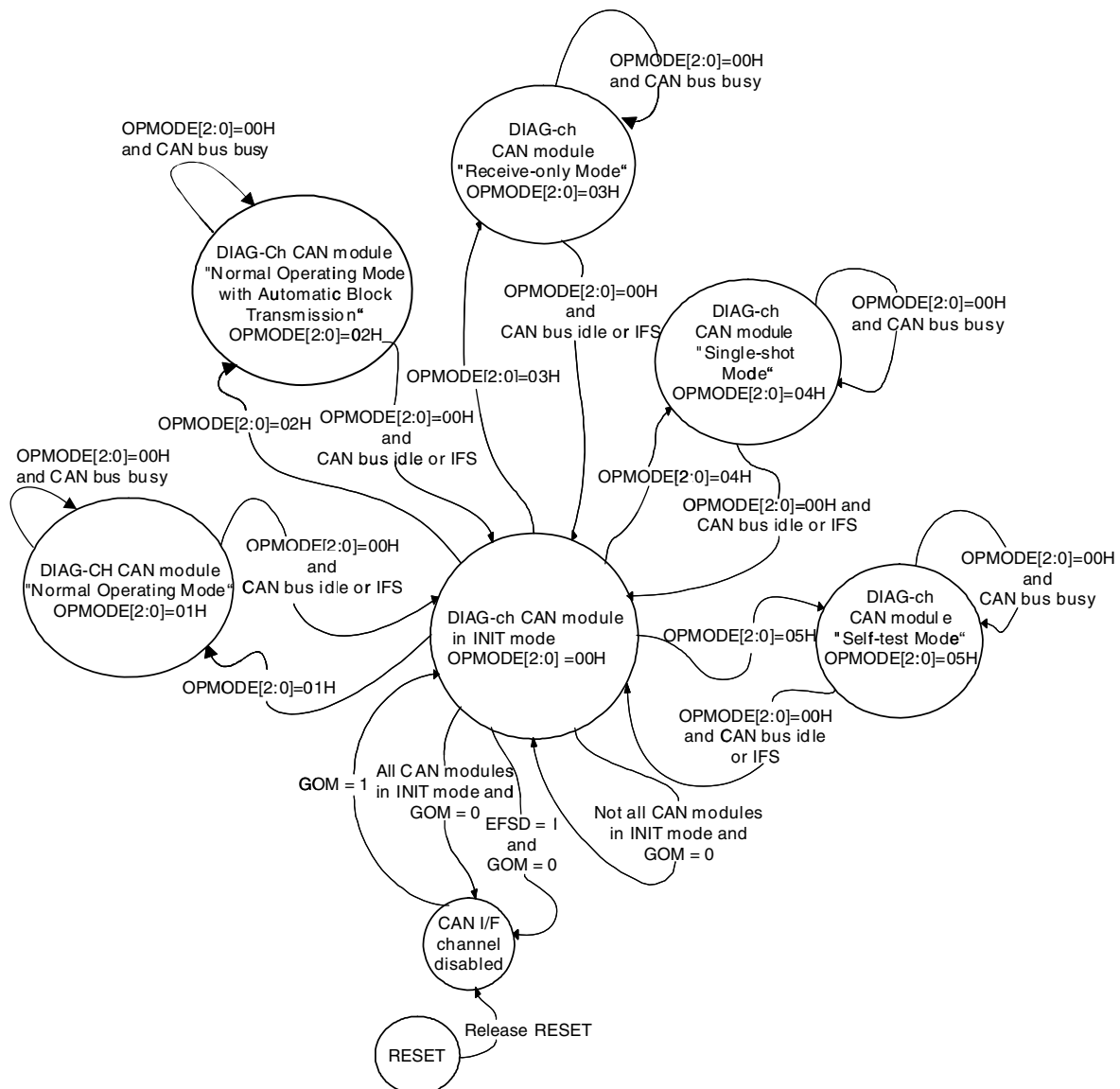
After proper setting of the CAN bit time, the CAN module can be released from INIT mode into one of its operational modes.

5.1.1 Transitions for Operational Modes of DIAG_CH

The DIAG_CH CAN machine can be switched the following operational modes:

- “Normal Operating Mode”
- “Normal Operating Mode with Automatic Block Transmission”
- “Receive-only Mode”
- “Single-shot Mode”
- “Self-test Mode”

Figure 5-1: Transitions for Operational Modes of the DIAG_CH CAN module



The transitions from INIT mode to the operational modes is controlled by the bit-string OPMODE[2:0] in the CnCTRL register.

CHAPTER 5 DAFCAN MACRO INITIALISATION AND CONTROL

Changing from one operational mode into another operational mode requires to transit into INIT mode intermittently. The CAN module refuses CPU attempts to change from one operational mode into another operational mode directly.

Transition requests from the operational modes to the INIT mode are not directly accepted by the CAN module when the CAN bus is not idle (i.e. frame reception or transmission is ongoing), but it has to be kept until when the CAN module detects the first bit of intermission. As soon the above mentioned condition is detected, the transition from the operational mode to the INIT mode is executed and the bit-string OPMODE[2:0] value changes to 000B. The CPU has to confirm the proper transition into INIT mode by reading the OPMODE[2:0] bit-string until OPMODE[2:0] = 000B.

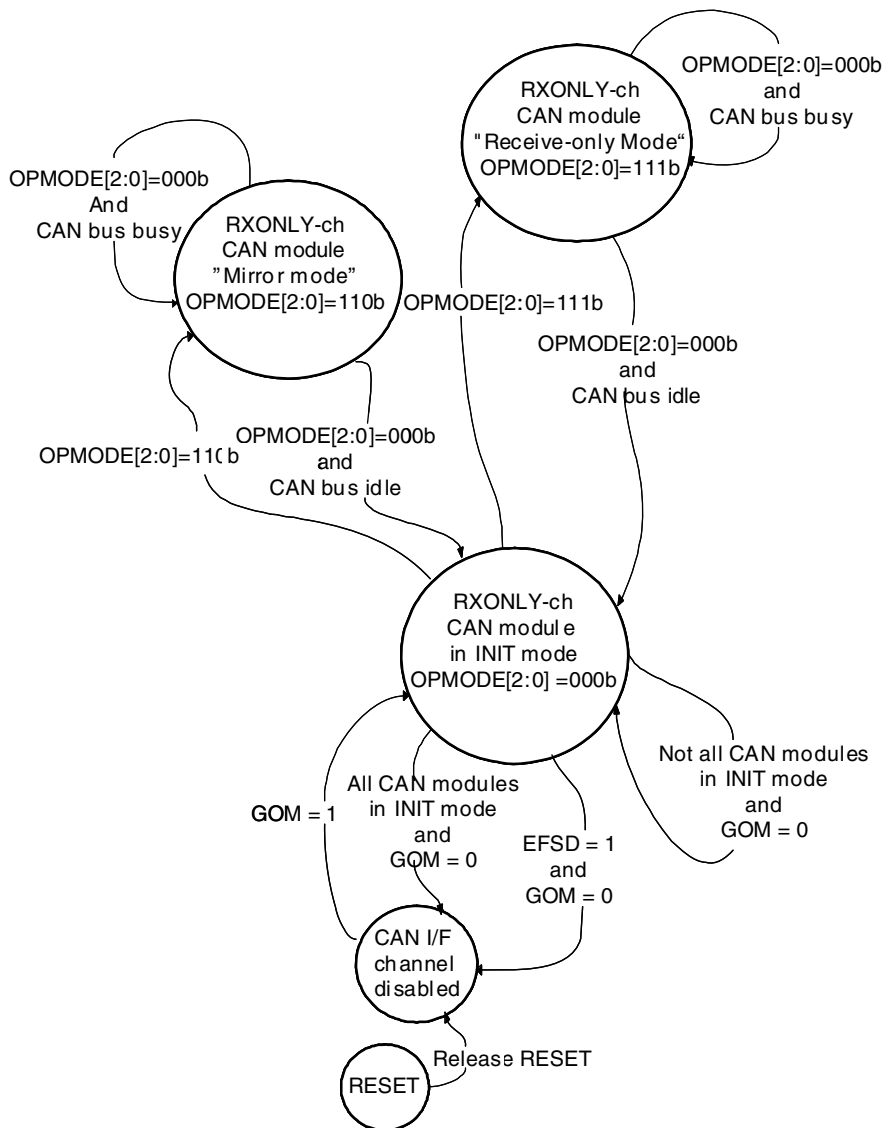
When a successful receive interrupt is detected for a specific CAN module that already entered INIT mode, this interrupt was generated by a reception process that coincided with the request of INIT mode by the CPU. The received message that caused this interrupt was still stored before INIT mode was becoming valid.

5.1.2 Transition for Operational Modes of RXONLY_CH

The RXONLY-H CAN machine can be switched the following operational modes:

- “Mirror Mode”
- “Mirror Mode with Transfer ID Filter function (w/ TIF)”
- “Receive-only Mode”

Figure 5-2: Transitions for operational modes of the RXONLY_CH CAN module



The transitions from INIT mode to the operational modes is controlled by the bit-string OPMODE[2:0] in the CnCTRL_R register.

The rules for changing from one to another operational mode are the same as for the DIAG_CH.

CHAPTER 6 MACRO INTERRUPTS

The DAFCAN macro provides 12 different interrupt source events. The occurrence of those interrupt source events is stored in interrupt status registers. Four separate interrupt request signals are generated from the 12 source events. The signals are routed to the interrupt controller in the microcomputer system. In case the interrupt controller in the microcomputer system does not provide a sufficient number of interrupt request signal inputs, the 4 interrupt request signals of a CAN module can be grouped (i.e. logical OR). With help of the interrupt status register the user can determine the actual interrupt source event for a particular interrupt. After determination of the interrupt source event the user must clear the corresponding interrupt status bit.

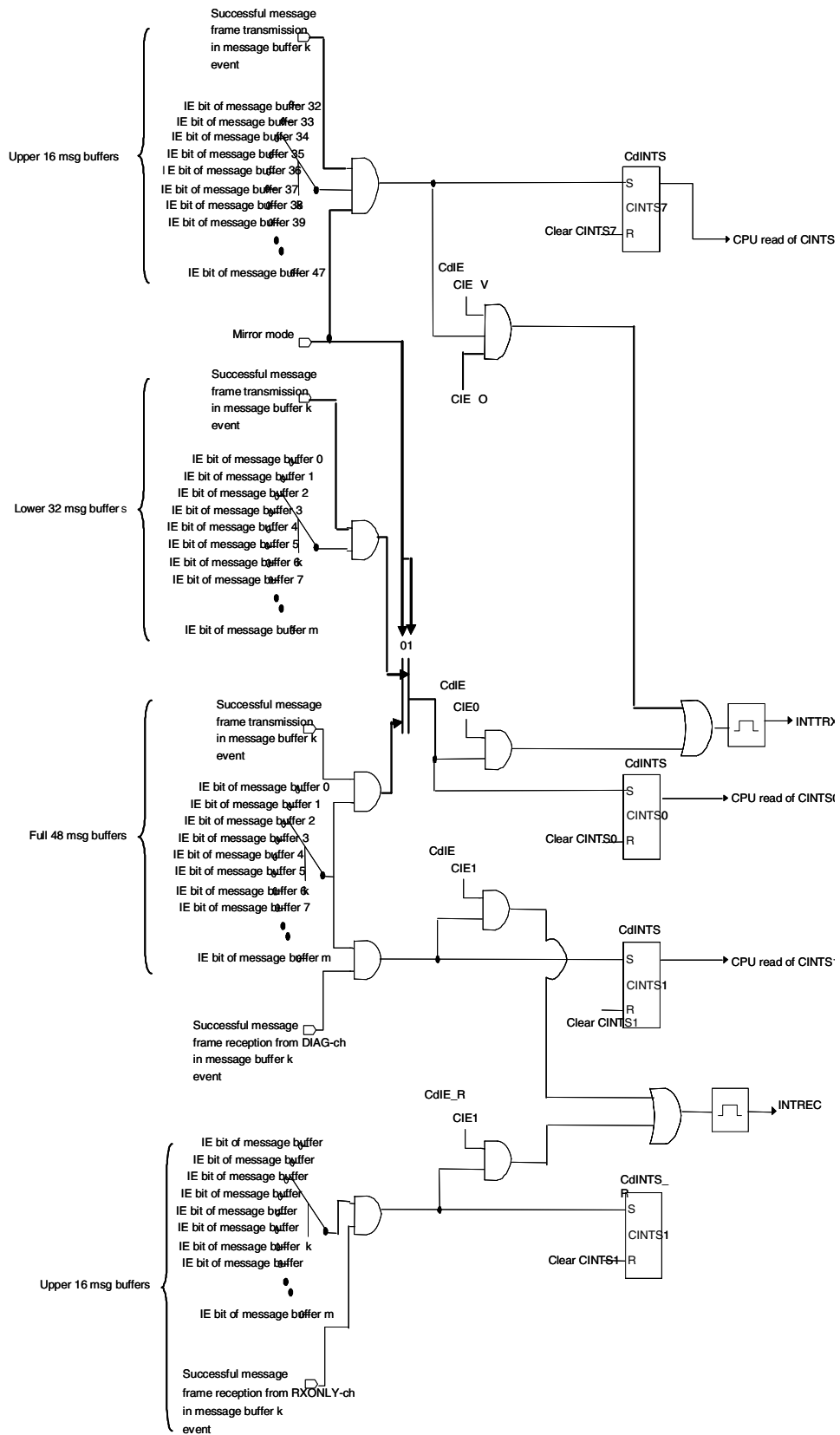
Table 6-1: List of all Macro Interrupt Sources

#	Interrupt Status Bit		Interrupt Enable Bit		Interrupt Request Signal	Interrupt Source Description
	Name	Register	Name	Register		
1	CINTS0	CnINTS	CIE0 ^{Note}	CnIE	INTRRX	CAN module interrupt status bit for interrupt event 'Message frame successfully transmitted from message buffer m' by the DIAG_CH. The interrupt is also provided for mirrored messages.
2	CINTS1	CnINTS	CIE1 ^{Note}	CnIE	INTREC	CAN module interrupt status bit for interrupt event 'Valid message frame reception in message buffer m' from the DIAG_CH.
3	CINTS1	CnINTS_R	CIE1 ^{Note}	CnIE_R		CAN module interrupt status bit for interrupt event 'Valid message frame reception in one of the upper 16 message buffers from the RXONLY_CH.
4	CINTS2	CnINTS	CIE2	CnIE	INTERR	CAN module error state interrupt status of the DIAG_CH
5	CINTS2	CnINTS_R	CIE2	CnIE_R		CAN module error state interrupt status of the RXONLY_CH
6	CINTS3	CnINTS	CIE3	CnIE		CAN module protocol error interrupt status of the DIAG_CH
7	CINTS3	CnINTS_R	CIE3	CnIE_R		CAN module protocol error interrupt status of the RXONLY_CH
8	CINTS4	CnINTS	CIE4	CnIE		CAN module arbitration loss interrupt status of the DIAG_CH
9	CINTS5	CnINTS	CIE5	CnIE	INTWUP	DIAG_CH CAN machine wake-up interrupt status bit from SLEEP mode by CAN bus
10	CINTS5	CnINTS_R	CIE5	CnIE_R		RXONLY_CH CAN machine wake-up interrupt status bit from SLEEP mode by CAN bus
11	CINTS6	CnINTS_R	CIE6	CnIE_R	INTERR	The buffer overflow interrupt status bit, which is set 1 when the RXONLY_CH CAN machine fails to store a message because the upper 16 message buffer are all occupied (i.e. $DN < CnLIPT_R + 1 > = 1$).
12	CINTS7	CnINTS	CIE7	CnIE	INTRRX	Signals a completed transmission for upper 16 message buffer during Mirror Mode. The status bit and CIE7 are only meaningful when RXONLY_CH is operated in Mirror mode. CIE7 has no function when CIE0 in CnIE is cleared.

Note: The IE bit (message buffer interrupt enable bit) in the MCTRL register must set (1) for the message buffers, which shall participate in the interrupt generation process.

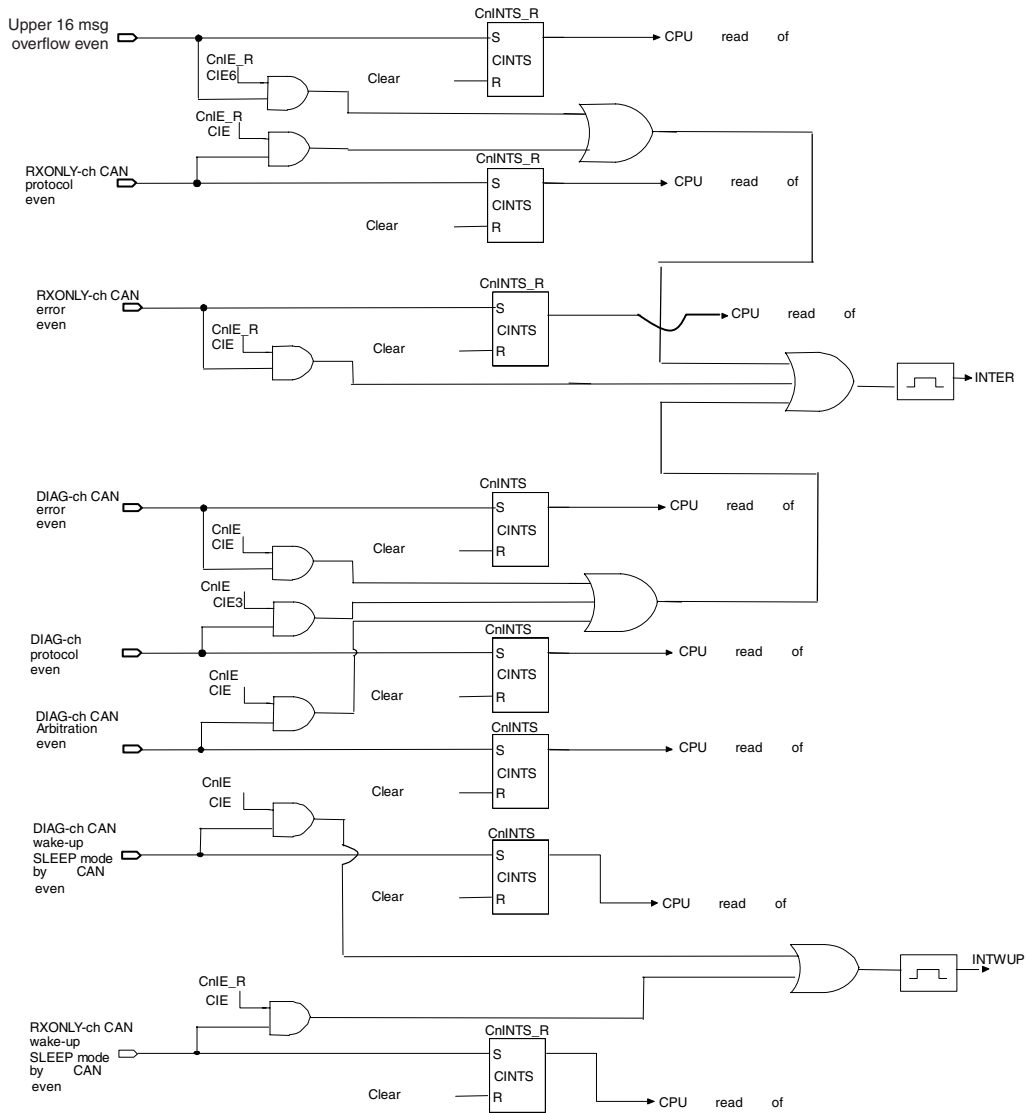
CHAPTER 6 MACRO INTERRUPTS

Figure 6-1: Schematic of the DAFCAN Macro Interrupt Generation (1/2)



CHAPTER 6 MACRO INTERRUPTS

Figure 6-1: Schematic of the DAFCAN Macro Interrupt Generation (2/2)



7.1 Principal Reception Process

In general all operations linked to acceptance filtering are only available on the DIAG_CH. For the RXONLY_CH only simple receive operations apply. These are mentioned more detailed in chapter Chapter 8 "Operational Modes of RXONLY_CH" on page 45. The features of DIAG_CH regarding reception are identical with those of the AFCAN macro, so these are not mentioned within this addendum.

For the RXONLY_CH the acceptance filtering is obsolete. Any valid message will be stored in the upper 16 message buffer of the DAFCAN macro when Mirror Mode, mirror mode w/ TIF or Receive-Only Mode is selected. In both modes the RXONLY_CH will store the message in the next higher buffer number where it finds a buffer with its DN bit cleared. At buffer #47 it wraps around to buffer #32.

In Receive-Only Mode the application needs to read the messages and clear the DN bits in time to avoid the generation of the interrupt CINTS6 in CnINTS_R.

In Mirror Mode the reception process will also set the DN bit but no receive interrupt is generated. Rather another state machine is triggered to evaluate if the TRQ of this message buffer can be set, and thus launch the transmission of the message on the DIAG_CH. The state machine handling the setting of TRQ's for the upper 16 message buffers is activated every time the DIAG_CH or the host CPU clears a TRQ bit. Even TRQ bits for the lower 32 message buffers need to be considered. When the state machine detects that the TRQ for the message buffer it flagged for transmission the last time was cleared, the TRQ for the next buffer is set unless the history list implies no pending entry.

As the DN bit is used by the RXONLY_CH to detect an empty (already transmitted) buffer, the DN bit is cleared automatically after the message was transmitted successfully on the DIAG_CH.

7.2 Reception History

The RXONLY Channel does not support the Reception History List (RHL), as it is known for the standard AFCAN channels and DIAG_CH. However, the functions of the Last-In-Message Pointer (LIPT) are implemented within the CnLIPT_R register.

Note: The LIPT_R value is maintained throughout the PSMODE = STOP for RXONLY_CH. This register is only cleared when entering OPMODE = INIT.

Caution: The RGPT of the DAFCAN channel is not updated by the reception from the RXONLY_CH.

7.3 Reception of Remote Frames

The reception of remote frames applies to both, the DIAG_CH and the RXONLY_CH. However, for RXONLY_CH message storing follows different rules and acceptance filtering is not available.

A remote frame reception on the RXONLY_CH is handled as a normal data frame reception into the upper 16 message buffers without any acceptance filtering. The message will be stored according the ring buffer method.

7.4 Message Transmission

As the RXONLY_CH does only receive messages, only specific transmit operations by the DIAG_CH that are linked to the mirror mode of the RXONLY_CH are to be explained in chapter 8.2 on page 49. Regarding the transmission functionality of the DIAG_CH, please refer to the standard AFCAN manual.

8.1 Receive-Only Mode

In Receive-only mode of the RXONLY_CH CAN module every data and every remote frame, which is received from the CAN bus in the RXONLY_CH CAN module without an error (i.e. valid reception), is stored in the upper 16 message buffers without any acceptance filtering.

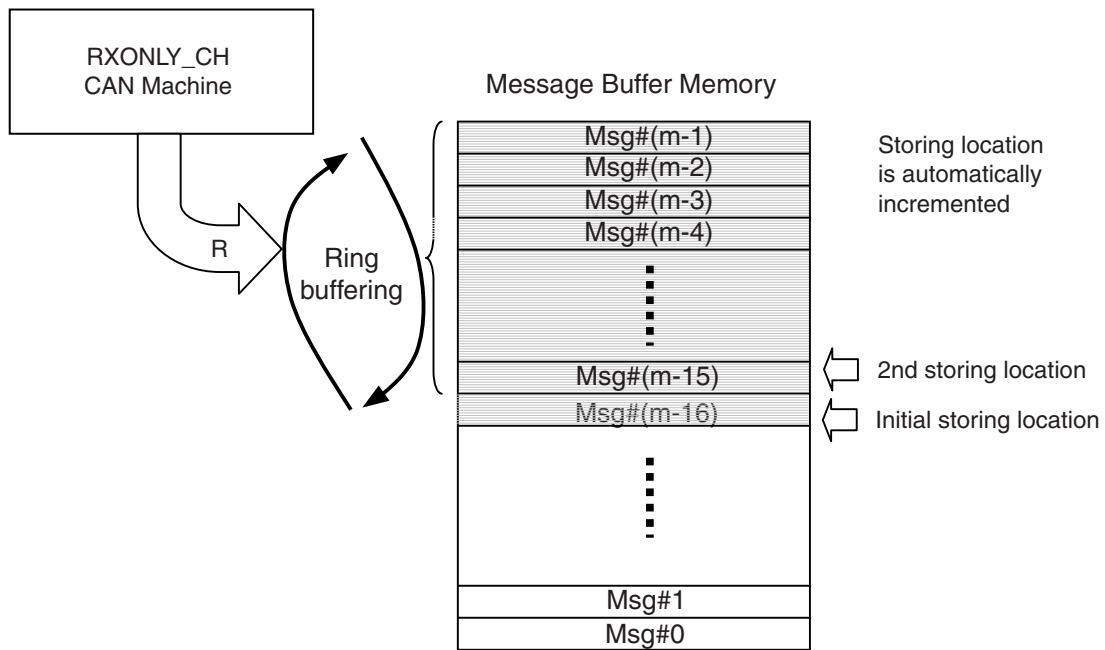
Storing received messages from RXONLY_CH requires the following conditions:

- The message buffer has to be assigned to the CAN I/F channel, which received the data frame (MA[2:0] bit-string in CnMCONFm register has to hold the values 001B).
- The message buffer has to be marked ready for CAN protocol processing (RDY bit set (1) in CnMCTRLm register)
- Additional condition necessary for Mirror Mode operation:
The upper 16 message buffer need to be configured as transmit message buffer (MT[2:0] bit string in CnMCONFm has to hold the value 0x000B).

Caution: In difference to a regular AFCAN channel, any message is accepted by the buffer although it is configured as a transmit message buffer that normally would not except to receive data frames.
All transmit request bits of the upper 16 message buffer have to be cleared (TRQ bit reset (0) in CnMCTRLm register) especially before Mirror Mode shall be invoked. This is necessary because the setting of TRQ for the upper 16 message buffer is handled by the mirror mode machine.

The upper 16 message buffer behave as ring buffer as shown in the figure below.

Figure 8-1: Reception Process of Receive-Only Mode



m = maximum number of message buffers = 48

CHAPTER 8 OPERATIONAL MODES OF RXONLY_CH

The storing pointer is initialised to (m-16) at the time when switching the assignment of upper 16 message buffer.

The reception process of the RXONLY_CH differs from the reception process of the regular AFCAN macro.

- There is no acceptance filtering; any data or remote frame will be received.
- No specific setting is required to receive standard and extended format frames in the upper 16 buffers during Receive-Only and Mirror Mode. IDE bit will be written by the RX-store machine with respect to the received frame type.
- For received remote frames the RTR bit will be set by the RX-Store machine.

The table below compares the differences for each data type of the buffer.

Table 8-1: Differences in Reception Process between regular AFCAN and RXONLY_CH

Register	Bit String	DIAG /regular AFCAN	RXONLY_CH
CnMDATAm0 – CnMDATAm7	-	Written by RX-Store machine for remote frames value is kept	Written by RX-Store machine
CnMDLcM	-	Written by RX-Store machine for remote frames value is kept	Written by RX-Store machine
CnMCONFm	OWS, MT2-MT0, MA2-MA0	Set up by CPU during configuration	Set up by CPU during configuration
	RTR	Set up by CPU when preparing TX message	Written by RX-Store machine according received frame type
CnMIDHm/ CnMIDLm	ID0-ID28	Written by RX-Store machine according rules of acceptance filtering	Written by RX-Store machine as received in message
	IDE	Set up by CPU during configuration	Set up at reception by RX-Store machine
CnMCTRLm	MOW IE	Set up by CPU during configuration	Set up by CPU during configuration
	DN	Set by RX-Store machine and cleared by CPU	Set by RX-Store machine and cleared by CPU in Receive-Only mode. Autonomously operated by RXONLY_CH in Mirror mode
	TRQ	Set by CPU (in ABT mode TRQ is set by ABT-engine autonomously)	Set by RXONLY_CH when operated in Mirror Mode
	RDY	Set/cleared by CPU	Cleared by CPU before RXONLY_CH is put to Receive-Only or Mirror mode. Then set again by CPU in order to enable storing messages.

The transition into Receive-only mode is described in chapter 5.1.2 "Transition for Operational Modes of RXONLY_CH" on page 40.

8.1.1 Processing received Frames in Receive-Only Mode of RXONLY_CH

When the RXONLY_CH is operated in Receive-Only mode, all valid messages are stored in the upper 16 message buffer. At initial start of that operating mode (i.e. when leaving INIT), the first message is stored in buffer 32. When resuming Receive-Only mode leaving SLEEP mode, the storage will resume from the last position (message buffer) reached previously. If the user has set the option to get an interrupt, a receive interrupt is signalled to the host CPU.

The CPU can identify the most recently stored message with the help of the CnLIPT_R register. In case the CPU chooses to randomly poll the upper 16 message buffer for new messages and more than 1 message has its DN flag set, the sequence of their reception can be retrieved by a special algorithm.

In that case the CPU reads the CnLIPT_R register and decrements the message buffer number by one. Then CPU checks if the DN flag of that buffer is set. If not set, the buffer with the oldest message was one position ahead (in reverse direction of the CPU search).

If the DN flag is set the CPU repeats decrementing the buffer number with wrap around from buffer number 32 to 47 until it encounters a DN flag that is not set or until it reaches the buffer number that equals CnLIPT_R + 1. In that case the CPU reads CnLIPT_R again in order to find out if newly received messages have been stored while the CPU was searching for the oldest message.

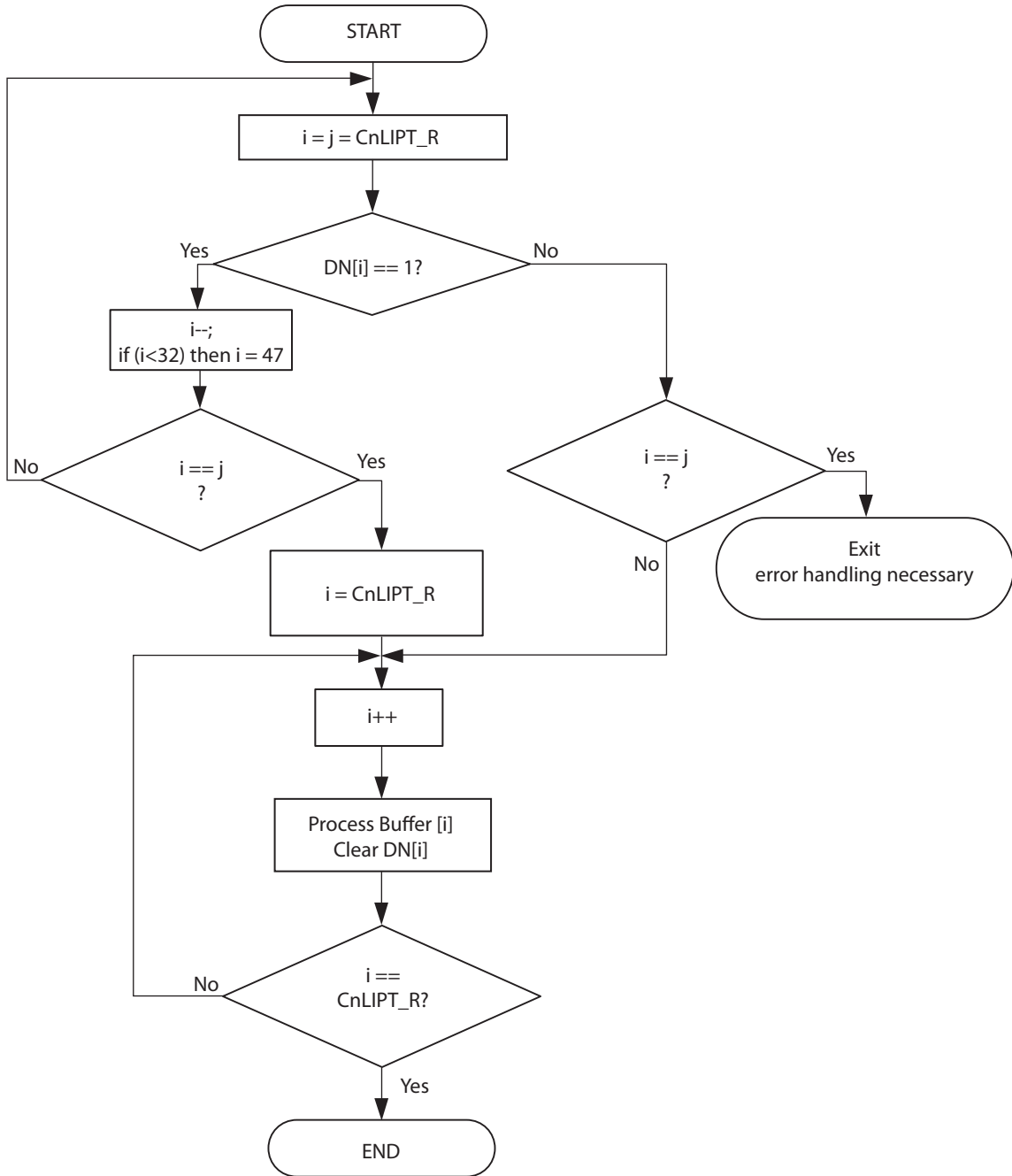
Then, this secondly read value incremented by one points to the oldest message.

The CPU needs to clear the DN flag after processing the received messages in order to avoid an overflow situation, which generates an error interrupt. In case this interrupt occurs, the host CPU can directly start processing messages at buffer CnLIPT_R+1, because the RXONLY_CH does not overwrite any buffer with DN=1. Once the CPU has cleared the DN flag of buffer CnLIPT_R+1, the RXONLY_CH will resume storing received messages. Clearing DN flags at other locations beforehand, will not let the reception process resume because the RXONLY_CH always stores messages in ascending order starting at CnLIPT_R+1.

The flow chart below illustrates the necessary algorithms for the software.

CHAPTER 8 OPERATIONAL MODES OF RXONLY_CH

Figure 8-2: Receive Operation for RXONLY_CH in Receive-Only Mode



The exit that requires error handling is normally not encountered. In that case the application would process a receive interrupt but did not find any newly received message. Typically, the application has failed to clear the interrupt pending bit in a previous interrupt routine.

8.2 Mirror Mode

The DAFCAN macro features a message mirroring function. It automatically copies messages from the RXONLY_CH to the bus served by the DIAG_CH. When this function, the mirror mode, is enabled, any valid data frame and any valid remote frame received by the RXONLY_CH will be stored in the upper 16 message buffers in the same manner as in Receive-Only mode. From that location these messages are automatically transmitted by the DIAG_CH in FIFO manner. No acceptance filtering is applied to the frame receptions.

The message mirroring is activated when the DAFCAN macro is configured as the follows:

- The OPMODE of RXONLY_CH CAN machine is configured as “Mirror mode”.
- The OPMODE of DIAG_CH CAN machine is configured as either “Normal operation mode” or “Normal operation mode with ABT”
- Neither of the CAN machines are in power save mode.

Other settings for the OPMODE for the DIAG_CH i.e., Single-shot mode are not forbidden, but they may lead to unexpected behaviour i.e. in case of transient bus errors on the diagnosis bus the mirrored message will not be repeated and thus be not visible although it was successfully communicated on the bus that was monitored. The application is strongly recommended to use only the configuration as mentioned in the listing above.

The mirror mode engine sets one TRQ at a time sequentially in FIFO manner. So there is only one TRQ bit, which is set (1) in the upper 16 buffers when mirroring is enabled. The application must not set any TRQ bit in this upper 16 buffers when mirroring or Receive-Only mode is enabled. Neither the application shall leave a TRQ bit set (1) when the mirror mode, mirror mode w/ TIF or Receive-Only mode shall be entered next.

Even when the mirroring function is enabled, the TX-search of the DIAG_CH will be applied to the upper 16 message buffers although they are assigned to the RXONLY_CH. The search engine will conduct the TX-search on the lower buffers and the candidate from the upper 16 buffers. Thus, the mirrored messages can suffer a delay before they can be seen on the diagnosis CAN bus depending on the message priority loaded in the lower buffers.

8.2.1 Operations of the Mirror Mode Engine

The following paragraphs describe the operation of the mirror mode engine (MME).

After RESET the LIPT_R points to the buffer number #32 where the first frame from the RXONLY_CH is stored. As soon as the DN flag is set (1), the MME starts to operate using current LIPT_R value. Note that LIPT_R value can be different as #32 when the user resumes Mirror mode from SLEEP mode of RXONLY_CH.

If there is no other copy process from a previously handled message from the upper 16 buffers pending, the TRQ flag of the buffer with DN = 1 is set (1). The MME simultaneously memorises that a mirror mode operation is in progress (internal MMP flag is set) in order to queue additional requests (newly received messages from RXONLY_CH) for later execution. Up to 16 requests are stored by the MME. The MMP flag is cleared when all pending requests have been dealt with.

When the transmission by the DIAG_CH is finished, the MME clears the respective DN flag and it clears the current request for the MME (i.e. decrement the up down counter). The buffer number can be obtained from THL of DIAG_CH.

If no further requests are pending (MMP = 0) the MME turns idle until the RXONLY_CH receives a new message. If there are other requests pending, the MME walks to the next buffer (in RX-Store direction of RXONLY_CH), writes MMP = 1, and sets the TRQ for that message buffer.

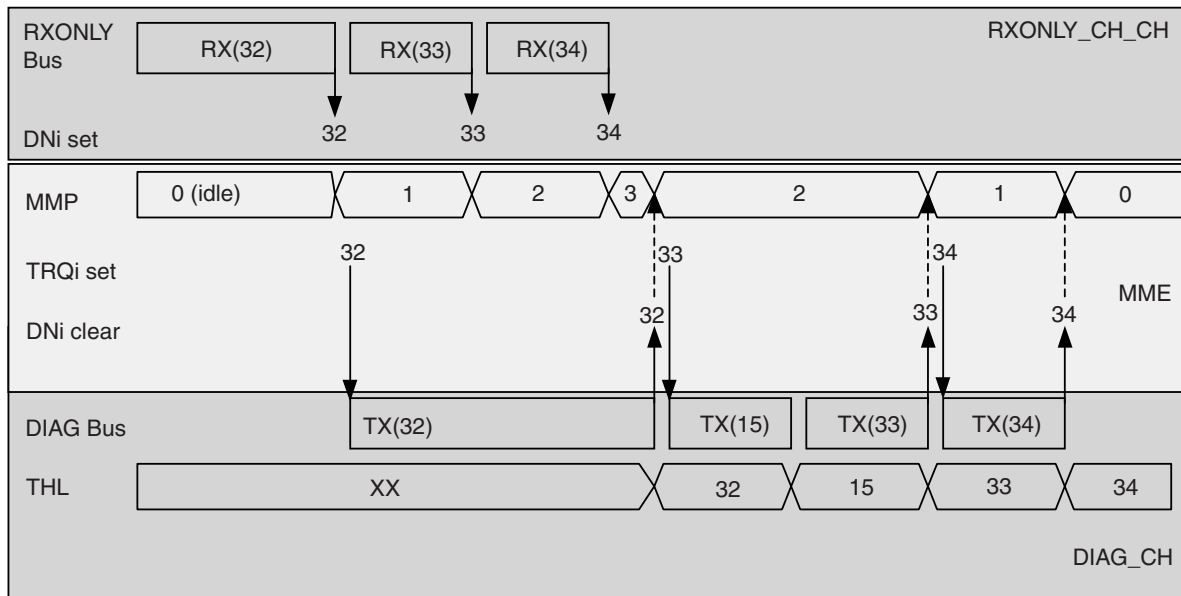
CHAPTER 8 OPERATIONAL MODES OF RXONLY_CH

The 'transmission complete' state can be easily detected by the TX-pending signal for the TX-complete interrupt. Thus, in Mirror mode the interrupt enable bit has to be set for all upper 16 message buffers. This configuration will also generate RX-interrupts from RXONLYX_CH. However the application can choose to suppress any interrupts generated by new receptions on the RXONLY_CH by masking the respective interrupt vector individually in the system interrupt controller or in the CnIE register of the AFCAN. The TX-complete interrupt can not be masked completely because the same interrupt vector is also used for the transmissions for the lower buffer numbers (#0 - #31). However the interrupt can be masked partially by the host CPU. For the transmissions from the upper message buffers the CPU can choose to mask the interrupt generation separately. This partial mask is only effective for transmit interrupt of the DIAG_CH in Mirror mode, i.e. when the upper 16 message buffers are assigned to the RXONLY_CH. Thus the CPU can configure both interrupts (RX for REXONLY_CH and sub-mask (#32 - #47) for TX-complete on DIAG_CH) such that it is able to follow up the Mirror mode operations (intrusive operation) or not (i.e. non-intrusive operation of Mirror mode). Any TX-complete signal generated by the upper 16 buffers, activates the MME. Then the MME clears the TX-interrupt pending signal before it clears the DN flag, and it decrements the MMP-counter as described above.

Note that the MMP flag is cleared when Mirror Mode is cancelled by a transition to INIT mode of RXONLY_CH. Any changes of the operational or power save modes of the DIAG_CH and switching the RXONLY_MODE to SLEEP and further to STOP mode have no influence on MMP flag. Thus the mirror mode can be resumed later on at the position where it suspended.

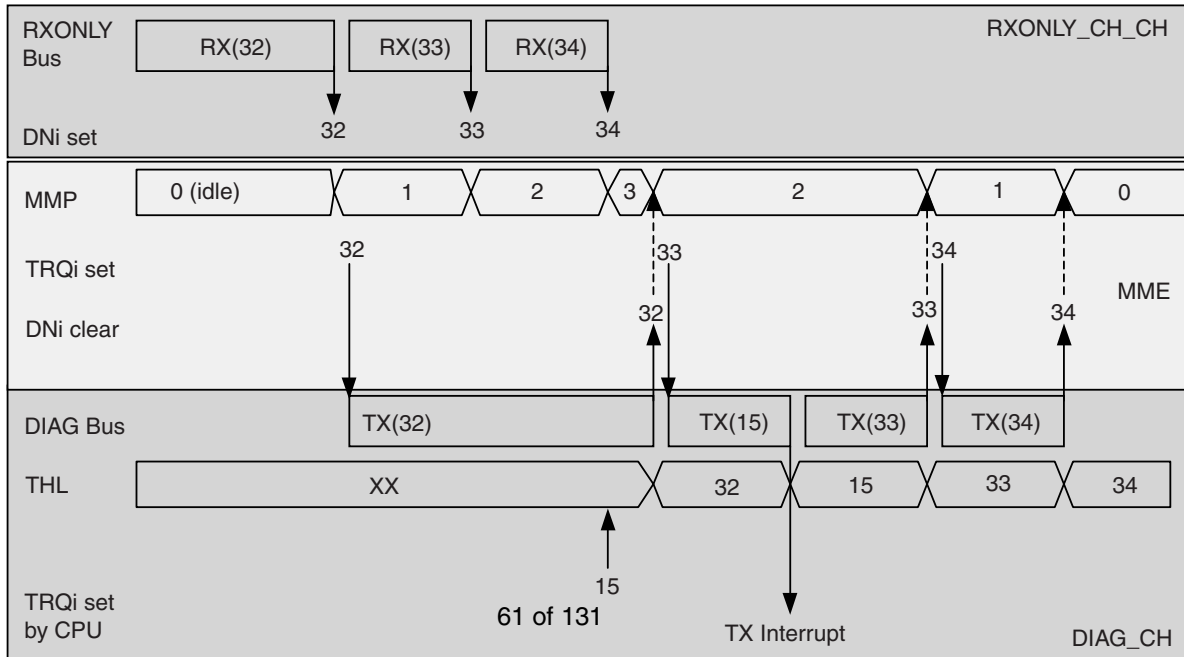
The following diagrams illustrates the phases of operation in Mirror mode.

Figure 8-3: Mirror Mode without Application Communication on DIAG_CH



CHAPTER 8 OPERATIONAL MODES OF RXONLY_CH

Figure 8-4: Mirror Mode with interleaved Application Communication on DIAG_CH



The RXONLY_CH can monitor up to eight other CAN channels. These sources can be selected via the BSEL[2:0] bits in the CnBSEL_R register during initialisation mode. Changing the selection during other operational modes is inhibited.

8.3 Mirror Mode with TIF

The mirror mode with TIF (Transfer Identifier filtering) applies acceptance filtering to the messages received on RXONLY_CH. All other operations are identical to Mirror Mode (OPMODE = 110b).

Up to 8 FullCAN identifiers can be specified in the transfer ID reference registers. Additionally one mask (Transfer ID Mask) register can be linked individually to each of the reference registers.

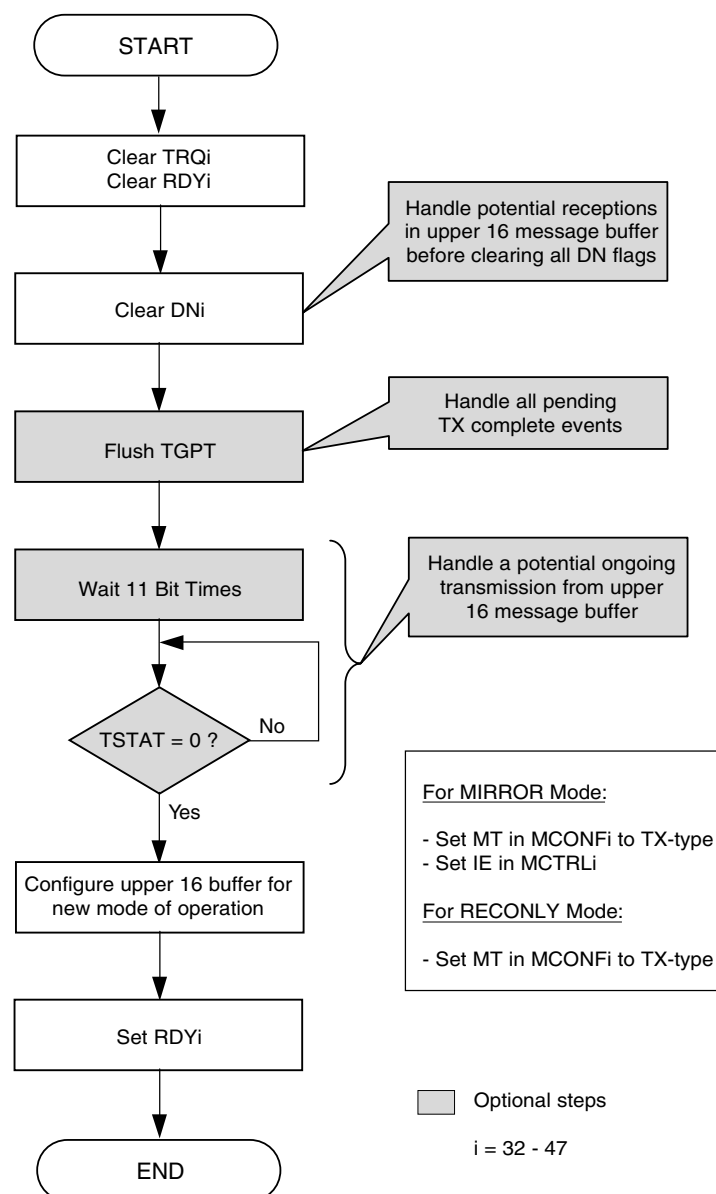
Any message matching one of the filter criteria will be stored in the upper 16 message buffer and thus participate in the mirror mode function. All messages not matching any filter criteria are not stored. This feature provides the application with a tool that minimizes the data throughput to a relevant subset of messages.

CHAPTER 9 TRANSITIONS FOR BUFFER ASSIGNMENT

The DAFCAN macro provides for each channel several operational modes and several power save modes. When considering all possible states with respect to the independent settings for the DIAG_CH and the RXONLY_CH, a huge number of transitions are basically available. However, the majority of mode transitions are not possible because i.e. the direct transition from NORMAL to NORMAL with ABT is not allowed, as well as STOP mode can not be directly followed by NORMAL mode.

This chapter describes the operations the host CPU has to perform in order to correctly switch between power save and operational modes of any of the two CAN channels of the DAFCAN macro. The flow chart below sketches the algorithm the host CPU needs to issue any time the buffer assignment changes.

Figure 9-1: Changing the Assignment of the upper 16 Buffers



CHAPTER 9 TRANSITIONS FOR BUFFER ASSIGNMENT

Whenever the assignment of the upper 16 buffers shall be changed, the host CPU needs to clear their RDY, TRQ, and DN bits at first. Before the DN flags are cleared, the application can optionally serve newly received messages.

In a second step the TGPT needs to be flushed, which forces the host CPU to handle all pending TX-complete events. If the application does not care on these events, this step can be skipped.

The third step is again optional and only applicable if the application used at least one of the upper 16 buffers for transmitting application messages. If the application likes to follow the sequence of these messages, it needs to wait 11 bit times and then check for bus idle. After this wait-algorithm any ongoing TX-messages from the DIAG_CH is completed and the transition flow can step ahead.

Then, the new buffer configuration is set up. When switching to RXONLY_CH assignment, all upper 16 buffers need to be configured as TX-type. If additionally Mirror mode shall be run, the IE bit of these buffers need to be set.

When switching the assignment to the DIAG_CH the application can setup its individual configuration.

In the second but last step, the new operational mode and the new power save mode for any of the channels can be invoked.

Finally, the RDY bits of all upper 16 message buffers need to be set again when the assignment is switched to the RXONLY_CH. In opposite direction (DIAG_CH) the RDY bits can be configured according the specific needs of the application.

CHAPTER 10 REGISTER DESCRIPTION

Not all registers will be discussed in detail within this addendum. Although all registers will be listed in the overview tables, the descriptions of registers, which have the same functionality as in the standard AFCAN channels can be found in the AFCAN User's Manual.

10.1 Registers of Global Macro Control

Table 10-1: Global Macro Registers (CPU read access) (1/2)

Register Name	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnGMCTRL	00H	0	0	0	0	0	0	EFSD	GOM
	01H	MBON	0	0	0	0	0	0	0
CnGMCS	02H	0	0	0	0	CCP3	CCP2	CCP1	CCP0
	03H	access prohibited							
CnGMCONF	04H	GCONF 7	GCONF 6	GCONF 5	GCONF 4	GCONF 3	GCONF 2	GCONF 1	GCONF 0
	05H	GCONF 15	GCONF 14	GCONF 13	GCONF 12	GCONF 11	GCONF 10	GCONF 9	GCONF 8
CnGMABT	06H	0	0	0	0	0	0	ABTCLR	ABTTRG
	07H	0	0	0	0	0	0	0	0
CnGMABTD	08H	0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0
	09H - 3FH	access prohibited							
CnTIDR0L	0CH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	0DH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR0H	0EH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	0FH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR1L	10H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	11H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR1H	12H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	13H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR2L	14H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	15H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR2H	16H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	17H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR3L	18H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	19H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR3H	1AH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	1BH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR4L	1CH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	1DH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR4H	1EH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	1FH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24

CHAPTER 10 REGISTER DESCRIPTION

Table 10-1: Global Macro Registers (CPU read access) (2/2)

Register	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnTIDR5L	20H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	21H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR5H	22H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	23H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR6L	24H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	25H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR6H	26H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	27H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR7L	28H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	29H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR7H	2AH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	2BH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDML	2CH	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	2DH	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnTIDMH	2EH	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
	2FH	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
	30H - 3FH	access prohibited							

CHAPTER 10 REGISTER DESCRIPTION

Table 10-2: Global Macro Registers (CPU write access) (1/2)

Register Name	Addr.-Off-set	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnGMCTRL	00H	0	0	0	0	0	0	0	clear GOM
	01H	0	0	0	0	0	0	set EFSD	set GOM
CnGMCS	02H	0	0	0	0	CCP3	CCP2	CCP1	CCP0
	03H - 05H	access prohibited							
CnGMABT	06H	0	0	0	0	0	0	0	clear ABTTRG
	07H	0	0	0	0	0	0	set ABTCLR	set ABTTRG
CnGMABTD	08H	0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0
	09H - 0BH	access prohibited							
CnTIDR0L	0CH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	0DH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR0H	0EH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	0FH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR1L	10H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	11H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR1H	12H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	13H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR2L	14H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	15H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR2H	16H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	17H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR3L	18H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	19H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR3H	1AH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	1BH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR4L	1CH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	1DH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR4H	1EH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	1FH	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR5L	20H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	21H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR5H	22H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	23H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24
CnTIDR6L	24H	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	25H	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnTIDR6H	26H	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	27H	IDE	IME	0	ID28	ID27	ID26	ID25	ID24

CHAPTER 10 REGISTER DESCRIPTION

CnTIDR1L									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	10H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	11H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR1H									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	12H	xxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	Undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	13H	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	Undef.	undef.		
CnTIDR2L									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	14H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	15H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR2H									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	16H	xxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	17H	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	undef.	undef.		

CHAPTER 10 REGISTER DESCRIPTION

									Address Offset	Initial Value
CnTIDR3L										
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	18H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	19H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR3H										
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	1AH	xxxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	1BH	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	undef.	undef.		
CnTIDR4L									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	1CH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	1DH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR4H										
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	1EH	xxxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	1FH	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	undef.	undef.		

CHAPTER 10 REGISTER DESCRIPTION

									Address Offset	Initial Value
CnTIDR5L										
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	20H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	21H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR5H										
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	22H	xxxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	23H	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	undef.	undef.		
CnTIDR6L									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	24H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	25H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
CnTIDR6H										
(read/write)	7	6	5	4	3	2	1	0		
	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	26H	xxxxxxxxB
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	IDE	IME	0	ID28	ID27	ID26	ID25	ID24	27H	xx0xxxxxB
Initial Value	undef.	undef.	0	undef.	undef.	undef.	undef.	undef.		

CHAPTER 10 REGISTER DESCRIPTION

CnTIDR7L		Address		Initial	
		Offset		Value	
(read/write)	7 6 5 4 3 2 1 0				
	ID7 ID6 ID5 ID4 ID3 ID2 ID1 ID0	28H		undef.	
Initial Value	undef. undef. undef. undef. undef. undef. undef. undef.				
	15 14 13 12 11 10 9 8				
	ID15 ID14 ID13 ID12 ID11 ID10 ID9 ID8	29H		undef.	
Initial Value	undef. undef. undef. undef. undef. undef. undef. undef.				
CnTIDR7H					
(read/write)	7 6 5 4 3 2 1 0				
	ID23 ID22 ID21 ID20 ID19 ID18 ID17 ID16	2AH		xxxxxxxxB	
Initial Value	undef. undef. undef. undef. undef. undef. undef. undef.				
	15 14 13 12 11 10 9 8				
	IDE IME 0 ID28 ID27 ID26 ID25 ID24	2BH		xx0xxxxxB	
Initial Value	undef. undef. 0 undef. undef. undef. undef. undef.				

ID28 to ID0	Message ID Reference for Transfer Message
ID28 to ID18	range for the 11-bit Standard identifier values
ID28 to ID0	range for the 29-bit Extended identifier values

IDE	Identifier Extension Bit for Transfer Message
0	11-bit Standard identifier ^{Note}
1	29-bit Extended identifier

Note: The bits ID17 to ID0 are not used and may contain undefined values.

IME	Mask Enable bit for Transfer Message
0	CnTIDRL/H register without a link to the CnTIDML/H
1	CnTIDRL/H register with a link to the CnTIDML/H

Caution: Be sure to write “0” to bit 13 of the CnTIDRHm registers. When CPU attempts to write CnTIDRLm and CnTIDRHm in a mode other than INIT, it is simply ignored.

CHAPTER 10 REGISTER DESCRIPTION

10.1.2 CAN Transfer ID Mask Registers (CnTIDML, CnTIDMH)

The CnTIDML and CnTIDMH registers are used to extend the number of receivable messages into the upper 16 message buffers by masking part of the ID of a message and invalidating the ID of the masked part.

CnTIDML									Address Offset	Initial Value
(read / write)	7	6	5	4	3	2	1	0		
	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0	2CH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		

	15	14	13	12	11	10	9	8		
	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	2DH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		

CnTIDMH									Address Offset	Initial Value
(read / write)	7	6	5	4	3	2	1	0		
	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16	2EH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		
	15	14	13	12	11	10	9	8		
	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24	2FH	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		

CMID28 to CMID0	Mask Identifier Pattern for Transfer message
0	Corresponding identifier bit in the received message frame and in the message buffer must match
1	Corresponding identifier bit in the received message frame and in the message do not care

Remark: When CPU attempts to write CnTIDML and CnTIDMH in a mode other than INIT, it is simply ignored.

CHAPTER 10 REGISTER DESCRIPTION

10.2 Registers of the CAN Module

Table 10-3: CAN Module Registers (CPU read access) (1/2)

Register Name	Addr.-Offset	Bit 7/ 15	Bit 6/ 14	Bit 5/ 13	Bit 4/ 12	Bit 3/ 11	Bit 2/ 10	Bit 1/ 9	Bit 0/ 8
CnMASK1L	00H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	01H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK1H	02H	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	03H	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK2L	04H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	05H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK2H	06H	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	07H	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK3L	08H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	09H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK3H	0AH	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	0BH	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK4L	0CH	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	0DH	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK4H	0EH	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
	0FH	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnCTRL	10H	CCERC	AL	VALID	PSMODE1	PSMODE0	OPMODE2	OPMODE1	OPMODE0
	11H	0	0	0	0	0	0	RSTAT	TSTAT
CnLEC	12H	0	0	0	0	0	LEC2	LEC1	LEC0
CnINFO	13H	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0
CnERC	14H	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
	15H	REPS	REC6	REC5	REC4	REC3	REC2	REC1	REC0
CnIE	16H	CIE7	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0
	17H	0	0	0	0	0	0	0	0
CnINTS	18H	CINTS7	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0
	19H	0	0	0	0	0	0	0	0
CnBRP	1AH	TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0
	1BH	access prohibited							
CnBTR	1CH	0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10
	1DH	0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
CnLIPT	1EH	LIPT7	LIPT6	LIPT5	LIPT4	LIPT3	LIPT2	LIPT1	LIPT0
	1FH	access prohibited							
CnRGPT	20H	0	0	0	0	0	0	RHPM	ROVF
	21H	RGPT7	RGPT6	RGPT5	RGPT4	RGPT3	RGPT2	RGPT1	RGPT0
CnLOPT	22H	LOPT7	LOPT6	LOPT5	LOPT4	LOPT3	LOPT2	LOPT1	LOPT0
	23H	access prohibited							
CnTGPT	24H	0	0	0	0	0	0	THPM	TOVF
	25H	TGPT7	TGPT6	TGPT5	TGPT4	TGPT3	TGPT2	TGPT1	TGPT0

CHAPTER 10 REGISTER DESCRIPTION

Table 10-3: CAN Module Registers (CPU read access) (2/2)

Register Name	Addr.-Offset	Bit 7/ 15	Bit 6/ 14	Bit 5/ 13	Bit 4/ 12	Bit 3/ 11	Bit 2/ 10	Bit 1/ 9	Bit 0/ 8
CnTS ^{Note}	26H	0	0	0	0	0	TSLOCK	TSSEL	TSEN
	27H	0	0	0	0	0	0	0	0
CnCTRL_R	28H	CCERC	0	VALID	PSMODE1	PSMODE0	OPMODE2	OPMODE1	OPMODE0
	29H	0	0	0	0	0	0	RSTAT	0
CnLEC_R	2AH	0	0	0	0	0	LEC2	LEC1	LEC0
	2BH	access prohibited							
CnERC_R	2CH	0	0	0	0	0	0	0	0
	2DH	REPS	REC6	REC5	REC4	REC3	REC2	REC1	REC0
CnIE_R	2EH	0	CIE6	CIE5	0	CIE3	CIE2	CIE1	0
	2FH	0	0	0	0	0	0	0	0
CnINTS_R	30H	0	CINTS6	CINTS5	0	CINTS3	CINTS2	CINTS1	0
	31H	0	0	0	0	0	0	0	0
CnBRP_R	32H	TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0
	33H	access prohibited							
CnBTR_R	34H	0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10
	35H	0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
CnLIPT_R	36H	LIPT7	LIPT6	LIPT5	LIPT4	LIPT3	LIPT2	LIPT1	LIPT0
	37H	access prohibited							
CnTS_R ^{Note}	38H	0	0	0	0	0	TSLOCK	TSSEL	TSEN
	39H	0	0	0	0	0	0	0	0
CnBSEL_R	3AH	0	0	0	0	0	BSEL2	BSEL1	BSEL0
	3BH - 3FH	access prohibited							

Note: The displayed CnTS and CnTS_R registers provide only the necessary bits for the Basic Time Stamp function.
Modifications will have to be done for the Advanced Time Stamp function.

CHAPTER 10 REGISTER DESCRIPTION

Table 10-4: CAN Module Registers (CPU write access) (1/2)

Register Name	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnMASK1L	00H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	01H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK1H	02H	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	03H	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK2L	04H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	05H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK2H	06H	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	07H	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK3L	08H	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	09H	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK3L	0AH	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
	0BH	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnMASK4L	0CH	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
	0DH	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
CnMASK4H	0EH	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID016
	0FH	0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
CnCTRL	10H	0	clear AL	clear VALID	clear PSMODE1	clear PSMODE0	clear OPMODE2	clear OPMODE1	clear OPMODE0
	11H	Set CCERC	set AL	set VALID	set PSMODE1	set PSMODE0	set OPMODE2	set OPMODE1	set OPMODE0
CnLEC	12H	0	0	0	0	0	LEC2	LEC1	LEC0
CnINFO	13H	0	0	0	0	0	0	0	0
CnERC	14H	0	0	0	0	0	0	0	0
	15H	0	0	0	0	0	0	0	0
CnIE	16H	clear CIE7	0	clear CIE5	clear CIE4	clear CIE3	clear CIE2	clear CIE1	clear CIE0
	17H	set CIE7	0	set CIE5	set CIE4	set CIE3	set CIE2	set CIE1	set CIE0
CnINTS	18H	clear CINTS7	0	clear CINTS5	clear CINTS4	clear CINTS3	clear CINTS2	clear CINTS1	clear CINTS0
	19H	0	0	0	0	0	0	0	0
CnBRP	1AH	TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0
	1BH	access prohibited							
CnBTR	1CH	0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10
	1DH	0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
CnLIPT	1EH	0	0	0	0	0	0	0	0
	1FH	access prohibited							
CnRGPT	20H	0	0	0	0	0	0	0	clear ROVF
	21H	0	0	0	0	0	0	0	0
CnLOPT	22H	0	0	0	0	0	0	0	0
	23H	access prohibited							

CHAPTER 10 REGISTER DESCRIPTION

Table 10-4: CAN Module Registers (CPU write access) (2/2)

Register Name	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnTGPT	24H	0	0	0	0	0	0	0	clear TOVF
	25H	0	0	0	0	0	0	0	0
CnTS ^{Note}	26H	0	0	0	0	0	clear TSLOCK	clear TSSEL	clear TSEN
	27H	0	0	0	0	0	set TSLOCK	set TSSEL	set TSEN
CnCTRL_R	28H	0	0	clear VALID	clear PSMODE1	clear PSMODE0	clear OPMODE2	clear OPMODE1	clear OPMODE0
	29H	Set CCERC	0	0	set PSMODE1	set PSMODE0	set OPMODE2	set OPMODE1	set OPMODE0
CnLEC_R	2AH	0	0	0	0	0	LEC2	LEC1	LEC0
	2BH	access prohibited							
CnERC_R	2CH	0	0	0	0	0	0	0	0
	2DH	0	0	0	0	0	0	0	0
CnIE_R	2EH	0	clear CIE6	clear CIE5	0	clear CIE3	clear CIE2	clear CIE1	0
	2FH	0	set CIE6	set CIE5	0	set CIE3	set CIE2	set CIE1	0
CnINTS_R	30H	0	clear CINTS6	clear CINTS5		clear CINTS3	clear CINTS2	clear CINTS1	0
	31H	0	0	0	0	0	0	0	0
CnBRP_R	32H	TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0
	33H	access prohibited							
CnBTR_R	34H	0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10
	35H	0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
CnLIPT_R	36H	0	0	0	0	0	0	0	
	37H	access prohibited							
CnTS_R ^{Note}	38H						clear TSLOCK	clear TSSEL	clear TSEN
	39H						set TSLOCK	set TSSEL	set TSEN
CnBSEL_R	3AH	0	0	0	0	0	BSEL2	BSEL1	BSEL0
	3BH - 3FH	access prohibited							

Note: The displayed CnTS and CnRS-R registers provide only the necessary bits for the Basic Time Stamp function. Modifications will have to be done for the Advanced Time Stamp function.

CHAPTER 10 REGISTER DESCRIPTION

10.2.1 RXONLY_CH CAN Module Control Register (CnCTRL_R)

CnCTRL_R	Address Offset	Initial Value
(read)		
7 6 5 4 3 2 1 0		
CCERC 0 VALID PSMODE1 PSMODE0 OPMODE2 OPMODE1 OPMODE0	28H	00H
Initial Value		0 0 0 0 0 0 0 0
15 14 13 12 11 10 9 8		
0 0 0 0 0 0 RSTAT 0	29H	00H
Initial Value		0 0 0 0 0 0 0 0

CnCTRL_R	Address Offset	Initial Value
(write)		
7 6 5 4 3 2 1 0		
0 0 clear clear Clear clear clear clear VALID PSMODE1 PSMODE0 OPMODE2 OPMODE1 OPMODE0	28H	
15 14 13 12 11 10 9 8		
set 0 0 set Set Set set set CCERC PSMODE1 PSMODE0 OPMODE2 OPMODE1 OPMODE0	29H	

OPMODE2	OPMODE1	OPMODE0	Operational Mode
0	0	0	No operational mode selected (RXONLY_CH CAN module is in INIT mode)
0	0	1	Prohibited to use / ignored by hardware
0	1	0	
0	1	1	"Receive-only mode"
1	0	0	Prohibited to use / ignored by hardware
1	0	1	
1	1	0	"Mirror mode"
1	1	1	"Mirror mode with Transfer ID Filter function (w/ TIF)"

PSMODE1	PSMODE0	Power Save Mode
0	0	No power save mode selected (RXONLY_CH CAN module is in INIT mode or in one of the operational modes)
0	1	SLEEP mode
1	0	Prohibited to use / ignored by hardware
1	1	STOP mode

CHAPTER 10 REGISTER DESCRIPTION

VALID	Valid Reception bit
0	No valid message frame reception into the RXONLY_CH CAN Protocol Transfer Layer since the VALID bit had been cleared (0) last time.
1	Valid message frame reception into the RXONLY_CH CAN Protocol Transfer Layer since the VALID bit had been cleared (0) last time.

Caution: User must not set the VALID bit (1) by using set VALID bit.
 A valid reception does not require an acceptance of the message frame into a receive message buffer (data or remote frame).
 Before switching from INIT mode to any operational mode the user must clear the VALID bit (0).
 In case only two CAN nodes are connected to a CAN bus and one of the CAN nodes is in “Normal Operating mode” transmitting message frames while the other CAN node is in “Receive-only mode”, the VALID bit will be set (1) not before the transmitting node becomes error passive.

CCERC	RXONLY_CH CAN Clear Error Counter bit
0	While in INIT mode the RXONLY_CH CAN module error counter CnERC_R will not be cleared.
1	While in INIT mode the RXONLY_CH CAN module error counter CnERC_R will be cleared

RSTAT	RXONLY_CH CAN Reception status
0	No receive activity on the RXONLY CAN bus
1	Receive activity on the RXONLY CAN bus

- Remarks:**
- RSTAT is set under the following condition:
SOF of RX frame
 - RSTAT is cleared under the following condition:
After detecting recessive bit at second bit of inter frame space
Transition to INIT mode at 1st bit of 'interframe space'

10.2.2 RXONLY_CH CAN Module Last Error Code Register (CnLEC_R)

CnLEC_R									Address Offset	Initial Value
(read / write)	7	6	5	4	3	2	1	0		
	0	0	0	0	0	LEC2	LEC1	LEC0	2AH	00H
Initial Value	0	0	0	0	0	0	0	0		

Remark: The actual content of the CnLEC_R is not cleared by switching the RXONLY_CH CAN module from an operational mode to the INIT mode.
 When CPU attempts to write CnLEC_R with data other than 00h, it is simply ignored.

CHAPTER 10 REGISTER DESCRIPTION

LEC2	LEC1	LEC0	Last Error Code of the Protocol Error Type
0	0	0	No error
0	0	1	Stuff error
0	1	0	Form error
1	1	0	CRC error
0	1	1	Unused
1	0	0	
1	0	1	
1	1	1	

10.2.3 RXONLY_CH CAN Module Error Counter Register (CnERC_R)

CnERC_R									Address Offset	Initial Value
(read-only)	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	2CH	00H
Initial Value	0	0	0	0	0	0	0	0		
		15	14	13	12	11	10	9	8	
	REPS	REC6	REC5	REC4	REC3	REC2	REC1	REC0	2DH	00H
Initial Value	0	0	0	0	0	0	0	0		

CHAPTER 10 REGISTER DESCRIPTION

10.2.4 RXONLY_CH CAN Module Interrupt Enable Register (CnIE_R)

CnIE_R									Address Offset	Initial Value
(read)	7	6	5	4	3	2	1	0		
	0	CIE6	CIE5	0	CIE3	CIE2	CIE1	0	2EH	00H
Initial Value	0	0	0	0	0	0	0	0		
	15	14	13	12	11	10	9	8		
	0	0	0	0	0	0	0	0	2FH	00H
Initial Value	0	0	0	0	0	0	0	0		

CnIE_R									Address Offset	Initial Value
(write)	7	6	5	4	3	2	1	0		
	0	clear CIE6	clear CIE5	0	clear CIE3	Clear CIE2	Clear CIE1	0	2EH	
	15	14	13	12	11	10	9	8		
	0	set CIE6	set CIE5	0	set CIE3	Set CIE2	set CIE1	0	2FH	

CIE1 to CIE6	RXONLY_CH CAN Interrupt Enable bits
0	Corresponding interrupt request signal INTxx is not generated when CINTSx is set by the interrupt source event.
1	Corresponding interrupt request signal INTxx is generated when CINTSx is set by the interrupt source event.

CHAPTER 10 REGISTER DESCRIPTION

10.2.5 RXONLY_CH CAN Module Interrupt Status Register (CnINTS_R)

CnINTS_R									Address Offset	Initial Value
(read)	7	6	5	4	3	2	1	0		
	0	CINTS6	CINTS5	0	CINTS3	CINTS2	CINTS1	0	30H	00H
Initial Value	0	0	0	0	0	0	0	0		
	15	14	13	12	11	10	9	8		
	0	0	0	0	0	0	0	0	31H	00H
Initial Value	0	0	0	0	0	0	0	0		

CnINTS_R									Address Offset	Initial Value
(write)	7	6	5	4	3	2	1	0		
	0	clear CINTS6	clear CINTS5	0	clear CINTS3	clear CINTS2	clear CINTS1	0	30H	
	15	14	13	12	11	10	9	8		
	0	0	0	0	0	0	0	0	31H	

CINTS1 to CINTS6	RXONLY_CH CAN Interrupt Status bits
0	The related interrupt source event is not pending
1	The related interrupt source event is pending

Interrupt status bit	Related interrupt source event
CINTS1	RXONLY_CH CAN module interrupt status bit for interrupt event 'Valid message frame reception in the FIFO area of message buffer
CINTS2	RXONLY_CH CAN module error state interrupt status
CINTS3	RXONLY_CH CAN module protocol error interrupt status
CINTS5	RXONLY_CH CAN module wake-up from SLEEP mode interrupt status bit Note
CINTS6	RXONLY_CH CAN module FIFO overflow interrupt status

Note: Only the wake-up from SLEEP mode by CAN bus activity generates the CINTS5 signal. The SLEEP mode release by CPU will not generate the CINTS5 signal.

CHAPTER 10 REGISTER DESCRIPTION

10.2.6 RXONLY_CH CAN Module Bit-Rate Prescaler Register (CnBRP_R)

CnBRP_R									Address Offset	Initial Value
(read / write)	7	6	5	4	3	2	1	0		
	TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0	32H	FFH
Initial Value	1	1	1	1	1	1	1	1		

TQPRS0 to TQPRS7	Time Quanta clock prescaler
0	$\phi_{TQ} = \phi_{CANMOD} / 1$
1	$\phi_{TQ} = \phi_{CANMOD} / 2$
N	$\phi_{TQ} = \phi_{CANMOD} / (n+1)$
...	...
255	$\phi_{TQ} = \phi_{CANMOD} / 256$

CHAPTER 10 REGISTER DESCRIPTION

10.2.8 RXONLY_CH CAN Module Last In-Pointer Register (CnLIPT_R)

CnLIPT_R											Address Offset	Initial Value
(read-only)	7	6	5	4	3	2	1	0				
	LIPT7	LIPT6	LIPT5	LIPT4	LIPT3	LIPT2	LIPT1	LIPT0	36H		undef.	
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.			undef.	

LIPT0 to LIPT7	Last In-Pointer of the Receive History List for the RXONLY_CH
message buffer 0 ... 47	<p>Reading the CnLIPT_R register delivers the message buffer number in which the last data frame has been saved or the last remote frame has been stored into the message buffer area of RXONLY_CH. The user can only evaluate CnLIPT_R when both of the following conditions are met:</p> <ul style="list-style-type: none"> - the CAN channel is in Receive-Only or MIRROR mode. - A newly received message on RXONLY_CH was signalled by interrupt CINTS1 in CnINTS_R. As a pre-condition RXONLY_CH needs to have been put in INIT or STOP mode beforehand at least once. <p>The usage of CnLIPT_R requires proper interrupt handling.</p>

Remark: As long no data frame has been stored into a message buffer or no received remote frame has been assigned, an undefined value is read from CnLIPT_R register. The user must not read the CnLIPT_R register until first message acceptance to the buffer from RXONLY_CH after switching the GOM bit.

This register shows the reception history of the messages from RXONLY_CH only.

CHAPTER 10 REGISTER DESCRIPTION

10.2.9 RXONLY_CH CAN Module Time Stamp Register (CnTS_R)

CnTS_R									Address Offset	Initial Value
(read)	7	6	5	4	3	2	1	0		
	0	0	0	0	0	TSLOCK	TSSEL	TSEN	38H	00H
Initial Value	0	0	0	0	0	0	0	0		
	15	14	13	12	11	10	9	8		
	0	0	0	0	0	0	0	0	39H	00H
Initial Value	0	0	0	0	0	0	0	0		

CnTS_R									Address Offset	Initial Value
(write)	7	6	5	4	3	2	1	0		
	0	0	0	0	0	clear TSLOCK	clear TSSEL	clear TSEN	38H	
	15	14	13	12	11	10	9	8		
	0	0	0	0	0	set TSLOCK	set TSSEL	set TSEN	39H	

- Remarks:**
- The displayed CnTS_R register provides only the necessary bits for the Basic Time Stamp function.
The Advanced Time Stamp function requires a modified hardware.
 - The Basic Time Stamp function must not be used when the DIAG_CH CAN module operates in 'Normal Operating mode with Automatic Block Transmission.

TSEN	Time Stamp Signal Generation Enable bit
0	The time stamp signal TSOUT_R generation is disabled.
1	The time stamp signal TSOUT_R generation is enabled.

TSSEL	Time Stamp Capture Event Selection bit
0	The time stamp capture event is the SOF event (Start-of-Frame on the RXONLY_CH CAN-bus)
1	The time stamp capture event is the EOF event (the last bit of the End-of-Frame field. TSOUT_R signal is generated at the sample point of the last bit in the EOF field)

TSLOCK	Time Stamp Lock Function Enable bit
0	The time stamp lock function is disabled. Upon every selected time stamp capture event the TSOUT_R signal is toggled.
1	The time stamp lock function is enabled. The TSOUT_R signal generation is locked after a data frame was successfully received in message buffer 0.

Caution: The TSLOCK function on RXONLY_CH is only invoked when Mirror mode, mirror mode w/ TIF or Receive-Only mode are active. At all other operational modes of the DAFCAN, the TSLOCK function via TSOUT_D is active for all message buffers.

CHAPTER 10 REGISTER DESCRIPTION

10.2.10 RXONLY_CH CAN Module Bus Selector (CnBSEL_R)

CnBSEL_R									Address Offset	Initial Value
(read / write)	7	6	5	4	3	2	1	0		
	0	0	0	0	0	BSEL2	BSEL1	BSEL0	3AH	00H
Initial Value	0	0	0	0	0	0	0	0		

BSEL2 to BSEL0	Bus selector for selecting to be monitored CAN BUS
0... CAN bus 4	Selecting the connection of RXONLY_CH

BSEL2	BSEL1	BSEL0	Selected CAN BUS input source
0	0	0	DIAG_CH CAN-bus
0	0	1	CAN-bus input 1
0	1	0	CAN-bus input 2
0	1	1	CAN-bus input 3
1	0	0	CAN-bus input 4
1	0	1	CAN-bus input 5
1	1	0	CAN-bus input 6
1	1	1	CAN-bus input 7

- Remarks:**
1. When CPU attempts to write CnBSEL_R in the mode other than INIT, it is simply ignored.
 2. When CnBSEL_R is "000B", the RXONLY_CH is connected to the DIAG_CH CAN BUS.

CHAPTER 10 REGISTER DESCRIPTION

10.3 Registers of Message Buffers

Table 10-5: CAN Message Buffer Registers (CPU read access)

Register Name	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/ 9	Bit 0/ 8
CnMDATAm0	00H	MDATA07	MDATA06	MDATA05	MDATA04	MDATA03	MDATA02	MDATA01	MDATA00
CnMDATAm1	01H	MDATA17	MDATA16	MDATA15	MDATA14	MDATA13	MDATA12	MDATA11	MDATA10
CnMDATAm2	02H	MDATA27	MDATA26	MDATA25	MDATA24	MDATA23	MDATA22	MDATA21	MDATA20
CnMDATAm3	03H	MDATA37	MDATA36	MDATA35	MDATA34	MDATA33	MDATA32	MDATA31	MDATA30
CnMDATAm4	04H	MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40
CnMDATAm5	05H	MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50
CnMDATAm6	06H	MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60
CnMDATAm7	07H	MDATA77	MDATA76	MDATA75	MDATA74	MDATA73	MDATA72	MDATA71	MDATA70
CnMDLcM	08H	0	0	0	0	MDLC3	MDLC2	MDLC1	MDLC0
CnMCONFm	09H	OVS	RTR	MT2	MT1	MT0	MA2	MA1	MA0
CnMIDLm	0AH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	0BH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnMIDHm	0CH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	0DH	IDE	0	0	ID28	ID27	ID26	ID25	ID24
CnMCTRLm	0EH	0	0	0	MOW	IE	DN	TRQ	RDY
	0FH	0	0	MUC	0	0	0	0	0
	10H - 1FH	Access prohibited							

CHAPTER 10 REGISTER DESCRIPTION

Table 10-6: CAN Message Buffer Registers (CPU write access)

Register Name	Addr.-Offset	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
CnMDATAm0	00H	MDATA07	MDATA06	MDATA05	MDATA04	MDATA03	MDATA02	MDATA01	MDATA00
CnMDATAm1	01H	MDATA17	MDATA16	MDATA15	MDATA14	MDATA13	MDATA12	MDATA11	MDATA10
CnMDATAm2	02H	MDATA27	MDATA26	MDATA25	MDATA24	MDATA23	MDATA22	MDATA21	MDATA20
CnMDATAm3	03H	MDATA37	MDATA36	MDATA35	MDATA34	MDATA33	MDATA32	MDATA31	MDATA30
CnMDATAm4	04H	MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40
CnMDATAm5	05H	MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50
CnMDATAm6	06H	MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60
CnMDATAm7	07H	MDATA77	MDATA76	MDATA75	MDATA74	MDATA73	MDATA72	MDATA71	MDATA70
CnMDLcM	08H	0	0	0	0	MDLC3	MDLC2	MDLC1	MDLC0
CnMCONFm	09H	OVS	RTR	MT2	MT1	MT0	MA2	MA1	MA0
CnMIDLm	0AH	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	0BH	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
CnMIDHm	0CH	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
	0DH	IDE	0	0	ID28	ID27	ID26	ID25	ID24
CnMCTRLm	0EH	0	0	0	clear MOW	clear IE	clear DN	clear TRQ	clear RDY
	0FH	0	0	0		set IE	set DN	set TRQ	set RDY
	10H - 1FH	Access prohibited							

CHAPTER 10 REGISTER DESCRIPTION

10.3.1 CAN Message Configuration Register (CnMCONFm)

CnMCONFm									Address Offset	Initial Value
(read/write)	7	6	5	4	3	2	1	0		
	OWS	RTR	MT2	MT1	MT0	MA2	MA1	MA0	09H	undef.
Initial Value	undef.	undef.	undef.	undef.	undef.	undef.	undef.	undef.		

MA2	MA1	MA0	Message Buffer Assignment
0	0	0	Message buffer is not assigned to any CAN I/F channel
0	0	1	Message buffer is assigned to CAN I/F channel 1
0	1	0	Prohibited
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Remark: The DAFCAN macro assigns RXONLY_CH and DIAG_CH to a particular message buffer automatically by the OPMODE/PSMODE of the RXONLY_CH.

CHAPTER 10 REGISTER DESCRIPTION

10.3.3 CAN Message Control Register (CnMCTRLm)

CnMCTRLm									Address Offset	Initial Value								
(read)	7	6	5	4	3	2	1	0		00x000000 00xx000B								
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">MOW</td> <td style="width: 12.5%; text-align: center;">IE</td> <td style="width: 12.5%; text-align: center;">DN</td> <td style="width: 12.5%; text-align: center;">TRQ</td> <td style="width: 12.5%; text-align: center;">RDY</td> </tr> </table>								0	0	0	MOW	IE	DN	TRQ	RDY	0EH	
0	0	0	MOW	IE	DN	TRQ	RDY											
Initial Value	0	0	0	undef.	undef.	0	0	0										
	15	14	13	12	11	10	9	8										
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">MUC</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> </tr> </table>								0	0	MUC	0	0	0	0	0	0FH	
0	0	MUC	0	0	0	0	0											
Initial Value	0	0	undef.	0	0	0	0	0										

CnMCTRLm									Address Offset	Initial Value								
(write)	7	6	5	4	3	2	1	0										
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">clear MOW</td> <td style="width: 12.5%; text-align: center;">clear IE</td> <td style="width: 12.5%; text-align: center;">clear DN</td> <td style="width: 12.5%; text-align: center;">clear TRQ</td> <td style="width: 12.5%; text-align: center;">clear RDY</td> </tr> </table>								0	0	0	clear MOW	clear IE	clear DN	clear TRQ	clear RDY	0EH	
0	0	0	clear MOW	clear IE	clear DN	clear TRQ	clear RDY											
	15	14	13	12	11	10	9	8										
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">0</td> <td style="width: 12.5%; text-align: center;">set IE</td> <td style="width: 12.5%; text-align: center;">set DN</td> <td style="width: 12.5%; text-align: center;">set TRQ</td> <td style="width: 12.5%; text-align: center;">set RDY</td> </tr> </table>								0	0	0	0	set IE	set DN	set TRQ	set RDY	0FH	
0	0	0	0	set IE	set DN	set TRQ	set RDY											

RDY	Message Buffer Ready bit
0	The CPU can write to the message buffer. The assigned CAN module does not access the message buffer.
1	The assigned CAN module accesses the message buffer. CPU write access to the message buffer is ignored (except write access to the RDY bit, TRQ bit, DN bit and MOW bit).

TRQ	Message Buffer Transmit Request bit
0	No message frame transmission request is pending or ongoing from the message buffer.
1	A message frame transmission request is pending or a message frame transmission is ongoing from the message buffer.

CHAPTER 10 REGISTER DESCRIPTION

DN	Message Buffer Data New bit
0	No remote frame has been stored into the message buffer (message buffer is defined as transmit message buffer (MT[2:0] = 000B)) No data frame has been stored into the message buffer (message buffer is defined as receive message buffer (MT[2:0] > 000B)).
1	A remote frame has been stored into the message buffer (message buffer is defined as transmit message buffer (MT[2:0] = 000B)) A data frame has been stored into the message buffer (message buffer is defined as receive message buffer (MT[2:0] > 000B)).

Remark: The user must not set the DN flag (1) by using the set-DN instruction. Special care applies to DN, TRQ, and RDY bit when Mirror Mode, mirror mode w/ TIF or Receive-Only mode is activated. Please, refer to chapter Chapter 8 "Operational Modes of RXONLY_CH" on page 45 for details.

IE	Interrupt Enable for message buffer interrupt event
0	Interrupt generation disabled for the following events: Interrupt events linked to CINTS0 interrupt status bit in CnINTS register {i.e. at MT[2:0] = 000B 'Data frame successfully transmitted from message buffer m', 'Remote frame successfully transmitted from message buffer m'} Interrupt events linked to CINTS1 interrupt status bit in CnINTS register {i.e. for MT[2:0] = 001B,010B,011B,100B or 101B 'Valid data frame reception in message buffer m' and for MT[2:0] = 000B 'Valid remote frame reception in message buffer m'} Interrupt events linked to CINTS1 interrupt status bit in CnINTS_R register when operated in MIRROR or Receive-Only mode {i.e. "Valid data or remote frame reception in message buffer m' for m= 32 – 47}
1	Interrupt generation enabled for the following events: Interrupt events linked to CINTS0 interrupt status bit in CnINTS register {i.e. MT[2:0] = 000B 'Data frame successfully transmitted from message buffer m', 'Remote frame successfully transmitted from message buffer m'} Interrupt events linked to CINTS1 interrupt status bit in CnINTS register {MT[2:0] = 001B,010B,011B,100B or 101B 'Valid data frame reception in message buffer m' and for MT[2:0] = 000B 'Valid remote frame reception in message buffer m'} Interrupt events linked to CINTS1 interrupt status bit in CnINTS_R register when operated in MIRROR or Receive-Only mode {i.e. "Valid data or remote frame reception in message buffer m' for m= 32 – 47}

MOW	Message buffer overwrite status bit
0	The message buffer is not overwritten by a newly received data frame.
1	The message buffer is overwritten by a newly received data frame.

Remark: The MOW bit is not set to 1 even if a remote frame is received and stored in the transmit message buffer with the DN bit = 1.

- Cautions:**
1. Do not set the TRQ bit and the RDY bit (1) at the same time. Set the RDY bit (1) before setting the TRQ bit.
 2. Do not clear the RDY bit (0) during message transmission.
 3. Follow the transmission abort process about clearing the RDY bit (0) for redefinition of the message buffer.
 4. Clear again when RDY bit is not cleared even if this bit is cleared.
 5. Be sure that RDY is cleared before writing to the other message buffer registers, by checking the status of the RDY bit.
 6. Set IE bit and RDY bit always separately.
 7. Clearing the MOW bit is not possible during reception of a new message by the RXONLY_CH. This is applicable, when the DAFCAN is operated in RXONLY or MIRROR modes. Therefore, software has to check the status of MOW, when clearing it, while RXONLY or MIRROR modes are active. If unsuccessful, the clearing of MOW has to be repeated.

[MEMO]

APPENDIX A INDEX

A	
AFCAN	17
B	
Baud rate	18
BSEL	76
C	
CCERC	68
Channels	18
CIE	70
CINTS	71
CMID	62
D	
DIAG_CH	17
Diagnostics	18
DN	82
E	
EFSD	25
F	
Forced Shutdown	26
I	
ID	80
IDE	61, 80
IE	82
IME	61
Initialisation	27
L	
LEC	68
LIPT	74
M	
MA	79
Masks	18
Message Organisation	18
Message Storage	18
Mirror Mode	49
O	
OPMODE	67
P	
Power Save Modes	18
Protocol	18
PSMODE	67
R	
RDY	81
REC	69
Receive Only Mode	45
Reception	44
Re-initialisation	28

APPENDIX A INDEX

Remote Frame	18
REPS	69
RESET	23, 24
RSTAT	68
RXONLY_CH	17
S	
Shutdown	25
SJW	73
T	
TIF	51
Time Stamp	18
TQPRS	72
Transfer ID	57
TRQ	81
TSEG	73
TSEN	75
TSLOCK	75
TSEL	75
V	
VALID	68

APPENDIX A INDEX

