# A CAN BUS PROTOCOL CONTROLLER MACRO

**Clive Tilbury**
**Senior Product Marketing Engineer**
**Fujitsu Mikroelektronik GmbH**

## 1. Introduction

The CAN Bus has already established itself as popular and reliable communications protocol in the European Automotive market and whilst well on its way to becoming a de facto standard there, is also finding much interest in Industrial applications. As part of Fujitsu's ongoing commitment to these segments, a powerful and flexible CAN Bus protocol controller has been designed to be included in the library of peripheral function macros for the 16 bit F²MC16LX and 32 bit FR Series microcontrollers.

Now that that Fujitsu has established a Microcontroller Design Centre at its Headquarters near Frankfurt, with the Charter to produce devices for European customers, the next 12 months will see this macro incorporated into several new MCU series.

The Fujitsu CAN controller conforms to both version 2.0A, (11 bit identifier) and version 2.0B, (extended 29 bit identifier). There are 16 message buffers that can be individually configured to receive all identifiers, (as per BASIC CAN), to receive only one defined identifier, (as per FULL CAN), or to accept groups of identifiers by use of a mask, (as per Extended CAN). By setting up more than one buffers to accept the same identifier, a multi layer buffer can be created which supports transmission and reception of multiple frames without additional microcontroller intervention. Up to two CAN macros can be implemented on one microcontroller.

## 2. Control and Status Register Structure

The microcontroller accesses the CAN Bus macro through nineteen control registers.

Twelve of these control the configuration of or report on the status of the sixteen message buffers in a bitwise manner, ie. the nth bit relating to the nth buffer. In this way the following functions can be managed:
Message Buffer Valid
Standard, (11 identifiers), or Extended (29 identifiers) Frame Format selection
Transmission Request, Cancel, Interrupt Enable and Complete
Remote Transmission Request
Transmit immediately or only after Remote Frame Request Received
Remote Frame Request Received (in buffer n)
Reception Complete, Overrun, Interrupt Enable

The main Control Status Register for the key functions of Transmit, Receive and Node Status. It is possible to enable an interrupt at any transition of Node Status between Error Active, Warning, Error Passive and Bus Off. There are two interrupt vectors for the

macro in the microcontroller – one for receive and one for both transmit and node status change.The Last Event Indicator Register shows whether this was Transmit or Receive Complete, with message buffer number or a Node Status transition. A special feature of the Fujitsu CAN controller is the ability to read the number of errors that have occurred via the Receive and Transmit Error Counter registers. One further register controls the bit timing parameters, (bit rate, sample point and resynchronisation jump width). The bit rate is adjustable from 10Kbit/sec to 2Mbit/sec from a CPU clock of 16MHz.

## 3. Acceptance Filtering

For each individual message buffer, it is possible to select one of four acceptance schemes. These are Full Bit Comparison, Full Bit Mask or to choose the compare / mask bit pattern according to the contents of one of two Acceptance Mask Registers, AMR0 and AMR1. (These registers are 29 bits long but use only the most significant eleven when using standard frame format.) This method provides the programmer with great flexibility in handling the message and buffer management.
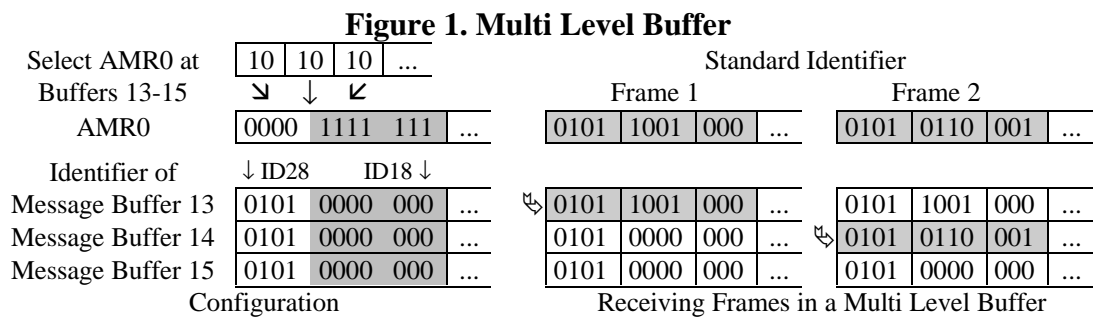
## 4. Message Buffers

There are 16 message buffers at fixed addresses. Each consists of an Identifier Register using the 11 or 29 most significant bits from 4 bytes, a 4 bit field, indicating the number of bytes in the frame within a Data Length Count Register and 8 bytes of Data Register, for the message itself. These are organised in functional blocks, ie. with all the identifiers together, etc.

The priority of a message buffer is higher with decreasing buffer number. Therefore if a message arrives, the identifier of which is accepted by two or more buffers, it is stored in the one with the lowest number. The same rule applies for transmitting. If a write is made to the Transmit Request Register that requires more than one buffer to send its data, then that with the lowest number will do so first.

A unique feature of the Fujitsu CAN Bus controller its ability to set up a multi-level buffer group. By setting buffers to use the same AMR, (or all selecting Full Bit Mask), and having the same compared identifier bits, it is possible to accumulate a sequence of messages. The group can be as large as there are buffers available but the block must be contiguous. More than one group is also permissable.The first frame will be stored in the highest priority buffer, the second in the next highest priority buffer and so on. The frames can be read in any order and any new frame arriving into the group buffer will be stored in the highest priority buffer available. In the event of the group becoming full, all subsequent frames received then overwrite the first buffer of the group and an overrun indicated. The benefit of this arrangement is that it is not so important that the microcontroller reads the message immediately and reduces the risk of messages being overwritten and lost. Also, in the case where the information being transmitted requires more than the maximum 8 bytes allowed per frame, then the controller can wait until all data is received before reading it. In the same way, larger messages can be transmitted in one operation. Because the registers and buffers are arranged by function and the group

must consist of contiguous buffers, it is then possible to copy this larger amount of data as a single block.

Figure 1 demonstrates the Multi Level Buffer concept diagrammatically. The group is set up with message buffers 13, 14 and 15 selecting AMR0 and having the same compared Identifier bits. The first frame is stored in buffer 13 and the second in buffer 14.

**Figure 1. Multi Level Buffer**

| | | | | | Standard Identifier | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Select AMR0 at | 10 | 10 | 10 | ... | | | | | | |
| Buffers 13-15 | ↘ | ↓ | ↙ | | Frame 1 | | | Frame 2 | | |
| AMR0 | 0000 | 1111 | 111 | ... | 0101 | 1001 | 000 | ... | 0101 | 0110 | 001 | ... |

| | ↓ ID28 | | ID18 ↓ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Identifier of | | | | | | | | | | | |
| Message Buffer 13 | 0101 | 0000 | 000 | ... | ↰ 0101 | 1001 | 000 | ... | 0101 | 1001 | 000 | ... |
| Message Buffer 14 | 0101 | 0000 | 000 | ... | 0101 | 0000 | 000 | ... | ↰ 0101 | 0110 | 001 | ... |
| Message Buffer 15 | 0101 | 0000 | 000 | ... | 0101 | 0000 | 000 | ... | 0101 | 0000 | 000 | ... |

Configuration                    Receiving Frames in a Multi Level Buffer

## 5. Reception Flow

Figure 2 shows the flow for the Reception operation from the viewpoint of the CAN Macro state machine.

From detection of the Start of Frame, the filtering and buffer selection is performed as described in sections 3 and 4 above. If the Reception Complete bit is still set, then an overrun is deemed to have occurred and the Receive Overrun bit is set, the data overwriting the existing contents. The received message is then examined to determine whether it is a Data Frame, (in which case the RRTR bit is set to 0), or a Remote Frame Request, (in which case it is set to 1). If it is a Data Frame, or if it is a Remote Frame and the Remote Transmission Request bit has been set by software, the Transmit Request bit is cleared to cancel pending transmissions from the designated buffer. Following this, the Reception Complete bit is set high and if enabled, a Reception Interrupt occurs.

## 6. Transmission Flow

Figure 3 shows the flow for the Transmission operation from the viewpoint of the CAN Macro state machine.

When 1 is written to the Transmit request bit of buffer x, the corresponding Transmission Complete bit is cleared to 0. The transmission procedure begins immediately unless the RFTWx bit has been set, denoting that it should wait until a Remote Frame has been received. If this bit is set, then the RRTRx bit is examined to see if the condition has been fulfilled. Providing Transmission Cancel is not set and TREQx is not cleared, (possible by receiving a message), the buffer remains primed until it is fulfilled. A check is made to see if more than one buffer has been initiated, (in which case, the one with the lowest number goes first), and also that bus state is idle. Depending on the state of the TRTRx bit, either a Data Frame or a Remote Frame is sent. If transmission is successful, then

RRTRx, TREQx are cleared and the Transmission Complete is set high and if enabled, a Transmission Interrupt Occurs.

**7. Design and Validation**

The CAN Bus macro was designed at Fujitsu Microelectronics in Kawasaki, Japan. It was created in Verilog HDL and simulated with Verilog-XL. It was then synthesised to FLDL, (Fujitsu Logic Description Language) and FTDL, (Fujitsu Test Description Language) using Synopsis. This was then resimulated with the in-house tool LCADFE. In-house layout tools were used to create the GDSII output and post layout delay extraction for final resimulation with LCADFE.

A special observer model was also written in Verilog HDL to interpret all events on the system level CAN Bus simulation model and send messages to the screen. This was developed by a separate team of engineers, as a check against any misinterpretation of the CAN specification. The test vectors were written in C, based on the Bosch Test Pattern.

Below is an example of the messages from the CAN Observer:

CANOBS:End of INTERMISSION. Bus is now IDLE!
CANOBS:***********************************
CANOBS:*SOF detected. Hard Synchronization!*
CANOBS:***********************************
CANOBS:End of BASE ID
CANOBS:Standard Format ID=1000 1111 100, RTR=0
CANOBS:End of DLC. DLC=0000
CANOBS:This packet is a Remote Frame or Data Frame of DLC=0
CANOBS:No Data field expected
CANOBS:Final CRC result 001010100000001
CANOBS:CRC Received    001010100000001
CANOBS:CRC check O.K.!
CANOBS:End of CRC field
CANOBS:Signalling ACK!
CANOBS:Acknowledgement O.K.!
CANOBS:End of ACK field!
CANOBS:End of End of Frame

Since it is not very practical to run too many test frames on a simulation model of a CAN Bus macro, it was implemented in an FPGA device as a standalone CAN Bus controller. An evaluation board was produced with the controller as a peripheral to an MB90T678 MCU, a ROMless device in the F²MC16L family. This implementation is actually able to run at full speed. This not only provided faster checking of the functional performance but it was possible to reiterate the design very quickly once a bug was found. This was done six times during the debugging process. Also the board was able to provide several internal signals for analysis and triggering purposes which would not be available on a finished microcontroller. The board was also provided to a consultancy, expert in CAN

Bus and member of CiA, who supported the "real-life" validation work. This included testing the protocol behaviour, the tolerance to protocol errors on the bus, the effect of bit errors, arbitration at low and high bus loadings and interaction with CAN controllers from other manufacturers.

When a new device series is designed, the first version to be produced is the evaluation chip which in a 256PGA package is the one used in Fujitsu's MB2140A in circuit emulator. Once this is completed, a customer can begin their design work, whilst the OTP or for all new types, Flash ROM is introduced. The Mask ROM version follows later.

## 8. F²MC16L, F²MC16LX and F²MC16F Microcontrollers

Fujitsu launched the F²MC16L and F²MC16F families of 16 bit microcontrollers 2 years ago in Europe and customers have been impressed by their price, performance and richness of features. The 16L's are targetted at low cost, low power and low voltage applications whilst the 16F's are higher performance. The newest additions to the lineup are the 16LX's which are implemented in 0.5um CMOS technology. In total, there are now sixteen different series available.

The features list of the first general market device series with the CAN Bus macro is planned to include the following functional blocks:
F²MC16LX CPU Core
ROM 64KB Mask, 128KB Mask or 128KB Flash
RAM 2KB or 4KB
CAN Bus x 1 channel
UART x 2 channels
Serial I/O x 1 channel
A/D Converter 10 bit x 8 channels
External Interrupt x 8 channels
Input Capture x 4 channels
Output Compare x 4 channels
Programmable Pulse Generator x 8 channels
Reload Timer 16 bit x 1 channel

## 9. Conclusion

The Fujitsu CAN Bus macro is a powerful microcontroller peripheral block that offers designers greater control over and flexibility in implementing their applications than other solutions. At present it is included in two F²MC16LX devices, which have been especially designed for specific customers. As an off the shelf function, it can be incorporated into different shapes and sizes of Fujitsu's microcontrollers to be able to fit many application requirements and is at the forefront of Fujitsu's strategy for the MCU productline. 1998 will see the introduction of the first general market F²MC16LX devices incorporating the CAN controller and the option of Mask ROM or Flash memory.

**Fig.2 Reception Flowchart**

Detection of start of data frame
or remote frame (SOF)

Is any message buffer (x) passing
the acceptance filter found?

NO

YES

Is reception

NO

YES

Determine message buffer (x) where
received messages to be stored.

**lossary for control / status bit for
essage buffer x**
Cx Reception complete
OVRx Receive overrun
RTRx Remote frame request received
RTRx Remote transmission request
REQx Transmission request
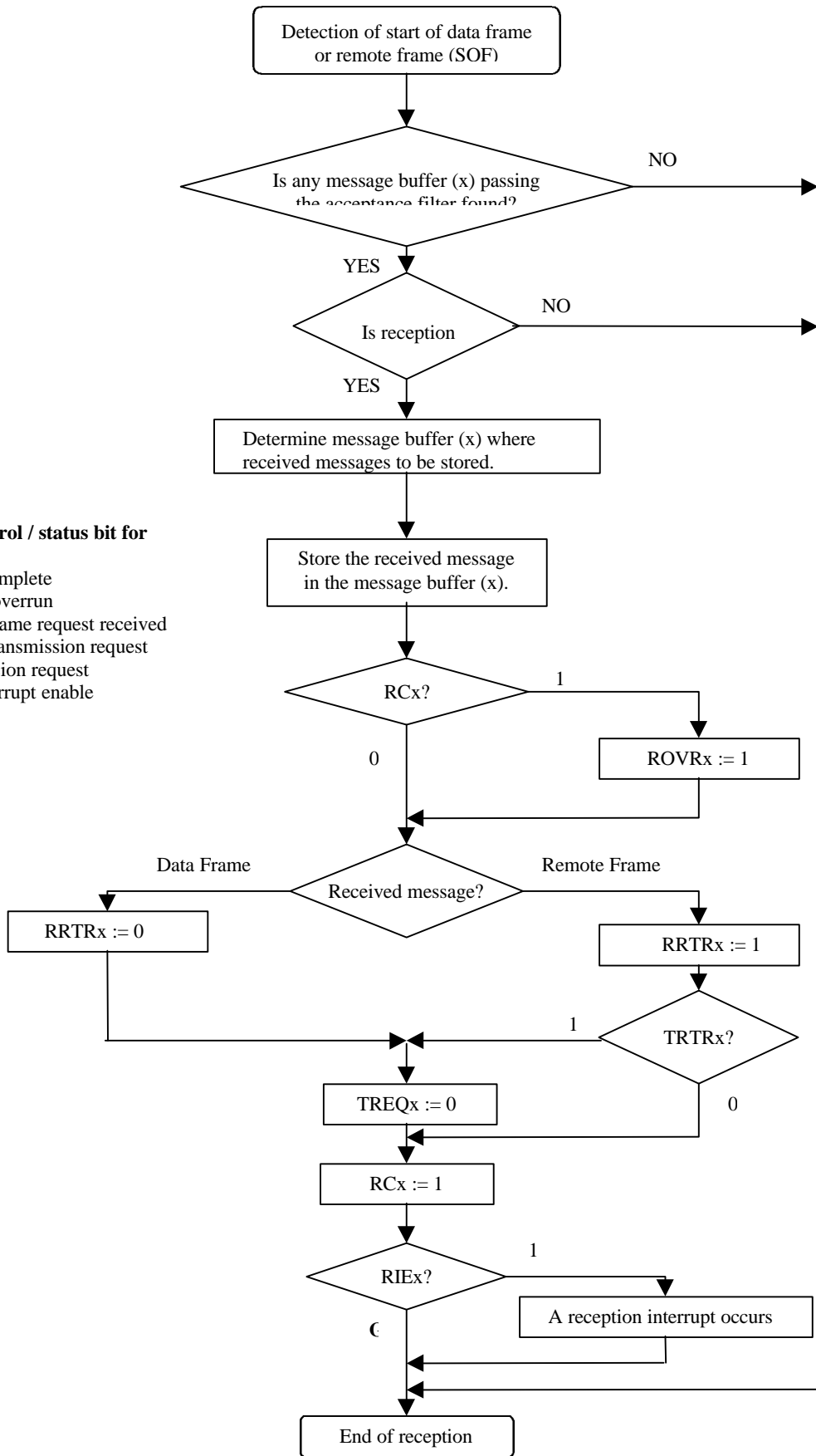IEx Receive interrupt enable

Store the received message
in the message buffer (x).

RCx?

1

0

ROVRx := 1

Data Frame

Received message?

Remote Frame

RRTRx := 0

RRTRx := 1
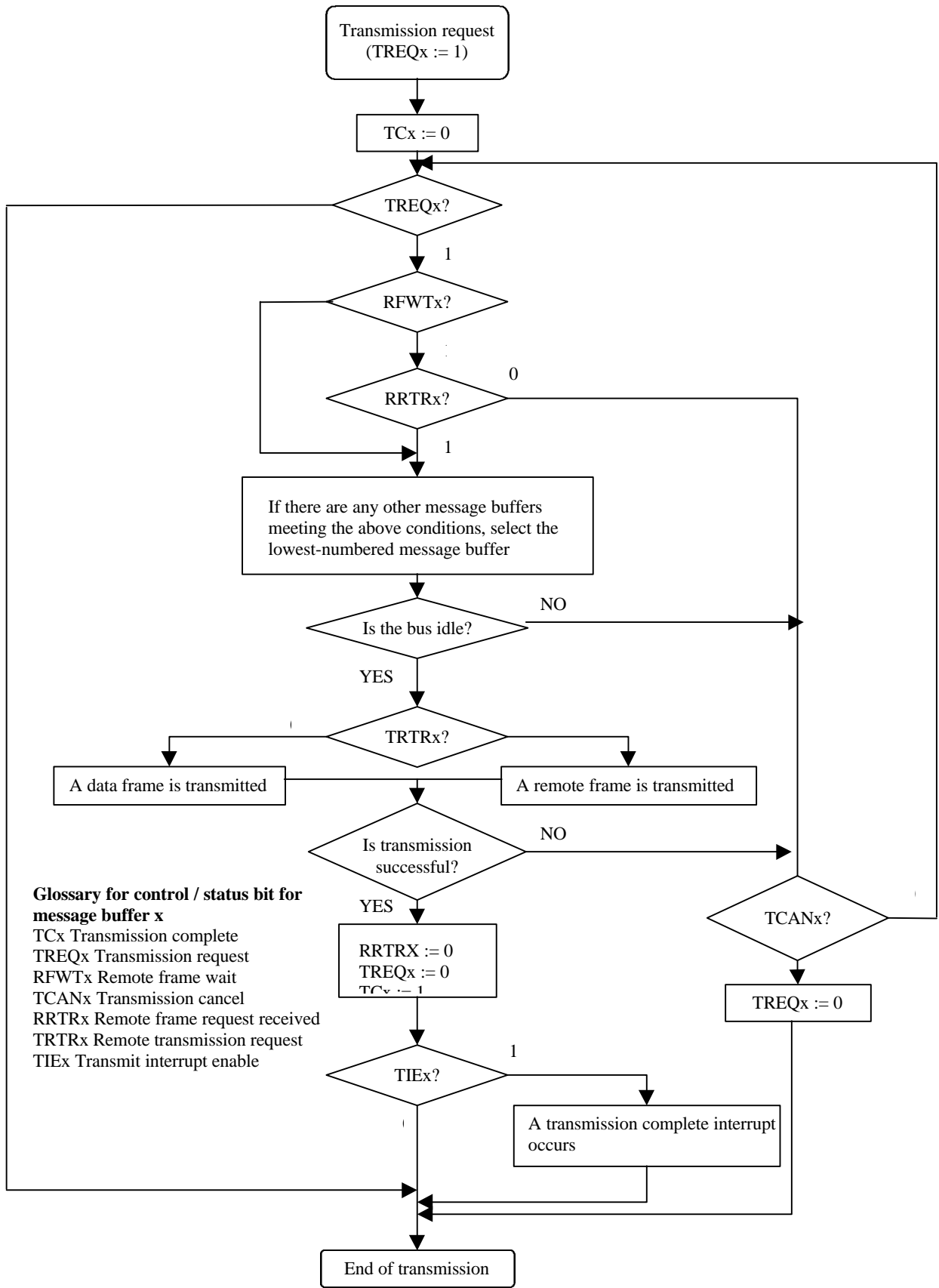
1

TRTRx?

0

TREQx := 0

RCx := 1

RIEx?

1

A reception interrupt occurs

End of reception

**Fig. 3 Transmission Flowchart**