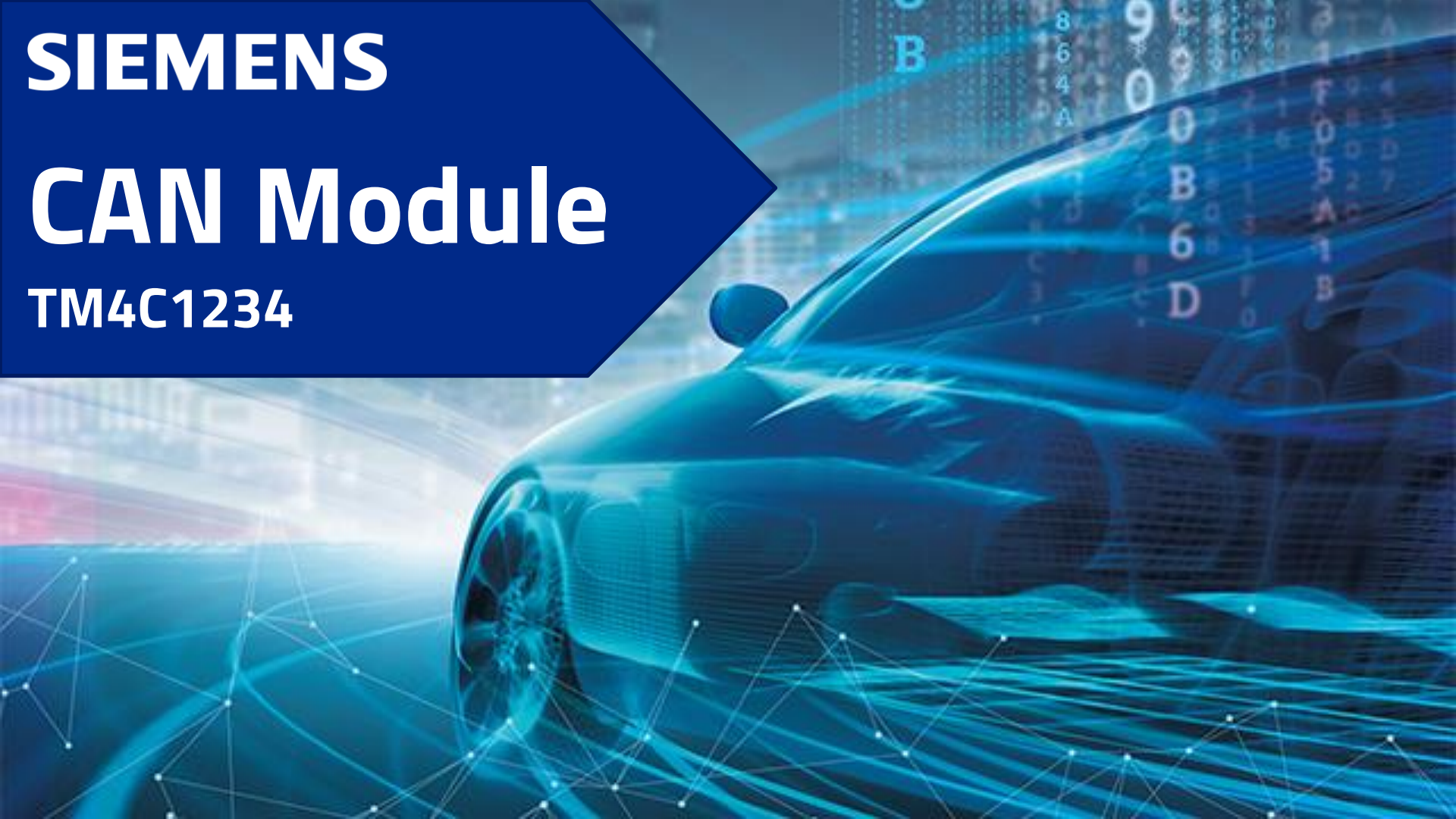
SIEMENS

CAN Module

TM4C1234



1. Features

- Supports 2 CAN modules with speed up to 1MB per second
- 32 Message Objects
- Programmable loopback mood for self-testing operation
- Storage mode for message objects to be used as FIFO
- External CAN support through CANnTx CANnRx signals
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- CAN protocol version 2.0. Standard 11-bit identifier and Extended 29-bit identifier

2.
Signal Description

CAN Module 0

Pin Name	Pin Assignment	Description
CAN0Rx	PF0 PB4 PE4	▪ CAN module 0 receive
CAN0Tx	PF3 PB5 PE5	▪ CAN module 0 transmit

CAN Module 1

Pin Name	Pin Assignment	Description
CAN0Rx	PA0	▪ CAN module 1 receive
CAN0Tx	PA1	▪ CAN module 1 transmit

3. Initialization

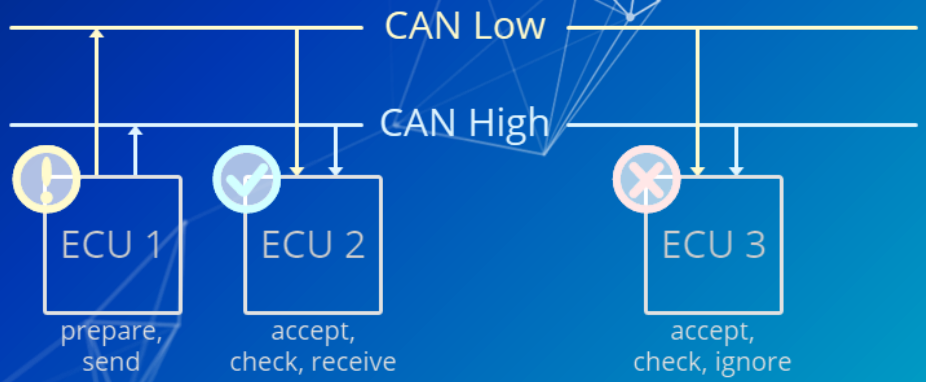
1. Enable clock for controller in RCGC0 register
   i. Set bit 24 for CAN0
   ii. Set bit 25 for CAN1
2. Enable clock for appropriate GPIO used in RCGC2 register
   - For CAN0
     - Enable Port F, B or E
   - For CAN1
     - Enable Port A
3. Set GPIOAFSEL
   - Set bits from 0 – 8 based on selected pins

1. Set GPIOPCTL
   - Values in p.1351 Table 23-5
2. Set INIT bit in CANCTL
   1. Bus operations stops
   2. CANnTx stays at high (recessive)
3. Set CANBIT register
   - Register details in p.1076
4. Clear INIT bit in CANCTL
   - Enable bus operations

4. Transmit

Transmitting Message Objects

- If CAN module is ready for loading, and if a data transfer is not occurring, the valid message object with the highest priority loaded and the transmission is started.
- After a successful transmission, the TXRQSTbit in the CANTXRQn register is cleared. If the CAN controller is configured to interrupt, the INTPNDbit in the CANIFnMCTL register is set after a successful transmission.
- If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.
- The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers.



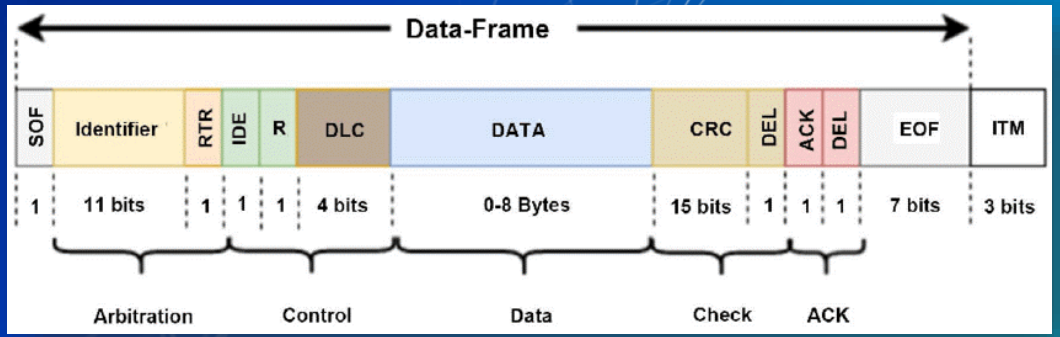
5.
Receive

Receiving Message Objects

- When the arbitration and control of an incoming message is completely shifted into the CAN controller, the message handling starts scanning the message RAM for a matching valid message object.
- To scan the message RAM for a matching message object, the controller uses the acceptance filtering. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

Receiving a Data Frame

- The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLCbits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used.
- The NEWDATbit of the CANIFnMCTL register is set to indicate that new data has been received.
- The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages.



Receiving a Remote Frame

- A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered

CANIFnMCTL	Description
DIR= 1 RMTEN= 1 UMASK= 1 or 0	At the reception of a remote frame, the TXRQSTbit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
DIR= 1 RMTEN= 0 UMASK= 0	At the reception of a remote frame, the TXRQSTbit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and nothing indicates that the remote frame ever happened.
DIR= 1 RMTEN= 0 UMASK= 1	At the reception of a remote frame, the TXRQSTbit of this message object is cleared. The arbitration and control field is stored into the message object in the message RAM, and the NEWDATbit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame.

6.
FIFO

- The reason behind having a Rx FIFO is simply to allow your program some time to do other things, while there are incoming messages. When you inspect the FIFO, it is often best to do so until you've read all messages and emptied the FIFO.
- The more modern/advanced CAN controllers uses something called "mailboxes" where specific CAN identifiers you are interested in end up in their own dedicated "mailbox" message buffer. The setup then is to have dedicated mailbox for all high priority messages you expect, and leave the RX FIFO for low priority stuff and/or messages that you aren't interested in.

- Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number
- The message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages
- The order of the received messages in the FIFO is not guaranteed

7. Interrupts

Handling of Interrupts

- If several interrupts are pending, the CAN Interrupt (CANINT) register points to the pending interrupt with the highest priority, disregarding their chronological order.

- An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's INTPNDbit at the same time by setting the CLRINTPND bit in the CANIFnCMSK register. Once the INTPNDbit has been cleared, the CANINT register contains the message number for the next message object with a pending interrupt.

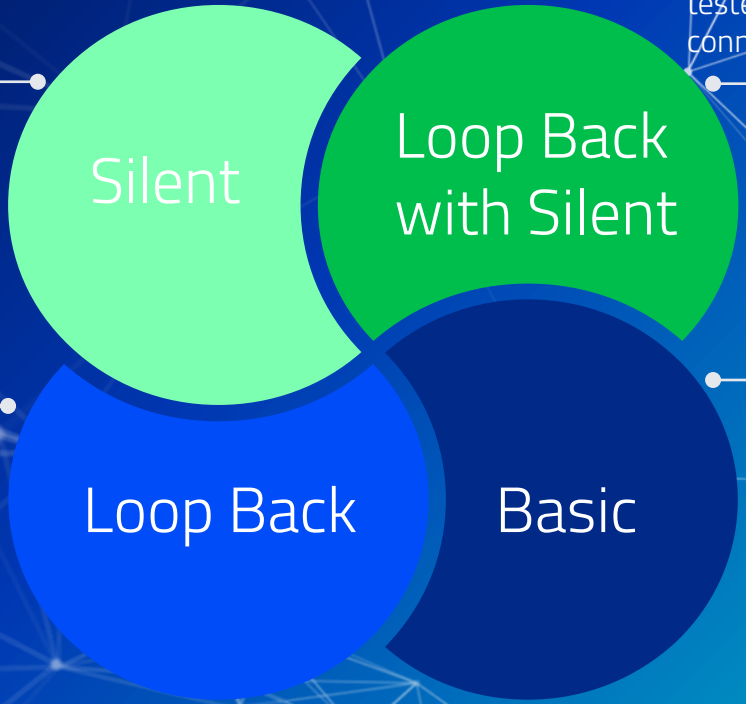
Interrupt Types

- Message Received
- Message Transmitted
- Warning Status
- Buss off status
- Error Passive status
- Errors
  - Stuff Error/Format Error/ACK Error/Bit Error/CRC Error/No Event

8.
Test Modes

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals



Loopback mode is useful for self-test functions.

Basic Mode allows the CAN Controller to be operated without the Message RAM.