

# EB421/D

## The Motorola MCAN Module

Kenneth Terry,  
Motorola East Kilbride

### Introduction

The Controller Area Network was developed by Robert Bosch GmbH as a serial communications protocol for use in automotive applications. It has been optimised for operation in interrupt-driven, real-time environments and meets the requirements of the following:

- medium speed applications (SAE class B), such as information transfer between body electronics modules
- high speed applications (SAE class C), which include data transfers between automotive modules such as Antilock Braking (ABS), Engine Management and Transmission Control Units.

In addition to its key role within the automotive multiplexing market, CAN is also finding widespread use in industrial control applications. The protocol is ideal for transfer of information between plant floor devices, such as actuators, sensors, switches and smart motor controllers within an industrial control system.

The wide range of CAN applications already in existence, and the number of products currently available from a range of major semiconductor manufacturers, give some indication of the key role that the protocol will have in both automotive and industrial markets. The wide range of applications will provide a market for an extensive range of products in the future. Motorola now produces several MC68HC05-based microcontrollers with on-board CAN modules.

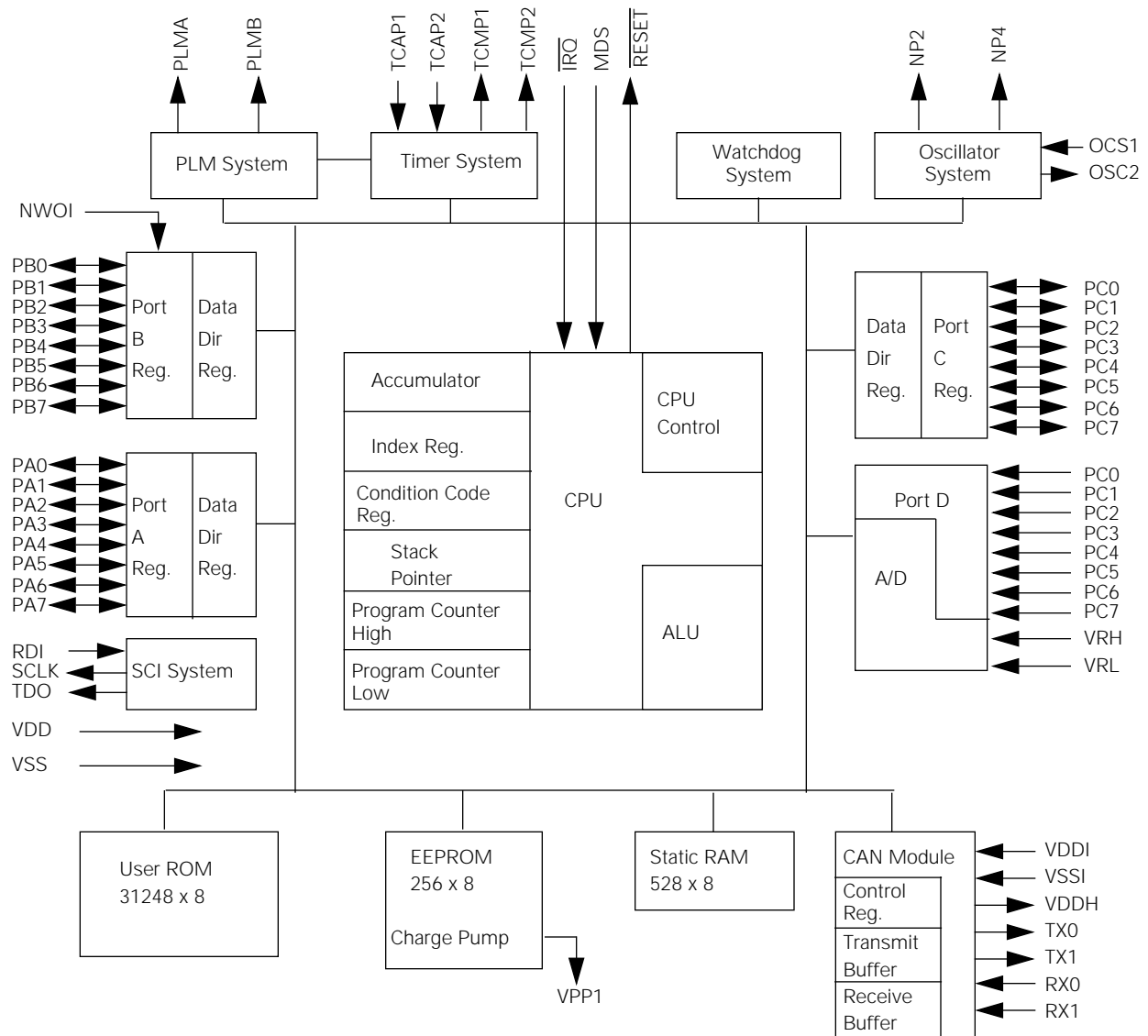
### MCAN Overview

The Motorola CAN (MCAN) module includes all the hardware blocks necessary to support the CAN protocol specification – Rev. 2.0 Part A.

The MCAN module has been integrated into a range of Motorola's MC68HC05 microcontrollers which includes the MC68HC05X4, X16 and X32.

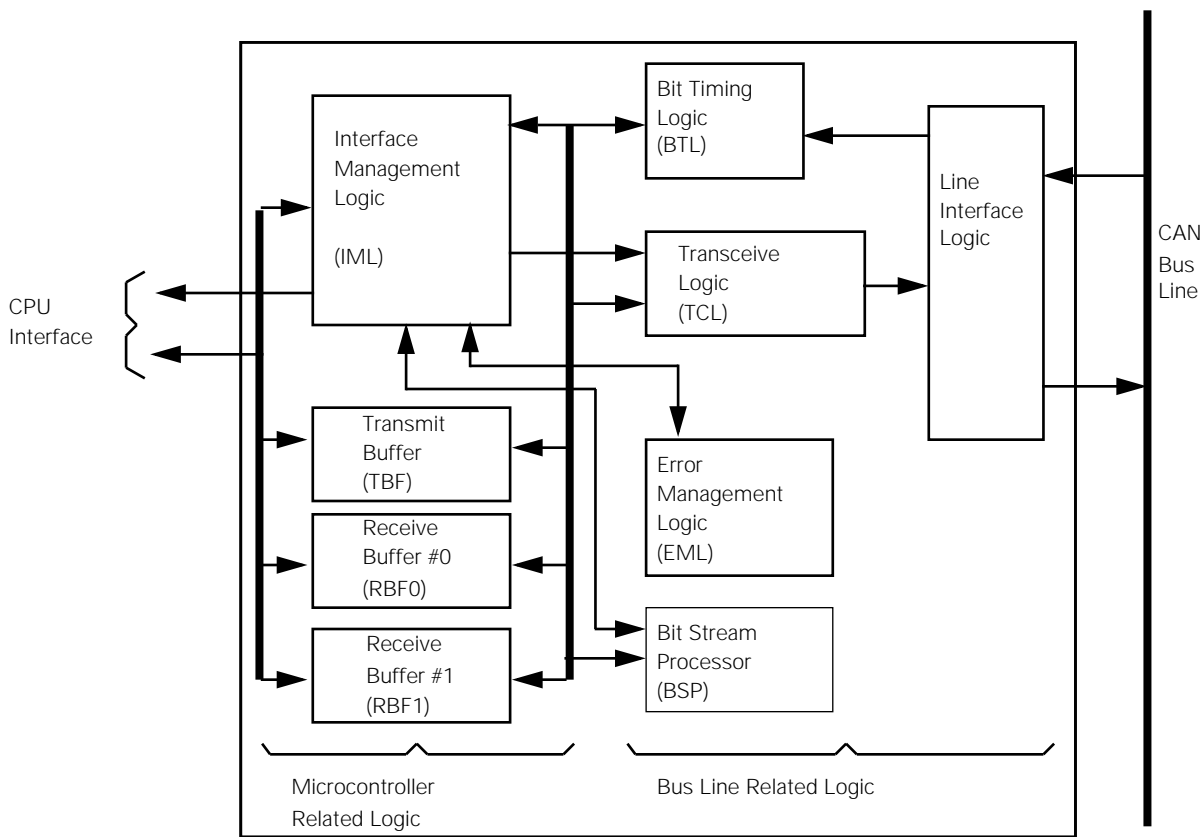
The MC68HC05X4 comes with 4 K bytes of ROM, 176 bytes RAM, a 16-bit timer and a separate core timer to provide watchdog and real-time interrupt facilities. The HC05X16 has been developed from the HC05B6 family and incorporates 15118 bytes of ROM, 352 bytes of RAM, 256 bytes of EEPROM, a free-running 16-bit timer with two input capture and two output compare channels, an 8-channel A/D converter, a Serial Communications Interface and two Pulse Length Modulation channels. The HC05X32 is the newest addition to the HC05X series. It has all the features of the HC05X16, but the memory has been extended to provide 31248 bytes of ROM and 528 bytes of RAM. EPROM versions of the HC05X4 and HC05X32 are also available.

Figure 1 shows a block diagram of the HC05X32.



**Figure 1** MC68HC05X32 Block Diagram

The MCAN module structure is identical on all the HC05X family variants. [Figure 2](#) shows a block diagram for the module.

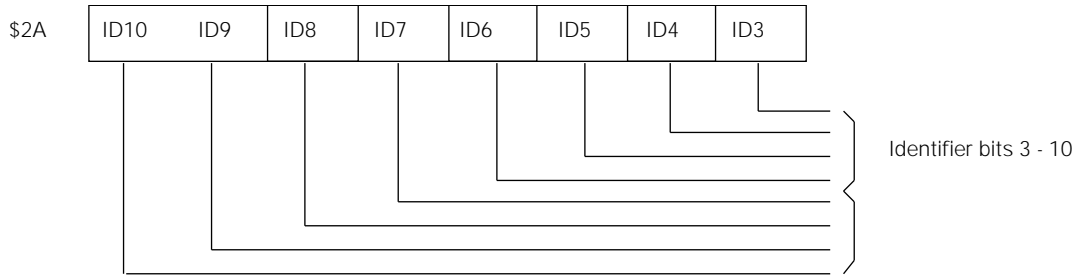


**Figure 2** MCAN Block Diagram

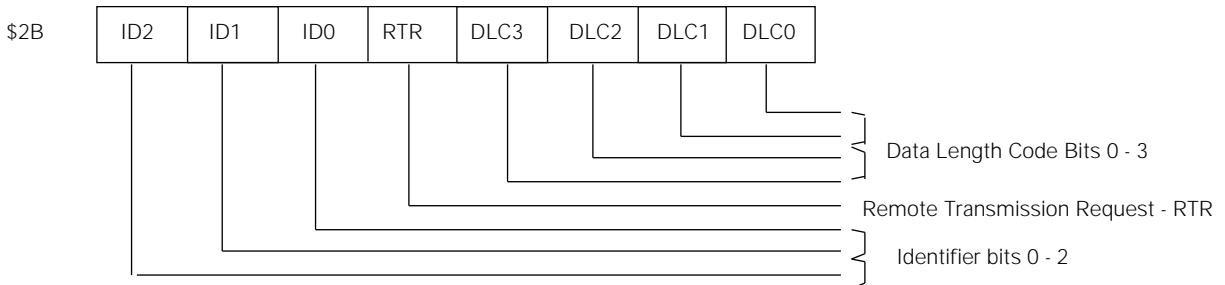
The Transmit Buffer acts as an interface between the CPU and the Bit Stream Processor (BSP). It is ten bytes long and is capable of storing a complete message and identifier. Figure 3 shows the contents of the Transmit Buffer Identifier register (TBI), the Remote Transmission Request and Data Length Code Register (TRTDL) and the Transmit Data Segment Registers (TDS) 1 – 8. The CPU may write to the Transmit Buffer when there is no message currently being transmitted or awaiting transmission. After a message has been stored in the Transmit Buffer and the CPU has requested a transmission (by setting Transmission Request in the MCAN Command Register), the buffer is locked and the BSP has exclusive access to it. After message transmission has completed, the buffer is unlocked allowing it to be accessed again by the CPU.

There are two full Receive Buffers (each ten bytes in length) used to handle all incoming messages being passed from the BSP to the CPU. Each Receive Buffer is capable of storing an entire message and has a layout identical to that of the Transmit Buffer. They are arranged in a double buffered configuration, with each buffer occupying the same area of the memory map. When the first buffer is filled it can be read by the CPU as a second incoming message is being transferred by the BSP into the second buffer. After the CPU has read a message from the Receive Buffer it can release that buffer, making it available for storing the next incoming message and allowing the second buffer to be read once it contains a complete message.

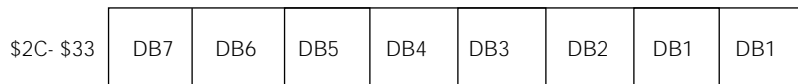
### Transmit Buffer Identifier Register (TBI)



### Remote Transmission Request and Data Length Code Register (TRTDL)

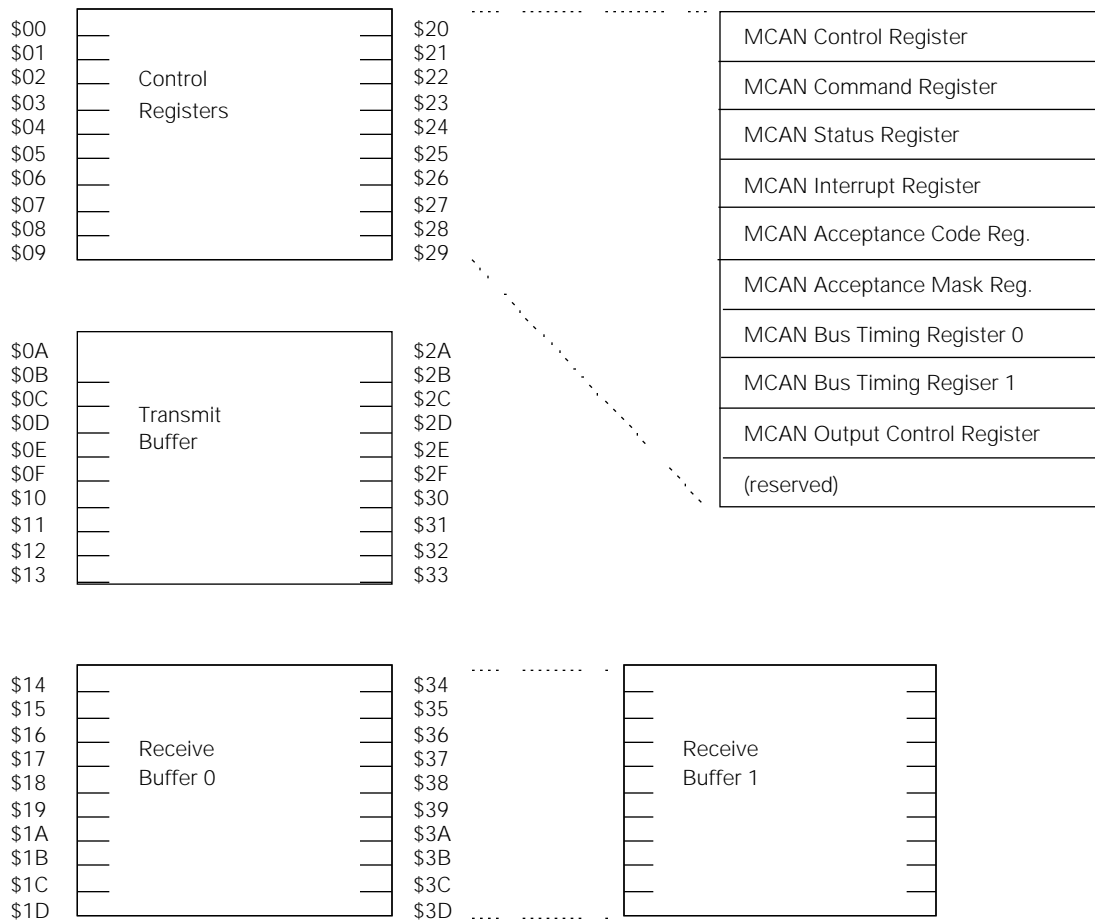


### Transmit Data Segment Registers (TDS) 1- 8



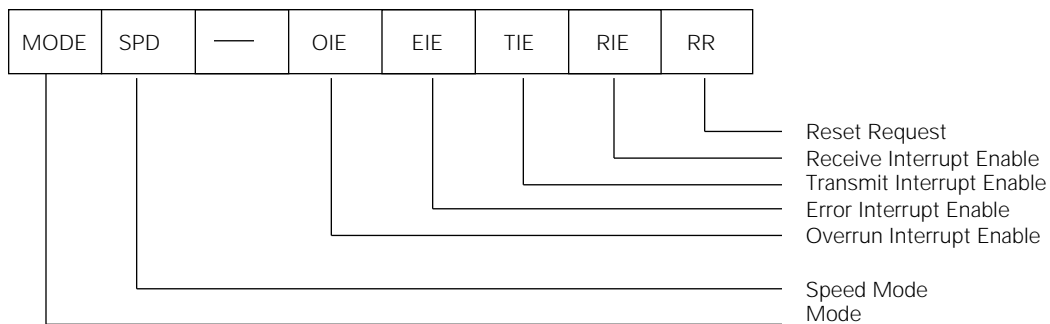
**Figure 3** Transmit Buffer Registers

Figure 4 shows the MCAN register addresses. In addition to the Transmit and Receive Buffers there are a further nine registers used to allow initialisation and control of the MCAN module and provide status information to the CPU.



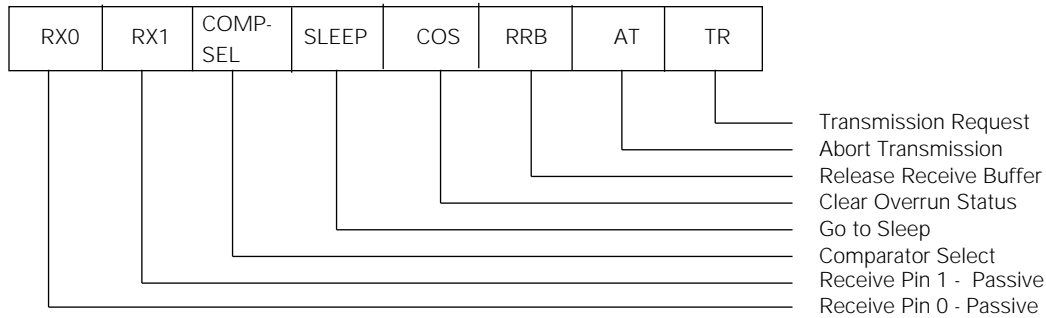
**Figure 4** MCAN Registers

**Control Register (CCNTRL)**



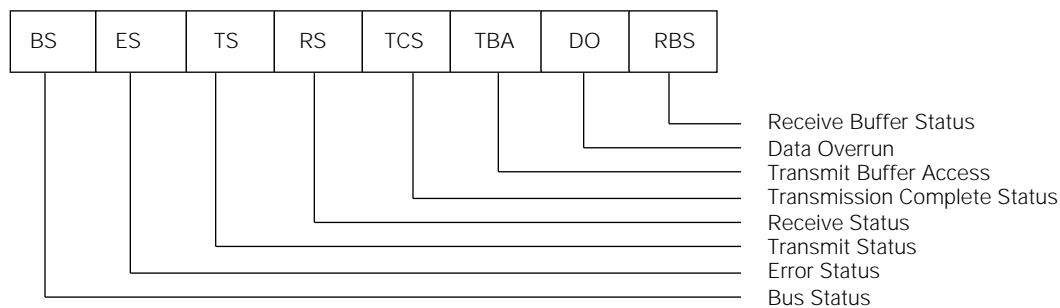
The Control Register provides the local mask bits for the CAN module interrupts. In addition it contains the Reset Request (RR) bit, which is set to disable the CAN module operation and allow access to the message filtering, bus timing and output control registers.

## Command Register (CCOM)



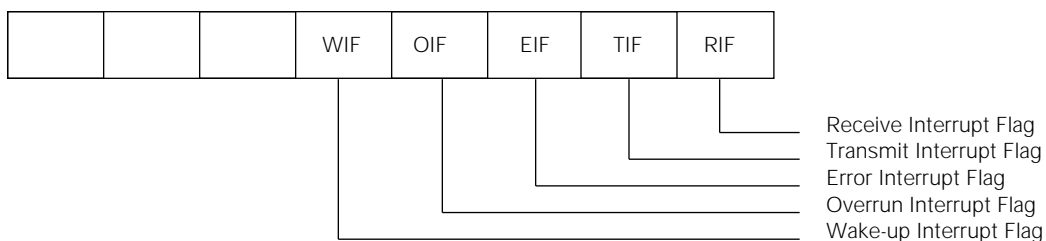
The Command Register is a write only register which contains the Release Receive Buffer (RRB) and Transmit Request (TR) bits. RX0, RX1 and COMPSEL are used to control the input comparator configuration, allowing it to operate correctly for single-wire and differential modes of operation.

## Status Register (CSTAT)



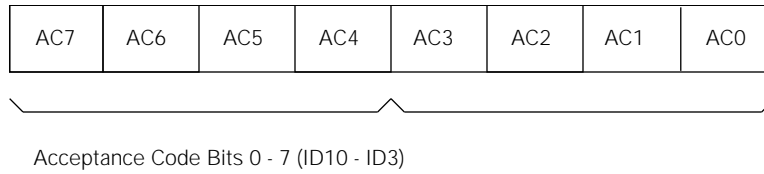
The Status Register provides information on a number of conditions which can occur in the CAN module. This includes information on the status of the last requested transmission, incoming messages and the availability of the Transmit Buffer for acceptance of new messages for transmission. It also provides a flag to indicate the module 'off bus' state and provides limited information on the status of the error counters.

## Interrupt Register (CINT)

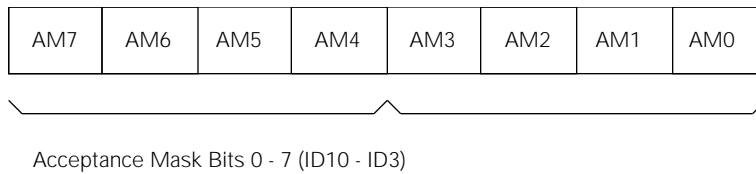


The MCAN module has only one interrupt vector assigned to it. The Interrupt Register can be read to determine the source of a CAN interrupt. The five interrupt sources include; Wake Up, Data Overrun ( a third message being received before either of the Receive Buffers have been released), Error (either read or write error counter reaching a pre-determined level), Transmission Complete and Receive Interrupt.

### Acceptance Code Register (CACC)

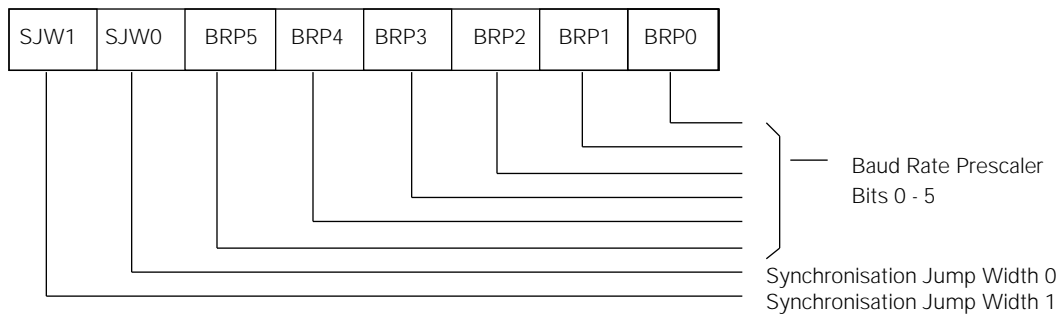


### Acceptance Mask Register (CACM)

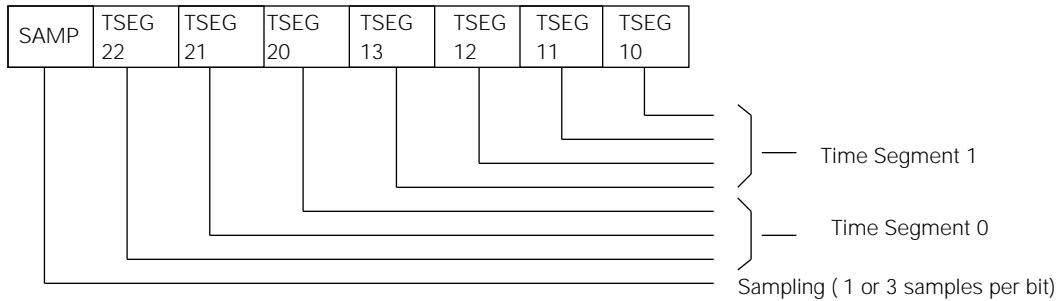


The Acceptance Code and Acceptance Mask Registers are used to provide limited message filtering on the eight most significant bits of the identifier. The Acceptance Mask Register defines which bits of the the Acceptance Code Register are compared with the identifier of the incoming message. If the MCAN module receives a message with an identifier which does not meet its acceptance criterion, then it will respond by transmitting a dominant bit in the correct position of the ACK field, but will not transfer the message to the receive buffers or indicate to the CPU that a new message has been received.

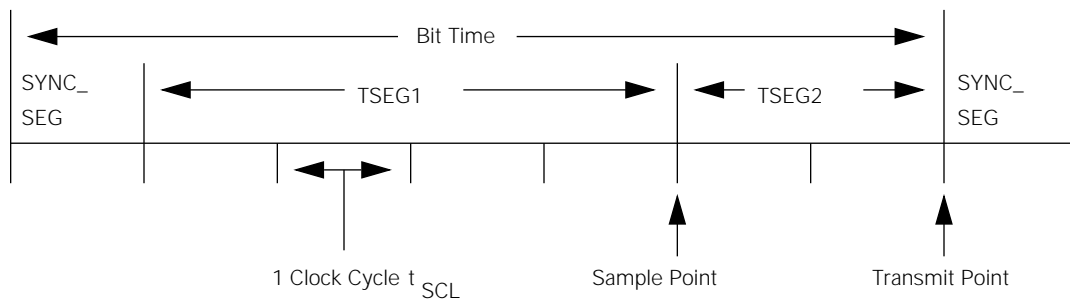
### Bus Timing Register 0 (CBT0)



## Bus Timing Register 1 (CBT1)

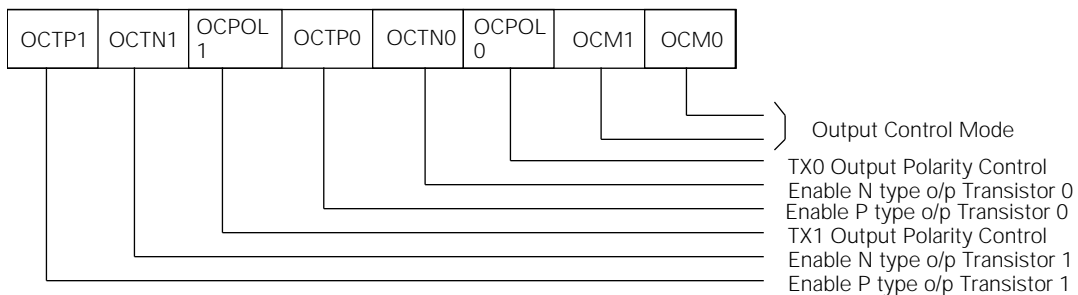


The Bus Timing Registers are used to select a suitable baud rate prescaler value to provide an appropriate  $t_{SCL}$  value, which is then used to derive the bit time and position of the sample point within the bit. Figure 5 shows the components of the CAN bit time. The Bus Timing Registers allow the two values, TSEG1 and TSEG2, to be defined. TSEG1 is the sum of PHASE\_SEG1 and the PROP\_SEG. TSEG2 is equal to PHASE\_SEG2. The Bus Timing Registers also define the size of the RESYNCHRONIZATION JUMP WIDTH. This is the amount by which a bit can be lengthened or shortened during the resynchronisation process.



**Figure 5** Segments within the Bit Time

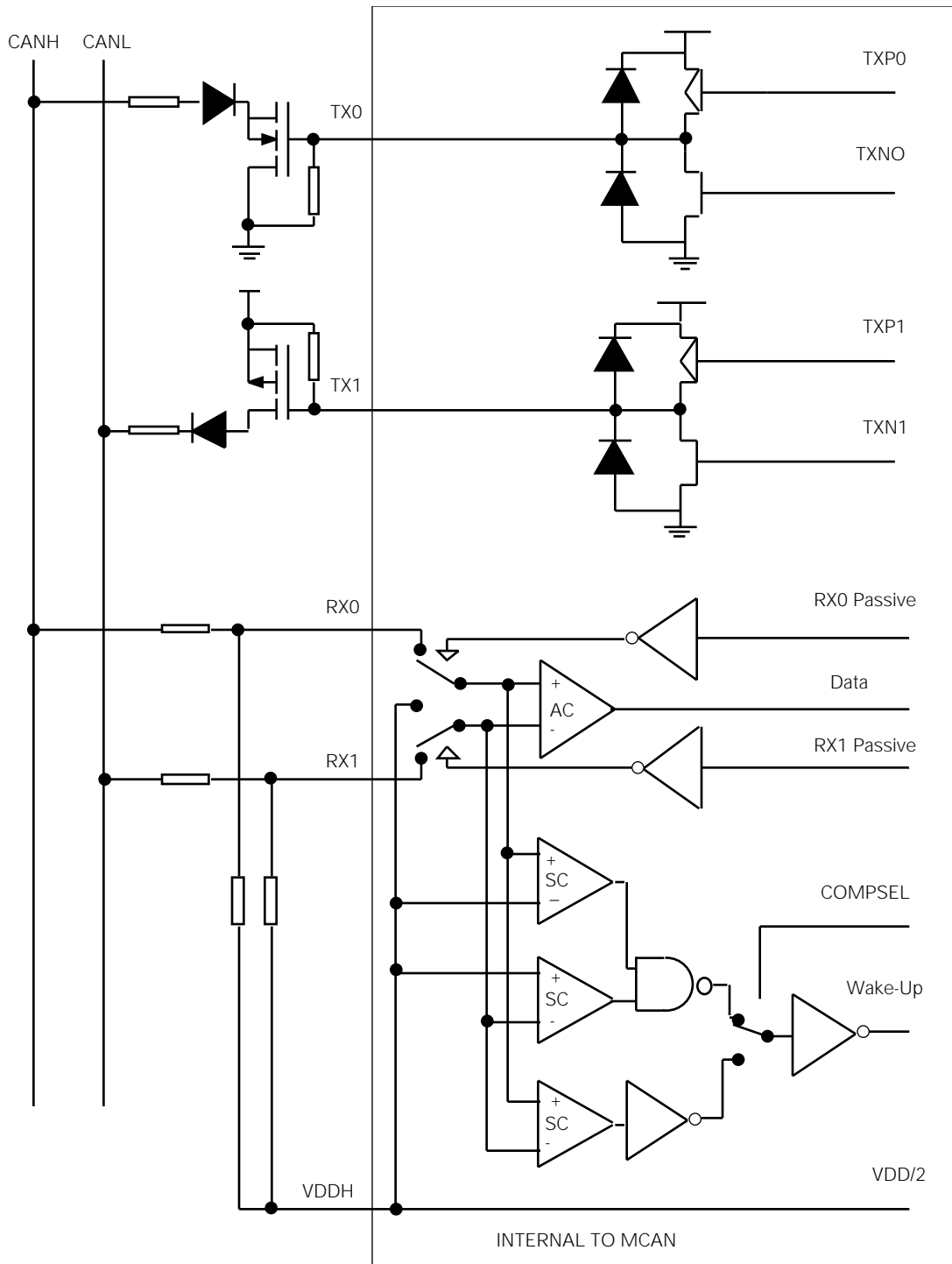
## Output Control Register (COCNTRL)



The Output Control Register (COCNTRL) is used to determine the configuration of the output drivers on the CAN transmit pins. The Output Control Mode bits allow normal differential operation, bi-phase operation or a special test mode to be selected. Figure 6 illustrates the internal logic of the MCAN driver and comparator logic and shows what a typical CAN bus physical interface might look like. The output drivers can be selected for pull up, pull down or push pull operation by selectively enabling or disabling the P type and N type transistors in the output driver circuits via the OCTN0/1 and OCTP0/1 bits. The COCNTRL



register also allows the data output to be inverted if required. The most usual configuration is for complementary levels to be transmitted on the TX0 and TX1 pins (two-wire differential operation). The bus termination network should ensure that the bus reverts to its recessive state when the driver transistors are switched off.



**Figure 6** Typical Physical Interface between MCAN Module and CAN Bus

Both the output drivers and the input comparators can be configured for single wire operation in the event of a bus fault occurring. Either input to the comparator can be disconnected from the device pin and connected to the VDD/2 signal to provide a mid-level reference against which the other input can be compared. The VDDH signal can be supplied either internally from the device or from an external source.

The MCAN module also incorporates a sleep mode. This is entered when the SLEEP bit in the Command register is set and there is no activity on the CAN bus. During sleep mode the active comparator is disabled to reduce power consumption. The low power sleep mode comparators will detect CAN bus activity and bring the module out of sleep mode. Because of the reduced accuracy of the sleep mode comparators, the CAN frame that reactivates the module from sleep mode will not be stored.

## MCAN Software Handling

The MCAN module with its simple transmit/receive buffer structure is required to handle messages on a message by message basis. This means that for each message transmitted or received via the CAN bus, CPU intervention is required. The following routines, written for the HC05X4, provide a simple illustration of how the CAN module is initialised and controlled by the CPU.

The initialisation routine INIT configures the bus timing registers to provide a 10 µs bit time comprising 10 tSCL segments, derived from a 4 MHz clock input. The output drivers are set up for complementary push pull operation to switch the driver transistors on the bus illustrated above. The Acceptance Mask register has been set up so that all messages are received. The transmit routine checks the state of the Transmit Buffer and, if necessary, waits for it to become available before transferring a message (up to 10 bytes in length) from RAM (TXSTART - TXSTART+9) to the Transmit Buffer registers. The Data Length Code (DLC) portion of the message is checked to determine the number of bytes to be transferred. The Transmit Request bit in the Command Register is set to initiate message transmission. The Receive routine should be called after an incoming message is detected (either through an interrupt or by polling). The DLC is checked and the appropriate number of data bytes are transferred from the Receive Buffer before it is released by setting the Release Receive Buffer bit in the Command register.

```

*****
*           HC05X4 CAN Initialisation, Transmit and Receive Examples
*****
*
*           Register Equates
PORTA      EQU      $0000
PORTB      EQU      $0004
*
*           CAN Registers
*
CANCTRL    EQU      $20           ;CAN Control Register
CANCOM     EQU      $21           ;CAN Command Register
CANSTAT    EQU      $22           ;CAN Status Register
CANINT     EQU      $23           ;CAN Interrupt Register
CANACC     EQU      $24           ;CAN Acceptance Code Register
CANACM     EQU      $25           ;CAN Acceptance Mask Register
CANBT0     EQU      $26           ;CAN Bus Timing Register 1
CANBT1     EQU      $27           ;CAN Bus Timing Register 2
CANOPC     EQU      $28           ;CAN O/P Control Register
CANTX      EQU      $2A           ;Start of CAN Tx Buffer
CANRX      EQU      $34           ;Start of CAN Rx Buffer
*
*           ORG      $0050
TXSTART    RMB      10;RAM Transmit Message Buffer
RXSTART    RMB      10;RAM Receive Message Buffer
*

```

```

*
*****
* CANINIT - Initialise CAN Module
*****
*
CANINIT   LDA    #$61
          STA    CANCTRL      ;Disable err. int.
*
          ;speed mode = slow, reset request high
          LDA    #$FF
          STA    CANACM      ;Accept all codes
          LDA    #$C1
          STA    CANBT0      ;Set up bus timing reg. 0
*
          ;(tSCL = 1 uS, tSJW = 4 x tSCL)
*
          ;(tBIT = 10uS with 4 MHz crystal)
          LDA    #$34
          STA    CANBT1      ;Set up bus timing reg. 1
*
          ;(tSEG1 = 5 x tSCL, tSEG2 = 4 x tSCL,
*
          ;tBIT = 10 x tSCL)
*
          LDA    #$DE
          STA    CANOPC      ;Set output control register
*
          ;(Normal mode 1, OCTP0 = OCTN0 = 1,
*
          ;OCTP1 = OCTN1 = 1, o/p drivers Tx0 and
*
          ;Tx1 push/pull - both NTRAN0 and PTRAN0
*
          ;enabled, OCPOL0 = 0, OCPOL1 = 1, Tx0 * *
;Tx'ed inverted, Tx1 tx'ed normally).
          LDA    #$60
          STA    CANCTRL      ;Set up CAN control reg.
*
          ;set reset request low
          RTS
*
*****
* TRANSMIT - Transfer CAN message from RAM to Trasmit Buffer and
*
          initiate message transmission.
*****
TRANSMIT  BRCLR  2,CANSTAT,*   ;Wait for Transmit Buffer Access set
*
          LDA    TXSTART+1
          AND    #$0F
          TAX
          ;X reg. contains no. of data bytes to * *
;Transmit
NXTBYTE   LDA    TXSTART+1,X   ;Fetch byte from RAM
          STA    CANTX+1,X     ;Store data byte in MCAN Transmit Buffer
          DECX
          BNE    NXTBYTE
          LDA    TXSTART+1
          STA    CANTX+1
          LDA    TXSTART
          STA    CANTX
*
          LDA    #$01
          STA    CANCOM      ;Request transmission
*
          RTS
*


```

```

*****
* RECEIVE - Transfer CAN message from Receive Buffer to RAM and
*          release Receive Buffer.
*****
*
RECEIVE   LDA    CANRX+1      ;Load RRTDL
          AND    #$0F        ;Acc contains no. of data bytes rx'ed
          TAX                      ;X reg. contains no. of data bytes
*                                     ;received
NXRBYTE   LDA    CANRX+1,X    ;Fetch byte from MCAN Receive Buffer
          STA    RXSTART+1,X  ;Store byte in RAM
          DECX
          BNE    NXRBYTE
          LDA    CANRX+1
          STA    RXSTART+1
          LDA    CANRX
          STA    RXSTART
*
          LDA    #$04
          STA    CANCOM      ;Release Receive Buffer
*
          RTS

```

All products are sold on Motorola's Terms & Conditions of Supply. In ordering a product covered by this document the Customer agrees to be bound by those Terms & Conditions and nothing contained in this document constitutes or forms part of a contract (with the exception of the contents of this Notice). A copy of Motorola's Terms & Conditions of Supply is available on request.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

The Customer should ensure that it has the most up to date version of the document by contacting its local Motorola office. This document supersedes any earlier documentation relating to the products referred to herein. The information contained in this document is current at the date of publication. It may subsequently be updated, revised or withdrawn.

Literature Distribution Centres:

EUROPE: Motorola Ltd., European Literature Centre, 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

ASIA PACIFIC: Motorola Semiconductors (H.K.) Ltd., Silicon Harbour Center, No. 2, Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

JAPAN: Nippon Motorola Ltd., 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.

USA: Motorola Literature Distribution, P.O. Box 20912, Phoenix, Arizona 85036.

