Application Report
SPRA500

Understanding the CAN Controller on the TMS320C24x DSP Controller

Claire Monnet *Digital Signal Processor Solutions*

Abstract

The Texas Instruments (TI™) TMS320F241, TMS320C241 and TMS320F243 digital signal processor (DSP) controllers contain an on-chip Control Area Network (CAN) module. This module is a FullCAN controller (Specification 2.0B). This application report describes the TMS320X241/3 CAN module. Software examples are included for different modes of operation of the on-chip CAN module along with an application example for controlling the speed of a three-phase induction motor using the CAN bus.

Contents

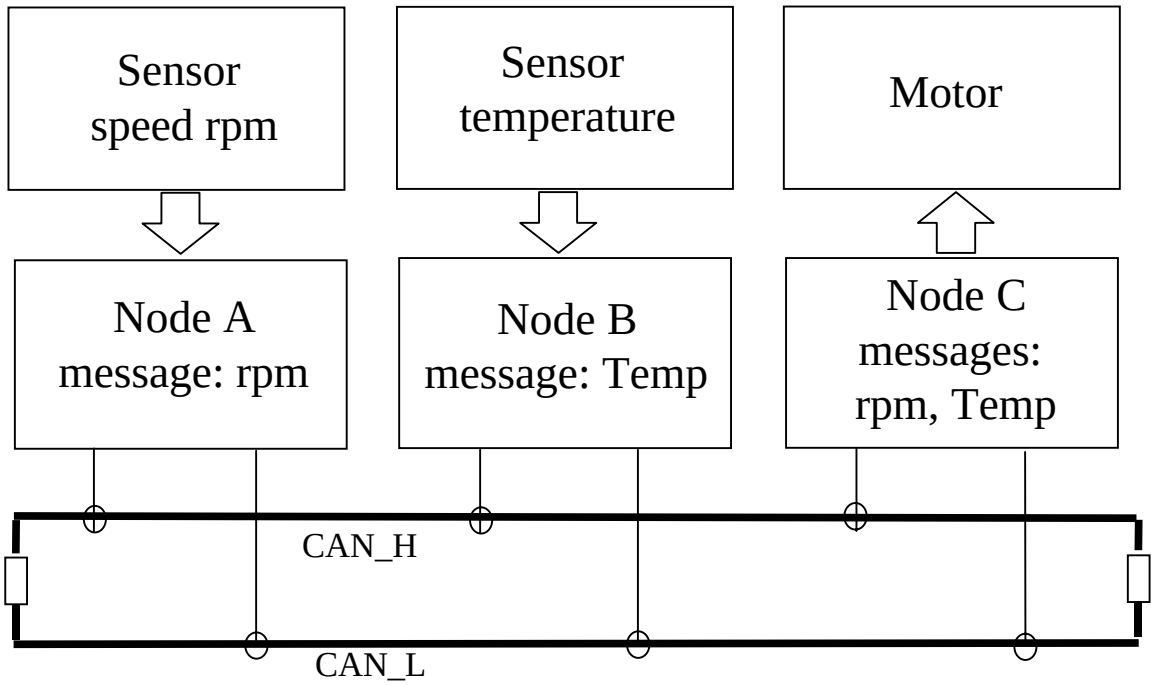
Figures

TMS320X241/TMS320F243 CAN Module Presentation

About CAN

The Controller Area Network (CAN) is a multi-master serial bus that uses broadcast to transmit to all CAN nodes. CAN protocol provides advantages over other communication protocols. For example, CAN protocol offers a very good price/performance ratio. It allows moving data with a fast transmission speed (up to 1 Mbit/s) and can be implemented in real-time systems. Furthermore, the data is very reliable and the error detection is sophisticated and robust. CAN is very flexible and offers hot swaps.

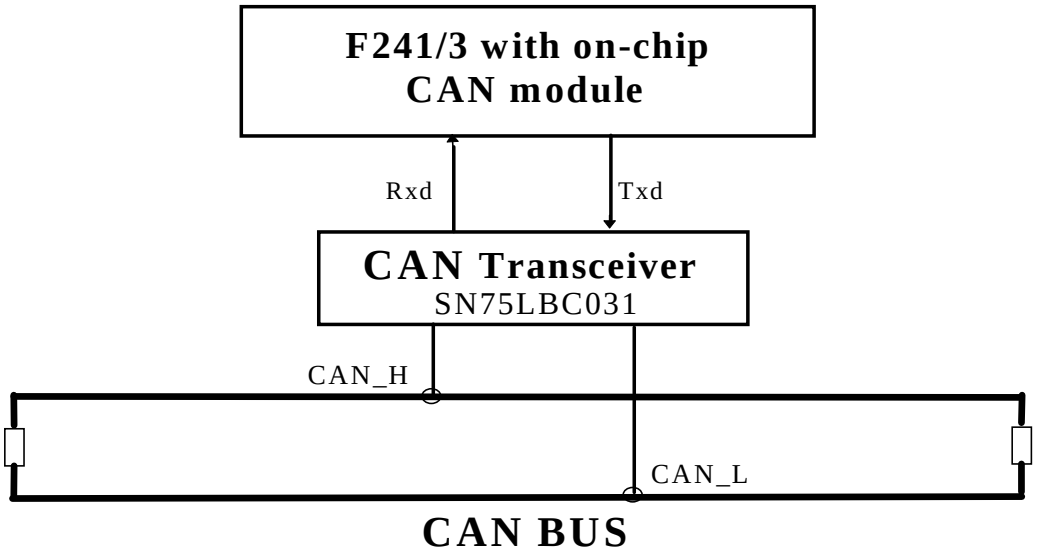
Figure 1. CAN Bus



CAN protocol does not address nodes with physical addresses but instead sends messages with an identifier that can be recognized by the different nodes. This identifier has two functions: it is used both for message filtering and for determining message priority. The ID determines if a transmitted message will be received by any particular CAN module and also determines the priority of the message when two or more nodes want to transmit at the same time.

The DSP controller needs a connection to a transceiver to be attached to the CAN Bus. The CAN bus is made with a twisted pair. The transmission rate depends on the bus length. For a bus smaller than 40 meters the transmission rate is up to 1 Mbit/s. The DSP controllers can be connected to the SN75LBC031, TPIC8233 and TPIC82501 TI CAN transceivers.

Figure 2. CAN Node



There are different types of CAN message frames:

- CAN data frame moves data (0 to 8 bytes) from a transmitter to receiver(s)
- CAN remote frame is used to request transmission of the data frame associated with the specified identifier.

The frames can be standard or extended. Standard contains an 11-bit ID and extended a 29-bit ID.

TMS320C241,TMS320F241 and TMS320F243 CAN Module

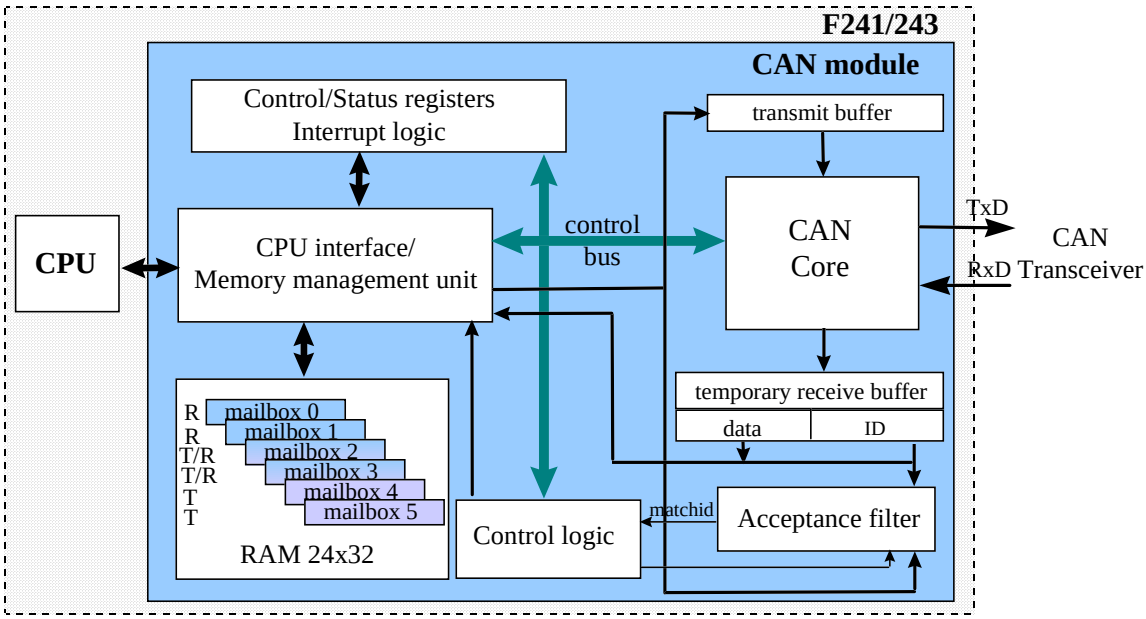
The TMS320x241 and TMS320F243 CAN module is a FullCAN Controller. It contains a message handler (for transmission and reception management and frames storage) and needs less CPU overhead than with the BasicCAN Controller. The specification is CAN 2.0B Active, meaning the module can send and accept standard (11-bit identifier) and extended frames (29-bit identifier)

The peripheral is 16 bit. The access to control/status registers and to CAN mailboxes are both 16 bit.

The controller contains six mailboxes for objects of 0 to 8 bytes data length:

- Two receive mailboxes (mailboxes 0 and 1)
- Two transmit mailboxes (mailboxes 4 and 5)
- Two configurable transmit/receive mailboxes (mailboxes 2 and 3)

Figure 3. TMS320X241/3 CAN Module Block Diagram

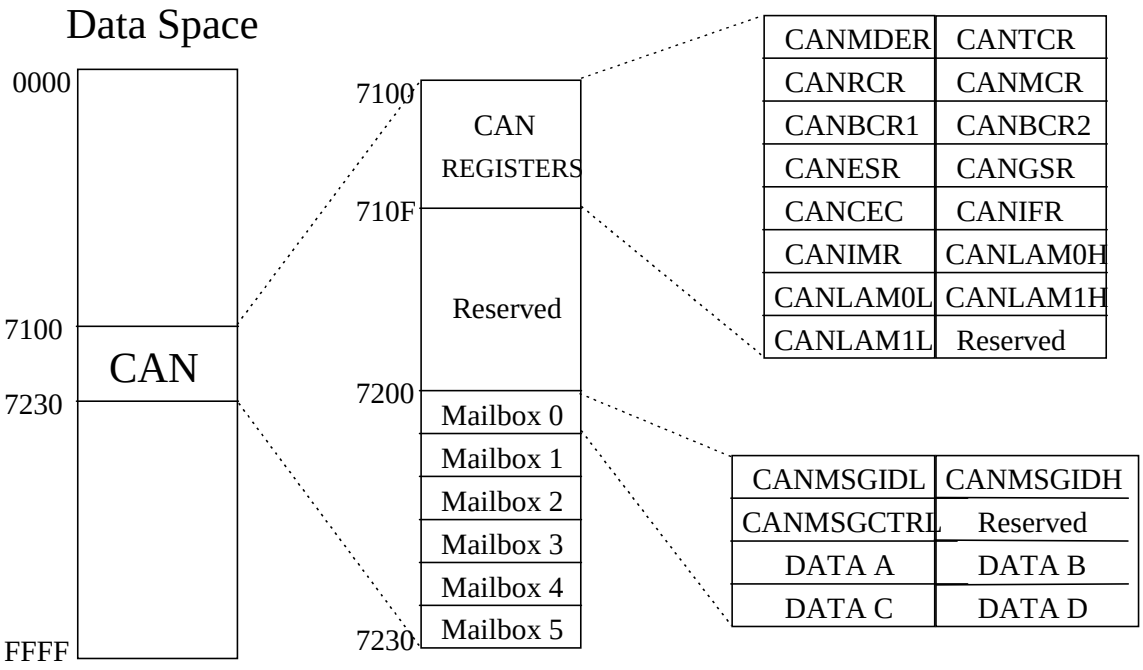


The CAN module contains 15 different 16-bit registers:

- Control registers
  - MDER: Mailbox Direction/Enable Register, to enable or disable the mailboxes and to configure mailboxes 2 and 3
  - TCR: Transmission Control Register used to transmit messages
  - RCR: Receive Control Register used to receive messages
  - MCR: Master Control Register, to change the bit timing configuration, to write in the CAN RAM or how to configure the chip in self test mode for instance
  - BCR1 and BCR2: Bit Configuration Register, to configure the bit timing
- Status Registers
  - ESR: Error Status Register, to display errors
  - GRS: Global Status Register
  - CEC: CAN Error Counter Register
- Interrupt Registers
  - IFR: Interrupt Flag Register
  - IMR: Interrupt Mask Register
- Local Acceptance Mask Register
  - LAM0H and LAM0L: local acceptance mask registers for the mailboxes 0 and 1
  - LAM1H and LAM1L: local acceptance mask registers for the mailboxes 2 and 3

These registers are located on the data memory from the address 0x7100h to 0x710Fh (see Figure 4).

Figure 4. TMS320X241/3 CAN Module Memory Space



The CAN module contains six mailboxes. Each mailbox is divided into several parts:

- MSGIDL and MSIDH contain the Identifier of the mailbox.
- MSGCTRL (Message control field) contains the length of the message (to transmit or to receive) and the RTR bit (Remote Transmission Request used to send remote frames).
- DATA_A to DATA_D contain the data. The data is divided into four words or into eight bytes.

Different Modes of Operation

Initialize CAN Module

To use the CAN module, the CAN registers and the CAN RAM must be initialized.

Bit Timing and Synchronization

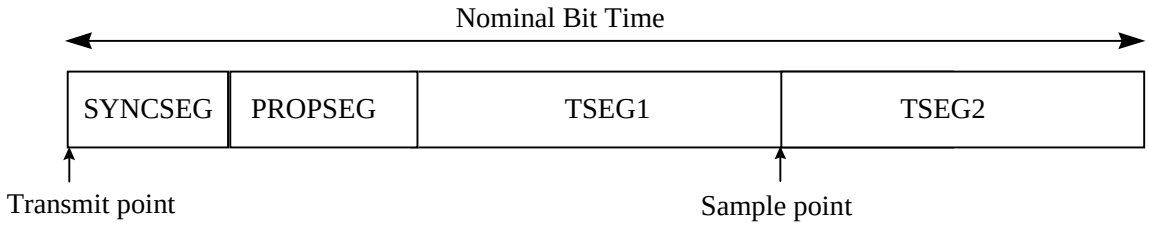
The nominal bit time is programmable at each node on a CAN bus. It must be equal for each node.

When any node receives a frame, it is necessary for the receiver to synchronize with the transmitter. Two types of synchronization exist:

- Hard Synchronization: when a Start of Frame is received

- Resynchronization: to compensate for oscillator drift and phase difference between transmitter and receiver oscillators. TSEG1 can be lengthened and TSEG2 shortened to move the sample point position. The maximum amount is SJW (Synchronization jump width).

Figure 5. Bit Time



SYNCSEG: segment used to synchronize the nodes on the bus. A bit edge is expected during this segment.

PROPSEG: period of time used to compensate for physical delay time within the network.

The nominal bit time is divided by time quanta:

1 bit time = (TSEG1 + TSEG2 + 1) * *Length_of_1_time_quantum*

$$Lenght\_of\_1\_time\_quantum = \frac{(BRP+1)}{I_{CLK}}$$

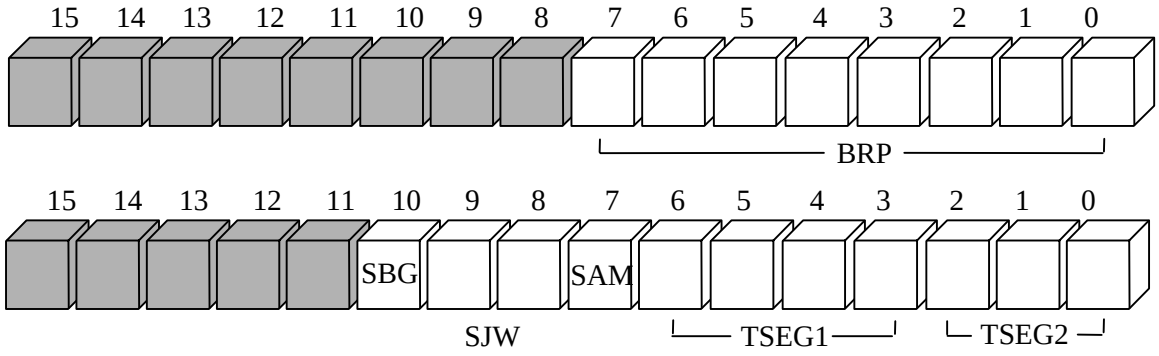
BRP= Baud rate prescaler
I_{CLK}= frequency of the clock = 20MHz

So for a transmission rate of 1Mbit/s (bit length =1 μs), the following settings are correct:

BRP=0 TSEG1=5 and TSEG2=4
BRP=1 TSEG1=12 and TSEG2=7

These parameters are configurable by the user (in the BCR register).

Figure 6. Bit Configuration Register (BCR)



Notes: 1) BRP: Baud Rate Prescaler
2) SBG: Synchronization on falling edge

3) SJW: Synchronization jump width
4) SAM: Sample point setting

To change the bit timing configuration:

Step 1: Set Change configuration Request bit in the MCR register.

CANMCR = 0001000000000000b
Bit 12 CCR = 1 ≥ Change configuration request

Step 2: Set BCR register (Bit Configuration Register).

All nodes in the bus must have the same nominal bit time and the same baud rate prescaler. If TSEG1 = TSEG2 = 0 the CAN cannot be activated.

$TSEG1 \geq TSEG2 \geq 2$ *if SBG = 0*

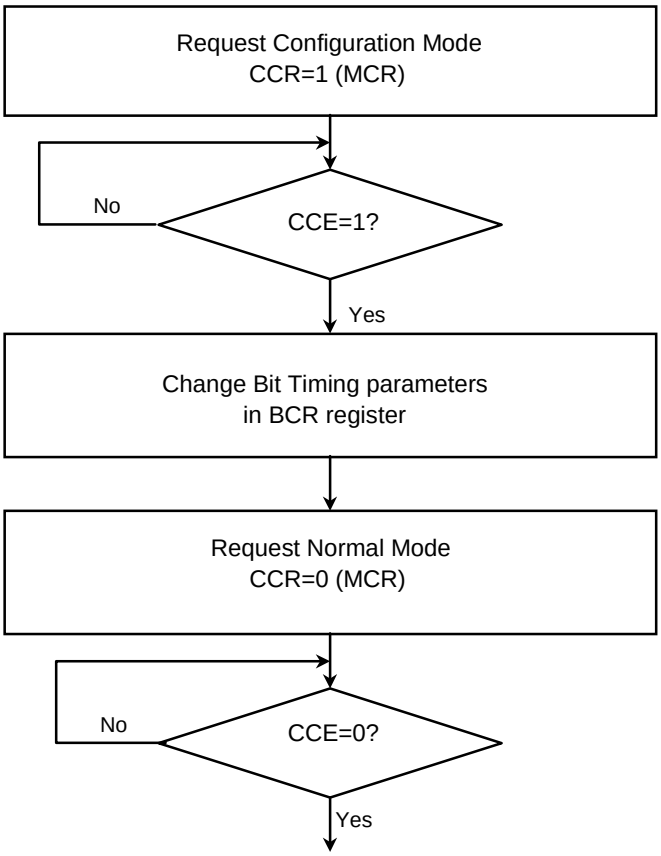
CANBCR2 = 0000000000000000b
baud rate prescaler = 0

CANBCR1 = 0000000101010111b
bit 10 SBG = 0 => Synchronization on falling edge
bit 8-9 SJW = 10 => Synchronization jump width
bit 7 SAM = 0 => CAN module samples only once
bit 3-6 TSEG1 = 1010
bit 0-2 TSEG2 = 111

Step 3: Request normal mode.

CANMCR = 0000000000000000b

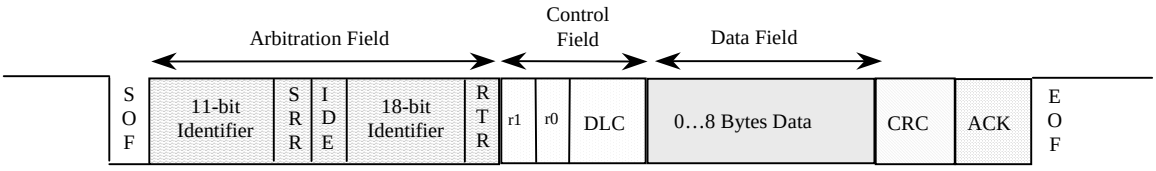
Figure 7. Bit Programming Flow Chart



CCE= Change Configuration Enable
(Global Status Register)

Mailboxes Initialization

Figure 8. Extended Data Frame



Each data frame is divided in several fields:

- The arbitration field contains the Identifier and the RTR (Remote Transmit Request) bit.
- The control field contains the DLC bit (data length).
- The data field

The user setting the mailboxes content can program these fields:

- MSGIDL and MSIDH contain the Identifier of the mailbox.
- MSGCTRL (message control field) contains the length of the message and the RTR bit (Remote Transmission Request used to send remote frames).
- DATA_A, DATA_B, DATA_C and DATA_D contain the data. The data is divided in four words or in eight bytes.

To initialize the mailboxes:

Step 1: Disable the mailboxes writing 0 in CANMDER.

CANMDER = 0000000000000000b

Step 2: Set Change Data Field Request bit in CANMCR.

CANMCR = 0000000100000000b bit 8 CDR = 1

Step 3: Change the mailbox contents (data, control and identifier fields). The data can be set only in the transmit mailboxes (2,3,4 or 5)

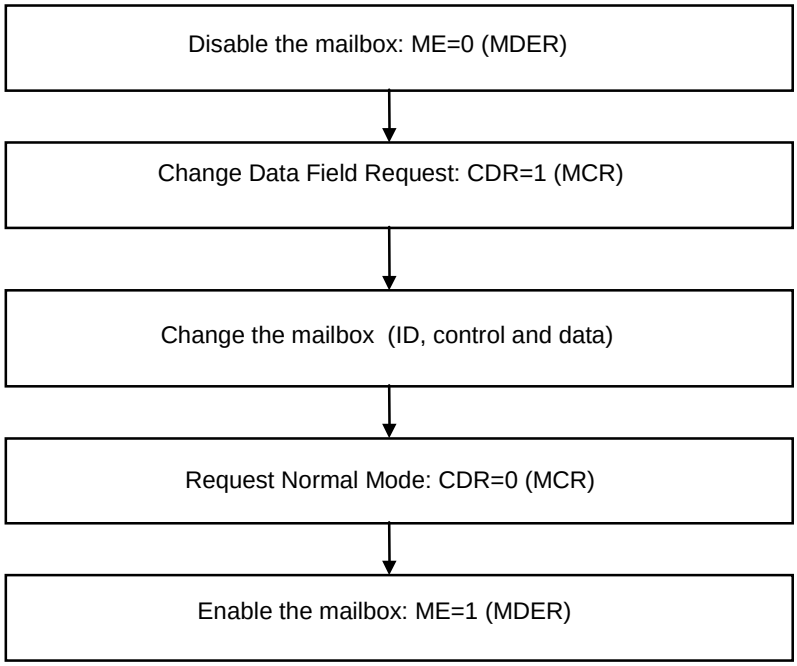
Step 4: Return in normal mode.

CANMCR = 0000000000000000b

Step 5: Enable the mailboxes.

CANMDER = 0000000000000100b
Bit 2 ME2= 1 => mailbox 2 enable
Bit 6 MD2= 0 => mailbox 2 configured as transmit mailbox

Figure 9. Mailbox Initialization Flow Chart



Example

```
;*****************************************************************
 ;************      CAN Registers configuration      ***************
;*****************************************************************

            SPLK   #0001000000000000b,CANMCR    ; Master Control Reg.
 ;                    FEDCBA9876543210
 ; Bit 12       1: Change configuration request

 W_CCE      BITCANGSR, #0Bh                     ; Wait for Change
            BCND   W_CCE, NTC                   ;Configuration request

            SPLK   #0000000000000000b,CANBCR2   ; Bit Configuration
;                   FEDCBA9876543210               register 2

            ; bit 0-7    Baud rate prescaler
            ; bit 8-15   Reserved

            SPLK   #0000000001010111b,CANBCR1   ; Bit Configuration
;                   FEDCBA9876543210               register

            ; bit 0-2    TSEG1
            ; bit 3-6    TSEG2
            ; bit 7      Sample point setting (1: 3 times, 0: once)
            ; bit 8-9    Synchronization jump width
            ; bit A      0: Synchronization on falling edge
            ; bit C-F    Reserved

            SPLK   #0000000000000000b,CANMCR    ; Master Control
;                   FEDCBA9876543210                register

            ;bit 12      0: Normal mode requested

W_NCCE      BITCANGSR,#0Bh                      ; Wait for normal mode
            BCND   W_NCCE,TC
```

```
;*********************************************************************
;*************     Configure CAN before writing  in RAM *************
;*********************************************************************

            LDP   #0E2h                       ; DP => 7100h
            SPLK  #0000000000000000b,CANMDER  ; Mailbox Direction
;                    FEDCBA9876543210           /Enable Register
            ; bit 0-5    disable mailboxes 0 to 5
            ; bit 6-7    mailbox 2 and 3 configured as transmit (0)
            ; bit 8-F    reserved

            SPLK  #0000000100000000b,CANMCR      ; Master Control
;                    FEDCBA9876543210              register
; bit 8         CDR: Change data field request

;*********************************************************************
;*************        Write CAN Mailboxes              *************
;*********************************************************************
            LDP   #0E4h                          ; DP => 7200h
            SPLK  #1111111111111111b,CANMSGID3H  ; Set the
;                    FEDCBA9876543210            message identifier

            ; bit 0-12   Upper 13 bits of extended identifier
            ; bit 13     Auto answer mode bit
            ; bit 14     Acceptance mask enable bit
            ; bit 15     Identifier extension bit

            SPLK  #1111111111111111b,CANMSGID3L
;                    FEDCBA9876543210
            ; bit 0-15      Lower part of extended identifier

            SPLK  #0000000000001000b,CANMSGCTRL3  ; Set control field
;                    FEDCBA9876543210

            ;bit 0-3     Data length code: 1000 = 8 bytes
            ;bit 4    0: data frame
```

```
                SPLK   #0123h,CANMBX3A          ; Message to transmit
                SPLK   #4567h,CANMBX3B
                SPLK   #89ABh,CANMBX3C
                SPLK   #0CDEFh,CANMBX3D

;*********************************************************************
;**************      Set parameters after writing      **************
;*********************************************************************
            LDP    #0E2h                       ; DP => 7100h
            SPLK   #0000000000000000b,CANMCR   ; Master Control
;                   FEDCBA9876543210              register
            ; bit 8      0 : Normal mode requested

            SPLK   #0000000001001100b,CANMDER  ; Mailbox Direction
;                   FEDCBA9876543210            /Enable Register

            ; bit 0      disable mailbox 0
            ; bit 1      disable mailbox 1
            ; bit 2      enable mailbox 2
            ; bit 3      enable mailbox 3
            ; bit 4      disable mailbox 4
            ; bit 5      disable mailbox 5
            ; bit 6      mailbox 2 configured as  receive(1) mailbox
            ; bit 7      mailbox 3 configured as transmit(0) mailbox
            ; bit 8      reserved
```

Transmit a Message

To Transmit a Message

Step 1: Initialization of the transmit mailbox:

- ⇨ Disable the mailboxes writing 0 in CANMDER.

  CANMDER = 0000000000000000b

- ⇨ Ask for a Change Data Field Request writing in CANMCR.

  CANMCR = 0000000100000000b

  bit 8 CDR = 1

- ⇨ Set a message ID for a transmit mailbox. Writing in CANMSGIDxH and CANMSGIDnL with n = 2,3,4 or 5.

  CANMSGIDnH = 1110000000000000b

bit 15 IDE = 1 => The message to be sent has an extended identifier (29 bits)
bit 14 AME = 1 => The corresponding acceptance mask is used (LAM register)
bit 13 AAM = 1 => Auto answer mode bit set. If this mailbox receive a remote frame, it will answer sending back its contents
bit 12-0 : upper part of the identifier.

CANMSGIDnL = 0000000000001111b
lower part of the identifier.

- ⇨ Set the message control field. Writing in CANMSGCTRLn with n = 2,3,4 or 5 If the message to send is a remote frame the RTR bit will be set to 1. The length of the message will be chosen here.

  CANMSGCTRLn = 0000000000001000b
  bit 5 RTR = 0 => a data frame will be sent (not a remote frame)
  bit 0-4 DLC = 1000 => data length = 8 bytes

- ⇨ Create the message (for a data frame only). The message will be written in CANMBXnA, CANMBXnB, CANMBXnC and CANMBXnD (with n = 2,3,4 or 5).

  CANMBXnA = 0ABCDh
  CANMBXnB = 0123h
  CANMBXnC = 0EF32h
  CANMBXnD = 6789h

- ⇨ Request Normal operation resetting the bit 8 in CANMCR.

  CANMCR = 0000000000000000b

- ⇨ Enable the mailbox writing in the CANMDER register. If the mailboxes 2 or 3 have been chosen, they will have to be configured as transmit mailboxes (CANDER register).

  CANMDER = 0000000000000100b
  bit 2 ME2= 1 => mailbox 2 enable
  bit 6 MD2= 0 => mailbox 2 configured as transmit mailbox

Step 2: Request to transmit a message setting a TRS bit in the TCR register.

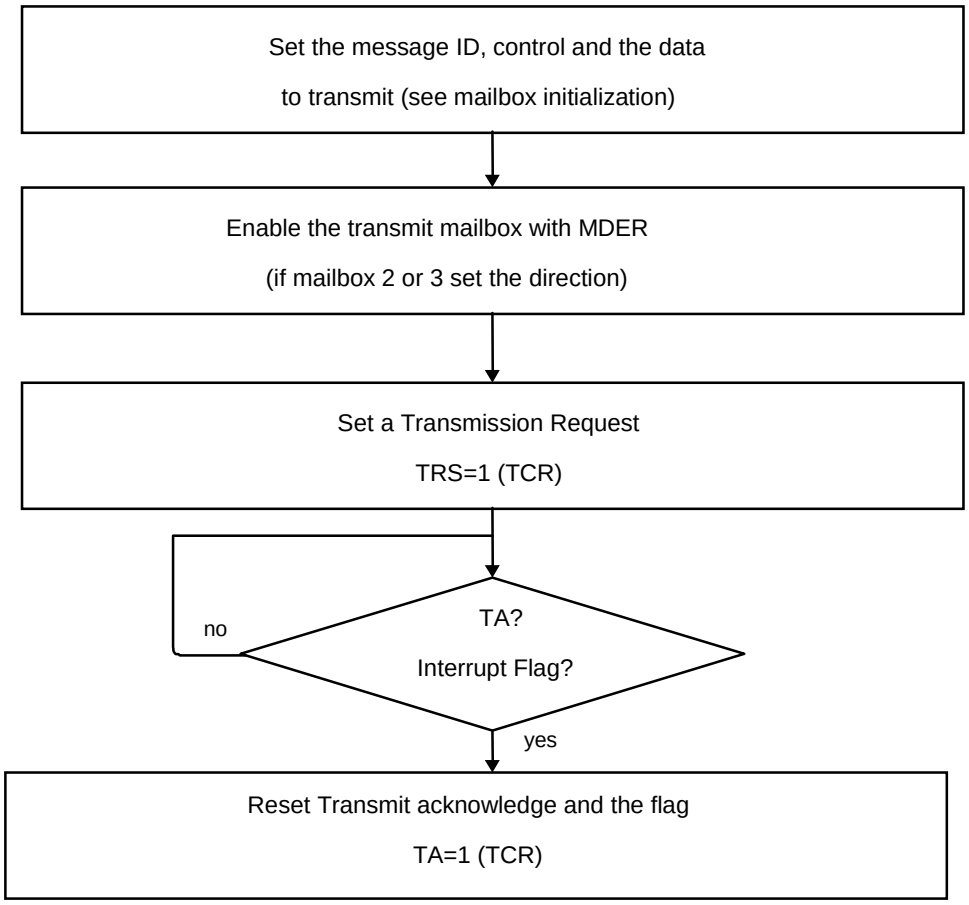
TCR = 0000000000010000b
bit 4 TRS2 = 1 => Transmission request for mailbox 2

Step 3: Wait for the transmit acknowledge (TA = 1 in TCR Register) and/or for the mailbox flag (CANIFR register).

Step 4: To reset TA and the transmit flag, a “1” need to be written in TA (TCR register).

TCR = 0001000000000000b
bit 12 TA2 = 1 => reset TA and interrupt flag for mailbox 2

Figure 10. Transmit Flow Chart



Example

```
;*********************************************************************
;************         Configure the Shared Pins          ***********
;*********************************************************************
                LDP    #225
                SPLK   #0FFFFH,OCRA
```

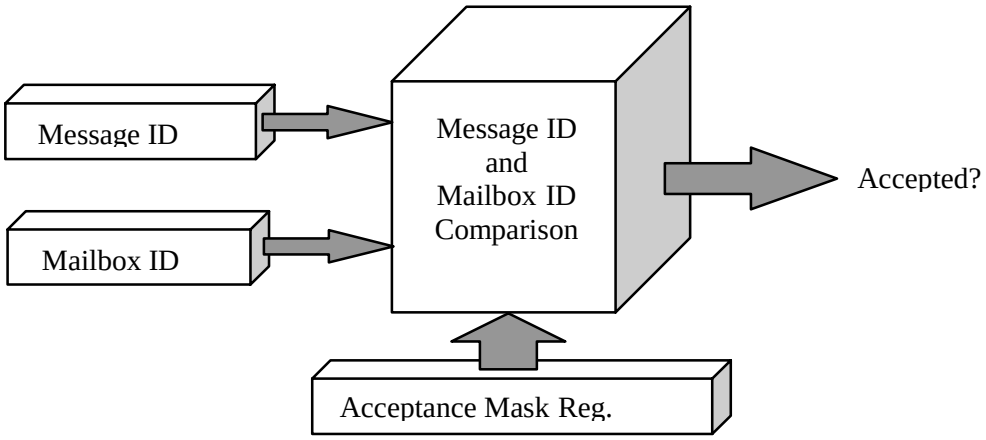
```
                SPLK   #0FFF3H,OCRB

;*********************************************************************
;************      Configure CAN before writing  in RAM   ************
;*********************************************************************
            LDP    #0E2h                        ; DP => 7100h
            SPLK   #0000000000000000b,CANMDER   ; Mailbox Direction
;                    FEDCBA9876543210            /Enable Register
            ; bit 0-5    disable mailboxes

            SPLK   #0000000100000000b,CANMCR    ; Master Control Reg.
;                    FEDCBA9876543210
            ; bit 8      CDR: Change data field request

;*********************************************************************
;************            Write CAN Mailboxes            *************
;*********************************************************************
            LDP    #0E4h                           ; DP => 7200h
            SPLK   #1111111111111111b,CANMSGID5H   ; Set the message
;                    FEDCBA9876543210                 identifier

            ;bit 0-12    Upper 13 bits of extended identifier
            ;bit 13      Auto answer mode bit
            ;bit 14      Acceptance mask enable bit
            ;bit 15      Identifier extension bit

            SPLK   #1111111111111111b,CANMSGID5L
;                    FEDCBA9876543210
            ; bit 0-15      Lower part of extended identifier

            SPLK   #0000000000001000b,CANMSGCTRL5  ; Set control field
;                    FEDCBA9876543210
            ; bit 0-3    Data length code: 1000 = 8 bytes
            ; bit 4         0: data frame

            SPLK   #0123h,CANMBX5A              ; Message to transmit
            SPLK   #4567h,CANMBX5B
```

```
            SPLK   #89ABh,CANMBX5C
            SPLK   #0CDEFh,CANMBX5D

;*********************************************************************
;************       Set parameters after writing        *************
;*********************************************************************

            LDP    #0E2h                        ; DP => 7100h
            SPLK   #0000000000000000b,CANMCR    ; Master Control Reg.
;                   FEDCBA9876543210
            ; bit 8      0 : Normal mode requested

            SPLK   #0000000001100000b,CANMDER   ; Mailbox Direction
;                   FEDCBA9876543210            /Enable Register
            ; bit 0-4    disable mailboxes 0 to 4
            ; bit 5      enable mailbox 5

;*********************************************************************
;**************          Mailbox 5 Transmission        **************
;*********************************************************************

               SPLK   #0080h,CANTCR      ; Transmit request for
                                         ; mailbox 5

W_TA           BITCANTCR,0               ; Wait for transmission ACK
               BCND   W_TA,NTC

W_FLAG         BITCANIFR,2               ; Wait for mailbox 5
               BCND   W_FLAG,NTC         ; interrupt flag
               SPLK   #2000h,CANTCR      ; Reset TA and mailbox flag
```

Receive a message

How the Acceptance Filter Works

Figure 11. Acceptance Filter



The bits that are not masked by the local Acceptance Mask register need to be identical in the received message ID and in the receive mailbox ID. If this is not the case, the message is neither accepted nor stored. The local acceptance mask can be disabled by fixing the AME (Acceptance Mask Enable bit) to 0 in the message Identifier high word (MSGIDn). Then all identifier bits must match to store the message.

Example:

Message ID = 1 0000 0000 0000 0000 0000 1111 0000
Mailbox ID = 1 0000 0000 0000 0000 0000 0000 0000

Acceptance mask = 1 0000 0000 0000 0000 0000 1111 0000
(1 = masked bit)

- message accepted

Acceptance mask = 1 0000 0000 0000 0000 0000 0000 1111

- message refused

How to Program the CAN Module

Step 1: Set the local acceptance mask register. LAM1 is used for mailboxes 2 and 3 and LAM0 is used for mailboxes 0 and 1.

LAM1H = 1000000000000000b
bit 15 LAMI = 1 standard and extended frames can be received
bit 12-0 0: the ID bit corresponding are not masked. For these bits the received message ID has to be the same than the mailbox ID.

LAM1L= 1111111111111111b
bit 15-0 1: Bit masked

Step 2: Set the mailbox Identifier and Control.

⇨ Disable the mailboxes writing 0 in MDER.

CANMDER = 0000000000000000b

- ⇨ Ask for a Change Data Field Request writing in MCR register.

  CANMCR = 0000000100000000b bit 8 CDR = 1

- ⇨ Set a message ID for a transmit mailbox. Writing in MSGIDxH and MSGIDnL with n = 2,3,4 or 5.

  CANMSGIDnH = 1110000000000000b
  bit 15 IDE = 1 => The received message has an extended identifier (29 bits)
  bit 14 AME = 1 => The corresponding acceptance mask is used (LAM register)
  bit 13 AAM = 1 => No influence for a receiver.

  CANMSGIDnL = 0000000000001110b Lower part of the identifier.

- ⇨ Set the message control field. Writing in MSGCTRLn with n = 2,3,4 or 5. If the message to send is a remote frame the RTR bit will be set to 1. The length of the message will be chosen here.

  CANMSGCTRLn = 0000000000001000b
  bit 5 RTR = 0 => no data frame is requested.
  bit 0-4 DLC = 1000 => data length = 8 bytes

- ⇨ Request Normal operation resetting the bit 8 in MCR.

  CANMCR = 0000000000000000b

- ⇨ Enable the mailbox writing in the MDER register. If mailbox 2 or 3 is chosen, it will have to be configured as a receive mailbox (MDER register).

  CANMDER = 0000000001000100b
  bit 2 ME2= 1 => mailbox 2 enable
  bit 6 MD2= 1 => mailbox 2 configured as receive mailbox

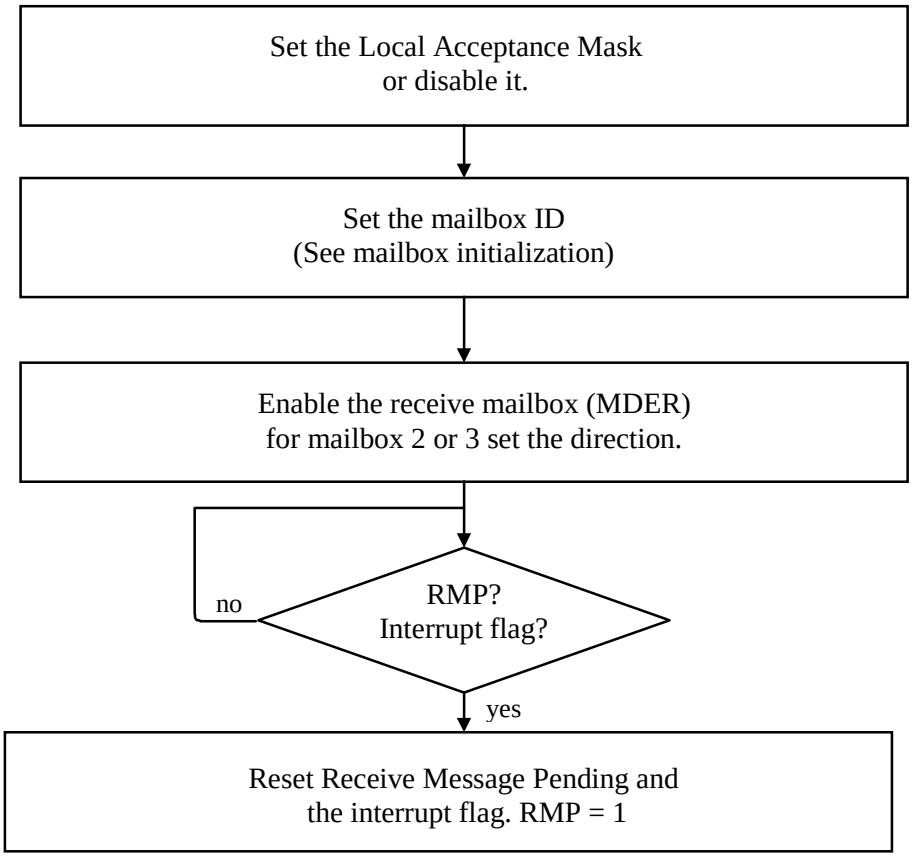
Step 3: Wait for receive acknowledge (RMP bit in RCR register) and for mailbox interrupt flag in CANIFR.

Step 4: To reset RMP and the receive flag, a “1” must be written in RMP (in the Receive Control Register).

CANRCR = 0000000001000000b
bit 6 RMP2 = 1 => reset RMP2 and mailbox 2 interrupt flag

The data is stored in MBXnA, MBXnB, MBXnC and MBXnD.

Figure 12. Receive Flow Chart



Example

```
;*********************************************************************
;*************       Configure the Shared Pins          **************
;*********************************************************************

                LDPK   #225
                SPLK   #0FFFFH,OCRA
                SPLK   #0FFF3H,OCRB

;*********************************************************************
;**************          Set Local Acceptance Mask       **************
;*********************************************************************

              LDP    #0E2h                          ; DP => 7100h
              SPLK   #1001111111111110b,CANLAM0H  ; Set local acceptance
```

```
                                                ;mask for mailboxes 0 & 1
            SPLK  #1111111111111111b,CANLAM0L  ; 1:don't care

;*********************************************************************
;***********       Configure CAN before writing in RAM   ************
;*********************************************************************

            SPLK  #0000000000000000b,CANMDER   ; Mailbox Direction
;                   FEDCBA9876543210            /Enable Register
            ; bit 0-5    disable mailboxes

            SPLK  #0000000100000000b,CANMCR    ; Master Control Reg.
;                   FEDCBA9876543210
            ; bit 8      CDR: Change data field request

;*********************************************************************
;****************           Write CAN Mailboxes        **************
;*********************************************************************

            LDP   #0E4h                           ; DP => 7200h
            SPLK  #1111111111111111b,CANMSGID0H    ; Set the
;                   FEDCBA9876543210              mailbox identifier
            ;bit 0-12    Upper 13 bits of extended identifier
            ;bit 13      Auto answer mode bit
            ;bit 14      Acceptance mask enable bit
            ;bit 15      Identifier extension bit

            SPLK  #1111111111111111b,CANMSGID0L
;                   FEDCBA9876543210
            ;bit 0-15    Lower part of extended identifier

;*********************************************************************
;*************         Set parameters after writing       ************
;*********************************************************************

            LDP   #0E2h                           ; DP => 7100h
            SPLK  #0000000000000000b,CANMCR    ; Master Control Reg.
```

```
;                   FEDCBA9876543210
             ; bit 8      0 : Normal mode requested

             SPLK   #0000000000000001b,CANMDER   ; Mailbox Direction
;                    FEDCBA9876543210              /Enable Register

             ; bit 0      enable mailbox 0
             ; bit 1-5    disable mailbox 1 to 5

;*********************************************************************
;*****************            RECEIVE                ****************
;*********************************************************************

W_RA            BITCANRCR,11             ; Wait for receive ACK
                BCND   W_RA,NTC

W_FLAG          BITCANIFR, 7             ; Wait for mailbox 0
                BCND   W_FLAG,NTC        ; interrupt flag

                SPLK   #0010h,CANRCR     ; Reset RA and CANIFR
```

Self-Test Mode

What is Self-Test Mode?

The TMS320F241/3 CAN module can operate in loop-back mode. It can receive its own transmitted message. The module generates its own acknowledge signal. This mode operates without a bus connected to the module, making it convenient to test programs.

How to Use Self-Test Mode

After the CAN module initialization, the user can request the Self-Test Mode by setting the Self-Test Mode bit in the MCR register.

CANMCR = 0000000010000000b

bit 6: STM = 1. The module is in self-test mode.

The messages transmitted will be received and stored in the appropriate receive mailbox.

Self-Test Mode Limitation

The remote frame handling with the Auto Answer Mode set is not implemented in Self-Test mode. This mode cannot be used for remote frame testing.

Self-Test Mode Example

```
;*********************************************************************
;************          CAN Bit Timing Configuration     *************
;*********************************************************************

            SPLK  #0001000000000000b,CANMCR
; bit 12        Change configuration request

W_CCE       BITCANGSR,#0Bh        ; Wait for Change configuration
            BCND  W_CCE,NTC       ; enable

            SPLK  #0000000000000000b,CANBCR2
            ; bit 0-7 Baud rate prescaler
            SPLK  #0000010101010111b,CANBCR1
            ; bit 0-2    TSEG1
            ; bit 3-6    TSEG2
            ; bit 7      Sample point setting (1: 3 times, 0: once)
            ; bit 8-A    Synchronization jump width
            ; bit B      0: Synchronization on falling edge
            ; bit C-F    Reserved

            SPLK  #0000000000000000b,CANMCR

W_NCCE      BITCANGSR,#0Bh        ; Wait for Change configuration
            BCND  W_NCCE,TC       ; disable

;*********************************************************************
;***********     Configure CAN before writing  in RAM    ************
;*********************************************************************

            LDP   #0E2h                       ; DP => 7100h
            SPLK  #0000000000000000b,CANMDER  ; Mailbox Direction
;                    FEDCBA9876543210           /Enable Register
            ; bit 0-5    disable mailboxes

            SPLK  #0000000100000000b,CANMCR   ; Master Control Reg.
            ; bit 8      CDR: Change data field request
```

```
;*********************************************************************
;***************        Write CAN Mailboxes          ****************
;*********************************************************************

            LDP   #0E4h                           ; DP => 7200h
            SPLK  #1111111111111111b,CANMSGID2H   ; Set the message
;                    FEDCBA9876543210                 identifier
            ;bit 0-12    Upper 13 bits of extended identifier
            ;bit 13      Auto answer mode bit
            ;bit 14      Acceptance mask enable bit
            ;bit 15      Identifier extension bit


            SPLK  #1111111111111010b,CANMSGID2L
            ;bit 0-15    Lower part of extended identifier

            SPLK  #0000000000001000b,CANMSGCTRL2  ; Set control field
;                    FEDCBA9876543210
            ;bit 0-3     Data length code: 1000 = 8 bytes
            ;bit 4       0: data frame

            SPLK  #1111111111111111b,CANMSGID3H   ; Set the message
;                    FEDCBA9876543210                 identifier
            ;bit 0-12    Upper 13 bits of extended identifier
            ;bit 13      Auto answer mode bit
            ;bit 14      Acceptance mask enable bit
            ;bit 15      Identifier extension bit

            SPLK  #1111111111111111b,CANMSGID3L
            ;bit 0-15    Lower part of extended identifier

            SPLK  #0000000000001000b,CANMSGCTRL3  ; Set control field
;                    FEDCBA9876543210
            ;bit 0-3     Data length code: 1000 = 8 bytes
            ;bit 4       0: data frame
```

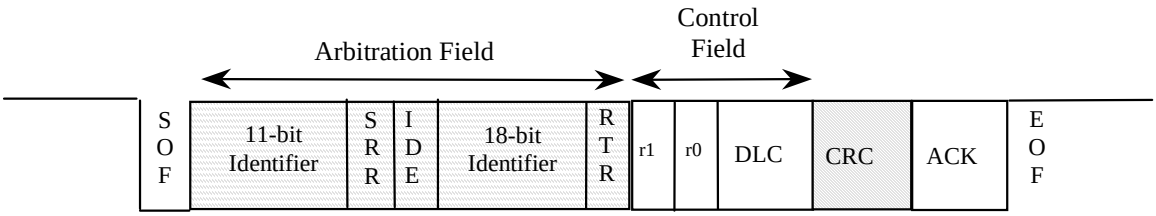
```
                SPLK   #0123h,CANMBX3A              ; Message to transmit
                SPLK   #4567h,CANMBX3B
                SPLK   #89ABh,CANMBX3C
                SPLK   #0CDEFh,CANMBX3D

;*********************************************************************
;**********        Set parameters after writing           ************
;*********************************************************************

                LDP    #0E2h                        ; DP => 7100h
                SPLK   #0000000001000000b,CANMCR    ; Master Control Reg.
;                       FEDCBA9876543210
                ;bit 6    Self mode test


                SPLK   #0000000001001100b,CANMDER   ; Mailbox Direction
;                       FEDCBA9876543210              /Enable Register

                ; bit 0      disable mailbox 0
                ; bit 1      disable mailbox 1
                ; bit 2      enable mailbox 2
                ; bit 3      enable mailbox 3
                ; bit 4      disable mailbox 4
                ; bit 5      disable mailbox 5
                ; bit 6      1: mailbox 2 receive
                ; bit 7      0: mailbox 3 transmit

;*********************************************************************
;***********                  TRANSMIT                     ************
;*********************************************************************

                SPLK   #0020h,CANTCR      ;Transmit request for mailbox 4

W_TA            BITCANTCR,2               ; Wait for transmission
                BCND   W_TA,NTC           ; acknowledge
```

```
W_FLAG3     BITCANIFR,4             ; Wait for interrupt flag
            BCND   W_FLAG3,NTC
            SPLK   #2000h,CANTCR    ; Reset TA and CANIFR


;*********************************************************************
;************             RECEIVE                          ************
;*********************************************************************

W_FLAG2     BITCANIFR,5             ; Wait for interrupt flag
            BCND   W_FLAG2,NTC

W_RA        BITCANRCR,9             ; Wait for receive acknowledge
            BCND   W_RA,NTC
            SPLK   #0040h,CANRCR    ; reset RMP and CANIFR
```

Remote Frame

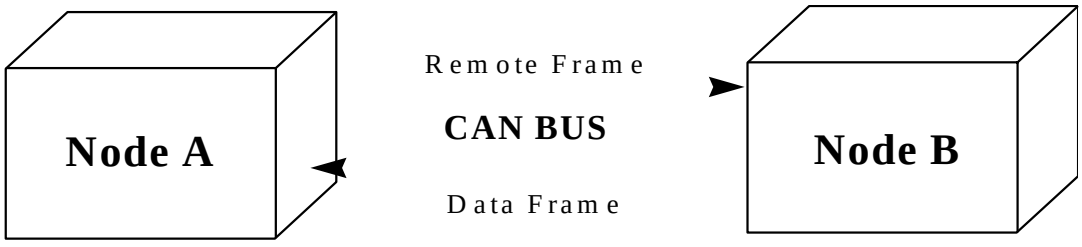
Remote frames have the same shape as data frames but contain no data. The RTR (Remote Transmission Request) bit of the remote frame is set to 1. Similar to the data frames, they can be standard or extended (11-bit ID or 29-bit ID).

Figure 13. Extended Remote Frame



Remote frames are usually for requesting information. Node A sends a remote frame to node B. If node B has a message to transmit with the same identifier as the remote frame, it will answer, sending the corresponding data frame to the bus.

Figure 14. Remote Frame Principle (with Auto Answer Bit Set)



How to Program Remote Frame

To Send a Remote Frame:

- ❒ Use mailbox 2, 3, 4 or 5. Mailboxes 2 and 3 can be configured either as transmit mailboxes or as receive mailboxes.
- ❒ Set the RTR (Remote Transmission Request) bit to 1 in the MSGCTRLn field.
- ❒ Set the TRS (Transmission Request Set) bit to 1.
- ❒ A remote frame will be sent to the CAN bus. If the remote frame is sent from a receive mailbox (2 or 3), no TA (Transmit Acknowledge) or mailbox flag is set after a successful transmission. The TRS bit is then reset.

Automatic Answer to a Remote Frame:

The mailbox that receives the remote frame answers automatically by sending a data frame.

- ❒ Only for mailbox 2 or 3 configured as transmit mailbox
- ❒ Set the Auto Answer Mode bit (AAM) in MSGIDn.
- ❒ If the node receives a remote frame with the same ID than the mailbox ID, it will automatically answer by sending a data frame (for a local acceptance mask disabled).

Sending a Remote Frame to a Receive Mailbox:

- ❒ Only for mailboxes 0,1 or 2,3 configured as receive mailboxes
- ❒ The message is handled like a data frame. The RMP (Receive Message Pending) bit and the RFP (Remote Frame Pending) bit are set.
- ❒ The CPU handles the situation.

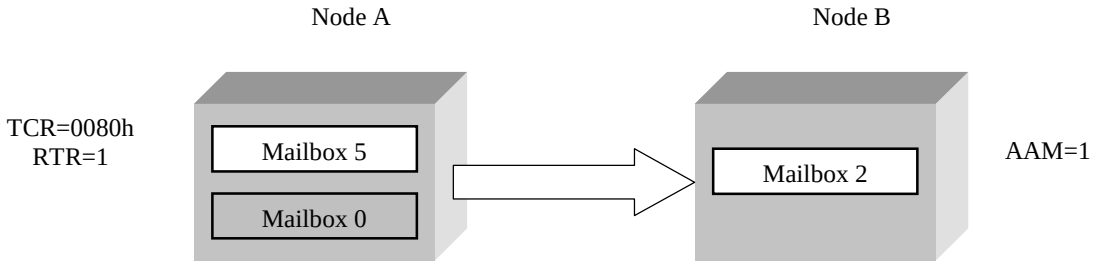
Example:

Step 1: Sending a remote frame from mailbox 5 (node A)

The RTR (Remote Transmission Request) bit for mailbox 5 is set. Mailbox 5 will send a remote frame when requested. Then, when the corresponding TRS (Transmission Request Set) bit in the TCR register (Transmit Control Register) is set, a remote frame is sent on the CAN bus.

As the local acceptance mask is disabled (MSGID2H bit 14), the transmitted ID bits (mailbox 5 node A) and the receive mailbox ID bits (mailbox 2 node B) must match to accept the frame. Node B recognizes the remote frame ID. Mailbox 5 (node A) and mailbox 2 (node B) have the same 29-bit identifier.

Figure 15. Remote Request

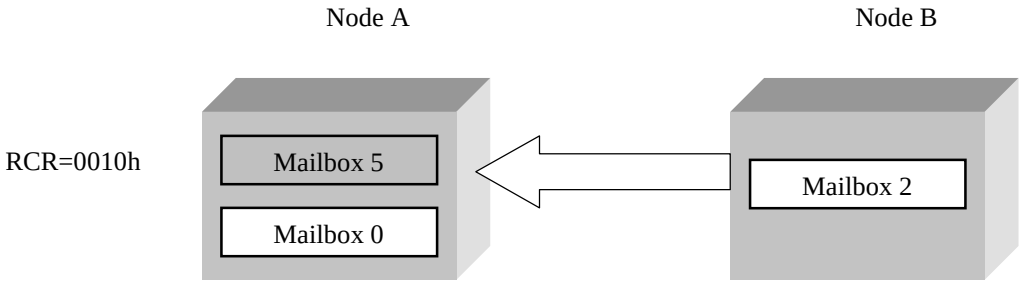


MAILBOX 5 Settings:	MAILBOX 2 Settings:
MSGID5H=1101001100001111b Bit 15: 1 Extended ID Bit 14: 1 Acceptance mask enable Bit 13: 0 Auto Answer Mode not set Bit 12-0: Identifier	MSGID2H=1011001100001111b Bit 15: 1 Extended ID Bit 14: 0 Acceptance mask disable Bit 13: 1 Auto Answer Mode set Bit 12-0: Identifier
MSGID5L=1111111111111111b Bit 15-0: Identifier	MSGID2L=1111111111111111b Bit 15-0: Identifier
MSGCTRL5=0000000000011000b Bit 4: 1 Remote Transmission Request	MSGCTRL2=0000000000001000b Bit 4: 0 RTR not set Bit 3-0: 1000 data length=8 bytes

Step 2: Node B Auto-answer

In node B, as the AAM (Auto Acceptance Mode) bit is set, mailbox 2 answers automatically by sending the corresponding data frame on the CAN bus. In node A, the local acceptance mask is enabled for mailbox 0 (MSGID0H, bit 14). As the non-masked bits of the data frame sent by the node B match with the mailbox 0 ID bits, the data frame is accepted and stored in the mailbox 0.

Figure 16. Auto Answer



MAILBOX 0 Settings:

LAM0H=1000000000000000b
Bit 15: 1 Extended and Standard ID accepted
Bit 12-0:0 Transmitted ID and mailbox ID must match identically

LAM0L=1111111111111111b
Bit 15-0: 1 Accept 0 or 1

MSGID0H=1101001100001111b
Bit 15: 1 Extended ID
Bit 14: 1 Acceptance mask enable
Bit 13: 0 Auto Answer Mode not set
Bit 12-0: Identifier

MSGID0L=1010101011110000b
Bit 15-0: Identifier

MSGCTRL5=0000000000001000b
Bit 3-0: 1000 data length=8 bytes

CAN Interrupts

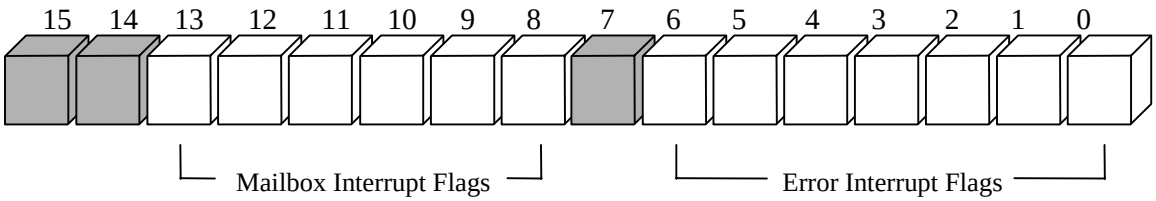
CAN module contains two interrupts registers:

- CANIFR: Interrupt Flag register
- CANIMR: Interrupt Mask register

There are two different types of interrupts:

- Interrupts generated by a mailbox, if a mailbox receives or transmits a message. Each mailbox has an interrupt flag bit on CANIFR and an interrupt mask bit on CANIMR.
- Interrupts generated by an error. Several events can generate error interrupts:
  - Abort acknowledge
  - Write denied
  - Wake up
  - Receive message lost
  - Bus off
  - Error Passive
  - Warning Level

Figure 17. CAN Interrupt Flag Register



These interrupts can assert either a high priority request or a low priority request.

Bits 15 and 7 in CANIMR are used to select the priority.

Two interrupt requests can be sent to the Peripheral Interrupt Expansion (PIE):

- CAN mailbox interrupt (high or low priority)
- CAN error interrupt (high or low priority)

Error Handling

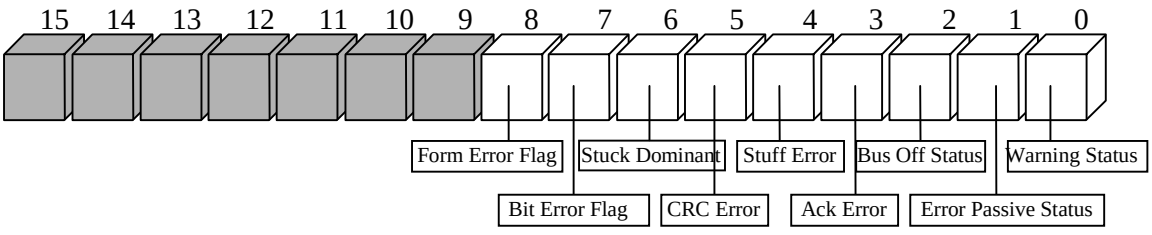
The CAN module includes error detection, internal error handling mechanism and error signaling. This provides a reliable and robust data handling mechanism.

The CAN module detects the following error types:

- Bit error if the transmitted bit and the received bit are different.
- Bit Stuffing error. After five consecutive equal bits, the sender is supposed to insert a stuff bit with the complementary value into the bit stream, which is removed by the receivers.
- CRC error, if the received CRC (Cyclic redundancy check) code does not match the transmitted CRC code.
- ACK error, if the transmitting node receives no ACK from receiver(s).
- Form error, if a violation of frame format occurs.

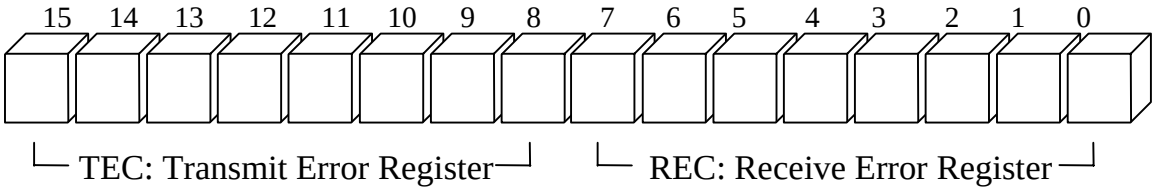
These errors are recorded on the Error Status Register:

Figure 18. Error Status Register (ESR)



Each node that detects an error, increments the error counter (transmit or receive). These two counters are in the CAN Error Counter Register (CEC).

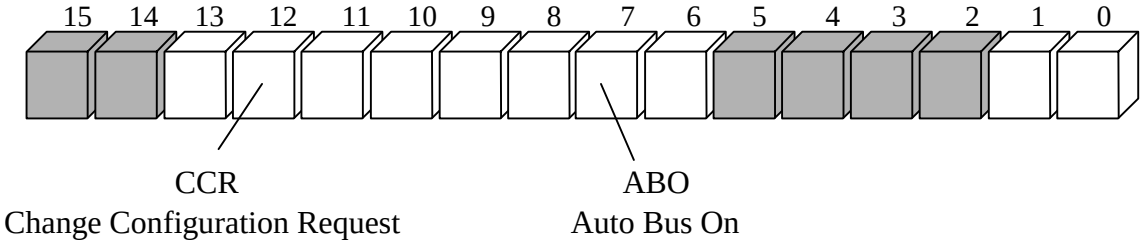
Figure 19. CAN Error Counter Register CEC



The CAN module can be in three different states:

- **Error Active State**: if the transmit and receive error counters are below 128. If one of the counters reach 96, a flag is set in the CANIFR register (bit 0, warning flag) and the bit 0 of the Error Status Register is equal to 1.
- **Error Passive State**: if the transmit error counter or the receive error counter is between 127 and 255. A flag in CANIFR (bit 1) is set and EP bit in the Error Status Register is equal to 1.
- **Bus Off State**: if the transmit error counter is greater than 255. Then the node is automatically disconnected from the bus. A flag is set in CANIFR (bit 2) and the BO bit in the Error Status Register is equal to 1.The CCR bit (MCR register) is set to 1 if ABO = 0 (MCR register).

Figure 20. Master Control Register (MCR)



To reconnect the module after a bus-off condition, two different solutions exist:

- Set the ABO (Auto Bus On) bit of the MCR register. The module will go back to the bus-on state after 128*11 consecutive recessive bits.
- Or clear the CCR (Change Configuration Request) bit of the MCR register.

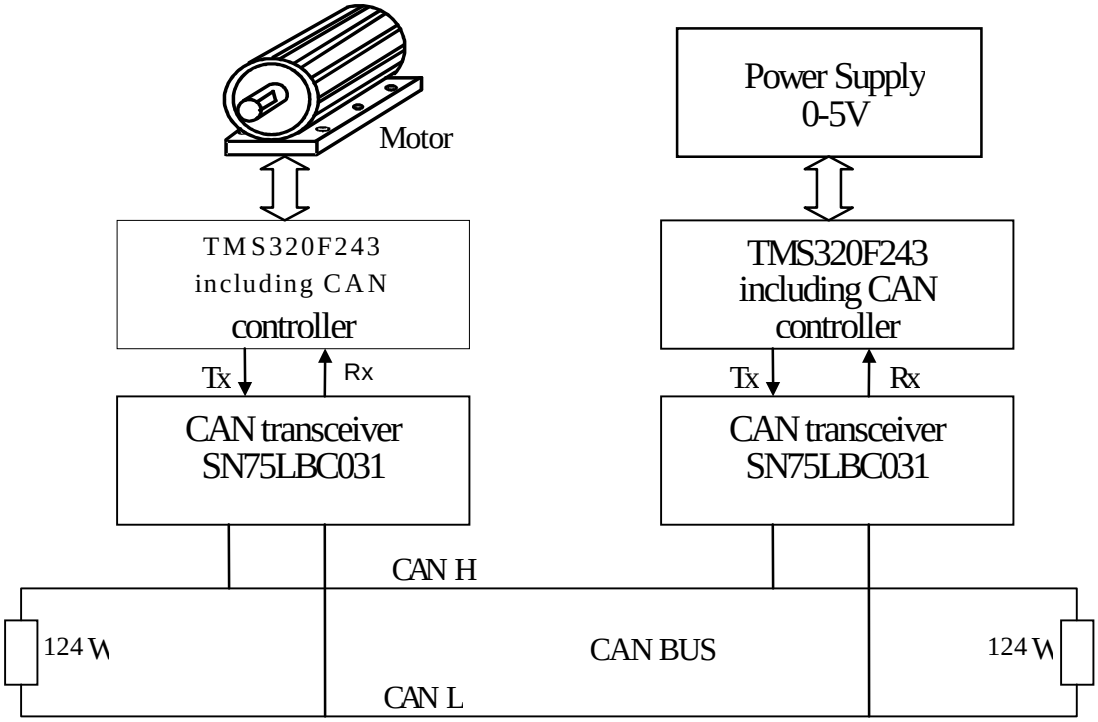
Application to Motor Control

General Description

The goal of this application is to send a speed command via a CAN bus to a motor drive to vary motor speed.

To implement this application, a twisted pair (as a CAN bus), two TI SN75LBC031 transceivers and two TMS320F243 DSPs, are used. The motor controlled is a three-phase AC induction motor.

Figure 21. Motor Control Application



A power supply is connected to the ADC0 input of a TMS320F243 DSP controller. This value is read as the frequency target for the motor control. The first DSP (node A) converts this value to a digital value using the ADC module and then scales it. 0 Volts corresponds to a frequency of 200 rpm and 5 Volts to 400 rpm. The data is stored in a CAN mailbox and then sent to the second DSP (node B) by the CAN bus. This DSP controls the motor using the frequency target received.

The motor speed can be changed in real time by varying the ADC input voltage.

To implement this application, two different programs are needed. On the first node, the software handles the transmission of the frequency target in the CAN bus. On the second node, the program handles the reception of the frequency and also performs motor control.

Node A: From the Power Supply to the CAN Bus

The DSP A is connected to the power supply (ADC0 input) and to the transceiver. The program running in this DSP is called send_frequency.asm. This program handles the DC and CAN module initialization, the analog-to-digital conversion, the scaling of the frequency, its storage inside the mailbox and sends it to the CAN bus. Figure 20 describes the process.

In this program the CAN is used three times:

- CAN initialization (mailbox and bit timing)
- Storage of the frequency in the mailbox
- Transmission of the message

The CAN initialization mailbox and bit timing are described in details in the part 2 of this application report.

The storage of the frequency follows the same principle as the mailbox initialization. Before updating the data value, the mailbox needs to be disabled and the CCR bit (Change configuration request) in the Master Control Register must be set. After changing the frequency, this bit must be cleared and the mailbox enabled.

Code:

```
                    LDP   #DP_CAN
                    SPLK  #0000000000000000b,CANMDER
;bit 0-5            Disable each mailbox

                    SPLK  #0000000100000000b,CANMCR
;                          ||||||||||||||||
;                          FEDCBA9876543210
;bit 8               CDR: Change data field request

                    LDP   #04h
                    LACL  GPR0            ; Load frequency value inside
                    LDP   #DP_CAN2        ; the CAN mailbox 3
                    SACL  CANMBX3A
                    LDP   #DP_CAN
                    SPLK  #0000010000000000b,CANMCR
;                          ||||||||||||||||
;                          FEDCBA9876543210
;bit 8               CDR: Change data field request
;bit 10              Data byte order. First sent:0,1
```

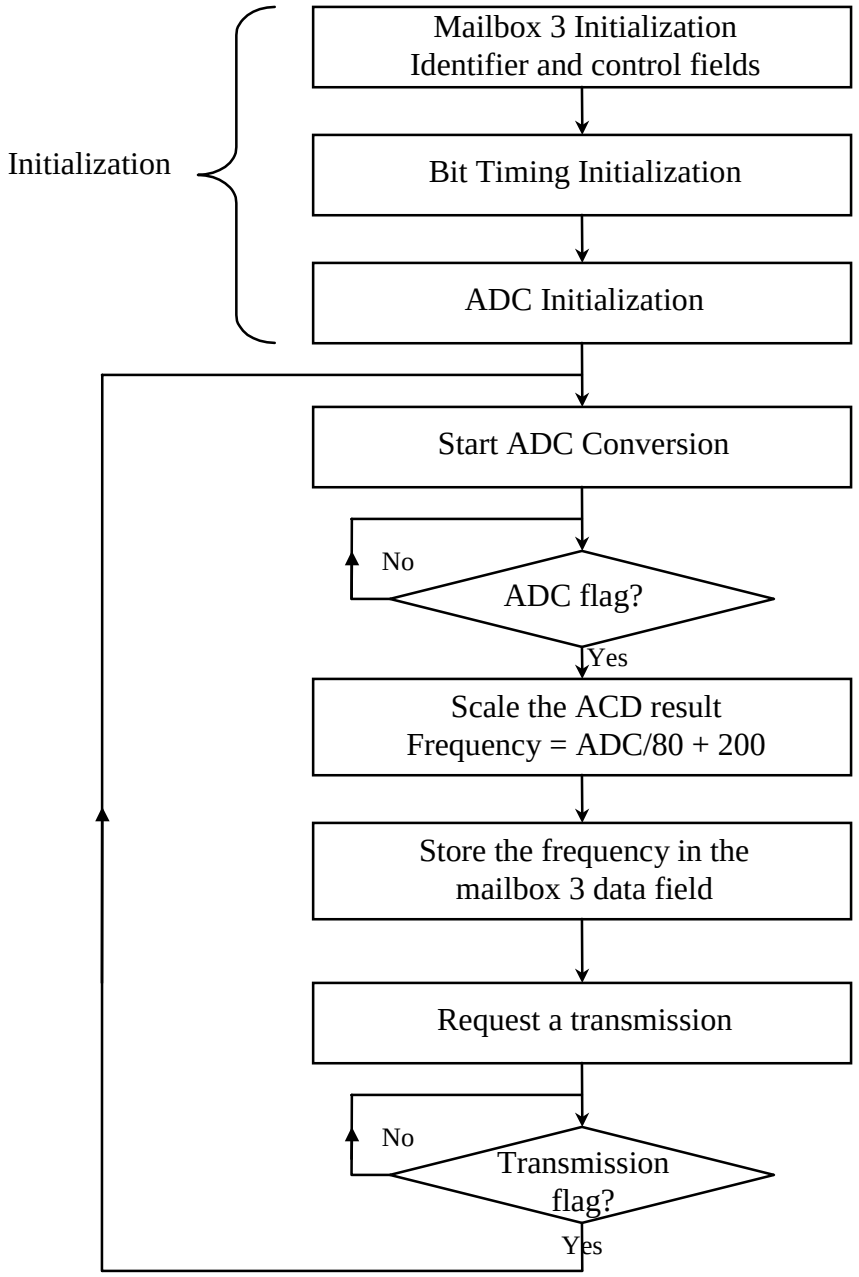
```
                    SPLK   #0000000001001000b,CANMDER
;                           ||||||||||||||||
;                           FEDCBA9876543210
;bit 0-5             Enable mailbox 3
;bit 7               0: mailbox 3 =transmit
```

To transmit the message, the TRS (Transmit request set) bit in the Transmit Control register is set. After the transmission a Transmit Acknowledge flag and the mailbox 3 interrupt flag appear.

Code:

```
                    SPLK   #0020h,CANTCR   ; Transmit request for mailbox 3

W_TA                BIT    CANTCR,2        ; Wait for transmission
                    BCND   W_TA,NTC        ; acknowledge

W_FLAG              BIT    CANIFR,4        ; Wait for interrupt flag
                    BCND   W_FLAG,NTC
                    SPLK   #2000h,CANTCR   ; Reset TA and flag
```

Figure 22. Send_frequence.asm Flow Chart



Node B: Motor Control

The second DSP handles the motor control using the target frequency value received from the CAN bus.

In this example application, an open loop control program for the AC motor is used. The detail description of the main motor control software can be found in the TI application report, *AC Induction Motor Control using Constant V/HZ Principle and Space Vector PWM Technique with TMS320C240*, Literature number SPRA284A. The motor control program works with the interrupt system. It contains two important parts: the main program and the interrupt service routine (ISR).

In the main program, different initializations are made, such as the CAN module initialization and the PWM initialization. At the end of the main program, the underflow interrupt for event manager is enabled.

In the interrupt service routine, the new PWM factors are calculated to adjust to the new frequency target received by CAN bus. Inside this ISR, the value of the frequency target is updated if the value received is different of the older value.

The CAN code can be divided into two parts: the first part is the CAN initialization where the CAN bit timing is set and mailbox 0 is initialized (Control field and identifier). This part is on the main program. The second part is inside the interrupt service routine. In case of the successful reception of a message from the CAN bus, this part will handle the copy of this message in the variable FREQ_TRGT if this value has changed. The old FREQ_TRGT value is subtracted to the value received. If the result is not zero, FREQ_TRGT is updated.

The section of the program reading the CAN bus follows:

```
CAN_RD      LDP       #0e2h           ; Load CAN registers data page (7100h)

W_FLAG      BIT       CANIFR,7        ; wait for mailbox 0 interrupt flag
            BCND      CAN_RD_END,NTC

W_RA        BIT       CANRCR,11       ; Wait for receive acknowledge
            BCND      CAN_RD_END,NTC


      SPLK        #0010h,CANRCR      ; reset RA and CANIFR

      LDP             #0E4h           ; Load CAN mailboxes data page (7200h)
      LACL            CANMBX0A        ; Load data received in Accumulator
      LDP             #04h            ; Load B0 data page
      SUB             FREQ_TRGT
      BCND            CAN_RD_END,EQ   ; If the value doesn't change
      LDP             #0E4h
      LACL            CANMBX0A        ; Freq_trgt is not updated
      LDP             #04h            ; load B0 data page (200h)
      SACL            FREQ_TRGT       ; Change FREQ_TRGT
```

```
CAN_RD_END:
```

When a successful reception from the CAN bus occurs, the mailbox interrupt flag and the Receive Message Pending bits are set. The program tests these bits (BIT instruction). If these bits are set, they are cleared by the next instruction. The data received is then copied into the accumulator and in the FREQ_TRGT variable, if this value has been changed. If the mailbox flag is not received or if there is no message pending (RMP bit), a branch to CAN_RD_END occurs.

The program used for this application can be found on the Appendix C.

Remarks

This example describes an application of motor control using CAN. The goal of this application is to demonstrate that using the CAN controller is simple and does not require a large memory space. The CAN initialization (that is used only once) requests 23 words and the CAN reading part needs 18 words. Of course, the use of a highest CAN protocol, such as CANopen, SDS, DeviceNet or CAN Kingdom, needs more memory space than this basic example application.

Applications with a higher number of nodes can also be implemented. The user decides the priority of each message and the different nodes can exchange information using the CAN bus.

Conclusion: CAN and DSP

The TMS320F241, TMS320C241 and TMS320F243 chips contain an on-chip CAN controller. CAN is a multi-master serial bus that allows an efficient transmission of data between different nodes. CAN is a flexible, reliable, robust and standardized protocol with real-time capabilities.

DSP controllers can improve the efficiency of electrical motors with higher performance and lower costs, offering a preferred solution to traditional microcontrollers and allowing more sophisticated control algorithms.

This application report shows how to program the CAN controller and how to include this program in a motor control application.

Appendix A. Header File: CAN.h

```
; CAN Registers.


CANMDER      .set 7100h      ; CAN Mailbox Direction/Enable register
CANTCR       .set 7101h      ; CAN Transmission Control Register
```

```
CANRCR      .set 7102h      ; CAN Receive Control Register
CANMCR      .set 7103h      ; CAN Master Control Register
CANBCR2     .set 7104h      ; CAN Bit Configuration Register 2
CANBCR1     .set 7105h      ; CAN Bit Configuration Register 1
CANESR      .set 7106h      ; CAN Error Status Register
CANGSR      .set 7107h      ; CAN Global Status Register
CANCEC      .set 7108h      ; CAN Transmit and Receive Err counters
CANIFR      .set 7109h      ; CAN Interrupt Flag Registers
CANIMR      .set 710ah      ; CAN Interrupt Mask Registers
CANLAM0H    .set 710bh      ; CAN Local Acceptance Mask MBx0/1
CANLAM0L    .set 710ch      ; CAN Local Acceptance Mask MBx0/1
CANLAM1H    .set 710dh      ; CAN Local Acceptance Mask MBx2/3
CANLAM1L    .set 710eh      ; CAN Local Acceptance Mask MBx2/3



; CAN Mailboxes


CANMSGID0L  .set 7200h      ; CAN Message ID for mailbox 0 (lower 16
bits)
CANMSGID0H  .set 7201h      ; CAN Message ID for mailbox 0 (upper 16
bits)
CANMSGCTRL0 .set 7202h      ; CAN RTR and DLC
CANMBX0A    .set 7204h      ; CAN 2 of 8 bytes of Mailbox 0
CANMBX0B    .set 7205h      ; CAN 2 of 8 bytes of Mailbox 0
CANMBX0C    .set 7206h      ; CAN 2 of 8 bytes of Mailbox 0
CANMBX0D    .set 7207h      ; CAN 2 of 8 bytes of Mailbox 0

CANMSGID1L  .set 7208h      ; CAN Message ID for mailbox 1 (lower 16
bits)
CANMSGID1H  .set 7209h      ; CAN Message ID for mailbox 1 (upper 16
bits)
CANMSGCTRL1 .set 720Ah      ; CAN RTR and DLC
CANMBX1A    .set 720Ch      ; CAN 2 of 8 bytes of Mailbox 1
CANMBX1B    .set 720Dh      ; CAN 2 of 8 bytes of Mailbox 1
CANMBX1C    .set 720Eh      ; CAN 2 of 8 bytes of Mailbox 1
CANMBX1D    .set 720Fh      ; CAN 2 of 8 bytes of Mailbox 1
```

```
CANMSGID2L   .set 7210h      ; CAN Message ID for mailbox 2 (lower 16
bits)
CANMSGID2H   .set 7211h      ; CAN Message ID for mailbox 2 (upper 16
bits)
CANMSGCTRL2  .set 7212h      ; CAN RTR and DLC
CANMBX2A     .set 7214h      ; CAN 2 of 8 bytes of Mailbox 2
CANMBX2B     .set 7215h      ; CAN 2 of 8 bytes of Mailbox 2
CANMBX2C     .set 7216h      ; CAN 2 of 8 bytes of Mailbox 2
CANMBX2D     .set 7217h      ; CAN 2 of 8 bytes of Mailbox 2
CANMSGID3L   .set 7218h      ; CAN Message ID for mailbox 3 (lower 16
bits)
CANMSGID3H   .set 7219h      ; CAN Message ID for mailbox 3 (upper 16
bits)
CANMSGCTRL3  .set 721Ah      ; CAN RTR and DLC
CANMBX3A     .set 721Ch      ; CAN 2 of 8 bytes of Mailbox 3
CANMBX3B     .set 721Dh      ; CAN 2 of 8 bytes of Mailbox 3
CANMBX3C     .set 721Eh      ; CAN 2 of 8 bytes of Mailbox 3
CANMBX3D     .set 721Fh      ; CAN 2 of 8 bytes of Mailbox 3

CANMSGID4L   .set 7220h      ; CAN Message ID for mailbox 4 (lower 16
bits)
CANMSGID4H   .set 7221h      ; CAN Message ID for mailbox 4 (upper 16
bits)
CANMSGCTRL4  .set 7222h      ; CAN RTR and DLC
CANMBX4A     .set 7224h      ; CAN 2 of 8 bytes of Mailbox 4
CANMBX4B     .set 7225h      ; CAN 2 of 8 bytes of Mailbox 4
CANMBX4C     .set 7226h      ; CAN 2 of 8 bytes of Mailbox 4
CANMBX4D     .set 7227h      ; CAN 2 of 8 bytes of Mailbox 4


CANMSGID5L   .set 7228h      ; CAN Message ID for mailbox 5 (lower 16
bits)
CANMSGID5H   .set 7229h      ; CAN Message ID for mailbox 5 (upper 16
bits)
CANMSGCTRL5  .set 722Ah      ; CAN RTR and DLC
CANMBX5A     .set 722Ch      ; CAN 2 of 8 bytes of Mailbox 5
CANMBX5B     .set 722Dh      ; CAN 2 of 8 bytes of Mailbox 5
CANMBX5C     .set 722Eh      ; CAN 2 of 8 bytes of Mailbox 5
CANMBX5D     .set 722Fh      ; CAN 2 of 8 bytes of Mailbox 5
```

Appendix B. CAN Abbreviations

Notation:	Signification:	Register:	Bit Nb:
AA:	Abort Acknowledge	TCR	11:8
AAIF:	Abort Acknowledge Interrupt Flag	IFR	5
AAIM:	Abort Acknowledge Interrupt Mask	IMR	5
AAM:	Auto Answer Mode	MSGIDn	13
ABO:	Auto Bus On	MCR	7
ACKE:	Acknowledge Error	ESR	3
AME:	Acceptance Mask Enable	MSGIDn	14
BEF:	Bit Error Flag	ESR	7
BO:	Bus Off Status	ESR	2
BOIF:	Bus Off Interrupt Flag	IFR	2
BOIM:	Bus Off Interrupt Mask	IMR	2
BRP:	Baud Rate Prescaler	BCR2	7:0
CCE:	Change Configuration Enable	GSR	4
CCR:	Change Configuration Request	MCR	12
CDR:	Change Data Field Request	MCR	12
CRCE:	CRC Error	ESR	5
DBO:	Data Byte Order	MCR	10
DLC:	Data Length Code	MSGCTRLn	3:0
EIL:	Error Interrupt Priority Level	IMR	7
EP:	Error Passive Status	ESR	1
EPIF:	Error Passive Interrupt Flag	IFR	1
EPIM:	Error Passive Interrupt Mask	IMR	1
EW:	Warning Status	ESR	0
FER:	Form Error Flag	ESR	8
IDE:	Identifier Extension	MSGIDn	15
LAMI:	Local Acceptance Mask Identifier	LAM	15
MBNR:	Mailbox Number	MCR	1:0
ME:	Mailbox Enable	MDER	5:0
MD:	Mailbox Direction	MDER	7:6
MIF:	Mailbox Interrupt Flag	IFR	13:8
MIL:	Mailbox Interrupt Priority Level	IMR	15
MIM:	Mailbox Interrupt Mask	IMR	13:8
OPC:	Overwrite Protection Control	RCR	3:0
PDA:	Power Down Mode Acknowledge	GSR	3
PDR:	Power Down Mode Request	MCR	11
REC:	Receive Error Counter	CEC	7:0
RFP:	Remote Frame Pending	RCR	15:12
RM:	Receive Mode	GSR	1
RML:	Receive Message Lost	RCR	11:8
RMLIF:	Receive Message Lost Interrupt Flag	IFR	6
RMLIM:	Receive Message Lost Interrupt Mask	IMR	6
RMP:	Receive Message Pending	RCR	7:4
RTR:	Remote Transmission Request	MSGCTRLn	4
SA1:	Stuck at dominant Error	ESR	6
SAM:	Sample Point Setting	BCR1	7
SBG:	Synchronization on Both Edge	BCR1	10
SER:	Stuff Error	ESR	4

SJW:	Synchronization Jump Width	BCR1	9:8
SMA:	Suspend Mode Acknowledge	GSR	5
STM:	Self Test Mode	MCR	6
SUSP:	Action on Emulator Suspend	MCR	13
TA:	Transmission Acknowledge	TCR	15:12
TEC:	Transmit Error Counter	CEC	15:8
TM:	Transmit Mode	GSR	0
TRS:	Transmission Request Set	TCR	4:7
TRR:	Transmission Request Reset	TCR	0:3
WDIF:	Write Denied Interrupt Flag	IFR	4
WDIM:	Write Denied Interrupt Mask	IMR	4
WLIF:	Warning Level Interrupt Flag	IFR	0
WLIM:	Warning Level Interrupt Mask	IMR	0
WUBA:	Wake Up on Bus Activity	MCR	9
WUIF:	Wake Up Interrupt Flag	IFR	3
WUIM:	Wake Up Interrupt Mask	IMR	3

Appendix C. Programs Used for Motor Control Application

NODE A: Send_Frequence.asm

```
;***************************************************************************
;* File Name:         Send_Frequence.asm                                   *
;* Originator:        Claire Monnet    (Texas Instruments)                 *
;* Target Sys:        TMS320F243 EVM                                       *
;* Description:       A voltage is sent to the F243 EVM, transformed  *
;*                     by ADC, scaled between 200 and 400 and store in    *
;*                     mailbox 3 before to be sent in the CAN bus.        *
;* May 29th 1998                                                           *
;***************************************************************************

;***************            Debug directives           *********************
                  .def   GPR0      ;General purpose register
                  .bss   GPR0,1

;***************          Peripheral Registers         *********************
                  .include  "X24x.h"

;***************           Constant definitions         ********************
      DP_PF1          .set   0E0h      ; Page 1 of peripheral file (7000h/80h
      DP_CAN          .set   0E2h       ; CAN  Registers page (7100h)
      DP_CAN2         .set   0E4h       ; CAN RAM page (7200h)
```

```
;***************         M A C R O – Definitions        *******************
      KICK_DOG        .macro          ; Watchdog reset macro
                      LDP    #00E0h
                      SPLK   #05555h, WDKEY
                      SPLK   #0AAAAh, WDKEY
                      LDP    #0h
                      .endm

;**************          Vector address declarations        ***************

                      .global _c_int0
                      .sect  ".vectors"

      RSVECT          B    _c_int0 ; PM 0   Reset Vector 1
      INT1            B    PHANTOM ; PM 2 Int level 1     4
      INT2            B    PHANTOM ; PM 4 Int level 2     5
      INT3            B    PHANTOM ; PM 6 Int level 3     6
      INT4            B    PHANTOM ; PM 8 Int level 4     7
      INT5            B    PHANTOM ; PM A Int level 5     8
      INT6            B    PHANTOM ; PM Cint level 6      9
      RESERVED        B    PHANTOM ; PM E (Analysis Int)  10
      SW_INT8         B    PHANTOM ; PM 10   User S/W int -
      SW_INT9         B    PHANTOM ; PM 12   User S/W int -
      SW_INT10        B    PHANTOM ; PM 14   User S/W int -
      SW_INT11        B    PHANTOM ; PM 16   User S/W int -
      SW_INT12        B    PHANTOM ; PM 18   User S/W int -
      SW_INT13        B    PHANTOM ; PM 1A   User S/W int -
      SW_INT14        B    PHANTOM ; PM 1C   User S/W int -
      SW_INT15        B    PHANTOM ; PM 1E   User S/W int -
      SW_INT16        B    PHANTOM ; PM 20   User S/W int -
      TRAP            B    PHANTOM ; PM 22   Trap vector     -
      NMI             B    PHANTOM ; PM 24   Non maskable Int   3
      EMU_TRAP        B    PHANTOM ; PM 26   Emulator Trap   2
      SW_INT20        B    PHANTOM ; PM 28   User S/W int -
      SW_INT21        B    PHANTOM ; PM 2A   User S/W int -
      SW_INT22        B    PHANTOM ; PM 2C   User S/W int -
```

```
      SW_INT23        B    PHANTOM ; PM 2E   User S/W int -

;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
                  .text
_c_int0:
                  SETC      INTM            ; Disable interrupts
                  CLRC      SXM             ; Clear Sign Extension Mode
                  CLRC      OVM             ; Reset Overflow Mode

                  LDP       #DP_PF1
                  LACC      #006Fh
                  SACL      WDCR            ; WD control reg. at 7028h
                  KICK_DOG


;*************          Configure Wait State Generator       **************

                  SPLK  #0,61h
                  OUT   61h,0ffffh


;**************          Configure the shared pins          ***************

                  LDPK  #225
                  SPLK  #0FFFFH,OCRA
                  SPLK  #0FFF3H,OCRB


;*************          CAN  Mailbox Initialization           **************

                  LDP   #DP_CAN
                  SPLK  #0000000000000000b,CANMDER  ; disable each mailbox
                  SPLK  #0000000100000000b,CANMCR
      ;                    ||||||||||||||||
      ;                    FEDCBA9876543210
```

```
;bit 8             CDR: Change data field request

                   LDP   #DP_CAN2
                   SPLK  #1111111111111111b,CANMSGID3H
;                         ||||||||||||||||
;                         FEDCBA9876543210

;bit 0-12          Upper 13 bits of extended identifier
;bit 13            Auto answer mode bit
;bit 14            Acceptance mask enable bit
;bit 15            Identifier extension bit

                   SPLK  #1111111111111111b,CANMSGID3L
;bit 0-15          lower part of extended identifier

                   SPLK  #0000000000000010b,CANMSGCTRL3
;                         ||||||||||||||||
;                         FEDCBA9876543210
;bit 0-3           Data length code. 0010 = 2 bytes
;bit 4             0: data frame

                   LDP   #DP_CAN

                   SPLK  #0000000000000000b,CANMCR
;                         ||||||||||||||||
;                         FEDCBA9876543210

;bit 8                CDR: Change data field request

                   SPLK  #0000000001001000b,CANMDER
;                         ||||||||||||||||
;                         FEDCBA9876543210

;bit 0-5           Enable mailbox 3
;bit 7             0: mailbox 3 configured as a transmit mailbox
```

```
;*************        CAN Bit Timing Configuration          **************

                  SPLK  #0001000000000000b,CANMCR
;                        ||||||||||||||||
;                        FEDCBA9876543210
;bit 12            Change configuration register

W_CCE             BIT   CANGSR,#0Bh        ; Wait for Change configuration
                  BCND  W_CCE,NTC          ; enable

                  SPLK  #0000000000000000b,CANBCR2
; bit 0-7          Baud rate prescaler

                  SPLK  #0000010101010111b,CANBCR1
;                        ||||||||||||||||
;                        FEDCBA9876543210
; bit 0-2          TSEG1
; bit 3-6          TSEG2
; bit 7            Sample point setting (1: 3 times, 0: once)
; bit 8-A          Synchronization jump width
; bit B            Synchronization on falling edge
; bit C-F          Reserved

                  SPLK  #0000010000000000b,CANMCR
;                        ||||||||||||||||
;                        FEDCBA9876543210
; bit 10           1: data Byte order: 0,1 first
; bit 12           0: normal mode

W_NCCE            BIT   CANGSR,#0Bh     ; Wait for Change configuration
                  BCND  W_NCCE,TC       ; Disable


;************          Configure ADC Control 2 register        ************

                  LDP   #DP_PF1
                  SPLK  #0000000000000000b, ADCTRL2
;                        ||||||||||||||||
```

```
;                         FEDCBA9876543210
; bit 0-2          000 Prescaler value
; bit 3-4          FIFO2 status
; bit 5            Reserved
; bit 6-7          FIFO1 status
; bit 8            Reserved
; bit 9 0          Mask external SOC input
; bit A            0  Mask EV SOC input
; bit B-F          Reserved
;*************************************************************************
;                           Beginning of the loop
;*************************************************************************
LOOP

                  SPLK  #1101000110000001b, ADCTRL1  ; Start ADC conversion
;                        ||||||||||||||||
;                     FEDCBA9876543210
; bit 0            1  Start of conversion
; bit 1-3          000 Channel 0 address
; bit 7            0  End of convert
; bit 8            1  Interrupt flag - write 1 to clear
; bit 9            1  Interrupt mask - enable with 1, mask 0
; bit A            0  Continuous run mode disabled
; bit C            1  Enable ADC1
; bit D            1  Immediate start - 0 = no action
; bit E            1  Free run - ignore suspend
; bit F            1  Soft - Not applicable with bit E = 1


READ_ADC          LACL  ADCTRL1               ; wait until end of
                  AND   #0000000100000000B    ; conversion flag
; bit 8            1  Interrupt flag
                  SUB   #0000000100000000B
                  BCND  READ_ADC, NEQ
```

```
;***************        Scale  the ADC value      ******************
; The result of the conversion has to be scaled between 200 (0 volt)
; and 400 (5 volt)
; Frequence_target = ADC_value/80h + 200

                  SFR
                  SFR
                  SFR
                  SFR
                  SFR
                  SFR
                  SFR                   ; shift 7 times (division by 80h)
                  ADD   #200h           ; add 200h
                  LDP   #04h
                  SACL  GPR0            ; result stored in GPR0


;**********         Store frequency in the mailbox 3            **********

                  LDP   #DP_CAN
                  SPLK  #0000000000000000b,CANMDER
;bit 0-5           Disable each mailbox

                  SPLK  #0000000100000000b,CANMCR
;                        ||||||||||||||||
;                        FEDCBA9876543210
;bit 8             CDR: Change data field request

                  LDP   #04h
                  LACL  GPR0            ; Load frequency value inside
                  LDP   #DP_CAN2        ; the CAN mailbox 3
                  SACL  CANMBX3A
                  LDP   #DP_CAN
                  SPLK  #0000010000000000b,CANMCR
;                        ||||||||||||||||
;                        FEDCBA9876543210
;bit 8             CDR: Change data field request
```

```
;bit 10           Data byte order. First sent:0,1

                  SPLK  #0000000001001000b,CANMDER
;                        ||||||||||||||||
;                        FEDCBA9876543210
;bit 0-5          Enable mailbox 3
;bit 7            0: mailbox 3 =transmit

;***********       Transmit data to the CAN Bus          **********

                  SPLK  #0020h,CANTCR   ; Transmit request for mailbox 3

W_TA              BIT   CANTCR,2        ; Wait for transmission
                  BCND  W_TA,NTC        ; acknowledge

W_FLAG            BIT   CANIFR,4        ; Wait for interrupt flag
                  BCND  W_FLAG,NTC
                  SPLK  #2000h,CANTCR   ; Reset TA and flag

                  B     LOOP            ; Branch to the beginning of
                                        ; the loop.

;*********************************************************************
; MAIN CODE - ends here
;*********************************************************************

;=====================================================================
; ISR:  PHANTOM          TYPE: ISR
;=====================================================================

PHANTOM           RET                   ; return
```

NODE B: CAN_application.asm

```
;*************************************************************************
; File Name:       CAN_application.ASM
; Originator:      David Figoli (Texas Instruments)
;                  updated by Claire Monnet (Texas Instruments)
; Target Sys:      TMS320F243 EVM Board + Spectrum digital's Inverter
; Description:     Open loop program
;                  This is an implementation of 3 phase Space vector PWM
;                  running the F243 device. External frequency control is
;                  provided by CAN bus.
; Last Update:     01 may 1998
;*************************************************************************
;-------------------------------------------------------------------------
; Debug directives
;-------------------------------------------------------------------------
      .def         GPR0      ;General purpose registers.
      .def         GPR1
      .def         GPR2
      .def         ALPHA
      .def         STEP_ANGLE
      .def         FREQ_SETPT
      .def         ENTRY_NEW
      .def         ENTRY_OLD
      .def         dx
      .def         dy
      .def         Ta
      .def         Tb
```

```
      .def        Tc
      .def        V
      .def        SPEED_HI
      .def        SPEED_LO
      .def        SPEED_fb
      .def        SPEED_sp
      .def        BCAVG
      .def        PCNT_SETPT

      .include    x24x.h

;-------------------------------------------------------------------------
; Constant Declarations
;-------------------------------------------------------------------------
; Used by the SBIT0 & SBIT1 Macro
B15_MSK .set      8000h    ;Bit Mask for 15
B14_MSK .set      4000h    ;Bit Mask for 14
B13_MSK .set      2000h    ;Bit Mask for 13
B12_MSK .set      1000h    ;Bit Mask for 12
B11_MSK .set      0800h    ;Bit Mask for 11
B10_MSK .set      0400h    ;Bit Mask for 10
B9_MSK  .set      0200h    ;Bit Mask for 9
B8_MSK  .set      0100h    ;Bit Mask for 8
B7_MSK  .set      0080h    ;Bit Mask for 7
B6_MSK  .set      0040h    ;Bit Mask for 6
B5_MSK  .set      0020h    ;Bit Mask for 5
B4_MSK  .set      0010h    ;Bit Mask for 4
B3_MSK  .set      0008h    ;Bit Mask for 3
B2_MSK  .set      0004h    ;Bit Mask for 2
B1_MSK  .set      0002h    ;Bit Mask for 1
B0_MSK  .set      0001h    ;Bit Mask for 0

WSGR    .set      0FFFFh
DP_PF1  .set      0E0h     ;page 1 of peripheral file (7000h/80h)
DP_PF2  .set      0E1h     ;page 2 of peripheral file (7080h/80h)
DP_PF3  .set      0E2h     ;page 3 of peripheral file (7100h/80h)
DP_EV   .set      0E8h     ;EV register data mem page (7400h/80h)
```

```
DP_CAN  .set       0E2h      ; CAN Registers (7100h)
DP_CAN2 .set       0E4h      ; CAN RAM (7200h)


;Space vector PWM constants
;--------------------------
F1                .set  0256     ;Low Freq point on profile(=15Hz)
F2                .set  1024     ;High Freq point on profile(=60Hz)
VF_SLOPE          .set  15291    ;Volts/Hz slope 1.87 in Q13 format
INTERCEPT         .set  00546    ;Line equation intercept 0.07 in Q13
Vmax              .set  032767   ;0.99999..  in Q15
Vmin              .set  09830    ;0.40000..  in Q15
BCNT_MAX          .set  100      ;100x40uS=0.004 Sec depress to be valid
RMP_DLY_MAX       .set  100      ;100x40uS=0.004 sec between steps.
BC_SIZE           .set  50       ;Box car average size of 50
BC_BUF_STRT       .set  300h     ;Start of BC buffer

;-----------------------------------------------------------------------
; Variable Declarations for on chip RAM Block B0
;-----------------------------------------------------------------------
      .bss        GPR0,1       ;General purpose registers.
      .bss        GPR1,1
      .bss        GPR2,1
      .bss        FREQ_SETPT,1 ;Value from 0 --> 255
      .bss        FREQ_TRGT,1     ;Frequency Target value 0 --> 255
      .bss        XF_STATE,1      ;State of XF pin (i.e. a Flag)
      .bss        B1_CNT,1     ;B1 button counter (Inc Freq)
      .bss        B2_CNT,1     ;B2 button counter (Dec Freq)
      .bss        RMP_DLY_CNT,1   ;Ramp rate in adjusting to Target freq.
      .bss        REPRESS_DLY,1   ;Forced delay between Re-presses.
      .bss        S_TABLE,1    ;Data address to store Sine table addr.
      .bss        ALPHA,1
      .bss        STEP_ANGLE,1
      .bss        ENTRY_NEW,1
      .bss        ENTRY_OLD,1
      .bss        SINVAL,1
      .bss        SR_ADDR,1
```

```
       .bss         SECTOR_PTR,1

       .bss         SPEED_HI,1
       .bss         SPEED_LO,1
       .bss         SPEED_fb,1
       .bss         SPEED_sp,1
       .bss         BCAVG,1
       .bss         PCNT_SETPT,1

       .bss         dx,1
       .bss         dy,1
       .bss         T,1
       .bss         Ta,1
       .bss         Tb,1
       .bss         Tc,1
       .bss         V,1
       .bss         vf_slope,1
       .bss         FREQ_3BIT,1
       .bss         LED_MASK,1
       .bss         mSEC,1

;-------------------------------------------------------------------------
; M A C R O - Definitions
;-------------------------------------------------------------------------
SBIT0              .macro
                   DMA, MASK ;Clear bit Macro
                   LACC  DMA
                   AND#(0FFFFh-MASK)
                   SACL  DMA
                   .endm

SBIT1              .macro
                   DMA, MASK ;Set bit Macro
                   LACC  DMA
                   OR #MASK
                   SACL  DMA
                   .endm
```

```
KICK_DOG          .macro          ;Watchdog reset macro
                  LDP#00E0h
                  SPLK  #05555h, WDKEY
                  SPLK  #0AAAAh, WDKEY
                  LDP#0h
                  .endm

POINT_PG0         .macro
                  LDP#00h
                  .endm

POINT_B0          .macro
                  LDP#04h
                  .endm

POINT_PF1         .macro
                  LDP#0E0h
                  .endm

POINT_PF2         .macro
                  LDP#0E1h
                  .endm

POINT_EV          .macro
                  LDP#0E8h
                  .endm

;-------------------------------------------------------------------------
; Vector address declarations
;-------------------------------------------------------------------------
                  .global _c_int0
                  .sect  ".vectors"

RSVECT     B      _c_int0   ; PM 0   Reset Vector     1
INT1       B      PHANTOM  ; PM 2   Int level 1       4
INT2       B      PWM_ISR  ; PM 4   Int level 2       5
```

```
INT3      B       PHANTOM  ; PM 6   Int level 3       6
INT4      B       PHANTOM  ; PM 8   Int level 4       7
INT5      B       PHANTOM  ; PM A   Int level 5       8
INT6      B       PHANTOM  ; PM C   Int level 6       9
RESERVED  B       PHANTOM  ; PM E   (Analysis Int)    10
SW_INT8   B       PHANTOM  ; PM 10  User S/W int      -
SW_INT9   B       PHANTOM  ; PM 12  User S/W int      -
SW_INT10  B       PHANTOM  ; PM 14  User S/W int      -
SW_INT11  B       PHANTOM  ; PM 16  User S/W int      -
SW_INT12  B       PHANTOM  ; PM 18  User S/W int      -
SW_INT13  B       PHANTOM  ; PM 1A  User S/W int      -
SW_INT14  B       PHANTOM  ; PM 1C  User S/W int      -
SW_INT15  B       PHANTOM  ; PM 1E  User S/W int      -
SW_INT16  B       PHANTOM  ; PM 20  User S/W int      -
TRAP      B       PHANTOM  ; PM 22  Trap vector       -
NMI       B       PHANTOM  ; PM 24  Non maskable Int   3
EMU_TRAP  B       PHANTOM  ; PM 26  Emulator Trap      2
SW_INT20  B       PHANTOM  ; PM 28  User S/W int      -
SW_INT21  B       PHANTOM  ; PM 2A  User S/W int      -
SW_INT22  B       PHANTOM  ; PM 2C  User S/W int      -
SW_INT23  B       PHANTOM  ; PM 2E  User S/W int      -

;=========================================================================
; M A I N   C O D E  - starts here
;=========================================================================
                  .text
_c_int0:
      POINT_PG0
      SETC        INTM              ;Disable interrupts
      SPLK        #0h, IMR          ;Mask all Ints
      SPLK        #0FFh, IFR        ;Clear all Int Flags
      CLRC        SXM               ;Clear Sign Extension Mode
      CLRC        OVM               ;Reset Overflow Mode
      CLRC        CNF               ;Configure Block B0 to Data memory.
      POINT_B0
      SPLK        #04h, GPR0        ;Set 0 wait states for XMIF
      OUT         GPR0, WSGR
```

```
      POINT_PF1
      SPLK        #40C0h,SCSR       ;CLKOUT=CPUCLK

;Comment out if WD is to be active
      SPLK        #006Fh, WDCR      ;Disable WD if VCCP=5V
      KICK_DOG


;*********************************************************************
; Activate Lab drive and configure CAN pins
;*********************************************************************

      LDPK        #225
      SPLK        #0H,OCRA
      SPLK        #00e0H,OCRB
      SPLK        #2020h, PCDATDIR      ; IOPC5 pin high
      SPLK        #1000h,PDDATDIR       ; IOPD4 pin low

      POINT_PF1
;---------------------------------------------------------------------
; CAN Initialization
;---------------------------------------------------------------------

      LDP         #DP_CAN

      SPLK        #1001111111111110b,CANLAM0H     ; Set local acceptance
      mask
      SPLK        #1111111111111111b,CANLAM0L     ; 1:don't care
      SPLK        #03f7fh,CANIMR                  ; Set interrupt mask

      SPLK        #0000000000000000b,CANMDER
;                  ||||||||||||||||
;                  FEDCBA9876543210
;bit 0-5           disable each mailbox

      SPLK        #0000000100000000b,CANMCR
;                  ||||||||||||||||
```

```
;                  FEDCBA9876543210
;bit 8             CDR: Change data field request

       LDP             #DP_CAN2

       SPLK         #1111111111111111b,CANMSGID0H
;                   ||||||||||||||||
;                   FEDCBA9876543210
;bit 0-15          lower part of extended identifier

       SPLK         #1111111111111011b,CANMSGID0L
;                   ||||||||||||||||
;                   FEDCBA9876543210
;bit 0-12          upper 13 bits of extended identifier
;bit 13            Auto answer mode bit
;bit 14            Acceptance mask enable bit
;bit 15            Identifier extension bit


       LDP          #DP_CAN
       SPLK         #0000000000000000b,CANMCR
;                   ||||||||||||||||
;                   FEDCBA9876543210
;bit 8             CDR: Change data field request

       SPLK         #0000000000000001b,CANMDER
;                   ||||||||||||||||
;                   FEDCBA9876543210
;bit 0-5           enable  mailbox 0


       SPLK         #0011000000000000b,CANMCR
;                   ||||||||||||||||
;                   FEDCBA9876543210
;bit 12            Change configuration request

W_CCE              BIT   CANGSR,#0Bh        ; Wait for Change config Enable
```

```
                  BCND  W_CCE,NTC

      LDP         #DP_CAN
      SPLK        #0000000000000000b,CANBCR2
;                  ||||||||||||||||
;                  FEDCBA9876543210
; bit 0-7          Baud rate prescaler
; bit 8-15         Reserved

      SPLK        #0000010101010111b,CANBCR1
;                  ||||||||||||||||
;                  FEDCBA9876543210

; bit 0-2          TSEG1
; bit 3-6          TSEG2
; bit 7            Sample point setting (1: 3 times, 0: once)
; bit 8-A          Synchronization jump width
; bit B            Synchronization on falling edge
; bit C-F          Reserved

      SPLK        #0010000000000000b,CANMCR
;                  ||||||||||||||||
;                  FEDCBA9876543210
;bit 12            Change configuration request
;
W_NCCE            BIT   CANGSR,#0Bh        ; Wait for Change configuration
                  BCND  W_NCCE,TC          ; disable

;------------------------------------------------------
; Initialize Counter, Step parameters, & AR pointers
;------------------------------------------------------
SV_PWM:
      POINT_B0
      SPLK        #STABLE, S_TABLE    ;Used only to save a cycle
      SPLK        #VF_SLOPE, vf_slope ;Used later for multiply.

      LACC        #0h             ;Start at 0 deg.
```

```
       SACL         ALPHA           ;Clear ANGLE integrator

       LACC         #0h             ;Start at 0 deg.
       SACL         ENTRY_NEW       ;Clear Sine Table Pointer

       LACC         #0h             ;Start at sector 0
       SACL         SECTOR_PTR         ;Init Sector table index pointer

       LACC         #1040           ;Use 41.6 uS period (1040 x 40nS)
                                              ;i.e. 24.039 KHz
      SACL          T               ;Init the PWM period

       LACC         #0512           ;Use ~30Hz as Frequency
       SACL         FREQ_SETPT         ;Init the angular speed
       SACL         FREQ_TRGT       ;same speed for Target value

       LAR          AR1, #CMPR1        ;Init Timer Comp reg pointers
       LAR          AR2, #CMPR2
       LAR          AR3, #CMPR3
       MAR          *, 1

;-------------------------------------------------------------
;EV Config starts here.
;-------------------------------------------------------------
EV_CONFIG:
;Configure all I/O pins to I/O function pins
       POINT_PF2
       SPLK         #0FFFFh,OCRA
       SPLK         #0h,OCRB
       EV_LP
       SPLK         #0C0Ch,PADATDIR       ;A3,A2=O/P,  A1,A0=I/P, A3,A2=1,1
       POINT_B0
       SPLK         #500, mSEC            ;Wait approx 0.5 sec
       CALL         mS_DELAY
       POINT_PF2
       SPLK         #00000h,PBDATDIR      ;Configure Port B as I/P
```

```
; Mask all EV interrupts
; (prevent stray PDPINTs from disabling compare outputs)
      POINT_EV                          ; DP => EV Registers
      SPLK        #00000h,EVIMRA        ; Mask all Group A interrupt flags
      SPLK        #00000h,EVIMRB        ; Mask all Group B interrupt flags
      SPLK        #00000h,EVIMRC        ; Mask all Group C interrupt flags

; Clear EV control registers
      SPLK        #00000h,T1CON         ; GP Timer 1 control
      SPLK        #00000h,T2CON         ; GP Timer 2 control
      SPLK        #00000h,DBTCON        ; Dead band control register
      SPLK        #00000h,COMCON        ; Compare control
      SPLK        #00000h,CAPCON        ; Capture control
      SPLK        #000FFh,CAPFIFO       ; Capture FIFO status bits

;-------------------------------------------------------
; Clear all EV interrupts before operation starts
;-------------------------------------------------------
      SPLK        #0FFFFh,EVIFRA  ; Clear all Group A interrupt flags
      SPLK        #0FFFFh,EVIFRB  ; Clear all Group B interrupt flags
      SPLK        #0FFFFh,EVIFRC  ; Clear all Group C interrupt flags

;-------------------------------------------------------
; Setup GP Timers
;-------------------------------------------------------
; Initialize counter registers
      SPLK        #00000h,T1CNT   ; GP Timer 1 counter
      SPLK        #00000h,T2CNT   ; GP Timer 2 counter


; Initialize period registers
      POINT_B0
      LACC        T
      POINT_EV
      SACL        T1PR              ; GP Timer 1 period
      SPLK        #07FFFh, T2PR     ; Limit counter values to +ve only
; Initialize compare registers
```

```
      SPLK        #00100,T1CMPR      ; GP Timer 1 comp value
      SPLK        #00250,T2CMPR      ; GP Timer 2 Comp Value

;-------------------------------------------------------
; Configure GP Timer registers
;-------------------------------------------------------
      SPLK        #0000000001101010b,GPTCON
;                  ||||!!!!!!!!!!!!
;                  5432109876543210

;                  5432109876543210
;                  ||||!!!!!!!!!!!!
      SPLK        #1001010101000010b,T2CON  ;Cont Up, /32,
      SPLK        #1001000001000000b,T1CON  ;Asym

;-------------------------------------------------------
; Configure Full Compare registers
;-------------------------------------------------------
      POINT_EV
      SPLK        #00100,CMPR1          ; F. Comp U 1 compare value
      SPLK        #00250,CMPR2          ; F. Comp U 2 compare value
      SPLK        #00400,CMPR3          ; F. Comp U 3 compare value

;Start the "Ball rolling" with Timers & Compare units.

      POINT_EV
      SPLK        #0000111111101000b,DBTCON
      SPLK        #0000011001100110b,ACTR     ; Full Action Cntl
      SPLK        #0010001000000000b,COMCON   ; Compare Cntl
      SPLK        #1010001000000000b,COMCON   ; Compare Cntl
;                  ||||!!!!!!!!!!!!
;                  5432109876543210
;-------------------------------------------------------
; Enable appropriate Interrupts - EV & DSP core
;-------------------------------------------------------
      POINT_EV
      SPLK        #0000001000000000b,EVIMRA   ; Enable Underflow Int
```

```
;                  ||||!!!!!!!!!!!!!           ; for Evt Mgr
;                  5432109876543210

       SPLK        #0FFFFh,EVIFRA  ; Clear all Group A interrupt flags
       SPLK        #0FFFFh,EVIFRB  ; Clear all Group B interrupt flags
       SPLK        #0FFFFh,EVIFRC  ; Clear all Group C interrupt flags

       POINT_PG0
       SPLK        #0000000000000010b,IMR   ; Enable Int lvl 2 for
;                  ||||!!!!!!!!!!!!!        ; DSP core  & Emu Int.
;                  5432109876543210

       SPLK        #0FFFFh, IFR          ; Clear any pending Ints
       CLRC        INTM                  ; Enable global Ints

       ;Init for Capture
       LAR         AR4, #CAP1FIFO           ; Point to Capture FIFO reg
       LAR         AR5, #BC_BUF_STRT        ; Point to start of BC buffer

       MAIN        B  MAIN

;=============================================================================
; Routine Name: P W M _ I S R             Routine Type: ISR
;
; Description:
;   - Load Timer compare regs with previously calculated Ta, Tb, Tc
;   - Calculate new Angle (alpha)
;   - Deduce dx & dy
;   - Determine current Sector Pointer
;   - Do Calculated Branch to Sector Subroutine
; Last Update:     MAY 98
;=============================================================================
PWM_ISR:


       POINT_EV
       SPLK        #0FFFFh,EVIFRA     ; Clear all Group A interrupt flags
```

```
      ; Calculate Speed Setpoint
      POINT_B0
      LT              FREQ_SETPT          ; SPEED_sp
      MPY             #22                 ; = FREQ_SETPT x 22
      PAC
      SACL        SPEED_sp

      LACC        FREQ_SETPT              ; Load FREQ_SETPT
      SACL        STEP_ANGLE              ; Update new angle increment

      ; Calculate new Voltage V based on Volts/Freq. profile
PROFILE1          LACC  FREQ_SETPT
                  SUB   #F1               ; Is Freq.<=F1
                  BCND  PROFILE2, GT
                  LACC  #Vmin
                  SACL  V                 ; V is in Q15
                  B     NEW_ALPHA

PROFILE2          LACC  FREQ_SETPT
                  SUB   #F2
                  BCND  PROFILE3, GT

                  LACC  FREQ_SETPT,4      ;Convert FCV to Q15 format
                  SACL  GPR0
                  LT    GPR0
                  MPY   vf_slope          ; P = vf_slope * FCV
                  PAC                     ; Q13    * Q15 --> Q28
                  SACH  V,1               ;convert result to Q13 format
                  LACC  V
                  ADD   #INTERCEPT        ; INTERCEPT is in Q13
                  SACL  V,2               ;result V is in Q15 & <1.0
                  B     NEW_ALPHA

PROFILE3          LACC  #Vmax
                  SACL  V                 ;V is in Q15
```

```
      ; Calculate new angle ALPHA
NEW_ALPHA          LACC   ENTRY_NEW
                   SACL   ENTRY_OLD
                   LACC   ALPHA
                   ADD    STEP_ANGLE         ;Inc angle.
                   SACL   ALPHA
                   LACC   ALPHA,8
                   SACH   ENTRY_NEW
                   LACC   S_TABLE
                   ADD    ENTRY_NEW
                   TBLR   dy              ;dy=Sin(ALPHA)

                   LT     dy              ;dy is in Q15
                   MPY    V               ;V  is in Q15
                   PAC                    ;P = V * dy
                   SACH   dy,1            ;shift 1 to restore Q15 format

                   LACC   dy,11           ;scale for 10 bit integer
      resolution
                   SACH   dy              ;Save in Q0 format
                   LACC   #0FFh           ;ACC=60 deg
                   SUB    ENTRY_NEW
                   ADD    S_TABLE
                   TBLR   dx              ;dx=Sin(60-ALPHA)

                   LT     dx
                   MPY    V
                   PAC                    ;P = V * dx
                   SACH   dx,1            ;shift 1 to restore Q15 format

                   LACC   dx,11           ;scale for 10 bit integer
      resolution
                   SACH   dx              ;Save in Q0 format

      ;Determine which Sector
                   LACC   ENTRY_NEW
                   SUB    ENTRY_OLD
                   BCND   BRNCH_SR, GEQ      ;If negative need to change
      Sector
```

```
                                        ;If positive continue
MODIFY_SEC_PTR      LACC   SECTOR_PTR
                    SUB    #05h             ;Check if at last sector (S6)
                    BCND   PISR1,EQ         ;If yes, re-init AR1= 1st
       Sector (S1)
                    LACC   SECTOR_PTR       ;If no, select next Sector (Sn-
       >Sn+1)
                    ADD    #01h
                    SACL   SECTOR_PTR       ;i.e. inc SECTOR_PTR
                    B      BRNCH_SR
PISR1               SPLK   #00, SECTOR_PTR  ;Reset Sector pointer to 0

BRNCH_SR:
                    LACC   #SECTOR_TBL
                    ADD    SECTOR_PTR
                    TBLR   SR_ADDR
                    LACC   SR_ADDR
                    BACC

;---------------------------------------------------------------------
;Sector 1 calculations - a,b,c --> a,b,c
;---------------------------------------------------------------------
SECTOR_SR1:
                    LACC   T              ;Acc = T
                    SUB    dx             ;Acc = T-dx
                    SUB    dy             ;Acc = T-dx-dy
                    SFR                   ;Acc = Ta = 1/2(T-dx-dy)  <A>
                    SACL   Ta

                    ADD    dx             ;Acc = Tb = dx+Ta     <B>
                    SACL   Tb

                    LACC   T              ;ACC = T
                    SUB    Ta             ;ACC = T-Ta
                    SACL   Tc             ;ACC = Tc = T-Ta     <C>
                    B      LOAD_COMPARES

;---------------------------------------------------------------------
```

```
;Sector 2 calculations - a,b,c --> b,a,c  &  dx <--> dy
;---------------------------------------------------------------
SECTOR_SR2:
                  LACC   T              ;Acc = T
                  SUB    dx             ;Acc = T-dx
                  SUB    dy             ;Acc = T-dx-dy
                  SFR                   ;Acc = Tb = 1/2(T-dx-dy)  <A>
                  SACL   Tb

                  ADD    dy             ;Acc = Ta = dy+Tb     <B>
                  SACL   Ta

                  LACC   T              ;ACC = T
                  SUB    Tb             ;ACC = T-Tb
                  SACL   Tc             ;ACC = Tc = T-Tb     <C>
                  B      LOAD_COMPARES

;---------------------------------------------------------------
;Sector 3 calculations - a,b,c --> c,a,b
;---------------------------------------------------------------
SECTOR_SR3:
                  LACC   T              ;Acc = T
                  SUB    dx             ;Acc = T-dx
                  SUB    dy             ;Acc = T-dx-dy
                  SFR                   ;Acc = Tc = 1/2(T-dx-dy)  <A>
                  SACL   Tb

                  ADD    dx             ;Acc = Ta = dx+Tc     <B>
                  SACL   Tc

                  LACC   T              ;ACC = T
                  SUB    Tb             ;ACC = T-Tc
                  SACL   Ta             ;ACC = Tb = T-Tc     <C>
                  B      LOAD_COMPARES

;---------------------------------------------------------------
;Sector 4 calculations - a,b,c --> c,b,a  &  dx <--> dy
```

```
;-----------------------------------------------------------------
SECTOR_SR4:
                    LACC   T               ;Acc = T
                    SUB    dx              ;Acc = T-dx
                    SUB    dy              ;Acc = T-dx-dy
                    SFR                    ;Acc = Tc = 1/2(T-dx-dy)  <A>
                    SACL   Tc

                    ADD    dy              ;Acc = Tb = dx+Ta     <B>
                    SACL   Tb

                    LACC   T               ;ACC = T
                    SUB    Tc              ;ACC = T-Tc
                    SACL   Ta              ;ACC = Ta = T-Tc     <C>
                    B      LOAD_COMPARES

;-----------------------------------------------------------------
;Sector 5 calculations - a,b,c --> b,c,a
;-----------------------------------------------------------------
SECTOR_SR5:
                    LACC   T               ;Acc = T
                    SUB    dx              ;Acc = T-dx
                    SUB    dy              ;Acc = T-dx-dy
                    SFR                    ;Acc = Tb = 1/2(T-dx-dy)  <A>
                    SACL   Tc

                    ADD    dx              ;Acc = Tc = dx+Ta     <B>
                    SACL   Ta

                    LACC   T               ;ACC = T
                    SUB    Tc              ;ACC = T-Tb
                    SACL   Tb              ;ACC = Ta = T-Tb     <C>
                    B      LOAD_COMPARES

;-----------------------------------------------------------------
;Sector 6 calculations - a,b,c --> a,c,b  &  dx <--> dy
;-----------------------------------------------------------------
```

```
SECTOR_SR6:
                    LACC   T              ;Acc = T
                    SUB    dx             ;Acc = T-dx
                    SUB    dy             ;Acc = T-dx-dy
                    SFR                   ;Acc = Ta = 1/2(T-dx-dy)  <A>
                    SACL   Ta

                    ADD    dy             ;Acc = Tc = dx+Ta     <B>
                    SACL   Tc

                    LACC   T              ;ACC = T
                    SUB    Ta             ;ACC = T-Ta
                    SACL   Tb             ;ACC = Tb = T-Ta     <C>

      ;Transfer new Compare values for this PWM period
LOAD_COMPARES       POINT_B0
                    MAR    *, AR1
                    LACC   Ta
                    SACL   *,0,AR2        ;Load Compare2 Register with Ta
                    LACC   Tb
                    SACL   *,0,AR3        ;Load Compare3 Register with Tb
                    LACC   Tc
                    SACL   *,0,AR1        ;Load Compare4 Register with Tc


     ; Receive Frequency target from CAN bus.

CAN_RD       LDP       #DP_CAN
W_FLAG       BIT       CANIFR,7        ; wait for mailbox 0 interrupt flag
            BCND       CAN_RD_END,NTC


W_RA         BIT       CANRCR,11       ; Wait for receive acknowledge
            BCND       CAN_RD_END,NTC
       SPLK          #0010h,CANRCR      ; reset RA and CANIFR

       LDP             #DP_CAN2
```

```
       LACL            CANMBX0A        ; Load data in Accu
       POINT_B0
       SUB             FREQ_TRGT
       BCND            CAN_RD_END,EQ   ; If the value doesn't change
       LDP             #DP_CAN2
       LACL            CANMBX0A        ; Freq_trgt is not updated
       POINT_B0
       SACL            FREQ_TRGT       ; Change Freq_trg

CAN_RD_END:



;Adjust frequency demand from RT debugger
FREQ_DMD:
       POINT_B0
       LACC         FREQ_TRGT
       SUB          FREQ_SETPT
       BCND         FD_END, EQ

       LACC         RMP_DLY_CNT
       ADD          #1
       SACL         RMP_DLY_CNT
       SUB          #RMP_DLY_MAX
       BCND         FD_END2, LT

CHNG_FREQ:
       LACC         FREQ_TRGT
       SUB          FREQ_SETPT
       BCND         INC_FREQ, GT

DEC_FREQ            LACC   FREQ_SETPT
                    SUB    #1
                    SACL   FREQ_SETPT
                    BCND   FD_END, GEQ
                    SPLK   #1, FREQ_SETPT
                    B      FD_END
```

```
INC_FREQ          LACC   FREQ_SETPT      ;If max then Inc Frequency
                  ADD    #1
                  SACL   FREQ_SETPT
                  SUB    #2047
                  BCND   FD_END, LEQ
                  SPLK   #2047, FREQ_SETPT

FD_END:
      SPLK        #0, RMP_DLY_CNT

                     CLRC   INTM
                     RET

      ;-------------------------------------------------------
      ;Sector routine jump table - used with BACC inst.
      ;-------------------------------------------------------
      SECTOR_TBL:
      SR0          .word  SECTOR_SR1
      SR1          .word  SECTOR_SR2
      SR2          .word  SECTOR_SR3
      SR3          .word  SECTOR_SR4
      SR4          .word  SECTOR_SR5
      SR5          .word  SECTOR_SR6


;=============================================================================
; Routine Name: m S _ D E L A Y  (1mS version)        Routine Type: SR
;
; Description: Produces a multiple of 1mS delays using the RPT instruction.
;              The Delay produced is based on the value loaded in mSEC.
;              i.e. Delay = mSEC x 1mS
;
; Originator: David Figoli

; Last Update: 13 Jan 97
;=============================================================================
```

```
mS_DELAY:
       POINT_B0
       LACC         #25000
       SACL         GPR0
       LACC         mSEC
mS_LP
       RPT             GPR0         ;25,000 cycles = 1 mS @ 25MHz
       NOP                          ;1 cycle
       SUB             #1           ;1 cycle
       BCND         mS_LP,GT        ;4 cycles
       RET                          ;4 cycles
;-------------------------------------------------------
       ;LED display table - used to indicate speed setting
       ;-------------------------------------------------------
       LED_TABLE:
                    .word 00080h
                    .word 00040h
                    .word 00020h
                    .word 00010h
                    .word 00008h
                    .word 00004h
                    .word 00002h
                    .word 00001h
                    .word 00001h

       ;-------------------------------------------------------
       ;No. Samples  256   Angle Range  60
       ;-------------------------------------------------------
       ;            SINVAL    ;     Index   Angle   Sin(Angle)
       STABLE       .word 0   ;   0   0   0.00
              .word   134;   1   0.23   0.00
              .word   268;   2   0.47   0.01
              .word   402;   3   0.70   0.01
              .word   536;   4   0.94   0.02
              .word   670;   5   1.17   0.02
              .word   804;   6   1.41   0.02
              .word   938;   7   1.64   0.03
              .word   1072   ;   8   1.88   0.03
```

```
.word   1206  ;  9  2.11  0.04
.word   1340  ;  10 2.34  0.04
.word   1474  ;  11 2.58  0.04
.word   1608  ;  12 2.81  0.05
.word   1742  ;  13 3.05  0.05
.word   1876  ;  14 3.28  0.06
.word   2009  ;  15 3.52  0.06
.word   2143  ;  16 3.75  0.07
.word   2277  ;  17 3.98  0.07
.word   2411  ;  18 4.22  0.07
.word   2544  ;  19 4.45  0.08
.word   2678  ;  20 4.69  0.08
.word   2811  ;  21 4.92  0.09
.word   2945  ;  22 5.16  0.09
.word   3078  ;  23 5.39  0.09
.word   3212  ;  24 5.63  0.10
.word   3345  ;  25 5.86  0.10
.word   3479  ;  26 6.09  0.11
.word   3612  ;  27 6.33  0.11
.word   3745  ;  28 6.56  0.11
.word   3878  ;  29 6.80  0.12
.word   4011  ;  30 7.03  0.12
.word   4144  ;  31 7.27  0.13
.word   4277  ;  32 7.50  0.13
.word   4410  ;  33 7.73  0.13
.word   4543  ;  34 7.97  0.14
.word   4675  ;  35 8.20  0.14
.word   4808  ;  36 8.44  0.15
.word   4941  ;  37 8.67  0.15
.word   5073  ;  38 8.91  0.15
.word   5205  ;  39 9.14  0.16
.word   5338  ;  40 9.38  0.16
.word   5470  ;  41 9.61  0.17
.word   5602  ;  42 9.84  0.17
.word   5734  ;  43 10.08 0.17
.word   5866  ;  44 10.31 0.18
.word   5998  ;  45 10.55 0.18
```

```
.word   6130  ;  46 10.78 0.19
.word   6261  ;  47 11.02 0.19
.word   6393  ;  48 11.25 0.20
.word   6524  ;  49 11.48 0.20
.word   6655  ;  50 11.72 0.20
.word   6787  ;  51 11.95 0.21
.word   6918  ;  52 12.19 0.21
.word   7049  ;  53 12.42 0.22
.word   7180  ;  54 12.66 0.22
.word   7310  ;  55 12.89 0.22
.word   7441  ;  56 13.13 0.23
.word   7571  ;  57 13.36 0.23
.word   7702  ;  58 13.59 0.24
.word   7832  ;  59 13.83 0.24
.word   7962  ;  60 14.06 0.24
.word   8092  ;  61 14.30 0.25
.word   8222  ;  62 14.53 0.25
.word   8351  ;  63 14.77 0.25
.word   8481  ;  64 15.00 0.26
.word   8610  ;  65 15.23 0.26
.word   8740  ;  66 15.47 0.27
.word   8869  ;  67 15.70 0.27
.word   8998  ;  68 15.94 0.27
.word   9127  ;  69 16.17 0.28
.word   9255  ;  70 16.41 0.28
.word   9384  ;  71 16.64 0.29
.word   9512  ;  72 16.88 0.29
.word   9640  ;  73 17.11 0.29
.word   9768  ;  74 17.34 0.30
.word   9896  ;  75 17.58 0.30
.word   10024 ;  76 17.81 0.31
.word   10151 ;  77 18.05 0.31
.word   10279 ;  78 18.28 0.31
.word   10406 ;  79 18.52 0.32
.word   10533 ;  80 18.75 0.32
.word   10660 ;  81 18.98 0.33
.word   10786 ;  82 19.22 0.33
```

```
.word   10913 ;  83 19.45 0.33
.word   11039 ;  84 19.69 0.34
.word   11165 ;  85 19.92 0.34
.word   11291 ;  86 20.16 0.34
.word   11417 ;  87 20.39 0.35
.word   11543 ;  88 20.63 0.35
.word   11668 ;  89 20.86 0.36
.word   11793 ;  90 21.09 0.36
.word   11918 ;  91 21.33 0.36
.word   12043 ;  92 21.56 0.37
.word   12167 ;  93 21.80 0.37
.word   12292 ;  94 22.03 0.38
.word   12416 ;  95 22.27 0.38
.word   12540 ;  96 22.50 0.38
.word   12664 ;  97 22.73 0.39
.word   12787 ;  98 22.97 0.39
.word   12910 ;  99 23.20 0.39
.word   13033 ;  10023.44 0.40
.word   13156 ;  10123.67 0.40
.word   13279 ;  10223.91 0.41
.word   13401 ;  10324.14 0.41
.word   13524 ;  10424.38 0.41
.word   13646 ;  10524.61 0.42
.word   13767 ;  10624.84 0.42
.word   13889 ;  10725.08 0.42
.word   14010 ;  10825.31 0.43
.word   14131 ;  10925.55 0.43
.word   14252 ;  11025.78 0.43
.word   14373 ;  11126.02 0.44
.word   14493 ;  11226.25 0.44
.word   14613 ;  11326.48 0.45
.word   14733 ;  11426.72 0.45
.word   14852 ;  11526.95 0.45
.word   14972 ;  11627.19 0.46
.word   15091 ;  11727.42 0.46
.word   15210 ;  11827.66 0.46
.word   15328 ;  11927.89 0.47
```

```
.word   15447 ;  12028.13 0.47
.word   15565 ;  12128.36 0.48
.word   15683 ;  12228.59 0.48
.word   15800 ;  12328.83 0.48
.word   15917 ;  12429.06 0.49
.word   16035 ;  12529.30 0.49
.word   16151 ;  12629.53 0.49
.word   16268 ;  12729.77 0.50
.word   16384 ;  12830.00 0.50
.word   16500 ;  12930.23 0.50
.word   16616 ;  13030.47 0.51
.word   16731 ;  13130.70 0.51
.word   16846 ;  13230.94 0.51
.word   16961 ;  13331.17 0.52
.word   17075 ;  13431.41 0.52
.word   17190 ;  13531.64 0.52
.word   17304 ;  13631.88 0.53
.word   17417 ;  13732.11 0.53
.word   17531 ;  13832.34 0.53
.word   17644 ;  13932.58 0.54
.word   17757 ;  14032.81 0.54
.word   17869 ;  14133.05 0.55
.word   17981 ;  14233.28 0.55
.word   18093 ;  14333.52 0.55
.word   18205 ;  14433.75 0.56
.word   18316 ;  14533.98 0.56
.word   18427 ;  14634.22 0.56
.word   18538 ;  14734.45 0.57
.word   18648 ;  14834.69 0.57
.word   18758 ;  14934.92 0.57
.word   18868 ;  15035.16 0.58
.word   18978 ;  15135.39 0.58
.word   19087 ;  15235.63 0.58
.word   19195 ;  15335.86 0.59
.word   19304 ;  15436.09 0.59
.word   19412 ;  15536.33 0.59
.word   19520 ;  15636.56 0.60
```

```
.word   19627 ;  15736.80 0.60
.word   19735 ;  15837.03 0.60
.word   19841 ;  15937.27 0.61
.word   19948 ;  16037.50 0.61
.word   20054 ;  16137.73 0.61
.word   20160 ;  16237.97 0.62
.word   20265 ;  16338.20 0.62
.word   20371 ;  16438.44 0.62
.word   20475 ;  16538.67 0.62
.word   20580 ;  16638.91 0.63
.word   20684 ;  16739.14 0.63
.word   20788 ;  16839.38 0.63
.word   20891 ;  16939.61 0.64
.word   20994 ;  17039.84 0.64
.word   21097 ;  17140.08 0.64
.word   21199 ;  17240.31 0.65
.word   21301 ;  17340.55 0.65
.word   21403 ;  17440.78 0.65
.word   21504 ;  17541.02 0.66
.word   21605 ;  17641.25 0.66
.word   21706 ;  17741.48 0.66
.word   21806 ;  17841.72 0.67
.word   21906 ;  17941.95 0.67
.word   22006 ;  18042.19 0.67
.word   22105 ;  18142.42 0.67
.word   22204 ;  18242.66 0.68
.word   22302 ;  18342.89 0.68
.word   22400 ;  18443.13 0.68
.word   22498 ;  18543.36 0.69
.word   22595 ;  18643.59 0.69
.word   22692 ;  18743.83 0.69
.word   22788 ;  18844.06 0.70
.word   22884 ;  18944.30 0.70
.word   22980 ;  19044.53 0.70
.word   23075 ;  19144.77 0.70
.word   23170 ;  19245.00 0.71
.word   23265 ;  19345.23 0.71
```

```
.word   23359 ;  19445.47 0.71
.word   23453 ;  19545.70 0.72
.word   23546 ;  19645.94 0.72
.word   23640 ;  19746.17 0.72
.word   23732 ;  19846.41 0.72
.word   23824 ;  19946.64 0.73
.word   23916 ;  20046.88 0.73
.word   24008 ;  20147.11 0.73
.word   24099 ;  20247.34 0.74
.word   24189 ;  20347.58 0.74
.word   24279 ;  20447.81 0.74
.word   24369 ;  20548.05 0.74
.word   24459 ;  20648.28 0.75
.word   24548 ;  20748.52 0.75
.word   24636 ;  20848.75 0.75
.word   24724 ;  20948.98 0.75
.word   24812 ;  21049.22 0.76
.word   24900 ;  21149.45 0.76
.word   24986 ;  21249.69 0.76
.word   25073 ;  21349.92 0.77
.word   25159 ;  21450.16 0.77
.word   25245 ;  21550.39 0.77
.word   25330 ;  21650.63 0.77
.word   25415 ;  21750.86 0.78
.word   25499 ;  21851.09 0.78
.word   25583 ;  21951.33 0.78
.word   25667 ;  22051.56 0.78
.word   25750 ;  22151.80 0.79
.word   25833 ;  22252.03 0.79
.word   25915 ;  22352.27 0.79
.word   25997 ;  22452.50 0.79
.word   26078 ;  22552.73 0.80
.word   26159 ;  22652.97 0.80
.word   26239 ;  22753.20 0.80
.word   26320 ;  22853.44 0.80
.word   26399 ;  22953.67 0.81
.word   26478 ;  23053.91 0.81
```

```
        .word   26557 ;  23154.14 0.81
        .word   26635 ;  23254.38 0.81
        .word   26713 ;  23354.61 0.82
        .word   26791 ;  23454.84 0.82
        .word   26868 ;  23555.08 0.82
        .word   26944 ;  23655.31 0.82
        .word   27020 ;  23755.55 0.82
        .word   27096 ;  23855.78 0.83
        .word   27171 ;  23956.02 0.83
        .word   27246 ;  24056.25 0.83
        .word   27320 ;  24156.48 0.83
        .word   27394 ;  24256.72 0.84
        .word   27467 ;  24356.95 0.84
        .word   27540 ;  24457.19 0.84
        .word   27612 ;  24557.42 0.84
        .word   27684 ;  24657.66 0.84
        .word   27756 ;  24757.89 0.85
        .word   27827 ;  24858.13 0.85
        .word   27897 ;  24958.36 0.85
        .word   27967 ;  25058.59 0.85
        .word   28037 ;  25158.83 0.86
        .word   28106 ;  25259.06 0.86
        .word   28175 ;  25359.30 0.86
        .word   28243 ;  25459.53 0.86
        .word   28311 ;  25559.77 0.86

;=====================================================================
; I S R  -  PHANTOM
;
; Description:  Dummy ISR, used to trap spurious interrupts.
;
; Modifies:
;
; Last Update:  16-06-95
;=====================================================================
PHANTOM     B  PHANTOM
```

INTERNET

www.ti.com

Register with TI&ME to build custom information pages and receive new product updates automatically via email.

TI Semiconductor Home Page
http://www.ti.com/sc

TI Distributors
http://www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone	+1(972) 644-5580
Fax	+1(972) 480-7800
Email	sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone	
Deutsch	+49-(0) 8161 80 3311
English	+44-(0) 1604 66 3399
Francais	+33-(0) 1-30 70 11 64
Italiano	+33-(0) 1-30 70 11 67
Fax	+33-(0) 1-30-70 10 32
Email	epic@ti.com

Japan

Phone	
International	+81-3-3457-0972
Domestic	+0120-81-0026
Fax	
International	+81-3-3457-1259
Domestic	+0120-81-0036
Email	pic-japan@ti.com

Asia

Phone	
International	+886-2-3786800
Domestic	
Australia	1-800-881-011
TI Number	-800-800-1450
China	10811
TI Number	-800-800-1450
Hong Kong	800-96-1111
TI Number	-800-800-1450
India	000-117
TI Number	-800-800-1450
Indonesia	001-801-10
TI Number	-800-800-1450
Korea	080-551-2804
Malaysia	1-800-800-011
TI Number	-800-800-1450
New Zealand	+000-911
TI Number	-800-800-1450
Philippines	105-11
TI Number	-800-800-1450
Singapore	800-0111-111
TI Number	-800-800-1450
Taiwan	080-006800
Thailand	0019-991-1111
TI Number	-800-800-1450

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.