Programming Examples for the 24x/240xA CAN

Hareesh Janakiraman *Advanced Embedded Control Group*

ABSTRACT

The 24x (TMS320F243 and TMS320F241) and 240xA (TMS320LF2407A, 2406A, and 2403A) series of digital signal processor (DSP) controllers feature an on-chip Controller Area Network (CAN) module. This module is a Full-CAN controller, compliant with CAN specification 2.0B. This application report describes the operation of the CAN module and its control registers. Several programming examples have been included in SPRA.zip to illustrate how the CAN module is set up for different modes of operation. All programs have been extensively commented to aid comprehension. The sample code described in this application report can be downloaded from http://www.ti.com/lit/zip/SPRA890.

Contents

List of Tables

1 Introduction

CAN is a multi-master serial protocol which was originally developed for automotive applications. Due to its robustness and reliability, it now finds applications in diverse areas such as Industrial automation, appliances, medical electronics, maritime electronics etc. CAN protocol features sophisticated error detection (and isolation) mechanisms and lends itself to simple wiring at the physical level.

The CAN module available in 24x and 240xA devices are exactly identical. This means that all registers have the same addresses and bit functions. However, there are some important differences between these two DSP families at the core level, which have a bearing on programming the CAN module. The first difference pertains to the clock speed of the DSP; The 24x devices have a maximum clock speed of 20 MHz, whereas the 240xA devices can run at 40 MHz. The clock speeds at which the devices operate affect the values that are written in the Bit Configuration Registers (BCRn). To operate the CAN modules at a given baud rate, different values may have to be written in BCRn depending on the DSP clock frequency.

The second difference pertains to the peripheral clock enable/disable feature available in 240xA devices. By writing the appropriate value in System Control and Status Register 1 (SCSR1), clock to any peripheral can be selectively enabled or disabled. This feature is absent in the 24x devices. The SCSR1 register also has the clock pre-scaler bits for the PLL, which allow the operation of 240xA devices at several frequencies, unlike the fixed (x4) PLL multiplication factor for the 24x devices. If these differences are taken care of, CAN code can be migrated across the 24x/’240xA families.

The 24x/240xA CAN Features include the following:

- Ability to work with both standard (11-bits) and extended (29-bits) identifiers.
- A total of six mailboxes with each mailbox capable of storing up to 8 bytes of data.
  - Mailboxes 0 and 1 are receive mailboxes
  - Mailboxes 4 and 5 are transmit mailboxes
  - Mailboxes 2 and 3 are configurable as receive or transmit mailboxes
- Two Local Acceptance Mask (LAMn) registers for the four receive mailboxes to facilitate message filtering.
- Programmable bit-rate and Re-Synchronization Jump Width (SJW).
- Automatic reply to a Remote Transmission Request (RTR).
- Programmable interrupt scheme for mailbox and error interrupts.
- Self-test mode – CAN module operates in a standalone mode obviating the need to be connected to another CAN module.

2 CAN Frames

The CAN protocol employs four different types of frames to enable communication between nodes. Once the data to be transmitted is written to the appropriate mailbox, the CAN module takes over the control of data transmission and formats the data (according to the format of CAN protocol) for transmission over the CAN bus. The usage of frames results in a well-structured and reliable communication. The four frame types are:

- Data frame
- Remote frame
- Error frame
- Overload frame

2.1 Data Frame

As the name implies, the data frame is used to transmit data between CAN nodes. It is generated by a CAN node to transmit data over the CAN bus. The structure of the data frame changes depending on whether standard-identifiers or extended-identifiers are used. Figure 1 shows the structure of a data frame using extended identifiers.

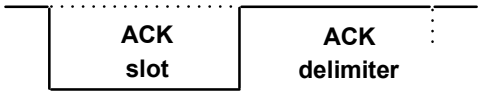
Legend:
SOF = Start of frame
Std ID = Standard Identifier
SRR = Substitute remote request
IDE = Identifier extension
Ext ID = Extended identifier
RTR = Remote transmission request
r1, r0 = Reserved bits
DLC = Data length code
CRC = Cyclic redundancy check
ACK = Acknowledge field
EOF = End of frame

Figure 1. Data Frame (With Extended Identifier)

The data frame includes the following:

- Start of frame (SOF) – The data frame begins with a dominant SOF bit (In the figure, the period preceding SOF is the recessive bus-idle period). This bit is used for the hard synchronization of all the CAN nodes.
- The 11-bit standard identifier follows the SOF bit.
- Substitute remote request (SRR) – The SRR bit occupies the same position as the RTR bit would in a standard frame.
- Identifier extension (IDE) – The IDE bit differentiates a standard frame from an extended frame. A dominant IDE bit indicates a standard frame whereas a recessive IDE bit indicates an extended frame.
- The 18-bit extended identifier follows the IDE bit.
- Remote transmission request (RTR) – The RTR bit differentiates a data frame from a remote frame. A dominant RTR bit indicates a data frame whereas a recessive RTR bit indicates a remote frame.
- r1, r0 – Reserved bits

- Data length code (DLC) – The four DLC bits specify the number of data bytes (0 – 8) contained in the message.
- Data – 0 to 8 bytes
- Cyclic redundancy check (CRC) – The 15-bit CRC value holds a checksum sequence which is used to detect errors in transmission. The 15-bit checksum is followed by a recessive CRC delimiter bit, making the CRC field a 16-bit entity.
- Acknowledge field (ACK) – The ACK field consists of two bits. The first bit is the ACK slot, which is sent out as a recessive bit by the transmitter. This recessive bit will be overwritten by a dominant bit transmitted by any node that has successfully received the message. Failure to sense this bit as a dominant bit by the transmitter results in an acknowledge error. The second bit in this field is the recessive ACK delimiter as shown below:



- End of frame (EOF) – The EOF field consists of seven recessive bits. The EOF is followed by a 3-bit (recessive) inter-frame space which separate two frames on the CAN bus.

2.2 Remote Frame

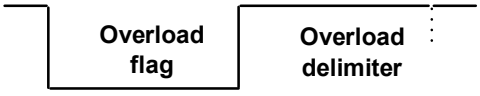
Remote frames are sent by a node that requests data from another node. The structure of a remote frame is similar to that of a data frame with the exception that RTR bit is recessive and the data byte length is zero. The identifiers of the remote frame and the data frame must be identical. In addition, the Auto Answer Mode (AAM) bit of the node that has the required data must be set to complete the event automatically.

2.3 Error Frame

The CAN protocol provides for sophisticated error-detection and correction mechanism. If an error is detected by any node while transmission is in progress, it immediately generates an error frame, even before the transmission is complete. The transmitter of the message aborts the transmission and attempts retransmission. The 24x/240x DSP's have error counters that are incremented or decremented based on certain rules.

2.4 Overload Frame

An overload frame is generated by a CAN node to delay the next message on the CAN bus. This is done if the node is busy doing something else and is unable to process CAN data. This type of frame can be generated during the inter-frame space only. Similar in format to an active- error frame, it has two fields. The first field is the overload flag. It can be from 6 bits up to a maximum of 12 dominant bits. It will be 6 bits when only one node generates an overload frame; when two or more nodes generate overload frames, the length may go up to 12 bits. The overload delimiter consists of eight recessive bits.



3 Layout of a Mailbox

The layout of all mailboxes in the 24x/240xA CAN module is exactly identical to each other. Each mailbox occupies eight 16-bit locations in the mailbox RAM. If any mailbox is not used, then the corresponding eight locations can be used as normal RAM in the data-memory space. Each mailbox has the structure shown in Table 1.

Table 1. Layout of Mailbox "n" in 24x/240x CAN Module

Address Offset	Register
0	MSG_ID_nL
1	MSG_ID_nH
2	MSG_CTRLn
3	Reserved
4	MBOXnA
5	MBOXnB
6	MBOXnC
7	MBOXnD

3.1 MSG_ID_nL

If an extended-identifier is used, then the lower part of the extended-identifier (bits 15–0) is stored in this register. A standard-identifier does not use this register.

3.2 MSG_ID_nH

If an extended-identifier is used, then the upper 13-bits of the extended-identifier (bits 28–6) is stored in this register. If a standard-identifier is used, the 11-bit ID is stored in bits 12–2 of this register. This register also has the IDE, AME and AAM bits.

3.3 MSG_CTRLn

The MSG_CTRLn register has the Data Length Code (DLC) bits. These bits specify the number of bytes to be transmitted in a message. For example, If a length of six bytes is specified, only six bytes of the mailbox RAM will be transmitted. The value of DBO bit in the MCR register determines which six bytes (of the transmit-mailbox RAM) will be transmitted. The DLC field of the receive mailbox will be overwritten by the DLC value embedded in the data frame.

3.4 MBOXnA/B/C/D

Each MBOXnX register can hold two bytes, giving a total of eight-byte storage space for each mailbox.

4 Effect of DBO Bit on Data Storage/Transmission in Mailbox RAM

Table 2 shows how received data is stored when 8 bytes (0, 1, 2, 3, 4, 5, 6, 7) are transmitted on the CAN bus with Mailbox 2 as an example.

Table 2. Mailbox 2 Example of Data Storage

Mailbox	Address	When DBO = 0		When DBO = 1	
MBOX_2A	7214h	Byte-2	Byte-3	Byte-1	Byte-0
MBOX_2B	7215h	Byte-0	Byte-1	Byte-3	Byte-2
MBOX_2C	7216h	Byte-6	Byte-7	Byte-5	Byte-4
MBOX_2D	7217h	Byte-4	Byte-5	Byte-7	Byte-6

When mailbox contents are as shown in Table 3, they will be transmitted as follows:

3, 2, 1, 0, 7, 6, 5, 4 - When DBO = 0

0, 1, 2, 3, 4, 5, 6, 7 - When DBO = 1

Table 3. Mailbox Content Addresses

Mailbox	Address	MBX Content	
MBOX_2A	7214h	Byte-1	Byte-0
MBOX_2B	7215h	Byte-3	Byte-2
MBOX_2C	7216h	Byte-5	Byte-4
MBOX_2D	7217h	Byte-7	Byte-6

5 CAN Module Initialization

The CAN module must be initialized before it can transmit/receive message packets over the CAN bus. This is done by writing the appropriate values to the various configuration and control registers. The initialization task can be broadly divided into three steps.

5.1 Steps in Initialization

Step 1: Assigning the MSGIDs

The assignment of unique message ID's to the various mailboxes makes a CAN module capable of filtering messages, thereby accepting only those messages intended for it. All mailboxes (both transmit and receive) that are used should be assigned unique message IDs. The MSGID of the transmitting mailbox is always attached to the transmitted message. When a message is received, a decision has to be made whether to accept (and store) that particular message. There are two decision mechanisms based on whether message (or acceptance) filtering is enabled or not. If acceptance filtering is not used, then the MSGID of the incoming message must exactly match (bit for bit) the MSGID of a receive mailbox. If this is the case, then the incoming message is received and stored in that receive mailbox.

Step 2: Enabling acceptance-filtering

(Note that this step is optional. An application may not use acceptance-filtering, although in practice, this is generally not the case). If acceptance filtering is used, then the decision whether to accept (and store) a particular message is taken based on three parameters:

- MSGID of the incoming message
- MSGID of the receive mailbox
- Configuration of the Local Acceptance Mask (LAMn) bits corresponding to that receive mailbox.

This filtering is accomplished with the help of MSGIDn and LAMn registers.

Step 3: Configuring the bit-timing parameters

The bit-timing configuration of a CAN module must match that of other nodes on the network. The "Bit configuration registers (BCRn)" are used for this purpose. Table 4 shows the parameters for some common bit rates employed.

Table 4. Bit Rate Parameters (for CLKOUT = 40 MHz)

Bit-rate	Sampling point (SP)	BRP register value	Length of 1 TQ	Bit-time (BT)	TSEG1	TSEG2
1 Mbps	70 %	3	100 nS	10	5	2
1 Mbps	85 %	1	50 nS	20	15	2
500 Kbps	80 %	3	100 nS	20	14	3
250 Kbps	80 %	7	200 nS	20	14	3
100 Kbps	80 %	15	400 nS	25	18	4
50 Kbps	65 %	39	1 ìS	20	11	6

NOTE: The parameters BRP, TSEG1 and TSEG2 shown are the actual values written into the registers. These parameters are enhanced by one by the CAN module when these registers are accessed.

5.2 CAN Transceivers for 24x/240xA Devices

The SN65HVD251 CAN transceiver from Texas Instruments is ideally suited for operation with 5v DSPs (and microcontrollers) such as the ‘24x series. The features of SN65HVD251 are as follows:

- Drop-in improved replacement for the PCA82C250 and PCA82C251
- Bus-Fault protection of ±36 V
- Bus pin ESD protection exceeds 12-kV HBM
- Meets or exceeds the requirements of ISO 11898
- Designed for signaling rates up to 1 Mbps
- High input-impedance allows up to 120 nodes
- Unpowered Node does not disturb the bus
- Low-current standby mode: 200 µA Typical

Table 5 lists CAN transceivers from Texas Instruments that are true 3.3-V transceivers needing a 3.3-V rail only and are ideally suited to work with the TM320LF240xA series of DSPs.

Table 5. CAN Transceivers

Part Number	Description	Special features
SN65HVD230	3.3-V CAN Transceiver with Standby Mode	Controlled Slew Rate & Vref Pin
SN65HVD231	3.3-V CAN Transceiver with Sleep Mode	Controlled Slew Rate & Vref Pin
SN65HVD232	3.3-V CAN Transceiver	–

6 Program Examples

The programs are first summarized with a brief description, followed by the code.

ST–STD32.asm This program transmits data back-to-back at high speed. This program illustrates the use of self-test mode.

TXLOOP.asm This program transmits data to another CAN module using mailbox 5. The transmit loop can be executed a predetermined number of times or infinite times. Useful to check the transmit functionality.

RXLOOP.asm This is an example of how data may be received.

LPTX5POL.asm Transmit loop using Mailbox 5. This program, in conjunction with ECRX0POL, provides a quick and easy way to determine if two 24x/240x DSPs are able to communicate via the CAN bus.

ECRX0POL.asm Receive & Echo loop using Mailbox 0. This program, in conjunction with LPTX5POL, provides a quick and easy way to determine if two 24x/240x DSPs are able to communicate via the CAN bus.

REM–REQ.asm This program transmits a remote frame and expects a data frame in response. Transmission of a remote frame by (and reception of the data frame in) MBX3. To be used along with REM–ANS.asm.

REM–ANS.asm Program to auto-answer a remote frame request in CAN. To be used along with REM–REQ.asm.

PWRDOWN.asm Program to illustrate entry/exit into/out of Low-power mode of the CAN module.

6.1 ST–STD32.asm

This program transmits data back-to-back at high speed. This program illustrates the use of self-test mode.

```
* PROGRAM TO DEMONSTRATE THE SELF-TEST MODE IN 24x/240xA CAN    *
* Simple loop back test: the CAN sends a message to itself.     *
; MAILBOX 3 for TRANSMISSION / MAILBOX 2 for RECEPTION

    .include    ”240x.h”      ; Variable and register declaration
    .include    ”vector.h”    ; Vector table (takes care of dummy password)
    .global    START

;-------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------
DP_PF1      .set    224         ; Page 1 of peripheral file (7000h/80h
DP_CAN      .set    0E2h           ; Can Registers (7100h)
DP_CAN2     .set    0E4h        ; Can RAM (7200h)

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

;=====================================================================
; M A I N   C O D E - starts here
;=====================================================================
        .text
START:  KICK_DOG                   ; Reset Watchdog counter
        SPLK    #0,60h
        OUT     60h,WSGR           ; Set waitstates for external memory (if used)
        SETC    INTM               ; Disable interrupts
        SPLK    #0000h,IMR         ; Mask all core interrupts
        LDP     #0E0h
        SPLK    #006Fh, WDCR       ; Disable WD
        SPLK    #0010h,SCSR1       ; Enable clock to CAN module (For 240xA only)

        LAR     AR1,#300h          ; AR1 => Copy CAN RAM (B0)
        LAR     AR2,#7214h         ; AR2 => Mailbox 2 RAM (Rcv)
        LAR     AR4,#721ch         ; AR4 => Mailbox 3 RAM (Xmi)
        LAR     AR3,#3             ; AR3 => counter
        LAR     AR5,#7100h         ; AR5 => CAN control registers


        LDP #DP_CAN
        SPLK #03f7fh,CANIMR ; Enable all ints.
;**********************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL       **********
;**********************************************************************

        SPLK #0000000000000000b,CANMDER ; Disable all mailboxes
;               |||||||||||||||          ; Required b4 writing
;               FEDCBA9876543210         ; to MBX RAM

;**********************************************************************
;***********          SET MSGIDs OF CAN MAILBOXES                 **********
;**********************************************************************

        LDP #DP_CAN2

        SPLK    #1111111111111111b,CANMSGID2H     ; Set mailbox 2 ID
;               |||||||||||||||
;               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 #1111111111111010b,CANMSGID2L ; 1FFF FFFA --> ID
;               |||||||||||||||
;               FEDCBA9876543210

;bit 0-15  lower part of extended identifier
;---------------------------------------------------------------------
        SPLK    #1111111111111111b,CANMSGID3H     ; Set mailbox 3 ID
```

```
;               |||||||||||||||
;               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    #1111111111111010b,CANMSGID3L     ; 1FFF FFFA --> ID
;               |||||||||||||||
;               FEDCBA9876543210
;*************************************************************************
;***********                Write CAN Mailboxes                  **********
;*************************************************************************

;bit 0-15  lower part of extended identifier
;-------------------------------------------------------------------------
        SPLK    #0000000000001000b,CANMSGCTRL3
;               |||||||||||||||
;               FEDCBA9876543210

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

        SPLK    #00123h,CANMBX3A          ; Message to transmit
        SPLK    #04567h,CANMBX3B
        SPLK    #089ABh,CANMBX3C
        SPLK    #0CDEFh,CANMBX3D

LOOP_READ2      MAR *,AR4              ; AR4 => Mailbox 3 RAM (Xmi)
        LACL    *+,AR1                    ; Copy the Mailbox 0 in ACC
        SACL    *+,AR3                    ; Copy the Mailbox 0 in B0
        BANZ    LOOP_READ2
        LAR     AR3,#0Bh       ;AR3 => counter

;*************************************************************************
;***********    Enable Mailboxes after writing     **********
;*************************************************************************

        LDP     #DP_CAN

        SPLK    #0000000001001100b,CANMDER
;               |||||||||||||||
;               FEDCBA9876543210

;bit 0-5   enable mailboxes 3 and 2
;bit 6     1: mailbox 2 receive
;bit 7     0: mailbox 3 transmit

;*************************************************************************
;***********    Bit timing Registers configuration   **********************
;*************************************************************************

        LDP     #DP_CAN
        SPLK    #0001000000000000b,CANMCR
;               |||||||||||||||
;               FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,#0Bh ; Wait for Change config Enable
        BCND    W_CCE,NTC  ; bit to be set in GSR

         ;SPLK #0000000000000000b,CANBCR2 ; For 1 Mbps @ 20 MHz CLKOUT
         SPLK   #0000000000000001b,CANBCR2; For 1 Mbps @ 40 MHz CLKOUT
;               |||||||||||||||
;               FEDCBA9876543210

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

        SPLK    #0000000011111010b,CANBCR1    ; For 1 Mbps @ 85 % samp. pt
;               |||||||||||||||
;               FEDCBA9876543210

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

        SPLK    #0000000001000000b,CANMCR ; Enable self-test mode
;               |||||||||||||||
;               FEDCBA9876543210
```

```
;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh           ; Wait for Change config disable
        BCND    W_NCCE,TC

;*************************************************************************
;***********                      TRANSMIT                       **********
;*************************************************************************
        SPLK    #0020h,CANTCR             ; transmit request for mailbox 3

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

COPY     MAR    *,AR5                     ; AR5 => CAN control registers
        LACL    *+,AR1                    ; Copy the CAN control regs in Accu
        SACL    *+,AR3                    ; Copy the CAN control regsin B0
        BANZ    COPY                      ; 11 times

W_FLAG3 BIT     CANIFR,4              ; wait for interrupt flag
        BCND    W_FLAG3,NTC
        SPLK    #2000h,CANTCR             ; reset TA and CANIFR
;*************************************************************************
;***********                       RECEIVE                       **********
;*************************************************************************
W_FLAG2 BIT     CANIFR,BIT10   ; Wait for MBX2 RCV interrupt
        BCND    W_FLAG2,NTC

W_RA    BIT     CANRCR,9              ; Wait for receive acknowledge
        BCND    W_RA,NTC                  ; RMP bit set for MBX2?
        SPLK    #0040h,CANRCR         ; reset RMP and CANIFR

;*************************************************************************
;***********                  READ CAN RAM                       **********
;*************************************************************************

        LAR     AR3,#3h            ; AR3->3

LOOP_READ  MAR *,AR2              ; AR2 => Mailbox 2 RAM (Rcv)
        LACL    *+,AR1                ; Copy MBX 2 in Accu
        SACL    *+,AR3                ; Copy MBX 2 in B0
        BANZ    LOOP_READ      ;

LOOP        B   LOOP           ; loop

GISR1:  RET
GISR2: RET
GISR3: RET
GISR4: RET
GISR5: RET
GISR6: RET
PHANTOM:   RET

; When program works correctly,
; mailbox data will be seen beginning in 300h and 310h. The transmitted data is
; stored beginning at 300h and the received data is stored beginning at 310h.
; If things are ok, the transmitted & received data should be the same.
```

6.2 TXLOOP.asm

This program transmits data to another CAN module using mailbox 5. The transmit loop can be executed a predetermined number of times or infinite times. Useful to check the transmit functionality.

```
*       TXLOOP - Transmit loop using Mailbox 5

* This program TRANSMITS data to another CAN module using MAILBOX5
* This program could either loop forever or transmit "n+1" # of times,
* where "n" is the TXCOUNT value. The # of times data was actually
* transmitted is recorded in 304h in Data memory.

* COMMENTS : The two CAN modules must be connected to each other with
* appropriate termination resistors. Program does not use any interrupts.
* A CLKOUT of 40 MHz yields a baud rate of 1 Mbits/s.

; XF is pulsed everytime a packet is transmitted. At 40 MHz CLKOUT, XF will
; pulse every 136 uS.

; Last update 12/27/2002

    .title     "TXLOOP"       ; Title
    .include    "240x.h"      ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
     .global   START

TXCOUNT    .set    01000           ; Determines the # of Xmit cycles

;-------------------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------------------
DP_PF1     .set    0E0h       ; Page 1 of peripheral file (7000h/80h)E0
DP_CAN     .set    0E2h       ; CAN Registers (7100h)
DP_CAN2    .set    0E4h    ; CAN RAM (7200h)

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

       .text

START:  KICK_DOG                ; Reset Watchdog counter
       SPLK    #0,60h
       OUT     60h,WSGR       ; Set waitstates for external memory (if used)
       SETC    INTM              ; Disable interrupts
       SPLK    #0000h,IMR     ; Mask all core interrupts
       LDP     #0E0h
       SPLK    #006Fh, WDCR   ; Disable WD
       SPLK    #0010h,SCSR1   ; Enable clock to CAN module (For 240xA only)


       LDP     #225
       SPLK    #00C0H,MCRB    ; Configure CAN pins

       CALL    AR_INIT

       MAR     *,AR6          ; Initialize counter @ 304h
       SPLK    #0h,*

        LDP    #DP_CAN        ; Enable all CAN interrupts. This is reqd
       SPLK    #03F7Fh,CANIMR ; to poll flags.
;***************************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX5  **********
;***************************************************************************

       SPLK #0000000000000000b,CANMDER ; Disable all mailboxes
;               ||||||||||||||||
;               FEDCBA9876543210

;***************************************************************************
;***********    Set MSGID/MSGCTRL for transmit mailbox           **********
;***************************************************************************

       LDP #DP_CAN2
```

```
        SPLK    #1010111000010101b,CANMSGID5H ; Set mailbox 5 ID
;                |||||||||||||||||                ; XMIT Mailbox
;                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    #1101110000110101b,CANMSGID5L ; AE15 DC35 --> ID
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 0-15  lower part of extended identifier

        SPLK    #0000000000001000b,CANMSGCTRL5    ; 0008
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 0-3   Data length code. 1000 = 8 bytes
;bit 4     0: data frame
;*****************************************************************************
;******     ENABLE MBX AFTER WRITING TO MSGID/MSGCTRL OF MBX5 **********
;*****************************************************************************

        LDP     #DP_CAN
        SPLK    #0000000000100000b,CANMDER
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 0-5   enable mailbox 5

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

        LDP     #DP_CAN2
        SPLK    #00100h,CANMBX5A   ; Message to transmit
        SPLK    #00302h,CANMBX5B
        SPLK    #00504h,CANMBX5C
        SPLK    #00706h,CANMBX5D

COPY5   MAR     *,AR5           ; AR5 => Mailbox 5 RAM (XMIT)
        LACL    *+,AR0          ; Copy Mailbox 5 RAM in B0 (300 & above)
        SACL    *+,AR4          ; AR4 => Counter
        BANZ    COPY5           ; All four words read?

;*****************************************************************************
;***********    Bit timing Registers configuration   **********************
;*****************************************************************************

        LDP     #DP_CAN
        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 12    Change configuration request for write-access to BCR (CCR=1)
W_CCE   BIT     CANGSR,#0Bh ; Wait for Change config Enable
        BCND    W_CCE,NTC  ; bit to be set in GSR

        ;SPLK  #0000000000000000b,CANBCR2    ; For 1 Mbps @ 20 MHz CLKOUT
        SPLK    #0000000000000001b,CANBCR2    ; For 1 Mbps @ 40 MHz CLKOUT
;                |||||||||||||||||
;                FEDCBA9876543210
; bit 0-7  Baud rate prescaler
; bit 8-15 Reserved

        SPLK    #0000000011111010b,CANBCR1        ; For 1 Mbps @ 85 % samp. pt
;                |||||||||||||||||
;                FEDCBA9876543210
; bit 0-2  TSEG2
; bit 3-6  TSEG1
; bit 7    Sample point setting (1: 3 times, 0: once)
; bit 8-9  Synchronization jump width
; bit A-F  Reserved
        SPLK    #0000000000000000b,CANMCR
;                |||||||||||||||||
;                FEDCBA9876543210
```

```
;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh ; Wait for Change config disable
        BCND    W_NCCE,TC

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

        SETC    XF                        ; A toggling XF bit indicates
        RPT     #080h                     ; that the program is still
        NOP                               ; running.
        CLRC    XF

W_TA    BIT     CANTCR,BIT15              ; Wait for transmission acknowledge
        BCND    W_TA,NTC

W_FLAG3 BIT     CANIFR,BIT13              ; wait for interrupt flag
        BCND    W_FLAG3,NTC

        MAR     *,AR6                     ; This loop merely keeps a
        LACL    *                           ; count of the number of times
        ADD     #1                        ; data packets were transmitted
        SACL    *                           ; to the remote node.

        SPLK    #8000h,CANTCR ; reset TA

        MAR     *,AR1
         ;BANZ  TX_LOOP        ; Uncomment this line for ”n” transmissions
         B      TX_LOOP        ; Uncomment this line for infinite transmissions

LOOP    B       LOOP    ; Loop here after completion of transmissions

;*************************************************************************
;***********              COMMON ROUTINES             **********
;*************************************************************************

; AR Initializing routine

AR_INIT LAR AR0,#0300h ; AR0 => Xmitted data
        LAR AR1,#TXCOUNT ; AR1 => Counter for TX loops
        LAR AR4,#03    ; AR4 => Counter for copying data
        LAR AR5,#722Ch ; AR5 => Mailbox 5 RAM (TRANSMIT)
        LAR AR6,#304h  ; AR6 keeps track of transmit cycles
        RET

GISR1: RET
GISR2: RET
GISR3: RET
GISR4: RET
GISR5: RET
GISR6: RET
PHANTOM:   RET
        .end
```

6.3 RXLOOP.asm

This is an example of how data may be received.

```
*       RXLOOP – Transmit loop using Mailbox 2                                  *
* This program RECEIVES data from another CAN module using MAILBOX2    *
* The number of times data was transmitted is recorded in AR0.         *
* A CLKOUT of 40 MHz yields a baud rate of 500 kbits/s.                *
; XF is pulsed everytime a packet is received
; Last update 12/27/2002

    .title     ”RXLOOP”       ; Title
    .include    ”240x.h”      ; Variable and register declaration
    .include    ”vector.h”    ; Vector table (takes care of dummy password)
    .global    START

;-------------------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------------------
DP_PF1      .set    0E0h           ; Page 1 of peripheral file (7000h/80h)E0
DP_CAN      .set    0E2h           ; CAN Registers (7100h)
DP_CAN2     .set    0E4h           ; CAN RAM (7200h)
        .text
START:  SETC    INTM               ; Disable interrupts
        LDP     #0E0h
        SPLK    #006Fh, WDCR       ; Disable WD
        SPLK    #0010h,SCSR1       ; Enable clock to CAN module (For 240xA only)
        LAR     AR0,#0             ; AR0 keeps track of the # of receive cycles
        MAR     *,AR0


        LDP     #225
        SPLK    #00C0H,MCRB        ; Configure CAN pins

         LDP    #DP_CAN            ; Enable all CAN interrupts
        SPLK    #03F7Fh,CANIMR
;************************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX0  **********
;************************************************************************

        SPLK    #0000000000000000b,CANMDER   ; Disable all mailboxes
;                |||||||||||||||
;                FEDCBA9876543210

;************************************************************************
;***********    Set MSGID/MSGCTRL for transmit mailbox           **********
;************************************************************************

        LDP     #DP_CAN2

        SPLK    #1100111000010101b,CANMSGID2H
;                |||||||||||||||
;                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    #1101110000110101b,CANMSGID2L ; E15 DC35 --> ID
;                |||||||||||||||
;                FEDCBA9876543210
;bit 0-15  lower part of extended identifier
;************************************************************************
;******     ENABLE MBX AFTER WRITING TO MSGID **********
;************************************************************************

         LDP    #DP_CAN
        SPLK    #0000000011000100b,CANMDER
;                |||||||||||||||
;                FEDCBA9876543210
;bit 0-5   enable mailbox 0
```

```
;***************************************************************************
;***********    Bit timing Registers configuration   **********************
;***************************************************************************

        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||
;                FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,#0Bh ; Wait for Change config Enable
        BCND    W_CCE,NTC  ; bit to be set in GSR

        SPLK    #3,CANBCR2                      ; For 500 kbits/s @ @ 40 MHz CLKOUT
;                |||||||||||||||
;                FEDCBA9876543210

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


        SPLK    #0000000011110011b,CANBCR1      ; For 500 kbits/s @ 80 % samp. pt
;                |||||||||||||||
;                FEDCBA9876543210

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

        SPLK    #0000010000000000b,CANMCR
;                |||||||||||||||
;                FEDCBA9876543210

;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh                     ; Wait for Change config disable
        BCND    W_NCCE,TC

;***************************************************************************
;***********                        RECEIVE                      **********
;***************************************************************************
RX_LOOP

W_RMP   BIT     CANRCR,BIT6    ; Wait for transmission acknowledge
        BCND    W_RMP,NTC

        SPLK    #00F0h,CANRCR  ; reset RMP

        MAR     *+             ; Increment receive counter
        SETC    XF             ; A toggling XF bit indicates
        RPT     #080h          ; that data is being received
        NOP                    ;
        CLRC    XF

LOOP    B       RX_LOOP        ; Execute the receive loop again

GISR1:  RET
GISR2:  RET
GISR3:  RET
GISR4:  RET
GISR5:  RET
GISR6:  RET
PHANTOM:   RET

        .end

Notes: The transmitting node should transmit with an ID of E15 DC35 @ 500 Kbps
```

6.4 LPTX5POL.asm

Transmit loop using Mailbox 5. This program, in conjunction with ECRX0POL, provides a quick and easy way to determine if two 24x/240x DSPs are able to communicate via the CAN bus.

```
*       PROGRAM TO CHECK THE CAN OF 24x/240x DSP                              *
*       LPTX5POL - Transmit loop using Mailbox 5                              *
*   MBX5 used for TRANSMISSIONMBX0 used for RECEPTION                         *
* This program TRANSMITS data to another CAN module using MAILBOX5 .          *
* ECRX0POL program should be running on the remote CAN module. The            *
* receiving CAN module,   after receiving the data packets, will echo the     *
* same data back to the   transmitting module which then verifies the         *
* Xmitted and Received data. The program terminates if there is an            *
* error. Else it loops forever.                                               *
* COMMENTS :
; This program, in conjunction with ECRX0POL, provides a quick and easy way
; to determine if two 24x/240x DSPs are able to communicate via the CAN bus.
; This program does not use any interrupts and employs POLLING. Hence, it
; can be run anywhere in Program memory.
; This program employs message filtering.

    .title     "LPTX5POL"     ; Title
    .include   "240x.h"       ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
    .global    START

;---------------------------------------------------------------------
; Other constant definitions
;---------------------------------------------------------------------
DP_PF1      .set    0E0h           ; Page 1 of peripheral file (7000h/80h)E0
DP_CAN      .set    0E2h            ; CAN Registers (7100h)
DP_CAN2     .set    0E4h           ; CAN RAM (7200h)
KICK_DOG    .macro                 ; Watchdog reset macro
        LDP     #00E0h
        SPLK  #05555h,  WDKEY
        SPLK  #0AAAAh,  WDKEY
        LDP #0h
            .endm
        .text
START:  KICK_DOG
        SPLK #0,60h
        OUT     60h,WSGR           ; Set waitstates for external memory (if used)
        SETC    INTM               ; Disable interrupts
        LDP     #DP_PF1            ; Set PLL to x4 and enable clock to CAN module
        SPLK    #0010h,SCSR1       ; '240xA only - Comment out for '24x
        SPLK    06Fh,WDCR          ; Disable watchdog
        CALL    AR_INIT
        LDP     #225
        SPLK    #00C0H,MCRB        ; Configure CAN pins

        LAR     AR7,#300h          ; *AR7 keeps track of transmit cycles
        MAR     *,AR7
        SPLK    #0h,*
;*********************************************************************
;**************    Disable all mailboxes       ********************
;*********************************************************************
        LDP     #DP_CAN

        SPLK    #1001101011011001b,CANLAM0H   ; Set LAM (9AD9 64D2)
        SPLK    #0110010011010010b,CANLAM0L   ; 1:don't care

        SPLK    #0000000000000000b,CANMDER    ; Disable all mailboxes
;                |||||||||||||||              ; Required before writing
;                FEDCBA9876543210            ; to MSGID
;*********************************************************************
;***********                Write CAN Mailboxes                  ******
;*********************************************************************
; Set MSGIDs for both transmit and receive mailboxes
        LDP     #DP_CAN2

        SPLK    #1000111000010101b,CANMSGID5H     ; Set mailbox 5 ID
;               |||||||||||||||                      ; XMIT Mailbox
;               FEDCBA9876543210                ; 8E15
```

```
;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    #1101110000110101b,CANMSGID5L        ; DC35

;bit 0-15  lower part of extended identifier

        SPLK    #1100110010010001b,CANMSGID0H        ; Set mailbox 0 ID
;                |||||||||||||||||                   ; RCV Mailbox
;                FEDCBA9876543210                    ; CC91

         SPLK   #1101100111111001b,CANMSGID0L ; D9F9

; Write to MSGCTRL register
        SPLK    #0000000000001000b,CANMSGCTRL5       ; 0008
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 0-3   Data length code. 1000 = 8 bytes
;bit 4     0: data frame
;*********************************************************************
;******     ENABLE MBX AFTER WRITING TO MSGID/MSGCTRL     **********
;*********************************************************************

        LDP     #DP_CAN
        SPLK    #0000000000100001b,CANMDER
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 0-5   enable mailboxes 0 & 5
;*********************************************************************
;***********                Write CAN Mailboxes           **********
;*********************************************************************

         LDP    #DP_CAN2
        SPLK    #00123h,CANMBX5A                ; Message to transmit
        SPLK    #04567h,CANMBX5B
        SPLK    #089ABh,CANMBX5C
        SPLK    #0CDEFh,CANMBX5D
;*********************************************************************
;***********  Bit timing registers configuration    ****************
;*********************************************************************

         LDP    #DP_CAN
        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 12 Change configuration request for write-access to BCR (CCR=1)
W_CCE   BIT     CANGSR,#0Bh              ; Wait for Change config Enable
        BCND    W_CCE,NTC                ; bit to be set in GSR
        SPLK    #39,CANBCR2              ; BRP + 1 = 40
; bit 0-7  Baud rate prescaler
; bit 8-15 Reserved
        SPLK    #0000000011111010b,CANBCR1   ; 50 kbps with 85 % sampling pt
;                |||||||||||||||||
;                FEDCBA9876543210
; bit 0-2  TSEG2
; bit 3-6  TSEG1
; bit 7    Sample point setting (1: 3 times, 0: once)
; bit 8-9  Synchronization jump width
; bit A-F  Reserved
        SPLK    #0000000000000000b,CANMCR
;                |||||||||||||||||
;                FEDCBA9876543210
;bit 12    Change conf register
W_NCCE  BIT     CANGSR,#0Bh    ; Wait for Change config disable
        BCND    W_NCCE,TC
;*********************************************************************
;***********                    TRANSMIT                     **********
```

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

W_TA    BIT     CANTCR,BIT15           ; Wait for transmission acknowledge
        BCND    W_TA,NTC

        SPLK    #8000h,CANTCR ; reset TA

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

W_RA    BIT     CANRCR,BIT4    ; Wait for data from remote
        BCND    W_RA,NTC       ; node

        SPLK    #0010h,CANRCR  ; reset RMP and hence CANIFR

                               ; Check if transmitted and received data are same

CHECK   MAR     *,AR5
        LACL    *+,AR0
        XOR     *+,AR6
        BCND    ERROR,NEQ
        BANZ    CHECK
        CALL    AR_INIT

        MAR     *,AR7          ; This loop merely keeps a
        LACL    *              ; count of the number of times
        ADD     #1             ; data packets were transmitted
        SACL    *              ; to the remote node.

        SETC    XF             ; A toggling XF bit indicates
        RPT     #40h           ; that the program is still
        NOP                    ; running. This diagnostic aid
        CLRC    XF             ; slows the traffic of CAN though

LOOP    B       TX_LOOP        ; Start all over again!

ERROR   B       ERROR          ; Keep looping here in case of error..

;***************************************************************************
;***********             COMMON ROUTINES             **********
;***************************************************************************

; AR Initializing routine

AR_INIT LAR     AR0,#7204h     ; AR0 => Mailbox 0 RAM (RECEIVE)
        LAR     AR5,#722Ch        ; AR5 => Mailbox 5 RAM (TRANSMIT)
        LAR     AR6,#03        ; AR6 => Data Counter
        RET

GISR1:
GISR2:
GISR3:
GISR4:
GISR5:
GISR6:
PHANTOM:   RET

        .end

How filtering is achieved :
---------------------------
; Note: The CAN node in which this program is running is referred as ”Xmitting” node

MSGID-MBX5: 0 1110 0001 0101 1101 1100 0011 0101 (Xmit MBX of Xmitting node)
MSGID-MBX0: 0 0111 0011 0101 1001 0100 0101 1001 (Rcv MBX of Receiving node)
      LAM0: 0 1101 1010 0010 1100 1000 1110 1101 (LAM0 of Receiving node)

; Note that wherever there is a zero in the LAM, the bits of the transmitted
; MSGID and the MSGID of the Receive MBX are identical. The corresponding
; bits of transmit and receive MSGIDs could differ only if the corresponding
; LAM bit is a 1. The MSGID of the receive Mailbox is overwritten with the MSGID
; of the received message. Hence, filtering happens only for the first receive.

; Whenever the program on either node is terminated, the contents of memory
; location 300h should be the same (or +/- 1).

/* CANayzer config file: 50k80spRx.cfg */
```

6.5 ECRX0POL.asm

Receive & Echo loop using Mailbox 0. This program, in conjunction with LPTX5POL, provides a quick and easy way to determine if two 24x/240x DSPs are able to communicate via the CAN bus.

```
*       PROGRAM TO CHECK THE CAN OF 24X/240x DSP                          *
*       ECRX0POL - Receive & Echo loop using Mailbox 0                    *
*   MBX0 used for RECEPTION       MBX5 used for TRANSMISSION              *
* This program RECEIVES & ECHOES data to another CAN module.              *
* LPTX5POL program should be running on the other CAN module. This CAN    *
* module, after receiving the data packets, will echo the same data back  *
* to the transmitting module which then verifies the Xmitted and Recvd    *
* data. The program loops forever.                                        *
* COMMENTS :
; This program, in conjunction with LPTX5POL, provides a quick and easy way
; to determine if two 24x/240x DSPs are able to communicate via the CAN bus.
; This program does not use any interrupts and employs POLLING. Hence, it
; can be run anywhere in Program memory.
; This program employs message filtering.

    .title     "ECRX0POL"     ; Title
    .include    240x.h        ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
    .global    START

;-------------------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------------------
DP_PF1      .set    0E0h       ; Page 1 of peripheral file (7000h/80h)E0
DP_CAN      .set    0E2h       ; CAN Registers (7100h)
DP_CAN2     .set    0E4h       ; CAN RAM (7200h)

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

        .text

START:  KICK_DOG               ; Reset Watchdog counter
        SPLK    #0,60h
        OUT     60h,WSGR       ; Set waitstates for external memory (if used)
        SETC    INTM           ; Disable interrupts
        LDP     #DP_PF1        ; Set PLL to x4 and
        SPLK    #0010h,SCSR1   ; enable clock to CAN module
        SPLK    06Fh,WDCR      ; Disable watchdog
        CALL    AR_INIT
        LDP     #225
        SPLK    #00C0H,MCRB    ; Configure CAN pins

        LAR     AR7,#300h      ; AR7 keeps track of Receive cycles
        MAR     *,AR7
        SPLK    #0h,*

;************************************************************************
;**************    Disable all mailboxes      ************************
;************************************************************************

        LDP     #DP_CAN

        SPLK    #1000110110100010b,CANLAM0H   ; Set LAM (8DA2 C8ED)
        SPLK    #1100100011101101b,CANLAM0L       ; 1:don't care

        SPLK    #0000000000000000b,CANMDER    ; Disable all mailboxes
;                ||||||||||||||||              ; Required before writing
;                FEDCBA9876543210              ; to MSGID

;************************************************************************
;************                Write CAN Mailboxes                 **********
;************************************************************************
; Set MSGIDs for both transmit and receive mailboxes
        LDP     #DP_CAN2

        SPLK    #1001011001011000b,CANMSGID5  H   ; Set mailbox 5 ID
;                ||||||||||||||||                  ; XMIT Mailbox
;                FEDCBA9876543210                  ; 9658
```

```
         SPLK   #1101110101101011b,CANMSGID5L     ; DD6B

        SPLK    #1100011100110101b,CANMSGID0H     ; Set mailbox 0 ID
;                |||||||||||||||||                 ; RCV Mailbox
;                FEDCBA9876543210                 ; C735

         SPLK   #1001010001011001b,CANMSGID0L     ; 9459

        SPLK    #0000000000001000b,CANMSGCTRL5    ; 0008
;                |||||||||||||||||
;                FEDCBA9876543210

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

;*********************************************************************
;******     ENABLE MBX AFTER WRITING TO MSGID/MSGCTRL     **********
;*********************************************************************

        LDP     #DP_CAN
        SPLK    #0000000000100001b,CANMDER
;                |||||||||||||||||
;                FEDCBA9876543210

;bit 0-5   enable mailboxes 0 & 5

;*********************************************************************
;***********  Bit timing registers configuration    ****************
;*********************************************************************

        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||||
;                FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,#0Bh               ; Wait for Change config Enable
        BCND    W_CCE,NTC                 ; bit to be set in GSR

        SPLK    #39,CANBCR2               ; BRP + 1 = 40

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

        SPLK    #0000000011111010b,CANBCR1; 50 kbps with 85 % sampling pt
;                |||||||||||||||||
;                FEDCBA9876543210

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

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

;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh               ; Wait for Change config disable
        BCND    W_NCCE,TC

;*********************************************************************
;***********                    RECEIVE                     **********
;*********************************************************************
RX_LOOP:
W_RA    BIT     CANRCR,BIT4           ; Wait for data from remote node
        BCND    W_RA,NTC

        SPLK    #0010h,CANRCR         ; reset RMP and hence CANIFR

                ; Copy received data from MBX0 into MBX5 for transmission

COPY    MAR     *,AR0
        LACL    *+,AR5
        SACL    *+,AR6
        BANZ    COPY
        CALL    AR_INIT

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

W_TA    BIT     CANTCR,BIT15          ; Wait for transmission acknowledge
        BCND    W_TA,NTC
```

```
        SPLK    #8000h,CANTCR          ; reset TA

        SETC    XF                     ; A toggling XF bit indicates
        RPT     #080h                  ; that the program is still
        NOP                            ; running.
        CLRC    XF

        MAR     *,AR7                  ; This loop merely keeps a
        LACL    *                      ; count of the number of times
        ADD     #1                     ; data packets were received
        SACL    *                      ; from the transmitting node.
LOOP    B       RX_LOOP

;***************************************************************************
;***********               COMMON ROUTINES              **********
;***************************************************************************

; AR Initializing routine

AR_INIT LAR     AR0,#7204h         ; AR0 => Mailbox 0 RAM (RECEIVE)
        LAR     AR5,#722Ch         ; AR5 => Mailbox 5 RAM (TRANSMIT)
        LAR     AR6,#03            ; AR4 => Counter
        RET

GISR1:
GISR2:
GISR3:
GISR4:
GISR5:
GISR6:
PHANTOM:    RET

            .end

/* CANayzer config file: 50k80spRx.cfg */
```

6.6 REM–REQ.asm

This program transmits a remote frame and expects a data frame in response. Transmission of a remote frame by (and reception of the data frame in) MBX3. To be used along with REM–ANS.asm

```
*    PROGRAM TO TRANSMIT A REMOTE FRAME REQUEST IN THE 24x/240xA CAN
* This program transmits a remote frame and expects a data frame in response
* Transmission of a remote frame by (and reception of the data frame in) MBX3
* To be used along with REM-ANS.asm


    .title     "REM_REQ"      ; Title
    .include    "240x.h"      ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
    .global    START
;-------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------
DP_PF1      .set    0E0h       ; Page 1 of peripheral file (7000h/80h
DP_CAN      .set    0E2h       ; Can Registers (7100h)
DP_CAN2     .set    0E4h       ; Can RAM (7200h)

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


;===============================================================================
; M A I N   C O D E - starts here
;===============================================================================
       .text
START:  KICK_DOG               ; Reset Watchdog counter
       SPLK    #0,60h
       OUT     60h,WSGR        ; Set waitstates for external memory (if used)
       SETC    INTM            ; Disable interrupts
       SPLK    #0000h,IMR      ; Mask all core interrupts
       LDP     #0E0h
       SPLK    #006Fh, WDCR    ; Disable WD
       SPLK    #0010h,SCSR1    ; Enable clock to CAN module (For 240xA only)

       LDP #225
       SPLK #00C0H,MCRB ; Configure CAN pins

       LDP #DP_CAN
       SPLK #1011111111111111b,CANIMR ; Enable all CAN interrupts

;*******************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX3  **********
;*******************************************************************

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


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

       LDP     #DP_CAN2


       SPLK    #1001111111111111b,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
;                ||||||||||||||||
;                FEDCBA9876543210

;bit 0-15  lower part of extended identifier

        SPLK    #0000000000011000b,CANMSGCTRL3
;                ||||||||||||||||
;                FEDCBA9876543210

;bit 0-3   Data length code. 1000 = 8 bytes
;bit 4     1: Remote frame

;*************************************************************************
;***********          Enable Mailbox                          **********
;*************************************************************************

        LDP     #DP_CAN

        SPLK    #0000000010001000b,CANMDER
;                ||||||||||||||||
;                FEDCBA9876543210

;bit 0-5   enable mailbox 3
;bit 7     1: mailbox 3 = receive


;*************************************************************************
;***********    Bit timing Registers configuration   **********************
;*************************************************************************

        SPLK    #0001000000000000b,CANMCR
;                ||||||||||||||||
;                FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,#0Bh ; Wait for Change config Enable
        BCND    W_CCE,NTC  ; bit to be set in GSR

         ;SPLK  #0000000000000000b,CANBCR2    ; For 1 Mbps @ 20 MHz CLKOUT
         SPLK   #0000000000000001b,CANBCR2    ; For 1 Mbps @ 40 MHz CLKOUT
;                ||||||||||||||||
;                FEDCBA9876543210

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

        SPLK    #0000000011111010b,CANBCR1    ; For 1 Mbps @ 85 % samp. pt
;                ||||||||||||||||
;                FEDCBA9876543210

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

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

;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh                   ; Wait for Change config disable
        BCND    W_NCCE,TC


;*************************************************************************
;***********                        TRANSMIT                     **********
;*************************************************************************
        SPLK    #0020h,CANTCR         ; Transmit request for MBX3

W_TA    BIT     CANTCR,2              ; Wait for transmission acknowledge
        BCND    W_TA,NTC
        SPLK    #2000h,CANTCR         ; reset TA

RX_LOOP:
W_RA    BIT     CANRCR,BIT7           ; Wait for data from remote node
        BCND    W_RA,NTC              ; to be written into MBX3

LOOP    B       LOOP
```

```
GISR1:
GISR2:
GISR3:
GISR4:
GISR5:
GISR6:
PHANTOM RET
        .end
```

6.7 REM–ANS.asm

Program to auto–answer a remote frame request in CAN. To be used along with REM–REQ.asm.

```
* PROGRAM TO AUTO-ANSWER TO A REMOTE FRAME REQUEST IN 24x/240xA CAN    *
* To be used along with REM-REQ.asm                                          *
* Reception and transmission by MBX2. Low priority interrupt used *

; Transmit acknowledge for MBX2 is set after running this program
; and the message is transmitted.

    .title     "REM_ANS"      ; Title
    .include    "240x.h"      ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
    .global    START

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

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

;===============================================================================
; M A I N   C O D E - starts here
;===============================================================================
       .text
START:  KICK_DOG              ; Reset Watchdog counter
       SPLK    #0,60h
       OUT     60h,WSGR           ; Set waitstates for external memory (if used)
       LDP     #0E0h
       SPLK #006Fh, WDCR ; Disable WD
       SPLK #0010h,SCSR1 ; Enable clock to CAN module (For 240xA only)

       LDP #225
       SPLK #00C0H,MCRB ; Configure CAN pins

;*******************************************************************
; Enable 1 core interrupt
;*******************************************************************

       LDPK #0
       SPLK    #0000000000010000b, IMR   ; core interrupt mask register
;               ||||||||||||||||             ; Enable INT5 for CAN
;               FEDCBA9876543210
       SPLK    #000FFh,IFR                  ; Clear all core interrupt flags
       CLRC    INTM                         ; enable interrupt

       LDP #DP_CAN
       SPLK #1011111111111111b,CANIMR ; Enable all CAN interrupts

;*******************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX2  **********
;*******************************************************************

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

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

       LDP     #DP_CAN2

       SPLK    #1011111111111111b,CANMSGID2H
;               ||||||||||||||||
;               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,CANMSGID2L
;                |||||||||||||||
;                FEDCBA9876543210

;bit 0-15  lower part of extended identifier

        SPLK    #0000000000001000b,CANMSGCTRL2
;                |||||||||||||||
;                FEDCBA9876543210

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

         LDP    #DP_CAN
        SPLK    #0000000100000000b,CANMCR ; Set CDR bit before writing
;                |||||||||||||||
;                FEDCBA9876543210

         LDP    #DP_CAN2
        SPLK    #0BEBEh,CANMBX2A          ; Message to transmit
        SPLK    #0BABAh,CANMBX2B
        SPLK    #0DEDEh,CANMBX2C
        SPLK    #0DADAh,CANMBX2D

        LDP     #DP_CAN
        SPLK    #0000000000000000b,CANMCR ; Clear CDR bit after writing
;                |||||||||||||||
;                FEDCBA9876543210

;***************************************************************************
;***********          Enable Mailbox                          **********
;***************************************************************************

        SPLK    #0000000000000100b,CANMDER
;                |||||||||||||||
;                FEDCBA9876543210

;bit 0-5   Enable MBX2
;bit 6     MBX2 configured as Transmit MBX

;***************************************************************************
;***********    Bit timing Registers configuration   **********************
;***************************************************************************

        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||
;                FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,#0Bh ; Wait for Change config Enable
        BCND    W_CCE,NTC   ; bit to be set in GSR

         ;SPLK #0000000000000000b,CANBCR2 ; For 1 Mbps @ 20 MHz CLKOUT
         SPLK   #0000000000000001b,CANBCR2; For 1 Mbps @ 40 MHz CLKOUT
;                |||||||||||||||
;                FEDCBA9876543210

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

        SPLK    #0000000011111010b,CANBCR1    ; For 1 Mbps @ 85 % samp. pt
;                |||||||||||||||
;                FEDCBA9876543210

; bit 0-2  TSEG2
; bit 3-6  TSEG1
; bit 7    Sample point setting (1: 3 times, 0: once)
; bit 8-9  Synchronization jump width
; bit A-F  Reserved
        SPLK    #0000000000000000b,CANMCR
;                |||||||||||||||
;                FEDCBA9876543210

;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh               ; Wait for Change config disable
        BCND    W_NCCE,TC
```

```
ELOOP      B   ELOOP                                ; Wait for Receive Interrupt

;=====================================================================
; ISR used to toggle XF to indicate remote frame was received
;=====================================================================

GISR5:

LOOP2   MAR     *,AR0
        SETC    XF
        CALL    DELAY
        CLRC    XF
        CALL    DELAY
        B       LOOP2

DELAY   LAR     AR0,#0FFFFh
LOOP    RPT     #080h
        NOP
        BANZ    LOOP
        RET

GISR1:      RET
GISR2:      RET
GISR3:      RET
GISR4:      RET
GISR6:      RET

PHANTOM     RET

        .end

; When data in MBX2 is transmitted in response to a ”Remote frame request,”
; XF is toggled. Note that TRS bit is not set for MBX2. The transmission of
; MBX2 data is automatic ,in response to a ”Remote frame request.”
```

6.8 PWRDOWN.asm

Program to illustrate entry/exit into/out of Low–power mode of the CAN module.

```
*       PWRDOWN  - Demonstrates the powerdown feature of CAN     *
* This program puts the CAN module in low-power mode.              *
; XF is pulsed slowly after the CAN module comes out of LPM
    .title     "TXLOOP"       ; Title
    .include    "240x.h"      ; Variable and register declaration
    .include    "vector.h"    ; Vector table (takes care of dummy password)
    .globalSTART

;-------------------------------------------------------------------------
; Other constant definitions
;-------------------------------------------------------------------------
DP_PF1      .set    0E0h       ; Page 1 of peripheral file (7000h/80h)E0
DP_CAN      .set    0E2h       ; CAN Registers (7100h)
DP_CAN2     .set    0E4h    ; CAN RAM (7200h)
KICK_DOG    .macro             ; Watchdog reset macro
        LDP     #00E0h
        SPLK  #05555h,  WDKEY
        SPLK  #0AAAAh,  WDKEY
        LDP #0h
        .endm
        .text
START:  KICK_DOG           ; Reset Watchdog counter
        SPLK    #0,60h
        OUT     60h,WSGR       ; Set waitstates for external memory (if used)
        SETC    INTM           ; Disable interrupts
        SPLK    #0000h,IMR     ; Mask all core interrupts
        LDP     #0E0h
        SPLK    #006Fh, WDCR   ; Disable WD
        SPLK    #0010h,SCSR1   ; Enable clock to CAN module (For 240xA only)
        LDP     #225
        SPLK    #00C0H,MCRB    ; Configure CAN pins
        LDP     #DP_CAN        ; Enable all CAN interrupts. This is reqd
        SPLK    #03F7Fh,CANIMR ; to poll flags.
;***************************************************************************
;******     DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX5  **********
;***************************************************************************
        SPLK    #0000000000000000b,CANMDER   ; Disable all mailboxes
;                |||||||||||||||
;                FEDCBA9876543210
;***************************************************************************
;***********    Set MSGID/MSGCTRL for transmit mailbox           **********
;***************************************************************************
        LDP     #DP_CAN2
        SPLK    #1010111000010101b,CANMSGID5H ; Set mailbox 5 ID
;                |||||||||||||||                ; XMIT Mailbox
;                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    #1101110000110101b,CANMSGID5L ; AE15 DC35 --> ID
;                |||||||||||||||
;                FEDCBA9876543210
;bit 0-15  lower part of extended identifier
        SPLK    #0000000000001000b,CANMSGCTRL5    ; 0008
;                |||||||||||||||
;                FEDCBA9876543210
;bit 0-3   Data length code. 1000 = 8 bytes
;bit 4     0: data frame
```

```
        SPLK    #1000000000000000b,CANMSGID0H
        SPLK    #0000000000000001b,CANMSGID0L

;*************************************************************************
;******     ENABLE MBX AFTER WRITING TO MSGID/MSGCTRL OF MBX5 **********
;*************************************************************************

        LDP     #DP_CAN
        SPLK    #0000000000100001b,CANMDER
;                |||||||||||||||
;                FEDCBA9876543210

;bit 0-5   enable mailbox 5

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

        LDP     #DP_CAN2
        SPLK    #00100h,CANMBX5A                ; Message to transmit
        SPLK    #00302h,CANMBX5B
        SPLK    #00504h,CANMBX5C
        SPLK    #00706h,CANMBX5D

;*************************************************************************
;***********    Bit timing Registers configuration   *********************
;*************************************************************************

        LDP     #DP_CAN
        SPLK    #0001000000000000b,CANMCR
;                |||||||||||||||
;                FEDCBA9876543210

;bit 12    Change configuration request for write-access to BCR (CCR=1)

W_CCE   BIT     CANGSR,BIT4 ; Wait for Change config Enable
        BCND    W_CCE,NTC      ; bit to be set in GSR

         ;SPLK  #0000000000000000b,CANBCR2    ; For 1 Mbps @ 20 MHz CLKOUT
         SPLK   #0000000000000001b,CANBCR2    ; For 1 Mbps @ 40 MHz CLKOUT
;                |||||||||||||||
;                FEDCBA9876543210

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

        SPLK    #0000000011111010b,CANBCR1        ; For 1 Mbps @ 85 % samp. pt
;                |||||||||||||||
;                FEDCBA9876543210

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

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

;bit 12    Change conf register

W_NCCE  BIT     CANGSR,#0Bh ; Wait for Change config disable
        BCND    W_NCCE,TC

;*************************************************************************
;***********   TRANSMIT & DRIVE CAN INTO LPM                  **********
;*************************************************************************

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

W_TA    BIT     CANTCR,BIT15              ; Wait for transmission acknowledge
        BCND    W_TA,NTC

        SPLK    #8000h,CANTCR ; reset TA

        SPLK    #0000101000000000b,CANMCR ; Set PDR & WUBA bits = 1
;                |||||||||||||||
;                FEDCBA9876543210

W_PDA   BIT     CANGSR,BIT3               ; Wait for PDA
        BCND    W_PDA,NTC      ; bit to be set in GSR

; CAN is now in LPM waiting for some data on the CAN bus to wake it up..
```

```
W_PDAe  BIT     CANGSR,BIT3           ; Wait for PDA bit to be cleared
        SETC    XF                            ; A fast toggling XF bit indicates
        RPT     #0FFh                         ; that the module is waiting to be
        NOP                                   ; pulled out of LPM
        CLRC    XF
        RPT     #0FFh
        NOP
        BCND    W_PDAe,TC

LOOP3
LOOP2   MAR     *,AR0                    ; A slowly toggling XF bit indicates
        SETC    XF                              ; that the module has come out of LPM
        CALL    DELAY
        CLRC    XF
        CALL    DELAY
        B       LOOP2

DELAY   LAR     AR0,#0FFFFh
LOOP    RPT     #080h
        NOP
        BANZ    LOOP
        RET
        B       LOOP3                           ; Loop here after exiting LPM

GISR1:
GISR2:
GISR3:
GISR4:
GISR5:
GISR6:
PHANTOM:    RET
        .end
```

7 Reference

CAN Specification 2.0, Part A and B. Robert Bosch GmbH, Stuttgart.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265