CHAPTER 10

Control Area Network (CAN) Communications

CAN Bus Communication Study Guideline

Read the following materials:

[1] TMS320LF/LC240x DSP Controllers Reference Guide: System and Peripherals, TI Literature Number: SPRU357

Chapter 10: CAN Controller Module.

[2] Application Report: Programming Examples for the 24x/240xA CAN – TI Literature Number: SPRA890A by Hareesh Janakiraman of Advanced Embedded Control Group.

[3] The example presented below.

Pay attention to the following terms: *node*, *data frame*, *remote frame*, *RTR*, *identifier*, *identifier extension*, *mailbox*, *auto-answer mode* (*AAM*), *acceptance filter*, *transmission request*, *transmission acknowledge*, *receive message pending*, *change configuration request* (*CCR*), *change data field request* (*CDR*), *nominal bit time*, *change configuration enable*, and *mailbox interrupt*.

The Example

Two 2407 EVM nodes, namely Node 1 and Node 2, are connected on the CAN bus. Node 1 is supposed to send a *remote frame* to MBX2 on Node 2 from MBX3 to request data. Node 2 responds to this *remote frame* by transmitting a *data frame* from its MBX2 to MBX3 on Node 1. Node 1 acknowledges the receiving of the *data frame*.

Identifier extension is used on both nodes. MBX3 of Node 1 and MBX2 of Node 2 have the same *identifier* and *identifier extension*. MBX2 of Node 2 is set to be *auto-answer mode*, therefore no *transmission request* needs to be given by CPU to transmit the *data frame*. The TAn bits in the transmission control register TCR cannot be used to

acknowledge the transmission of *remote frame*, but a *data frame*, since there is 0 byte of data is transmitted from the *mailbox*. Therefore, Node 1 does not check this acknowledgment while Node 2 does.

In the following example programs, `RMT_REQ.ams` is for Node 1 and `RMT_ANS.asm` is for Node 2.

For observing real-time changes of the DSP while running, both *nodes* are set under real-time running mode. If a memory window is opened under Code Composer IDE of Node 1 to monitor *mailbox* data field of MBX3 by typing in "MBX3A" in the "Address" field, the change of the contents of the 4 consecutive memory locations can be observed in real time.

The real-time running mode also allows user to use software *trigger* technique to control the execution of the program. The *trigger* is a variable used to control the execution at run time. In `RMT_REQ.ams`, the *trigger* variable `trigger` is preset to be 1 and the program will be trapped to an infinite loop before the *remote frame* is transmitted. User can change the value of `trigger` from 1 to 0 to let the program jump out of the loop and start transmitting. This value change has to be accomplished in a memory window under Code Composer. This feature of Code Composer allows user to interfere a running program by modifying variables in real time.

The register settings in both programs are clearly commented. Please read carefully.

The procedures of doing the test correctly are listed as follows:

1) Load `RMT_REQ.out` on Node 1 and `RMT_ANS.out` on Node 2. Run them under real time mode by following the process: Reset DSP → `go MON_GO` → Enable real-time mode → Run.
2) Open a memory window under Node 1 Code Composer, type in "MBX3A" in "Address" field and click "OK", change the size of the window with mouse to

allow 4 consecutive memory locations shown in the window. They are MBX3A, MBX3B, MBX3C, and MBX3D, respectively.

3) Open a memory window under Node 1 Code Composer for variable `trigger` in the same way as above. Double-click on the content of `trigger`, modify the value, and click "Done" button. The transmission of the *remote frame* is triggered.
4) The content of the MBX3 window opened in Step 2 should be changed into the preset Node 2 MBX2 content immediately after the triggering, which indicates a successful test.

The complete code is given below –

Node 1 code:

```
;*********************************************************************
; File Name:       RMT_REQ.asm
; Target System: C240x Evaluation Board
; Description:     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.
;
;                  Real-time mode enabled. MBX3 values can be observed
;                  in real time.
;
;                  To be used along with REM-ANS.asm
;*********************************************************************

      .title      "RMT_REQ"         ; Title

;                        SYSTEM OPTIONS
;*********************************************************************
real_time   .set  1     ; 1 for real time mode, otherwise set 0
;*********************************************************************

    .include "f2407.h"        ; Variable and register declaration
    .include "c200mnrt.i"     ; Include conditional assembly options.
    .global  _c_int0,GISR1,GISR2,GISR3,GISR4,GISR5,GISR6,PHANTOM
    .global  MON_RT_CNFG
    .ref     SYS_INIT
    .bss     trigger,1  ; launch the remote frame transmit by change
                        ; the value of trigger from 1 to 0 in real time.

;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
            .text

_c_int0     CALL SYS_INIT
```

```
;---------------------------------------------------------
; Initialise the Real time monitor
;---------------------------------------------------------
;---Real Time option--------------
      .if (real_time)
            CALL        MON_RT_CNFG       ;For Real-Time
      .endif
;---------------------------------

;---------------------------------------------------------
; System Interrupt Init.
;---------------------------------------------------------

;---Real Time option -----------------------------------------------------
      .if (real_time)
            SPLK  #0000000001000000b,IMR  ;En Int  7 for  RT
                  ;||||||||!||||||
                  ;5432109876543210
      .endif

      .if (real_time != 1)
            SPLK  #0000000000000000b,IMR  ;Disable all INT's
                  ;||||!!!!!!!!!!!!
                  ;5432109876543210
      .endif

            SPLK  #0FFFFh, IFR            ;Clear any pending Ints
;-----------------------------------------------------------------------

      LDP   #DP_PF2           ; Load Date Page for GP IO registers

      SPLK  #00C0H,MCRB       ; Configure CAN pins

      LDP   #DP_CAN1
      SPLK  #1011111111111111b,CAN_IMR    ; Enable all CAN interrupts

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

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

;***********************************************************************
;********                Write CAN Mailboxes                  **********
;***********************************************************************
      LDP   #DP_CAN2
      SPLK  #1001111111111111b,MSGID3H
;             ||||||||||||||||
;             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,MSGID3L
;             ||||||||||||||||
;             FEDCBA9876543210
```

```
;bit 0-15   lower part of extended identifier

      SPLK  #0000000000011000b,MSGCTRL3
;             ||||||||||||||||
;             FEDCBA9876543210

;bit 0-3    Data length code. 1000 = 8 bytes
;bit 4            1: this is a Remote Frame

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

      LDP   #DP_CAN1
      SPLK  #0000000010001000b,MDER
;             ||||||||||||||||
;             FEDCBA9876543210

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

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

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

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

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

      ;SPLK #0000000000000000b,BCR2 ; For 1 M bits/s @ 20 MHz CLKOUT
      SPLK  #0000000000000001b,BCR2 ; For 1 M bits/s @ 40 MHz CLKOUT
;             ||||||||||||||||
;             FEDCBA9876543210

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

      SPLK  #0000000011111010b,BCR1     ; For 1 M bits/s @ 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,MCR
;             ||||||||||||||||
;             FEDCBA9876543210

;bit 12     Change conf register
```

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

      LDP         #trigger
      SPLK  #1,trigger  ; Initialize the trigger value to 1
TRIGGER
      NOP
      LACC  trigger
      BCND  TRIGGER,NEQ ; Wait till the value of trigger is modified
                        ; to 0 by Code Composer in real time

      LDP         #DP_CAN1

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

;W_TA BIT   TCR,2 ; Wait for transmission acknowledge
                  ; (not useful for remote frame, so commented out)
;     BCND  W_TA,NTC
;     SPLK  #2000h,TCR  ; reset TA


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

LOOP  B     LOOP

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

Node 2 code:

```
;*********************************************************************
; File Name:      RMT_ANS.asm
; Target System:  C240x Evaluation Board
; Description:    Program to auto-answer to a remote frame request in
;                 24x/240xA CAN. Reception and transmission by MBX2.
;                 Low priority interrupt used.
;
;                 Transmit acknowledge for MBX2 is set after the message
;                 is transmitted.
;
;                 Real-time mode enabled.
;
;                 To be used along with REM-REQ.asm
;*********************************************************************

      .title      "RMT_ANS"   ; Title
;*********************************************************************
```

```
;                                SYSTEM OPTIONS
;*********************************************************************
real_time   .set  1      ; 1 for real time mode, otherwise set 0
;*********************************************************************
*******
    .include  "f2407.h"       ; Variable and register declaration
    .include  "c200mnrt.i"    ; Include conditional assembly options.
    .global   _c_int0,PHANTOM,GISR1,GISR2,GISR3,GISR4,GISR5,GISR6
    .global   MON_RT_CNFG
    .ref      SYS_INIT
; User variables:
    .bss      ctr,1           ;counter for background loop

;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
            .text

_c_int0      CALL SYS_INIT
;-------------------------------------------------------
; Initialise the Real time monitor
;-------------------------------------------------------
;---Real Time option---------------
      .if (real_time)
            CALL        MON_RT_CNFG       ;For Real-Time
      .endif
;---------------------------------

;-------------------------------------------------------
; System Interrupt Init.
;-------------------------------------------------------

;---Real Time option ------------------------------------------------
      .if (real_time)
            SPLK  #0000000001010000b,IMR  ;En Int INT5,7 for CAN and RT
                  ;||||||||!||||||
                  ;5432109876543210
      .endif

      .if (real_time != 1)
            SPLK  #0000000000010000b,IMR  ;Enable Int 5 only for CAN
                  ;||||!!!!!!!!!!!!
                  ;5432109876543210
      .endif

      SPLK  #0FFFFh, IFR            ;Clear any pending Ints
;---------------------------------------------------------------------

      LDP   #DP_PF2           ; Load Data Page for GPIO
      SPLK  #00C0H,MCRB       ; Configure CAN pins

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

      LDPK  #0
      SPLK  #000FFh,IFR       ; Clear all core interrupt flags

      LDP   #DP_CAN1
```

```
      SPLK  #1011111111111111b,CAN_IMR    ; Enable all CAN interrupts

;*********************************************************************
;******  DISABLE MBX BEFORE WRITING TO MSGID/MSGCTRL OF MBX2  **********
;*********************************************************************
      SPLK  #0000000000000000b,MDER
;             ||||||||||||||||
;             FEDCBA9876543210

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

      LDP   #DP_CAN2

      SPLK  #1011111111111111b,MSGID2H
;             ||||||||||||||||
;             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,MSGID2L
;             ||||||||||||||||
;             FEDCBA9876543210

;bit 0-15   lower part of extended identifier

      SPLK  #0000000000001000b,MSGCTRL2
;             ||||||||||||||||
;             FEDCBA9876543210

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

      LDP   #DP_CAN1
      SPLK  #0000000100000000b,MCR  ; Set CDR bit before writing
;             ||||||||||||||||
;             FEDCBA9876543210

      LDP   #DP_CAN2
      SPLK  #0BEBEh,MBX2A     ; Message to transmit
      SPLK  #0BABAh,MBX2B
      SPLK  #0DEDEh,MBX2C
      SPLK  #0DADAh,MBX2D

      LDP   #DP_CAN1
      SPLK  #0000000000000000b,MCR  ; Clear CDR bit after writing
;             ||||||||||||||||
;             FEDCBA9876543210

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

      SPLK  #0000000000000100b,MDER
;             ||||||||||||||||
```

```
;            FEDCBA9876543210

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

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

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

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

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

      ;SPLK #0000000000000000b,BCR2     ; For 1 M bits/s @ 20 MHz CLKOUT
      SPLK  #0000000000000001b,BCR2     ; For 1 M bits/s @ 40 MHz CLKOUT
;            ||||||||||||||||
;            FEDCBA9876543210

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

      SPLK  #0000000011111010b,BCR1     ; For 1 M bits/s @ 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,MCR
;            ||||||||||||||||
;            FEDCBA9876543210

;bit 12     Change conf register

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

      EINT                    ; Enable interrupt
;=======================================================
;Main system background loop
;=======================================================
      LDP   #ctr
MAIN: SPLK  #7FFFh, ctr ;An infinite loop is running in background
loop: LACC  ctr         ;doing counter decrementing operation while
      SUB   #1          ;waiting for next interrupt request.
      SACL  ctr         ;This background loop can be used to test
      BCND  loop,NEQ    ;the real-time running mode
      B     MAIN

;=================================================================
```

```
; GISR5 MBXn (n=2 here) receive interrupt
;     - check the transmit acknowledge after remote frame was received
;=================================================================

GISR5 NOP
      ;Context save regs
      MAR   *,AR1       ;AR1 is stack pointer
      MAR   *+          ;skip one position
      SST   #1, *+      ;save ST1
      SST   #0, *+      ;save ST0
      SACH  *+          ;save ACC high
      SACL  *           ;save ACC low

;=========================================================
;Start main section of ISR
;=========================================================
W_TA  BIT   TCR,3       ; Wait for transmission acknowledge
      BCND  W_TA,NTC
      SPLK  #1000h,TCR  ; reset TA, clear CAN_IFR flag bit

DELAY RPT   #080h
      NOP

;=========================================================
;End main section of ISR
;=========================================================
      ;Context restore regs
      POINT_PG0         ;load data page 0
      MAR   *, AR1      ;make stack pointer active
      LACL  *-          ;Restore Acc low
      ADDH  *-          ;Restore Acc high
      LST   #0, *-      ;load ST0
      LST   #1, *-      ;load ST1
      EINT              ;Enable all interrupts; clear the INTM to 0.
      RET

GISR1       RET
GISR2       RET
GISR3       RET
GISR4       RET
;GISR5      RET
GISR6       RET

PHANTOM     RET

            .end

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

Laboratory Experiment 9

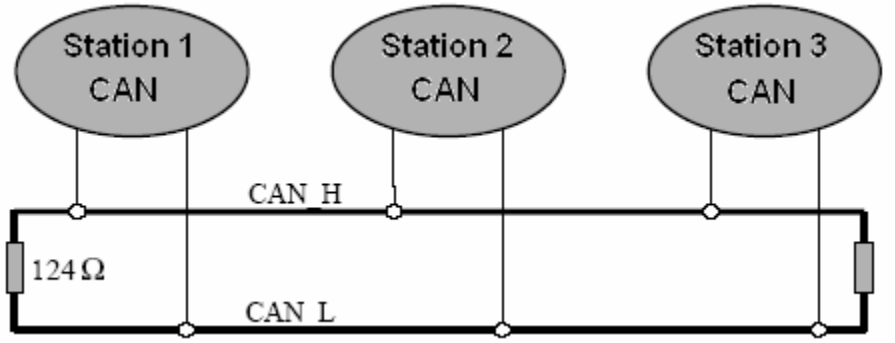
Objectives

To practice programming on 2407 DSP to perform CAN bus communications in a multi-node system and experience the capability and flexibility of CAN.

The Experimental Setup

Hardware:

- Three stations, each of which consists of a PC and a 2407 EVM target board with emulator/parallel port connection to the PC.
- A laboratory CAN bus board with the CAN termination resistors installed.
- Three CAN cables with miniature DIN connectors on both ends. The connection is as follows:



Software:

- Windows XP Professional
- Code Composer 4.12
- User source code

Laboratory Assignments

1. Run the example programs `RMT_REQ.asm` and `RMT_ANS.asm` to perform two station communications and observe the data transmission in real time. This test can be conducted together with any other group.
2. Relay operation of three nodes. Write a program for your own group to perform 3-station communications with CAN bus. Three TMS320LF2407 EVMs, namely

Station 1, Station 2, and Station 3, are connected to the bus with their P7 ports. Each module receives 4 16-bit integers from one other module, increments them by 1, and sends the updated ones to the third module in 1 second. The data flow is in a “relay” manner initiated by Station 1 and follows the direction:

Station 1 => Station 2 => Station 3 => Station 1 ...

which forms an infinite relay loop. On each module, the data is updated every 3 seconds and incremented by 3. Each module uses MBX0 to receive and MBX5 to transmit. All activated mailbox share the same identifier (bit 0-12 of MSGIDnH) 1111111111111b, but the identifier extensions are different as follows:

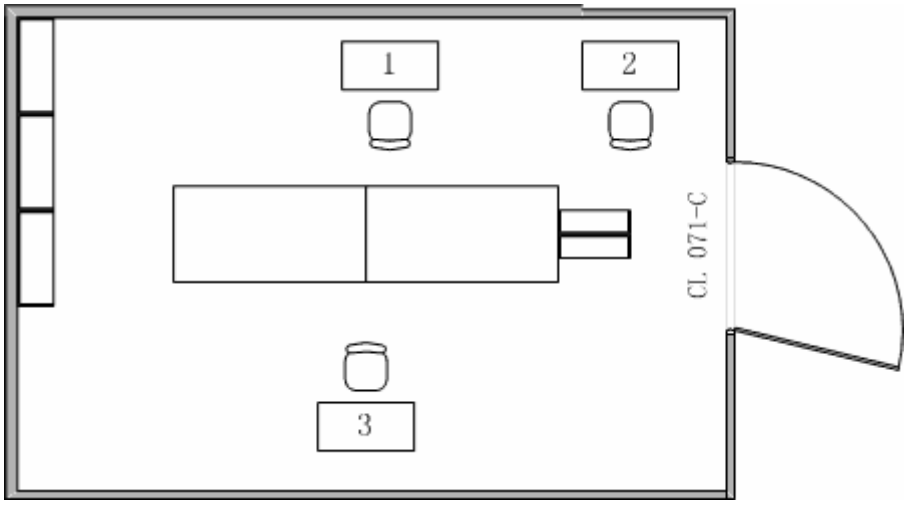
MSGID0L of Station 2 = MSGID5L of Station 1 = 01b

MSGID0L of Station 3 = MSGID5L of Station 2 = 10b

MSGID0L of Station 1 = MSGID5L of Station 3 = 11b

Two interrupts are used: Timer 1 period interrupt and CAN MIFn mailbox receive interrupt. The T1PR is programmed to 0.25 second on a 30MHz DSP. Therefore 4 cycles are needed for each second. On Station 1, the relay process is launched by sending a data frame of 4 initial integers to Station 2. This launching is triggered by a changing the value of a variable "trigger" from 1 to 0 by user in real time. Real-time mode is enabled. User can observe the data refreshing continuously by opening two memory windows for MBX0A and MBX5A respectively.

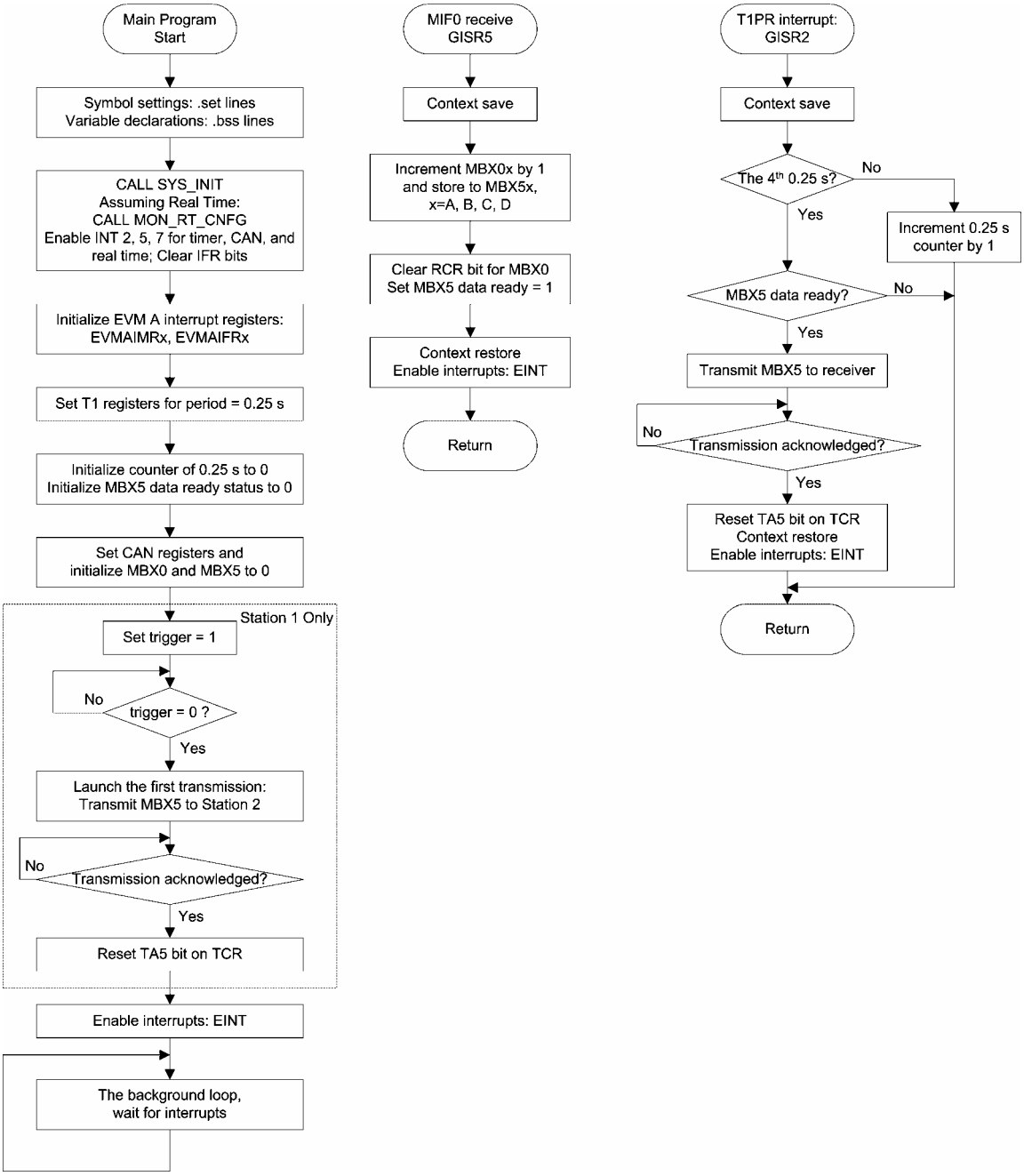
The three stations in the lab are defined as follows:



The desired way to perform the test is that every group runs its own program on its own station and performs relay operation together with the other two groups' programs.

To achieve this goal, the test and debugging process can be split into two steps. The first step, each group tests whether it can receive data from its upper reach group and transmit data to its lower reach group. If all groups pass the first step of test, the second step can be taken to link the receiving and sending up, i.e., to test the relay operation.

In order to reduce the difficulty, a set of suggested flow charts are given for reference.


Main Program Start
Symbol settings: .set lines
Variable declarations: .bss lines
CALL SYS_INIT
Assuming Real Time:
CALL MON_RT_CNFG
Enable INT 2, 5, 7 for timer, CAN, and real time; Clear IFR bits
Initialize EVM A interrupt registers: EVMAIMRx, EVMAIFRx
Set T1 registers for period = 0.25 s
Initialize counter of 0.25 s to 0
Initialize MBX5 data ready status to 0
Set CAN registers and initialize MBX0 and MBX5 to 0
Station 1 Only
Set trigger = 1
No
trigger = 0 ?
Yes
Launch the first transmission: Transmit MBX5 to Station 2
No
Transmission acknowledged?
Yes
Reset TA5 bit on TCR
Enable interrupts: EINT
The background loop, wait for interrupts
MIF0 receive GISR5
Context save
Increment MBX0x by 1 and store to MBX5x, x=A, B, C, D
Clear RCR bit for MBX0
Set MBX5 data ready = 1
Context restore
Enable interrupts: EINT
Return
T1PR interrupt: GISR2
Context save
The 4th 0.25 s?
No
Yes
Increment 0.25 s counter by 1
MBX5 data ready?
No
Yes
Transmit MBX5 to receiver
No
Transmission acknowledged?
Yes
Reset TA5 bit on TCR
Context restore
Enable interrupts: EINT
Return