CHAPTER 3

Software Programming

Library functions

Quick reference table

The following table lists the available functions and their corresponding syntax and descriptions.

Library functions

Function	Syntax (in C)	Description
1	canInitHW()	Sets IRQs
2	canExitHW()	Releases settings
3	canReset()	Resets CAN port
4	canConfig()	Controls CAN port settings
5	canNormalRun()	Sets mode
6	canSendMsg()	Sends message
7	canReceiveMsg()	Reads data

Complete function description

Function 1
Sets an IRQ number for Port1 and Port 2.

å **Command** canInitHW (UI segment, BYTE IRQ1, BYTE IRQ2)

å **Argument** UI segment, BYTE IRQ1, BYTE IRQ2
segment=c000-df00 step 0x100
IRQ1=Port 1 IRQ number 0 (polling),
3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15
IRQ2=Port 2 IRQ number 0 (polling),
3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15
0: polling

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   UI gSegment=0xDA00;
   BYTE CAN1_IRQ, CAN2_IRQ;
   CAN1_IRQ=12;
   CAN2_IRQ=15;
   if (canInitHw (gSegment, CAN1_IRQ, CAN2_IRQ)==0)
      printf ("HARDWARE INITIALIZATION ERROR!\n");
}
```

Function 2

Releases all settings of the CAN card.

å **Command** canExitHW()

å **Argument** None

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   if (canExitHw()==0)
      printf ("CAN RELEASE FAIL!\n");
}
```

Function 3

Resets CAN port and flushes the TX/RX buffers.

å **Command** int canReset (BYTE port);

å **Argument** BYTE port;
port= port number (0 or 1)

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   if (canReset (0)==0)
      printf ("RESET PORT 1 FAIL!\n");
}
```

Function 4

Controls the setting of the CAN port's acceptance code, acceptance mask, and bus timing register.

å **Command** canConfig (BYTE port, CAN_STRUCT can);

å **Argument** BYTE port, CAN_STRUCT can;
port= port number (0 or 1)
can= CAN struct pointer

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   CAN_STRUCT can1, can2;
   can1.acc_code=0;
   can1.acc_mask=0xff;
   can1.bt0=0;
   can1.bt1=0x1c;
   if (canConfig(0,can1)==0)
      printf ("CAN PORT 1 CONFIGURE ERROR!\n");
}
```

Function 5

Sets a CAN port to normal mode for normal operation.

å **Command** canNormalRun (BYTE port);

å **Argument** BYTE port;
port= port number (0 or 1)

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   if (canNormalRun(0)==0)
      printf ("CAN Port 1 can't change to Normal Mode!\n");
}
```

Function 6

Tells the CAN port to send a message.

å **Command** canSendMsg (BYTE port, MSG_STRUCT send_msg);

å **Argument** BYTE port, MSG_STRUCT send_msg;
port= port number (0 or 1)
send_msg= send buffer pointer

å **Response** 1=successful
0=fail

å **Example**

```
#include "can841.h"
main()
{
   MSG_STRUCT smsg1;
   UI i;
   smsg1.id=0x015;
   smsg1.rtr=0;
   smsg1.dlen=8;
   for(i=0; i<smsg1.dlen; i++)
         smsg1.data[i]=i;
   if (canSendMsg(0,smsg1)==1)
         printf ("TRANSMISSION SUCCESSFUL!\n");
}
```

Function 7

Read data from CAN port input buffer.

å **Command** int canReceiveMsg (BYTE port, MSG_STRUCT *msg_ptr);

å **Argument** BYTE port, MSG_STRUCT *msg_ptr;
port= port number (0 or 1)
*msg_ptr= input buffer pointer

å **Response** 1=message received
0=no message received

å **Example**

```
#include "can841.h"
main()
{
   MSG_STRUCT rmsg2;
   if (canReceiveMSG, *rmsg2)==1)
   {
      printf ("Port2 receive: ID=%3X RTR=%ld
         Length=%ld", rmsg2.id, rmsg2.rtr, rmsg2.dlen);
      for (i=0; i<rmsg2.dlen; i++)
         cprintf ("%2X", rmsg2.data[i]);
   }
}
```

Example program

The following example program, can841.lib, implements the sending and receiving of messages over the CAN controller. The program is written in C.

```
#include "can841.h" /*Library function declaration*/
/*-----------------------------------------------*/

/* CAN controller interrupt connection */
#define  CAN1_IRQ 12 /* 0 means polling */
#define  CAN2_IRQ 15 /* 0 means polling */
#define  PORT1    0
#define  PORT2    1
#define  FAIL     0
#define  SUCCESS  1

void main(void)
{
   /* Declare the CAN card segment address. */
   UI gSegment=0xDA00;
   CAN_STRUCT can1, can2;
   MSG_STRUCT smsg1, smsg2;
   MSG_STRUCT rmsg1, rmsg2;
   UI i;

   if(canInitHW(gSegment,CAN1_IRQ,CAN2_IRQ)==FAIL)
      {
         clrscr();
         cprintf("\n\n Hardware Initialization Error");
         return;
      }

      /* Reset CAN controller */
      canReset(PORT1);
      canReset(PORT2);

      can1.acc_code=0;          /*                    */
      can1.acc_mask=0xff;       /*                    */
      can1.bt0=03;              /* baud rate 1Mbps    */
      can1.bt1=0x1c;
      if(canConfig(PORT1,can1)==FAIL)
      {
         clrscr();
         cprintf("\n\n CAN Port %d Configuration Error",1);
         return;
      }
```

```
   memcpy(&can2, &can1, sizeof(CAN_STRUCT));
   if(canConfig(PORT2,can2)==FAIL)
   {
      clrscr();
      cprintf("\n\n CAN Port %d Configuration Error", 2);
      return;
   }
   canNormalRun(PORT1); /* Put CAN1 into normal mode. */
   canNormalRun(PORT2); /* Put CAN2 into normal mode. */

   clrscr();

   smsg1.id = 0x015; /* Set ID =8       */
   smsg1.rtr=1;      /* Data lengths =8 */
   smsg1.dlen=8;
   for(i=0; i<smsg1.dlen; i++)
   smsg1.data[i] =i;
   while(1)
   {
      canSendMsg(PORT1, smsg1); /* Send to CAN1 */
      if(canReceiveMsg(PORT2, &rmsg2)==1)
      {
         cprintf("PORT2 receive:ID=%3X RTR=%1d Length=%1d",
            rmsg2.id,rmsg2.rtr, rmsg2.dlen);
         for(i=0; i< rmsg2.dlen; i++)
            cprintf(" %2X",rmsg2.data[i]);
         printf("\n");
      }
      if (kbhit())
      {
         getch();
         break;
      }
   }

   /* Reset CAN controller. */
   canReset(PORT1);
   canReset(PORT2);
   canExitHW();
   clrscr();
}
```