ภาคผนวก

ภาคผนวก ก

โปรแกรมบนเครื่องเอเอส/400

```
#include <stdio.h>              /* Standard I/O header          */
#include <stdlib.h>             /* General utilities            */
#include <string.h>             /* String handling utilities    */
#include <stddef.h>             /* Standard definition          */
#include <xxfdbk.h>             /* Feedback area structures     */
#include <recio.h>              /* record i/o routines          */
#include <cmc.h>                /* CPI-Communications pseudonyms */
FILE    *product_file;
struct Product {
      char product_code??(5??);
      char product_name??(30??);
      char price??(5??);
      };
struct Product  prod;
/*********************************************************************/
/* Define variables used with CPI-Communications calls.          */
/*********************************************************************/
CM_INT32 data_received;
CM_INT32 prep_to_receive_type;
CM_INT32 requested_length;
CM_INT32 received_length;
CM_INT32 request_to_send_received;
CM_INT32 return_code;
CM_INT32 send_length;
CM_INT32 send_type;
```

```
CM_INT32 status_received;
CM_INT32 deallocate_type;
CM_INT32 received_type;
char    conversation_ID??(8??);
char    buffer??(5??);
/*********************************************************************/
/* Define constants/flags used in program.                           */
/*********************************************************************/
#define ERROR 1                 /* error during I/O processing       */
#define NOERROR 0
/*********************************************************************/
/* Declare global variables/functions.                               */
/*********************************************************************/
void cleanup(void);
size_t          size;           /* "size_t" is a synonym for the     */
                                /* type of the value returned by     */
                                /* the "sizeof" operator.            */

char yes_no='Y';

/*********************************************************************/
/* START OF PROGRAM                                                  */
/*********************************************************************/
main()
{
/*****************************************/
/*          Open product file            */
/*****************************************/
 if (( product_file=fopen("chai/product(product)",
       "rb+,lrecl=40"))==NULL)
```

```
    {
    printf("Open file fail product");
    exit(1);
    }
/*********************************************************************/
/* Accept conversation from remote                                   */
/*********************************************************************/
 CMACCP(conversation_ID, &return_code);
 if (return_code != CM_OK)    {
   printf("Accept error code is %d \n",return_code);
   cleanup();
    }
 else
    printf("Accept ready\n");
/*********************************************************************/
/* The Set_Send_Type (CMSST)                                         */
/*********************************************************************/
 send_type = CM_SEND_AND_FLUSH;
 CMSST(conversation_ID, &send_type, &return_code);

/*********************************************************************/
/* The Set prepare to received Type (CMSPTR)                         */
/*********************************************************************/
 prep_to_receive_type = CM_PREP_TO_RECEIVE_FLUSH;
 CMSPTR(conversation_ID, &prep_to_receive_type, &return_code);
/*********************************************************************/
/* The SetReceived Type (CMSPTR)                                     */
/*********************************************************************/
 received_type =CM_RECEIVE_AND_WAIT;
 CMSRT (conversation_ID,&received_type,&return_code);
```

```
/*********************************************************************/
/* set length to recieved and received record                        */
/*********************************************************************/
for (;;){
requested_length=5;
/************************************/
/*     Receive Product code from remote    */
/************************************/
CMRCV(conversation_ID, buffer, &requested_length,
      &data_received, &received_length, &status_received,
      &request_to_send_received, &return_code);
if (return_code == CM_OK)    {
  printf ("DATA is %s \n",buffer);  }
else  {
  printf ("error on received is %d \n",return_code);
  cleanup();
  }
/************************************/
/*     search record for product code          */
/************************************/
fseek (product_file,0,SEEK_SET);
yes_no='Y';
while (yes_no=='Y') {
  memset (&prod,0,sizeof(prod));
  fread(&prod,1,sizeof(prod),product_file);
  if (strncmp(buffer,prod.product_code,4 )==0)
    { printf("Compare ok\n");
      printf ("Buffer is %s \n",buffer);
      printf ("Product code is %s \n",prod.product_code);
      printf ("Product name is %s \n",prod.product_name);
```

```
      yes_no='N';}
    else {
      printf ("Cannot compare\n"); }
    if (feof(product_file)) {
      memset (&prod,'9',sizeof(prod));
      printf ("end of fiile");
      yes_no='N';}
      fseek(product_file,0,SEEK_CUR);
  }
/*********************************************/
/*  receive request fot change state from receive to send */
/*********************************************/
  CMRCV(conversation_ID, buffer, &requested_length,
        &data_received, &received_length, &status_received,
        &request_to_send_received, &return_code);
 if (return_code == CM_OK)  {
   printf ("status received is %d \n",status_received);
   printf ("requesed is %d \n",request_to_send_received);
   }
 else   {
   printf ("error on received is %d \n",return_code);
   }
   send_length =strlen(prod.product_name);
/*************************************/
/*   send product description and price     */
/*************************************/
   CMSEND(conversation_ID, prod.product_name, &send_length,
          &request_to_send_received, &return_code);
    if (return_code != CM_OK)  {
      printf ("error on send to remote is %d \n",return_code);
```

```
    }
   else
     printf ("Send data is ready send %s length %d \n",
            buffer,send_length);
/*********************************************/
/*    send request to change state from send to receive  */
/*********************************************/
  CMPTR(conversation_ID,&return_code);
   if (return_code != CM_OK)  {
     printf ("error on send to remote is %d \n",return_code);
    }
   else
     printf ("Send data is ready send %s length %d \n",
            buffer,send_length);
 }
 cleanup();
}
/*********************************************************************/
/*                                                                   */
/*   ****************                ****************   */
/*   *                INTERNAL FUNCTIONS                 *   */
/*   ****************                ****************   */
/*                                                                   */
/*********************************************************************/



/*********************************************************************/
/* "CLEANUP" function.                                               */
/*                                                                   */
```

```
/* The following code handles the end-of-program processing.        */
/* This includes the ending of the conversation with                */
/* the remote system (if conversation is active), and the           */
/* closing of opened files.                                         */
/*******************************************************************/
void cleanup()
{
 if ((return_code != CM_ALLOCATE_FAILURE_RETRY) &&
     (return_code != CM_ALLOCATE_FAILURE_NO_RETRY) &&
     (return_code != CM_DEALLOCATED_ABEND) &&
     (return_code != CM_DEALLOCATED_NORMAL) &&
     (return_code != CM_PRODUCT_SPECIFIC_ERROR) &&
     (return_code != CM_RESOURCE_FAILURE_RETRY) &&
     (return_code != CM_RESOURCE_FAILURE_NO_RETRY))
   {
    deallocate_type = CM_DEALLOCATE_ABEND;
    CMSDT(conversation_ID, &deallocate_type, &return_code);
    CMDEAL(conversation_ID, &return_code);
   }
 exit(0);
}                              /* end cleanup...          */
```

ภาคผนวก ข

โปรแกรมที่ใช้บนเครื่องไมโครคอมพิวเตอร์

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#include <stddef.h>
#include "c:\nsd\include\cmc.h"    /* Include file for CPIC */
#include <graphics.h>
#include <stdarg.h>
#include <math.h>
#include <ctype.h>
#define MaxRow 5
#define MaxCode 5
#define MaxDigit 4
char Quantity[MaxRow][MaxDigit];
char ProdCode[MaxRow][MaxCode];
double ProdTotal[MaxRow];
double Total;
char str[10];
#define DEFUALT_TP_NAME          "CPICT"
#define DEFUALT_MODE_NAME        "BLANK"
#define DEFUALT_REMOTE_SYSTEM    "DCS400E2"
#define DEFUALT_PARTNER          "S7801131"
/* ASCII to EBCDIC translate table  */

char ascii_to_ebcdic_table[] = {
"\x00\x01\x02\x03\x37\x2D\x2E\x2F\x16\x05\x15\x0B\x0C\x0D\x0E\x0F"  /* 00-0F */
```

```
"\x10\x11\x12\x13\x3C\x3D\x32\x26\x18\x19\x3F\x27\x22\x1D\x35\x1F"  /* 10-1F */
"\x40\x5A\x7F\x7B\x5B\x6C\x50\x7D\x4D\x5D\x5C\x4E\x6B\x60\x4B\x61"  /* 20-2F */
"\xF0\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\x7A\x5E\x4C\x7E\x6E\x6F"  /* 30-3F */
"\x7C\xC1\xC2\xC3\xC4\xC5\xC6\xC7\xC8\xC9\xD1\xD2\xD3\xD4\xD5\xD6"  /* 40-4F */
"\xD7\xD8\xD9\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xAD\xE0\xBD\x5F\x6D"  /* 50-5F */
"\x79\x81\x82\x83\x84\x85\x86\x87\x88\x89\x91\x92\x93\x94\x95\x96"  /* 60-6F */
"\x97\x98\x99\xA2\xA3\xA4\xA5\xA6\xA7\xA8\xA9\xC0\x4F\xD0\xA1\x07"  /* 70-7F */
"\x43\x20\x21\x1C\x23\xEB\x24\x9B\x71\x28\x38\x49\x90\xBA\xEC\xDF"  /* 80-8F */
"\x45\x29\x2A\x9D\x72\x2B\x8A\x9A\x67\x56\x64\x4A\x53\x68\x59\x46"  /* 90-9F */
"\xEA\xDA\x2C\xDE\x8B\x55\x41\xFE\x58\x51\x52\x48\x69\xDB\x8E\x8D"  /* A0-AF */
"\x73\x74\x75\xFA\x15\xB0\xB1\xB3\xB4\xB5\x6A\xB7\xB8\xB9\xCC\xBC"  /* B0-BF */
"\xAB\x3E\x3B\x0A\xBF\x8F\x3A\x14\xA0\x17\xCB\xCA\x1A\x1B\x9C\x04"  /* C0-CF */
"\x34\xEF\x1E\x06\x08\x09\x77\x70\xBE\xBB\xAC\x54\x63\x65\x66\x62"  /* D0-DF */
"\x30\x42\x47\x57\xEE\x33\xB6\xE1\xCD\xED\x36\x44\xCE\xCF\x31\xAA"  /* E0-EF */
"\xFC\x9E\xAE\x8C\xDD\xDC\x39\xFB\x80\xAF\xFD\x78\x76\xB2\x9F\xFF"  /* F0-FF */
};

/* EBCDIC to ASCII translate table  */

unsigned char ebcdic_to_ascii_table[] = {
"\x00\x01\x02\x03\xCF\x09\xD3\x7F\xD4\xD5\xC3\x0B\x0C\x0D\x0E\x0F"  /* 00-0F */
"\x10\x11\x12\x13\xC7\x0A\x08\xC9\x18\x19\xCC\xCD\x83\x1D\xD2\x1F"  /* 10-1F */
"\x81\x82\x1C\x84\x86\x0A\x17\x1B\x89\x91\x92\x95\xA2\x05\x06\x07"  /* 20-2F */
"\xE0\xEE\x16\xE5\xD0\x1E\xEA\x04\x8A\xF6\xC6\xC2\x14\x15\xC1\x1A"  /* 30-3F */
"\x20\xA6\xE1\x80\xEB\x90\x9F\xE2\xAB\x8B\x9B\x2E\x3C\x28\x2B\x7C"  /* 40-4F */
"\x26\xA9\xAA\x9C\xDB\xA5\x99\xE3\xA8\x9E\x21\x24\x2A\x29\x3B\x5E"  /* 50-5F */
"\x2D\x2F\xDF\xDC\x9A\xDD\xDE\x98\x9D\xAC\xBA\x2C\x25\x5F\x3E\x3F"  /* 60-6F */
"\xD7\x88\x94\xB0\xB1\xB2\xFC\xD6\xFB\x60\x3A\x23\x40\x27\x3D\x22"  /* 70-7F */
"\xF8\x61\x62\x63\x64\x65\x66\x67\x68\x69\x96\xA4\xF3\xAF\xAE\xC5"  /* 80-8F */
"\x8C\x6A\x6B\x6C\x6D\x6E\x6F\x70\x71\x72\x97\x87\xCE\x93\xF1\xFE"  /* 90-9F */
```

```
"\xC8\x7E\x73\x74\x75\x76\x77\x78\x79\x7A\xEF\xC0\xDA\x5B\xF2\xF9"  /* A0-AF */
"\xB5\xB6\xFD\xB7\xB8\xB9\xE6\xBB\xBC\xBD\x8D\xD9\xBF\x5D\xD8\xC4"  /* B0-BF */
"\x7B\x41\x42\x43\x44\x45\x46\x47\x48\x49\xCB\xCA\xBE\xE8\xEC\xED"  /* C0-CF */
"\x7D\x4A\x4B\x4C\x4D\x4E\x4F\x50\x51\x52\xA1\xAD\xF5\xF4\xA3\x8F"  /* D0-DF */
"\x5C\xE7\x53\x54\x55\x56\x57\x58\x59\x5A\xA0\x85\x8E\xE9\xE4\xD1"  /* E0-EF */
"\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\xB3\xF7\xF0\xFA\xA7\xFF"  /* F0-FF */
};
struct Product {
      char product_code [5];
      char product_name [30];
      char price[5];
      };
struct Product prod,prod1;
void ascii_to_ebcdic (char *,CM_INT32 );
void ebcdic_to_ascii (unsigned char *,CM_INT32);
char Conversation_id[8];
CM_INT32 received_type;
CM_INT32 Requested_Length;
CM_INT32 Received_Length;
CM_INT32 prep_to_received_type;
CM_INT32 data_received;
CM_INT32 status_received;
CM_INT32 Return_Code;
CM_INT32 send_type;
CM_INT32 Send_Length;
CM_INT32 requested_to_send_received;
unsigned char Received_Buffer[35];
main() {

/*********************************/
```

```
/* Set graphic mode and display screen  */
/************************************/
   int GraphDriver,GraphMode;
   int i,j,ErrorCode;
   int x,y;
   unsigned char Received_Buffer[30];
   GraphDriver=0;
   GraphMode=0;
   initgraph(&GraphDriver,&GraphMode,"");
   setcolor(5);
   setbkcolor(8);
   setusercharsize(4,3,4,3);
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,USER_CHAR_SIZE);
   moveto(175,20);
   outtext("ORDER ENTRY");
   setlinestyle(SOLID_LINE,0,THICK_WIDTH);
   line (175,65,450,65);
   setusercharsize(1,2,1,2);
   send_type=CM_SEND_AND_FLUSH;
/**************************/
/* Initialize to reserve session  */
/**************************/
   cminit(Conversation_id,DEFUALT_REMOTE_SYSTEM,&Return_Code);
   if (Return_Code != CM_OK ) {
     printf("Init Error Code is %d \n",Return_Code);
     exit(0);
     }
   else  {
     /* printf ("Init Complete with error code %d\n ",Return_Code); */
   }
```

```
    cmsst(Conversation_id,&send_type,&Return_Code);
     if (Return_Code != CM_OK) {
       printf ("Set Send Type Error is %d \n",Return_Code);
       exit(0);
       }
     else
       /* printf ("Set Send Type OK.....\n"); */
/***************************/
/* allocate remote program      */
/***************************/
    cmallc(Conversation_id,&Return_Code);
     if (Return_Code != CM_OK) {
          printf ("Allocate Error Code is %d \n",Return_Code);
         exit(0);
         }
     else {
         /* printf("Allocate Ready Code is %d  \n",Return_Code); */
         }
    memset (ProdCode,'\0',sizeof(ProdCode));
    memset (Quantity,'\0',sizeof(Quantity));
    memset (str,'\0',sizeof(str));
    OrderTable();     /*   Make table on screen   */
    InputOrder();     /*   Input Order    */
    /*************/
    /* Print Total  */
    /*************/
     for (i=0;i<MaxRow;i++)
       Total=ProdTotal[i]+Total;
     sprintf (str,"%7.2f",Total);
       moveto (550,400);
```

```
        outtext(str);
    getch();
/*******************/
/*  deallocate session */
/*******************/
    cmdeal(Conversation_id,&Return_Code);
    closegraph();
}

void   ascii_to_ebcdic  (char *ascii_string,CM_INT32 length1)
{
    int i;
    for (i = 0;i<length1;i++)
          ascii_string[i] = ascii_to_ebcdic_table[ascii_string[i]];
}

void   ebcdic_to_ascii  (unsigned char *ebcdic_string,CM_INT32 length1)
{
    int i;
    for (i = 0;i<length1;i++)
          ebcdic_string[i] = ebcdic_to_ascii_table[ebcdic_string[i]];
}

OrderTable() {
int x,y;
setcolor(3);
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
```

```
line(0,100,0,460);
line(630,100,630,460);
for (y=100;y<=460;y+=20) {
   line (0,y,630,y); }
line (60,100,60,460);
line (330,100,330,460);
line (430,100,430,460);
line (530,100,530,460);

}

InputOrder()
{
int i,j,x,y;
char buffer1[30];
char buffer2[5];
char c;
setcolor(6);
x=20;
y=102;
/**********************/
/*   Input Product Code   */
/**********************/
for (j=0;j<MaxRow;j++) {
   for(i=0;i<MaxCode-1;i++) {
      ProdCode[j][i]= getch();
      moveto (x,y);
      outtext(ProdCode[j]);
      }
    memset (&prod,'\0',sizeof(prod));
```

```
memset (&prod1,'\0',sizeof(prod1));
strcpy (prod.product_code,ProdCode[j]);
Send_Length=strlen(prod.product_code);
ascii_to_ebcdic(prod.product_code,Send_Length);
 /*******************************/
 /*     Send product code to remote    */
 /*******************************/
cmsend(Conversation_id,prod.product_code,&Send_Length,
        &requested_to_send_received,&Return_Code);
if (Return_Code != CM_OK) {
      printf ("Send error Code is %d \n",Return_Code);
  }
else {
      /* printf("Send data %s is ready \n",prod.product_name) ; */
  }
prep_to_received_type=CM_PREP_TO_RECEIVE_SYNC_LEVEL;
cmsptr(Conversation_id,&prep_to_received_type,&Return_Code);
received_type=CM_RECEIVE_AND_WAIT;
cmsrt(Conversation_id,&received_type,&Return_Code);
      if (Return_Code != CM_OK) {
      /*  printf ("error on SET received type"); */
        }
/*********************************************/
/* Send request to change state from send to receive    */
/*********************************************/
cmptr(Conversation_id,&Return_Code);
      if (Return_Code != CM_OK) {
        printf ("Return Status Error is %d \n",Return_Code);
        }
      else
```

```
        /* printf ("Return status is OK.\n"); */
 Requested_Length=35;
 memset (Received_Buffer,'0',35);
 /*****************************/
 /*  receive product description      */
 /*****************************/
 cmrcv(Conversation_id,Received_Buffer,&Requested_Length,
        &data_received,&Received_Length,&status_received,
        &requested_to_send_received,&Return_Code);
      if (Return_Code == CM_OK){
        /* printf ("received data is %s length %d \n",
                    Received_Buffer,Received_Length);  */   }
      else
         printf("error on received is %d \n",Return_Code);
ebcdic_to_ascii (Received_Buffer,Received_Length);
strncpy(prod.product_name,Received_Buffer,35);
memset (buffer1,'\0',30);
memset (buffer2,'\0',5);
strncpy(buffer1,prod.product_name,29);
strncpy(buffer2,prod.price,4);
if (strncmp(prod.product_name,"9999",4)==0) {
  setcolor(8);
  moveto(x,y);
  outtext(ProdCode[j]);
  for (i=0;i<MaxCode-1;i++){
      ProdCode[j][i]=' '; }
  setcolor(6);
  j--;
  }
else {
```

```
moveto(80,y);
outtext(buffer1);
moveto(350,y);
outtext(buffer2);
x=500;
/*****************/
/*  Input quantity   */
/*****************/
for(i=0;i<MaxDigit-1;i++){
     c=getch();
     if (c!=0x0d) {
      if (isdigit(c) ){
        Quantity[j][i]= c;
        moveto (x,y);
        outtext(Quantity[j]);
        }
      else {
        i--; }
     }
 }
 ProdTotal[j]=atof(prod.price)*atof(Quantity[j]);
 sprintf (str,"%7.2f",ProdTotal[j]);
 moveto (550,y);
 outtext(str);
x=20;
y+=20;}
/*****************************************/
/*  Receive request to change form receive to send */
/*****************************************/
cmrcv(Conversation_id,Received_Buffer,&Requested_Length,
```

```
        &data_received,&Received_Length,&status_received,
        &requested_to_send_received,&Return_Code);
      if (Return_Code == CM_OK){
        /* printf ("received data is %s length %d \n",
                    Received_Buffer,Received_Length);  */   }
      else
        printf("error on received is %d \n",Return_Code);
}
}
```

รายการอ้างอิง

Dymak,Walt. "IBM System Network Architecture (SNA) ," Datapro,8230,1992

Altman,Ross S. and Rubin,Jonathan L. "An Overview of Cooperative Processing in an IBM PC and IBM Mainframe Enviroment ,"Datapro ,1276,1992

Berson,Alex . APPC:introduction to LU 6.2 , Mcgraw-Hill Inc., 1990

Kapoor,Atul .SNA Architecute,Protocol and Implementation , McGraw-Hill Inc., 1992

Seery, Mark.QED Publishing Group,1993

_____,System Network Architecture : concept and product , IBM,1986

_____,System Network Architecture : technical Overview , IBM,1986

_____,Transaction Programmer's reference manual for LU Type 6.2 , IBM.1990

_____,AS/400 Communication : Advanced Program to Program Commincation Programmer guide version 2 , IBM 1992

_____,AS/400 Communication : Advanced Peer to Peer Networking guide version 2 , IBM,1992

_____,PC support /400 Application Program Interface , IBM , 1991

_____, Networking Service/DOS , IBM ,1992

_____, CPIC Communications Reference , IBM , 1991

ประวัติผู้เขียน

นายชาญชัย จิตลดาพร เกิดวันที่ 30 มิถุนายน 2504 สำเร็จการศึกษาปริญญาตรี วิศวกรรมศาสตร์บัณฑิต สาขาอิเลคโทรนิค คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

เข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ. 2535