PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Zespół Urządzeń i Systemów Sterowania

440 BE10

Główny wykonawca dr inż. Andrzej Syryczyński

Wykonawcy dr inż. Andrzej Syryczyński
dr inż. Wiesław Stańczak

Konsultant

Nr zlecenia S1393

Opracowanie sieciowego oprogramowania firmowego MINI MAP

Etap 1. Adaptacja i dołączenie oprogramowania MINI MAP

Zleceniodawca

Pracę rozpoczęto dnia 01.07.93 **zakończono dnia** 22.12.93

Kierownik Zespołu
dr inż. A. Syryczyński

Z-ca Dyrektora d/s Badawczo-Rozwojowych
dr inż. J. Jabłkowski

Praca zawiera:

stron 10
rysunków
fotografii
tabel
tablic
załączników 2

Nr rejestr. 7028

Rozdzielnik - ilość egz: 3

Egz. 1 BOINTE
Egz. 2 ZSS
Egz. 3 ZSS
Egz. 4
Egz. 5
Egz. 6

Analiza deskryptorowa

URZADZENIA AUTOMATYCZNEJ REGULACJI I STEROWANIA: SIEC LOKALNA + MAP +MMS + OPROGRAMOWANIE

Analiza dokumentacyjna

W oparciu o obiekty architektury MINI MAP wyróżnione uprzednio w oprogramowaniu sieciowym firmy AEG/MODICON opracowano przykładowy program realizujący wybrane usługi MMS w architekturze MINI MAP.

Tytuły poprzednich sprawozdań

1. Badania i promocja sieci MAP. Etap 4. Sprzężenie komputera standardu IBM z siecią MAP stałego stanowiska. Nr rej.6910
2. Implementacja oprogramowania MAP 3.0 i MMS. Etap 1. Rozpoznanie oprogramowania firmy MODICON w nawiązaniu do zbioru usług przewidzianych w MMS i Specyfikacji MAP 3.0. Identyfikacja narzędzi języka C użytych do realizacji podstawowych usług. Nr rejestr. 6967.
3. Implementacja oprogramowania MAP 3.0 i MMS. Etap 2.Analiza działania oprogramowania w sieci w tym z użyciem stacji MINI MAP oraz ewentualne adaptacje. Nr rejestr.6997.

UKD

PIAP 41/88 10000

Spis treści

Załączniki:

1. Wprowadzenie

Jednolity otwarty system komunikacji OSI (Open System Interconnection), którego realizację stanowi MAP / TOP, definiuje siedem poziomów zwanych warstwami, a mianowicie:

- poziom 1 zwany warstwą 1 - fizyczną (physical layer);
- poziom 2 zwany warstwą 2 - łącza danych (data link layer);
- poziom 3 zwany warstwą 3 - sieci (network layer);
- poziom 4 zwany warstwą 4 - transportową (transport layer);
- poziom 5 zwany warstwą 5 - sesji (session layer);
- poziom 6 zwany warstwą 6 - prezentacji (presentation layer);
- poziom 7 zwany warstwą 7 - zastosowań (application layer).

Architektura Mini-MAP obejmuje warstwy 1, 2 oraz 7.

Celem pracy było wyodrębnienie z uprzednio zakupionego oprogramowania firmy AEG/Modicon/Computrol części realizujących architekturę Mini - MAP, posadowienie uzyskanego w ten sposób programu w stacjach sieci MAP i przeprowadzenie próbnych seansów współpracy.

2. Charakterystyka oprogramowania AEG/Modicon/Computrol

Pakiet oprogramowania firmy AEG/Modicon/Computrol składa się z 2 części:

- -oprogramowania MAP 3.0 (Computrol's DOS ISOcomm MAP 3.0 802.4/802.3 Release 2.3 (Feb. 12, 1992));
- -oprogramowania MMS (Computrol's MMS MAP 3.0 ACSE Model Release 2.3 on MS-DOS (Feb. 1, 1992)).

4

Oprogramowanie MAP 3.0 realizuje 6 niższych warstw wg. ISO/OSI oraz dostarcza interfejs do MMS lub FTAM, a także drugi interfejs - do wewnętrznego oprogramowania sterownika sieci skonstruowanego np. jako karta do komputera typu IBM - PC. To ostatnie oprogramowanie jest dla użytkownika niedostępne, gdyż posadowione zostało w pamięci typu EPROM karty. W skład omawianego oprogramowania MAP 3.0 wchodzą między innymi biblioteki, pliki nagłówkowe, pliki konfiguracyjne ICPDOSDR.EXE i PC8024.IMG oraz program MAPGO.EXE (korzystający z PC8024.IMG), który powoduje wejście stacji do logicznego pierścienia obiegu uprawnienia w sieci Token Bus (MAP 3.0 wg. IEEE 802.4).

Oprogramowanie MMS realizuje pełny zbiór usług MMS lub, w razie potrzeby, wybrany jego podzbiór. Jest ono dostarczane w formie bibliotek, programów demonstracyjnych, przykładowych plików źródłowych oraz plików nagłówkowych ułatwiających współpracę z bibliotekami. Współpraca z oprogramowaniem MAP 3.0 odbywa się poprzez bibliotekę CI.LIB zawartą w pakiecie Computrol's DOS ISOcomm MAP 3.0 802.4/802.3 Release 2.3 (Feb. 12, 1992). MMSACSE.EXE stanowi jeden ze wspomnianych programów demonstracyjnych. Realizuje on wybrany podzbiór usług MMS (w tym Status, GetNameList, Identify, Read, Rename, Write, GetVariableAccessAttributes, DefineNamedVariable, DeleteVariableAccess, DefineNamedVaria-
-bleList, GetNamedVariableListAttributes, DeleteNamedVariab-
-leList, DefineNamedType, GetNamedTypeAttributes, DeleteNa-
-medType, ObtainFile, FileOpen, FileRead, FileClose, FileRe-
-name, FileDelete, FileDirectory, UnsolicitedStatus, Infor-

-mationReport, Conclude, Cancel). Ponadto MMSACSE.EXE umożliwia śledzenie poprawności wykonywania usług MMS poprzez wyświetlanie (na żądanie użytkownika) stanu ich wykonania.

Obydwa wymienione wyżej pakiety oprogramowania (tj. MAP 3.0 i MMS) przeznaczone są do wykorzystania w komputerach kompatybilnych z IBM - PC (wersja 80286 lub wyższa), pracujących w systemie operacyjnym DOS (co najmniej wersja 3.30).

3. Oprogramowanie Mini - MAP

Przy konstrukcji oprogramowania Mini - MAP wzorowano się na przykładowych plikach dostarczonych wraz z firmowymi pakietami oprogramowania. Zmodyfikowano plik nagłówkowy MMS_DEFS.H odblokowując (usuwając nawiasy komentarza) z definicji:

```
#define MAP30_LLC
```

Potem skonstruowano plik MMSAPP.C, którego tekst źródłowy zamieszczono w Załączniku 1. Plik ten skompilowano wykorzystując kompilator firmy Microsoft C ver. 6.00A z następującymi opcjami: /DDOS, /DASE, /c, /AL, /Gt4. Następnie zmodyfikowano bibliotekę MMSACSE.LIB (wykorzystując program bibliotekarza - LIB.EXE - ze wspomnianego przed chwilą pakietu firmy Microsoft) modyfikując moduł MMSACON.OBJ porzez dołączenie dodatkowej funkcji X_do_init, której tekst źródłowy zamieszczono w Załączniku 2.

Program wynikowy otrzymano przez konsolidację (zlinkowanie) pliku MMSAPP.OBJ z następującymi bibliotekami: LLIBCE.LIB (Microsoft C ver. 6.00A), ASN1.LIB (MMS), MMSACSE.LIB (zmodyfikowana zgodnie z ww. uwagami - MMS), MMS.LIB (MMS), SUICACSE.LIB (MMS), U_CORE.LIB (MMS), U_ACSE.LIB (MMS), CI.LIB (MAP 3.0) przy wykorzystaniu następujących opcji: /STACK:8192, /NOE, /SE:460.

Oprogramowanie Mini - MAP działa analogicznie jak program MMSACSE.EXE. Mianowicie realizuje ten sam podzbiór usług MMS a ponadto umożliwia śledzenie poprawności wykonywania usług MMS poprzez wyświetlanie (na żądanie użytkownika) stanu ich wykonania. Oprogramowanie Mini - MAP jest obsługiwane przez operatora w trybie menu z szeroko rozwiniętą diagnostyką błędów, co czyni zbędnym opracowanie instrukcji użytkowania. Oprogramowanie Mini - MAP jest przeznaczone do wykorzystania w komputerach kompatybilnych z IBM - PC (wersja 80286 lub wyższa), pracujących w systemie operacyjnym DOS (co najmniej wersja 3.30).

4. Weryfikacja oprogramowania Mini - MAP

Do badania stacji oprogramowania Mini - MAP wykorzystano zestaw składający się z następujących elementów fizycznych:

01. Komputer 486DX/50MHz, 256kB Cache, 16MB RAM,
    FDD 1.2MB + 1.44MB, obudowa midi tower,
    klaw.-101, karta I/O(2*RS 232 + 1*Centr.),
    HDD: 120MB/16ms + 170MB/15ms,

7

karta SVGA512k + monitor 14" SVGA mono, mysz mech. wyposażony w kartę AEG / Modicon / Computrol LP - 25 MAP Controller Board sprzężoną z modemem AEG / Modicon / Computrol BK - 4 CM Carrierband Modem.

02. Komputer 386DX/33MHz, 128kB Cache, 12MB RAM, FDD 1.2MB + 360kB obudowa big tower, klaw.-101, karta I/O(2*RS 232 + 1*Centr.), HDD 100MB/16ms,
karta Hercules + monitor 14" Hercules, mysz mech. wyposażony w kartę AEG / Modicon / Computrol LP - 25 MAP Controller Board sprzężoną z modemem AEG / Modicon / Computrol BK - 4 CM Carrierband Modem.

03. Komputer 486DX/50MHz, 256kB Cache, 16MB RAM, FDD 1.2MB + 1.44MB, obudowa midi tower, klaw.-101, karta I/O(2*RS 232 + 1*Centr.), HDD 120MB/16ms,
karta SVGA512k + monitor 14" SVGA color, mysz mech. wyposażony w kartę AEG / Modicon / Computrol LP - 25 MAP Controller Board sprzężoną z modemem AEG / Modicon / Computrol BK - 4 CM Carrierband Modem.

04. Osprzęt sieciowy MAP 3.0 carrierband 5 Mbit/s składający się z okablowania (kabel koncentryczny główny RG 11 - 2 odcinki po

ok. 4 m każdy, kable koncentryczne odgałęzień stacyjnych RG 6 - 5 odcinków po ok. 1 m), rozgałęźników oraz terminatorów dostarczonych przez firmę AEG / Modicon / Computrol.

Komputerom 01, 02 i 03 przypisano, odpowiednio, następujące bezwzględne adresy sieciowe (hex):
39 840F 454E45 00000001 0001 A026 01,
39 840F 454E45 00000001 0001 A024 01,
39 840F 454E45 00000001 0001 A022 01,
oraz, w kolejności, adresy podwarstwy MAC (Medium Access Control) (hex): 00010001A026, 00010001A024, 00010001A022. Komputery 01 i 03 pracowały w systemie operacyjnym MS DOS 6.0, zaś komputer 02 miał posadowiony MS DOS 5.0. Ponadto komputer 03 wykorzystywał w formie nakładki system MS Windows 3.1. Pliki konfiguracyjne komputerów (CONFIG.SYS i AUTOEXEC.BAT) oraz rozmieszczenie na dysku twardym plików ICPDOSDR.EXE, PC8024.IMG, MAPGO.EXE, UMAP_2.DIR, SUIC.CFG i programu realizującego Mini - MAP były analogiczne do zawartości plików konfiguracyjnych oraz rozmieszczenia na dysku twardym plików ICPDOSDR.EXE, PC8024.IMG, MAPGO.EXE, UMAP_2.DIR, SUIC.CFG i programu MMSACSE.EXE opisanych w sprawozdaniu nr rej. 7027 "Opracowanie sieciowego oprogramowania firmowego MINI MAP. Dokończenie badań stacji obiektowej REFLEX". Konfiguracja kart AEG / Modicon / Computrol LP - 25 MAP Controller Board we wszystkich komputerach była taka sama (różniła się jedynie adresem stacji). Celem przykładu przytoczy się

parametry konfiguracyjne (zapisane zgodnie z pozycjami menu konfiguracyjnego) karty w stacji o adresie 39840F454E450000000010001A02401:

I. Display/Modify Non-volatile RAM Parameters

A.	Station Address	0x00010001A024
B.	Group Address Mask	0xFFFFFFFFFFFF
C.	Individual Address Mask	0xFFFFFFFFFFFF
D.	Slot Time Value	0x0100
E.	Modem Selection Value	0x03
F.	MAC Transmission Priority	0x07
G.	No Stat Tracking (0=N,1=Y)	0x01
H.	In Ring Desired (0=N,1=Y)	0x01
I.	Prescalar Mode (0->6,1->3)	0x01
J.	High Priority THT	0x03FF
K.	TRT(4)	0x7FFF
L.	TRT(2)	0x7FFF
M.	TRT(0)	0x7FFF
N.	TRT Ring Maintenance	0x7FFF
O.	RM Initial Value	0x7FFF
P.	Max Inter Solicit Count	0x10
R.	PTP Register Value	0x9A55

A.	Network PDU Flags	0x0001
B.	Network Config Timer (Sec)	0x00FF
C.	Max NPDU Segment Size	0x1FE0
D.	NPDU Lifetime (500ms unts)	0x28
E.	Subnet Size of NSAP	0x02
F.	Network Link SAP Option	0xFE

G.	Network QOS Option Value	0x00
H.	Netwk Priority Option Val	0x00
I.	Netwk Padding Option Size	0x00
J.	Record of Rte Option Size	0x00
K.	Intermediate Bcast Addr	0x000000000000
L.	End System Bcast Addr	0x000000000000

A.	FDB Pool Size	0x004B
B.	BDB Pool Size	0x0096
C.	Reserved	0x0096
D.	Download Server MAC Addr	0x000000000000
E.	Broadband Modem Channel	0x00
F.	Broadband Modem Xmit Level	0x00
G.	Broadband Modem User Data	0x000000000000

II. Display/Modify MAC Layer Parameters

Current TBC transmit priority: 7

III. Display/Modify Transport Layer Parameters

1. Maximum TPDU size (in bytes)

   Current maximum TPDU size: 8192

2. Maximum Number of Connections

   Current maximum number of connections: 64

3. Display/Modify Minimum Credit

   Current minimum credit: 1

4. Display/Modify Maximum Credit

   Current maximum credit: 8

5. Maximum Number of Retries

   Current maximum number of retries: 4

11

6. Retry Timer for Data (in seconds)

Current retry timer (in seconds): 15

7. Retry Timer for Expedited Data (in seconds)

Current expedited data retry timer (in seconds): 15

8. Window Timer (in seconds)

Current window timer (in seconds): 12

9. Reference Wait Timer (in seconds)

Current reference wait timer (in seconds): 30

IV. Display/Modify System Parameters

1. Display BDB Count

Current BDB count: 123

2. Display/Modify User Limit

Current system user limit: 75

3. Display/Modify Management Limit

Current system management limit: 37

4. Display/Modify Network Limit

Current system network limit: 112

W komputerach O1 i O2 posadowiono oprogramowanie realizujące Mini - MAP, zaś komputer O3 wyposażono w pakiet firmy AEG / Modicon / Computrol o nazwie Computrol MAP Network Monitor Ver. 1.1, służący do obserwowania aktywności sieci i działający w środowisku MS Windows 3.1.

W próbnych seansach współpracy testowano wykonanie usług MMS przez obserwację aktywności sieci przy użyciu stacji monitorującej (komputer O3), a także śledząc komunikaty o poprawności wykonania poszczególnych usług dostarczane przez oprogramowanie realizujące Mini - MAP. Podczas tego typu seansów próbnych nie zauważono wadliwego działania

oprogramowania. Następnie skupiono uwagę na podzbiorze usług odnoszącym się do pracy z plikami (ObtainFile, FileOpen, FileRead, FileClose, FileRename, FileDelete) i przeprowadzono testy polegające na przenoszeniu i usuwaniu poszczególnych plików, a następnie na próbie uzyskania dostępu do uprzednio usuniętych plików. Ponownie otrzymano pozytywne wyniki, polegające na:

- braku możliwości dostępu do uprzednio usuniętych plików;
- identyczności przesłanego pliku z plikiem źródłowym;
- identyczności pliku o zmienionej nazwie z plikiem źródłowym.

Na podstawie uzyskanych rezultatów należy sądzić, że skonstruowane oprogramowanie Mini - MAP działa poprawnie w zakresie testowanych usług MMS.

Załącznik 1.

Plik MMSAPP.C

```
static  char ident[] = "mmsapp.c";
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/*MODULE NAME : mmsapp.c                                         */
/*PRODUCT(S)  : MMSEASE                                          */
/*                                                               */
/*MODULE DESCRIPTION :                                           */
/*This module contains the 'main' function, powerup code,        */
/*SUIC function, invoke functions, screen and function key       */
/*setup, etc.                                                    */
/*                                                               */
/*GLOBAL FUNCTIONS DEFINED IN THIS MODULE :                      */
/*VOID  cserve()                                                 */
/*VOID  main()                                                   */
/*VOID  set_main()                                               */
/*VOID  do_ctx_ops()                                             */
/*VOID  set_ctx()                                                */
/*VOID  do_vmd_ops()                                             */
/*VOID  set_vmd()                                                */
/*VOID  do_dom_ops()                                             */
/*VOID  set_dom()                                                */
/*VOID  do_prg_ops()                                             */
/*VOID  set_prg()                                                */
/*VOID  do_var_ops()                                             */
/*VOID  set_var()                                                */
/*VOID  do_var_defs()                                            */
/*VOID  set_var_defs()                                           */
/*VOID  do_more_ops()                                            */
/*VOID  set_more()                                               */
/*VOID  do_sem_ops()                                             */
/*VOID  set_sem()                                                */
/*VOID  do_ocs_ops()                                             */
/*VOID  set_ocs()                                                */
/*VOID  do_evn_ops()                                             */
/*VOID  set_evn()                                                */
/*VOID  do_jou_ops()                                             */
/*VOID  set_jou()                                                */
/*VOID  do_fil_ops()                                             */
/*VOID  set_file()                                               */
/*VOID  do_file_mgt()                                            */
/*VOID  set_file_mgt()                                           */
/*VOID  do_misc()                                                */
/*VOID  set_misc()                                               */
/*VOID  do_suic()                                                */
/*VOID  set_suic()                                               */
/*VOID  print_demo()                                             */
```

```
/*                                                                    */
/*MODIFICATION LOG :                                                  */
/*Date       Who    Rev          Comments                            */
/*--------   ---   ------   ------------------------------------------*/
/*12-15-93   WS      xx      Mini-MAP version                         */
/*10-21-91   AO      04      Modified msg size prompt to allow        */
/*                           user to enter a msg size of 100 to*/
/*                           8k bytes.                                */
/*06-03-91   KW,AO   03      fixed extern of s_max_chan               */
/*01-25-91   AO      02      Removed ifdef for mmsvar()               */
/*08-14-90   AO              Modified module header tag ID            */
/*02/20/90   AO     4.50     ifdef set_suic_param according to        */
/*                           version type i.e. ACSE, MAP21,           */
/*                           or LLC.                                  */
/*01/14/90   AO     4.50     Initialized menu_set_fun in case         */
/*                           of exception before being fully          */
/*                           initialized. u_mmsexcept_ind calls*/
/*                           menu_set_fun.                            */
/*11/08/89   MDE    4.50     Added CPU # input text                   */
/*                           Changes l_max_chan to l_num_chans        */
/*11/07/89   MDE    4.50     Added CPU # query                        */
/*                           Added LLC function to menu               */
/*11/06/89   MDE    4.50     Changed load_local_titles use            */
/*10/31/89   MDE    4.50     Added LLC calls                          */
/*12/22/88   BWJ    4.00     Fixed print_header for computrol         */
/*                           and menu selections for <CR> to          */
/*                           pop menues. Also commented out           */
/*                           comm_serve calls                         */
/*08/15/88           4.00     MMSEASE 4.0 release.                    */
/*                           See MODL40.DOC for history.              */
/*02/21/89   AO      4.00     Re-instated call to comm_serve.         */
/*06/19/89   ASM     4.01     Cleaned up menu displays for MSOFT*/
/*                           MSOFT uses function keys                 */
/*09/22/89   DP              Added pragma directive for PC-DOS        */
/*                           to disable stack checking in             */
/*                           event_service because it will be         */
/*                           called at interrupt level.               */
/*11-27-89   KAW             Replaced #pragma check_stack(off)        */
/*                           with #include cstk_off.h and             */
/*                           #pragma check_stack(on) with             */
/*                           cstk_on.h so the pragma directive        */
/*                           will not confuse other compilers         */
/*                           besides the Microsoft (MSOFT)            */
/*                           compiler                                 */
/*06-21-91   WCM             Included global definition for           */
/*                           s_msgsize. Set s_msgsize equal to        */
/*                           user supplied value in the range         */
```

```
/*                          (+100 <= mms_max_msgsize <=       */
/*                          +1000), if user inputs a value    */
/*                          else s_msgsize equals             */
/*                          DEFAULT_MAX_MSGSIZE               */
/************************************************************/

#include "glbtypes.h"
#include "sysincs.h"
#include <dos.h>
#include <process.h>

#include "mms_usr.h"   /*to access MMS fucntions, variables*/
#include "mmsop_en.h"
#include "fkeydefs.h"  /*Function key handling*/
#include "asn1defs.h"  /*to access ASN.1 variables*/
#include "mem_chk.h"
#include "userdefs.h"
#include "gvaldefs.h"
#include "scrndefs.h"

/*variables global to this module :*/
extern SHORT background_en;
extern SHORT s_msgsize;
extern USHORT m_file_blk_size;

static BYTE c = 'F';  /*MSDOS machines only use fun keys*/
static BYTE cret = ' ';

/************************************************************/
/************************************************************/
/*                        cserve                            */
/*Function to be executed while waiting for keyboard input. */
/*allows this system to process communications in the       */
/*background                                                */
/************************************************************/
SHORT cserve()
{
 USHORT sav1,sav2;

 if(background_en)
   {
    sav1 = asn1_debug_sel;
    sav2 = mms_debug_sel;

    asn1_debug_sel = 0;
    mms_debug_sel = 0;
    comm_serve();
```

```
    asn1_debug_sel = sav1;
    mms_debug_sel = sav2;
   }
 return(0);
}

/*****************************************************************/
/*                EVENT SERVICE FUNCTION                         */
/*****************************************************************/

SHORT event_count = 0;          /*outstanding events        */
SHORT use_events;
LONG total_events = 0;
LONG comm_serve_count = 0;

#include "cstk_off.h"

VOID event_service(count)
SHORT count;
{
  total_events += count;
  event_count += count;
}

#include "cstk_on.h"

VOID mms_comm_service()
{
 if(use_events)
   {
    if(event_count)
     {
      comm_serve_count++;
      comm_serve();             /*do all MMSEASE service */
      ms_event_disable();
      event_count--;
      ms_event_enable();
     }
   }
 else
   comm_serve();
}
```

```
/*****************************************************************/
/*****************************************************************/
/*                    main                                       */
/*This is the main function for the MMS sample application.      */
/*This is where the various initialization of the MAP            */
/*modules takes place, as well as any system specific setup.     */
/*****************************************************************/
DEFAULT main()
{
 VOID configure_channels();
 VOID set_suic_param();
 VOID do_debugset();
 VOID set_main();
 SHORT doit;
 BYTE str[10];
 register i;
 SHORT temp;
 SHORT ret;
 extern SHORT s_base_cpu_num;

/*First do any system specific initialization, for screen, etc.*/

 mmsvar();              /*Must call mmsvar before strt_MMS*/
                /*Sets environmnet variables to default values.*/

#ifndef NOSUIC
 CLEARSCR;
 if((use_events = ask("\n\n Use Event Mechanism(N) ? ",0)))
 u_mms_event_notify = event_service;
#endif

 printf(" Maximum MMS Message Size(%d) ? ",DEFAULT_MAX_MSGSIZE);
 intget(&mms_max_msgsize);  /*get initial user input*/
 while((mms_max_msgsize > DEFAULT_MAX_MSGSIZE) ||
        (mms_max_msgsize <= -1) || (mms_max_msgsize < 100))
  {
   printf("\n Message Size must be a Positive number between 100 and
%d\n",
                DEFAULT_MAX_MSGSIZE);
   printf(" Maximum MMS Message Size(%d) ? ",DEFAULT_MAX_MSGSIZE);
   intget(&mms_max_msgsize);
  }
 s_msgsize = mms_max_msgsize;
 if(mms_max_msgsize <= DEFAULT_MAX_MSGSIZE)
   {
     m_file_blk_size =(USHORT)mms_max_msgsize - 64;
   }
```

```
 menu_set_fun = NULL;  /*Interface not up yet*/

 set_user_data();      /*allocate large user data elements*/
 set_suic_param();     /*setup CASE configuration         */

/*Set up some MMS parameters.                               */
/*Select debug print levels for the asn1 and mms tools, then*/
/*start MMS                                                 */

 asn1_debug_sel = 0;          /*Select NO ASN.1 debug print */
 mms_debug_sel = MMSCONF_PRINT | MMSIND_PRINT;

 background_en = 0;                  /*for debug, turn it off*/
 if(ret = strt_MMS())
  {
   printf("\n MMS failed to start err: ");
   ms_perror(ret);            /*display a desc. of the error*/
  }

 configure_channels();

/*AO* 11-06-90 TPY.DIB for 3rd party support*/
#if    0                     /*change to 1 to load 3rd party*/
 ms_load_tpy_dib();
#endif

#if MMS_VA_EN
/*Now register local variables that are to be accessable over*/
/*the network. Before we do this, need to write standard     */
/*types to the VMD                                          */

 set_user_doms ();     /*make domains                    */
 set_user_pis  ();     /*make program invocations        */
 set_user_types();     /*write types                     */
 set_user_vars ();     /*write variables                 */
#endif

 servefun = cserve;    /*setup background service function*/
 set_main();           /*Write the main screen, funct ptrs*/
 doit = 1;
 while(doit)
  {
   while(check_key() != 1)
                      /*Execute function keys until F10 hit*/
          mms_comm_service();
```

```
    CLEARSCR;
    temp = background_en;
    background_en = 0;

    if(ask("\n\n\n Really want to exit ? ",0))
       doit = 0;
    else
       {
        if(ask("\n Modify Print Levels ? ",0))
                do_debugset();
        else
           {
            set_main();
            background_en = temp;
           }
       }
   }
 CLEARSCR;
 if(ret = end_MMS()) /*Shut Down MMS-EASE*/
   {
    printf("\n MMS failed to stop err: ");
    ms_perror(ret);         /*display a desc. of the error*/
   }

 return(0);
}

/*************************************************************/
/*                  set_suic_param                           */
/*Function to set up any SUIC specific parameters. These     */
/*parameters may differ depending on the OS and board used. */
/*************************************************************/
VOID set_suic_param()
 {
  /*declare set_suic_param_xxx extern according to version
                                                        type*/
#ifdef MAP21_CASE
  extern set_suic_param_case();
#endif
#ifdef MAP30_ACSE
  extern set_suic_param_acse();
#endif
#ifdef MAP30_LLC
  extern set_suic_param_llc();
#endif
  extern SHORT s_max_chan;
```

```
/*Select the number of channels to be used by MMS-EASE*/

  CLEARSCR;
  print_demo();
  printf("\n\n MMS DEMONSTRATION APPLICATION SETUP ");

  load_local_titles(GET_CHANNEL_CONFIG);

#ifdef MAP30_ACSE
#ifndef NOSUIC    /*if SUIC used(must be used in sample)*/

/*s_max_chan = MAX_SUIC_CHAN;    number of SUIC channels*/
/*s_msgsize = 2048;              SUIC max message size*/

  if(max_mms_chan < s_max_chan)
    s_max_chan = max_mms_chan;
  set_suic_param_acse();
#endif
#endif

#ifdef MAP30_LLC
#ifndef NOSUIC    /*if SUIC used(must be used in sample)*/
  if(max_mms_chan < l_num_chans)
      l_num_chans = max_mms_chan;
  set_suic_param_llc();
#endif
#endif
}

/*************************************************************/
/*                 configure_channels                        */
/*Function to initialize any channel specific information or*/
/*states. This must be executed after strt_MMS has been     */
/*called, so that the LLP has been initialized.             */
/*************************************************************/
VOID configure_channels()
{
 VOID load_local_titles();

                        /*load, register, activate AR names*/
 load_local_titles(GET_TITLE_CONFIG);

#ifdef NOSUIC           /*if NO SUIC used                   */
/*mms_chan_info[0].ctxt.chan_state = M_ACTIVE;
  mms_chan_info[1].ctxt.chan_state = M_ACTIVE;*/
#endif
}
```

```
/*************************************************************/
/*                  set_main                                 */
/*function to set up the main screen and function key table */
/*************************************************************/
VOID set_main()
{
 VOID do_ctx_ops();
 VOID do_vmd_ops();
 VOID do_prg_ops();
 VOID do_dom_ops();
 VOID do_var_ops();
 VOID do_more_ops();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - CONTEXT MANAGEMENT\n",c);
 printf("\t     %c2    - VMD SUPPORT\n",c);
 printf("\t     %c3    - DOMAIN MANAGEMENT OPERATIONS\n",c);
 printf("\t     %c4    - PROGRAM INVOCATION MANAGEMENT\n",c);
 printf("\t     %c5    - VARIABLE ACCESS\n",c);
 printf("\t     %c6    - MORE !!\n",c);
 printf("\t   %c<RET> - EXIT MMSEASE DEMO\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_ctx_ops;
 funct_2 = do_vmd_ops;
 funct_3 = do_dom_ops;
 funct 4 = do_prg_ops;
 funct_5 = do_var_ops;
 funct_6 = do_more_ops;

 menu_set_fun = set_main;       /*used to reset the menu*/
}

/*************************************************************/
/*                  do_more_ops                              */
/*This functions executes the second main menu operations.  */
/*************************************************************/
VOID do_more_ops()
{
 VOID set_more();

 set_more();
```

```
 while(check_key() != 1)
                        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*****************************************************************/
/*                    set_more                                   */
/*function to set up the second screen and function key table*/
/*****************************************************************/
VOID set_more()
{
 VOID do_sem_ops();
 VOID do_ocs_ops();
 VOID do_evn_ops();
 VOID do_jou_ops();
 VOID do_fil_ops();
 VOID do_misc();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - SEMAPHORE MANAGEMENT OPERATIONS\n",c);
 printf("\t     %c2    - OPERATOR COMMUNICATION OPERATIONS\n",c);
 printf("\t     %c3    - EVENT MANAGEMENT OPERATIONS\n",c);
 printf("\t     %c4    - JOURNAL MANAGEMENT OPERATIONS\n",c);
 printf("\t     %c5    - FILE OPERATIONS\n",c);
 printf("\t     %c6    - MISCELLANEOUS\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_sem_ops;
 funct_2 = do_ocs_ops;
 funct_3 = do_evn_ops;
 funct_4 = do_jou_ops;
 funct_5 = do_fil_ops;
 funct_6 = do_misc;

 menu_set_fun = set_more;          /*used to reset the menu*/
}

/*****************************************************************/
/*                    do_ctx_ops                                 */
/*This functions executes the context management menu            */
/*****************************************************************/
```

```
VOID do_ctx_ops()
{
 VOID set_ctx();

 set_ctx();
 while(check_key() != 1)
                        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*************************************************************/
/*                  set_ctx                                  */
/*function to set up the context management screen and       */
/*function key table                                         */
/*************************************************************/
VOID set_ctx()
{
 VOID X_do_init();
 VOID do_listen();
 VOID do_stop_listen();
 VOID do_concl();
 VOID do_abort();
 VOID do_cancel();
/*VOID do_mchan();*/
 VOID do_init_rel();
 VOID do_init_abort();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t      %c1    - INITIATE\n",c);
 printf("\t      %c2    - START ASSOCIATE LISTEN\n",c);
 printf("\t      %c3    - STOP ASSOCIATE LISTEN\n",c);
 printf("\t      %c4    - CONCLUDE\n",c);
 printf("\t      %c5    - ABORT\n",c);
 printf("\t      %c6    - CANCEL\n",c);
 printf("\t      %c7    - INITIATE / RELEASE TEST\n",c);
 printf("\t      %c8    - INITIATE / ABORT TEST\n",c);
 printf("\t      %c9    - DISPLAY MMS CHANNEL INFO\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = X_do_init;
 funct_2 = do_listen;
```

```
 funct_3 = do_stop_listen;
 funct_4 = do_concl;
 funct_5 = do_abort;
 funct_6 = do_cancel;
 funct_7 = do_init_rel;
 funct_8 = do_init_abort;
 funct_9 = do_mchan;
 menu_set_fun = set_ctx;        /*used to reset the menu*/
}

/*****************************************************************/
/*                    do_vmd_ops                                 */
/*This functions executes the vmd support menu.                  */
/*****************************************************************/
VOID do_vmd_ops()
{
 VOID set_vmd();

 set_vmd();
 while(check_key() != 1)
                /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*****************************************************************/
/*                    set_vmd                                    */
/*function to set up the vmd support screen and function key*/
/*table                                                          */
/*****************************************************************/
VOID set_vmd()
{
 VOID do_stat();
 VOID do_ustat();
 VOID do_ident();
 VOID do_rename();
 VOID do_getcl();
 VOID do_vmd_mgnt_ops();
 VOID do_namelist();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1   - STATUS\n",c);
 printf("\t     %c2   - UNSOLICITED STATUS\n",c);
 printf("\t     %c3   - IDENTIFY\n",c);
```

```
 printf("\t      %c4    - RENAME\n",c);
 printf("\t      %c5    - GET NAME LIST\n",c);
 printf("\t      %c6    - GET CAPABILITY LIST\n",c);
 printf("\t      %c7    - VMD MANAGEMENT MENU\n",c);
 printf("\t    %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_stat;
 funct_2 = do_ustat;
 funct_3 = do_ident;
 funct_4 = do_rename;
 funct_5 = do_namelist;
 funct_6 = do_getcl;
 funct_7 = do_vmd_mgnt_ops;
 menu_set_fun = set_vmd;        /*used to reset the menu  */
}

/*************************************************************/
/*                  do_vmd_mgnt_ops                          */
/*This function executes the vmd management operations      */
/*************************************************************/
VOID do_vmd_mgnt_ops()
{
 VOID set_vmd_mgnt_ops();

 set_vmd_mgnt_ops();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_vmd();
}

/*************************************************************/
/*                  set_vmd_mgnt_ops                         */
/*function to set up the vmd management screen and function */
/*key table                                                  */
/*************************************************************/
VOID set_vmd_mgnt_ops()
{
 VOID  do_add_vmd();
 VOID  do_delete_vmd();
 VOID  do_dismantle_vmd();
 VOID  do_list_vmd();
 VOID  do_select_vmd();

 CLEARSCR;
```

```
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - ADD VMD\n",c);
 printf("\t     %c2    - DISMANTLE VMD\n",c);
 printf("\t     %c3    - DELETE VMD\n",c);
 printf("\t     %c4    - LIST ALL VMDs\n",c);
 printf("\t     %c5    - SELECT CURRENT VMD\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_add_vmd;
 funct_2 = do_dismantle_vmd;
 funct_3 = do_delete_vmd;
 funct_4 = do_list_vmd;
 funct_5 = do_select_vmd;
 menu_set_fun = set_vmd_mgnt_ops;   /*used to reset the menu*/
}

/*************************************************************/
/*                  do_dom_ops                               */
/*This function executes the domain management operations    */
/*************************************************************/
VOID do_dom_ops()
{
 VOID set_dom();

 set_dom();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*************************************************************/
/*                  set_dom                                  */
/*function to set up the domain management screen and        */
/*function key table                                         */
/*************************************************************/
VOID set_dom()
{
 VOID do_udload_dom_ops();
 VOID do_loaddom();
 VOID do_storedom();
 VOID do_deldom();
 VOID do_getdom();
```

```
VOID do_add_named_dom();
VOID do_del_named_dom();
VOID do_list_named_dom();

CLEARSCR;
print_demo();
printf("\n\n");
printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
printf("\t     %c1    - DOMAIN UPLOAD/DOWNLOAD MENU\n",c);
printf("\t     %c2    - LOAD DOMAIN CONTENT\n",c);
printf("\t     %c3    - STORE DOMAIN CONTENT\n",c);
printf("\t     %c4    - DELETE DOMAIN\n",c);
printf("\t     %c5    - GET DOMAIN ATTRIBUTE\n",c);
printf("\t     %c6    - ADD NAMED DOMAIN\n",c);
printf("\t     %c7    - DELETE NAMED DOMAIN\n",c);
printf("\t     %c8    - LIST ALL NAMED DOMAINS\n",c);
printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

flush_keys();
fun_null();
funct_1 = do_udload_dom_ops;
funct_2 = do_loaddom;
funct_3 = do_storedom;
funct_4 = do_deldom;
funct_5 = do_getdom;
funct_6 = do_add_named_dom;
funct_7 = do_del_named_dom;
funct_8 = do_list_named_dom;
menu_set_fun = set_dom;     /*used to reset the menu  */
}

/*************************************************************/
/*                  do_udload_dom_ops                        */
/*This function executes the domain management operations    */
/*************************************************************/
VOID do_udload_dom_ops()
{
 VOID set_dom_udload();

 set_dom_udload();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_dom();
}
```

```
/*************************************************************/
/*                  set_dom_upload                           */
/*function to set up the domain management screen and        */
/*function key table for uploading and downloading of        */
/*domains.                                                   */
/*************************************************************/
VOID set_dom_udload()
{
 VOID do_initdown();
 VOID do_download();
 VOID do_termdown();
 VOID do_initupl();
 VOID do_upload();
 VOID do_termupl();
 VOID do_rddwn();
 VOID do_rdupl();
 VOID do_vmud_load();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t  SELECT DESIRED MMS UPLOAD/DOWNLOAD ACTIVITY\n\n");
 printf("\t   %c1  - INITIATE DOWNLOAD SEQUENCE\n",c);
 printf("\t   %c2  - DOWNLOAD SEGMENT\n",c);
 printf("\t   %c3  - TERMINATE DOWNLOAD SEQUENCE\n",c);
 printf("\t   %c4  - INITIATE UPLOAD SEQUENCE\n",c);
 printf("\t   %c5  - UPLOAD SEGMENT\n",c);
 printf("\t   %c6  - TERMINATE UPLOAD SEQUENCE\n",c);
 printf("\t   %c7  - REQUEST DOMAIN DOWNLOAD\n",c);
 printf("\t   %c8  - REQUEST DOMAIN UPLOAD\n",c);
 printf("\t   %c9  - VM UPLOAD/DOWNLOAD MENU\n",c);
 printf("\t%c<RET> - RETURN TO MMS COMMUNICATION ACTIVITY MENU\n",
                                                        cret);

 flush_keys();
 fun_null();
 funct_1 = do_initdown;
 funct_2 = do_download;
 funct_3 = do_termdown;
 funct_4 = do_initupl;
 funct_5 = do_upload;
 funct_6 = do_termupl;
 funct_7 = do_rddwn;
 funct_8 = do_rdupl;
 funct_9 = do_vmud_load;
 menu_set_fun = set_dom_udload; /*used to reset the menu*/
}
```

```
/*************************************************************/
/*                  do_vmud_load                             */
/*do Virtual Machine uploads and downloads                   */
/*************************************************************/
VOID do_vmud_load()
{
 VOID set_vmud_load();

 set_vmud_load();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_dom_udload();
}

/*************************************************************/
/*                  set_vmud_load                            */
/*function to set up the domain management screen and        */
/*function key table for uploading and downloading of        */
/*domains.                                                   */
/*************************************************************/
VOID set_vmud_load()
{
 VOID do_vmupload();
 VOID do_vmdownload();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t  SELECT DESIRED MMS VM UPLOAD/DOWNLOAD ACTIVITY\n\n");
 printf("\t    %c1   - VM UPLOAD\n",c);
 printf("\t    %c2   - VM DOWNLOAD\n",c);
 printf("\t    %c9   - VM UPLOAD/DOWNLOAD MENU\n",c);
 printf("\tc%<RET> - RETURN TO PREVIOUS MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_vmupload;
 funct_2 = do_vmdownload;
 menu_set_fun = set_vmud_load;   /*used to reset the menu*/
}
```

```
/*************************************************************/
/*                  do_prg_ops                               */
/*execute program invocation management menu                 */
/*************************************************************/
VOID do_prg_ops()
{
 VOID set_prg();

 set_prg();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*************************************************************/
/*                  set_prg                                  */
/*function to set up the program invocation screen and       */
/*function key table.                                        */
/*************************************************************/
VOID set_prg()
{
 VOID do_crepi();
 VOID do_delpi();
 VOID do_start();
 VOID do_stop();
 VOID do_resume();
 VOID do_reset();
 VOID do_kill();
 VOID do_getpi();
 VOID do_local_pi_menu();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1   - CREATE PROGRAM INVOCATION\n",c);
 printf("\t     %c2   - DELETE PROGRAM INVOCATION\n",c);
 printf("\t     %c3   - START\n",c);
 printf("\t     %c4   - STOP\n",c);
 printf("\t     %c5   - RESUME\n",c);
 printf("\t     %c6   - RESET\n",c);
 printf("\t     %c7   - KILL\n",c);
 printf("\t     %c8   - GET PROGRAM INVOCATION ATTRIBUTE\n",c);
 printf("\t     %c9   - LOCAL PI MENU\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);
```

```
 flush_keys();
 fun_null();
 funct_1 = do_crepi;
 funct_2 = do_delpi;
 funct_3 = do_start;
 funct_4 = do_stop;
 funct_5 = do_resume;
 funct_6 = do_reset;
 funct_7 = do_kill;
 funct_8 = do_getpi;
 funct_9 = do_local_pi_menu;
 menu_set_fun = set_prg;        /*used to reset the menu*/
}

/*****************************************************************/
/*                  do_local_pi_menu                             */
/*This functions executes the local Program Invocations          */
/*****************************************************************/
VOID do_local_pi_menu()
{
 VOID set_local_pi_menu();

 set_local_pi_menu();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_prg();
}

/*****************************************************************/
/*                  set_type_menu                                */
/*function to set up the local named type definition menu        */
/*****************************************************************/
VOID set_local_pi_menu()
{
 VOID do_add_pi();
 VOID do_del_pi();
 VOID do_list_local_pis();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS LOCAL PI ACTIVITY\n\n");
 printf("\t   %c1  - ADD LOCAL PROGRAM INVOCATION\n",c);
 printf("\t   %c2  - DELETE LOCAL PROGRAM INVOCATION\n",c);
 printf("\t   %c3  - LIST LOCAL PROGRAM INVOCATIONS\n",c);
```

```
 printf("\t%c<RET> - RETURN TO PI MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_add_pi;
 funct_2 = do_del_pi;
 funct_3 = do_list_local_pis;
 menu_set_fun = set_local_pi_menu;/*used to reset the menu*/
}

/*****************************************************************/
/*                    do_var_ops                                 */
/*This functions executes the variable access operations.        */
/*****************************************************************/
VOID do_var_ops()
{
 VOID set_var();

 set_var();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_main();
}

/*****************************************************************/
/*                    set_var                                    */
/*function to set up the variable access screen and function*/
/*key table                                                      */
/*****************************************************************/
VOID set_var()
{
 VOID do_read();
 VOID do_write();
 VOID do_info();
 VOID do_var_defs();
 VOID do_sa_vl_defs();
 VOID do_type_menu();
 VOID do_var_menu();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - READ\n",c);
 printf("\t     %c2    - WRITE\n",c);
 printf("\t     %c3    - INFORMATION REPORT\n",c);
```

```
        printf("\t     %c4    - TYPE/NAME DEFINITION MENU\n",c);
        printf("\t     %c5    - SCATTERED ACCESS/VARIABLE LIST DEFINITION ME
NU\n",c);
        printf("\t     %c6    - LOCAL NAMED TYPE MENU\n",c);
        printf("\t     %c7    - LOCAL NAMED VARIABLE MENU\n",c);
        printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

        flush_keys();
        fun_null();
        funct_1 = do_read;
        funct_2 = do_write;
        funct_3 = do_info;
        funct_4 = do_var_defs;
        funct_5 = do_sa_vl_defs;
        funct_6 = do_type_menu;
        funct 7 = do_var_menu;
        menu_set_fun = set_var;        /*used to reset the menu  */
       }

       /*************************************************************/
       /*                  do_var_defs                              */
       /*This functions executes the variable/type definition menu */
       /*************************************************************/
       VOID do_var_defs()
       {
        VOID set_var_defs();

        set_var_defs();
        while(check_key() != 1)
              /*Execute function keys until F10 hit*/
           mms_comm_service();
        set_var();
       }

       /*************************************************************/
       /*                    set_var_defs                           */
       /*function to set up the var def screen and function key table*/
       /*************************************************************/
       VOID set_var_defs()
       {
        VOID do_getvar();
        VOID do_defvar();
        VOID do_delvar();
        VOID do_gettype();
        VOID do_deftype();
        VOID do_deltype();

        CLEARSCR;
```

```
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - GET VARIABLE NAME DEFINITION\n",c);
 printf("\t     %c2    - DEFINE VARIABLE NAME\n",c);
 printf("\t     %c3    - DELETE VARIABLE NAME\n",c);
 printf("\t     %c4    - GET TYPE DEFINITION\n",c);
 printf("\t     %c5    - DEFINE TYPE NAME\n",c);
 printf("\t     %c6    - DELETE TYPE NAME\n",c);
 printf("\t   %c<RET> - RETURN TO VARIABLE ACCESS MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_getvar;
 funct_2 = do_defvar;
 funct_3 = do_delvar;
 funct_4 = do_gettype;
 funct_5 = do_deftype;
 funct_6 = do_deltype;
 menu_set_fun = set_var_defs;  /*used to reset the menu*/
}

/*************************************************************/
/*                  do_type_menu                             */
/*This functions executes the local named type definition    */
/*menu                                                       */
/*************************************************************/
VOID do_type_menu()
{
 VOID set_type_menu();

 set_type_menu();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_var();
}

/*************************************************************/
/*                  set_type_menu                            */
/*function to set up the local named type definition menu    */
/*************************************************************/
VOID set_type_menu()
{
 VOID do_add_named_type();
 VOID do_add_custom_type();
 VOID do_list_named_type();
```

```
VOID do_del_named_type();

CLEARSCR;
print_demo();
printf("\n\n");
printf("\t     SELECT DESIRED MMS LOCAL NAMED TYPE ACTIVITY\n\n");
printf("\t     %c1   - ADD STANDARD NAMED TYPES\n",c);
printf("\t     %c2   - ADD CUSTOM NAMED TYPE\n",c);
printf("\t     %c3   - LIST NAMED TYPES\n",c);
printf("\t     %c4   - DELETE NAMED TYPE\n",c);
printf("\t   %c<RET> - RETURN TO VARIABLE ACCESS MENU\n",cret);

flush_keys();
fun_null();
funct_1 = do_add_named_type;
funct_2 = do_add_custom_type;
funct_3 = do_list_named_type;
funct_4 = do_del_named_type;
menu_set_fun = set_type_menu;  /*used to reset the menu*/
}

/*****************************************************************/
/*                  do_sa_vl_defs                                */
/*This functions executes the scattered access/variable list*/
/*definition menu.                                               */
/*****************************************************************/
VOID do_sa_vl_defs()
{
 VOID set_sa_vl_defs();

 set_sa_vl_defs();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_var();
}

/*****************************************************************/
/*                  set_sa_vl_defs                               */
/*function to set up the scattered access & variable list   */
/*definition screen and function key table                       */
/*****************************************************************/
VOID set_sa_vl_defs()
{
 VOID do_getscat();
 VOID do_defscat();
 VOID do_getvlist();
```

```
VOID do_defvlist();
VOID do_delvlist();

CLEARSCR;
print_demo();
printf("\n\n");
printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
printf("\t    %c1  - GET SCATTERED ACCESS ATTRIBUTES\n",c);
printf("\t    %c2  - DEFINE SCATTERED ACCESS\n",c);
printf("\t    %c3  - GET NAMED VARIABLE LIST ATTRIBUTES\n",c);
printf("\t    %c4  - DEFINE NAMED VARIABLE LIST\n",c);
printf("\t    %c5  - DELETE NAMED VARIABLE LIST\n",c);
printf("\t%c<CRET> - RETURN TO VARIABLE ACCESS MENU\n",cret);

flush_keys();
fun_null();
funct_1 = do_getscat;
funct_2 = do_defscat;
funct_3 = do_getvlist;
funct_4 = do_defvlist;
funct_5 = do_delvlist;
menu_set_fun = set_sa_vl_defs;  /*used to reset the menu*/
}

/*************************************************************/
/*                  do_var_menu                              */
/*This functions executes the local named type definition    */
/*menu                                                       */
/*************************************************************/
VOID do_var_menu()
{
 VOID set_var_menu();

 set_var_menu();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_var();
}
```

```
/*************************************************************/
/*                    set_var_menu                           */
/*function to set up the var def screen and function key     */
/*table                                                      */
/*************************************************************/
VOID set_var_menu()
{
 VOID do_add_named_var();
 VOID do_del_named_var();
 VOID do_list_named_var();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS LOCAL NAMED VARIABLE ACTIVITY\n\n
");
 printf("\t      %c1    - ADD LOCAL NAMED VARIABLE\n",c);
 printf("\t      %c2    - DELETE LOCAL NAMED VARIABLE\n",c);
 printf("\t      %c3    - LIST LOCAL NAMED VARIABLES\n",c);
 printf("\t    %c<RET> - RETURN TO VARIABLE ACCESS MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_add_named_var;
 funct_2 = do_del_named_var;
 funct_3 = do_list_named_var;
 menu_set_fun = set_var_menu;  /*used to reset the menu*/
}

/*************************************************************/
/*                    do_sem_ops                             */
/*This functions executes the semaphore management operations.*/
/*************************************************************/
VOID do_sem_ops()
{
 VOID set_sem();

 set_sem();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}
```

```
/*****************************************************************/
/*                    set_sem                                    */
/*function to set up the semaphore management screen and         */
/*function key table                                             */
/*****************************************************************/
VOID set_sem()
{
 VOID do_takectrl();
 VOID do_relctrl();
 VOID do_rsstat();
 VOID do_rspool();
 VOID do_rsentry();
 VOID do_defsem();
 VOID do_delsem();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - TAKE CONTROL\n",c);
 printf("\t     %c2    - RELINQUISH CONTROL\n",c);
 printf("\t     %c3    - REPORT SEMAPHORE STATUS\n",c);
 printf("\t     %c4    - REPORT POOL SEMAPHORE STATUS\n",c);
 printf("\t     %c5    - REPORT SEMAPHORE ENTRY STATUS\n",c);
 printf("\t     %c6    - DEFINE SEMAPHORE\n",c);
 printf("\t     %c7    - DELETE SEMAPHORE\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_takectrl;
 funct_2 = do_relctrl;
 funct_3 = do_rsstat;
 funct_4 = do_rspool;
 funct_5 = do_rsentry;
 funct_6 = do_defsem;
 funct_7 = do_delsem;
 menu_set_fun = set_sem;      /*used to reset the menu*/
}
```

```
/*****************************************************************/
/*                  do_ocs_ops                                   */
/*This functions executes the operator communication             */
/*operations.                                                    */
/*****************************************************************/
VOID do_ocs_ops()
{
 VOID set_ocs();

 set_ocs();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}

/*****************************************************************/
/*                  set_ocs                                      */
/*function to set up the operator communication screen and       */
/*function key table                                             */
/*****************************************************************/
VOID set_ocs()
{
 VOID do_output();
 VOID do_input();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - OUTPUT OPERATION\n",c);
 printf("\t     %c2    - INPUT OPERATION\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_output;
 funct_2 = do_input;
 menu_set_fun = set ocs;       /*used to reset the menu*/
}
```


41

```
/*************************************************************/
/*                    do_evn_ops                             */
/*This functions executes the event management operations.   */
/*************************************************************/
VOID do_evn_ops()
{
 VOID set_evn();

 set_evn();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}

/*************************************************************/
/*                    set_evn                                */
/*function to set up the event management screen            */
/*and function key table                                    */
/*************************************************************/
VOID set_evn()
{
 VOID do_ec();
 VOID do_ea();
 VOID do_ee();
 VOID do_trige();
 VOID do_evnot();
 VOID do_ackevnot();
 VOID do_getas();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t      %c1  - EVENT CONDITION MENU\n",c);
 printf("\t      %c2  - EVENT ACTION MENU\n",c);
 printf("\t      %c3  - EVENT ENROLLMENT MENU\n",c);
 printf("\t      %c4  - TRIGGER EVENT\n",c);
 printf("\t      %c5  - EVENT NOTIFICATION\n",c);
 printf("\t      %c6  - ACKNOWLEDGE EVENT NOTIFICATION\n",c);
 printf("\t      %c7  - GET ALARM SUMMARY\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_ec;
```

```
 funct_2 = do_ea;
 funct_3 = do_ee;
 funct_4 = do_trige;
 funct_5 = do_evnot;
 funct_6 = do_ackevnot;
 funct_7 = do_getas;
 menu_set_fun = set_evn; /*used to reset the menu*/
}

/*************************************************************/
/*                  do_ec                                    */
/*This functions executes the event condition operations.    */
/*************************************************************/
VOID do_ec()
{
 VOID set_ec();

 set_ec();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_evn();
}

/*************************************************************/
/*                  set ec                                   */
/*function to set up the event condition screen and function*/
/*key table                                                  */
/*************************************************************/
VOID set_ec()
{
 VOID do_defec();
 VOID do_delec();
 VOID do_geteca();
 VOID do_repecs();
 VOID do_altecm();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - DEFINE EVENT CONDITION\n",c);
 printf("\t     %c2    - DELETE EVENT CONDITION\n",c);
 printf("\t     %c3    - GET EVENT CONDITION ATTRIBUTES\n",c);
 printf("\t     %c4    - REPORT EVENT CONDITION STATUS\n",c);
 printf("\t     %c5    - ALTER EVENT CONDITION MONITORING\n",c);
 printf("\t  %c<CRET> - RETURN TO MAIN MENU\n",cret);
```

```
 flush_keys();
 fun_null();
 funct_1 = do_defec;
 funct_2 = do_delec;
 funct_3 = do_geteca;
 funct_4 = do_repecs;
 funct_5 = do_altecm;
 menu_set_fun = set_ec;    /*used to reset the menu*/
}

/*************************************************************/
/*                  do_ea                                    */
/*This functions executes the event action operations.       */
/*************************************************************/
VOID do_ea()
{
 VOID set_ea();

 set_ea();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_evn();
}

/*************************************************************/
/*                  set_ea                                   */
/*function to set up the event action screen and function    */
/*key table                                                  */
/*************************************************************/
VOID set_ea()
{
 VOID do_defea();
 VOID do_delea();
 VOID do_geteaa();
 VOID do_repeas();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t    %c1   - DEFINE EVENT ACTION\n",c);
 printf("\t    %c2   - DELETE EVENT ACTION\n",c);
 printf("\t    %c3   - GET EVENT ACTION ATTRIBUTES\n",c);
 printf("\t    %c4   - REPORT EVENT ACTION STATUS\n",c);
 printf("\t %c<CRET> - RETURN TO MAIN MENU\n",cret);
```


HH

```
 flush_keys();
 fun_null();
 funct_1 = do_defea;
 funct_2 = do_delea;
 funct_3 = do_geteaa;
 funct_4 = do_repeas;
 menu_set_fun = set_ea;        /*used to reset the menu  */
}

/*****************************************************************/
/*                  do_ee                                        */
/*This functions executes the event enrollment operations.  */
/*****************************************************************/
VOID do_ee()
{
 VOID set_ee();

 set_ee();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_evn();
}

/*****************************************************************/
/*                   set_ee                                      */
/*function to set up the event enrollment screen and function*/
/*key table                                                     */
/*****************************************************************/
VOID set_ee()
{
 VOID do_defee();
 VOID do_delee();
 VOID do_altee();
 VOID do_repees();
 VOID do_geteea();
 VOID do_getaes();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t    %c1   - DEFINE EVENT ENROLLMENT\n",c);
 printf("\t    %c2   - DELETE EVENT ENROLLMENT\n",c);
 printf("\t    %c3   - ALTER EVENT ENROLLMENT\n",c);
 printf("\t    %c4   - REPORT EVENT ENROLLMENT STATUS\n",c);
```

```
 printf("\t    %c5   - GET EVENT ENROLLMENT ATTRIBUTES\n",c);
 printf("\t    %c6   - GET ALARM ENROLLMENT SUMMARY\n",c);
 printf("\t %c<CRET> - RETURN TO MAIN MENU\n",c);

 flush_keys();
 fun_null();
 funct_1 = do_defee;
 funct_2 = do_delee;
 funct_3 = do_altee;
 funct_4 = do_repees;
 funct_5 = do_geteea;
 funct_6 = do_getaes;
 menu_set_fun = set_ee;        /*used to reset the menu*/
}

/*************************************************************/
/*                  do_jou_ops                               */
/*This functions executes the journal management operations.*/
/*************************************************************/
VOID do_jou_ops()
{
 VOID set_jou();

 set_jou();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}

/*************************************************************/
/*                  set_jou                                  */
/*function to set up the journal management screen and       */
/*function key table                                         */
/*************************************************************/
VOID set_jou()
{
 VOID do_jwrite();
 VOID do_jread();
 VOID do_jinit();
 VOID do_jstat();
 VOID do_jcreate();
 VOID do_jdelete();

 CLEARSCR;
 print_demo();
 printf("\n\n");
```

```
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1  - WRITE JOURNAL\n",c);
 printf("\t     %c2  - READ JOURNAL\n",c);
 printf("\t     %c3  - INITIALIZE JOURNAL\n",c);
 printf("\t     %c4  - REPORT JOURNAL STATUS\n",c);
 printf("\t     %c5  - CREATE JOURNAL\n",c);
 printf("\t     %c6  - DELETE JOURNAL\n",c);
 printf("\t  %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_jwrite;
 funct_2 = do_jread;
 funct_3 = do_jinit;
 funct_4 = do_jstat;
 funct_5 = do_jcreate;
 funct_6 = do_jdelete;
 menu_set_fun = set_jou;  /*used to reset the menu*/
}

/*************************************************************/
/*                  do_fil_ops                               */
/*This function executes the file operation menu             */
/*************************************************************/
VOID do_fil_ops()
{
 VOID set_file();

 set_file();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}

/*************************************************************/
/*                  set_file                                 */
/*function to set up the file operations screen and function*/
/*key table                                                  */
/*************************************************************/
VOID set_file()
{
 VOID do_fcopy();
 VOID do_obtfile();
 VOID do_fopen();
 VOID do_fread();
 VOID do_fclose();
```

```
 VOID do_file_mgt();
 VOID do_disp_rem_files();
 VOID do_disp_loc_files();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1  - COPY FILE\n",c);
 printf("\t     %c2  - OBTAIN FILE\n",c);
 printf("\t     %c3  - FILE OPEN\n",c);
 printf("\t     %c4  - FILE READ\n",c);
 printf("\t     %c5  - FILE CLOSE\n",c);
 printf("\t     %c6  - FILE MANAGEMENT MENU\n",c);
 printf("\t     %c7  - DISPLAY OPEN LOCAL FILES\n",c);
 printf("\t     %c8  - DISPLAY OPEN REMOTE FILES\n",c);
 printf("\t  %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_fcopy;
 funct_2 = do_obtfile;
 funct_3 = do_fopen;
 funct_4 = do_fread;
 funct_5 = do_fclose;
 funct_6 = do_file_mgt;
 funct_7 = do_disp_loc_files;
 funct_8 = do_disp_rem_files;
 menu_set_fun = set_file;     /*used to reset the menu*/
}

/*************************************************************/
/*                  do_file_mgt                              */
/*function to execute file management menu                   */
/*************************************************************/
VOID do_file_mgt()
{
 VOID set_file_mgt();

 set_file_mgt();
 while(check_key() != 1)
       /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_file();
}
```

```
/*************************************************************/
/*                  set_file_mgt                             */
/*function to set up the file management screen and function*/
/*key table                                                  */
/*************************************************************/
VOID set_file_mgt()
{
 VOID do_frename();
 VOID do_fdelete();
 VOID do_fdir();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED MMS COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - FILE RENAME\n",c);
 printf("\t     %c2    - FILE DELETE\n",c);
 printf("\t     %c3    - FILE DIRECTORY\n",c);
 printf("\t   %c<RET> - RETURN TO FILE OPERATIONS MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_frename;
 funct_2 = do_fdelete;
 funct_3 = do_fdir;
 menu_set_fun = set_file_mgt;    /*used to reset the menu*/
}

/*************************************************************/
/*                  do_misc                                  */
/*This function executes the miscellaneous set of test      */
/*features                                                   */
/*************************************************************/
VOID do_misc()
{
 VOID set_misc();

 set_misc();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_more();
}
```

```
/*****************************************************************/
/*                    set_misc                                   */
/*function to set up the miscellaneous screen and function       */
/*key table                                                      */
/*****************************************************************/
VOID set_misc()
{
 VOID do_sys_com();
 VOID do_debugset();
 VOID do_suic();
 VOID do_set_cont();
 VOID do_wait_event();
 VOID do_mms_err();
 VOID do_set_mod();
 VOID do_test();


 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED ACTIVITY\n\n");
 printf("\t     %c1    - SETUP CONTINUOUS SEND\n",c);
 printf("\t     %c2    - ISSUE SYSTEM COMMAND\n",c);
 printf("\t     %c3    - SET MMS DEBUG LEVELS\n",c);
 printf("\t     %c4    - WAIT FOR MMS EVENT\n",c);
 printf("\t     %c5    - SUIC MENU\n",c);
 printf("\t     %c6    - DISPLAY MMS-EASE ERROR CODE DESCRIPTION\n",c
);

 printf("\t     %c7    - SET MMS SERVICE MODIFIER\n",c);
 printf("\t     %c8    - TEST MENU\n",c);
 printf("\t   %c<RET> - RETURN TO MAIN MENU\n",cret);

 flush_keys();
 fun_null();
 funct_1 = do_set_cont;
 funct_2 = do_sys_com;
 funct_3 = do_debugset;
 funct_4 = do_wait_event;
 funct_5 = do_suic;
 funct_6 = do_mms_err;
 funct_7 = do_set_mod;
 funct_8 = do_test;                    /*Internal use only*/
 menu_set_fun = set_misc;            /*used to reset the menu*/
}
```

```
/*****************************************************************/
/*                    do_suic                                    */
/*execute SUIC function menu                                     */
/*****************************************************************/
VOID do_suic()
{
 VOID set_suic();

 set_suic();
 while(check_key() != 1)
        /*Execute function keys until F10 hit*/
    mms_comm_service();
 set_misc();
}

/*****************************************************************/
/*                    set_suic                                   */
/*function to set up the suic screen and function key table */
/*****************************************************************/
VOID set_suic()
{
 VOID do_activate();
 VOID do_deactivate();
 VOID do_register();
 VOID comm_stat();
 VOID debug_set();
 VOID set_remote_dest_addr();
 VOID set_name_alias();

 CLEARSCR;
 print_demo();
 printf("\n\n");
 printf("\t     SELECT DESIRED SUIC COMMUNICATION ACTIVITY\n\n");
 printf("\t     %c1    - ACTIVATE AR NAME\n",c);
 printf("\t     %c2    - DE-ACTIVATE AR NAME\n",c);
 printf("\t     %c3    - REGISTER AR NAME\n",c);
 printf("\t     %c4    - DISPLAY STATUS OF COMMUNICATION SYSTEM\n",c)
;

 printf("\t     %c5    - MODIFY DEBUG MESSAGE LEVEL\n",c);
#ifdef MAP30_LLC
 printf("\t     %c6    - Set Remote Alias\n",c);
 printf("\t     %c7    - Set Remote Address Via Alias\n",c);
#endif
 printf("\t   %c<RET> - RETURN TO MISC. MENU\n",cret);

 flush_keys();
 fun_null();
```

```
 funct_1 = do_activate;
 funct_2 = do_deactivate;
 funct_3 = do_register;
 funct_4 = comm_stat;
 funct_5 = debug_set;
#ifdef MAP30_LLC
 funct_6 = set_name_alias;
 funct_7 = set_remote_dest_addr;
#endif
 menu_set_fun = set_suic;       /*used to reset the menu*/
}

/*************************************************************/
/*                  print_demo                               */
/*function to display the demo heading                       */
/*************************************************************/
VOID print_demo()
{
 printf("*************************************************");
 printf("***************\n");
 printf("****                  MMSEASE DEMO RELEASE VERSION 2.3 ");
 printf("             ***\n");
 printf("****                Mini - MAP DEMO RELEASE VERSION 0.");
 printf("0            ***\n");
 printf("****             (c) 1989  COMPUTROL a division of AE");
 printf("G            ***\n");
 printf("****                              (c) 1993  PIAP          ");
 printf("             ***\n");
 printf("*************************************************");
 printf("***************");
}
```

Załącznik 2.

Funkcja X_do_init

```
/*************************************************************/
/*                          X_do_init                        */
/*create and send an initiate request.                       */
/*************************************************************/
VOID X_do_init()
{
 BYTE title[180];
 SHORT chan;
 unsigned int value;
 SHORT AP_Flag;
 char tmp[50];
 SHORT i,len;
 SHORT app_fcn;
 SHORT app_dpy_sel;
 BOOLEAN enable;
 static BOOLEAN ap ae_invoke;
 BOOLEAN enable1 = 0;
 static BOOLEAN user;
 UBYTE selector;
 SHORT ret_code;

#if MMS_INIT_EN & REQ_EN
 printf("\n Enter the Channel # To Use : ");
 if(!intget(&chan))
   {
    (*menu_set_fun)();
    return;
   }
 if(chan >= max_mms_chan || chan < 0) /*if not legal channel*/
   {
    wait_msg("Illegal channel number");
    (*menu_set_fun)();
    return;
   }

 printf(" Enter the Application Process Title to Connect To : ");
 if(!strget(title))
   {
    (*menu_set_fun)();
    return;
   }
```

```
/*We now have all information required to establish
                                                an association*/
/*mms_chan_info[chan].segsize = m_segsize;*/

/*-------- AP TITLE Stuff Begin ---------*/

 app_fcn = 0; /*assume disable all AP title stuff(0)*/
 selector = 3; /*default is clean up both local and partner*/

 if((ask("\n\n Use AP Titles(N) ?",0)))
   {
    enable = 1;
    enable1 = 1;
    app_fcn = 10;        /*Assume DIB*/
    if((!ask("\n\n Use DIB File(Y) ?",1)))
/*fall thru if not using DIB*/
     {
      user = 1;
      app_fcn = 2; /*assume local and partner stuff*/
      ap_ae_invoke = 1;
      if((!ask("\n\n Use existing AP Titles(Y) ?",1)))
       {
        app_dpy_sel = 0; /*initially assume nothing set*/
        selector = 0; /*default set nothing*/

  /*------- get and stuff Partner obj id ----------------*/

        printf(" \n All input is in hex \n");
        printf(" \n Enter the PARTNER OBJ ID component 1 ");
        printf("CCITT(0) ISO(1) ISO-CCITT(2): ");
        if(!strget(tmp))
          {
           (*menu_set_fun)();
           return;
          }
        if(tmp[0] < '0' || tmp[0] > '2')
          {
           printf("\nInvalid value - allowed values 0-2\n");
           (*menu_set_fun)();
           return;
          }

        APP_INFO.part_AP_title.comps[0] = (SHORT) atoi(&tmp[0]);
```

```
printf(" Enter the PARTNER OBJ ID component 2 STD(0)");
printf(", REG AUTH(1), MEMBER BODY(2), IDENT ORG(3) : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
if(tmp[0] < '0' || tmp[0] > '3')
 {
  printf("\nInvalid value - allowed values 0-3\n");
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_title.comps[1] = (SHORT) atoi(&tmp[0]);

printf(" \n Enter the PARTNER OBJ ID component 3  : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }

len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
     value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
     value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
     value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_title.comps[2] = value;

printf(" \n Enter the PARTNER OBJ ID component 4  : ");
```

```
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
       value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
       value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
       value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_title.comps[3] = value;

printf(" \n Enter the PARTNER OBJ ID component 5  : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
```

```
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_title.comps[4] = value;

printf(" \n Enter the PARTNER OBJ ID component 6   : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
     value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
     value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
     value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_title.comps[5] = value;
APP_INFO.part_AP_title.num_comps = 6;

printf(" \n Enter the PARTNER AE QUALIFIER : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
```

```
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
     value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
     value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
     value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AE_qual = value;

printf(" \n Enter the PARTNER AP Invoke ID : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
     value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
     value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
     value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AP_invoke_id = value;

printf(" \n Enter the PARTNER AE Invoke ID : ");
```

```
        if(!strget(tmp))
          {
           (*menu_set_fun)();
           return;
          }
        len = strlen(tmp);
        value = 0;
        for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
         {
          if(tmp[i] >= '0' && tmp[i] <= '9')
            value = (value*16) + (tmp[i] - '0');
          if(tmp[i] >= 'a' && tmp[i] <= 'f')
            value = (value*16) + (tmp[i] - 'W');
          if(tmp[i] >= 'A' && tmp[i] <= 'F')
            value = (value*16) + tmp[i] - '7';
         }
        if(i != len)
         {
          key_err("Must be a hex digit");
          sleep(2);
          (*menu_set_fun)();
          return;
         }
        APP_INFO.part_AE_invoke_id = value;

/*------- Get and stuff Local obj id ------------------*/
        printf(" \n Enter LOCAL the OBJ ID component 1 ");
        printf(" CCITT(0) ISO(1) ISO-CCITT(2): ");
        if(!strget(tmp))
         {
          (*menu_set fun) ();
          return;
         }

        if(tmp[0] < '0' || tmp[0] > '2')
         {
          printf("\nInvalid value - allowed values 0-2\n");
          (*menu_set_fun)();
          return;
         }

        APP_INFO.AP_title.comps[0] = (SHORT) atoi(&tmp[0]);

        printf(" Enter the LOCAL OBJ ID component 2 STD(0),");
        printf(" REG AUTH(1), MEMBER BODY(2), IDENT ORG(3) : ");
```

```
if(!strget(tmp))
  {
   (*menu_set_fun)();
   return;
  }
if(tmp[0] < '0' || tmp[0] > '3')
  {
   printf("\nInvalid value - allowed values 0-3\n");
   (*menu_set_fun)();
   return;
  }
APP_INFO.AP_title.comps[1] =(SHORT) atoi(&tmp[0]);

printf(" \n Enter the LOCAL OBJ ID component 3  : ");
if(!strget(tmp))
  {
   (*menu_set_fun)();
   return;
  }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
  {
   key_err("Must be a hex digit");
   sleep(2);
   (*menu_set_fun)();
   return;
  }
APP_INFO.AP_title.comps[2] = value;

printf(" \n Enter the LOCAL OBJ ID component 4 : ");
if(!strget(tmp))
  {
   (*menu_set_fun)();
   return;
  }
len = strlen(tmp);
value = 0;
```

```
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
     value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
     value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
     value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AP_title.comps[3] = value;

printf(" \n Enter the LOCAL OBJ ID component 5 : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AP_title.comps[4] = value;

printf(" \n Enter the LOCAL OBJ ID component 6 : ");
```

```
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AP_title.comps[5] = value;
APP_INFO.AP_title.num_comps = 6;

printf(" \n Enter the LOCAL AE QUALIFIER : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
```

```
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AE_qual = value;

printf(" \n Enter the LOCAL AP Invoke ID : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AP_invoke_id = value;

printf(" \n Enter the LOCAL AE Invoke ID : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
```

```
    for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
     {
      if(tmp[i] >= '0' && tmp[i] <= '9')
        value = (value*16) + (tmp[i] - '0');
      if(tmp[i] >= 'a' && tmp[i] <= 'f')
        value = (value*16) + (tmp[i] - 'W');
      if(tmp[i] >= 'A' && tmp[i] <= 'F')
        value = (value*16) + tmp[i] - '7';
     }
    if(i != len)
     {
      key_err("Must be a hex digit");
      sleep(2);
      (*menu_set_fun)();
      return;
     }
    APP_INFO.AE_invoke_id = value;

    if(app_fcn == 2 || app_fcn == 10)
                     /*sel is override DIB or use DIB*/
      {
       app_dpy_sel = 3;
       switch(app_fcn)
        {
         case 2:
            if(app_dpy_sel == 3)
                 selector = 3;
            else if(app_dpy_sel == 2)
                 selector = 2;
            else if(app_dpy_sel == 1)
                 selector = 1;
            user = 1;
            ap_ae_invoke = 1;
            enable = 1;
            enable1 = 1;
            break;
         case 10:
            user = 0;
            selector = 3;
            enable = 1;
            enable1 = 1;
            break;
        }
      }
   }
}
```

```
else
 {
  user = 0;
  if((!ask("\n\n Use AP,AE Invoke ID(Y) ?",1)))
    {
     enable1 = 0;
     ap_ae_invoke = 0;
    }
  else
  if((!ask("\n\n Use existing AP,AE Invoke ID(Y) ?",1)))
   {
    ap_ae_invoke = 1;
    enable1 = 1;
    printf(" \n Enter the PARTNER AP Invoke ID : ");
    if(!strget(tmp))
     {
      (*menu_set_fun)();
      return;
     }
    len = strlen(tmp);
    value = 0;
    for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
     {
      if(tmp[i] >= '0' && tmp[i] <= '9')
        value = (value*16) + (tmp[i] - '0');
      if(tmp[i] >= 'a' && tmp[i] <= 'f')
        value = (value*16) + (tmp[i] - 'W');
      if(tmp[i] >= 'A' && tmp[i] <= 'F')
        value = (value*16) + tmp[i] - '7';
     }
    if(i != len)
     {
      key_err("Must be a hex digit");
      sleep(2);
      (*menu_set_fun)();
      return;
     }
    APP_INFO.part_AP_invoke_id = value;

    printf(" \n Enter the PARTNER AE Invoke ID : ");
    if(!strget(tmp))
     {
      (*menu_set_fun)();
      return;
     }
    len = strlen(tmp);
    value = 0;
```

```
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.part_AE_invoke_id = value;

printf(" \n Enter the LOCAL AP Invoke ID : ");
if(!strget(tmp))
 {
  (*menu_set_fun)();
  return;
 }
len = strlen(tmp);
value = 0;
for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
 {
  if(tmp[i] >= '0' && tmp[i] <= '9')
    value = (value*16) + (tmp[i] - '0');
  if(tmp[i] >= 'a' && tmp[i] <= 'f')
    value = (value*16) + (tmp[i] - 'W');
  if(tmp[i] >= 'A' && tmp[i] <= 'F')
    value = (value*16) + tmp[i] - '7';
 }
if(i != len)
 {
  key_err("Must be a hex digit");
  sleep(2);
  (*menu_set_fun)();
  return;
 }
APP_INFO.AP_invoke_id = value;

printf(" \n Enter the LOCAL AE Invoke ID : ");
```

```
        if(!strget(tmp))
         {
          (*menu_set_fun)();
          return;
         }
        len = strlen(tmp);
        value = 0;
        for(i = 0; i <= len && len && (isxdigit(tmp[i])); ++i)
         {
          if(tmp[i] >= '0' && tmp[i] <= '9')
            value = (value*16) + (tmp[i] - '0');
          if(tmp[i] >= 'a' && tmp[i] <= 'f')
            value = (value*16) + (tmp[i] - 'W');
          if(tmp[i] >= 'A' && tmp[i] <= 'F')
            value = (value*16) + tmp[i] - '7';
         }
        if(i != len)
         {
          key_err("Must be a hex digit");
          sleep(2);
          (*menu_set_fun)();
          return;
         }
        APP_INFO.AE_invoke_id = value;
       }

    }

  }/*end of main if any app stuff clause*/
else
  {
   enable = 0;
   ap_ae_invoke = 0;
  }

if(ret_code = ms_set_ap_title(chan, enable, user, selector,
       (OBJ_ID *)&APP_INFO.AP_title,
       (OBJ_ID *)&APP_INFO.part_AP_title))
  printf("ms_set_ap_title: ERROR - %d", ret_code);

if(ret_code = ms_set_ae_qual(chan, enable, user, selector,
       APP_INFO.AE_qual, APP_INFO.part_AE_qual))
  printf("ms_set_ae_qual: ERROR - %d", ret_code);
```

```
 if(ret_code = ms_set_ap_invoke(chan, enable1, ap_ae_invoke,
         selector, APP_INFO.AP_invoke_id,
         APP_INFO.part_AP_invoke_id))
   printf("ms_set_ap_invoke: ERROR - %d", ret_code);

 if(ret_code = ms_set_ae_invoke(chan, enable1, ap_ae_invoke,
         selector, APP_INFO.AE_invoke_id,
         APP_INFO.part_AE_invoke_id))
   printf("ms_set_ae_invoke: ERROR - %d", ret_code);

/*-------  AP TITLE Stuff End  --------*/
/*We now have all information required to establish
                                        an association*/

 if(!mv_init(chan,title))
      print_req_error();

 (*menu_set_fun)();

#endif
}
```