```
/*
  Selector.c, 1 1.1 1995/09/21 Yoshiyuki Fukuda
 *
 *    v 1.1: 1995/09/21
 *           first definition using OSF/Motif
 *    v 1.2: 1996/07/31
 *           Interigent control and easy operation
 *    v 1.3: 1996/11/14
 *           Change message length 132 -> 512
 *    v 1.4: 1997/11/25
 *           Disable same run number for starting next run
 *    v 1.5: 1997/12/09
 *           Some modification for indicator (not font listed)
 *    v 1.6: 1998/07/10
 *           Add decay electron mode
 *    v 1.7  1998/11/28
 *           Add AFTER_TRIGGER mode
 *    v 1.8  1999/01/20
 *           Add AFTER TRIGGER in NORMAL run mode
 *    v 1.9  1999/11/05
 *           Add DTcal and DTtrig
 *    v 1.10 2000/03/07
 *           Change DTtriger SLE threshold as 212mV from 222mV
 *    v 1.11 2000/09/11
 *           Debug run_title changed when run mode changed and released
 *           old_sle_threshold
 *    v 1.12 2000/10/01
 *           Change SLE threshold
 *    v 2.00 2002/08/08
 *           Direct assign the XtCallbackProc and XtPointer on XtCallback
 *           Change the font_list_entry to fontlist
 *           Add dummy label at XmCreteOptionMenu
 *    v 2.01 Change lenght of hostname (sukon0X 6 to 7 characters)
 *           Add color map on 22-Oct-2002
 *
 *    v 2.02 2002/12/03
 *           Renamed from selector.inter.c to selector.c. Original selector.c
 *           was backed up as selector.c.021203 . Y.Obayashi
 *    v 2.03 2002/12/17
 *           Renamed from selector.c to Selector.c.  M.Shiozawa
 *    v 3.00 2003/08/01
 *           Reduce number of button for operations. only start and stop
 *           and correct small bugs (not influenced)  Y.Fukuda
 *    v 3.01 2003/10/03
 *           Add Prescaling clock trigger Y.Obayashi
 *    v 4.00 2005/07/01
 *           Add Automatic run change bnutton Y.Obayashi
 */
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xutil.h>
#include <X11/Xatom.h>
#include <Xm/Xm.h>
#include <Xm/SelectioBP.h>
#include <Xm/SelectioB.h>
#include <Xm/PushB.h>
#include <Xm/PushBG.h>
#include <Xm/ToggleB.h>
#include <Xm/ToggleBG.h>
#include <Xm/Command.h>
```

```
#include <Xm/DrawingA.h>
#include <Xm/MainW.h>
#include <Xm/RowColumn.h>
#include <Xm/ScrollBar.h>
#include <Xm/CascadeB.h>
#include <Xm/Frame.h>
#include <Xm/ScrolledW.h>
#include <Xm/Text.h>
#include <Xm/TextF.h>
#include <Xm/XmStrDefs.h>
#include <Xm/PanedW.h>
#include <Xm/Form.h>
#include <Xm/DragDrop.h>
#include <Xm/Screen.h>
#include <Xm/DragIcon.h>
#include <Xm/Label.h>
#include <Xm/LabelG.h>
#include <Xm/MessageB.h>
#include <Xm/MenuShell.h>
#include <Xm/SeparatoG.h>
#include <sys/ioctl.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/sem.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/param.h>
#include <dirent.h>
#include "process.h"
#include "network.h"
#include "shmdef.h"
#include "keydef.h"
#include "cmd.h"
#include "runmode.h"
#include "discri.h"

#define DEFAULT_HOST "sukonh"
#define KEY  0x700
#define PERM 0666
#define NUMSMP 6
#define MAX_LINE 80
#define TRIGGER_NUM_EXTRA 1

/* definition for default value of SLE threshold in mV/2
   i.e. in the case of 110mV: SLE_THRESHOLD=55
   23-Jul-2003 Y.Obayashi */
/* 23-Jul-2003 SLE threshold = 110mV */
#define SLE_THRESHOLD 55

extern void emerg_exit();
```

```c
extern int errno;

struct shmdef *shm;
struct pidtab *abo;
struct msgbuf *buffer;

long int msgid;
Widget toplevel;
Widget mainw;
Widget menu, menus;
Widget menub[2];
Widget frame, popup[NUM_SERV];
Widget run_number, num_field;
Widget leader, leader_field;
Widget member, member_field;
Widget comment, comment_field;
Widget max_record, max_record_field;
Widget record_path, record_path_field;
Widget atm_threshold, threshold_field;
Widget master_threshold[8], master_field[8];
Widget pedestal, disk_write, raw_save, elec[7];
Widget button, setup, setup2, setup3, setup5,network, popupBtn[15];
Widget server[NUM_SERV], server_type[NUM_SERV], server_label[NUM_SERV];
Widget submenu, submenus, localform[NUM_SERV], dragIcon;
Widget swindow, scrollbar, row_column, panedw, form1, form2, form3, form4, form5
, form6, form7;
Widget control, create, destroy, start, stop, change, quit;
Widget abort, srvnum, smp_mask, puBtn[NUM_SERV], mask_field;
Widget pdmenu, serverBtn[NUM_SERV][NUMSMP], smp[NUM_SERV][NUMSMP];
Widget smp_on[NUM_SERV][NUMSMP], smp_off[NUM_SERV][NUMSMP], smp_mask,  head[NUM_
SERV];
Widget trigger_select[TRIGGER_NUM+TRIGGER_NUM_EXTRA], run_mode, run_type, mode_l
abel, high_rate;
Widget pbg[10], stop_mode, sel_status;

u_long  fg, bg;
Pixmap  source;
XColor  exact, green, red, blue, yellow;
Colormap cmap;
Display *dsp;
GC gc;
Font font;
Window win, root;
u_int pixel, server_number;
Status result;
int sem_id;
int wx, wy, status;
u_int wwidth, wheight, wdepth, wborder;
/*
char font1[] = "-adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1";
*/
char font1[] = "-adobe-courier-bold-r-normal--14-100-100-100-m-90-iso8859-1";
XmFontListEntry font_list_entry;
XmFontList fontlist;
char host[10];
char word[80];
char node[NUM_SERV][10];
u_short continue_flag, flag_freeze = FALSE, flag_save_run = FALSE;
u_short f_error = FALSE;

XmTextPosition last_pos;
XmString label_string, title_table[NUM_SERV];
```

```c
Arg args[10];

#define MSGLEN 512 /* debugging for 132 -> 512 */
#define SNDTYP    1
#define RCVTYP    2
int msgchk();

int old_le_threshold, old_sle_threshold;
FILE *fd;
time_t ptime;

void handler();

sigset_t nsigset;
struct sigaction act;
struct sigaltstack sigstk, osigstk;

void QuitCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  exit(0);
}

void HelpCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  Widget mbox, btn;
  int i;
  char message[512];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;

  if((int)client_data == CNTL_INSTALL)
    strcpy(message,
            "Make a start client and novad in all workstation. \n");
  else if((int)client_data == CNTL_INITIALIZE)
    strcpy(message,
            "Make a start collector, sorter, sender... in all workstation. \n");
  else if((int)client_data == CNTL_SETUP)
    strcpy(message,
            "Check electronics and after then start data taking. \n");
  else if ((int)client_data == CNTL_START)
    strcpy(message,
            "Make a start data taking. \n");
  else if((int)client_data == CNTL_STOP)
    strcpy(message,
            "Make a stop data taking. \n");
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("help", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
```

```
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(btn);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(btn);
  XtManageChild(mbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
}

void FreezeCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = TRUE;
}

void AbortProcCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i, j;
  int status, rval;
  Widget mbox, btn;
  char message[512];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  char fnam[1024], msg[80];
/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: abort button pushed\n", ctime(&ptime));
  fflush(fd);
#endif DEBUG
/*
 *  disabel double operation
 */
  if(shm->flag_status == CNTL_ABORT) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: abort has already done.\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    strcpy(message, "abort has already done.");
    f_error = TRUE;
    goto ERRatABP;
  }
/*
 *  reset error flag
 */
```

```
  f_error = FALSE;
/*
 *  insensitive abort button
 */
/*
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(abort, args, i);
*/
/*
 *  set process killing flag
 */
  shm->flag_process_exist = FALSE;
/*
 *  set command message
 */
  strncpy(buffer->mtext, "abort\0", 6);
/*
 *  send message to controller
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  read answer from controller
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_ABORT);
/*
 *  set process produce flag
 */
  if(rval == 0) {
    shm->flag_process_exist = FALSE;
    strncpy(message, "Abort has done succesfully\0", 27);
  }
  else if(rval ==-2){
    f_error = TRUE;
    strcpy(message, "Still data transferring. Not aborted.\0");
  }
  else {
    f_error = TRUE;
    strncpy(message, "Abort was failed. Ohhh\0", 23);
  }
/*
```

```
 *  reaching points
 */
 ERRatABP:
/*
 *  create status message board
 */
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(btn);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(btn);
  i = 0;
  if(f_error)
    XtSetArg(args[i], XmNdialogType, XmDIALOG_ERROR);
  else
    XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);
  i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
/*
 *  sensitive start button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues(start, args, i);
/*
 *  insensitive stop button
 */
/* Commented out by Y.Obayashi 20040430
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(stop, args, i);
*/
/*
 *  sensitive text fields
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues(num_field, args, i);
  XtSetValues(leader_field, args, i);
  XtSetValues(member_field, args, i);
  XtSetValues(comment_field, args, i);
  XtSetValues(run_mode, args, i);
  XtSetValues(threshold_field, args, i);
  for(j=0; j<8; j++)
    XtSetValues(master_field[j], args, i);
/* Bug-fix: following 4 lines are added by Y.Obayashi 20040702 */
```

```
  for(j=0; j<NUM_SERV; j++)
    XtSetValues(server[j], args, i);
  for(j=0; j<TRIGGER_NUM+TRIGGER_NUM_EXTRA; j++)
    XtSetValues(trigger_select[j], args, i);
/*
 *  set new run number
 */
  sprintf(msg, "%d", shm->run_num);
  i = 0;
  XtSetArg(args[i], XmNvalue, msg); i++;
  XtSetValues(num_field, args, i);
  flag_freeze = TRUE;
  return;
}

void ClearEnvCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: clear button pushed\n", ctime(&ptime));
  fflush(fd);
#endif DEBUG
/*
 *  disabel double operation
 */
  if(shm->flag_status == CNTL_CLEAR) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: clear has already done.\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
  }
/*
 *  insensitive clear button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(w, args, i);
/*
 * set command message
 */
  strncpy(buffer->mtext, "clear", 5);
/*
 *  Just wait 480 seconds for ntsrvit and SKITTER
 */
/*  sleep(480); */
/*
 * send message with type 1
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed. %s.\n", ctime(&ptime),
              strerror(errno));
```

```
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 * read answer
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check answer (we must install new lines)
 */

/*
 *  normal end
 */
  exit(0);
}

void RunNumberCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  sscanf(XmTextFieldGetString(w), "%d", &(shm->run_num));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void LeaderCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  strcpy(shm->shift_leader, XmTextFieldGetString(w));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void MemberCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  strcpy(shm->shift_member, XmTextFieldGetString(w));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
```

```
  flag_freeze = FALSE;
}

void CommentCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  strcpy(shm->run_title, XmTextFieldGetString(w));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void MaxRecordCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  sscanf(XmTextFieldGetString(w), "%d", &(shm->max_record_block));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void RecordPathCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  strcpy(shm->record_path_name, XmTextFieldGetString(w));
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void AtmThreshCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int tmp, i;
  sscanf(XmTextFieldGetString(w), "%d", &tmp);
  shm->stab[0].atm_thrdac = (short)(tmp&0xFFFF);
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues((Widget)client_data, args, i);
  flag_freeze = FALSE;
}

void MasterThreshCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int tmp, i, j;
```

```
  u_short val;
  char msg[80];
  if((int)client_data == 2) {
    j = 4;
  } else if((int)client_data == 3) {
    j = 2;
  } else if((int)client_data == 4) {
    j = 3;
  } else {
    j = (int)client_data;
  }
  sscanf(XmTextFieldGetString(w), "%d", &tmp);
  val = (tmp/2);
  if ((int)val*2<DSCR_MIN_LEVEL) val = (u_short)DSCR_MIN_LEVEL/2;
  if ((int)val*2>DSCR_MAX_LEVEL) val = (u_short)DSCR_MAX_LEVEL/2;
  val = val&0xFF;
  sprintf(msg, "%d", val*2);
  i = 0;
  XtSetArg(args[i], XmNvalue, msg); i++;
  XtSetValues(master_field[(int)client_data], args, i);
  shm->master_trigger_threshold[j]
    = (char)val;
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues(master_threshold[(int)client_data], args, i);
  flag_freeze = FALSE;
}

void PedestalCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->flag_pedestal_request = XmToggleButtonGetState(w);
}

void DiskWriteCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->flag_write_possible = XmToggleButtonGetState(w);
}

void HighRateCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->high_rate_switch = XmToggleButtonGetState(w);
}

void RawSaveCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->disk_save_switch = XmToggleButtonGetState(w);
}

void TriggerSelectCB(w, client_data, call_data)
Widget w;
```

```
caddr_t client_data;
caddr_t call_data;
{
  if((int)client_data<TRIGGER_NUM){
    shm->master_trigger_select[(int)client_data]
      = XmToggleButtonGetState(w);
  }else if((int)client_data == TRIGGER_NUM){
    shm->scaler[SCLRNUM-1][SCLR_CHNUM-2]
      = XmToggleButtonGetState(w);
  }

/*  add color map on 22-Oct-2002 */
  XtSetArg(args[0], XmNselectColor, yellow.pixel);
  XtSetValues(w, args, 1);
  XmUpdateDisplay(w);

}

void maskOnCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int server_number, smp_number, i;
  short SET_ON;

  server_number = (int)client_data / 100;
  smp_number = (int)client_data%100;
  SET_ON = 1 << smp_number;
  shm->stab[server_number].smp_mask
    = shm->stab[server_number].smp_mask | SET_ON;
  i = 0;
  sprintf(word, "%s : SMP status %.2x",
            ONL_SRV[server_number], shm->stab[server_number].smp_mask);
  label_string = XmStringCreateLtoR(word, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetValues(head[server_number], args, i);
  XmStringFree(label_string);
}

void maskOffCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int server_number, smp_number, i;
  short SET_ON, BIT;

  server_number = (int)client_data / 100;
  smp_number = (int)client_data%100;
  SET_ON = 1 << smp_number;
  BIT = 0x3f;
  SET_ON = SET_ON ^ BIT;
  shm->stab[server_number].smp_mask
    = shm->stab[server_number].smp_mask & SET_ON;
  i = 0;
  sprintf(word, "%s : SMP status %.2x",
            ONL_SRV[server_number], shm->stab[server_number].smp_mask);
  label_string = XmStringCreateLtoR(word, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetValues(head[server_number], args, i);
  XmStringFree(label_string);
```

```
}

void WSDownCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].f_server = 0;
}

void ServerOnCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].f_server = TKO_SERVER;
}

void ClientOnCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].f_server = XXX_CLIENT;
}

void TriggerOnCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].f_server = TRG_SERVER;
}

void MonitorOnCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].f_server = ONL_MONITOR;
}

void ModeCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i;
  u_short val;
  char msg[80];

  shm->run_mode  = (int)client_data;

  for( i=0; i<TRIGGER_NUM+TRIGGER_NUM_EXTRA; i++) {
    XmToggleButtonSetState(trigger_select[i], FALSE, NULL);
    if(i<TRIGGER_NUM) shm->master_trigger_select[i] = FALSE;
  }
  shm->scaler[SCLRNUM-1][SCLR_CHNUM-2] = 0;

/* change strncpy to memset for initialize title on 11-SEP-2000 */
  memset(shm->run_title, 0, TITLE_NUM);
```

```
  if((int)client_data == NORMAL) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[HIGH_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[HIGH_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[ANTI_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[ANTI_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[NULL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[NULL_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[SUPER_LOW_ENERGY_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[ASSOCIATED_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[ASSOCIATED_TRIGGER] = TRUE;
/* Set origincal SLE thereshold on 05-NOV-1999 */
/* released on 11-SEP-2000 */
    if(old_sle_threshold != 0) {
      val = old_sle_threshold;
    }
    else {
      val = SLE_THRESHOLD;
      old_sle_threshold = val;
    }
    val = val&0xFF;
    sprintf(msg, "%d", val*2);
    i = 0;
    XtSetArg(args[i], XmNvalue, msg); i++;
    XtSetValues(master_field[2], args, i);
    shm->master_trigger_threshold[4] = old_sle_threshold;
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(master_threshold[2], args, i);
    flag_freeze = FALSE;
/* released */
/* add AFTER trigger in NORMAL run mode on 20-Jan-99 */
    XmToggleButtonSetState(trigger_select[AFTER_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[AFTER_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Normal trigger", 14);
  } else if((int)client_data == LASER) {
    XmToggleButtonSetState(trigger_select[LASER_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LASER_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Laser trigger", 13);
  } else if((int)client_data == XE) {
    XmToggleButtonSetState(trigger_select[XE_LAMP_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[XE_LAMP_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Xe trigger", 10);
  } else if((int)client_data == NI) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
/* Add SLE trigger on Ni mode on 9-Mar-98 */
    XmToggleButtonSetState(trigger_select[SUPER_LOW_ENERGY_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[NICKEL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[NICKEL_TRIGGER] = TRUE;
/* Add ANTI_NICKEL_TRIGGER mode on 05-NOV-1999 */
    XmToggleButtonSetState(trigger_select[ANTI_NICKEL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[ANTI_NICKEL_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Ni trigger", 10);
  } else if((int)client_data == RANDOM) {
    XmToggleButtonSetState(trigger_select[RANDOM_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[RANDOM_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Random trigger", 14);
  } else if((int)client_data == LINAC) {
```

```
    XmToggleButtonSetState(trigger_select[LINAC_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LINAC_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[DECAY_E_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[DECAY_E_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
/*     XmToggleButtonSetState(trigger_select[ANTI_TRIGGER], TRUE, NULL);
     shm->master_trigger_select[ANTI_TRIGGER] = TRUE; */
    XmToggleButtonSetState(trigger_select[HIGH_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[HIGH_ENERGY_TRIGGER] = TRUE;
    strncpy(shm->run_title, "LINAC trigger", 13);
  } else if((int)client_data == TEST_RUN) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
/* removed on 6th-Oct-2000
     XmToggleButtonSetState(trigger_select[SUPER_LOW_ENERGY_TRIGGER],TRUE,NULL);
     shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] = TRUE;
*/
    XmToggleButtonSetState(trigger_select[HIGH_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[HIGH_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[ANTI_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[ANTI_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[NULL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[NULL_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Test Run", 8);
/* add decay electron mode  on 10-JUL-1998 Y.F */
  } else if((int)client_data == DECAY_E) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[ANTI_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[ANTI_TRIGGER] = TRUE;
/*     XmToggleButtonSetState(trigger_select[LINAC_TRIGGER], TRUE, NULL);
     shm->master_trigger_select[LINAC_TRIGGER] = TRUE; */
    XmToggleButtonSetState(trigger_select[DECAY_E_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[DECAY_E_TRIGGER] = TRUE;
    strncpy(shm->run_title, "Decay electron Run", 18);
/* add DTcal mode  on 05-NOV-1999 Y.F */
  } else if((int)client_data == DTcal) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[HIGH_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[HIGH_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[SUPER_LOW_ENERGY_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] = TRUE;
    strncpy(shm->run_title, "DT calibration Run", 18);
/* Get origincal thereshold from SLE range on 05-NOV-1999 */
    old_sle_threshold = shm->master_trigger_threshold[4];
    val = SLE_THRESHOLD;
    val = val&0xFF;
    sprintf(msg, "%d", val*2);
    i = 0;
    XtSetArg(args[i], XmNvalue, msg); i++;
    XtSetValues(master_field[2], args, i);
    shm->master_trigger_threshold[4] = (char)val;
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(master_threshold[2], args, i);
    flag_freeze = FALSE;
    strncpy(shm->run_title, "DTcal trigger", 13);
  } else if((int)client_data == DTtrig) {
    XmToggleButtonSetState(trigger_select[LOW_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[LOW_ENERGY_TRIGGER] = TRUE;
```

```
    XmToggleButtonSetState(trigger_select[HIGH_ENERGY_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[HIGH_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[SUPER_LOW_ENERGY_TRIGGER],TRUE,NULL);
    shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[NICKEL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[NICKEL_TRIGGER] = TRUE;
    XmToggleButtonSetState(trigger_select[ANTI_NICKEL_TRIGGER], TRUE, NULL);
    shm->master_trigger_select[ANTI_NICKEL_TRIGGER] = TRUE;
    strncpy(shm->run_title, "DT trigger Run", 14);
/* Get origincal thereshold from SLE range on 05-NOV-1999 */
    old_sle_threshold = shm->master_trigger_threshold[4];
/* old value for 222mV 7-Mar-2000
     val = 111;
*/
/* This is new thresshold value for 212 mV on 7-Mar-2000 */
/*     val = 106;   */
/* now 186 mV on 1-Oct-2000 */
/*     val = 93;    */
    val = SLE_THRESHOLD;
    val = val&0xFF;
    sprintf(msg, "%d", val*2);
    i = 0;
    XtSetArg(args[i], XmNvalue, msg); i++;
    XtSetValues(master_field[2], args, i);
    shm->master_trigger_threshold[4] = (char)val;
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(master_threshold[2], args, i);
    flag_freeze = FALSE;
    strncpy(shm->run_title, "DTtrig trigger", 14);
  }
  i = 0;
  XtSetArg(args[i], XmNvalue, shm->run_title); i++;
  XtSetValues(comment_field, args, i);
/* add color map on 22-Oct-2002 */
  for( i=0; i<TRIGGER_NUM; i++) {
    if(shm->master_trigger_select[i] == TRUE) {
      XtSetArg(args[0], XmNselectColor, yellow.pixel);
      XtSetValues(trigger_select[i], args, 1);
      XmUpdateDisplay(trigger_select[i]);
    }
  }
}

void ElecCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  shm->stab[(int)client_data].electronics_check_flag =
    (u_short)XmToggleButtonGetState(w);
}
PostIt(w, popup, event)
Widget w;
Widget popup;
XButtonEvent *event;
{
  if(event->button != Button3) return;

  XmMenuPosition(popup, event);
  XtManageChild(popup);
}
```

```
void AutoGoCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
/*
 *  This CB includes install/initialize/setup/start proceedure
 */
  int i, j, pid, rval, len, run_num, flag_same_run = FALSE;
  Widget mbox, btn;
  char message[512];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  DIR *dirp;
  struct dirent *direntp;
  char dummy[7];
/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: AutoGoCB start\n", ctime(&ptime));
  fflush(fd);
#endif DEBUG
/*
 *  reset error flag
 */
  f_error = FALSE;
/*
 *  insensitive start button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(start, args, i);
/*
 *  insensitive stop button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(stop, args, i);
  XtSetValues(change, args, i);
/*
 *  select operation
 */
  if(shm->flag_status == CNTL_INSTALL ||
     shm->flag_status == CNTL_ABORT) goto AutoInitialize;
  if(shm->flag_status == CNTL_INITIALIZE ||
     shm->flag_status == CNTL_STOP) goto AutoSetup;
  if(shm->flag_status == CNTL_START ||
     shm->flag_status == CNTL_SETUP) {
/*
 *  sensitive stop button
 */
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(stop, args, i);

    return;
  }
```

```
/*
 *  copy message
 */
  strcpy(message, "Install OK");
/*
 *  disabel double operation
 */
  if(shm->flag_status == CNTL_INSTALL) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: Install has already done.\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    f_error = TRUE;
    strcpy(message, "Install has already done.");
    goto ERRatGo;
  }
/*
 *  add check server switch on 2001-May-8
 */
  for(i=0; i<8; i++) {
    if(f_error == FALSE && shm->stab[i].f_server == 0) {
      for(i=0; i<100; i++) {
         printf("%c", 7);
         fflush(stdout);
         usleep(10000);
      }
      strcpy(message,
               "sukon[1-9] or sukant is OFF.  Is it OK ?");
      f_error = TRUE;
      message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
      title_string =
          (XmString)XmStringCreateLtoR("AutoSetupEnv", XmSTRING_DEFAULT_CHARSET);
      button_string =
          (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
      i = 0;
      XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
      XtSetArg(args[i], XmNmessageString, message_string);  i++;
      XtSetArg(args[i], XmNokLabelString, button_string);  i++;
      XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
      XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
      XtSetArg(args[i], XmNtextFontList, fontlist); i++;
      mbox = XmCreateMessageDialog(w, "mbox", args, i);
      btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
      XtUnmanageChild(btn);
      btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
      XtUnmanageChild(btn);
      i = 0;
      XtSetArg(args[i], XmNdialogType, XmDIALOG_ERROR);
      i++;
      XtSetValues(mbox, args, i);
      XtManageChild(mbox);
      XmStringFree(title_string);
      XmStringFree(message_string);
      XmStringFree(button_string);
    }
  }
/*
 *  set command message
 */
  strncpy(buffer->mtext, "install\0", 8);
/*
```

```
 *  send message to controller
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at install. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  read answer from controller
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at install. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_INSTALL);
/*
 *  create "pmonitor" in window
 */
  if(rval == 0) {
    switch (pid = fork()) {
    case -1:
#ifdef DEBUG
      time(&ptime);
      fprintf(fd, "%.24s :: fork failed. %s.\n", ctime(&ptime),
                strerror(errno));
      fflush(fd);
#endif DEBUG
      break;
    case 0:
      execl("/home/online/daq/pmonitor", "pmonitor", NULL);
      if(errno == ENOENT) {
#ifdef DEBUG
          time(&ptime);
          fprintf(fd, "%.24s :: no empty. %s.\n", ctime(&ptime),
                  strerror(errno));
          fflush(fd);
#endif DEBUG
      }
      break;
    default:
      break;
    }
  }
  else {
    f_error = TRUE;
    strcpy(message, "Install has failed. Clear environment. \n Are you sure ? ");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("clear", XmSTRING_DEFAULT_CHARSET);
```

```
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)ClearEnvCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_CLEAR);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
 AutoInitialize:
/*
 *  copy message
 */
  strcpy(message, "Initialize OK");
/*
 *  disabel double operation
 */
  if(shm->flag_status == CNTL_INITIALIZE) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: initialize has already done.\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    f_error = TRUE;
    strcpy(message, "Initialize has already done.");
    goto ERRatGo;
  }
/*
 *  set command message
 */
  strncpy(buffer->mtext, "initialize\0", 11);
/*
 *  send message to controller
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at initialize. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  read answer from controller
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at initialize. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
```

```
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_INITIALIZE);
/*
 *  set process existence flag
 */
  if(rval == 0) {
    shm->flag_process_exist = TRUE;
  }
  else {
    strcpy(message, "Initialize has failed. Abort DAQ process. \n Are you sure ?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
 AutoSetup:
/*
 *  Copy message
 */
  strcpy(message,"Run started normally");
/*
 *  disable double operation
 */
  if((shm->flag_status == CNTL_START) ||
     (shm->flag_status == CNTL_SETUP)) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: setup/start has already done", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    f_error = TRUE;
    strcpy(message, "Setup/start has already done.");
    goto ERRatGo;
  }
/*
 *  Open the data from summary
 */
  if((dirp = opendir("/home/online/summary")) != NULL) {
/*
 *  read directory
 */
    while((direntp = readdir(dirp)) != NULL) {
      if(strncmp(direntp->d_name, ".", 1) == 0) continue;
      if(strncmp(direntp->d_name, "..", 2) == 0) continue;
      len = strlen(direntp->d_name);
```

```
      if(len < 8) continue;
/*
 *  read run #
 */
      sscanf(direntp->d_name,"%7s%d", dummy, &run_num);
      if(run_num == shm->run_num) {
          flag_same_run = TRUE;
      }
      else
          continue;
    }
/*
 *  Close directory
 */
    closedir(dirp);
  }
/*
 *  Check message
 */
  if(flag_same_run == TRUE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    sprintf(message,
              "Run %d was used before. You must change Run number !!",
              shm->run_num);
    f_error = TRUE;
    goto ERRatGo;
  }
/*
 *  set command message
 */
  strncpy(buffer->mtext, "setup\0", 6);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at setup. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  read answer
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at setup. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
```

```
 */
  rval = msgchk(message, CNTL_SETUP);
/*
 *  normal message
 */
  if(rval != 0) {
    f_error = TRUE;
    strcpy(message, "Setup has failed. Abort DAQ process. \n Are you sure ?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
/*
 *  clear run end message
 */
  memset(shm->run_end_message, 0, MESSAGE_NUM);
/*
 *  insensitive text fields
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(num_field, args, i);
  XtSetValues(leader_field, args, i);
  XtSetValues(member_field, args, i);
  XtSetValues(comment_field, args, i);
  XtSetValues(run_mode, args, i);
  XtSetValues(threshold_field, args, i);
  for(j=0; j<8; j++)
    XtSetValues(master_field[j], args, i);
  for(j=0; j<NUM_SERV; j++)
    XtSetValues(server[j], args, i);
  for(j=0; j<TRIGGER_NUM+TRIGGER_NUM_EXTRA; j++)
    XtSetValues(trigger_select[j], args, i);
/*
 *  set command message
 */
  strncpy(buffer->mtext, "start\n", 6);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at start. %s.\n", ctime(&ptime),
             strerror(errno));
    fflush(fd);
```

```
#endif DEBUG
    exit(1);
  }
/*
 *  read answer
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at start. %s.\n", ctime(&ptime),
             strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_START);
/*
 *  normal return
 */
  if(rval != 0) {
    strcpy(message, "Run start has failed. Abort DAQ process. \n Are you sure ?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
/*
 *  reaching point
 */
 ERRatGo:
/*
 *  create status message board
 */
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("AutoGo", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
```

```
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(btn);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(btn);
  i = 0;
  if(f_error)
    XtSetArg(args[i], XmNdialogType, XmDIALOG_ERROR);
  else
    XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);
  i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
/*
 *  sensitive start button in case of error
 */
  if(f_error) {
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(start, args, i);
/* removed to avoid stop button disappearing problem. Y.Obayashi, 20040806
    i = 0;
    XtSetArg(args[i], XmNsensitive, FALSE); i++;
    XtSetValues(stop, args, i);
  YO20040806 */
  } else {
    i = 0;
    XtSetArg(args[i], XmNsensitive, FALSE); i++;
    XtSetValues(start, args, i);
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(stop, args, i);
  }
  return;
}

void DownCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i, j, rval;
  int timeout = 0,sec = 0;
  char msg[10];
  Widget mbox, pbox, btn, pbtn;
  char message[512];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  DIR *dirp;
  struct dirent *direntp;
  char dummy[7];
  int run_num, len, flag_same_run = FALSE;
/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: stop button pushed\n", ctime(&ptime));
```

```
  fflush(fd);
#endif DEBUG
/*
 *  disable double operation
 */
  if(shm->flag_status == CNTL_STOP) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: stop has already done\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    f_error = TRUE;
    strcpy(message, "stop has already done");
    goto ERRatDWN;
  }
/*
 *  reset error flag
 */
  f_error = FALSE;
/*
 *  copy runend message
 */
  strcpy(shm->run_end_message, XmTextGetString(SB_Text(w)));
/*
 *  insensitive stop button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(stop, args, i);
/*
 *  set command message
 */
  strncpy(buffer->mtext, "stop\0", 5);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 * set timer
 */
  sigemptyset(&act.sa_mask);
  act.sa_flags = SA_NOCLDWAIT;
  sigaction(SIGCHLD, &act, NULL);
  signal(SIGALRM, (void (*)(int))handler);
  alarm(300);
/*
 *  create status message board
 */
/*
  sprintf(message,"Data daking still continue. Please wait...");
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
     (XmString)XmStringCreateLtoR("wait", XmSTRING_DEFAULT_CHARSET);
```

```
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);i++;
  pbox = XmCreateMessageDialog(w, "pbox", args, i);
  pbtn = XmMessageBoxGetChild(pbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(pbtn);
  pbtn = XmMessageBoxGetChild(pbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(pbtn);
  XtManageChild(pbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
*/
/*
 *  local loop
 */
  while (1) {
/*
 *  read answer
 */
    if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
      time(&ptime);
      fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
                strerror(errno));
      fflush(fd);
#endif DEBUG
      exit(1);
    }
/*
 *  reset alarm
 */
    alarm(0);
    timeout ++;
/*
 *  check message at first
 */
    if(strncmp(buffer->mtext,"status",6) == 0) {
      sec = 400 - timeout*10;
      if(timeout%4 == 0) sprintf(message,"status: | Please wait %d sec",sec);
      if(timeout%4 == 1) sprintf(message,"status: / Please wait %d sec",sec);
      if(timeout%4 == 2) sprintf(message,"status: – Please wait %d sec",sec);
      if(timeout%4 == 3) sprintf(message,"status: \\ Please wait %d sec",sec);
/*
      message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
      i=0;
      XtSetArg(args[i], XmNlabelString, message_string);  i++;
      XtSetValues(sel_status, args, i);
      XmStringFree(message_string);
*/
    } else
      break;
  }
/*
 *  check returned message
```

```
 */
  rval = msgchk(message, CNTL_STOP);
/*
 *  normal return
 */
  if(rval == 0)
    strncpy(message, "Run has stopped succesfully\0", 29);
  else {
    f_error = TRUE;
    strcpy(message, "Run stopped abnormally. Abort process. \n Are you sure?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
/*
 *  reaching point
 */
 ERRatDWN:
/*
 *  create status message board
 */
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("stop", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(btn);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(btn);
  i = 0;
  if(f_error)
    XtSetArg(args[i], XmNdialogType, XmDIALOG_ERROR);
  else
    XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);
  i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
  XmStringFree(title_string);
```

```
  XmStringFree(message_string);
  XmStringFree(button_string);
/*
 *  sensitive text fields
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues(num_field, args, i);
  XtSetValues(leader_field, args, i);
  XtSetValues(member_field, args, i);
  XtSetValues(comment_field, args, i);
  XtSetValues(run_mode, args, i);
  XtSetValues(threshold_field, args, i);
  for(j=0; j<8; j++)
    XtSetValues(master_field[j], args, i);
  for(j=0; j<NUM_SERV; j++)
    XtSetValues(server[j], args, i);
  for(j=0; j<TRIGGER_NUM+TRIGGER_NUM_EXTRA; j++)
    XtSetValues(trigger_select[j], args, i);
/*
 *  Set new run number
 */
  sprintf(msg, "%d", shm->run_num);
  i = 0;
  XtSetArg(args[i], XmNvalue, msg); i++;
  XtSetValues(num_field, args, i);
  flag_freeze = TRUE;
/*
 *  insensitive stop button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(stop, args, i);
/*
 *  sensitive start button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, TRUE); i++;
  XtSetValues(start, args, i);
/*
 *  normal return
 */
  return;
}


void StopStartCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  int i, j, rval;
  int timeout = 0,sec = 0;
  char msg[10];
  Widget mbox, pbox, btn, pbtn;
  char message[512];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  DIR *dirp;
  struct dirent *direntp;
  char dummy[7];
```

```
  int run_num, len, flag_same_run = FALSE;
/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: RunChange button pushed\n", ctime(&ptime));
  fflush(fd);
#endif DEBUG
/*
 *  disable double operation
 */
  if(shm->flag_status == CNTL_STOP) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: stop has already done\n", ctime(&ptime));
    fflush(fd);
#endif DEBUG
    f_error = TRUE;
    strcpy(message, "stop has already done.");
    goto ERRatCHG;
  }
/*
 *  reset error flag
 */
  f_error = FALSE;
/*
 *  copy runend message
 */
  strcpy(shm->run_end_message, XmTextGetString(SB_Text(w)));
/*
 *  insensitive stop button
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(stop, args, i);
  XtSetValues(change, args, i);
/*
 *  set command message
 */
  strncpy(buffer->mtext, "stop\0", 5);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 * set timer
 */
  sigemptyset(&act.sa_mask);
  act.sa_flags = SA_NOCLDWAIT;
  sigaction(SIGCHLD, &act, NULL);
  signal(SIGALRM, (void (*)(int))handler);
  alarm(300);
```

```
/*
 *  create status message board
 */
/*
  sprintf(message,"Data taking still continue. Please wait...");
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("wait", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);i++;
  pbox = XmCreateMessageDialog(w, "pbox", args, i);
  pbtn = XmMessageBoxGetChild(pbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(pbtn);
  pbtn = XmMessageBoxGetChild(pbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(pbtn);
  XtManageChild(pbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
*/
/*
 *  local loop
 */
 while (1) {
/*
 *  read answer
 */
   if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
     time(&ptime);
     fprintf(fd, "%.24s :: msgrcv failed. %s.\n", ctime(&ptime),
             strerror(errno));
     fflush(fd);
#endif DEBUG
     exit(1);
   }
/*
 *  reset alarm
 */
   alarm(0);
   timeout ++;
/*
 *  check message at first
 */
   if(strncmp(buffer->mtext,"status",6) == 0) {
     sec = 400 - timeout*10;
     if(timeout%4 == 0) sprintf(message,"status: | Please wait %d sec",sec);
     if(timeout%4 == 1) sprintf(message,"status: / Please wait %d sec",sec);
     if(timeout%4 == 2) sprintf(message,"status: – Please wait %d sec",sec);
     if(timeout%4 == 3) sprintf(message,"status: \\ Please wait %d sec",sec);
/*
      message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
      i=0;
      XtSetArg(args[i], XmNlabelString, message_string);  i++;
```

```
      XtSetValues(sel_status, args, i);
      XmStringFree(message_string);
*/
    } else
      break;
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_STOP);
/*
 *  normal return
 */
  if(rval == 0)
    strncpy(message, "Run has stopped succesfully\0", 29);
  else {
    f_error = TRUE;
    strcpy(message, "Run stopped abnormally. Abort process. \n Are you sure?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }

/***********************************************
 * Run stopped successfully, then start next run
 ***********************************************/

/*
 *  Set new run number
 */
  sprintf(msg, "%d", shm->run_num);
  i = 0;
  XtSetArg(args[i], XmNvalue, msg); i++;
  XtSetValues(num_field, args, i);
  flag_freeze = FALSE;

/*
 *  log file
 */
#ifdef DEBUG
  time(&ptime);
  fprintf(fd, "%.24s :: AutoChangeCB stop succeed and trying to start next run # %d\n", ctime(&ptime)
,shm->run_num);
  fflush(fd);
#endif DEBUG

/*
```

```
 *  Copy message
 */
  strcpy(message,"Run started normally");
/*
 *  Open the data from summary
 */
  if((dirp = opendir("/home/online/summary")) != NULL) {
/*
 *  read directory
 */
    while((direntp = readdir(dirp)) != NULL) {
      if(strncmp(direntp->d_name, ".", 1) == 0) continue;
      if(strncmp(direntp->d_name, "..", 2) == 0) continue;
      len = strlen(direntp->d_name);
      if(len < 8) continue;
/*
 *  read run #
 */
      sscanf(direntp->d_name,"%7s%d", dummy, &run_num);
      if(run_num == shm->run_num) {
        flag_same_run = TRUE;
      }
      else
        continue;
    }
/*
 *  Close directory
 */
    closedir(dirp);
  }
/*
 *  Check message
 */
  if(flag_same_run == TRUE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    sprintf(message,
            "Run %d was used before. You must change Run number !!",
            shm->run_num);
    f_error = TRUE;
    goto ERRatCHG;
  }
/*
 *  set command message
 */
  strncpy(buffer->mtext, "setup\0", 6);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at setup. %s.\n", ctime(&ptime),
            strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
```

```
/*
 *  read answer
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at setup. %s.\n", ctime(&ptime),
            strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_SETUP);
/*
 *  normal message
 */
  if(rval != 0) {
    f_error = TRUE;
    strcpy(message, "Setup has failed. Abort DAQ process. \n Are you sure ?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
/*
 *  clear run end message
 */
  memset(shm->run_end_message, 0, MESSAGE_NUM);
/*
 *  insensitive text fields
 */
  i = 0;
  XtSetArg(args[i], XmNsensitive, FALSE); i++;
  XtSetValues(num_field, args, i);
  XtSetValues(leader_field, args, i);
  XtSetValues(member_field, args, i);
  XtSetValues(comment_field, args, i);
  XtSetValues(run_mode, args, i);
  XtSetValues(threshold_field, args, i);
  for(j=0; j<8; j++)
    XtSetValues(master_field[j], args, i);
  for(j=0; j<NUM_SERV; j++)
    XtSetValues(server[j], args, i);
  for(j=0; j<TRIGGER_NUM+TRIGGER_NUM_EXTRA; j++)
    XtSetValues(trigger_select[j], args, i);
```

```c
/*
 *  set command message
 */
  strncpy(buffer->mtext, "start\n", 6);
/*
 *  send message with type SNDTYP
 */
  buffer->mtype = SNDTYP;
  if(msgsnd(msgid, buffer, MSGLEN, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgsnd failed at start. %s.\n", ctime(&ptime),
            strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  read answer
 */
  if(msgrcv(msgid, buffer, MSGLEN, RCVTYP, 0) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgrcv failed at start. %s.\n", ctime(&ptime),
            strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  check returned message
 */
  rval = msgchk(message, CNTL_START);
/*
 *  normal return
 */
  if(rval != 0) {
    strcpy(message, "Run start has failed. Abort DAQ process. \n Are you sure ?");
    message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
    title_string =
      (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
    i = 0;
    XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
    XtSetArg(args[i], XmNmessageString, message_string);  i++;
    XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
    XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
    XtSetArg(args[i], XmNtextFontList, fontlist); i++;
    mbox = XmCreateMessageDialog(w, "mbox", args, i);
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
    XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL
_ABORT);
    i = 0;
    XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
    XtSetValues(mbox, args, i);
    XtManageChild(mbox);
    return;
  }
/*
 *  reaching point
 */
 ERRatCHG:
/*
```

```c
 *  create status message board
 */
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("RunChange", XmSTRING_DEFAULT_CHARSET);
  button_string =
    (XmString)XmStringCreateLtoR("close", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNokLabelString, button_string);  i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_CANCEL_BUTTON);
  XtUnmanageChild(btn);
  btn = XmMessageBoxGetChild(mbox, XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(btn);
  i = 0;
  if(f_error)
    XtSetArg(args[i], XmNdialogType, XmDIALOG_ERROR);
  else
    XtSetArg(args[i], XmNdialogType, XmDIALOG_INFORMATION);
  i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
  XmStringFree(title_string);
  XmStringFree(message_string);
  XmStringFree(button_string);
/*
 *  sensitive start button in case of error
 */
  if(f_error) {
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(start, args, i);
  } else {
    i = 0;
    XtSetArg(args[i], XmNsensitive, FALSE); i++;
    XtSetValues(start, args, i);
    i = 0;
    XtSetArg(args[i], XmNsensitive, TRUE); i++;
    XtSetValues(stop, args, i);
  }

/*
 *  normal return
 */
  return;
}



void AbortCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  Widget mbox, btn;
  int i, run_num, len, flag_same_run = FALSE;
  char message[132];
```

```
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  DIR *dirp;
  struct dirent *direntp;
  char dummy[7];
/*
 *  Open the data from summary
 */
  if((dirp = opendir("/home/online/summary")) != NULL) {
/*
 *  read directory
 */
    while((direntp = readdir(dirp)) != NULL) {
      if(strncmp(direntp->d_name, ".", 1) == 0) continue;
      if(strncmp(direntp->d_name, "..", 2) == 0) continue;
      len = strlen(direntp->d_name);
      if(len < 8) continue;
/*
 *  read run #
 */
      sscanf(direntp->d_name,"%7s%d", dummy, &run_num);
      if(run_num == shm->run_num) {
          flag_same_run = TRUE;
      }
      else
          continue;
    }
/*
 *  Close directory
 */
    closedir(dirp);
  }
/*
 *  Check message
 */
  if(flag_same_run == TRUE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    sprintf(message,
            "Summary file for Run %d has already existed.\n Is it OK to overwrite it ?",
             shm->run_num);
  }
  else
    strcpy(message,
           "Abort DAQ process. Check On-line Data Aquisition Rate Monitor!!!\n If average rate is not zero, dat
a are transfering to sukonh. \n Are you sure ?");
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("abort", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AbortProcCB, NULL);
  XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL_A
```

```
BORT);
  i = 0;
  XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
}

void AutoStartCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  Widget mbox, btn;
  int i, run_num;
  char message[132];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
  FILE *fp;
  char line[80];
/*
 * in case of install
 */
  if(shm->flag_status == CNTL_CLEAR)
    strcpy(message, "Satrt from INSTALL procedure. Are you ready ? ");
/*
 * in case of initialize
 */
  else if(shm->flag_status == CNTL_INSTALL ||
            shm->flag_status == CNTL_ABORT)
    strcpy(message,
             " Start from initialize DAQ process. Are you ready ? ");
/*
 * in case of initialize
 */
  else if(shm->flag_status == CNTL_INITIALIZE ||
            shm->flag_status == CNTL_STOP)
    strcpy(message,
             " Start from setup DAQ process. Are you ready ? ");


/* add check server switch on 2001-May-8 */
  for(i=0; i<8; i++) {
    if(shm->stab[i].f_server == 0) {
      for(i=0; i<100; i++) {
          printf("%c", 7);
          fflush(stdout);
          usleep(10000);
      }
      strcpy(message,
               "sukon[1-9] or sukant is OFF.  Is it OK ?");
    }
  }
/*
 *  Check run number
 */
  fp = fopen("/home/online/EVNT_TBL_FILE", "r");
  while(fgets(line, MAX_LINE, fp) != NULL) {
    sscanf(line+24, "%.6d", &run_num);
    if(shm->run_num == run_num) {
      flag_save_run = TRUE;
      break;
```

```
      }
  }
  fclose(fp);
/*
 *  Check message
 */
  if(shm->flag_write_possible == FALSE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    strcpy(message,
             "Data will not be saved into disk!!! \n Is that OK ???");
  }
  if(shm->flag_pedestal_request == FALSE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    strcpy(message,
             "Nothing to take pedestal data !!!. \n Is that OK ???.");
  }
  if(shm->high_rate_switch == FALSE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    strcpy(message,
             "High rate switch has not set !!!. \n Is that OK ???");
  }
  if(shm->disk_save_switch == FALSE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    strcpy(message,
             "Raw data save switch has not set !! \n Is that OK ??");
  }
  for(i=0; i<8; i++) {
    if(shm->stab[i].electronics_check_flag == FALSE) {
      for(i=0; i<100; i++) {
          printf("%c", 7);
          fflush(stdout);
          usleep(10000);
      }
      strcpy(message,
               "Electronics check will not be done !! \n Please replace broken ATM bord after you are ready. \n R
un will start. Are you ready ?");
    }
    if(shm->stab[i].smp_mask != 0) {
      for(i=0; i<100; i++) {
          printf("%c", 7);
          fflush(stdout);
          usleep(10000);
      }
      strcpy(message,
               "SMP mask has been set !! All lemo cables from \n corresponding SMP must be pulled out. Are yo
u sure ?");
```

```
    }
  }
  if(flag_save_run == TRUE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    sprintf(message,
              "Data for Run %d have already stored and sent OFFLINE. \n Cancel !!! ",
              shm->run_num);
  }
  if(flag_freeze == TRUE) {
    for(i=0; i<100; i++) {
      printf("%c", 7);
      fflush(stdout);
      usleep(10000);
    }
    strcpy(message,
             "Confirm selected parameters by pressing Retrun key !!!");
  }
  message_string = XmStringCreateLtoR(message, XmSTRING_DEFAULT_CHARSET);
  title_string =
    (XmString)XmStringCreateLtoR("install", XmSTRING_DEFAULT_CHARSET);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNmessageString, message_string);  i++;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNbuttonFontList, fontlist); i++;
  XtSetArg(args[i], XmNtextFontList, fontlist); i++;
  mbox = XmCreateMessageDialog(w, "mbox", args, i);
  if((flag_save_run == FALSE) && (flag_freeze == FALSE))
    XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)AutoGoCB, NULL);
  XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB,
                    (XtPointer)CNTL_INSTALL);
  i = 0;
  XtSetArg(args[i], XmNdialogType, XmDIALOG_QUESTION); i++;
  XtSetValues(mbox, args, i);
  XtManageChild(mbox);
  return;
}

void StopCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  Widget mbox, btn;
  int i;
  char message[132];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString text_string = NULL;
/*
 *  Stop with run end statement
 */
  strcpy(message, "Stop present Run with run end message. Are you sure ?");
  message_string = XmStringCreateLtoR(message, XmFONTLIST_DEFAULT_TAG);
  title_string =
    (XmString)XmStringCreateLtoR("stop", XmFONTLIST_DEFAULT_TAG);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
```

```
  XtSetArg(args[i], XmNselectionLabelString, message_string);  i++;
  mbox = (Widget)XmCreatePromptDialog(w, "mbox", args, i);
  XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)DownCB, NULL);
  XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL_S
TOP);
  XtManageChild(mbox);
  return;
}

void RunChangeCB(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
  Widget mbox, btn;
  int i;
  char message[132];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString text_string = NULL;
/*
 *  Stop and Start Next run
 */
  strcpy(message, "Stop present Run and Start Next Run. Are you sure ?");
  message_string = XmStringCreateLtoR(message, XmFONTLIST_DEFAULT_TAG);
  title_string =
    (XmString)XmStringCreateLtoR("Run Change", XmFONTLIST_DEFAULT_TAG);
  i = 0;
  XtSetArg(args[i], XmNdialogTitle, title_string);  i++;
  XtSetArg(args[i], XmNselectionLabelString, message_string);  i++;
  mbox = (Widget)XmCreatePromptDialog(w, "mbox", args, i);
  XtAddCallback(mbox, XmNokCallback, (XtCallbackProc)StopStartCB, NULL);
  XtAddCallback(mbox, XmNhelpCallback, (XtCallbackProc)HelpCB, (XtPointer)CNTL_S
TOP);
  XtManageChild(mbox);
  return;
}

/*
 *  main
 */
main(argc, argv)
int argc;
char *argv[];
{
  XtAppContext app_context;
  XmString s[4];
  int i, j, k;
  char msg[80], stat[10];
  key_t key;
  int shmflg;
  int shmid;
  int status, mode, np, significance;
  int rval, msgflg;
  int f_initialize = FALSE, f_setup = FALSE, f_start = FALSE, f_stop = FALSE;
  int f_abort = FALSE, f_clear = FALSE, f_install = FALSE;
/*
 * Open local colsole for local message
 */
#ifdef DEBUG
  fd = fopen("/home/online/log/selector.log", "a+");
  if (fd == NULL) exit(1);
```

```
#endif DEBUG
/*
 *  Map shared memory
 */
  shm = (struct shmdef *)easy_init(NULL);
/*
 *  Initialize semaphore
 */
  if ((sem_id = semget((key_t)KEY, 1, PERM|IPC_CREAT))==-1) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: semget failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  Attachment shared memory
 */
  key = (key_t)KEY_ABORT;
  shmflg = 0666 | IPC_CREAT;
  if((shmid = shmget(key, 400, shmflg)) == -1) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: shmget failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
  abo = 0;
  shmflg = 020000;
  abo = (struct pidtab *)shmat(shmid, abo, shmflg);
/*
 *  allocate message buffer area
 */
  key = (key_t)KEY_CTRL;
  msgflg = 0666 | IPC_CREAT;
  if((msgid = msgget(key, msgflg)) < 0){
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: msgget failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
  buffer = (struct msgbuf *)malloc(sizeof(long) + MSGLEN);
  if((int)buffer == -1) {
#ifdef DEBUG
    time(&ptime);
    fprintf(fd, "%.24s :: malloc failed. %s.\n", ctime(&ptime),
              strerror(errno));
    fflush(fd);
#endif DEBUG
    exit(1);
  }
/*
 *  Initialize X toolkit
 */
  toplevel = XtAppInitialize(&app_context, "controller", NULL, NULL,
```

```
                                &argc, argv, NULL, NULL, 0);
/*
 *  Install font list
 */
 font_list_entry =
/*    XmFontListEntryLoad(XtDisplay(mainw), font1, XmFONT_IS_FONT, "font_tag");
*/
   XmFontListEntryLoad(XtDisplay(toplevel), font1, XmFONT_IS_FONT,
                          XmFONTLIST_DEFAULT_TAG);
 fontlist = XmFontListAppendEntry(NULL, font_list_entry);
/*
 *  Create main window
 */
 mainw = (Widget)XmCreateMainWindow((Widget)toplevel, "main", NULL, NULL);
 XtManageChild(mainw);
/*
 *  Setup frame in main window
 */
 i = 0;
 XtSetArg(args[i], XmNshadowThickness, 1); i++;
 XtSetArg(args[i], XmNshadowType, XmSHADOW_OUT); i++;
 frame = XmCreateFrame(mainw, "frame", args, i);
 XtManageChild(frame);
/*
 *  Create paned window
 */
 i = 0;
 XtSetArg(args[i], XmNseparatorOn, TRUE); i++;
 XtSetArg(args[i], XmNallowResize, TRUE); i++;
 panedw = XmCreatePanedWindow(frame, "panedw", args, i);
 XtManageChild(panedw);
/*
 * Create color map
 */
 dsp = XtDisplay(panedw);
 cmap = DefaultColormap(dsp, DefaultScreen(dsp));
 result = XAllocNamedColor(dsp, cmap, "Yellow", &exact, &yellow);
/*
 *  Create Form 1
 */
 i = 0;
 /*  XtSetArg(args[i], XmNheight, 365); i++;*/
 XtSetArg(args[i], XmNheight, 200); i++;
 form1 = XmCreateForm(panedw, "form1", args, i);
 XtManageChild(form1);
/*
 *  Create row-column
 */
 i = 0;
 XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
 XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
 XtSetArg(args[i], XmNnumColumns, 16); i++;
 setup = XmCreateRowColumn(form1, "setup", args, i);
 XtManageChild(setup);
 XmAddTabGroup(setup);
/*
 *  Create label and text field for run number
 */
 i = 0;
 label_string = XmStringCreateLtoR("run number :", XmSTRING_DEFAULT_CHARSET);
 XtSetArg(args[i], XmNfontList, fontlist); i++;
 XtSetArg(args[i], XmNlabelString, label_string); i++;
```

```
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  run_number = XmCreateLabel(setup, "run-number", args, i);
  XtManageChild(run_number);
  XmStringFree(label_string);
  i = 0;
  sprintf(msg, "%d", shm->run_num);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 5); i++;
  XtSetArg(args[i], XmNcolumns, 5); i++;
  XtSetArg(args[i], XmNvalue, msg); i++;
  num_field = XmCreateTextField(setup, "num_field", args, i);
  XtAddCallback(num_field, XmNactivateCallback, (XtCallbackProc)RunNumberCB, (Xt
Pointer)run_number);
  XtAddCallback(num_field, XmNmodifyVerifyCallback, (XtCallbackProc)FreezeCB, (X
tPointer)run_number);
  XtManageChild(num_field);
/*
 *  Create label and text field for shift leader
 */
  i = 0;
  label_string = XmStringCreateLtoR("shift leader :",
                                     XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  leader = XmCreateLabel(setup, "leader", args, i);
  XtManageChild(leader);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 20); i++;
  XtSetArg(args[i], XmNvalue, shm->shift_leader); i++;
  leader_field = XmCreateTextField(setup, "leader_field", args, i);
  XtAddCallback(leader_field, XmNactivateCallback, (XtCallbackProc)LeaderCB, (Xt
Pointer)leader);
  XtAddCallback(leader_field, XmNmodifyVerifyCallback, (XtCallbackProc)FreezeCB,
 (XtPointer)leader);
  XtManageChild(leader_field);
/*
 *  Create label and text field for shift member
 */
  i = 0;
  label_string = XmStringCreateLtoR("shift member :",
                                     XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  member = XmCreateLabel(setup, "member", args, i);
  XtManageChild(member);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 50); i++;
  XtSetArg(args[i], XmNvalue, shm->shift_member); i++;
  member_field = XmCreateTextField(setup, "member_field", args, i);
  XtAddCallback(member_field, XmNactivateCallback, (XtCallbackProc)MemberCB, (Xt
```

```
Pointer)member);
  XtAddCallback(member_field, XmNmodifyVerifyCallback, (XtCallbackProc)FreezeCB,
 (XtPointer)member);
  XtManageChild(member_field);
/*
 *  Create text field for run comment
 */
  i = 0;
  label_string = XmStringCreateLtoR("comment :", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  comment = XmCreateLabel(setup, "comment", args, i);
  XtManageChild(comment);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 50); i++;
  XtSetArg(args[i], XmNvalue, shm->run_title); i++;
  comment_field = XmCreateTextField(setup, "comment_field", args, i);
  XtAddCallback(comment_field, XmNactivateCallback, (XtCallbackProc)CommentCB, (
XtPointer)comment);
  XtAddCallback(comment_field, XmNmodifyVerifyCallback, (XtCallbackProc)FreezeCB
, (XtPointer)comment);
  XtManageChild(comment_field);
/*
 *  Create pulldown menu for run mode
 */
  i = 0;
  label_string = XmStringCreateLtoR("Run mode :", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  mode_label = XmCreateLabel(setup, "mode_label", args, i);
  XtManageChild(mode_label);
  XmStringFree(label_string);

  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  run_mode = XmCreatePulldownMenu(setup, "run_mode", args, i);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’1’); i++;
  popupBtn[5] = XmCreatePushButtonGadget(run_mode, "normal", args, i);
  XtAddCallback(popupBtn[5], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPoi
nter)NORMAL);
  XtManageChild(popupBtn[5]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’2’); i++;
  popupBtn[6] = XmCreatePushButtonGadget(run_mode, "laser", args, i);
  XtAddCallback(popupBtn[6], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPoi
nter)LASER);
  XtManageChild(popupBtn[6]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’3’); i++;
  popupBtn[7] = XmCreatePushButtonGadget(run_mode, "Xe", args, i);
  XtAddCallback(popupBtn[7], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPoi
nter)XE);
```

```
  XtManageChild(popupBtn[7]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’4’); i++;
  popupBtn[8] = XmCreatePushButtonGadget(run_mode, "Ni", args, i);
  XtAddCallback(popupBtn[8], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPoi
nter)NI);
  XtManageChild(popupBtn[8]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’5’); i++;
  popupBtn[9] = XmCreatePushButtonGadget(run_mode, "random", args, i);
  XtAddCallback(popupBtn[9], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPoi
nter)RANDOM);
  XtManageChild(popupBtn[9]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’6’); i++;
  popupBtn[10] = XmCreatePushButtonGadget(run_mode, "LINAC", args, i);
  XtAddCallback(popupBtn[10], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPo
inter)LINAC);
  XtManageChild(popupBtn[10]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’7’); i++;
  popupBtn[11] = XmCreatePushButtonGadget(run_mode, "test", args, i);
  XtAddCallback(popupBtn[11], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPo
inter)TEST_RUN);
  XtManageChild(popupBtn[11]);
/* add decay e mode on 10-JUL-1998 Y.F */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’8’); i++;
  popupBtn[12] = XmCreatePushButtonGadget(run_mode, "decay e", args, i);
  XtAddCallback(popupBtn[12], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPo
inter)DECAY_E);
  XtManageChild(popupBtn[12]);
/* add DTcal and DTtrig mode on 05-Nov-1999 Y.F */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’9’); i++;
  popupBtn[13] = XmCreatePushButtonGadget(run_mode, "DTcal", args, i);
  XtAddCallback(popupBtn[13], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPo
inter)DTcal);
  XtManageChild(popupBtn[13]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNmnemonic, ’10’); i++;
  popupBtn[14] = XmCreatePushButtonGadget(run_mode, "DTtrig", args, i);
  XtAddCallback(popupBtn[14], XmNactivateCallback, (XtCallbackProc)ModeCB, (XtPo
inter)DTtrig);
  XtManageChild(popupBtn[14]);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  if(shm->run_mode == NORMAL) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[5]); i++;
  } else if(shm->run_mode == LASER) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[6]); i++;
  } else if(shm->run_mode == XE) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[7]); i++;
  } else if(shm->run_mode == NI) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[8]); i++;
```

```
  } else if(shm->run_mode == RANDOM) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[9]); i++;
  } else if(shm->run_mode == LINAC) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[10]); i++;
  } else if(shm->run_mode == TEST_RUN) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[11]); i++;
  } else if(shm->run_mode == DECAY_E) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[12]); i++;
  } else if(shm->run_mode == DTcal) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[13]); i++;
  } else if(shm->run_mode == DTtrig) {
    XtSetArg(args[i], XmNmenuHistory, popupBtn[14]); i++;
  }
  XtSetArg(args[i], XmNmnemonic, ’1’); i++;
  XtSetArg(args[i], XmNsubMenuId, run_mode); i++;
  run_type = XmCreateOptionMenu(setup, "run_type", args, i);
  XtManageChild(run_type);
/*
 *  Create text field for ATM threshold
 */
  i = 0;
  label_string = XmStringCreateLtoR("ATM threshold :",
                                    XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  atm_threshold = XmCreateLabel(setup, "atm_threshold", args, i);
  XtManageChild(atm_threshold);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 4); i++;
  sprintf(msg, "%d", shm->stab[0].atm_thrdac);
  XtSetArg(args[i], XmNvalue, msg); i++;
  threshold_field = XmCreateTextField(setup, "num_field", args, i);
  XtAddCallback(threshold_field, XmNactivateCallback, (XtCallbackProc)AtmThreshC
B, (XtPointer)atm_threshold);
  XtAddCallback(threshold_field, XmNmodifyVerifyCallback, (XtCallbackProc)Freeze
CB, (XtPointer)atm_threshold);
  XtManageChild(threshold_field);
/*
 *  Create text field for master trigger threshold
 */
  for(j = 0; j < 8; j++) {
    if(j == 0) {
      sprintf(msg, "LE thr  [mV (10mV/hit)]");
      k = 0;
    } else if(j == 1) {
      sprintf(msg, "HE thr  [mV (1.0mV/hit)]");
      k = 1;
    } else if(j == 2) {
      sprintf(msg, "SLE thr [mV (10mV/hit)]");
      k = 4;
    } else if(j == 3) {
      sprintf(msg, "Ni thr  [mV (10mV/hit)]");
      k = 2;
    } else if(j == 4) {
      sprintf(msg, "veto thr[mV (1.0mV/hit)]");
      k = 3;
    } else if(j == 5) {
```

```
      sprintf(msg, "BST thr [mV (10mV/hit)]");
      k = 5;
    } else if(j == 6) {
      sprintf(msg, "FADC thr[mV (5mV/hit)]");
      k = 6;
    }
    else {
      sprintf(msg, "not used [----------------]");
      k = i;
    }
    i = 0;
    label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNlabelString, label_string); i++;
    XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
    master_threshold[j] = XmCreateLabel(setup, "master_threshold", args, i);
    XtManageChild(master_threshold[j]);
    XmStringFree(label_string);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNeditable, TRUE); i++;
    XtSetArg(args[i], XmNblinkRate, 200); i++;
    XtSetArg(args[i], XmNmaxLength, 4); i++;
    sprintf(msg, "%d", ((u_short)(shm->master_trigger_threshold[k])*2));
    XtSetArg(args[i], XmNvalue, msg); i++;
    master_field[j] = XmCreateTextField(setup, "master_field", args, i);
/* bug found on 11 AUG 98
    XtAddCallback(master_field[j], XmNactivateCallback, (XtCallbackProc)MasterTh
reshCB, (XtPointer)k);
*/
/* Here is bug fixed */
    XtAddCallback(master_field[j], XmNactivateCallback, (XtCallbackProc)MasterTh
reshCB, (XtPointer)j);
    XtAddCallback(master_field[j], XmNmodifyVerifyCallback, (XtCallbackProc)Free
zeCB, (XtPointer)master_threshold[j]);
    XtManageChild(master_field[j]);
  }
/*
 *  Create text field for record path name
 */
  i = 0;
  label_string = XmStringCreateLtoR("record path :", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  record_path = XmCreateLabel(setup, "record_path", args, i);
  XtManageChild(record_path);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 40); i++;
  XtSetArg(args[i], XmNvalue, shm->record_path_name); i++;
  record_path_field = XmCreateTextField(setup, "record_path_field", args, i);
  XtAddCallback(record_path_field, XmNactivateCallback, (XtCallbackProc)RecordPa
thCB, (XtPointer)record_path);
  XtAddCallback(record_path_field, XmNmodifyVerifyCallback, (XtCallbackProc)Free
zeCB, (XtPointer)record_path);
  XtManageChild(record_path_field);
/*
 *  Create text field for maximum record size
```

```
 */
  i = 0;
  label_string = XmStringCreateLtoR("max record :", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
  max_record = XmCreateLabel(setup, "max_record", args, i);
  XtManageChild(max_record);
  XmStringFree(label_string);
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNeditable, TRUE); i++;
  XtSetArg(args[i], XmNblinkRate, 200); i++;
  XtSetArg(args[i], XmNmaxLength, 10); i++;
  sprintf(msg, "%d", shm->max_record_block);
  XtSetArg(args[i], XmNvalue, msg); i++;
  max_record_field = XmCreateTextField(setup, "max_record_field", args, i);
  XtAddCallback(max_record_field, XmNactivateCallback, (XtCallbackProc)MaxRecord
CB, (XtPointer)max_record);
  XtAddCallback(max_record_field, XmNmodifyVerifyCallback, (XtCallbackProc)Freez
eCB, (XtPointer)max_record);
  XtManageChild(max_record_field);
/*
 *  Create Form 5
 */
  i = 0;
  XtSetArg(args[i], XmNheight, 1); i++;
  form5 = XmCreateForm(panedw, "form5", args, i);
  XtManageChild(form5);
/*
 *  Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  XtSetArg(args[i], XmNnumColumns, 2); i++;
  setup2 = XmCreateRowColumn(form5, "setup2", args, i);
  XtManageChild(setup2);
  XmAddTabGroup(setup2);
/*
 *  Create Push button for pedestal taking flag
 */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  label_string = XmStringCreateLtoR("pedestal", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  pedestal = XmCreateToggleButton(setup2, "pedestal", args,i);
  XtManageChild(pedestal);
  XtAddCallback(pedestal, XmNdisarmCallback, (XtCallbackProc)PedestalCB, (XtPoin
ter)0);
  if(shm->flag_pedestal_request == 1) {
    XmToggleButtonSetState(pedestal, TRUE, NULL);
    XtSetArg(args[0], XmNselectColor, yellow.pixel);
    XtSetValues(pedestal, args, 1);
    XmUpdateDisplay(pedestal);
  }
/*
 *  Create Push button for writing disk flag
 */
  i = 0;
  label_string = XmStringCreateLtoR("disk write", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
```

```
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  disk_write = XmCreateToggleButton(setup2, "disk_write", args,i);
  XtManageChild(disk_write);
  XtAddCallback(disk_write, XmNdisarmCallback, (XtCallbackProc)DiskWriteCB, (XtP
ointer)0);
  if(shm->flag_write_possible == 1) {
    XmToggleButtonSetState(disk_write, TRUE, NULL);
    XtSetArg(args[0], XmNselectColor, yellow.pixel);
    XtSetValues(disk_write, args, 1);
    XmUpdateDisplay(disk_write);
  }
/*
 *  Create Push button for high rate switch for SMP
 */
  i = 0;
  label_string = XmStringCreateLtoR("high rate switch", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  high_rate = XmCreateToggleButton(setup2, "high_rate", args,i);
  XtManageChild(high_rate);
  XtAddCallback(high_rate, XmNdisarmCallback, (XtCallbackProc)HighRateCB, (XtPoi
nter)0);
  if(shm->high_rate_switch == 1) {
    XmToggleButtonSetState(high_rate, TRUE, NULL);
    XtSetArg(args[0], XmNselectColor, yellow.pixel);
    XtSetValues(high_rate, args, 1);
    XmUpdateDisplay(high_rate);
  }
/*
 *  Create Push button for raw data save switch for collector
 */
  i = 0;
  label_string = XmStringCreateLtoR("raw data save", XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  raw_save = XmCreateToggleButton(setup2, "raw_save", args,i);
  XtManageChild(raw_save);
  XtAddCallback(raw_save, XmNdisarmCallback, (XtCallbackProc)RawSaveCB, (XtPoint
er)0);
  if(shm->disk_save_switch == 1) {
    XmToggleButtonSetState(raw_save, TRUE, NULL);
    XtSetArg(args[0], XmNselectColor, yellow.pixel);
    XtSetValues(raw_save, args, 1);
    XmUpdateDisplay(raw_save);
  }

/*
 *  Create Form 6
 */
  i = 0;
  XtSetArg(args[i], XmNheight, 190); i++;
  form6 = XmCreateForm(panedw, "form6", args, i);
  XtManageChild(form6);
/*
 *  Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  XtSetArg(args[i], XmNnumColumns, 8); i++;
  setup3 = XmCreateRowColumn(form6, "setup3", args, i);
  XtManageChild(setup3);
```

```c
  XmAddTabGroup(setup3);
/*
 *  Create Push button for trigger enable
 */
  i = 0;
  sprintf(msg, "Low energy trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[0] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[0]);
  XtAddCallback(trigger_select[0], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)0);
  if(shm->master_trigger_select[LOW_ENERGY_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[0], TRUE, NULL);
  i = 0;
  sprintf(msg, "High energy trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[1] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[1]);
  XtAddCallback(trigger_select[1], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)1);
  if(shm->master_trigger_select[HIGH_ENERGY_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[1], TRUE, NULL);
  i = 0;
  sprintf(msg, "Super LE trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[11] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[11]);
  XtAddCallback(trigger_select[11], XmNdisarmCallback, (XtCallbackProc)TriggerSe
lectCB, (XtPointer)11);
  if(shm->master_trigger_select[SUPER_LOW_ENERGY_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[11], TRUE, NULL);
  i = 0;
  sprintf(msg, "Anti trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[3] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[3]);
  XtAddCallback(trigger_select[3], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)3);
  if(shm->master_trigger_select[ANTI_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[3], TRUE, NULL);
  i = 0;
  sprintf(msg, "Laser trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[4] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[4]);
  XtAddCallback(trigger_select[4], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)4);
  if(shm->master_trigger_select[LASER_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[4], TRUE, NULL);
  i = 0;
  sprintf(msg, "Xe lamp trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
```

```c
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[5] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[5]);
  XtAddCallback(trigger_select[5], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)5);
  if(shm->master_trigger_select[XE_LAMP_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[5], TRUE, NULL);
  i = 0;
  sprintf(msg, "Ni trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[6] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[6]);
  XtAddCallback(trigger_select[6], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)6);
  if(shm->master_trigger_select[NICKEL_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[6], TRUE, NULL);
  i = 0;
  sprintf(msg, "LINAC trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[7] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[7]);
  XtAddCallback(trigger_select[7], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)7);
  if(shm->master_trigger_select[LINAC_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[7], TRUE, NULL);
  i = 0;
  sprintf(msg, "Random trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[8] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[8]);
  XtAddCallback(trigger_select[8], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)8);
  if(shm->master_trigger_select[RANDOM_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[8], TRUE, NULL);
  i = 0;
  sprintf(msg, "Periodic trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[9] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[9]);
  XtAddCallback(trigger_select[9], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)9);
  if(shm->master_trigger_select[NULL_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[9], TRUE, NULL);
  i = 0;
  sprintf(msg, "Anti Nickel trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[10] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[10]);
  XtAddCallback(trigger_select[10], XmNdisarmCallback, (XtCallbackProc)TriggerSe
lectCB, (XtPointer)10);
  if(shm->master_trigger_select[ANTI_NICKEL_TRIGGER] == 1)
```

```
    XmToggleButtonSetState(trigger_select[10], TRUE, NULL);
  i = 0;
  sprintf(msg, "Burst test trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[2] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[2]);
  XtAddCallback(trigger_select[2], XmNdisarmCallback, (XtCallbackProc)TriggerSel
ectCB, (XtPointer)2);
  if(shm->master_trigger_select[ASSOCIATED_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[2], TRUE, NULL);
  i = 0;
  sprintf(msg, "Decay electron trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[12] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[12]);
  XtAddCallback(trigger_select[12], XmNdisarmCallback, (XtCallbackProc)TriggerSe
lectCB, (XtPointer)12);
  if(shm->master_trigger_select[DECAY_E_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[12], TRUE, NULL);
  i = 0;
  sprintf(msg, "After trigger");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[13] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[13]);
  XtAddCallback(trigger_select[13], XmNdisarmCallback, (XtCallbackProc)TriggerSe
lectCB, (XtPointer)13);
  if(shm->master_trigger_select[AFTER_TRIGGER] == 1)
    XmToggleButtonSetState(trigger_select[13], TRUE, NULL);

/* Add prescaling clock selection, due to shortage of TRIGGER_NUM, scaler[SCLRNU
M-1][SCLR_CHNUM-2]
  is used instead of master_trigger_select[14]. 2003/10/03 YO */
  i = 0;
  sprintf(msg, "Prescaling clock");
  label_string = XmStringCreateLtoR(msg, XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  trigger_select[14] = XmCreateToggleButton(setup3, "trigger_select", args,i);
  XtManageChild(trigger_select[14]);
  XtAddCallback(trigger_select[14], XmNdisarmCallback, (XtCallbackProc)TriggerSe
lectCB, (XtPointer)14);
  if(shm->scaler[SCLRNUM-1][SCLR_CHNUM-2] == 1)
    XmToggleButtonSetState(trigger_select[14], TRUE, NULL);

/* add color map on 22-Oct-2002 */
  for( i=0; i<TRIGGER_NUM; i++) {
    if(shm->master_trigger_select[i] == TRUE) {
      XtSetArg(args[0], XmNselectColor, yellow.pixel);
      XtSetValues(trigger_select[i], args, 1);
      XmUpdateDisplay(trigger_select[i]);
    }
  }
/* add color map for prescaling clock 03-OCT-2003 YO */
  {
    i=TRIGGER_NUM;
    if(shm->scaler[SCLRNUM-1][SCLR_CHNUM-2] == 1) {
```

```
      XtSetArg(args[0], XmNselectColor, yellow.pixel);
      XtSetValues(trigger_select[i], args, 1);
      XmUpdateDisplay(trigger_select[i]);
    }
  }
/*
 *  Create Form 4
 */
  i = 0;
  XtSetArg(args[i], XmNlabelFontList, fontlist); i++;
  XtSetArg(args[i], XmNheight, 1); i++;
  form4 = XmCreateForm(panedw, "form4", args, i);
  XtManageChild(form4);
/*
 *  Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  XtSetArg(args[i], XmNnumColumns, 2); i++;
  smp_mask = XmCreateRowColumn(form4, "smp_mask", args, i);
  XtManageChild(smp_mask);
/*
 * Create Push button for mask set of SMP module
 */
  for (j = 0; j < NUM_SERV; j++) {
/*
 *  Only TKO server WS
 */
    if((shm->stab[j].f_server != TRG_SERVER) &&
       (shm->stab[j].f_server != XXX_CLIENT)) {
/*
 *  Create smp status
 */
      i = 0;
      sprintf(word, "%s : SMP status %.2x", ONL_SRV[j], shm->stab[j].smp_mask);
      label_string = XmStringCreateLtoR(word, XmSTRING_DEFAULT_CHARSET);
      XtSetArg(args[i], XmNfontList, fontlist); i++;
      XtSetArg(args[i], XmNlabelString, label_string); i++;
      XtSetArg(args[i], XmNalignment, XmALIGNMENT_END); i++;
      head[j] = XmCreateLabel(smp_mask, "head", args, i);
      XtManageChild(head[j]);
      XmStringFree(label_string);
/*
 *  Create popup menu
 */
      popup[j] = XmCreatePopupMenu(head[j], "popup", args, NULL);
      XtAddEventHandler(head[j], ButtonPressMask, False, (XtEventHandler)PostIt,
 popup[j]);
/*
 * Create pulldown menu for mask on/off
 */
      for(k = 0; k < NUMSMP; k++) {
          i = 0;
          XtSetArg(args[i], XmNfontList, fontlist); i++;
          serverBtn[j][k] = XmCreatePulldownMenu(popup[j], "server_button", args, i);
          i = 0;
          sprintf(word, "smp %d", k);
          label_string = XmStringCreateLtoR(word, XmSTRING_DEFAULT_CHARSET);
          XtSetArg(args[i], XmNlabelString, label_string); i++;
          XtSetArg(args[i], XmNfontList, fontlist); i++;
          XtSetArg(args[i], XmNsubMenuId, serverBtn[j][k]); i++;
          smp[j][k] = (Widget)XmCreateCascadeButtonGadget(popup[j], "smp", args, i
```

```
);
          XtManageChild(smp[j][k]);
          XmStringFree(label_string);
          i = 0;
          label_string = XmStringCreateLtoR("off", XmSTRING_DEFAULT_CHARSET);
          XtSetArg(args[i], XmNfontList, fontlist); i++;
          XtSetArg(args[i], XmNlabelString, label_string); i++;
          smp_on[j][k] = (Widget)XmCreatePushButtonGadget(serverBtn[j][k], "smp_on
", args, i);
          XtAddCallback(smp_on[j][k], XmNactivateCallback, (XtCallbackProc)maskOnC
B, (char *)(100*j+k));
          XtManageChild(smp_on[j][k]);
          XmStringFree(label_string);
          i = 0;
          label_string = XmStringCreateLtoR("on", XmSTRING_DEFAULT_CHARSET);
          XtSetArg(args[i], XmNfontList, fontlist); i++;
          XtSetArg(args[i], XmNlabelString, label_string); i++;
          smp_off[j][k] = (Widget)XmCreatePushButtonGadget(serverBtn[j][k], "smp_of
f", args, i);
          XtAddCallback(smp_off[j][k], XmNactivateCallback, (XtCallbackProc)maskOf
fCB, (char *)(100*j+k));
          XtManageChild(smp_off[j][k]);
          XmStringFree(label_string);
      }
    }
  }
/*
 * Create Form 2
 */
  i = 0;
  /*  XtSetArg(args[i], XmNheight, 240); i++;*/
  XtSetArg(args[i], XmNheight, 1); i++;
  form2 = XmCreateForm(panedw, "form2", args, i);
  XtManageChild(form2);
/*
 * Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
  XtSetArg(args[i], XmNnumColumns, 16); i++;
  network = XmCreateRowColumn(form2, "network", args, i);
  XtManageChild(network);
/*
 * Create Push button for sever mode
 */
  for (j = 0; j < NUM_SERV; j++) {
    char* shost;

    i = 0;
    sprintf(host, "%s:", ONL_SRV[j]);
    label_string = XmStringCreateLtoR(host, XmSTRING_DEFAULT_CHARSET);
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNlabelString, label_string); i++;
    XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
    if(shost=strstr(host,"NULL")){
      XtSetArg(args[i], XmNsensitive, FALSE); i++;
    }else{
      XtSetArg(args[i], XmNsensitive, TRUE); i++;
    }
    server_label[j] = XmCreateLabel(network, "server_label", args, i);
    XtManageChild(server_label[j]);
```

```
    XmStringFree(label_string);

    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    server[j] = XmCreatePulldownMenu(network, "server_name", args, i);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNmnemonic, ’1’); i++;
    popupBtn[0] = XmCreatePushButtonGadget(server[j], "off", args, i);
    XtAddCallback(popupBtn[0], XmNactivateCallback, (XtCallbackProc)WSDownCB, (X
tPointer)j);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNmnemonic, ’2’); i++;
    popupBtn[1] = XmCreatePushButtonGadget(server[j], "server", args, i);
    XtAddCallback(popupBtn[1], XmNactivateCallback, (XtCallbackProc)ServerOnCB,
(XtPointer)j);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNmnemonic, ’3’); i++;
    popupBtn[2] = XmCreatePushButtonGadget(server[j], "client", args, i);
    XtAddCallback(popupBtn[2], XmNactivateCallback, (XtCallbackProc)ClientOnCB,
(XtPointer)j);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNmnemonic, ’4’); i++;
    popupBtn[3] = XmCreatePushButtonGadget(server[j], "trigger", args, i);
    XtAddCallback(popupBtn[3], XmNactivateCallback, (XtCallbackProc)TriggerOnCB,
 (XtPointer)j);
    i = 0;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNmnemonic, ’5’); i++;
    popupBtn[4] = XmCreatePushButtonGadget(server[j], "monitor", args, i);
    XtAddCallback(popupBtn[4], XmNactivateCallback, (XtCallbackProc)MonitorOnCB,
 (XtPointer)j);
    XtManageChildren(popupBtn, 5);

    i = 0;
    sprintf(host, " ");
    label_string = XmStringCreateLtoR(host, XmSTRING_DEFAULT_CHARSET);
    XtSetArg(args[i], XmNlabelString, label_string); i++;
    XtSetArg(args[i], XmNfontList, fontlist); i++;
    XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
    if(shm->stab[j].f_server == 0) {
      XtSetArg(args[i], XmNmenuHistory, popupBtn[0]); i++;
    } else if(shm->stab[j].f_server == TKO_SERVER) {
      XtSetArg(args[i], XmNmenuHistory, popupBtn[1]); i++;
    } else if(shm->stab[j].f_server == TRG_SERVER) {
      XtSetArg(args[i], XmNmenuHistory, popupBtn[3]); i++;
    } else if(shm->stab[j].f_server == XXX_CLIENT) {
      XtSetArg(args[i], XmNmenuHistory, popupBtn[2]); i++;
    } else if(shm->stab[j].f_server == ONL_MONITOR) {
      XtSetArg(args[i], XmNmenuHistory, popupBtn[4]); i++;
    } else {
      shm->stab[j].f_server == 0;
      XtSetArg(args[i], XmNmenuHistory, popupBtn[0]); i++;
    }
    XtSetArg(args[i], XmNmnemonic, ’6’); i++;
    XtSetArg(args[i], XmNsubMenuId, server[j]); i++;
    if(shost){
      XtSetArg(args[i], XmNsensitive, FALSE); i++;
    }else{
```

```
      XtSetArg(args[i], XmNsensitive, TRUE); i++;
    }
    server_type[j] = XmCreateOptionMenu(network, "server_type", args, i);
    XtManageChild(server_type[j]);
    XmStringFree(label_string);
  }
/*
 *  Create Form 7
 */
  i = 0;
  XtSetArg(args[i], XmNheight, 1); i++;
  form7 = XmCreateForm(panedw, "form7", args, i);
  XtManageChild(form7);
/*
 *  Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  XtSetArg(args[i], XmNnumColumns, 4); i++;
  setup5 = XmCreateRowColumn(form7, "setup5", args, i);
  XtManageChild(setup5);
  XmAddTabGroup(setup5);
/*
 *  Create Push button for electronics checking on sukon1
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon1",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  elec[0] = XmCreateToggleButton(setup5, "elec[0]", args,i);
  XtManageChild(elec[0]);
  XtAddCallback(elec[0], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)0
);
  if(shm->stab[0].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[0], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon2
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon2",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  elec[1] = XmCreateToggleButton(setup5, "elec[1]", args,i);
  XtManageChild(elec[1]);
  XtAddCallback(elec[1], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)1
);
  if(shm->stab[1].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[1], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon3
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon3",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  elec[2] = XmCreateToggleButton(setup5, "elec[2]", args,i);
  XtManageChild(elec[2]);
  XtAddCallback(elec[2], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)2
```

```
);
  if(shm->stab[2].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[2], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon4
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon4",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  elec[3] = XmCreateToggleButton(setup5, "elec[3]", args,i);
  XtManageChild(elec[3]);
  XtAddCallback(elec[3], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)3
);
  if(shm->stab[3].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[3], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon5
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon5",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  elec[4] = XmCreateToggleButton(setup5, "elec[4]", args,i);
  XtManageChild(elec[4]);
  XtAddCallback(elec[4], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)4
);
  if(shm->stab[4].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[4], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon6
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon6",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  elec[5] = XmCreateToggleButton(setup5, "elec[5]", args,i);
  XtManageChild(elec[5]);
  XtAddCallback(elec[5], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)5
);
  if(shm->stab[5].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[5], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon7
 */
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon7",
                                         XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  elec[6] = XmCreateToggleButton(setup5, "elec[6]", args,i);
  XtManageChild(elec[6]);
  XtAddCallback(elec[6], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)6
);
  if(shm->stab[6].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[6], TRUE, NULL);
/*
 *  Create Push button for electronics checking on sukon8
 */
```

```
  i = 0;
  label_string = XmStringCreateLtoR("elec check sukon8",
                                    XmSTRING_DEFAULT_CHARSET);
  XtSetArg(args[i], XmNlabelString, label_string); i++;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  elec[7] = XmCreateToggleButton(setup5, "elec[7]", args,i);
  XtManageChild(elec[7]);
  XtAddCallback(elec[7], XmNdisarmCallback, (XtCallbackProc)ElecCB, (XtPointer)7
);
  if(shm->stab[7].electronics_check_flag == 1)
    XmToggleButtonSetState(elec[7], TRUE, NULL);
/* add color map on 22-Oct-2002 */
  for( i=0; i<8; i++) {
    if(shm->stab[i].electronics_check_flag == 1) {
      XtSetArg(args[0], XmNselectColor, yellow.pixel);
      XtSetValues(elec[i], args, 1);
      XmUpdateDisplay(elec[i]);
    }
  }
/*
 * Create Form 3
 */
  form3 = XmCreateForm(panedw, "form3", NULL, NULL);
  XtManageChild(form3);
/*
 * Create row-column
 */
  i = 0;
  XtSetArg(args[i], XmNorientation, XmHORIZONTAL); i++;
  XtSetArg(args[i], XmNpacking, XmPACK_COLUMN); i++;
  /* change number of columns on Aug-2003
   XtSetArg(args[i], XmNnumColumns, 2); i++;
   */
  XtSetArg(args[i], XmNnumColumns, 1); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmarginWidth, 15); i++;
  control = XmCreateRowColumn(form3, "control", args, i);
  XtManageChild(control);
/*
 * create push button for RunChange processes
 */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmultiClick, XmMULTICLICK_DISCARD); i++;
  change = XmCreatePushButton(control, "RunChange", args, i);
  XtAddCallback(change, XmNactivateCallback, (XtCallbackProc)RunChangeCB, NULL);
  XtManageChild(change);
/*
 * create push button for setup(start) processes
 */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmultiClick, XmMULTICLICK_DISCARD); i++;
  start = XmCreatePushButton(control, "Start   ", args, i);
  XtAddCallback(start, XmNactivateCallback, (XtCallbackProc)AutoStartCB, NULL);
  XtManageChild(start);
/*
 * create push button for stop for run
 */
  i = 0;
```

```
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmultiClick, XmMULTICLICK_DISCARD); i++;
  stop = XmCreatePushButton(control, "stop", args, i);
  XtAddCallback(stop, XmNactivateCallback, (XtCallbackProc)StopCB, NULL);
  XtManageChild(stop);
/*
 * create push button for abort processes
 */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmultiClick, XmMULTICLICK_DISCARD); i++;
  abort = XmCreatePushButton(control, "abort", args, i);
  XtAddCallback(abort, XmNactivateCallback, (XtCallbackProc)AbortCB, NULL);
  XtManageChild(abort);
/*
 * create push button for quit
 */
  i = 0;
  XtSetArg(args[i], XmNfontList, fontlist); i++;
  XtSetArg(args[i], XmNalignment, XmALIGNMENT_CENTER); i++;
  XtSetArg(args[i], XmNmultiClick, XmMULTICLICK_DISCARD); i++;
  quit = XmCreatePushButton(control, "QUIT", args, i);
  XtAddCallback(quit, XmNactivateCallback, (XtCallbackProc)QuitCB, NULL);
  XtManageChild(quit);
/*
 *  set buttton status
 */
  if(shm->flag_status == CNTL_START) {
/*
 *  insensitive text fields
 */
      i = 0;
      XtSetArg(args[i], XmNsensitive, FALSE); i++;
      XtSetValues(num_field, args, i);
      XtSetValues(leader_field, args, i);
      XtSetValues(member_field, args, i);
      XtSetValues(comment_field, args, i);
      XtSetValues(threshold_field, args, i);
      for(j=0; j<8; j++)
         XtSetValues(master_field[j], args, i);
      for(j=0; j<NUM_SERV; j++)
         XtSetValues(server[j], args, i);
      for(j=0; j<TRIGGER_NUM+TRIGGER_NUM_EXTRA; j++)
         XtSetValues(trigger_select[j], args, i);
/*
 *  insensitive "start" button
 */
      i = 0;
      XtSetArg(args[i], XmNsensitive, FALSE); i++;
      XtSetValues(start, args, i);
/*
 *  insensitive "stop" button
 */
      i = 0;
      XtSetArg(args[i], XmNsensitive, TRUE); i++;
      XtSetValues(stop, args, i);
      XtSetValues(change, args, i);
  }
  else {
/*
```

```
 *  sensitive "start" button
 */
      i = 0;
      XtSetArg(args[i], XmNsensitive, TRUE); i++;
      XtSetValues(start, args, i);
/*
 *  insensitive "stop" button
 */
      i = 0;
      XtSetArg(args[i], XmNsensitive, FALSE); i++;
      XtSetValues(stop, args, i);
      XtSetValues(change, args, i);
  }
/*
 * Define call back function
 */
  XmMainWindowSetAreas(mainw, menu, NULL, NULL, NULL, frame);
/*
 * Realize Widget
 */
  XtRealizeWidget(toplevel);
/*
 * Infinite loop
 */
  XtAppMainLoop(app_context);
}

int  msgchk(message, flag)
char message[512];
u_int flag;
{
  int i;

  if(flag == CNTL_ABORT) {
    if(strncmp(buffer->mtext,"Still ",6) == 0){
      strcpy(message,buffer->mtext);
      return -2;
    }
    i = 0;
    if(strncmp(buffer->mtext, "collec", 6) == 0) {
      strncpy(message, "Collector is not aborted : \0", 28);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sorter", 6) == 0) {
      i++;
      strncat(message, "Sorter is not aborted : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0){
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sender", 6) == 0) {
      i++;
```

```
    strncat(message, "Sender is not aborted : \0", 25);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
        strncat(message, (buffer->mtext + (i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
    if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
    strncat(message, "\n", 1);
  }
  if(strncmp(buffer->mtext+(i+1)*8, "evbild", 6) == 0) {
    i++;
    strncat(message, "EventBuilder is not aborted : \0", 31);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
        strncat(message, (buffer->mtext + (i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
    return -1;
  }
}
else if(flag == CNTL_INITIALIZE) {
  i = 0;
  if(strncmp(buffer->mtext, "collec", 6) == 0) {
    strncpy(message, "Collector is not running : \0", 28);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
        strncat(message, (buffer->mtext+(i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
    if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
    strncat(message, "\n", 1);
  }
  if(strncmp(buffer->mtext+(i+1)*8, "sorter", 6) == 0) {
    i++;
    strncat(message, "Sorter is not running : \0", 25);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0){
        strncat(message, (buffer->mtext+(i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
    if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
    strncat(message, "\n", 1);
  }
  if(strncmp(buffer->mtext+(i+1)*8, "sender", 6) == 0) {
    i++;
    strncat(message, "Sender is not running : \0", 25);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
        strncat(message, (buffer->mtext + (i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
    if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
    strncat(message, "\n", 1);
  }
  if(strncmp(buffer->mtext+(i+1)*8, "evbild", 6) == 0) {
    i++;
    strncat(message, "EventBuilder is not running : \0", 31);
    while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
        strncat(message, (buffer->mtext + (i+1)*8), 8);
        strncat(message, " ", 1);
        i++;
    }
```

```
      return -1;
    }
  } else if(flag == CNTL_SETUP) {
    i = 0;
    if(strncmp(buffer->mtext, "collec", 6) == 0) {
      strncpy(message, "Collector is not running : \0", 28);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext + (i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sorter", 6) == 0) {
      i++;
      strncat(message, "Sorter is not running : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0){
          strncat(message, (buffer->mtext + (i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sender", 6) == 0) {
      i++;
      strncat(message, "Sender is not running : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext + (i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "evbild", 6) == 0) {
      i++;
      strncat(message, "EventBuilder is not running : \0", 31);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext + (i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      return -1;
    }
  } else if(flag == CNTL_INSTALL) {
    i = 0;
    if(strncmp(buffer->mtext, "client", 6) == 0) {
      strncpy(message, "Client is not running : \0", 25);
      while(strncmp(buffer->mtext+i*8, ":", 1) == 0) {
          strncat(message, (buffer->mtext+i*8+1), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+i*8+1, "server", 6) == 0) {
      strncpy(message, "Server is not running : \0", 25);
      i++;
      while(strncmp(buffer->mtext+i*8, ":", 1) == 0) {
```

```
          strncat(message, (buffer->mtext+i*8+1), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+i*8+1, "msgsrv", 6) == 0) {
      strncpy(message, "msgserv is not running : \0", 26);
      i++;
      while(strncmp(buffer->mtext+i*8, ":", 1) == 0) {
          strncat(message, (buffer->mtext+i*8+1), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
  } else if(flag == CNTL_START) {
    i = 0;
    if(strncmp(buffer->mtext, "collec", 6) == 0) {
      strncpy(message, "Collector is not running : \0", 28);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sorter", 6) == 0) {
      i++;
      strncat(message, "Sorter is not running : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0){
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sender", 6) == 0) {
      i++;
      strncat(message, "Sender is not running : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "evbild", 6) == 0) {
      i++;
      strncat(message, "EventBuilder is not running : \0", 31);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      return -1;
    }
```

```
  } else if(flag == CNTL_STOP) {
    if(strncmp(buffer->mtext, "dead", 4) == 0) return -2;
    i = 0;
    if(strncmp(buffer->mtext, "collec", 6) == 0) {
      strncpy(message, "Collector is not stopped : \0", 28);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+6, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sorter", 6) == 0) {
      i++;
      strncat(message, "Sorter is not stopped : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0){
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "sender", 6) == 0) {
      i++;
      strncat(message, "Sender is not stopped : \0", 25);
      while(strncmp(buffer->mtext+i*8+7, ":", 1) == 0) {
          strncat(message, (buffer->mtext+(i+1)*8), 8);
          strncat(message, " ", 1);
          i++;
      }
      if(strncmp(buffer->mtext+i*8+7, "+", 1) == 0) return -1;
      strncat(message, "\n", 1);
    }
    if(strncmp(buffer->mtext+(i+1)*8, "evbild", 6) == 0) {
      i++;
      strncat(message, "EventBuilder is not stopped : sukonh", 36);
      return -1;
    }
  }
  return 0;
}

void handler()
{
  int i, j, mark, len, status, pid, pre_run_num, time_diff, nevent;
  static int icnt = 0;
  char message[80];
  XmString title_string = NULL;
  XmString message_string = NULL;
  XmString button_string = NULL;
/*
 *  reset alarm
 */
  alarm(0);
/*
 *  kill message queue and controller automatically
 */
  rpc_create("sukonh", PMN_PORT);
  return;
}
```