KEITHLEY

Application Note Series

Number 2425

Sending Email within KTXE

Introduction

It can often take days to disposition a lot that's held or failed due to a critical parameter that measures out of control at electrical parametric wafer probe, while a summary report is generated and analyzed. The time required for process diagnostics can be cut from days to minutes by giving KTXE the ability to send "email" text messages automatically to convey the pass/fail status of a lot or wafer to the test engineer or test manager during execution of a cassette plan. This note describes a KULT routine for sending email from within KTXE. A test engineer or field applications engineer can implement this routine in about an hour.

Applicability

This capability applies to S400 and S600 series systems running KTE v3.x and higher. The Sun system controller must have access to a network that supports email. The mailer daemons running under the Sun Solaris operating system also must be active.

Description

The example KULT routine supplied here is called *SendEmail()* and can be executed at any User Access Point (UAP) available in KTXE. The three arguments to this routine are:

RECIPIENT	A string containing either a file name containing a list of recipients or a comma-separated list of recipients.
SUBJECTLINE	A string containing the text appearing in the email's subject field.
MSGFILE	A string containing the file name of a text file to be included as the main part of the email.

RECIPIENT, SUBJECTLINE, and MSGFILE are *required* arguments. The RECIPIENT argument may specify a filename, which would contain the list of recipients (separated by commas). If the filename does not exist, this argument contains the list of recipients.

For example, consider an email message to be sent to the fab manager and the test engineer that contains the lot summary file at the end of testing a lot:

```
    UAP_LOT_END
    uap_lib : SendEmail
    ("fabmgr@semi.com,tsteng@semi.com",
    "lot:J64N_shrink summary report",
"/opt/ki/db/J64N_shrink.sum")
```

NOTE: In this example, the *SendEmail()* routine is located in a KULT library called: *uap_lib*

To take another example, consider an email sent to the Device Group that includes the wafer pass/fail evaluation report generated after each wafer has been tested. The list of recipients is contained in a file named "*/export/home/kthmgr/devgrp.list*".

```
UAP_WAFER_END uap_lib : SendEmail
("/export/home/kthmgr/devgrp.list",
"wafer:1 Eval report",
"/export/home/rpt/wafer1.eval")
```

If, as in the previous example, the list of recipients is contained in a file, each recipient is delimited by a comma or specified on separate lines within the file. For example, the file "*/export/home/kthmgr/devgrp.list*" contains:

```
kthmgr
fabmgr,bob,
sr_device_person
jr_device_person, tom, mike
```

To enhance readability, it's acceptable to use blank lines and spaces between names.

Installation

Download the *SendEmail.c* file to a working directory on the Sun sorkstation so it can be added to a specified KULT library. To add this routine to a KULT library, type in this command at the Solaris command prompt: *kult add_mod -lmylib SendEmail.c* When entering this command, make sure "-l" is a lower case L; replace *mylib* with the name of the actual KULT library.

To use this routine in the execution of KTXE, use KTPM (Keithley Test Plan Manager) to select the appropriate UAP and provide the proper arguments to the *SendEmail()* routine.

Other Considerations and Risk Analysis

The supplied *SendEmail.c* code is intended to be used exactly as-is without any customer modification. Improper modification, installation, or use of this code may render this feature inoperable, but will have no impact on tester uptime or measurement results. Modification to the code may be required at some future Solaris release if Sun changes the supporting mailer capability. If this occurs, please contact the factory.

```
SendEmail.c code
/* USRLIB MODULE INFORMATION
       MODULE NAME: SendEmail
       MODULE RETURN TYPE: int
       NUMBER OF PARMS: 3
       ARGUMENTS:
             Recipient,   char *,      Input, ,       ,
             Subjectline, char *,      Input, ,       ,
             Msgfile,     char *,      Input, ,       ,
       INCLUDES:
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include <string.h>
#include <time.h>
#define NO_MEMORY -1
#define CMD_TOO_BIG -2
#define NO_DESTINATION -3
#define NO_SUBJECT -4
#define NO_MSG_FILE -5
#define GOOD 1
#define DEBUG
       END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION
       END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE PARAMETER LIST */
#include <stdio.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
#include <string.h>
#include <time.h>
#define NO_MEMORY -1
#define CMD_TOO_BIG -2
#define NO_DESTINATION -3
#define NO_SUBJECT -4
#define NO_MSG_FILE -5
#define GOOD 1
#define DEBUG
int SendEmail( char *Recipient, char *Subjectline, char *Msgfile )
{
/* USRLIB MODULE CODE */
char   *mailcmd;
char   *sendlist;
char   finput[512];
char   *comma = ",";
char   chr = NULL;
int    status;
int    size_Rec  = 0;
int    size_Subj = 0;
int    size_Msg  = 0;
int    size_mailcmd = 0;
FILE   *fptr = NULL;
/* Variables for the system time */
time_t        sys_time;
struct tm  *timeptr;
/* ============================================== */
#ifdef DEBUG
       printf ("SendEmail...\n");
#endif
/* Verify the required inputs */
if (Recipient[0] == '\0')     /* No one to recieve the email */
       return NO_DESTINATION;
if (Subjectline[0] == '\0')   /* No subject, No mail ediquette */
       return NO_SUBJECT;
if (Msgfile[0] == '\0')        /* No Message file */
       return NO_MSG_FILE;
/* Check to see if the Message file is valid */
if ( (fptr = fopen (Msgfile, "r") ) == NULL)
       return NO_MSG_FILE;       /* Msgfile couldn't be opened */
fclose (fptr);
/* Get the system time, will be appended in Subject field */
sys_time = time ( (time_t *) 0 );
timeptr  = localtime (&sys_time);
/*
Check to see if Recipient is a filename.  If so, get
the # of chars in the file.  If not, Recipient contains
the email destination name(s) - still get it’s size.
*/
size_Rec = 0;
if ( (fptr = fopen (Recipient, “r”) ) != NULL)
{
       while (!feof (fptr))
       {
              chr = fgetc (fptr);
              if (feof (fptr) )
                     break;
              if (chr != ‘\n’)
                     size_Rec++;
       }
       fclose (fptr);
}
else                                /* Recipient is *not* a filename */
       size_Rec  = strlen (Recipient);
/*
Allocate space for the mail command.
Allocate more space just for good measure.
*/
size_Subj    = strlen (Subjectline);
size_Msg     = strlen (Msgfile);
size_mailcmd = size_Rec + size_Subj + size_Msg + 100;
if (size_mailcmd > 512)
       return CMD_TOO_BIG;
mailcmd  = (char *) malloc (sizeof(char)*size_mailcmd );
sendlist = (char *) malloc (sizeof(char)*(size_Rec + 100));
if (mailcmd == NULL  ||  sendlist == NULL)
       return NO_MEMORY;
memset (mailcmd,  ‘\0’, size_mailcmd);   /* set to NULLs */
memset (sendlist, ‘\0’, size_Rec);
if ( (fptr = fopen (Recipient, “r”) ) != NULL)
{
       while (!feof (fptr))
       {
              fscanf (fptr, “%s”, &finput);
              if (feof (fptr) )
                     break;
              RemoveWhiteSpace (finput);
              strcat (sendlist, finput);
              strcat (sendlist, comma);
       }
       fclose (fptr);
}
else
       strcpy (sendlist, Recipient);
/*
Now build the mail command:
       “mailx -s “subject time” sendlist < msgfile”
Note: the system time is added to the Subject text.
The “\042” is for the double quotes, used to start
and end the Subject field.
 */
sprintf (mailcmd, “mailx -s \042%s %s\042 %s < %s\0”,
       Subjectline, asctime(timeptr), sendlist, Msgfile);
#ifdef DEBUG
       printf (“mailcmd:%s:\n”, mailcmd);
#endif
status = system (mailcmd);
#ifdef DEBUG
       printf (“mail status:%d:\n”, status);
#endif
/* Free up the allocate space - don’t be a memory hog!! */
free (mailcmd);
free (sendlist);
if (status < 0)
   return status;
return GOOD;
/* USRLIB MODULE END  */
}      /* End SendEmail.c */
```

Glossary of Abbreviations Used

KTE–Keithley Test Environment

KTPM–Keithley Test Plan Manager

KTXE–Keithley Test Execution Engine

KULT–Keithley User Library Tool

TCP/IP–Transmission Control Protocol/Internet Protocol

UAP–User Access Point

KEITHLEY

Keithley Instruments, Inc. 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168
1-888-KEITHLEY (534-8453) • www.keithley.com