Trigger VME/ONL Interface
working document

This document proposes a scheme for the various interfaces between TRG VME and ONL. Various pieces of this scheme have been written, and it may be more efficient to consolidate the software into a general framework and have it eventually incorporated into a general TRG GUI. It is assumed that this will be a stand–alone system and can be interfaced with the overall STAR ONL system. This document is a strawman to help define the information available to TRG ONL for eventual interfacing with STAR ONL. The following components are deemed necessary for the system:

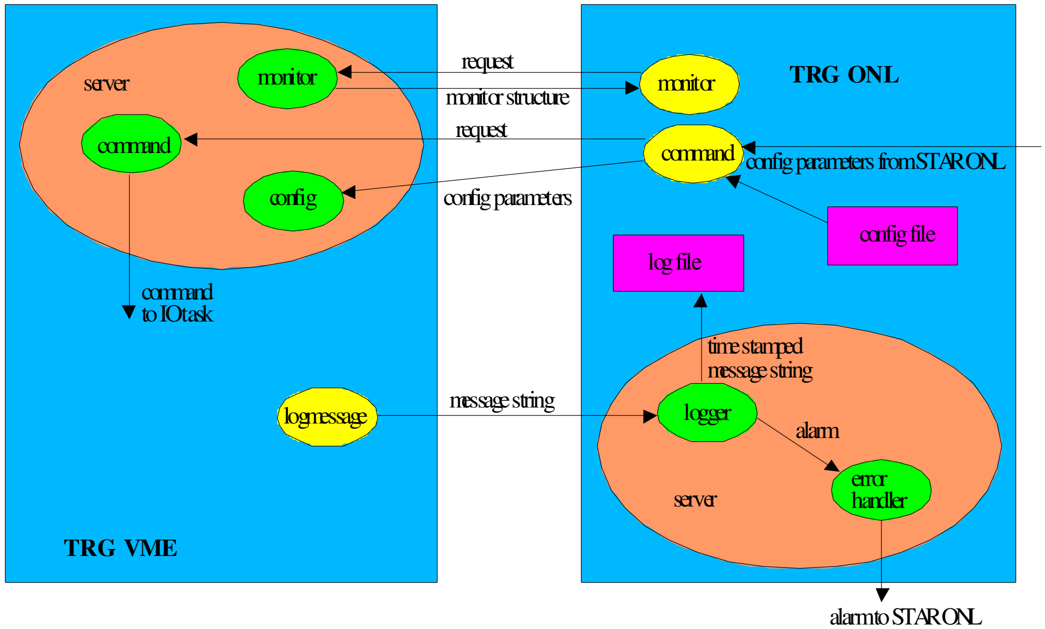
Requirement. User Commands
Justification. The STAR user needs to start, stop and configure a run in an easy manner from the TRG console.
Implementation. This part of the system is the most mature. It has been used for the summer 2000 run and has no major problems. Possible improvements are using commands that are more intuitive to the user, and a more friendlier configuration procedure. This will be discussed in detail later. GUI interface from the TRG console also possible.

Requirement. Message Logging.
Justification. The TRG system needs to log messages so that the TRG experts can review at any time the history of the way the system has functioned.
Implementation. This system is in place and has been tested, although not extensively. This was not used for the summer 2000 run.

Requirement. Error handling.
Justification. The STAR user needs to be informed through TRG ONL that there is something wrong with the TRG system. This needs to be done through an alarm and a message telling what the error is and what to do about it. In some cases it could be possible to automatically correct the error.
Implementation. Proposal below.

Requirement Monitoring.
Justification. The functioning of the TRG system must be done interactively during running.
Implementation. Various pieces of software have been written to monitor thing such as state, token flow, etc. Need a general interface for monitoring.

The following diagram illustrates the proposed scheme for the VME/ONL interface. It implies that one server process is running on each VME CPU and one on the TRG ONL workstation (startrg). This is not a set requirement since it may be decided that separate server processes are more efficient at handling the various tasks. The arrows indicate structures passed along the interface, and this will be discussed in detail below. The interface is assumed to be a socket connection. It is not clear whether this will be an open socket connection, or one which is opened and closed per request.



Interfaces

logmessage/logger:	string of length 255 characters.
Logger/log file:	string of length 255 characters written to log file of the following format – the first 17 characters are the time stamp, the 18th is the log level, and the rest is the truncated string (truncated to add a time stamp) of the message from the VME processor, which includes the task sending the message

Proposed TRG Error Handler

A simple way of doing this would be to write a function inside the server process. When the log level is less than a certain value, say LOG_NOTICE, then the error handler is called. The only information that need be passed to the error handler would be an integer indicating the level of the message. The error handler would then send an alarm to the console of TRG ONL (startrg) as well as send an alarm to STAR ONL. Again, the information passed from TRG ONL to STAR ONL need only be an integer. Alternatively, the full character string could also be passed from the logger to the error handler to STAR ONL. The following are the priorities defined in the syslog.h file and used by the TRG system.

```
/*
 *  Priorities (these are ordered)
 */
#define LOG_EMERG       0       /* system is unusable */
#define LOG_ALERT       1       /* action must be taken immediately */
```

```
#define LOG_CRIT      2     /* critical conditions */
#define LOG_ERR       3     /* error conditions */
#define LOG_WARNING   4     /* warning conditions */
#define LOG_NOTICE    5     /* normal but signification condition */
#define LOG_INFO      6     /* informational */
#define LOG_DEBUG     7     /* debug-level messages */
```

It is possible to define, for each level of message, an action that needs to be taken by the user. For instance, a LOG_WARNING, could be an alarm where no action is needed, but serves as a notification to the user that an abnormal condition occurred. An example would be the RHIC clock dropping out and being reset automatically. The following list of actions is a proposal for the various levels of the messages.

LOG_EMERG	Call TRG expert.
LOG_ALERT	Condition requiring power cycle.
LOG_CRIT	Critical condition requiring reboot.
LOG_ERR	Error has occurred. Redo procedure – restart run, reconfigure, etc.
LOG_WARNING	Abnormal condition, but no action needed.
LOG_NOTICE	General information for completion of some task – run has started, or ended, configuration completed, etc.
LOG_INFO	Used to monitor token flow. No action or alarm to general user.
LOG_DEBUG	Debug information, usually data flow integrity. No action or notification to the general user.

The above is open for discussion. It is not yet clear whether all errors encountered will fit in the above categories. The following is a list of all possible messages currently being written to the log files. The error messages that are less than a level of LOG_NOTICE were attempted to be highlighted. To determine if the above scheme will work, a review of all of the highlighted errors needs to be made to see if anything would fall outside of the above guidelines, or if new actions are needed for the various error levels.

L1CTL

```
  sprintf(logstr, "Trigger is NOT CONFIGURING. L1_Hardware_Interface should NOT exist\n");
  logmessage(LOG_EMERG,"L1_Hard_Conf",logstr);
   sprintf(logstr, "Failed to configure TCU - ERROR = %i\n",err);
   logmessage(LOG_ERR,"L1_Hard_Conf",logstr);
  sprintf(logstr, "Unable to configure TCU - no base address\n");
  logmessage(LOG_ERR,"L1_Hard_Conf",logstr);
         sprintf(logstr, "Failed to configure DSM %i - ERROR = %i\n",i,err);
         logmessage(LOG_ERR, "L1_Hard_Conf",logstr);
        sprintf(logstr, "Unable to configure DSM %i - no base address\n",i);
        logmessage(LOG_ERR, "L1_Hard_Conf",logstr);
         sprintf(logstr, "Failed to configure last DSM %i - ERROR = %i\n",i,err);
         logmessage(LOG_ERR, "L1_Hard_Conf",logstr);
   sprintf(logstr, "Failed to find last DSM\n");
   logmessage(LOG_ERR, "L1_Hard_Conf", logstr);
         sprintf(logstr, "Unable to reset address to zero, value = 0x%i\n",addr) ;
         logmessage( LOG_EMERG, "L1_Hard_Conf", logstr);
          sprintf(logstr, "Failed to increment address by %i: %i\n",RT->L0.DSM_Addr_Incr[i],addr) ;
```

```
              logmessage( LOG_EMERG, "L1_Hard_Conf", logstr);
            sprintf(logstr, "Unable to resynchronize DSM %i - no base address\n",i);
            logmessage(LOG_ERR, "L1_Hard_Conf",logstr);
  sprintf(logstr, "Finished Configuring L0 DSMs\n");
  logmessage(LOG_NOTICE, "L1_Hard_Conf",logstr);
    sprintf("Serious error received:  errno is: %s\n",strerror(errno));/* Fatal error */
          sprintf(logmsg, "You can only CONFIG when TRIGGER is ENABLED or READY\n");
          logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Configuration Option %i is INVALID\n",RT->L1_Par.config_opt);
          logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Configuration Option = %i\n",RT->L1_Par.config_opt);
          logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Unable to open L1 configuration file\n");
          logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
            sprintf(logmsg, "MAX_TOKEN_IN_SYSTEM = %i is out of range. Setting to %i\n",
            logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
            sprintf(logmsg, "Maximum token value will be %i\n",new_max_tokens);
            logmessage( LOG_INFO, "L1_Software_Config", logmsg);
        sprintf(logmsg, "Min Token %i must be less than or equal to Max Token %i\n",
            logmessage( LOG_ERR, "L1_Software_Config", logmsg);
        sprintf(logmsg, "Min Token %i must be greater than 0\n",
            logmessage( LOG_ERR, "L1_Software_Config", logmsg);
        sprintf(logmsg, "TM will take tokens starting at %d\n",RT->L1_Par.Min_tkn_in_system);
            logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
            sprintf(logmsg, "%i is INVALID, 1 =< tokenMod\n",RT->L1_Par.tokenMod);
            logmessage( LOG_ERR, "L1_Software_Config", logmsg);
            sprintf(logmsg, "%i is INVALID, tokenMod =< (Max_tkn - Min_tkn)\n",RT->L1_Par.tokenMod);
            logmessage( LOG_ERR, "L1_Software_Config", logmsg);
            sprintf(logmsg, "TM will load every %d th token",RT->L1_Par.tokenMod);
            logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
           sprintf(logmsg, "TCU Base Address = 0x%lx, Config File = %s\n",
           logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
            sprintf(logmsg, "Too many trigger words read from L1_Conf.dat\n");
            logmessage(LOG_ERR, "L1_Software_Config", logmsg);
                  sprintf(logmsg, "DSM %i Base Address = 0x%lx, Config File = %s\n",
                  logmessage( LOG_NOTICE, "L1_Software_Config", logmsg);
                   sprintf(logmsg, "Unable to open DSM configuration file\n");
                   logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
                    sprintf(logmsg, "Address Increment for DSM 0x%x = %i\n",
                    logmessage( LOG_DEBUG, "L1_Software_Config", logmsg);
                          sprintf(logmsg, "Multiplicity Register %i = %i\n",
                          logmessage( LOG_DEBUG, "L1_Software_Config", logmsg);
                          sprintf(logmsg, "Multiplicity Register %i is not allow\n",eng_reg_index);
                          logmessage( LOG_ERR, "L1_Software_Config", logmsg);
                          sprintf(logmsg, "ZDC Register %i = %i\n",
                          logmessage( LOG_DEBUG, "L1_Software_Config", logmsg);
                          sprintf(logmsg, "ZDC Register %i is not allow\n",eng_reg_index);
                          logmessage( LOG_ERR, "L1_Software_Config", logmsg);
                        sprintf(logmsg, "Engine Register %i has been found in DSM
%i\n",eng_reg_index,i);
                        logmessage( LOG_ERR, "L1_Software_Config", logmsg);
                 sprintf(logmsg, "L0dataSrc = DSMs, but NO DSMs are defined\n");
                 logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
            sprintf(logmsg, "logopt 1\n");
            logmessage(LOG_INFO, "L1_Software_Config", logmsg);
            sprintf(logmsg, "logopt 2\n");
            logmessage(LOG_INFO, "L1_Software_Config", logmsg);
            sprintf(logmsg, "logopt 3\n");
            logmessage(LOG_INFO, "L1_Software_Config", logmsg);
            sprintf(logmsg, "Invalid logopt flag, using LOToConsole by default\n");
            logmessage(LOG_ERR, "L1_Software_Config", logmsg);
            sprintf(logmsg, "Invalid LogThreshold flag, using LOG_DEBUG by default\n");
            logmessage(LOG_ERR, "L1_Software_Config", logmsg);
```

```
          sprintf(logmsg, "LogThreshold = %i\n",LogLevelThreshold);
          logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Unrecognized L1 Configuration Parameter %s\n",Config_Name);
          logmessage(LOG_ERR, "L1_Software_Config", logmsg);
        sprintf(logmsg, "Finished reading L1 Configuration File\n");
        logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
        sprintf(logmsg, "Base Address of DC Window = 0x%x\n",RT->L1_Par.DC_Win_Base);
        logmessage(LOG_INFO, "L1_Software_Config", logmsg);
        sprintf(logmsg, "Base Address of CTB Window = 0x%x\n",RT->L1_Par.CB_Win_Base);
        logmessage(LOG_INFO, "L1_Software_Config", logmsg);
         sprintf( logmsg, "No available tokens: Max = %i, Min = %i, Mod = %i, Ntokens = %i\n",
         logmessage( LOG_WARNING, "L1_Software_Config", logmsg);
         sprintf(logmsg, "Sending a message to L1_ANA to config\n");
         logmessage(LOG_INFO, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Sending a message to DC to config\n");
          logmessage(LOG_INFO, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Sending a message to DC to resynch DSMs\n");
          logmessage(LOG_INFO, "L1_Software_Config", logmsg);
         sprintf(logmsg, "Could not spawn L1_Hardware_Config\n");
         logmessage( LOG_EMERG, "L1_Software_Config", logmsg);
         sprintf(logmsg, "Pointer to DC Buffer = 0x%x\n",RT->L1_Par.DC_Pointer);
         logmessage(LOG_INFO, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Sending a message to CTB to config\n");
          logmessage(LOG_INFO, "L1_Software_Config", logmsg);
         sprintf(logmsg, "DONE_CONFIG command received when NO CONFIG command was issued\n");
         logmessage(LOG_INFO, "L1_Software_Config", logmsg);
          sprintf(logmsg, "Sending a message to CTB to resynch\n");
          logmessage(LOG_INFO, "L1_Software_Config", logmsg);
         sprintf(logmsg, "DONE_RESYNCH command received when NO RESYNCH command was issued\n");
         logmessage(LOG_INFO, "L1_Software_Config", logmsg);
   sprintf(logmsg, "Unknown message has been received. %d\n",src_task);
   logmessage(LOG_ERR, "L1_Software_Config", logmsg);
  sprintf(logmsg, "Trigger State is ENABLED\n");
  sprintf(logmsg, "Trigger State is CONFIGURING\n");
  sprintf(logmsg, "Trigger State is READY\n");
  sprintf(logmsg, "Trigger State is RUNNING\n");
  sprintf(logmsg, "Trigger State is PAUSED\n");
  sprintf(logmsg, "Trigger State is STOPPING\n");
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "L1_evb = %i\n",RT->L1_Par.L1_evb);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "CTBcrate = %i\n",RT->L1_Par.CTBcrate);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "L0dataSrc = %i\n",RT->L1_Par.L0dataSrc);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "nL0DSM = %i\n",RT->L1_Par.nL0DSM);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Max_tkns_in_system = %i\n",RT->L1_Par.Max_tkns_in_system);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Minimum token in system = %d \n",RT->L1_Par.Min_tkn_in_system);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Taking every %d th token \n",RT->L1_Par.tokenMod);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Number of available tokens = %i\n",RT->L1_Par.Ntokens_avail);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Tokens_in_use = %i\n",RT->L1_Par.Tokens_in_use);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Events_Taken = %i\n",RT->L1_Par.Events_Taken);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "TCU_Token_Return = %i\n",RT->L1_Par.TCU_Token_Return);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Last_Token_Issued = %i\n",RT->L1_Par.Last_Token_Issued);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Taking every %d th token \n",RT->L1_Par.tokenMod);
```

```
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "DC_Win_Base = 0x%x\n",RT->L1_Par.DC_Win_Base);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "DC_Pointer = 0x%x\n",RT->L1_Par.DC_Pointer);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "CB_Win_Base = 0x%x\n",RT->L1_Par.CB_Win_Base);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "NTW = %i\n",RT->L1_Par.NTW);
logmessage(LOG_NOTICE, "printconfig", logmsg);
  sprintf(logmsg, "Trigger Word=%i, Npre=%i, Npost=%i\n",
  logmessage(LOG_NOTICE, "printconfig", logmsg);
  sprintf(logmsg, "Multiplicity threshold %i = %i\n",i,RT->L1_Par.Mult_Reg[i]);
  logmessage(LOG_NOTICE, "printconfig", logmsg);
  sprintf(logmsg, "ZDC threshold %i = %i\n",i,RT->L1_Par.ZDC_Reg[i]);
  logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "tokenTimer = %i\n",RT->L1_Par.tokenTimer);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "nTokenReturn = %i\n",RT->L1_Par.nTokenReturn);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "config_opt = %i\n",RT->L1_Par.config_opt);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "logOpt = %i\n",RT->L1_Par.logOpt);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "LogThr = %i\n",RT->L1_Par.LogThr);
logmessage(LOG_NOTICE, "printconfig", logmsg);
sprintf(logmsg, "Operations Mode = %i\n",(*Oper_Mode_Addr&CSR1_MASK));
logmessage( LOG_INFO, "TCU_Oper_Mode", logmsg);
sprintf(logmsg, "TCU Local Address = %08x\n",(*Local_Address & CSR3_MASK));
logmessage( LOG_INFO, "TCU_Read_Local_Address", logmsg);
sprintf(logmsg, "Fifo Status = %08x\n",(*FIFO_Status&CSR4_MASK));
logmessage( LOG_DEBUG, "TCU_FIFO_Status", logmsg);
  sprintf(logmsg, "Unable to open TCU file\n");
  logmessage( LOG_EMERG, "TCU_Mem_File", logmsg);
    sprintf(logmsg, "Not enough entries in TCU file\n");
    logmessage( LOG_ERR, "TCU_Mem_File", logmsg);
   sprintf(logmsg, "Wrote %lx , readback %lx , at address %lx\n",
   logmessage( LOG_CRIT, "TCU_Mem_All", logmsg);
        sprintf(logmsg, "Wrote %lx , readback %lx , at address %lx\n",
        logmessage( LOG_CRIT, "TCU_Mem_Bit", logmsg);
  sprintf(logmsg, "Wrote %lx , readback %lx , at address %lx\n",
  logmessage( LOG_CRIT, "TCU_Mem_Entry", logmsg);
   sprintf(logmsg, "Wrote %lx , readback %lx , at address %lx\n",
   logmessage( LOG_CRIT, "TCU_Mem_Function", logmsg);
  sprintf(logmsg, "Wrote %lx, at address %lx\n",data,((long) address) );
  logmessage( LOG_INFO, "TCU_FIFO_Function", logmsg);
sprintf(logmsg, "Wrote %lx , at address %lx\n",data,((long) address) ) ;
logmessage( LOG_INFO, "TCU_FIFO_Entry", logmsg);
  sprintf(logmsg, "Unable to open TCU config file\n");
  logmessage( LOG_EMERG, "TCUinit", logmsg);
       sprintf(logmsg, "Using input connectors\n");
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Simulated data is in file %s\n",SIM_File);
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Simulated data will be generated using %i\n",
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Simulation Memory will be filled with %08x\n",
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Simulation Memory entry %08x will be set %08x\n",
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Simulation Memory entries with address bits %08x
       logmessage( LOG_INFO, "TCUinit", logmsg);
       sprintf(logmsg, "Invalid TCU_INPUT_Mode Flag: %i\n",Config_Val);
       logmessage( LOG_EMERG, "TCUinit", logmsg);
    sprintf(logmsg, "Too many TCU_TWD_LUT lines in Configuration file %s\n"
```

```
    logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "Trigger Word data is in file %s\n",TWD_File);
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Trigger Word data will be generated using %i\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Trigger Word LUT will be filled with %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Trigger Word LUT entry %08x will be set to %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Trigger Word LUT entries with address bits %08x
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_TWD_LUT Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "Pre−scaler data is in file %s\n",PSC_File);
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Pre−scaler data will be generated using %i\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Pre−scale table will be filled with %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Pre−scale table entry %08x will be set to %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Pre−scale table entries with address bits %08x
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_PRE_SC Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "Action Word data is in file %s\n",AWD_File);
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Action Word data will be generated using %i\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Action Word LUT will be filled with %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Action Word LUT entry %08x will be set to %08x\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Action Word LUT table entries with address bits %08x
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_AWD_LUT Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "NO tokens will be loaded into the Priority
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Tokens will be read from file %s\n",
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "%i Priority Tokens will be generated using %i
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_PRI_TKN_FIFO Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "NO tokens will be loaded into the Token FIFO\n");
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Tokens will be read from file %s\n",TKN_File);
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "%i Non−Priority Tokens will be generated using %i
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_TKN_FIFO Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
      sprintf(logmsg, "TCU will be left in LOAD Mode after configuration\n");
      logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "TCU will be put in RUN Mode after configuration\n");
      logmessage( LOG_NOTICE, "TCUinit", logmsg);
      sprintf(logmsg, "Invalid TCU_BOARD_MODE Flag: %i\n",Config_Val);
      logmessage( LOG_CRIT, "TCUinit", logmsg);
  sprintf(logmsg, "Unrecognized Configuration Name %s\n",Config_Name);
  logmessage( LOG_CRIT, "TCUinit", logmsg);
sprintf(logmsg, "TCU Input Mode configuration NOT specified\n");
logmessage( LOG_ALERT, "TCUinit", logmsg);
sprintf(logmsg, "Trigger Word LUT configuration NOT specified\n");
```

```
 logmessage( LOG_ALERT, "TCUinit", logmsg);
 sprintf(logmsg, "Pre-scaler configuration NOT specified\n");
 logmessage( LOG_ALERT, "TCUinit", logmsg);
 sprintf(logmsg, "Action Word LUT configuration NOT specified\n");
 logmessage( LOG_ALERT, "TCUinit", logmsg);
sprintf(logmsg, "Unable to take TCU out of RUN mode\n");
logmessage( LOG_EMERG, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to fill Simulation Memory with zeros\n");
          logmessage( LOG_ERR, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to configure Simulation Memory from %s\n",
     logmessage( LOG_ALERT, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Simulation Memory using function %i\n",
     logmessage( LOG_ALERT, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to fill Simulation Memory with %i\n",
          logmessage( LOG_ALERT, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to set Simulation Memory entry 0x%8x\n",
          logmessage( LOG_ALERT, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to set Simulation Memory Entries with bits 0x%x\n",
          logmessage( LOG_ALERT, "TCUinit", logmsg);
  sprintf(logmsg, "Invalid TCU_INPUT_MODE Flag: %i\n",Config_Input[i]);
  logmessage( LOG_ALERT, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to configure Trigger Word LUT from %s\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Trigger Word LUT using function %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Trigger Word LUT with %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to set Trigger Word LUT entry 0x%8x\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to set Trigger Word LUT entries with bits 0x%x\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
   sprintf(logmsg, "Invalid TCU_TWD_LUT Flag: %i\n",Config_TWD[i]);
   logmessage( LOG_ALERT, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to configure Prescale Counter from %s\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to configure Prescale Table from %s\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Prescale Counter using function %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Prescale Table using function %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Prescale Counter with %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Prescale Table with %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to set Prescale Counter entry 0x%8x\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
      sprintf(logmsg, "Unable to set Prescale Table entry 0x%8x\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to set Prescale Counter entries with bits 0x%x\n",
          logmessage( LOG_EMERG, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to set Prescale Table entries with bits 0x%x\n",
          logmessage( LOG_EMERG, "TCUinit", logmsg);
    sprintf(logmsg, "Invalid TCU_PSC_LUT Flag: %i\n",Config_PSC[i]);
    logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to configure Action Word LUT from %s\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Action Word LUT using function %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to fill Action Word LUT with %i\n",
     logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Unable to set Action Word LUT entry 0x%8x\n",
```

```
      logmessage( LOG_EMERG, "TCUinit", logmsg);
          sprintf(logmsg, "Unable to set Action Word LUT entries with bits 0x%x\n",
          logmessage( LOG_EMERG, "TCUinit", logmsg);
     sprintf(logmsg, "Invalid TCU_AWD_LUT Flag: %i\n",Config_AWD[i]);
     logmessage( LOG_ALERT, "TCUinit", logmsg);
    sprintf(logmsg, "Unable to reset address to zero\n") ;
    logmessage( LOG_EMERG, "TCUinit", logmsg);
    sprintf(logmsg, "Unable to reset FIFOs\n") ;
    logmessage( LOG_EMERG, "TCUinit", logmsg);
   sprintf(logmsg, "Downloading Tokens %lx\n",((long) address0));
   logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Failed to load Priority Token FIFO\n") ;
      logmessage( LOG_EMERG, "TCUinit", logmsg);
   sprintf(logmsg, "Downloading Tokens %lx\n",((long) address0));
   logmessage( LOG_INFO, "TCUinit", logmsg);
      sprintf(logmsg, "Failed to load non-Priority Token FIFO\n") ;
      logmessage( LOG_EMERG, "TCUinit", logmsg);
  sprintf(logmsg, "Unable to put TCU in Mode %i\n",Oper_Cntrl) ;
  logmessage( LOG_EMERG, "TCUinit", logmsg);
 sprintf(logmsg, "Finished configuring TCU\n");
 logmessage( LOG_NOTICE, "TCUinit", logmsg);
            sprintf(logstr, "DSM CBLT Read Address = 0x%lx\n",(long)addr_cblt);
            logmessage( LOG_DEBUG, "L1Analysis",logstr);
            sprintf(logstr, "DSM CBLT Latch Address = 0x%lx\n",(long)addr_latch);
            logmessage( LOG_DEBUG, "L1Analysis",logstr);
              sprintf(logstr, "ERROR - initdma FAILED\n");
              logmessage(LOG_EMERG, "L1Analysis",logstr);
            sprintf( logstr, " Received event with token %i from L1CON \n",cur_token);
            logmessage( LOG_DEBUG, "L1Analysis", logstr);
            sprintf( logstr, "Data will be latched at Input Buffer Address 0x%lx\n",
            logmessage( LOG_DEBUG, "L1Analysis", logstr);
              sprintf(logstr, " buffer points to address 0x%lx\n",(long)buffer);
              logmessage( LOG_DEBUG, "L1Analysis",logstr);
              sprintf(logstr, "Attempting DMA Transfer from 0x%lx to 0x%lx\n",
              logmessage( LOG_DEBUG, "L1Analysis",logstr);
                sprintf(logstr, "DMA transaction failed\n");
                logmessage( LOG_EMERG, "L1Analysis",logstr);
                sprintf(logstr, "DMA Transaction Successfull - ret = %i\n",c);
                logmessage( LOG_DEBUG, "L1Analysis",logstr);
        sprintf( logstr, " Error in filling L0 data \n");
        logmessage( LOG_ERR, "L1Analysis", logstr);
              sprintf( logstr, "Attempting to send data to address 0x%x\n", \
       logmessage ( LOG_DEBUG, "L1Analysis", logstr );
        sprintf( logstr," SCI Data Send ERROR %d for token %d \n",err,cur_token);
        logmessage ( LOG_ERR, "L1ANA", logstr);
              sprintf( logstr,"Sent token %i to Event Builder \n",cur_token);
       logmessage ( LOG_DEBUG, "L1Analysis", logstr);
                sprintf( logstr,"Sent token %i to L1CTB \n",cur_token);
         logmessage ( LOG_DEBUG, "L1Analysis", logstr);
      sprintf( logstr,  "Unknown message from L1Control \n" );
      logmessage( LOG_ERR, "L1Analysis", logstr);
    sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
    logmessage ( LOG_ERR, "L1Analysis get_message", logstr);
             sprintf( logstr, "Sending event with token %i to L1ANA \n",cur_token);
             logmessage( LOG_DEBUG, "L1Control", logstr);
      sprintf( logstr, "Unknown message from L2L1Interface \n");
      logmessage( LOG_ERR, "L1Control", logstr);
    sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
    logmessage( LOG_ERR, "L1Control get_message", logstr);
      sprintf( logstr, "Unknown message from L1 Control \n" );
      logmessage( LOG_ERR, "L1QueManager", logstr);
     sprintf( logstr,  "Unknown message from L1 Analysis \n" );
     logmessage( LOG_ERR, "L1QueManager", logstr);
```

```
     sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
     logmessage ( LOG_ERR, "L1QueManager", logstr );
     sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno));/* Fatal error */
     logmessage( LOG_ERR, "L1_Hardware_Interface get_message", logstr);
     sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno));/* Fatal error */
     logmessage( LOG_ERR, "L1_Hardware_Interface num_message", logstr);
   sprintf(logstr, "TCU Base Address = 0x%lx\n",(long)RT->L0.TCU_Address);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "Token FIFO Address = 0x%lx\n",(long)Token_FIFO);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "Response FIFO Address = 0x%lx\n",(long)Response_FIFO);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "RESET FIFO Address = 0x%lx\n",(long)Reset_FIFO);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "FIFO Status Address = 0x%lx\n",(long)FIFO_Status);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "INFO_FIFO1 Address = 0x%lx\n",(long)INFO_FIFO1);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "INFO_FIFO2 Address = 0x%lx\n",(long)INFO_FIFO2);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
   sprintf(logstr, "INFO_FIFO3 Address = 0x%lx\n",(long)INFO_FIFO2);
   logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
            sprintf(logstr, "Received command LOAD_TKN %d from TM\n", cur_token);
            logmessage( LOG_INFO, "L1_Hardware_Interface", logstr);
            sprintf(logstr, "Received command LOAD_ABORT of token %i from TM\n",cur_token);
            logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
            sprintf(logstr, "Sending ABORT command 0x%lx to Response FIFO\n",abort_response);
            logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
            sprintf(logstr, "Received command STOP_RUN from TM\n");
            logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
             sprintf( logstr, "Failed to Clear TCU Token FIFOs at STOP_RUN\n");
             logmessage( LOG_ERR, "L1_Hardware_Interface", logstr);
       sprintf( logstr, " Unknown command has been received %d\n",mptr->cmd);
       logmessage( LOG_ERR, "L1_Hardware_Interface", logstr);
      sprintf( logstr,"Unknown message has been received. %d\n",src_task);
      logmessage( LOG_ERR, "L1_Hardware_Interface", logstr);
     sprintf(logstr, "Got an Event\n");
     logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
          sprintf( logstr, "ABORT Issued, sending RELEASE of token %i to TM\n",cur_token);
          logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
           sprintf( logstr,"Error in filling Event Descriptor \n");
           logmessage( LOG_ERR, "L1_Hardware_Interface", logstr);
          sprintf( logstr, " Sending Event with token %i to L1CON\n",cur_token);
          logmessage( LOG_DEBUG, "L1_Hardware_Interface", logstr);
    sprintf( logstr, "Trigger Word %i not found in list. Setting npre/npost = 0\n",
    logmessage ( LOG_WARNING, "L1_HI fillEventDesc", logstr );
   sprintf( logstr, "token=%d aw=0x%x dsm_data=0x%x dsm_adr=0x%x busy=0x%x pri=0x%x tw=0x%x npre=%d
 npost=%d\n",
   logmessage ( LOG_DEBUG, "L1_HI fillEventDesc", logstr );
    sprintf(logstr, "Sent command CONFIG to L1_SC\n");
    logmessage( LOG_NOTICE, "L1_IO", logstr);
    sprintf(logstr, "Sent command START_RUN to TM with %i events \n",nev);
    logmessage( LOG_NOTICE, "L1_IO", logstr);
    sprintf(logstr, "Sent command ADD_TOKEN to TM with %i events \n",nev);
    logmessage( LOG_NOTICE, "L1_IO", logstr);
    sprintf(logstr, "Sent command STOP_RUN to TM\n");
    logmessage( LOG_NOTICE, "L1_IO", logstr);
    sprintf(workName, "tTcpWork%d", ii++);
     sprintf(logstr,"Failed to create message queue %i\n",nq);
     logmessage ( LOG_ERR, "L1_Startup",logstr);
   sprintf(logstr, "Tokens zeroed\n");
   logmessage( LOG_NOTICE, "L1_Startup", logstr);
   sprintf(logstr, "event counter zeroed\n");
   logmessage( LOG_NOTICE, "L1_Startup", logstr);
```

```
sprintf(logstr, "Now in READY STATE \n");
logmessage( LOG_NOTICE, "L1_Startup", logstr);
sprintf(workName, "tTokWork%d", ii++);
sprintf(logstr, "Serious error received:  errno is: %s\n",strerror(errno));/* Fatal error */
logmessage (LOG_ERR, "L1_Token_Return get_message", logstr );
sprintf(logstr, "Trigger is NOT READY. You can only START RUN when TRIGGER is READY\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr);
sprintf( logstr, "Zero is an INVALID number of events. RUN NOT STARTED.\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr );
sprintf( logstr, "Starting run for %i events\n " ,nevents);
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
sprintf( logstr, "Error in Token Counter Initialization\n ");
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
sprintf( logstr, "Error in Event Counter Initialization\n ");
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
sprintf( logstr, "Error in Timer");
logmessage( LOG_WARNING, "L1_Token_Manager", logstr );
sprintf( logstr, "No available tokens: Max = %i, Min = %i, Mod = %i, Ntokens = %i\n",
logmessage( LOG_ERR, "L1_Token_Manager", logstr);
sprintf( logstr, "Number of available tokens = %i\n",RT->L1_Par.Ntokens_avail);
logmessage( LOG_INFO, "L1_Token_Manager", logstr);
sprintf( logstr, "tokens %i = %i\n",i,RT->L1_Par.tokens_avail[i]);
logmessage( LOG_DEBUG, "L1_Token_Manager", logstr);
sprintf(logstr, "Tokens will NOT be recycled\n");
logmessage(LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Tokens WILL be recycled\n");
logmessage(LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Now in RUNNING STATE \n");
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Now in STOPPING STATE \n");
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Sending command STOP_RUN to Hardware Interface\n");
logmessage( LOG_INFO, "L1_Token_Manager", logstr);
sprintf (logstr,"looking for token %d from DC processor - stat %d\n",kk,L1token[kk].status);
logmessage(LOG_NOTICE, "L1TM", logstr);
sprintf(logstr,"all tokens are here \n");
logmessage(LOG_NOTICE,"L1TM",logstr);
sprintf(logstr, "Now in READY STATE \n");
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Trigger State = RUNNING. You CANNOT ADD TOKENS unless TRIGGER is
PAUSED\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr);
sprintf(logstr, "Trigger State = READY. You CANNOT ADD TOKENS unless TRIGGER is
PAUSED\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr);
sprintf(logstr, "Trigger State = STOPPING. You CANNOT ADD TOKENS unless TRIGGER is
PAUSED\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr);
sprintf( logstr, "Zero is an INVALID number of events. Tokens NOT ADDED.\n");
logmessage( LOG_ERR, "L1_Token_Manager", logstr );
sprintf( logstr, "Adding enough tokens to take %i more events\n",nevents);
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Tokens will NOT be recycled\n");
logmessage(LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf(logstr, "Tokens WILL be recycled\n");
logmessage(LOG_NOTICE, "L1_Token_Manager", logstr);
sprintf( logstr, "Received token %i back from L1DC Token Manager \n",cur_token);
logmessage( LOG_INFO, "L1_Token_Manager", logstr );
sprintf( logstr, "Error in Timer");
logmessage( LOG_WARNING, "L1_Token_Manager", logstr );
sprintf(logstr, "Taken %i events, moving to PAUSED state\n",RT->L1_Par.Events_Taken);
logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
sprintf( logstr,"Wrote token %i to HI Queue \n",cur_token);
```

```
            logmessage( LOG_DEBUG, "L1_Token_Manager", logstr );
        sprintf( logstr, "Received L2_ABORT of token %i back from L1DC Token Manager \n",cur_token);
        logmessage( LOG_INFO, "L1_Token_Manager", logstr );
        sprintf( logstr, "Sent ABORT of token %i to HI Queue \n",cur_token);
        logmessage( LOG_DEBUG, "L1_Token_Manager",logstr );
        sprintf(logstr, "Unknown command received for L1TR: %c\n",mptr->cmd);
        logmessage( LOG_ERR, "L1_Token_Manager", logstr );
        sprintf( logstr, "Received token %i back from L1CTL \n",cur_token);
        logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
            sprintf( logstr, "Error in Timer");
            logmessage( LOG_WARNING, "L1_Token_Manager", logstr );
            sprintf( logstr,"Wrote token %i to HI Queue \n",cur_token);
            logmessage( LOG_DEBUG, "L1_Token_Manager", logstr );
        sprintf( logstr, "Received L1_ABORT of token %i back from L1 Analysis \n",cur_token);
        logmessage( LOG_INFO, "L1_Token_Manager", logstr );
        sprintf( logstr, "Sent ABORT of token %i to HI Queue \n",cur_token);
        logmessage( LOG_DEBUG, "L1_Token_Manager",logstr );
        sprintf(logstr, "Unknown command received from L1ANA: %c\n",mptr->cmd);
        logmessage( LOG_ERR, "L1_Token_Manager", logstr );
        sprintf( logstr, "Received aborted token %i back from L1_HI\n",cur_token);
        logmessage( LOG_NOTICE, "L1_Token_Manager", logstr );
            sprintf( logstr,"Wrote token %i to HI Queue \n",cur_token);
            logmessage( LOG_DEBUG, "L1_Token_Manager", logstr );
        sprintf(logstr, "Unknown command received from L1_HI: %c\n",mptr->cmd);
        logmessage( LOG_ERR, "L1_Token_Manager", logstr );
    sprintf(logstr,"message from DCTM\n");
    logmessage(LOG_DEBUG,"L1TM",logstr);
             sprintf(logstr, "token %d OK in DCTM\n",cur_token);
             logmessage(LOG_NOTICE,"L1TM",logstr);
            sprintf(logstr, "ERROR: token %d L2 status %d L1CTL status
%d\n",cur_token,msgbuf[3],tokensOut);
            logmessage(LOG_NOTICE,"L1TM",logstr);
                sprintf(logstr,"all tokens are now here \n");
                logmessage(LOG_NOTICE,"L1TM",logstr);
                    sprintf(logstr, "Now in READY STATE \n");
                    logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
        sprintf(logstr, "DCTM ERR for token %d status %d\n",cur_token,msgbuf[3]);
        logmessage(LOG_NOTICE,"L1TM",logstr);
                sprintf(logstr, "ERROR: token %d L2 status %d L1CTL status
%d\n",cur_token,msgbuf[3],tokensOut);
                logmessage(LOG_NOTICE,"L1TM",logstr);
                    sprintf(logstr,"all tokens are now here \n");
                    logmessage(LOG_NOTICE,"L1TM",logstr);
                        sprintf(logstr, "Now in READY STATE \n");
                        logmessage( LOG_NOTICE, "L1_Token_Manager", logstr);
        sprintf(logstr, "unknown command from DCTM \n");
        logmessage(LOG_DEBUG,"L1TM",logstr);
    sprintf( logstr, "Unknown message has been received. %d\n",src_task);
    logmessage( LOG_DEBUG, "L1_Token_Manager", logstr);
   sprintf(logstr,"clock start %2f \n",startTime);
   logmessage( LOG_NOTICE, "L1TM", logstr );
   sprintf(logstr,"clock stop %2f \n", trgTime);
   logmessage( LOG_NOTICE, "L1TM", logstr );
   sprintf(logstr,"clock running %2f \n", trgTime);
   logmessage( LOG_NOTICE, "L1TM", logstr );
   sprintf(logstr,"waiting for DAQ to finish...\n");
   logmessage(LOG_NOTICE, "L1TM", logstr);
      sprintf (logstr,"Will have to wait for token %d from DC processor - stat %d\n",ii,L1token[ii].status);
      logmessage(LOG_DEBUG, "L1TM", logstr);
       sprintf (logstr,"token %d being overwritten \n",ii);
       logmessage(LOG_ERR, "L1TM", logstr);
    sprintf (logstr,"Will have to wait for %d tokens from DC processor \n",jj);
    logmessage(LOG_NOTICE, "L1TM", logstr);
   sprintf (logstr,"all tokens here \n");
```

logmessage(LOG_NOTICE, "L1TM", logstr);
sprintf (logstr,"token %d not in L1CTL – stat %d \n",whichToken,L1token[whichToken].status);
logmessage(LOG_DEBUG, "L1TM", logstr);
sprintf (logstr,"token %d stat %d OK \n",whichToken,L1token[whichToken].status);
logmessage(LOG_DEBUG, "L1TM", logstr);
sprintf(logmsg, "Trigger is NOT in the READY State, Max_tkns_in_system is UNCHANGED\n");
logmessage(LOG_ERR, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is TOO SMALL. Max_tkns_in_system is UNCHANGED\n",new_max_token);
logmessage(LOG_ERR, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is TOO big. Setting to %i\n",new_max_token,MAX_TOKEN);
logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
sprintf(logmsg, "Maximum token value will be %i\n",new_max_token);
logmessage(LOG_INFO, "L1_Software_Config", logmsg);
sprintf(logmsg, "Trigger is NOT in the READY State, Min_tkn is UNCHANGED\n");
logmessage(LOG_ERR, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is bigger than Max–TKns. Min_tkn is UNCHANGED\n",new_min_token);
logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is smaller than 1. Min_tkn is UNCHANGED\n",new_min_token);
logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
sprintf(logmsg, "TM will now start taking tokens at %i\n",new_min_token);
logmessage(LOG_INFO, "L1_Software_Config", logmsg);
sprintf(logmsg, "Trigger is NOT in the READY State, TokenMod is UNCHANGED\n");
logmessage(LOG_ERR, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is INVALID. TokenMod is UNCHANGED\n",new_mod_token);
logmessage(LOG_ERR, "L1_Software_Config", logmsg);
sprintf(logmsg, "%i is TOO big. TokenMod is UNCHANGED\n",new_mod_token);
logmessage(LOG_NOTICE, "L1_Software_Config", logmsg);
sprintf(logmsg, "TM will now take every %i th token\n",new_mod_token);
logmessage(LOG_INFO, "L1_Software_Config", logmsg);

L1CTB

sprintf(logstr, "Failed to configure DSM %i – ERROR = %i\n",i,err);
logmessage(LOG_ERR, "CTB_Hard_Conf",logstr);
sprintf(logstr, "Unable to configure DSM %i – no base address\n",i);
logmessage(LOG_ERR, "CTB_Hard_Conf",logstr);
sprintf(logstr,"Finished with Configuration of CTB DSMs\n");
logmessage(LOG_NOTICE,"CTB_HC",logstr);
sprintf(logstr, "Received message from L1IO to configure \n");
logmessage(LOG_DEBUG, "L1CTBConfig", logstr);
sprintf(logstr, "L1CTB about to go from %d state –> Configuring state \n",L1CTBState);
logmessage(LOG_NOTICE, "L1CTBConfig", logstr);
sprintf(logstr, "Pointer to DC Buffer = 0x%x\n",CTBc.DCdataPtr);
logmessage(LOG_INFO, "L1CTBConfig", logstr);
sprintf(logstr, "Could not spawn CTB_Hardware_Config\n");
logmessage(LOG_EMERG, "L1CTBConfig", logstr);
sprintf(logstr, "L1CTB is in %d state \n",L1CTBState);
logmessage(LOG_EMERG, "L1CTBConfig", logstr);
sprintf(logstr, " L1CTBConfig received message from L1_SC to resynchronize \n");
logmessage(LOG_NOTICE,"L1CTBConfig",logstr);
sprintf(logstr, "L1CTB about to go from %d state –> Configuring state \n",L1CTBState);
logmessage(LOG_NOTICE, "L1CTBConfig", logstr);
sprintf(logstr, "L1CTB now in READY state \n");
logmessage(LOG_NOTICE, "L1CTBConfig", logstr);
sprintf(logstr, "L1CTB is in %d state \n",L1CTBState);
logmessage(LOG_EMERG, "L1CTBConfig", logstr);
sprintf(logstr, "Unknown message has been received. %d\n",src_task);
logmessage(LOG_ERR, "L1CTBConfig", logstr);
sprintf(logstr,"ERROR – DMA initialization failed \n");
logmessage(LOG_EMERG,"L1CTBConfig",logstr);
sprintf(logstr, "DMA Initialized with %d bytes \n",nDSMbytes);
logmessage(LOG_DEBUG,"L1CTBConfig",logstr);

```
      sprintf (logstr,
      logmessage( LOG_CRIT, "L1CTBRdConfig", logstr);
   sprintf(logstr, "DSM boards %d,log %d \n",CTBc.nCTBboards,CTBc.logOpt);
   logmessage( LOG_DEBUG, "L1CTBRdConfig", logstr);
      sprintf(logstr,"Serious error received:  errno is: %s\n",strerror(errno));
      logmessage(LOG_ERR,"L1CTBConf",logstr);
            sprintf(logstr, "Unable to reset address to zero, value = 0x%i\n",addr) ;
            logmessage( LOG_EMERG, "L1CTBResynch", logstr);
             sprintf(logstr, "Failed to increment address by %i: %i\n",CTBc.DSM_Addr_Incr[i],addr) ;
             logmessage( LOG_EMERG, "L1CTBResynch", logstr);
      sprintf(logstr, "Unable to resynchronize DSM %i - no base address\n",i);
      logmessage(LOG_ERR, "L1CTBResynch",logstr);
   sprintf(logstr,"Finished with Resynchronization of CTB DSMs\n");
   logmessage(LOG_NOTICE,"L1CTBResynch",logstr);
       sprintf(logstr, "Failed to configure DSM %i - ERROR = %i\n",i,err);
       logmessage(LOG_ERR, "CTB_Hard_Conf",logstr);
      sprintf(logstr, "Unable to configure DSM %i - no base address\n",i);
      logmessage(LOG_ERR, "CTB_Hard_Conf",logstr);
   sprintf(logstr,"Finished with Configuration of CTB DSMs\n");
   logmessage(LOG_NOTICE,"CTB_HC",logstr);
   sprintf(logstr, "L1CTB now in READY state \n");
   logmessage( LOG_NOTICE, "L1CTBConfig", logstr);
         sprintf(logstr,"Payload from L1ANA: DSMAdr %x, Npre %d, Npost %d \n",DSMAdr[0],Npre, Npost);
         logmessage(LOG_DEBUG,"L1CTBCTL",logstr);
            sprintf(logstr,"Latching addresses - address 0x%8.8x %x",(int)CTBc.DSMBaseAdr,DSMAdr[ii]);
            logmessage(LOG_DEBUG,"L1CTBCTL",logstr);
            sprintf(logstr,"Reading - reg 0x%8.8x address 0x%8.8x",(int)DSM_READ_REG,(int)CTB);
            logmessage(LOG_DEBUG,"L1CTBCTL",logstr);
             sprintf( logstr, "Error DMA read of CTB \n" );
             logmessage ( LOG_EMERG, "L1CTBCTL", logstr );
          sprintf( logstr,"CTB  0 0x%2.2x 1 0x%2.2x 2 0x%2.2x 3 0x%2.2x 4 0x%2.2x 5 0x%2.2x
\n",CTB[0],CTB[1], CTB[2], CTB[3], CTB[4], CTB[5]);
          logmessage( LOG_DEBUG, "L1Control", logstr );
          sprintf( logstr, "Attempting to send data to address 0x%x\n", \
          logmessage ( LOG_DEBUG, "L1Control", logstr );
            sprintf( logstr," SCI Data Send ERROR %d for token %d \n",err,cur_token);
            logmessage ( LOG_ERR, "L1CTBControl", logstr);
         sprintf( logstr," L1CTBControl: Sent token %i to Event Builder \n",cur_token);
         logmessage ( LOG_DEBUG, "L1CTBControl", logstr);
          sprintf( logstr,  "Unknown message from L1ANA \n" );
          logmessage( LOG_ERR, "L1CTB", logstr);
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage( LOG_ERR, "L1CTBCTL get_message", logstr);
    sprintf(logstr, "L1CTB about to go from Disabled -> Enables state \n");
    logmessage( LOG_NOTICE, "L1CTBIO", logstr);
          sprintf(logstr, "Could not add host dccctl \n");
          logmessage( LOG_EMERG, "L1CTBIO", logstr);
          sprintf(logstr, "dccctl added to host table \n");
          logmessage( LOG_NOTICE, "L1CTBIO", logstr);
          sprintf(logstr, "Could not add host l1ctl \n");
          logmessage( LOG_EMERG, "L1DCIO", logstr);
          sprintf(logstr, "l1ctl added to host table \n");
          logmessage( LOG_NOTICE, "L1DCIO", logstr);
          sprintf(logstr, "ERROR in L1_NodeId \n");
          logmessage( LOG_EMERG, "L1CTBIO", logstr);
          sprintf(logstr, "L1_NodeId defined %d \n",L1_NodeId);
          logmessage( LOG_NOTICE, "L1CTBIO", logstr);
          sprintf(logstr, "ERROR in DC_NodeId \n");
          logmessage( LOG_EMERG, "L1CTBIO", logstr);
          sprintf(logstr, "DC_NodeId defined %d \n", DC_NodeId);
          logmessage( LOG_NOTICE, "L1CTBIO", logstr);
          sprintf(logstr, "ERROR in CB_NodeId \n");
          logmessage( LOG_EMERG, "L1CTBIO", logstr);
```

```
        sprintf(logstr, "CB_NodeId defined %d \n", CB_NodeId);
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not initialize SCI \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "SCI initialized successfully \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not map SCI DC node \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "SCI DC node mapped \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not map SCI L1 node \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "SCI L1 node mapped \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not create SCI message Q's \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "mesagge queues set up \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not spawn L1CTBConfig \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "Spawned L1CTBConfig \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not spawn L1CTBControl \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "Spawned L1CTBControl \n");
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
    sprintf(logstr, "L1CTB now in ENABLED state \n");
    logmessage( LOG_NOTICE, "L1CTBIO", logstr);
    sprintf(logstr, "L1CTB Not in DISABLED state!! State is %d \n",L1CTBState);
    logmessage( LOG_ERR, "L1CTBIO", logstr);
   sprintf(logstr, "received CONFIG command from ONL – nothing to do for now \n");
   logmessage( LOG_NOTICE, "L1CTBIO", logstr);
   sprintf(logstr, "received START command from ONL – nothing to do for now \n");
   logmessage( LOG_NOTICE, "L1CTBIO", logstr);
   sprintf(logstr, "received STOP command from ONL – nothing to do for now \n");
   logmessage( LOG_NOTICE, "L1CTBIO", logstr);
   sprintf(logstr, "received PAUSE command from ONL – nothing to do for now \n");
   logmessage( LOG_NOTICE, "L1CTBIO", logstr);
   sprintf(logstr, "received RESUME command from ONL – nothing to do for now \n");
   logmessage( LOG_NOTICE, "L1CTBIO", logstr);
     sprintf(logstr, " init_test_Q: Failed to create message queue %i\n",nq);
     logmessage ( LOG_ERR, "L1CTB_Startup",logstr);
```

L1DC

```
     sprintf(logstr, " L1DCConfig received message from DCIO to configure \n");
     logmessage(LOG_DEBUG,"L1DCConf",logstr);
     sprintf(logstr, " L1DCConfig received message from L1_SC to configure \n");
     logmessage(LOG_DEBUG,"L1DCConf",logstr);
       sprintf(logstr, "L1DC about to go from %d state -> Configuring state \n",L1DCState);
       logmessage( LOG_NOTICE, "L1DCConfig", logstr);
         sprintf(logstr, "Could not initialize DC memory \n");
         logmessage( LOG_NOTICE, "L1DCConf", logstr);
         sprintf(logstr, "DC memory initialized \n");
         logmessage( LOG_NOTICE, "L1DCConf", logstr);
         sprintf(logstr, "Could not initialize DC tokens \n");
         logmessage( LOG_NOTICE, "L1DCConf", logstr);
         sprintf(logstr, "DC tokens initialized \n");
         logmessage( LOG_NOTICE, "L1DCConf", logstr);
       sprintf( logstr," L1DCConfig: will send 0x%8.8x to L1IO \n", msgbuf[3]);
       logmessage(LOG_DEBUG,"L1DCCOnf",logstr);
            sprintf(logstr, "Could not spawn MWC_Hardware_Config\n");
```

```
          logmessage( LOG_EMERG, "L1DCConf", logstr);
         sprintf(logstr, "L1DC is in %d state \n",L1DCState);
         logmessage(LOG_EMERG, "L1DCConfig", logstr);
    sprintf(logstr, " L1DCConfig received message from L1_SC to resynchronize \n");
    logmessage(LOG_NOTICE,"L1DCConf",logstr);
     sprintf(logstr, "L1DC about to go from %d state -> Configuring state \n",L1DCState);
     logmessage( LOG_NOTICE, "L1DCConfig", logstr);
      sprintf(logstr, "Could not initialize DC memory \n");
      logmessage( LOG_NOTICE, "L1DCConf", logstr);
      sprintf(logstr, "DC memory initialized \n");
      logmessage( LOG_NOTICE, "L1DCConf", logstr);
      sprintf(logstr, "Could not initialize DC tokens \n");
      logmessage( LOG_NOTICE, "L1DCConf", logstr);
      sprintf(logstr, "DC tokens initialized \n");
      logmessage( LOG_NOTICE, "L1DCConf", logstr);
     sprintf(logstr, "L1DC now in READY state \n");
     logmessage( LOG_NOTICE, "L1DCConfig", logstr);
         sprintf(logstr, "L1DC is in %d state \n",L1DCState);
         logmessage(LOG_EMERG, "L1DCConfig", logstr);
   sprintf(logstr, "Unknown message has been received. %d\n",src_task);
   logmessage(LOG_ERR, "L1DCConf", logstr);
      sprintf(logstr,"ERROR - DMA initialization failed \n");
      logmessage(LOG_EMERG,"L1DCConfig",logstr);
 sprintf(logstr, "boards %d,trgsource %d,dwrite %d,log %d
\n",L1DCc.nMWCboards,L1DCc.DCtrgSrc,L1DCc.DCdataWrite,L1DCc.logOpt);
 logmessage( LOG_INFO, "L1DCRdConfig", logstr);
   sprintf(logstr,"Serious error received:  errno is: %s\n",strerror(errno));
   logmessage(LOG_ERR,"L1DCConf",logstr);
        sprintf(logstr, "Failed to configure DSM %i - ERROR = %i\n",i,err);
        logmessage(LOG_ERR, "MWC_Hard_Conf",logstr);
   sprintf(logstr, "Unable to configure DSM %i - no base address\n",i);
   logmessage(LOG_ERR, "MWC_Hard_Conf",logstr);
 sprintf(logstr,"Finished with Configuration of MWC DSMs\n");
 logmessage(LOG_NOTICE,"MWC_Hard_Conf",logstr);
 sprintf(logstr, "L1DC now in READY state \n");
 logmessage( LOG_NOTICE, "L1DCConfig", logstr);
        sprintf(logstr, "Unable to reset address to zero, value = 0x%i\n",addr) ;
        logmessage( LOG_EMERG, "L1DCResynch", logstr);
         sprintf(logstr, "Failed to increment address by %i: %i\n",L1DCc.DSM_Addr_Incr[i],addr) ;
         logmessage( LOG_EMERG, "L1DCResynch", logstr);
   sprintf(logstr, "Unable to resynchronize DSM %i - no base address\n",i);
   logmessage(LOG_ERR, "L1DCResynch",logstr);
 sprintf(logstr,"Finished with Resynchronization of MWC DSMs\n");
 logmessage(LOG_NOTICE,"L1DCResynch",logstr);
   sprintf(logstr,"L1_IO: Sent command START_RUN to TM with %i tokens \n", Ntokens);
   logmessage(LOG_DEBUG, "L1DCIO",logstr);
   sprintf(logstr,"L1_IO: Sent command ADD_TOKEN to TM with %i tokens \n", Ntokens);
   logmessage(LOG_DEBUG, "L1DCIO",logstr);
   sprintf(logstr,"L1_IO: Sent command STOP_RUN to TM\n");
   logmessage(LOG_DEBUG,"L1DCIO",logstr);
   sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno))
   logmessage( LOG_ERR, "L1DCIO get_message", logstr);
  sprintf(logstr, "L1DC about to go from Disabled -> Enables state \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
      sprintf(logstr, "Could not add host l1ctl \n");
      logmessage( LOG_EMERG, "L1DCIO", logstr);
      sprintf(logstr, "l1ctl added to host table \n");
      logmessage( LOG_NOTICE, "L1DCIO", logstr);
      sprintf(logstr, "Could not add host ctbctl \n");
      logmessage( LOG_EMERG, "L1DCIO", logstr);
      sprintf(logstr, "ctbctl added to host table \n");
      logmessage( LOG_NOTICE, "L1DCIO", logstr);
      sprintf(logstr, "ERROR in L1_NodeId \n");
```

```
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "L1_NodeId defined %d \n",L1_NodeId);
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "ERROR in DC_NodeId \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "DC_NodeId defined %d \n", DC_NodeId);
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "ERROR in CB_NodeId \n");
        logmessage( LOG_EMERG, "L1CTBIO", logstr);
        sprintf(logstr, "CB_NodeId defined %d \n", CB_NodeId);
        logmessage( LOG_NOTICE, "L1CTBIO", logstr);
        sprintf(logstr, "Could not initialize SCI \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "SCI initialized successfully \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not map SCI L1 node \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "SCI L1 node mapped \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not map SCI CB node \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "SCI CB node mapped \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not create SCI message Q's \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "mesagge queues set up \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not initialize SCRAMnet \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "SCRAMnet initialized \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not set up SCRAMnet interrupts \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Interrupts set up \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn L1DCConfig \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned L1DCConfig \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn DCTokenManager \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned DCTokenManager \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn L1EventBuilder \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned L1EventBuilder \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn L1L2Interface \n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned L1L2Interface \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn L1TokenReturn\n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned L1TokenReturn \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
        sprintf(logstr, "Could not spawn L1DCTokenMonitor\n");
        logmessage( LOG_EMERG, "L1DCIO", logstr);
        sprintf(logstr, "Spawned L1DCTokenMonitor \n");
        logmessage( LOG_NOTICE, "L1DCIO", logstr);
    sprintf(logstr, "L1DC now in ENABLED state \n");
    logmessage( LOG_NOTICE, "L1DCIO", logstr);
    sprintf(logstr, "L1DC Not in DISABLED state!! State is %d \n",L1DCState);
    logmessage( LOG_ERR, "L1DCIO", logstr);
```

```
  sprintf(logstr, "received CONFIG command from ONL – nothing to do for now \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
  sprintf(logstr, "received START command from ONL – nothing to do for now \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
  sprintf(logstr, "received STOP command from ONL – nothing to do for now \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
  sprintf(logstr, "received PAUSE command from ONL – nothing to do for now \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
  sprintf(logstr, "received RESUME command from ONL – nothing to do for now \n");
  logmessage( LOG_NOTICE, "L1DCIO", logstr);
  sprintf(logstr,"L1DCTM initialization done \n");
  logmessage(LOG_DEBUG,"L1DCTM",logstr);
        sprintf(logstr, " L1DCTM: ntokens more than allowed MAX_TOKENS – %d \n",Ntokens);
        logmessage(LOG_ERR,"L1DCTM",logstr);
       sprintf( logstr, "L1DCTM: Starting run with tokens 1 – %i\n " ,Ntokens);
       logmessage( LOG_DEBUG, "L1DCTokenManager", logstr );
         sprintf(logstr," L1DCTokenManager about to send token %d to L1EventBuilder \n",sendToken );
         logmessage(LOG_DEBUG,"L1DCTM",logstr);
        sprintf (logstr," L1DCTokenManager: received token zero from Token Return \n");
        logmessage(LOG_ERR,"L1DCTM",logstr);
        sprintf(logstr," L1DCTokenManager: sent event to event builder\n");
        logmessage(LOG_DEBUG,"L1DCTM",logstr);
      sprintf(logstr,"message from L1TM\n");
      logmessage(LOG_DEBUG,"L1DCTM",logstr);
   sprintf(logstr," getToken loop at %d \n",ii);
   logmessage(LOG_DEBUG,"L1DCTM getToken",logstr);
    sprintf(logstr," getToken mockTokenFifo[%d] is %d \n",ii,mockTokenFifo[ii]);
    logmessage(LOG_DEBUG,"L1DCTM getToken",logstr);
    sprintf(logstr," getToken mockTokenFifo[%d] is %d \n",ii,mockTokenFifo[ii]);
    logmessage(LOG_DEBUG,"L1DCTM getToken",logstr);
   sprintf(logstr," returnToken error – token %d alreay there – %d \n",xtoken, mockTokenFifo[xtoken] );
   logmessage(LOG_DEBUG,"L1DCTM tokemToFifo",logstr);
    sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno))
    logmessage( LOG_ERR, "L1DCIO get_message", logstr);
  sprintf(logstr,"looking for token %d ... \n",tokenFind);
  logmessage(LOG_NOTICE, "DCTM", logstr);
   sprintf(logstr,"from L1 %d to L2 %d from L2 %d to L1 %d
\n",DCtoken[tokenFind].statCount[DC_FROM_L1ANA],DCtoken[tokenFind].statCount[DC_TO_L2],DCtoken[toke
nFind].statCount[DC_FROM_L2],DCtoken[tokenFind].statCount[DC_TO_L1]);
  logmessage(LOG_DEBUG, "DCTM", logstr);
   sprintf(logstr, "not set up for this configuration \n");
   logmessage(LOG_NOTICE,"DCTM",logstr);
   sprintf(logstr, "configuration unknown \n");
   logmessage(LOG_NOTICE,"DCTM",logstr);
   sprintf(tokenWorkName, "tTokWork%d", ii++);
    sprintf(logstr,"ERROR creating watchdog timer\n");
    logmessage(LOG_ERR,"L1EVB", logstr);
    sprintf(logstr, "MSG from L1ANA – EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
     sprintf(logstr,"EVBStat[%d] is %d \n",ii,EVBStat[ii]);
     logmessage( LOG_DEBUG, "L1EVB", logstr );
     sprintf(logstr, "token recieved from L1ANA is %d\n", curToken);
     logmessage( LOG_INFO, "L1EventBuilder", logstr );
       sprintf(logstr, "token recieved from L1ANA is ZERO \n");
       logmessage( LOG_ERR, "L1EventBuilder", logstr );
       sprintf(logstr,"Payload from L1ANA: DSMAdr %x, Npre %d, Npost %d \n",DSMAdr[0],Npre, Npost);
       logmessage(LOG_DEBUG,"L1DC",logstr);
          sprintf( logstr, "rawDataFill ERROR \n" );
          logmessage( LOG_ERR, "L1EventBuilder", logstr);
         sprintf( logstr," Reading data from DSM boards \n" );
         logmessage( LOG_DEBUG, "L1EventBuilder", logstr);
         sprintf(logstr,"Latching addresses – address 0x%8.8x %x",(uint *)L1DCc.DSMBaseAdr,DSMAdr[ii]);
         logmessage(LOG_DEBUG,"L1EVB",logstr);
         sprintf(logstr,"Reading – reg 0x%8.8x address 0x%8.8x",DSM_READ_REG,buffer);
```

```
        logmessage(LOG_DEBUG,"L1EVB",logstr);
         sprintf( logstr, "Error DMA read of MWC \n" );
         logmessage ( LOG_EMERG, "L1EVB", logstr );
            sprintf( logstr, "No other data source specified \n");
            logmessage(LOG_WARNING,"L1EventBuild",logstr);
         sprintf(logstr, "ERROR in MWC configuration \n");
         logmessage( LOG_ERR, "L1EVB", logstr );
        sprintf(logstr, "ERROR in EVB Status\n");
        logmessage( LOG_ERR, "L1EVB", logstr );
        sprintf(logstr, "ERROR in EVB Status - STATUS not cleared \n");
        logmessage( LOG_ERR, "L1EVB", logstr );
      sprintf(logstr, "L1ANA stat EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
      logmessage( LOG_DEBUG, "L1EVB", logstr );
       sprintf ( logstr, "debugPrintData routine error \n" );
       logmessage ( LOG_DEBUG, "L1EventBuilder", logstr );
         sprintf( logstr,"Error starting watchdog timer\n");
         logmessage(LOG_ERR,"L1EVB", logstr);
      sprintf(logstr,"Not expecting Message 0x%x from L1ANA",mptr->cmd);
      logmessage(LOG_WARNING,"L1EVB",logstr);
     sprintf(logstr,"Error: Not configured to accept messages from L1ANA and TCU \n");
     logmessage(LOG_ERR,"L1EventBuilder",logstr);
    sprintf(logstr, "MSG from L1CTB - EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
    sprintf(logstr,"EVBStat[%d] is %d \n",ii,EVBStat[ii]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
     sprintf(logstr, "token recieved from L1CTB is %d\n", curToken);
     logmessage( LOG_INFO, "L1EventBuilder", logstr );
      sprintf(logstr, "token recieved from L1CTB is ZERO\n");
      logmessage( LOG_ERR, "L1EventBuilder", logstr );
       sprintf(logstr, "ERROR in EVB Status\n");
       logmessage( LOG_ERR, "L1EVB", logstr );
       sprintf(logstr, "ERROR in EVB Status - STATUS not cleared \n");
       logmessage( LOG_ERR, "L1EVB", logstr );
     sprintf(logstr, "L1CTB stat EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
     logmessage( LOG_DEBUG, "L1EVB", logstr );
      sprintf ( logstr, "debugPrintData routine error \n" );
      logmessage ( LOG_DEBUG, "L1EventBuilder", logstr );
        sprintf( logstr,"Error starting watchdog timer\n");
        logmessage(LOG_ERR,"L1EVB", logstr);
     sprintf(logstr,"Error: Not configured to accept messages from L1CTB and TCU \n");
     logmessage(LOG_ERR,"L1EventBuilder",logstr);
    sprintf(logstr, "MSG from L1DCTM - EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
      sprintf(logstr, "token recieved from L1DCTokenManager is ZERO\n");
      logmessage( LOG_ERR, "L1EventBuilder", logstr );
       sprintf( logstr, "rawDataFill ERROR \n" );
       logmessage( LOG_ERR, "L1EventBuilder", logstr);
      sprintf( logstr, "No other data source specified \n");
      logmessage(LOG_WARNING,"L1EventBuild",logstr);
     sprintf( logstr," Reading data from L1 Hardware Interface \n" );
     logmessage( LOG_DEBUG, "L1EventBuilder", logstr);
      sprintf ( logstr, "debugPrintData routine error \n" );
      logmessage ( LOG_DEBUG, "L1EventBuilder", logstr );
     sprintf(logstr,"Error: Not configured to accept messages from L1DC Token Manager \n");
     logmessage(LOG_ERR,"L1EventBuilder",logstr);
    sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
    logmessage ( LOG_ERR, "L1EventBuilder get_message", logstr);
  sprintf( logstr, " L1 building event ... \n ");
  logmessage( LOG_DEBUG, "L1EVB: rDf", logstr);
   sprintf( logstr, " NO CTB CRATE: filling CTB data here \n ");
   logmessage( LOG_DEBUG, "L1EVB: rDf", logstr);
  sprintf( logstr,"  sizeof L1ptr[0] is 0x%8.8x \n"
  logmessage( LOG_DEBUG, "L1EVB: dPD", logstr );
```

```
  sprintf(logstr,"  L0Sum addresses 0 is 0x%8.8x \n"
  logmessage( LOG_DEBUG, "L1EVB: dPD", logstr );
    sprintf( logstr,"About to take semaphore...\n");
    logmessage( LOG_DEBUG," L0SumWrite", logstr);
    sprintf(logstr, "Interrupt has been received !! \n");
    logmessage( LOG_DEBUG, "L0SumWrite", logstr);
        sprintf(logstr, "token received from L1EVB is %d\n", cur_token);
        logmessage( LOG_INFO, "L1L2I", logstr );
          sprintf( logstr, " cur_token is 0x%8.8x\n LOW_SCRAM_BLOCK is %d \n"
          logmessage( LOG_DEBUG, "L1L2Interface", logstr);
              sprintf(logstr," tokens do not agree \n");
              logmessage(LOG_CRIT,"L1L2I", logstr);
              sprintf(logstr,"ACKptr 0x%8.8x,intAdr3 %d *ACKptr %d, cur_token
%d\n",ACKptr,DATACKAdr,*ACKptr,cur_token);
              logmessage(LOG_CRIT,"L1L2I", logstr);
            sprintf(logstr, "sem timed out \n");
            logmessage(LOG_INFO,"L1L2I",logstr);
          sprintf(logstr,"L1DC destination undefined \n");
          logmessage( LOG_CRIT,"L1L2I",logstr);
      sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno));
      logmessage( LOG_ERR, "L1L2Interface get_message", logstr);
      sprintf( logstr,"About to take semaphore...\n");
      logmessage( LOG_DEBUG," L1TR", logstr);
      sprintf(logstr, "Interrupt has been received !! \n");
      logmessage( LOG_DEBUG, "L1TokenReturn", logstr);
      sprintf(logstr, "token received from L2 is %d\n", cur_token);
      logmessage( LOG_INFO, "L1TR", logstr );
        sprintf ( logstr, "token is ZERO \n");
        logmessage( LOG_ERR, "L1TokenReturn", logstr);
      sprintf( logstr, "Token received back is %d \n", cur_token);
      logmessage( LOG_INFO, "L1TR", logstr);
    sprintf(logstr,"L1TR not configured \n");
    logmessage(LOG_CRIT,"L1TR",logstr);
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage ( LOG_ERR, "L2QueManager", logstr );
  sprintf(logstr,"initializing scramnet ... \n");
  logmessage(LOG_INFO,"L1DCConf",logstr);
    sprintf(logstr,"Error:  Carrier Detect Fail\n");
    logmessage(LOG_ERR,"L1DCConf",logstr);
  sprintf(logstr,"Node Id for this board: 0x%2.2x\n", csr[3] >> 8);         /* Get Node Id */
  logmessage(LOG_DEBUG,"L1DCConf",logstr);
  sprintf(logstr,"CSR1: 0x%2.2x, and memory size: 0x%2.2x\n",(u_s) csr[1],(u_s) (csr[8] & 0x7000) >> 12);
  logmessage(LOG_DEBUG,"L1DCConf",logstr);
  sprintf(logstr,"CSR15: %d\n",csr[15]);
  logmessage(LOG_DEBUG,"L1DCConf",logstr);
  sprintf(logstr,"ACR RAM enabled \n");
  logmessage(LOG_DEBUG,"L1DCConf",logstr);
    sprintf(logstr," set to initialize for level 2 crate \n");
    logmessage(LOG_DEBUG,"L1DCconf",logstr);
    sprintf(logstr," HI_SCRAM_BLOCK %d in buf at address 0x%8.8x
\n",HI_SCRAM_BLOCK,(int)&buf[HI_SCRAM_BLOCK]);
    logmessage(LOG_DEBUG,"L1DCConf",logstr);
    sprintf(logstr," set to initialize for level 1 crate \n");
    logmessage(LOG_DEBUG,"L1DCconf",logstr);
    sprintf(logstr," HI_SCRAM_BLOCK %d in buf at address 0x%8.8x
\n",HI_SCRAM_BLOCK,&buf[HI_SCRAM_BLOCK]);
    logmessage(LOG_DEBUG,"L1DCConf",logstr);
  sprintf(logstr," SETUP ISR for level %d crate \n", Crate);
  logmessage (LOG_DEBUG,"L1DCConf",logstr);
      sprintf(logstr,"Could not create semaphore intsem2 \n");
      logmessage(LOG_ERR,"L1DCConf",logstr);
      sprintf(logstr,"Could not connect to ISR\n");
      logmessage(LOG_ERR,"L1DCConf",logstr);
```

```
    sprintf(logstr,"Could not create semaphore intsem1 \n");
    logmessage(LOG_ERR,"L1DCConf",logstr);
    sprintf(logstr,"Could not create semaphore intsem3 \n");
    logmessage(LOG_ERR,"L1DCConf",logstr);
    sprintf(logstr,"Could not connect to ISR\n");
    logmessage(LOG_ERR,"L1DCConf",logstr);
   sprintf(logstr,"Could not enable interrupts\n");
   logmessage(LOG_ERR,"L1DCConf",logstr);
    sprintf(logstr,"ERROR creating watchdog timer\n");
    logmessage(LOG_ERR,"L1EVB", logstr);
    sprintf(logstr, "MSG from L1ANA – EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
     sprintf(logstr,"EVBStat[%d] is %d \n",ii,EVBStat[ii]);
     logmessage( LOG_DEBUG, "L1EVB", logstr );
      sprintf(logstr, "token recieved from HI is ZERO\n");
      logmessage( LOG_ERR, "L1EventBuilder", logstr );
     sprintf(logstr,"Error: Not configured to accept messages from L1CTB and TCU \n");
     logmessage(LOG_ERR,"L1EventBuilder",logstr);
    sprintf(logstr, "MSG from L1DCTM – EVBStat[%d] is %d\n",curToken,EVBStat[curToken]);
    logmessage( LOG_DEBUG, "L1EVB", logstr );
      sprintf(logstr, "token recieved from L1DCTokenManager is ZERO\n");
      logmessage( LOG_ERR, "L1EventBuilder", logstr );
       sprintf( logstr, "rawDataFill ERROR \n" );
       logmessage( LOG_ERR, "L1EventBuilder", logstr);
      sprintf( logstr, "No other data source specified \n");
      logmessage(LOG_WARNING,"L1EventBuild",logstr);
     sprintf( logstr," Reading data from L1 Hardware Interface \n" );
     logmessage( LOG_DEBUG, "L1EventBuilder", logstr);
      sprintf ( logstr, "debugPrintData routine error \n" );
      logmessage ( LOG_DEBUG, "L1EventBuilder", logstr );
     sprintf(logstr,"Error: Not configured to accept messages from L1DC Token Manager \n");
     logmessage(LOG_ERR,"L1EventBuilder",logstr);
    sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
    logmessage ( LOG_ERR, "L1EventBuilder get_message", logstr);
  sprintf( logstr, " L1 building event ... \n ");
  logmessage( LOG_DEBUG, "L1EVB: rDf", logstr);
   sprintf( logstr, " NO CTB CRATE: filling CTB data here \n ");
   logmessage( LOG_DEBUG, "L1EVB: rDf", logstr);
  sprintf( logstr,"  sizeof L1ptr[0] is 0x%8.8x \n"
  logmessage( LOG_DEBUG, "L1EVB: dPD", logstr );
  sprintf(logstr,"  L0Sum addresses 0 is 0x%8.8x \n"
  logmessage( LOG_DEBUG, "L1EVB: dPD", logstr );
   sprintf( logstr,"About to take semaphore...\n");
   logmessage( LOG_DEBUG," L0SumWrite", logstr);
   sprintf(logstr, "Interrupt has been received !! \n");
   logmessage( LOG_DEBUG, "L0SumWrite", logstr);
  sprintf(logstr, "Tokens zeroed\n");
  logmessage( LOG_NOTICE, "DC_Startup", logstr);
```

L2CTL

```
      sprintf(logstr,"token %d received from L2CTL \n", cur_token);
      logmessage( LOG_INFO,"L2ANA",logstr);
        sprintf( logstr, "No L2 Analysis specified \n" );
        logmessage( LOG_ERR, "L2Analysis", logstr);
     sprintf( logstr, "Unknown message from L2 Control \n" );
     logmessage( LOG_ERR, "L2Analysis", logstr);
    sprintf( logstr,"Serious error received:  errno is: %s\n",strerror(errno));
    logmessage( LOG_ERR, "L2Analysis get_message", logstr);
   sprintf(logstr,"Cannot open %s\n", ctb_mapfile);
```

```
    logmessage(LOG_ERR,"get_ctbmap",logstr);
  sprintf(logstr,"Opening input file %s\n", ctb_mapfile);
  logmessage(LOG_NOTICE,"get_ctbmap",logstr);
    sprintf(workName, "tTokWork%d", ii++);
      sprintf(logstr,"ERROR creating watchdog timer\n");
      logmessage(LOG_ERR,"L1EVB", logstr);
        sprintf(logstr,"token %d received from L2L1I \n", cur_token);
        logmessage( LOG_INFO,"L2CTL",logstr);
          sprintf( logstr, "Unknown message from L2L1Interface \n");
          logmessage ( LOG_ERR, "L2Control", logstr);
               sprintf( logstr, "Event accepted L2 token %d \n",cur_token );
          logmessage ( LOG_INFO, "L2Control", logstr);
            sprintf( logstr,"Error starting watchdog timer\n");
            logmessage(LOG_ERR,"L2EVB", logstr);
                sprintf( logstr, "Event to be written out \n" );
           logmessage ( LOG_DEBUG, "L2Control", logstr);
                 sprintf( logstr, "Writing Event \n" );
            logmessage ( LOG_DEBUG, "L2Control", logstr);
                 sprintf( logstr, "buffer address %x \n",writeBuf );
            logmessage ( LOG_DEBUG, "L2Control", logstr);
                sprintf( logstr, "Writing Event \n" );
           logmessage ( LOG_DEBUG, "L2Control", logstr);
                sprintf( logstr, "buffer address %x \n",writeBuf );
           logmessage ( LOG_DEBUG, "L2Control", logstr);
                sprintf( logstr, "Improper L2/TDI switch set \n" );
           logmessage ( LOG_ERR, "L2Control", logstr);
                sprintf( logstr, "Abort event to TDI \n" );
           logmessage ( LOG_DEBUG, "L2Control", logstr);
                sprintf( logstr, "Abort event to L1 \n" );
           logmessage ( LOG_DEBUG, "L2Control", logstr);
                sprintf( logstr, "Improper L2/TDI switch set \n" );
           logmessage ( LOG_ERR, "L2Control", logstr);
          sprintf( logstr, "Unknown message from L2Analasis \n" );
          logmessage ( LOG_ERR, "L2Control", logstr);
        sprintf(logstr,"token %d received from TDI \n", cur_token);
        logmessage( LOG_INFO,"L2CTL",logstr);
         sprintf( logstr, "Unknown message from TDI \n" );
         logmessage ( LOG_ERR, "L2Control", logstr);
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage( LOG_ERR, "L2Control get_message", logstr);
    sprintf( logstr,"About to take semaphore...\n");
    logmessage( LOG_DEBUG," L2Write", logstr);
    sprintf(logstr, "Interrupt has been received !! \n");
    logmessage( LOG_DEBUG, "L2Write", logstr);
        sprintf(logstr,"Got token %d from L2CTL \n", cur_token);
        logmessage(LOG_INFO,"L2EVB",logstr);
        sprintf(logstr,"sending event to TDI with pre/post %d and size %d bytes \n",Npre,mptr->payload[0] );
        logmessage(LOG_DEBUG,"L2EVB",logstr);
        sprintf(logstr,"Sending token %d (event stream) to TDI
\n",(int)&L2Mem[cur_token].EvtDesc.TCU1.FIFO1.TrgToken);
        logmessage(LOG_INFO,"L2EVB",logstr);
        sprintf(logstr,"sending token %d to TDI \n", cur_token);
        logmessage(LOG_INFO,"L2EVB",logstr);
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage( LOG_ERR, "L2EventBuild get_message", logstr);
      sprintf ( logstr, "error printing data \n" );
      logmessage( LOG_WARNING, "L2L1Interface", logstr);
    sprintf( logstr," location of ptr 0x%8.8x \n",ptr);
    logmessage( LOG_DEBUG, "L2L1Interface", logstr);
    sprintf( logstr," token from L1DC is %d \n",cur_token);
    logmessage( LOG_INFO, "L2L1I", logstr);
      sprintf ( logstr, "error unpacking data \n" );
      logmessage( LOG_WARNING, "L1L2Interface", logstr);
```

```
    sprintf(logstr," token %d sent to L2CTL \n", cur_token);
    logmessage(LOG_INFO,"L2L1I",logstr);
        sprintf(logstr,"token %d received from L2CTL \n", curToken);
        logmessage( LOG_INFO,"L2TR",logstr);
          sprintf( logstr, "Unknown message from L2Control \n");
          logmessage ( LOG_ERR, "L2TokenReturn", logstr );
            sprintf(logstr," tokens do not agree \n");
            logmessage(LOG_CRIT,"L2TR", logstr);
            sprintf(logstr,"TACKptr 0x%8.8x,TACKAdr %d *ACKptr %d, curToken
%d\n",TACKptr,TACKAdr,*TACKptr,curToken);
            logmessage(LOG_CRIT,"L2TR", logstr);
          sprintf(logstr, "sem timed out \n");
          logmessage(LOG_INFO,"L2TR",logstr);
        sprintf(logstr, "Returned token %d to L1DC \n",curToken);
        logmessage ( LOG_INFO, "L2TR", logstr);
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage ( LOG_ERR, "get_message", logstr );
  sprintf ( logstr, " cur_token 0x%8.8x \n",UPtoken);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[0] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.bunchXing_hi);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[1] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.bunchXing_lo);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[2] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.TCU1.fifo1);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[3] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.TCU2.fifo2);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[4] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.TCU3.fifo3);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[5] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.npre);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " L0Sum[5a] 0x%8.8x \n", L2Mem[UPtoken].EvtDesc.npost);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[0] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[0]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[1] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[1]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[2] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[2]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[3] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[3]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[4] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[4]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[5] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[5]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[6] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].CTB[6]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[63] 0x%8.8x \n",L2Mem[UPtoken].RAW[0].CTB[63]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " MWC[21] 0x%8.8x \n", L2Mem[UPtoken].RAW[0].MWC[21]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " ZDC[2] 0x%8.8x \n", L2Mem[UPtoken].TrgSum.DSM.ZDC[2]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[0] pre1 0x%8.8x \n", L2Mem[UPtoken].RAW[1].CTB[0]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[1] pre1 0x%8.8x \n", L2Mem[UPtoken].RAW[1].CTB[1]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[2] pre1 0x%8.8x \n", L2Mem[UPtoken].RAW[1].CTB[2]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[3] pre1 0x%8.8x \n", L2Mem[UPtoken].RAW[1].CTB[3]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[0] pre2 0x%8.8x \n", L2Mem[UPtoken].RAW[2].CTB[0]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[1] pre2 0x%8.8x \n", L2Mem[UPtoken].RAW[2].CTB[1]);
```

```
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[2] pre2 0x%8.8x \n", L2Mem[UPtoken].RAW[2].CTB[2]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf ( logstr, " CTB[3] pre2 0x%8.8x \n", L2Mem[UPtoken].RAW[2].CTB[3]);
  logmessage ( LOG_DEBUG, "dataUnpack", logstr );
  sprintf( logstr, " DATA_ADR 0x%8.8x \n,csr4 0x%8.8x \n,SCRAM_BLOCKS 0x%8.8x \n sum 0x%8.8x \n
",DATA_ADR ,DATAdr, SCRAM_BLOCKS, ( DATA_ADR + DATAdr - SCRAM_BLOCKS ));
  logmessage( LOG_DEBUG, "printDataFromSCRAM", logstr );
        sprintf( logstr, "L2QueManager: Unknown message from L2 Control \n" );
        logmessage( LOG_ERR, "L2QueManager", logstr);
        sprintf( logstr, "L2QueManager: Unknown message from L2 Analysis \n" );
        logmessage( LOG_ERR, "L2QueManager", logstr );
      sprintf( logstr, "Serious error received:  errno is: %s\n",strerror(errno));
      logmessage ( LOG_ERR, "L2QueManager", logstr );
  sprintf(logstr,"initializing scramnet ... \n");
  logmessage(LOG_INFO,"L2CTLConf",logstr);
    sprintf(logstr,"Error:  Carrier Detect Fail\n");
    logmessage(LOG_ERR,"L2CTLConf",logstr);
  sprintf(logstr,"Node Id for this board: 0x%2.2x\n", csr[3] >> 8);          /* Get Node Id */
  logmessage(LOG_DEBUG,"L2CTLConf",logstr);
  sprintf(logstr,"CSR1: 0x%2.2x, and memory size: 0x%2.2x\n",(u_s) csr[1],(u_s) (csr[8] & 0x7000) >> 12);
  logmessage(LOG_DEBUG,"L2CTLConf",logstr);
  sprintf(logstr,"CSR15: %d\n",csr[15]);
  logmessage(LOG_DEBUG,"L2CTLConf",logstr);
  sprintf(logstr,"ACR RAM enabled \n");
  logmessage(LOG_DEBUG,"L2CTLConf",logstr);
    sprintf(logstr," set to initialize for level 2 crate \n");
    logmessage(LOG_INFO,"L2CTLConf",logstr);
    sprintf(logstr," HI_SCRAM_BLOCK %d in buf at address 0x%8.8x
\n",HI_SCRAM_BLOCK,(int)&buf[HI_SCRAM_BLOCK]);
    logmessage(LOG_DEBUG,"L2CTLConf",logstr);
    sprintf(logstr, " set to initialize for level 1 crate \n");
    logmessage(LOG_INFO,"L2CTLConf",logstr);
    sprintf(logstr," HI_SCRAM_BLOCK %d in buf at address 0x%8.8x
\n",HI_SCRAM_BLOCK,(int)&buf[HI_SCRAM_BLOCK]);
    logmessage(LOG_DEBUG,"L2CTLConf",logstr);
  sprintf(logstr," SETUP ISR for level %d crate \n", Crate);
  logmessage(LOG_INFO,"L2CTLConf",logstr);
      sprintf(logstr,"Could not create semaphore intsem2 \n");
      logmessage(LOG_ERR,"L2CTLConf",logstr);
      sprintf(logstr,"Could not connect to ISR\n");
      logmessage(LOG_ERR,"L2CTLConf",logstr);
      sprintf(logstr,"Could not create semaphore intsem1 \n");
      logmessage(LOG_ERR,"L2CTLConf",logstr);
      sprintf(logstr,"Could not connect to ISR\n");
      logmessage(LOG_ERR,"L2CTLConf",logstr);
    sprintf(logstr,"Could not enable interrupts\n");
    logmessage(LOG_ERR,"L2CTLConf",logstr);
  sprintf(logstr, "Tokens zeroed\n");
  logmessage( LOG_NOTICE, "L2_Startup", logstr);
```