```
*******************************************************************************
**                        ARECA FIRMWARE SPEC
*******************************************************************************
**      Usage of IOP331 adapter
**      (All In/Out is in IOP331's view)
**      1. Message 0 --> InitThread message and return code
**      2. Doorbell is used for RS-232 emulation
**              inDoorBell :    bit0 -- data in ready
**                      (DRIVER DATA WRITE OK)
**                              bit1 -- data out has been read
**                      (DRIVER DATA READ OK)
**              outDooeBell:    bit0 -- data out ready
**                      (IOP331 DATA WRITE OK)
**                              bit1 -- data in has been read
**                      (IOP331 DATA READ OK)
**      3. Index Memory Usage
**      offset 0xf00 : for RS232 out (request buffer)
**      offset 0xe00 : for RS232 in  (scratch buffer)
**      offset 0xa00 : for inbound message code message_rwbuffer
**                      (driver send to IOP331)
**      offset 0xa00 : for outbound message code message_rwbuffer
**                      (IOP331 send to driver)
**      4. RS-232 emulation
**              Currently 128 byte buffer is used
**                      1st uint32_t : Data length (1--124)
**                      Byte 4--127  : Max 124 bytes of data
**      5. PostQ
**      All SCSI Command must be sent through postQ:
**      (inbound queue port)    Request frame must be 32 bytes aligned
**      #bit27--bit31 => flag for post ccb
**      #bit0--bit26  => real address (bit27--bit31) of post arcmsr_cdb
**              bit31 :
**                      0 : 256 bytes frame
**                      1 : 512 bytes frame
**              bit30 :
**                      0 : normal request
**                      1 : BIOS request
**              bit29 : reserved
**              bit28 : reserved
**              bit27 : reserved
**  ---------------------------------------------------------------------------
**      (outbount queue port)   Request reply
**      #bit27--bit31
**              => flag for reply
**      #bit0--bit26
**              => real address (bit27--bit31) of reply arcmsr_cdb
**                      bit31 : must be 0 (for this type of reply)
**                      bit30 : reserved for BIOS handshake
**                      bit29 : reserved
**                      bit28 :
**                      0 : no error, ignore AdapStatus/DevStatus/SenseData
**                      1 : Error, error code in AdapStatus/DevStatus/SenseData
**                      bit27 : reserved
**      6. BIOS request
**              All BIOS request is the same with request from PostQ
**              Except :
```

```
**                      Request frame is sent from configuration space
**              offset: 0x78 : Request Frame (bit30 == 1)
**              offset: 0x18 : writeonly to generate
**                                      IRQ to IOP331
**              Completion of request:
**                      (bit30 == 0, bit28==err flag)
**      7. Definition of SGL entry (structure)
**      8. Message1 Out - Diag Status Code (????)
**      9. Message0 message code :
**              0x00 : NOP
**              0x01 : Get Config
**              ->offset 0xa00 :for outbound message code message_rwbuffer
**              (IOP331 send to driver)
**              Signature             0x87974060(4)
**              Request len           0x00000200(4)
**              numbers of queue      0x00000100(4)
**              SDRAM Size            0x00000100(4)-->256 MB
**              IDE Channels          0x00000008(4)
**              vendor                40 bytes char
**              model                  8 bytes char
**              FirmVer               16 bytes char
**              Device Map            16 bytes char
**              FirmwareVersion DWORD <== Added for checking of
**                                          new firmware capability
**              0x02 : Set Config
**              ->offset 0xa00 :for inbound message code message_rwbuffer
**              (driver send to IOP331)
**              Signature             0x87974063(4)
**              UPPER32 of Request Frame  (4)-->Driver Only
**              0x03 : Reset (Abort all queued Command)
**              0x04 : Stop Background Activity
**              0x05 : Flush Cache
**              0x06 : Start Background Activity
**                      (re-start if background is halted)
**              0x07 : Check If Host Command Pending
**                      (Novell May Need This Function)
**              0x08 : Set controller time
**              ->offset 0xa00 : for inbound message code message_rwbuffer
**              (driver to IOP331)
**              byte 0 : 0xaa <-- signature
**              byte 1 : 0x55 <-- signature
**              byte 2 : year (04)
**              byte 3 : month (1..12)
**              byte 4 : date (1..31)
**              byte 5 : hour (0..23)
**              byte 6 : minute (0..59)
**              byte 7 : second (0..59)
********************************************************************************
********************************************************************************
**              RS-232 Interface for Areca Raid Controller
**      The low level command interface is exclusive with VT100 terminal
**  --------------------------------------------------------------------
**      1. Sequence of command execution
**  --------------------------------------------------------------------
**      (A) Header : 3 bytes sequence (0x5E, 0x01, 0x61)
**      (B) Command block : variable length of data including length,
**              command code, data and checksum byte
**      (C) Return data : variable length of data
**  -------------------------------------------------------------------
**    2. Command block
**  -------------------------------------------------------------------
**      (A) 1st byte : command block length (low byte)
**      (B) 2nd byte : command block length (high byte)
**                note ..command block length shouldn't > 2040 bytes,
**              length excludes these two bytes
**      (C) 3rd byte : command code
**      (D) 4th and following bytes : variable length data bytes
**              depends on command code
**      (E) last byte : checksum byte (sum of 1st byte until last data byte)
**  -------------------------------------------------------------------
**    3. Command code and associated data
**  -------------------------------------------------------------------
**      The following are command code defined in raid controller Command
**      code 0x10--0x1? are used for system level management,
**      no password checking is needed and should be implemented in separate
**      well controlled utility and not for end user access.
**      Command code 0x20--0x?? always check the password,
**      password must be entered to enable these command.
**      enum
**      {
**              GUI_SET_SERIAL=0x10,
**              GUI_SET_VENDOR,
**              GUI_SET_MODEL,
**              GUI_IDENTIFY,
**              GUI_CHECK_PASSWORD,
**              GUI_LOGOUT,
**              GUI_HTTP,
**              GUI_SET_ETHERNET_ADDR,
**              GUI_SET_LOGO,
**              GUI_POLL_EVENT,
**              GUI_GET_EVENT,
**              GUI_GET_HW_MONITOR,
**              //    GUI_QUICK_CREATE=0x20, (function removed)
**              GUI_GET_INFO_R=0x20,
**              GUI_GET_INFO_V,
**              GUI_GET_INFO_P,
**              GUI_GET_INFO_S,
**              GUI_CLEAR_EVENT,
**              GUI_MUTE_BEEPER=0x30,
**              GUI_BEEPER_SETTING,
**              GUI_SET_PASSWORD,
**              GUI_HOST_INTERFACE_MODE,
**              GUI_REBUILD_PRIORITY,
**              GUI_MAX_ATA_MODE,
**              GUI_RESET_CONTROLLER,
**              GUI_COM_PORT_SETTING,
**              GUI_NO_OPERATION,
**              GUI_DHCP_IP,
**              GUI_CREATE_PASS_THROUGH=0x40,
**              GUI_MODIFY_PASS_THROUGH,
**              GUI_DELETE_PASS_THROUGH,
**              GUI_IDENTIFY_DEVICE,
**              GUI_CREATE_RAIDSET=0x50,
**              GUI_DELETE_RAIDSET,
**              GUI_EXPAND_RAIDSET,
**              GUI_ACTIVATE_RAIDSET,
**              GUI_CREATE_HOT_SPARE,
**              GUI_DELETE_HOT_SPARE,
**              GUI_CREATE_VOLUME=0x60,
**              GUI_MODIFY_VOLUME,
**              GUI_DELETE_VOLUME,
**              GUI_START_CHECK_VOLUME,
**              GUI_STOP_CHECK_VOLUME
**        };
**      Command description :
**        GUI_SET_SERIAL : Set the controller serial#
**              byte 0,1        : length
**              byte 2          : command code 0x10
**              byte 3          : password length (should be 0x0f)
**              byte 4-0x13     : should be "ArEcATecHnoLogY"
**              byte 0x14--0x23 : Serial number string (must be 16 bytes)
**        GUI_SET_VENDOR : Set vendor string for the controller
**              byte 0,1        : length
**              byte 2          : command code 0x11
**              byte 3          : password length (should be 0x08)
**              byte 4-0x13     : should be "ArEcAvAr"
**              byte 0x14--0x3B : vendor string (must be 40 bytes)
**        GUI_SET_MODEL : Set the model name of the controller
**              byte 0,1        : length
**              byte 2          : command code 0x12
**              byte 3          : password length (should be 0x08)
**              byte 4-0x13     : should be "ArEcAvAr"
**              byte 0x14--0x1B : model string (must be 8 bytes)
**        GUI_IDENTIFY : Identify device
**              byte 0,1        : length
**              byte 2          : command code 0x13
**                                return "Areca RAID Subsystem "
**        GUI_CHECK_PASSWORD : Verify password
**              byte 0,1        : length
**              byte 2          : command code 0x14
**              byte 3          : password length
**              byte 4-0x??     : user password to be checked
**        GUI_LOGOUT : Logout GUI (force password checking on next command)
**              byte 0,1        : length
**              byte 2          : command code 0x15
**        GUI_HTTP : HTTP interface (reserved for Http proxy service)(0x16)
**
**        GUI_SET_ETHERNET_ADDR : Set the ethernet MAC address
**              byte 0,1        : length
**              byte 2          : command code 0x17
**              byte 3          : password length (should be 0x08)
**              byte 4-0x13     : should be "ArEcAvAr"
**              byte 0x14--0x19 : Ethernet MAC address (must be 6 bytes)
**        GUI_SET_LOGO : Set logo in HTTP
**              byte 0,1        : length
**              byte 2          : command code 0x18
**              byte 3          : Page# (0/1/2/3) (0xff --> clear OEM logo)
**              byte 4/5/6/7    : 0x55/0xaa/0xa5/0x5a
**              byte 8          : TITLE.JPG data (each page must be 2000 bytes)
**                                note page0 1st 2 byte must be
**                                      actual length of the JPG file
**      GUI_POLL_EVENT : Poll If Event Log Changed
**              byte 0,1        : length
**              byte 2          : command code 0x19
**      GUI_GET_EVENT : Read Event
**              byte 0,1        : length
**              byte 2          : command code 0x1a
**              byte 3          : Event Page (0:1st page/1/2/3:last page)
**      GUI_GET_HW_MONITOR : Get HW monitor data
**              byte 0,1        : length
**              byte 2                  : command code 0x1b
**              byte 3                  : # of FANs(example 2)
**              byte 4                  : # of Voltage sensor(example 3)
**              byte 5                  : # of temperature sensor(example 2)
**              byte 6                  : # of power
**              byte 7/8        : Fan#0 (RPM)
**              byte 9/10       : Fan#1
**              byte 11/12              : Voltage#0 original value in *1000
**              byte 13/14              : Voltage#0 value
**              byte 15/16              : Voltage#1 org
**              byte 17/18              : Voltage#1
**              byte 19/20              : Voltage#2 org
**              byte 21/22              : Voltage#2
**              byte 23                 : Temp#0
**              byte 24                 : Temp#1
**              byte 25                 : Power indicator (bit0 : power#0,
**                                               bit1 : power#1)
**              byte 26                 : UPS indicator
**      GUI_QUICK_CREATE : Quick create raid/volume set
**          byte 0,1        : length
**          byte 2          : command code 0x20
**          byte 3/4/5/6    : raw capacity
**          byte 7                      : raid level
**          byte 8                      : stripe size
**          byte 9                      : spare
**          byte 10/11/12/13: device mask (the devices to create raid/volume)
**              This function is removed, application like
**              to implement quick create function
**      need to use GUI_CREATE_RAIDSET and GUI_CREATE_VOLUMESET function.
**      GUI_GET_INFO_R : Get Raid Set Information
**              byte 0,1        : length
**              byte 2          : command code 0x20
**              byte 3          : raidset#
**      typedef struct sGUI_RAIDSET
**      {
**              BYTE grsRaidSetName[16];
**              DWORD grsCapacity;
**              DWORD grsCapacityX;
**              DWORD grsFailMask;
**              BYTE grsDevArray[32];
**              BYTE grsMemberDevices;
**              BYTE grsNewMemberDevices;
**              BYTE grsRaidState;
**              BYTE grsVolumes;
**              BYTE grsVolumeList[16];
**              BYTE grsRes1;
**              BYTE grsRes2;
**              BYTE grsRes3;
**              BYTE grsFreeSegments;
**              DWORD grsRawStripes[8];
**              DWORD grsRes4;
**              DWORD grsRes5; //     Total to 128 bytes
**              DWORD grsRes6; //     Total to 128 bytes
**      } sGUI_RAIDSET, *pGUI_RAIDSET;
**      GUI_GET_INFO_V : Get Volume Set Information
**              byte 0,1        : length
**              byte 2          : command code 0x21
**              byte 3          : volumeset#
**      typedef struct sGUI_VOLUMESET
**      {
**              BYTE gvsVolumeName[16]; //     16
**              DWORD gvsCapacity;
**              DWORD gvsCapacityX;
**              DWORD gvsFailMask;
**              DWORD gvsStripeSize;
**              DWORD gvsNewFailMask;
**              DWORD gvsNewStripeSize;
**              DWORD gvsVolumeStatus;
**              DWORD gvsProgress; //     32
**              sSCSI_ATTR gvsScsi;
**              BYTE gvsMemberDisks;
**              BYTE gvsRaidLevel; //     8
**              BYTE gvsNewMemberDisks;
**              BYTE gvsNewRaidLevel;
**              BYTE gvsRaidSetNumber;
**              BYTE gvsRes0; //     4
**              BYTE gvsRes1[4]; //     64 bytes
**      } sGUI_VOLUMESET, *pGUI_VOLUMESET;
**      GUI_GET_INFO_P : Get Physical Drive Information
**              byte 0,1        : length
**              byte 2          : command code 0x22
**              byte 3          : drive # (from 0 to max-channels - 1)
**      typedef struct sGUI_PHY_DRV
**      {
**              BYTE gpdModelName[40];
**              BYTE gpdSerialNumber[20];
**              BYTE gpdFirmRev[8];
**              DWORD gpdCapacity;
**              DWORD gpdCapacityX; //     Reserved for expansion
**              BYTE gpdDeviceState;
**              BYTE gpdPioMode;
**              BYTE gpdCurrentUdmaMode;
**              BYTE gpdUdmaMode;
**              BYTE gpdDriveSelect;
**              BYTE gpdRaidNumber; //     0xff if not belongs to a raid set
**              sSCSI_ATTR gpdScsi;
**              BYTE gpdReserved[40]; //     Total to 128 bytes
**      } sGUI_PHY_DRV, *pGUI_PHY_DRV;
**      GUI_GET_INFO_S : Get System Information
**              byte 0,1        : length
**              byte 2          : command code 0x23
**      typedef struct sCOM_ATTR
**      {
**              BYTE comBaudRate;
**              BYTE comDataBits;
**              BYTE comStopBits;
**              BYTE comParity;
**              BYTE comFlowControl;
**      } sCOM_ATTR, *pCOM_ATTR;
**      typedef struct sSYSTEM_INFO
**      {
**              BYTE gsiVendorName[40];
**              BYTE gsiSerialNumber[16];
**              BYTE gsiFirmVersion[16];
**              BYTE gsiBootVersion[16];
**              BYTE gsiMbVersion[16];
**              BYTE gsiModelName[8];
**              BYTE gsiLocalIp[4];
**              BYTE gsiCurrentIp[4];
**              DWORD gsiTimeTick;
**              DWORD gsiCpuSpeed;
**              DWORD gsiICache;
**              DWORD gsiDCache;
**              DWORD gsiScache;
**              DWORD gsiMemorySize;
**              DWORD gsiMemorySpeed;
**              DWORD gsiEvents;
**              BYTE gsiMacAddress[6];
**              BYTE gsiDhcp;
**              BYTE gsiBeeper;
**              BYTE gsiChannelUsage;
**              BYTE gsiMaxAtaMode;
**              BYTE gsiSdramEcc; //     1:if ECC enabled
**              BYTE gsiRebuildPriority;
**              sCOM_ATTR gsiComA; //     5 bytes
**              sCOM_ATTR gsiComB; //     5 bytes
**              BYTE gsiIdeChannels;
**              BYTE gsiScsiHostChannels;
**              BYTE gsiIdeHostChannels;
**              BYTE gsiMaxVolumeSet;
**              BYTE gsiMaxRaidSet;
**              BYTE gsiEtherPort; //     1:if ether net port supported
**              BYTE gsiRaid6Engine; //     1:Raid6 engine supported
**              BYTE gsiRes[75];
**      } sSYSTEM_INFO, *pSYSTEM_INFO;
**      GUI_CLEAR_EVENT : Clear System Event
**              byte 0,1        : length
**              byte 2          : command code 0x24
**      GUI_MUTE_BEEPER : Mute current beeper
**              byte 0,1        : length
**              byte 2          : command code 0x30
**      GUI_BEEPER_SETTING : Disable beeper
**              byte 0,1        : length
**              byte 2          : command code 0x31
**              byte 3          : 0->disable, 1->enable
**      GUI_SET_PASSWORD : Change password
**              byte 0,1        : length
**              byte 2                  : command code 0x32
**              byte 3                  : pass word length ( must <= 15 )
**              byte 4                  : password (must be alpha-numerical)
**      GUI_HOST_INTERFACE_MODE : Set host interface mode
**              byte 0,1        : length
**              byte 2                  : command code 0x33
**              byte 3                  : 0->Independent, 1->cluster
**      GUI_REBUILD_PRIORITY : Set rebuild priority
**              byte 0,1        : length
**              byte 2                  : command code 0x34
**              byte 3                  : 0/1/2/3 (low->high)
**      GUI_MAX_ATA_MODE : Set maximum ATA mode to be used
**              byte 0,1        : length
**              byte 2                  : command code 0x35
**              byte 3                  : 0/1/2/3 (133/100/66/33)
**      GUI_RESET_CONTROLLER : Reset Controller
**              byte 0,1        : length
**              byte 2          : command code 0x36
**                             *Response with VT100 screen (discard it)
**      GUI_COM_PORT_SETTING : COM port setting
**              byte 0,1        : length
**              byte 2                  : command code 0x37
**              byte 3                  : 0->COMA (term port),
**                                        1->COMB (debug port)
**              byte 4                  : 0/1/2/3/4/5/6/7
**                      (1200/2400/4800/9600/19200/38400/57600/115200)
**              byte 5                  : data bit
**                                      (0:7 bit, 1:8 bit : must be 8 bit)
**              byte 6                  : stop bit (0:1, 1:2 stop bits)
**              byte 7                  : parity (0:none, 1:off, 2:even)
**              byte 8                  : flow control
**                      (0:none, 1:xon/xoff, 2:hardware => must use none)
**      GUI_NO_OPERATION : No operation
**              byte 0,1        : length
**              byte 2          : command code 0x38
**      GUI_DHCP_IP : Set DHCP option and local IP address
**              byte 0,1        : length
**              byte 2          : command code 0x39
**              byte 3          : 0:dhcp disabled, 1:dhcp enabled
**              byte 4/5/6/7    : IP address
**      GUI_CREATE_PASS_THROUGH : Create pass through disk
**              byte 0,1        : length
**              byte 2                  : command code 0x40
**              byte 3                  : device #
**              byte 4                  : scsi channel (0/1)
**              byte 5                  : scsi id (0-->15)
**              byte 6                  : scsi lun (0-->7)
**              byte 7                  : tagged queue (1 : enabled)
**              byte 8                  : cache mode (1 : enabled)
**              byte 9                  : max speed (0/1/2/3/4,
**                      async/20/40/80/160 for scsi)
**                      (0/1/2/3/4, 33/66/100/133/150 for ide  )
**      GUI_MODIFY_PASS_THROUGH : Modify pass through disk
**              byte 0,1        : length
**              byte 2                  : command code 0x41
**              byte 3                  : device #
**              byte 4                  : scsi channel (0/1)
**              byte 5                  : scsi id (0-->15)
**              byte 6                  : scsi lun (0-->7)
**              byte 7                  : tagged queue (1 : enabled)
**              byte 8                  : cache mode (1 : enabled)
**              byte 9                  : max speed (0/1/2/3/4,
**                                      async/20/40/80/160 for scsi)
**                      (0/1/2/3/4, 33/66/100/133/150 for ide  )
**      GUI_DELETE_PASS_THROUGH : Delete pass through disk
**              byte 0,1        : length
**              byte 2          : command code 0x42
**              byte 3          : device# to be deleted
**      GUI_IDENTIFY_DEVICE : Identify Device
**              byte 0,1        : length
**              byte 2          : command code 0x43
**              byte 3          : Flash Method
**                              (0:flash selected, 1:flash not selected)
**              byte 4/5/6/7    : IDE device mask to be flashed
**                           note .... no response data available
**      GUI_CREATE_RAIDSET : Create Raid Set
**              byte 0,1        : length
**              byte 2          : command code 0x50
**              byte 3/4/5/6    : device mask
**              byte 7-22       : raidset name (if byte 7 == 0:use default)
**      GUI_DELETE_RAIDSET : Delete Raid Set
**              byte 0,1        : length
**              byte 2          : command code 0x51
**              byte 3          : raidset#
**      GUI_EXPAND_RAIDSET : Expand Raid Set
**              byte 0,1        : length
**              byte 2          : command code 0x52
**              byte 3          : raidset#
**              byte 4/5/6/7    : device mask for expansion
**              byte 8/9/10     : (8:0 no change, 1 change, 0xff:terminate,
**                              9:new raid level,
**                              10:new stripe size
**                              0/1/2/3/4/5->4/8/16/32/64/128K )
**              byte 11/12/13   : repeat for each volume in the raidset
**      GUI_ACTIVATE_RAIDSET : Activate incomplete raid set
**              byte 0,1        : length
**              byte 2          : command code 0x53
**              byte 3          : raidset#
**      GUI_CREATE_HOT_SPARE : Create hot spare disk
**              byte 0,1        : length
**              byte 2          : command code 0x54
**              byte 3/4/5/6    : device mask for hot spare creation
**      GUI_DELETE_HOT_SPARE : Delete hot spare disk
**              byte 0,1        : length
**              byte 2          : command code 0x55
**              byte 3/4/5/6    : device mask for hot spare deletion
**      GUI_CREATE_VOLUME : Create volume set
**              byte 0,1        : length
**              byte 2          : command code 0x60
**              byte 3          : raidset#
**              byte 4-19       : volume set name
**                              (if byte4 == 0, use default)
**              byte 20-27      : volume capacity (blocks)
**              byte 28                 : raid level
**              byte 29                 : stripe size
**                              (0/1/2/3/4/5->4/8/16/32/64/128K)
**              byte 30                 : channel
**              byte 31                 : ID
**              byte 32                 : LUN
**              byte 33                 : 1 enable tag
**              byte 34                 : 1 enable cache
**              byte 35                 : speed
**              (0/1/2/3/4->async/20/40/80/160 for scsi)
**              (0/1/2/3/4->33/66/100/133/150 for IDE  )
**              byte 36                 : 1 to select quick init
**
**      GUI_MODIFY_VOLUME : Modify volume Set
**              byte 0,1        : length
**              byte 2          : command code 0x61
**              byte 3          : volumeset#
**              byte 4-19       : new volume set name
**              (if byte4 == 0, not change)
**              byte 20-27      : new volume capacity (reserved)
**              byte 28                 : new raid level
**              byte 29                 : new stripe size
**              (0/1/2/3/4/5->4/8/16/32/64/128K)
**              byte 30                 : new channel
**              byte 31                 : new ID
**              byte 32                 : new LUN
**              byte 33                 : 1 enable tag
**              byte 34                 : 1 enable cache
**              byte 35                 : speed
**              (0/1/2/3/4->async/20/40/80/160 for scsi)
**              (0/1/2/3/4->33/66/100/133/150 for IDE  )
**      GUI_DELETE_VOLUME : Delete volume set
**              byte 0,1        : length
**              byte 2          : command code 0x62
**              byte 3          : volumeset#
**      GUI_START_CHECK_VOLUME : Start volume consistency check
**              byte 0,1        : length
**              byte 2          : command code 0x63
**              byte 3          : volumeset#
**      GUI_STOP_CHECK_VOLUME : Stop volume consistency check
**              byte 0,1        : length
**              byte 2          : command code 0x64
** ---------------------------------------------------------------------
**    4. Returned data
** ---------------------------------------------------------------------
**      (A) Header          : 3 bytes sequence (0x5E, 0x01, 0x61)
**      (B) Length          : 2 bytes
**                      (low byte 1st, excludes length and checksum byte)
**      (C) status or data  :
**           <1> If length == 1 ==> 1 byte status code
**              #define GUI_OK                      0x41
**              #define GUI_RAIDSET_NOT_NORMAL      0x42
**              #define GUI_VOLUMESET_NOT_NORMAL    0x43
**              #define GUI_NO_RAIDSET              0x44
**              #define GUI_NO_VOLUMESET         0x45
**              #define GUI_NO_PHYSICAL_DRIVE    0x46
**              #define GUI_PARAMETER_ERROR      0x47
**              #define GUI_UNSUPPORTED_COMMAND  0x48
**              #define GUI_DISK_CONFIG_CHANGED  0x49
**              #define GUI_INVALID_PASSWORD     0x4a
**              #define GUI_NO_DISK_SPACE        0x4b
**              #define GUI_CHECKSUM_ERROR       0x4c
**              #define GUI_PASSWORD_REQUIRED    0x4d
**           <2> If length > 1 ==>
**              data block returned from controller
**              and the contents depends on the command code
**      (E) Checksum       : checksum of length and status or data byte
**************************************************************************
```