```
[Identification]
    OptionType = NetAdapter
[PlatformsSupported]
    PCI
    PCMCIA
[Options]
     MSMDGMPPCI
    MSMDGMPPCMCIA
[OptionsISA]
[OptionsMCA]
[OptionsEISA]
[OptionsPCI]
    MSMDGMPPCI
[OptionsPCMCIA]
    MSMDGMPPCMCIA
[SlotNumberOnlyOptions]
    MSMDGMPEISA
    MSMDGMPMCA
    MSMDGMPMC32
    MSMDGMPPCI
    MSMDGMPPCIBM
[MmioOptions]
    MSMDGMPPCI
[FileConstants]
DialogDllName = "MDGMPDLG.DLL"
SoftwareType  = "driver"
Exit_Code     = 0
Manufacturer        = "Madge"
ProductMajorVersion = "4"
ProductMinorVersion = "0"
ProductRevision     = ".00"
ProductVersion      = $(ProductMajorVersion)"."$(ProductMinorVersion)$
(ProductRevision)
ProductSoftwareName     = "MadgeMPort"
ProductSoftwareImagePath = "%SystemRoot%\system32\drivers\MdgMPort.sys"
NetRuleSoftwareType     = "mdgtrSys ndisDriver mdgtrDriver"
NetRuleSoftwareUse      = $(SoftwareType)
NetRuleSoftwareBindForm = """mdgtrSys"" yes no container"
NetRuleSoftwareClass    = {"mdgtrDriver basic"}
NetRuleSoftwareBindable = {"mdgtrDriver mdgtrAdapter non exclusive 100"}
ProductHardwareName         = "MadgeMPort"
NetRuleHardwareType         = "mdgtr mdgtrAdapter"
NetRuleHardwareBindForm     = " yes yes container"
NetRuleHardwareClass        = {"mdgtrAdapter basic"}
ProductOpSupport     = 134
ProductKeyName  = $(!NTN_SoftwareBase)"\"$(Manufacturer)"\"$(ProductSoftwareName)"\
CurrentVersion"
ParamKeyName    = $(!NTN_ServiceBase)"\"$(ProductHardwareName)"\Parameters"
MadgeEventLogFile = $(ProductSoftwareImagePath)
[GeneralConstants]
UtilityInf    = "UTILITY.INF"
SubroutineInf = "SUBROUTN.INF"
ParamInf      = "NCPARAM.INF"
From      = ""
To        = ""
ExitCodeOk     = 0
ExitCodeCancel = 1
ExitCodeFatal  = 2
MAXIMUM_ALLOWED    = 33554432
```

```
RegistryErrorIndex = NO_ERROR
KeyNull            = ""
KeyProduct         = ""
KeyParameters      = ""
TRUE    = 1
FALSE   = 0
NoTitle = 0
ExitState         = "Active"
OldVersionExisted = $(FALSE)
DriverPath      = $(!STF_NTPATH)\drivers
VALUE_UNKNOWN   = 65535
GENERAL_PIO     = 0
GENERAL_DMA     = 500
GENERAL_MMIO    = 501
TRANSFER_UNKNOWN = 65535
TRANSFER_PIO     = 0
TRANSFER_DMA     = 1
TRANSFER_MMIO    = 2
IoUserList    = ^(IoLocationChoices, 1)
IoRawList     = ^(IoLocationChoices, 2)
SlotUserList  = ^(SlotNumberChoices, 1)
SlotRawList   = ^(SlotNumberChoices, 2)
DmaUserList   = ^(DmaChannelChoices, 1)
DmaRawList    = ^(DmaChannelChoices, 2)
IrqUserList   = ^(IrqNumberChoices, 1)
IrqRawList    = ^(IrqNumberChoices, 2)
RxTxUserList  = ^(RxTxSlotsChoices,1)
RxTxRawList   = ^(RxTxSlotsChoices,2)
StatsUserList = ^(StatsChoices,1)
StatsRawList  = ^(StatsChoices,2)
MpUserList    = ^(MpChoices,1)
MpRawList     = ^(MpChoices,2)
MadgeAdapterOptions  = ^(Options,1)
SlotNumberAdapters   = ^(SlotNumberOnlyOptions,1)
MmioAdapters         = ^(MmioOptions,1)
MadgeHelpIdMin         = 6000
MadgeHelpIdMax         = 6003
MadgeHelpIdMDGMPISA    = 6001
MadgeHelpIdMDGMPATP    = 6001
MadgeHelpIdMDGMPISAC   = 6001
MadgeHelpIdMDGMPISACP  = 6001
MadgeHelpIdMDGMPPC     = 6001
MadgeHelpIdMDGMPSM16   = 6001
MadgeHelpIdMDGMPPNP    = 6001
MadgeHelpIdMDGMPPCMCIA = 6001
MadgeHelpIdMDGMPEISA   = 6002
MadgeHelpIdMDGMPMCA    = 6002
MadgeHelpIdMDGMPMC32   = 6002
MadgeHelpIdMDGMPPCI    = 6003
MadgeHelpIdMDGMPPCIBM  = 6003
IDMSMDGMPPCI =   135350
IDVMSMDGMPPCI = AdapterCFID
IDMSMDGMPPCIBM =   266422
IDVMSMDGMPPCIBM = AdapterCFID
IDMSMDGMPEISA =   34612
IDVMSMDGMPEISA = EisaCompressedId
IDMSMDGMPMCA =   45
IDVMSMDGMPMCA = McaPosId
IDMSMDGMPMC32 =   116
```

```
IDVMSMDGMPMC32 = McaPosId
[date]
    Now = {} ? $(!LIBHANDLE) GetSystemDate
[Identify]
    read-syms Identification
    set Status     = STATUS_SUCCESSFUL
    set Identifier = $(OptionType)
    set Media      = #("Source Media Descriptions", 1, 1)
    return $(Status) $(Identifier) $(Media)
[ReturnOptions]
    set Status         = STATUS_FAILED
    set OptionList     = {}
    set OptionTextList = {}
    set LanguageList = ^(LanguagesSupported, 1)
    ifcontains(i) $($0) in $(LanguageList)
      ifstr(i) $($1) == ""
          goto Return_Options
      endif
      set PlatformList = ^(PlatformsSupported, 1)
      ifcontains(i) $($1) in $(PlatformList)
          goto Return_Options
      else
          set Status = STATUS_NOTSUPPORTED
          goto Finish_ReturnOptions
      endif
    else
        set Status = STATUS_NOLANGUAGE
      goto Finish_ReturnOptions
    endif
Return_Options = +
    set OptionList     = ^(Options$($1), 1)
    set OptionTextList = ^(OptionsText$($1)$($0), 1)
    set Status         = STATUS_SUCCESSFUL
Finish_ReturnOptions = +
    return $(Status) $(OptionList) $(OptionTextList)
[InstallOption]
    StartWait
    set Status = STATUS_FAILED
    set Option   = $($1)
    set SrcDir   = $($2)
    set AddCopy  = $($3)
    set DoCopy   = $($4)
    set DoConfig = $($5)
    Debug-Output "MADGE: STF_CWDIR    = "$(!STF_CWDIR)
    Debug-Output "MADGE: STF_LANGUAGE = "$(!STF_LANGUAGE)
    Debug-Output "MADGE: Option       = "$(Option)
    Debug-Output "MADGE: SrcDir       = "$(SrcDir)
    Debug-Output "MADGE: AddCopy      = "$(AddCopy)
    Debug-Output "MADGE: DoCopy       = "$(DoCopy)
    Debug-Output "MADGE: DoConfig     = "$(DoConfig)
    set LanguageList = ^(LanguagesSupported, 1)
    ifcontains(i) $($0) NOT-IN $(LanguageList)
      return STATUS_NOLANGUAGE
    endif
    set-subst LF = "\n"
    read-syms GeneralConstants
    read-syms FileConstants
    read-syms DialogConstants$(!STF_LANGUAGE)
    ifstr(i) $(!NTN_Origination) == "NCPA"
```

```
      set Continue = $(OK)
    endif
    read-syms FileConstants$(!STF_LANGUAGE)
    SetHelpFile "mdgmpdlg.hlp" $(MadgeHelpIdMin) $(MadgeHelpIdMax)
    detect date
    set-title $(FunctionTitle)
    set To   = Begin
    set From = Begin
    set CommonStatus = STATUS_SUCCESSFUL
    EndWait
Begin = +
    set AdapterDetected     = FALSE
    set MappedRawParameters = FALSE
    ifstr(i) $(!NTN_InstallMode) == deinstall
      set StartLabel = Remove_Adapter
    else-ifstr(i) $(!NTN_InstallMode) == Update
      set StartLabel = Upgrade_Software
    else-ifstr(i) $(!NTN_InstallMode) == bind
      set StartLabel = Binding_Adapter
    else-ifstr(i) $(!NTN_InstallMode) == configure
      set CommonStatus = STATUS_REBOOT
      set StartLabel = Configure_Adapter
      ifstr(i) $(ProductKeyName) == $(!NTN_RegBase)
          Shell $(UtilityInf),RegistryErrorString,CANNOT_CONFIGURE_SOFTWARE
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
                goto Shell_Code_Error
            endif
              set Error = $($R0)
              set From  = End
              set To    = End
              goto Non_Fatal_Info
      endif
    else
          set StartLabel           = Install_Adapter
          set OEM_ABANDON_OPTIONS  = {}
          set OEM_ABANDON_SOFTWARE = FALSE
          set OEM_ABANDON_ON       = TRUE
    endif
    set From = Fatal
    set To   = Fatal
    goto $(StartLabel)
Install_Adapter = +
    OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED) KeyProduct
    Ifstr $(KeyProduct) != $(KeyNull)
          CloseRegKey $(KeyProduct)
          ifstr(i) $(!NTN_RegBase) == $(ProductKeyName)
             Shell $(UtilityInf), VerExistedDlg, $(ProductSoftwareTitle),+
                 $(ProductVersion)
             ifint $($ShellCode) != $(!SHELL_CODE_OK)
                 goto Shell_Code_Error
             endif
             goto End
          else
           ifstr(i) $(Option) != "MSMDGMPAUTO"
                Shell $(UtilityInf), CardExistedDlg
                ifint $($ShellCode) != $(!SHELL_CODE_OK)
                    goto Shell_Code_Error
                endif
                ifstr(i) $($R1) != "OK"
```

```
                    goto End
                endif
            endif
              set OldVersionExisted = $(TRUE)
            endif
    endif
Install_Files = +
    Debug-Output "OEMNADMA.INF: Install Files"
    ifint $(OldVersionExisted) == $(FALSE)
      ifstr(i) $(!NTN_InstallMode) == "install"
              Shell $(UtilityInf), DoAskSource, $(!STF_CWDDIR), $(!STF_SRCDIR),
"YES"
              ifint $($ShellCode) != $(!SHELL_CODE_OK)
                goto Shell_Code_Error
              else-ifstr(i) $($R0) == STATUS_FAILED
                  shell $(UtilityInf) RegistryErrorString "ASK_SOURCE_FAIL"
                  ifint $($ShellCode) != $(!SHELL_CODE_OK)
                    goto Shell_Code_Error
                  endif
                  set Error = $($R0)
                  goto Fatal
            else-ifstr(i) $($R0) == STATUS_USERCANCEL
                  goto Successful
              endif
              set SrcDir = $($R1)
              install "Install-Option"
              ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
                shell $(UtilityInf) RegistryErrorString "UNABLE_COPY_FILE"
                ifint $($ShellCode) != $(!SHELL_CODE_OK)
                    goto Shell_Code_Error
                endif
                    set Error = $($R0)
                    goto Fatal
              endif
            endif
    endif
    ifstr(i) $(!STF_NCDETECT) == YES
        ifstr(i) $(!STF_NCOPTION) == $(Option)
            set AdapterDetected = TRUE
            set DetectOption    = $(!STF_NCOPTION)
            set DetectIndex     = $(!STF_NCDETCARD)
            goto Set_Installation_Defaults
        endif
    endif
Set_Installation_Defaults = +
    Debug-Output "OEMNADMA.INF: Set Install Defaults"
    StartWait
    set MaxFrameSize  = 4096
    set LAA           = ""
    set RxTxSlots     = *($(RxTxRawList),3)
    set StatsFlag     = *($(StatsRawList),1)
    set SpeedFlag     = 0
    set OldIrqNumber  = $(VALUE_UNKNOWN)
    set OldDmaChannel = $(VALUE_UNKNOWN)
    set OldIoLocation = $(VALUE_UNKNOWN)
    set OldSlotNumber = $(VALUE_UNKNOWN)
    set OldMpFlag     = $(VALUE_UNKNOWN)
    ifcontains(i) $(Option) in $(SlotNumberAdapters)
        set TypeList = {{SLOTNUMBER,SlotList,SlotNumber},+
```

```
                        {DMACHANNEL,DmaList,DmaChannel},+
                        {MULTIPROCESSOR,MpList,MpFlag}}
    else
        set TypeList = {{IOLOCATION,IoList,IoLocation},+
                        {DMACHANNEL,DmaList,DmaChannel},+
                        {INTERRUPTNUMBER,IrqList,IrqNumber},+
                        {MULTIPROCESSOR,MpList,MpFlag}}
    endif
    Shell $(ParamInf) Param_BuildTypeLists $(Option) $(TypeList)
    set Status = $($R0)
    ifstr(i) $(Status) != STATUS_SUCCESSFUL
        Debug-Output "OEMNADMA.INF: Param_BuildTypeLists returned "$(Status)
        goto Fatal
    endif
    ifstr(i) $(AdapterDetected) == TRUE
        Debug-Output "OEMNADMA.INF: Calling Param_QueryCard"
        Shell $(ParamInf) Param_QueryCard $(DetectIndex)
        set Status    = $($R0)
        set ParamList = $($R1)
        ifstr(i) $(Status) != STATUS_SUCCESSFUL
            goto Fatal
        endif
    endif
    Debug-Output "OEMNADMA.INF: Calling Param_SetDefaults"
    Shell $(ParamInf) Param_SetDefaults $(ParamList)
    Debug-Output "MADGE: SlotNumber = "$(SlotNumber)
    Debug-Output "MADGE: IoLocation = "$(IoLocation)
    Debug-Output "MADGE: IrqNumber  = "$(IrqNumber)
    Debug-Output "MADGE: DmaChannel = "$(DmaChannel)
    Debug-Output "MADGE: MpFlag     = "$(MpFlag)
    Debug-Output "MADGE: SlotList   = "$(SlotList)
    Debug-Output "MADGE: IoList     = "$(IoList)
    Debug-Output "MADGE: IrqList    = "$(IrqList)
    Debug-Output "MADGE: DmaList    = "$(DmaList)
    Debug-Output "MADGE: MpList     = "$(MpList)
    ifstr(i) $(AdapterDetected) == TRUE
        set OldSlotNumber = $(SlotNumber)
        set OldIoLocation = $(IoLocation)
        set OldIrqNumber  = $(IrqNumber)
        set OldDmaChannel = $(DmaChannel)
        set OldMpFlag     = $(MpFlag)
    endif
    set OldValueTitle = $(OldValueTitleInstall)
    EndWait
    goto Get_Adapter_Options
Configure_Adapter = +
    StartWait
    ifcontains(i) $(Option) in $(SlotNumberAdapters)
        set TypeList = {{SLOTNUMBER,SlotList,SlotNumber},+
                        {DMACHANNEL,DmaList,DmaChannel},+
                        {MULTIPROCESSOR,MpList,MpFlag}}
    else
        set TypeList = {{IOLOCATION,IoList,IoLocation},+
                        {DMACHANNEL,DmaList,DmaChannel},+
                        {INTERRUPTNUMBER,IrqList,IrqNumber},+
                        {MULTIPROCESSOR,MpList,MpFlag}}
    endif
    Shell $(ParamInf) Param_BuildTypeLists $(Option) $(TypeList)
    set Status = $($R0)
```

```
    ifstr(i) $(Status) != STATUS_SUCCESSFUL
        goto fataldetect
    endif
    Shell $(ParamInf) Param_SetDefaults $(ParamList)
    set MaxFrameSize = 4096
    set LAA          = ""
    set RxTxSlots    = *($(RxTxRawList),3)
    set StatsFlag    = *($(StatsRawList),1)
    set NoMmioFlag   = 0
    set SpeedFlag    = 0
    set TransferType = $(TRANSFER_UNKNOWN)
    ifstr $(KeyProduct) == $(KeyNull)
          OpenRegKey $(!REG_H_LOCAL) "" $(!NTN_RegBase) $(MAXIMUM_ALLOWED)
KeyProduct
          ifstr $(KeyProduct) == $(KeyNull)
              set RegistryErrorIndex = CANNOT_FIND_COMPONENT_SERVICE
              goto Fatal_Registry
          endif
    endif
    Shell $(UtilityInf) FindService, $(KeyProduct)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    ifstr(i) $($R0) != NO_ERROR
      goto Fatal_Registry
    endif
    set KeyParameters = $($R2)
    CloseRegKey $($R1)
    ifstr $(KeyParameters) == $(KeyNull)
      set RegistryErrorIndex = CANNOT_FIND_COMPONENT_SERVICE
      goto Fatal_Registry
    endif
    set OldVersionExisted = $(TRUE)
    set ValueName = ""
    set ValueData = ""
    set ValueStr  = ""
    set ValueList = {}
    EnumRegValue $(KeyParameters) ValueList
    ForListDo $(ValueList)
          set ValueItem = $($)
          set ValueName = *($(ValueItem),1)
          set ValueData = *($(ValueItem),4)
          ifstr(i) $(ValueName) == "InterruptNumber"
              set IrqNumber = $(ValueData)
          else-ifstr(i) $(ValueName) == "IoLocation"
              set IoLocation = $(ValueData)
          else-ifstr(i) $(ValueName) == "IoBaseAddress"
              set IoLocation = $(ValueData)
          else-ifstr(i) $(ValueName) == "DmaChannel"
              set DmaChannel = $(ValueData)
          else-ifstr(i) $(ValueName) == "SlotNumber"
              set SlotNumber = $(ValueData)
          else-ifstr(i) $(ValueName) == $(McaEisaKeyword)
              set McaEisaId = $(ValueData)
          else-ifstr(i) $(ValueName) == "MaxFrameSize"
              set MaxFrameSize = $(ValueData)
          else-ifstr(i) $(ValueName) == "NetworkAddress"
              set LAA = $(ValueData)
        else-ifstr(i) $(ValueName) == "RxTxSlots"
```

```
            set RxTxSlots = $(ValueData)
        else-ifstr(i) $(ValueName) == "PromiscuousModeX"
            set StatsFlag = $(ValueData)
        else-ifstr(i) $(ValueName) == "Multiprocessor"
            set MpFlag = $(ValueData)
        else-ifstr(i) $(ValueName) == "NoMmio"
            set NoMmioFlag = $(ValueData)
        else-ifstr(i) $(ValueName) == "Force16"
            set SpeedFlag = 2
        else-ifstr(i) $(ValueName) == "Force4"
            set SpeedFlag = 1
        else-ifstr(i) $(ValueName) == "TransferType"
            set TransferType = $(ValueData)
          endif
    EndForListDo
    ifint $(DmaChannel) == 32768
        set DmaChannel = $(GENERAL_PIO)
        set MpFlag     = 1
    endif
    ifcontains(i) $(GENERAL_MMIO) in $(DmaList)
        ifint $(NoMmioFlag) != 0
            set DmaChannel = $(GENERAL_PIO)
        else
            set DmaChannel = $(GENERAL_MMIO)
        endif
    endif
    ifint $(TransferType) == $(TRANSFER_PIO)
        ifcontains(i) $(GENERAL_PIO) in $(DmaList)
            set DmaChannel = $(GENERAL_PIO)
        endif
    endif
    ifint $(TransferType) == $(TRANSFER_DMA)
        ifcontains(i) $(GENERAL_DMA) in $(DmaList)
            set DmaChannel = $(GENERAL_DMA)
        endif
    endif
    set OldSlotNumber = $(SlotNumber)
    set OldIoLocation = $(IoLocation)
    set OldIrqNumber  = $(IrqNumber)
    set OldDmaChannel = $(DmaChannel)
    set OldMpFlag     = $(MpFlag)
    set OldValueTitle = $(OldValueTitleConfigure)
    EndWait
Get_Adapter_Options = +
    Debug-Output "OEMNADMA.INF: Get_Adapter_Options"
    StartWait
    ifstr(i) $(!STF_GUI_UNATTENDED) == "YES"
        ifstr(i) $(!STF_NCDETINFO) == {}
            ifstr(i) $(!AutoNetInterfaceType) != ""
                set BusInterfaceType = $(!AutoNetInterfaceType)
            else
                set BusInterfaceType = 1
            endif
            ifstr(i) $(!AutoNetBusNumber) != ""
                set BusNumber = $(!AutoNetBusNumber)
            else
                set BusNumber = 0
            endif
        else
```

```
            ifstr(i) $(!AutoNetInterfaceType) != ""
                set BusInterfaceType = $(!AutoNetInterfaceType)
            else
                set BusInterfaceType = *($(!STF_NCDETINFO),5)
            endif
            ifstr(i) $(!AutoNetBusNumber) != ""
                set BusNumber = $(!AutoNetBusNumber)
            else
                set BusNumber = *($(!STF_NCDETINFO),6)
            endif
        endif
        goto adapterverify
    endif
    set OldLAA       = $(LAA)
    set OldStatsFlag = $(StatsFlag)
    set OldSpeedFlag = $(SpeedFlag)
    set SpeedRawList  = $(SpeedRawList$(Option))
    set SpeedUserList = $(SpeedUserList$(Option))
    ifstr(i) $(MappedRawParameters) == FALSE
        ifcontains(i) $(Option) not-in $(SlotNumberAdapters)
            set TempList = $(IoList)
            set IoList   = {}
            ForListDo $(TempList)
                set IoList = >($(IoList), *($(IoUserList),~($(IoRawList),$($))))
            EndForListDo
            set TempList = $(IrqList)
            set IrqList  = {}
            ForListDo $(TempList)
                set IrqList = >($(IrqList), *($(IrqUserList),~($(IrqRawList),$
($))))
            EndForListDo
        else
            set TempList = $(SlotList)
            set SlotList  = {}
            ForListDo $(TempList)
                set SlotList = >($(SlotList), *($(SlotUserList),~($(SlotRawList),$
($))))
            EndForListDo
        endif
        set TempList = $(DmaList)
        set DmaList  = {}
        ForListDo $(TempList)
            set DmaList = >($(DmaList), *($(DmaUserList),~($(DmaRawList),$($))))
        EndForListDo
        set TempList = $(MpList)
        set MpList   = {}
        ForListDo $(TempList)
            set MpList = >($(MpList), *($(MpUserList),~($(MpRawList),$($))))
        EndForListDo
        set MappedRawParameters = TRUE
    endif
    EndWait
Get_Adapter_Options_Restart = +
    StartWait
    set From = Get_Adapter_Options_Restart
    ifcontains(i) $(Option) not-in $(SlotNumberAdapters)
        set IoLocation    = *($(IoUserList),~($(IoRawList),$(IoLocation)))
        set OldIoLocation = *($(IoUserList),~($(IoRawList),$(OldIoLocation)))
        set IrqNumber     = *($(IrqUserList),~($(IrqRawList),$(IrqNumber)))
```

```
    set OldIrqNumber  = *($(IrqUserList),~($(IrqRawList),$(OldIrqNumber)))
else
    set SlotNumber    = *($(SlotUserList),~($(SlotRawList),$(SlotNumber)))
    set OldSlotNumber = *($(SlotUserList),~($(SlotRawList),$(OldSlotNumber)))
endif
set DmaChannel    = *($(DmaUserList),~($(DmaRawList),$(DmaChannel)))
set OldDmaChannel = *($(DmaUserList),~($(DmaRawList),$(OldDmaChannel)))
set MpFlag        = *($(MpUserList),~($(MpRawList),$(MpFlag)))
set OldMpFlag     = *($(MpUserList),~($(MpRawList),$(OldMpFlag)))
set StatsFlag     = *($(StatsUserList),~($(StatsRawList),$(StatsFlag)))
set RxTxSlots     = *($(RxTxUserList),~($(RxTxRawList),$(RxTxSlots)))
set SpeedFlag     = *($(SpeedUserList),~($(SpeedRawList),$(SpeedFlag)))
read-syms FileDependentDlg$(!STF_LANGUAGE)
Debug-Output "MADGE: Option             = "$(Option)
Debug-Output "MADGE: SlotNumberAdapters = "$(SlotNumberAdapters)
ifcontains(i) $(Option) in $(SlotNumberAdapters)
      set DlgTemplate = "MDGEISA"
else
    set DlgTemplate = "MDGISA"
endif
Debug-Output "MADGE: DlgTemplate = "$(DlgTemplate)
ifstr(i) $(Option) == "MSMDGMPPCI"
    set Combo8Label = $(PciCombo8Label)
endif
LoadLibrary "x" $(DialogDllName) MdgDialog
EndWait
ui start "InputDlg" $(MdgDialog)
StartWait
ifstr(i) $(DLGEVENT) == "EXIT"
      set CommonStatus = STATUS_USERCANCEL
      ui pop 1
      FreeLibrary $(MdgDialog)
      goto End
else-ifstr(i) $(DLGEVENT) != "CONTINUE"
      ui pop 1
      FreeLibrary $(MdgDialog)
      goto End
endif
set IrqNumber    = $(Combo1Out)
set IoLocation   = $(Combo2Out)
set DmaChannel   = $(Combo3Out)
set RxTxSlots    = $(Combo4Out)
set StatsFlag    = $(Combo5Out)
set MpFlag       = $(Combo7Out)
set SlotNumber   = $(Combo8Out)
set SpeedFlag    = $(Combo9Out)
set MaxFrameSize = *($(EditTextOut), 1)
set LAA          = *($(EditTextOut), 2)
ui pop 1
FreeLibrary $(MdgDialog)
Debug-Output "MADGE: SlotNumber = "$(SlotNumber)
Debug-Output "MADGE: IoLocation = "$(IoLocation)
Debug-Output "MADGE: IrqNumber  = "$(IrqNumber)
Debug-Output "MADGE: DmaChannel = "$(DmaChannel)
Debug-Output "MADGE: MpFlag     = "$(MpFlag)
Debug-Output "MADGE: RxTxSlots  = "$(RxTxSlots)
Debug-Output "MADGE: StatsFlag  = "$(StatsFlag)
ifcontains(i) $(Option) not-in $(SlotNumberAdapters)
    set IoLocation    = *($(IoRawList),~($(IoUserList),$(IoLocation)))
```

```
        set OldIoLocation = *($(IoRawList),~($(IoUserList),$(OldIoLocation)))
        set IrqNumber     = *($(IrqRawList),~($(IrqUserList),$(IrqNumber)))
        set OldIrqNumber  = *($(IrqRawList),~($(IrqUserList),$(OldIrqNumber)))
    else
        set SlotNumber    = *($(SlotRawList),~($(SlotUserList),$(SlotNumber)))
        set OldSlotNumber = *($(SlotRawList),~($(SlotUserList),$(OldSlotNumber)))
    endif
    set DmaChannel    = *($(DmaRawList),~($(DmaUserList),$(DmaChannel)))
    set OldDmaChannel = *($(DmaRawList),~($(DmaUserList),$(OldDmaChannel)))
    set MpFlag        = *($(MpRawList),~($(MpUserList),$(MpFlag)))
    set OldMpFlag     = *($(MpRawList),~($(MpUserList),$(OldMpFlag)))
    set StatsFlag     = *($(StatsRawList),~($(StatsUserList),$(StatsFlag)))
    set RxTxSlots     = *($(RxTxRawList),~($(RxTxUserList),$(RxTxSlots)))
    set SpeedFlag     = *($(SpeedRawList),~($(SpeedUserList),$(SpeedFlag)))
    Debug-Output "MADGE: SlotNumber = "$(SlotNumber)
    Debug-Output "MADGE: IoLocation = "$(IoLocation)
    Debug-Output "MADGE: IrqNumber  = "$(IrqNumber)
    Debug-Output "MADGE: DmaChannel = "$(DmaChannel)
    Debug-Output "MADGE: MpFlag     = "$(MpFlag)
    Debug-Output "MADGE: RxTxSlots  = "$(RxTxSlots)
    Debug-Output "MADGE: StatsFlag  = "$(StatsFlag)
    EndWait
Adapter_Verify = +
    StartWait
    ifstr $(LAA) != ""
        LoadLibrary "x" $(DialogDllName) MdgDialog
          set FLibraryErrCtl = 1
          LibraryProcedure MdgResult $(MdgDialog) MadgeLAACheck $(LAA)
          set FLibraryErrCtl = 0
        FreeLibrary $(MdgDialog)
      ifstr $(MdgResult) != "MADGE_STATUS_SUCCESS"
             set Error = $(MdgResult)
             goto Non_Fatal
          endif
    endif
    ifint $(MaxFrameSize) > 17839
          set MaxFrameSize = 17839
          set Error        = $(TRIMMED_MFS)
          set From         = Update_Registry
          goto Non_Fatal_Info
    endif
    ifstr(i) $(!STF_NCDETINFO) == {}
        Shell $(UtilityInf), GetBusTypeDialog, $(ProductHardware$
(Option)Description) $(BusNumber)
        ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
        endif
        set BusInterfaceType = $($R1)
        set BusNumber = $($R2)
    else
        set BusInterfaceType = *($(!STF_NCDETINFO),5)
        set BusNumber = *($(!STF_NCDETINFO),6)
    endif
adapterverify = +
    Debug-Output "At adapterverify"
    Shell "" DebugConfiguration "after running dialog"
    Ifstr(i) $(AdapterDetected) != TRUE
        Goto Update_Registry
    Endif
```

```
    Debug-Output "OEMNADMA.INF: Calling Param_VerifyCard"
    Shell $(ParamInf) Param_VerifyCard $(!STF_NCDETCARD)
    Ifstr(i) $($R0) == STATUS_SUCCESSFUL
        Debug-Output "OEMNADMA.INF: Param_VerifyCard succeeded"
        Goto Update_Registry
    Endif
    Set from = Get_Adapter_Options
    Set to = Update_Registry
    Shell $(UtilityInf),RegistryErrorString,VERIFY_WARNING
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error: cannot get an error string."
        goto ShellCodeError
    endif
    set Error = $($R0)
    Goto Warning
    EndWait
Update_Registry = +
    ifint $(OldVersionExisted) == $(TRUE)
          ifstr(i) $(!NTN_InstallMode) == configure
              goto Write_Parameters
          endif
    endif
    StartWait
    ifint $(OldVersionExisted) == $(FALSE)
          Shell $(UtilityInf), AddSoftwareComponent, $(Manufacturer), +
              $(ProductSoftwareName), +
              $(ProductSoftwareName), +
              $(ProductSoftwareTitle), $(STF_CONTEXTINFNAME), +
              $(ProductSoftwareImagePath), "kernel", "NDIS", {}, "",+
              $(MadgeEventLogFile)
          Set OEM_ABANDON_SOFTWARE = TRUE
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
              goto Shell_Code_Error
          endif
          set RegistryErrorIndex = $($R0)
          set KeyProduct         = $($R1)
          Set SoftNetRulesKey    = $($R2)
          CloseRegKey $($R3)
          CloseRegKey $($R4)
          CloseRegKey $($R5)
          ifstr(i) $(RegistryErrorIndex) != NO_ERROR
              CloseRegKey $(KeyProduct)
              CloseRegKey $(SoftNetRulesKey)
              goto Fatal_Registry
          endif
          set NewValueList = {{SoftwareType,$(NoTitle),$(!REG_VT_SZ),$
(SoftwareType)},+
                              {MajorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMajorVersion)},+
                              {MinorVersion,$(NoTitle),$(!REG_VT_DWORD),$
(ProductMinorVersion)},+
                              {Title,$(NoTitle),$(!REG_VT_SZ),$(ProductSoftwareTitle)},+
                              {Description,$(NoTitle),$(!REG_VT_SZ),$
(ProductSoftwareDescription)},+
                              {PathName,$(NoTitle),$(!REG_VT_SZ),$(!STF_CWDDIR)},+
                              {ServiceName,$(NoTitle),$(!REG_VT_SZ),$
(ProductSoftwareName)},+
                              {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),1)}}
          Shell $(UtilityInf), AddValueList, $(KeyProduct), $(NewValueList)
```

```
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
              goto Shell_Code_Error
          endif
          set RegistryErrorIndex = $($R0)
          ifstr(i) $(RegistryErrorIndex) != NO_ERROR
              CloseRegKey $(KeyProduct)
              CloseRegKey $(SoftNetRulesKey)
              goto Fatal_Registry
          endif
          set NewValueList = {{type,$(NoTitle),$(!REG_VT_SZ),$
(NetRuleSoftwareType)},+
                              {use,$(NoTitle),$(!REG_VT_SZ),$(NetRuleSoftwareUse)}, +
                              {bindform,$(NoTitle),$(!REG_VT_SZ),$
(NetRuleSoftwareBindForm)}, +
                              {class,$(NoTitle),$(!REG_VT_MULTI_SZ),$
(NetRuleSoftwareClass)}, +
                              {bindable,$(NoTitle),$(!REG_VT_MULTI_SZ),$
(NetRuleSoftwareBindable)}, +
                              {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}, +
                              {Infname ,$(NoTitle),$(!REG_VT_SZ),$(STF_CONTEXTINFNAME)}
}
          Shell $(UtilityInf), AddValueList, $(SoftNetRulesKey), $(NewValueList)
          ifint $($ShellCode) != $(!SHELL_CODE_OK)
              goto Shell_Code_Error
          endif
          set RegistryErrorIndex = $($R0)
          CloseRegKey $(KeyProduct)
          CloseRegKey $(SoftNetRulesKey)
          ifstr(i) $(RegistryErrorIndex) != NO_ERROR
              goto Fatal_Registry
          endif
    endif
    Shell $(UtilityInf), AddHardwareComponent, $(ProductHardwareName),$
(STF_CONTEXTINFNAME),$(ProductKeyName)
    ifint $($R4) != -1
      Set OEM_ABANDON_OPTIONS = >($(OEM_ABANDON_OPTIONS), $(!NTN_SoftwareBase)"\
Microsoft\Windows NT\CurrentVersion\NetworkCards\"$($R4))
    endif
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
    endif
    set RegistryErrorIndex = $($R0)
    ifstr(i) $(RegistryErrorIndex) != NO_ERROR
          CloseRegKey $($R1)
          CloseRegKey $($R2)
          CloseRegKey $($R3)
          goto Fatal_Registry
    endif
    set KeyParameters   = $($R3)
    set KeyAdapterRules = $($R2)
    set AdapterNumber   = $($R4)
    set NewValueList = {{Manufacturer,$(NoTitle),$(!REG_VT_SZ),$(Manufacturer)},+
                    {Title,$(NoTitle),$(!REG_VT_SZ),"["$($R4)"] "$(ProductHardware$
(Option)Title)},+
                    {Description,$(NoTitle),$(!REG_VT_SZ),$(ProductHardware$
(Option)Description)},+
                    {ProductName,$(NoTitle),$(!REG_VT_SZ),$(ProductHardwareName)},+
                    {ServiceName,$(NoTitle),$(!REG_VT_SZ),$($R5)},+
                {OperationsSupport,$(NoTitle),$(!REG_VT_DWORD),$(ProductOpSupport)},
```

```
+
                  {InstallDate,$(NoTitle),$(!REG_VT_DWORD),*($(Now),1)}}
    Shell $(UtilityInf), AddValueList, $($R1), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    CloseRegKey $($R1)
    set TempProdName = """"$(ProductHardwareName)$(AdapterNumber)""""
    set TempBindForm = $(TempProdName)$(NetRuleHardwareBindForm)
    set NewValueList = {{type,$(NoTitle),$(!REG_VT_SZ),$(NetRuleHardwareType)},+
                {bindform,$(NoTitle),$(!REG_VT_SZ),$(TempBindForm)}, +
                {class,$(NoTitle),$(!REG_VT_MULTI_SZ),$(NetRuleHardwareClass)}, +
                {InfOption,$(NoTitle),$(!REG_VT_SZ),$(Option)}, +
                {Infname ,$(NoTitle),$(!REG_VT_SZ),$(STF_CONTEXTINFNAME)}}
    Shell $(UtilityInf), AddValueList, $(KeyAdapterRules), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    set RegistryErrorIndex = $($R0)
    ifstr(i) $(RegistryErrorIndex) != NO_ERROR
          CloseRegKey $(KeyParameters)
          CloseRegKey $(KeyAdapterRules)
          goto Fatal_Registry
    endif
    CloseRegKey $(KeyAdapterRules)
    EndWait
Write_Parameters = +
    StartWait
    ifstr(i) $(Option) == "MSMDGMPPCMCIA"
        set BusInterfaceType = 1
    endif
    ifint $(DmaChannel) == $(GENERAL_MMIO)
        set NoMmioFlag = 0
    else
        set NoMmioFlag = 1
    endif
    ifint $(DmaChannel) == $(GENERAL_PIO)
        set TransferType = $(TRANSFER_PIO)
    else-ifint $(DmaChannel) == $(GENERAL_MMIO)
        set TransferType = $(TRANSFER_MMIO)
    else
        set TransferType = $(TRANSFER_DMA)
    endif
    ifstr(i) $(Option) == "MSMDGMPPCI"
        ifint $(DmaChannel) == $(GENERAL_PIO)
            set TransferType = $(TRANFER_PIO)
        else
            set TransferType = $(TRANSFER_DMA)
        endif
    endif
    set AdapterTypeFlag = $(AdapterTypeFlag$(Option))
    ifcontains(i) $(Option) not-in $(SlotNumberAdapters)
      set NewValueList = {{InterruptNumber,$(NoTitle),$(!REG_VT_DWORD),$
(IrqNumber)},+
                {BusType,        $(NoTitle),$(!REG_VT_DWORD),$
(BusInterfaceType)},+
                {BusNumber,      $(NoTitle),$(!REG_VT_DWORD),$(BusNumber)},+
                {MediaType,      $(NoTitle),$(!REG_VT_DWORD),2},+
                {IoLocation,     $(NoTitle),$(!REG_VT_DWORD),$(IoLocation)},+
```

```
                  {IoBaseAddress,  $(NoTitle),$(!REG_VT_DWORD),$(IoLocation)},+
                  {DmaChannel,     $(NoTitle),$(!REG_VT_DWORD),$(DmaChannel)},+
                  {NoMmio,         $(NoTitle),$(!REG_VT_DWORD),$(NoMmioFlag)},+
                  {TransferType,   $(NoTitle),$(!REG_VT_DWORD),$(TransferType)},+
                        {RxTxSlots,      $(NoTitle),$(!REG_VT_DWORD),$(RxTxSlots)},
+
                        {Multiprocessor, $(NoTitle),$(!REG_VT_DWORD),$(MpFlag)},+
                  {MaxFrameSize,   $(NoTitle),$(!REG_VT_DWORD),$(MaxFrameSize)},+
                        {AdapterType,    $(NoTitle),$(!REG_VT_DWORD),$
(AdapterTypeFlag)}}
    else
      set NewValueList = {+
                  {BusType,        $(NoTitle),$(!REG_VT_DWORD),$
(BusInterfaceType)},+
                  {BusNumber,      $(NoTitle),$(!REG_VT_DWORD),$(BusNumber)},+
                  {MediaType,      $(NoTitle),$(!REG_VT_DWORD),2},+
                  {SlotNumber,     $(NoTitle),$(!REG_VT_DWORD),$(SlotNumber)},+
                  {NoMmio,         $(NoTitle),$(!REG_VT_DWORD),$(NoMmioFlag)},+
                  {TransferType,   $(NoTitle),$(!REG_VT_DWORD),$(TransferType)},+
                        {RxTxSlots,      $(NoTitle),$(!REG_VT_DWORD),$(RxTxSlots)},
+
                        {Multiprocessor, $(NoTitle),$(!REG_VT_DWORD),$(MpFlag)},+
                  {MaxFrameSize,   $(NoTitle),$(!REG_VT_DWORD),$(MaxFrameSize)},+
                        {AdapterType,    $(NoTitle),$(!REG_VT_DWORD),$
(AdapterTypeFlag)}}
    endif
    ifstr(i) $(ID$(Option)) != ""
      set NewValueList = >($(NewValueList),+
                        {$(IDV$(Option)), $(NoTitle), $(!REG_VT_DWORD), $(ID$
(Option))})
    endif
    ifstr $(LAA) != ""
      set NewValueList = >($(NewValueList),+
                        {NetworkAddress, $(NoTitle), $(!REG_VT_SZ), $(LAA)})
    endif
    ifstr $(StatsFlag) == "1"
      set NewValueList = >($(NewValueList),+
                        {PromiscuousModeX, $(NoTitle), $(!REG_VT_DWORD), $
(StatsFlag)})
    endif
    ifstr $(SpeedFlag) == "1"
      set NewValueList = >($(NewValueList),+
                        {Force4, $(NoTitle), $(!REG_VT_DWORD), 1})
    endif
    ifstr $(SpeedFlag) == "2"
      set NewValueList = >($(NewValueList),+
                        {Force16, $(NoTitle), $(!REG_VT_DWORD), 1})
    endif
    ifstr(i) $(Option) == "MSMDGMPPCMCIA"
      set NewValueList = >($(NewValueList),+
                        {Pcmcia, $(NoTitle), $(!REG_VT_DWORD), 1})
    endif
    Shell  $(UtilityInf), AddValueList, $(KeyParameters), $(NewValueList)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    set RegistryErrorIndex = $($R0)
    Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
          CloseRegKey $(KeyParameters)
```

```
            goto Fatal_Registry
        endif
        ifstr(i) $(!STF_GUI_UNATTENDED) == "YES"
            Shell $(UtilityInf),AddDefaultNetCardParameters,$(KeyParameters)
        endif
        ifstr $(OldLAA) != ""
            ifstr $(LAA) == ""
                DeleteRegValue $(KeyParameters) NetworkAddress
            endif
        endif
        ifstr $(OldSpeedFlag) == "1"
            ifstr $(SpeedFlag) != "1"
                DeleteRegValue $(KeyParameters) Force4
            endif
        endif
        ifstr $(OldSpeedFlag) == "2"
            ifstr $(SpeedFlag) != "2"
                DeleteRegValue $(KeyParameters) Force16
            endif
        endif
        ifstr $(OldStatsFlag) != "0"
            ifstr $(StatsFlag) == "0"
                DeleteRegValue $(KeyParameters) PromiscuousModeX
            endif
        endif
        CloseRegKey $(KeyParameters)
        EndWait
        goto Successful
    Binding_Adapter =+
        set Error = "Binding: Sorry, not yet implemented."
        goto Fatal
    Remove_Adapter = +
        StartWait
        ifstr(i) $(ProductKeyName) == $(!NTN_RegBase)
            Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
                $(ProductSoftwareName)
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                goto Shell_Code_Error
            endif
            set RegistryErrorIndex = $($R0)
            ifstr(i) $(RegistryErrorIndex) != NO_ERROR
                goto Fatal_Registry
            endif
        else
            Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
                $(ProductSoftwareName), $(!NTN_RegBase)
            ifint $($ShellCode) != $(!SHELL_CODE_OK)
                goto Shell_Code_Error
            endif
            set RegistryErrorIndex = $($R0)
            ifstr(i) $(RegistryErrorIndex) != NO_ERROR
                goto Fatal_Registry
            endif
        endif
        EndWait
        goto End
    Upgrade_Software = +
        StartWait
        ifstr(i) $(ProductKeyName) == $(!NTN_RegBase)
```

```
          OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED)
KeyProduct
          ifstr $(KeyProduct) != $(KeyNull)
              GetRegValue $(KeyProduct),"MajorVersion", VersionInfo
              set Version = *($(VersionInfo), 4)
              shell $(UtilityInf) GetInfFileNameFromRegistry $(KeyProduct)
              ifint $($ShellCode) != $(!SHELL_CODE_OK)
                goto Shell_Code_Error
            endif
              set !UG_Filename = $($R0)
              ifstr(i) $(!UG_Filename) != ""
                install "Install-Update"
                    ifstr(i) $(STF_INSTALL_OUTCOME) != STF_SUCCESS
                    goto Fatal
                endif
              endif
              CloseRegKey $(KeyProduct)
              OpenRegKey $(!REG_H_LOCAL) "" $(ProductKeyName) $(MAXIMUM_ALLOWED)
KeyProduct
              SetRegValue $(KeyProduct) {MajorVersion,$(NoTitle),$(!REG_VT_SZ),$
(ProductMajorVersion)}
              SetRegValue $(KeyProduct) {MinorVersion,$(NoTitle),$(!REG_VT_SZ),$
(ProductMinorVersion)}
              ifint $(Version) != $(ProductMajorVersion)
            endif
              CloseRegKey $(KeyProduct)
          else
              goto Fatal_Registry
          endif
        set iSearch = 1
nextnetcard = +
        Shell $(UtilityInf), FindNextNetworkCard, $(ProductHardwareName), $
(iSearch)
        set KeyNetcard = $($R0)
        set iSearch = $($R1)
        Debug-Output "OemNadEp.Inf: FindNextNetworkCard "$(KeyNetcard)","$(iSearch)
        Ifstr $(KeyNetcard) != $(KeyNull)
            Debug-Output "OemNadEp.Inf: Setting OperationsSupport value"
            SetRegValue $(KeyNetcard) {OperationsSupport,$(NoTitle),$(!
REG_VT_DWORD),$(ProductOpSupport)}
            CloseRegKey $(KeyNetcard)
            goto nextnetcard
        Endif
    else
        set Error          = ""
        set OEM_ABANDON_ON = FALSE
        goto Not_Supported
    endif
    EndWait
    goto End
Successful = +
    goto End
Abandon = +
    ForListDo $(OEM_ABANDON_OPTIONS)
      Shell $(UtilityInf), RemoveHardwareComponent, $(Manufacturer), +
          $(ProductSoftwareName), $($)
      ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
      endif
```

```
      set RegistryErrorIndex = $($R0)
      Ifstr(i) $(RegistryErrorIndex) != NO_ERROR
          goto Fatal_Registry
      endif
    EndForListDo
    ifstr(i) $(OEM_ABANDON_SOFTWARE) == TRUE
      Shell $(UtilityInf), RemoveSoftwareComponent, $(Manufacturer), +
          $(ProductSoftwareName), FALSE
      ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
      endif
      set RegistryErrorIndex = $($R0)
      ifstr(i) $(RegistryErrorIndex) != NO_ERROR
          goto Fatal_Registry
      endif
    endif
    goto End
Warning = +
    EndWait
    Shell $(SubroutineInf) SetupMessage, $(!STF_LANGUAGE), "WARNING", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    ifstr(i) $($R1) == "OK"
      goto $(To)
    else-ifstr(i) $($R1) == "CANCEL"
      goto $(From)
    endif
    goto End
Non_Fatal_Info = +
    set Severity     = STATUS
    set CommonStatus = STATUS_USERCANCEL
    goto Non_Fatal_Msg
Non_Fatal = +
    set  Severity = NONFATAL
    goto Non_Fatal_Msg
Non_Fatal_Msg = +
    EndWait
    ifstr(i) $(Error) == ""
      set Severity = NONFATAL
      Shell $(UtilityInf) RegistryErrorString "SETUP_FAIL"
      ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
      endif
      set Error = $($R0)
    endif
    Shell $(SubroutineInf) SetupMessage, $(!STF_LANGUAGE), "NONFATAL", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    ifstr(i) $($R1) == "OK"
      goto $(From)
    endif
    goto End
Fatal_Registry = +
    Shell $(UtilityInf) RegistryErrorString $(RegistryErrorIndex)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
```

```
    set Error = $($R0)
    goto Fatal
fataldetect = +
    Debug-Output "At fataldetect"
    Shell $(UtilityInf),RegistryErrorString,CANNOT_DETECT
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
        Debug-Output "ShellCode error: cannot get an error string."
        goto ShellCodeError
    endif
    set Error = $($R0)
    Goto fatal
Fatal = +
    EndWait
    ifstr(i) $(Error) == ""
      Shell $(UtilityInf) RegistryErrorString "SETUP_FAIL"
      ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
      endif
      set Error = $($R0)
    endif
    Shell $(SubroutineInf) SetupMessage, $(!STF_LANGUAGE), "FATAL", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    goto Set_Failed
Not_Supported = +
    EndWait
    ifstr(i) $(Error) == ""
      Shell $(UtilityInf) RegistryErrorString "OPERATION_UNIMPLEMENTED"
      ifint $($ShellCode) != $(!SHELL_CODE_OK)
          goto Shell_Code_Error
      endif
      set Error = $($R0)
    endif
    Shell $(SubroutineInf) SetupMessage, $(!STF_LANGUAGE), "FATAL", $(Error)
    ifint $($ShellCode) != $(!SHELL_CODE_OK)
      goto Shell_Code_Error
    endif
    goto End
Shell_Code_Error = +
    EndWait
    set DlgType      = "MessageBox"
    set STF_MB_TITLE = $(ShellCodeErrorTitle)
    set STF_MB_TEXT  = $(ShellCodeErrorText)
    set STF_MB_TYPE  = 1
    set STF_MB_ICON  = 3
    set STF_MB_DEF   = 1
    ui start "Error Message"
    goto Set_Failed
Set_Failed = +
    set CommonStatus = STATUS_FAILED
    ifstr(i) $(OEM_ABANDON_ON) == TRUE
      set OEM_ABANDON_ON = FALSE
      goto Abandon
    endif
    goto End
End = +
    goto Term
Term = +
```

```
    Return $(CommonStatus)
[Install-Update]
   set STF_VITAL        = ""
   set STF_OVERWRITE    = "VERIFYSOURCEOLDER"
   AddSectionFilesToCopyList Files-Helper $(SrcDir) $(!STF_WINDOWSSYSPATH)
   AddSectionFilesToCopyList Files-Driver $(SrcDir) $(!STF_WINDOWSSYSPATH)\drivers
   exit
[Install-Option]
    set STF_VITAL        = ""
    AddSectionFilesToCopyList Files-Helper $(SrcDir) $(!STF_WINDOWSSYSPATH)
    AddSectionFilesToCopyList Files-Driver $(SrcDir) $(!STF_WINDOWSSYSPATH)\drivers
    ifstr(i) $(DoCopy) == "YES"
        set !STF_NCPA_FLUSH_COPYLIST = TRUE
        CopyFilesInCopyList
    else
        LibraryProcedure STATUS,$(!NCPA_HANDLE), CopySingleFile $(!STF_HWND) $
(SrcDir)\mdgmpdlg.DLL $(!STF_WINDOWSSYSPATH)\mdgmpdlg.DLL
        LibraryProcedure STATUS,$(!NCPA_HANDLE), CopySingleFile $(!STF_HWND) $
(SrcDir)\mdgmpdlg.HLP $(!STF_WINDOWSSYSPATH)\mdgmpdlg.HLP
    Endif
    exit
[Source Media Descriptions]
    1 = "Windows NT Workstation CD-ROM" , TAGFILE = cdrom_w.40
[Signature]
    FileType = MICROSOFT_FILE
[GetSignature]
    read-syms Signature
    return $(FileType)
[ProductType]
STF_PRODUCT  = Winnt
STF_PLATFORM = I386
[Files-Inf]
2,    oemsetup.inf,      SIZE=1000, RENAME=$(!UG_Filename)
[Files-Driver]
1,MDGMPORT.BIN , SIZE=999
1,MDGMPORT.SYS , SIZE=999
[Files-Helper]
1,MDGMPDLG.DLL , SIZE=999
1,MDGMPDLG.HLP , SIZE=999
[LanguagesSupported]
    ENG
[OptionsTextENG]
    MSMDGMPISA   = "Madge Smart 16/4 AT Ringnode"
    MSMDGMPATP   = "Madge Smart 16/4 AT Plus Ringnode"
    MSMDGMPISAC  = "Madge Smart 16/4 ISA Client Ringnode"
    MSMDGMPISACP = "Madge Smart 16/4 ISA Client Plus Ringnode"
    MSMDGMPPC    = "Madge Smart 16/4 PC Ringnode"
    MSMDGMPSM16  = "Madge Smart 16 Ringnode"
    MSMDGMPPNP   = "Madge Smart 16/4 ISA Client PnP Ringnode"
    MSMDGMPMCA   = "Madge Smart 16/4 MC Ringnode"
    MSMDGMPMC32  = "Madge Smart 16/4 MC32 Ringnode"
    MSMDGMPEISA  = "Madge Smart 16/4 EISA Ringnode"
    MSMDGMPPCI   = "Madge Smart 16/4 PCI Ringnode and BM2"
    MSMDGMPPCIBM = "Madge Smart 16/4 PCI Ringnode BM"
    MSMDGMPPCMCIA = "Madge Smart 16/4 PCMCIA Ringnode"
[OptionsTextISAENG]
    MSMDGMPISA   = "Madge Smart 16/4 AT Ringnode"
    MSMDGMPATP   = "Madge Smart 16/4 AT Plus Ringnode"
    MSMDGMPISAC  = "Madge Smart 16/4 ISA Client Ringnode"
```

```
    MSMDGMPISACP = "Madge Smart 16/4 ISA Client Plus Ringnode"
    MSMDGMPPC    = "Madge Smart 16/4 PC Ringnode"
    MSMDGMPSM16  = "Madge Smart 16 Ringnode"
    MSMDGMPPNP   = "Madge Smart 16/4 ISA Client PnP Ringnode"
[OptionsTextMCAENG]
    MSMDGMPMCA   = "Madge Smart 16/4 MC Ringnode"
    MSMDGMPMC32  = "Madge Smart 16/4 MC32 Ringnode"
[OptionsTextEISAENG]
    MSMDGMPEISA  = "Madge Smart 16/4 EISA Ringnode"
    MSMDGMPISA   = "Madge Smart 16/4 AT Ringnode"
    MSMDGMPATP   = "Madge Smart 16/4 AT Plus Ringnode"
    MSMDGMPISAC  = "Madge Smart 16/4 ISA Client Ringnode"
    MSMDGMPISACP = "Madge Smart 16/4 ISA Client Plus Ringnode"
    MSMDGMPPC    = "Madge Smart 16/4 PC Ringnode"
    MSMDGMPSM16  = "Madge Smart 16 Ringnode"
    MSMDGMPPNP   = "Madge Smart 16/4 ISA Client PnP Ringnode"
[OptionsTextPCIENG]
    MSMDGMPPCI   = "Madge Smart 16/4 PCI Ringnode and BM2"
    MSMDGMPPCIBM = "Madge Smart 16/4 PCI Ringnode BM"
[OptionsTextPCMCIAENG]
    MSMDGMPPCMCIA = "Madge Smart 16/4 PCMCIA Ringnode"
[IoLocationChoices]
    "0x0300",768
    "0x0a20",2592
    "0x1a20",6688
    "0x2a20",10784
    "0x3a20",14880
    "0x4a20",18976
    "0x4e20",20000
    "0x5a20",23072
    "0x5e20",24096
    "0x6a20",27168
    "0x6e20",28192
    "0x7a20",31264
    "0x7e20",32288
    "0x0140",320
    "0x0920",2336
    "0x0940",2368
    "0x0960",2400
    "0x0980",2432
    "0x0a40",2624
    "0x0a60",2656
    "0x0a80",2688
    "0x0aa0",2720
    "0x0b20",2848
    "0x0b40",2880
    "0x0b60",2912
    "0x0b80",2944
    "s",65535
[DmaChannelChoices]
    "PIO",0
    "DMA  01",1
    "DMA  02",2
    "DMA  03",3
    "DMA  04",4
    "DMA  05",5
    "DMA  06",6
    "DMA  07",7
    "DMA  08",8
```

```
    "DMA ��? 09",9
    "DMA ��? 10",10
    "DMA ��? 11",11
    "DMA ��? 12",12
    "DMA ��? 13",13
    "DMA ��? 14",14
    "DMA ��? 15",15
    "DMA",500
    "MMIO",501
    "?s??",65535
[IrqNumberChoices]
    "01",1
    "02",2
    "03",3
    "04",4
    "05",5
    "06",6
    "07",7
    "08",8
    "09",9
    "10",10
    "11",11
    "12",12
    "13",13
    "14",14
    "15",15
    "?s??",65535
[SlotNumberChoices]
    "00",0
    "01",1
    "02",2
    "03",3
    "04",4
    "05",5
    "06",6
    "07",7
    "08",8
    "09",9
    "10",10
    "11",11
    "12",12
    "13",13
    "14",14
    "15",15
    "16",16
    "17",17
    "18",18
    "19",19
    "20",20
    "21",21
    "22",22
    "23",23
    "24",24
    "25",25
    "26",26
    "27",27
    "28",28
    "29",29
    "30",30
```

```
    "31",31
    "▯s▯▯",65535
[RxTxSlotsChoices]
    "Rx=02 Tx=02",0
    "Rx=03 Tx=03",1
    "Rx=04 Tx=04",2
    "Rx=06 Tx=06",3
    "Rx=08 Tx=08",4
    "Rx=10 Tx=10",5
[StatsChoices]
    "▯▯▯▯",0
    "▯L▯▯",1
[MpChoices]
    "1",0
    "▯▯▯▯",1
    "▯s▯▯",65535
[FileConstantsENG]
FunctionTitle   = "MdgMPort v"$(ProductVersion)": Madge Smart Ringnode ▯▯ı▯▯▯"
ProductSoftwareTitle       = "MdgMPort"
ProductSoftwareDescription = "MdgMPort: Madge Smart Ringnode NDIS3 ▯▯▯▯ ▯▯''▯▯"
ProductHardwareMSMDGMPISATitle   = "Madge Smart 16/4 AT Ringnode"
ProductHardwareMSMDGMPATPTitle   = "Madge Smart 16/4 AT Plus Ringnode"
ProductHardwareMSMDGMPISACTitle  = "Madge Smart 16/4 ISA Client Ringnode"
ProductHardwareMSMDGMPISACPTitle = "Madge Smart 16/4 ISA Client Plus Ringnode"
ProductHardwareMSMDGMPPCTitle    = "Madge Smart 16/4 PC Ringnode"
ProductHardwareMSMDGMPEISATitle  = "Madge Smart 16/4 EISA Ringnode"
ProductHardwareMSMDGMPMCATitle   = "Madge Smart 16/4 MC Ringnode"
ProductHardwareMSMDGMPMC32Title  = "Madge Smart 16/4 MC32 Ringnode"
ProductHardwareMSMDGMPSM16Title  = "Madge Smart 16 Ringnode"
ProductHardwareMSMDGMPPNPTitle   = "Madge Smart 16/4 ISA Client PnP Ringnode"
ProductHardwareMSMDGMPPCITitle   = "Madge Smart 16/4 PCI Ringnode and BM2"
ProductHardwareMSMDGMPPCIBMTitle = "Madge Smart 16/4 PCI Ringnode BM"
ProductHardwareMSMDGMPPCMCIATitle  = "Madge Smart 16/4 PCMCIA Ringnode"
ProductHardwareMSMDGMPISADescription   = "Madge Smart 16/4 AT Ringnode"
ProductHardwareMSMDGMPATPDescription   = "Madge Smart 16/4 AT Plus Ringnode"
ProductHardwareMSMDGMPISACDescription  = "Madge Smart 16/4 ISA Client Ringnode"
ProductHardwareMSMDGMPISACPDescription = "Madge Smart 16/4 ISA Client Plus
Ringnode"
ProductHardwareMSMDGMPPCDescription    = "Madge Smart 16/4 PC Ringnode"
ProductHardwareMSMDGMPEISADescription  = "Madge Smart 16/4 EISA Ringnode"
ProductHardwareMSMDGMPMCADescription   = "Madge Smart 16/4 MC Ringnode"
ProductHardwareMSMDGMPMC32Description  = "Madge Smart 16/4 MC32 Ringnode"
ProductHardwareMSMDGMPSM16Description  = "Madge Smart 16 Ringnode"
ProductHardwareMSMDGMPPNPDescription   = "Madge Smart 16/4 ISA Client PnP Ringnode"
ProductHardwareMSMDGMPPCIDescription   = "Madge Smart 16/4 PCI Ringnode and BM2"
ProductHardwareMSMDGMPPCIBMDescription = "Madge Smart 16/4 PCI Ringnode BM"
ProductHardwareMSMDGMPPCMCIADescription   = "Madge Smart 16/4 PCMCIA Ringnode"
SpeedUserListMSMDGMPISA    = {"▯▯▯▯▯▯Ő16" ,"▯▫MBit/s ▯q▯▯▯"}
SpeedRawListMSMDGMPISA     = {0, 2}
SpeedUserListMSMDGMPATP    = {"▯▯▯▯▯▯Ő4" ,"▯▫MBit/s ▯q▯▯▯", "16MBit/s ▯q▯▯▯"}
SpeedRawListMSMDGMPATP     = {0, 1, 2}
SpeedUserListMSMDGMPISAC   = {"▯▯▯▯▯▯Ő16" ,"▯▫MBit/s ▯q▯▯▯"}
SpeedRawListMSMDGMPISAC    = {0, 2}
SpeedUserListMSMDGMPISACP  = {"▯▯▯▯▯▯Ő4" ,"▯▫MBit/s ▯q▯▯▯", "16MBit/s ▯q▯▯▯"}
SpeedRawListMSMDGMPISACP   = {0, 1, 2}
SpeedUserListMSMDGMPPC     = {"▯▯▯▯▯▯Ő16" ,"▯▫MBit/s ▯q▯▯▯"}
SpeedRawListMSMDGMPPC      = {0, 2}
SpeedUserListMSMDGMPEISA   = {"▯▯▯▯▯▯Ő▫▯"}
SpeedRawListMSMDGMPEISA    = {0}
```

```
SpeedUserListMSMDGMPMCA    = {"[illegible]Ő[illegible]"}
SpeedRawListMSMDGMPMCA     = {0}
SpeedUserListMSMDGMPMC32   = {"[illegible]Ő[illegible]"}
SpeedRawListMSMDGMPMC32    = {0}
SpeedUserListMSMDGMPSM16   = {"[illegible] 16MBit/s"}
SpeedRawListMSMDGMPSM16    = {0}
SpeedUserListMSMDGMPPNP    = {"[illegible]Ő4" ,"[illegible]MBit/s [illegible]q[illegible]", "16MBit/s [illegible]q[illegible]"}
SpeedRawListMSMDGMPPNP     = {0, 1, 2}
SpeedUserListMSMDGMPPCI    = {"[illegible]Ő4" ,"[illegible]MBit/s [illegible]q[illegible]", "16MBit/s [illegible]q[illegible]"}
SpeedRawListMSMDGMPPCI     = {0, 1, 2}
SpeedUserListMSMDGMPPCIBM  = {"[illegible]Ő4" ,"[illegible]MBit/s [illegible]q[illegible]", "16MBit/s [illegible]q[illegible]"}
SpeedRawListMSMDGMPPCIBM   = {0, 1, 2}
SpeedUserListMSMDGMPPCMCIA = {"[illegible]Ő4" ,"[illegible]MBit/s [illegible]q[illegible]", "16MBit/s [illegible]q[illegible]"}
SpeedRawListMSMDGMPPCMCIA  = {0, 1, 2}
AdapterTypeFlagMSMDGMPISA    = 100
AdapterTypeFlagMSMDGMPATP    = 100
AdapterTypeFlagMSMDGMPISAC   = 100
AdapterTypeFlagMSMDGMPISACP  = 100
AdapterTypeFlagMSMDGMPPC     = 100
AdapterTypeFlagMSMDGMPEISA   = 500
AdapterTypeFlagMSMDGMPMCA    = 600
AdapterTypeFlagMSMDGMPMC32   = 600
AdapterTypeFlagMSMDGMPSM16   = 400
AdapterTypeFlagMSMDGMPPNP    = 300
AdapterTypeFlagMSMDGMPPCI    = 700
AdapterTypeFlagMSMDGMPPCIBM  = 700
AdapterTypeFlagMSMDGMPPCMCIA = 200
CANNOT_FIND_ANY_CARD = "[illegible] Madge Smart Ringnode [illegible]. [illegible]B"+
                       "Madge Smart Ringode [illegible]Б[illegible]Ă[illegible]"+
                       "[illegible]I[illegible]?A[illegible]I[illegible]"+
                       "[illegible]BEISA [illegible] MCA [illegible]Á[illegible] EISA "+
                       "[illegible]hè[illegible] PS/2 [illegible]Ă[illegible]g[illegible]ı[illegible]"+
                       "[illegible]ъ[illegible]Бр[illegible]Ă[illegible]B"+
                       "[illegible]. [illegible]A[illegible]I[illegible]APCI [illegible]Б[illegible]"+
                       "[illegible]. [illegible]B"
TRIMMED_MFS = "[illegible]: 16Mbps [illegible]w[illegible] 17839 [illegible] [illegible]A"+
          "  [illegible]w[illegible]. [illegible]B[illegible]l[illegible]17839 [illegible]"+
          "  [illegible]X[illegible]. [illegible]B4Mbps [illegible]s[illegible]Ă[illegible]"+
          "  [illegible]4472 [illegible] [illegible]A[illegible]"+
          "  [illegible]I[illegible]l[illegible]A[illegible] [illegible]A[illegible]"+
          "  [illegible]L[illegible]. [illegible]B"
ShellCodeErrorTitle = "[illegible]: "$(FunctionTitle)
ShellCodeErrorText  = "[illegible] [illegible] [illegible]B"
ProCaption   = "Windows NT [illegible]ı[illegible]"
ProCancel    = "[illegible]"
ProCancelMsg = "Windows NT [illegible]I[illegible]Ă[illegible]. [illegible]B"+
               "[illegible]~[illegible]. [illegible]?"
ProCancelCap = "[illegible] [illegible]ı[illegible] [illegible]"
ProText1     = "[illegible]:"
ProText2     = "[illegible]:"
[DialogConstantsENG]
Help        = "[illegible](&H)"
Exit        = "[illegible](&C)"
OK          = "OK(&O)"
Continue    = "[illegible]s(&O)"
Cancel      = "[illegible](&A)"
HelpContext = ""
OldValueTitleInstall   = "[illegible]h[illegible][illegible]"
OldValueTitleConfigure = "[illegible]V\[illegible]Ă[illegible]"
```

```
ConsultHelp = "[illegible]"
[FileDependentDlgENG]
DlgType     = "RadioCombination"
DlgTemplate = $(Option)
Caption = $(FunctionTitle)
Combo1Label = "IRQ [illegible](&I):"
Combo2Label = "I/O [illegible](&L):"
Combo3Label = "[illegible](&T):"
Combo4Label = "Rx/Tx [illegible](&B):"
Combo5Label = "[illegible](&S):"
Combo6Label = "[illegible] Madge [illegible]"
Combo7Label = "PC [illegible](&P):"
Combo8Label = "[illegible](&N):"
Combo9Label = "[illegible](&R):"
PciCombo8Label = "PCI [illegible] ID(&D):"
Combo1List = $(IrqList)
Combo1Out  = $(IrqNumber)
Combo2List = $(IoList)
Combo2Out  = $(IoLocation)
Combo3List = $(DmaList)
Combo3Out  = $(DmaChannel)
Combo4List = $(RxTxUserList)
Combo4Out  = $(RxTxSlots)
Combo5List = $(StatsUserList)
Combo5Out  = $(StatsFlag)
Combo6List = $(AdapterDescList)
Combo6Out  = $(AdapterType)
Combo7List = $(MpList)
Combo7Out  = $(MpFlag)
Combo8List = $(SlotList)
Combo8Out  = $(SlotNumber)
Combo9List = $(SpeedUserList)
Combo9Out  = $(SpeedFlag)
ComboListItemsIn  = {Combo1List, Combo2List, Combo3List, Combo4List, Combo5List,
Combo6List, Combo7List, Combo8List, Combo9List}
ComboListItemsOut = {Combo1Out, Combo2Out, Combo3Out, Combo4Out, Combo5Out,
Combo6Out, Combo7Out, Combo8Out, Combo9Out}
Edit1Label  = "[illegible](&F):"
Edit2Label  = "LAA(&L):"
EditTextIn  = {$(MaxFrameSize), $(LAA)}
EditTextLim = {5,17}
EditTextOut = {}
CBOptionsGreyed = {}
NotifyFields = {NO, NO, NO, NO, NO, NO, NO, NO}
HelpContext = $(MadgeHelpId$(Option))
AdapterTitle = #(OptionsText$(!STF_LANGUAGE),$(Option),1)
```