INTISARI

Personal Computer (PC) adalah suatu perangkat yang dapat digunakan untuk membantu beberapa kegiatan manusia, terutama dalam bidang elektronika. Piranti *Polyphonic Keyboard* berbasis PC adalah piranti yang menggabungkan suatu perangkat luar dengan PC yang akan menjalankan program pengaksesan *Sound Card* sesuai dengan data masukan dari perangkat keras *Keyboard*.

Dalam proses pengiriman data dari *Keyboard* ke PC melalui suatu *Interface* sebagai penyesuai tegangan dan arus yang masuk ke PC. Penggunaan PPI 8255 dimaksudkan agar masukan data secara paralel 3 kali 8 *Bit* dapat dideteksi dengan cepat oleh program. Program yang dibuat menggunakan bahasa pemrograman *Borland Delphi 5.0* ini akan menerima data kemudian diproses untuk mengakses *Sound Card,* dan diterjemahkan sebagai bunyi nada yang sama sesuai dengan nada yang ditekan pada *Keyboard*.

Keyboard 2 oktaf yang dibuat dapat dimainkan dengan menyambungnya melalui *LPT port (port Printer)* dan menjalankan program *Controller* pada PC. *User* dapat memainkan dalam 128 pilihan jenis suara berbeda dan mengatur *Volume* suara melalui tombo-tombol yang sudah disediakan. Pengiriman data yang paralel memungkinkan dimaikannya 23 tuts dan menghasilkan suara secara bersamaan *(23 Polyphonic)*.

ABSTRACT

Personal Computer (PC) is a device which could be used for helping human activities, expecially in electric one. "Polyphonic keyboard based on PC" is a device which combining a component and PC that will run sound card accessing program associated with data input from Keyboard as a hardware.

Process of sending data from keyboard to PC trough an interface as a voltage and current adaptor that connected to PC. Using of PPI 8255 is for fast detecting 3 port parallel (8 bit) data input by program. Borland Delphi 5.0 programming language as a software will receive data and processed for accessing sound card, and changing as atone of sound as wellas tone that prassed on keyboard.

The 2 octave keyboard can be played by connecting it trough LPT port (printer port) and running the controller program on PC. One (User) can plays 128 different sound register and arranged volume of sound from layout controller. The parallel data transfer permints to playing 23 tuts and sounding at the same time (23 polyphonic).

```
function TMidiOut.GetChannelVolume(Channel: TStereoChannel): Word;
begin
  midiOutGetVolume(FHandle, @FVolume);
  Result := FVolume;
end;

procedure TMidiOut.SetChannelVolume(Channel: TStereoChannel; const Value: Word);
begin
  if Channel = scLeft then
    SetLRVolume(Value, ChannelVolume[scRight])
  else
    SetLRVolume(ChannelVolume[scLeft], Value);
end;

function TMidiOut.GetVolume: Word;
begin
  Result := GetChannelVolume(scLeft);
end;

procedure TMidiOut.SetVolume(const Value: Word);
begin
  SetLRVolume(Value, Value);
end;

procedure TMidiOut.SetLRVolume(const LeftValue, RightValue: Word);
var
  Value: DWord;
begin
  with LongRec(Value) do
  begin
    Lo := LeftValue;
    Hi := RightValue;
  end;
  if Value <> FVolume then
  begin
    if (MIDICAPS_VOLUME and FDeviceCaps.dwSupport) <> 0 then
      MidiOutCheck(midiOutSetVolume(FHandle, Value));
    FVolume := Value;
  end;
end;

initialization
finalization
  FMidiOutputs.Free;
end.
```

LAMPIRAN F : LISTING UNIT JCLResource.pas

File ini berfungsi untuk mendeteksi dan membaca sumber-suber data yang akan diakses.

```
{$I JCL.INC}

{$WEAKPACKAGEUNIT ON}

interface

// Base
resourcestring
  RsWin32Prefix        = 'Win32: %s (%u)';
  RsDynArrayError      = 'DynArrayInitialize: ElementSize out of bounds';
  RsSysErrorMessageFmt = 'Win32 Error %d (%x)';

// Classes
resourcestring
  RsVMTMemoryWriteError = 'Error writing VMT memory (%s)';

// COM
resourcestring
  RsComInvalidParam      = 'An invalid parameter was passed to the routine. If a
parameter was' +
    ' expected, it might be an unassigned item or nil pointer';
  RsComFailedStreamRead  = 'Failed to read all of the data from the specified
stream';
  RsComFailedStreamWrite = 'Failed to write all of the data into the specified
stream';

// Complex
resourcestring
  RsComplexInvalidString = 'Failed to create a complex number from the string
provided';

// Console
resourcestring
  RsCannotRaiseSignal = 'Cannot raise %s signal.';

// Counter
resourcestring
  RsNoCounter = 'No high performance counters supported';


//  DateTime
resourcestring
  RsMakeUTCTime     = 'Error converting to UTC time. Time zone could not be
determined';
  RsDateConversion = 'Error illegal date or time format';

//  Debug
  // Diagnostics

resourcestring
  RsDebugAssertValidPointer = 'Invalid Pointer passed to AssertValid';
  RsDebugAssertValidString  = 'Invalid string passed to AssertValid';

  // TMapFiles

  RsDebugMapFileExtension = '.map'; // do not localize
  RsDebugNoProcessInfo    = 'Unable to obtain process information';
  RsDebugSnapshot         = 'Failure creating toolhelp32 snapshot';

// MSG
```

```
resourcestring
  EDIError001 = 'Could not set interchange at index [%s], Index too high.';
  EDIError002 = 'Could not set interchange at index [%s], Index too low.';
  EDIError003 = 'Could not set interchange at index [%s].';
  EDIError004 = 'Could not save edi file.  File name and path not specified.';
  EDIError005 = 'Could not save edi file.  File name and path not specified.';
  EDIError006 = 'Could not open edi file.  File not specified.';
  EDIError007 = 'Could not get interchange at index [%s], Interchanges does not
exist.';
  EDIError008 = 'Could not get interchange at index [%s], Index too high.';
  EDIError009 = 'Could not get interchange at index [%s], Index too low.';
  EDIError010 = 'Could not get interchanges at index [%s], There were no
interchanges to get.';
  EDIError011 = 'Could not find interchange control header.';
  EDIError012 = 'Could not find interchange control trailer segment terminator.';
  EDIError013 = 'Could not find interchange control trailer.';
  EDIError014 = 'Could not find interchange control trailer or garbage at end of
file.';
  EDIError015 = 'Could not delete interchanges at index [%s].';
  EDIError016 = 'Could not delete interchange at index [%s].';
  EDIError017 = 'Could not set functional group at index [%s], Index too high.';
  EDIError018 = 'Could not set functional group at index [%s], Index too low.';
  EDIError019 = 'Could not set functional group at index [%s].';
  EDIError020 = 'Could not get functional group at index [%s], Functional Group
does not exist.';
  EDIError021 = 'Could not get functional group at index [%s], Index too high.';
  EDIError022 = 'Could not get functional group at index [%s], Index too low.';
  EDIError023 = 'Could not get functional group at index [%s], There were no
functional groups to get.';
  EDIError024 = 'Delimiters have not been assigned to interchange.  Dissassemble
cancelled.';
  EDIError025 = 'Could not find interchange control header segment terminator.';
  EDIError026 = 'Could not find interchange control header.';
  EDIError027 = 'Could not find functional group header.';
  EDIError028 = 'Could not find functional group trailer segment terminator.';
  EDIError029 = 'Could not find functional group trailer.';
  EDIError030 = 'Could not find interchange control trailer segment terminator.';
  EDIError031 = 'Could not find interchange control trailer.';
  EDIError032 = 'Could not delete functional groups at index [%s].';
  EDIError033 = 'Could not delete functional group at index [%s].';
  EDIError034 = 'Delimiters have not been assigned to interchange.  Assemble
cancelled.';
  EDIError035 = 'Could not set transaction set at index [%s], Index too high.';
  EDIError036 = 'Could not set transaction set at index [%s], Index too low.';
  EDIError037 = 'Could not set transaction set at index [%s].';
  EDIError038 = 'Could not get transaction set at index [%s], Transaction Set does
not exist.';
  EDIError039 = 'Could not get transaction set at index [%s], Index too high.';
  EDIError040 = 'Could not get transaction set at index [%s], Index too low.';
  EDIError041 = 'Could not get transaction set at index [%s], There were no
Transaction Sets to get.';
  EDIError042 = 'Could not assign delimiters to functional group.  Dissassemble
cancelled.';
  EDIError043 = 'Could not find functional group header segment terminator.';
  EDIError044 = 'Could not find functional group header.';
  EDIError045 = 'Could not find transaction set header.';
  EDIError046 = 'Could not find transaction set trailer segment terminator.';
  EDIError047 = 'Could not find transaction set trailer.';
  EDIError048 = 'Could not find functional group trailer segment terminator.';
  EDIError049 = 'Could not find functional group trailer..';
  EDIError050 = 'Could not delete transaction sets at index [%s].';
  EDIError051 = 'Could not delete transaction set at index [%s].';
  EDIError052 = 'Could not assign delimiters to functional group.  Assemble
cancelled.';
  EDIError053 = 'Could not set segment at index [%s], Index too high.';
  EDIError054 = 'Could not set segment at index [%s], Index too low.';
  EDIError055 = 'Could not set segment at index [%s].';
  EDIError056 = 'Could not get segment at index [%s], Segment does not exist.';
  EDIError057 = 'Could not get segment at index [%s], Index too high.';
  EDIError058 = 'Could not get segment at index [%s], Index too low.';
```

```
  EDIError059 = 'Could not get segment at index [%s], There were no segments to
get.';
  EDIError060 = 'Could not assign delimiters to transaction set.  Dissassemble
cancelled.';
  EDIError061 = 'Could not delete segment at index [%s].';
  EDIError062 = 'Could not delete segment at index [%s].';
  EDIError063 = 'Could not assign delimiters to transaction set.  Assemble
cancelled.';
  EDIError064 = 'Could not set element at index [%s], Index too high.';
  EDIError065 = 'Could not set element at index [%s], Index too low.';
  EDIError066 = 'Could not set element at index [%s].';
  EDIError067 = 'Could not get element at index [%s], Element does not exist.';
  EDIError068 = 'Could not get element at index [%s], Index too high.';
  EDIError069 = 'Could not get element at index [%s], Index too low.';
  EDIError070 = 'Could not get element at index [%s], There were no elements to
get.';
  EDIError071 = 'Could not assign delimiters to segment.  Dissassemble
cancelled.';
  EDIError072 = 'Could not delete element at index [%s].';
  EDIError073 = 'Could not delete element at index [%s].';
  EDIError074 = 'Could not assign delimiters to segment.  Assemble cancelled.';

//  ExprEval
resourcestring
  RsExprEvalRParenExpected = 'Parse error: ''('' expected';
  RsExprEvalFactorExpected = 'Parse error: Factor expected';
  RsExprEvalUnknownSymbol = 'Parse error: Unknown symbol: ''%s''';

  RsExprEvalFirstArg = 'Parse error: ''('' and function''s first parameter
expected';
  RsExprEvalNextArg = 'Parse error: '','' and another parameter expected';
  RsExprEvalEndArgs = 'Parse error: ''('' to close function''s parameters
expected';

  RsExprEvalExprNotFound = 'Expression compiler error: Expression ''%s'' not
found';
  RsExprEvalExprPtrNotFound = 'Expression compiler error: Expression pointer not
found';


//  StrHashMap
resourcestring
  RsStringHashMapMustBeEmpty = 'HashList: must be empty to set size to zero';
  RsStringHashMapDuplicate = 'Duplicate hash list entry: %s';
  RsStringHashMapInvalidNode = 'Tried to remove invalid node: %s';


//  FileUtils
resourcestring

  // Path manipulation

  RsPathInvalidDrive = '%s is not a valid drive';

  // Files and directories

  RsFileUtilsAttrUnavailable = 'Unable to retrieve attributes of %s';

  RsCannotCreateDir = 'Unable to create directory';

  // T FileVersionInfo

  RsFileUtilsNoVersionInfo = 'File contains no version information';
  RsFileUtilsLanguageIndex = 'Illegal language index';

  // Strings returned from OSIdentTOString()

  RsVosUnknown      = 'Unknown';
  RsVosDos          = 'MS-DOS';
  RsVosOS216        = '16-bit OS/2';
  RsVosOS232        = '32-bit OS/2';
```

```
  RsVosNT            = 'Windows NT';
  RsVosWindows16     = '16-bit Windows';
  RsVosPM16          = '16-bit PM';
  RsVosPM32          = '32-bit PM';
  RsVosWindows32     = '32-bit Windows';
  RsVosDosWindows16  = '16-bit Windows, running on MS-DOS';
  RsVosDosWindows32  = 'Win32 API, running on MS-DOS';
  RsVosOS216PM16     = '16-bit PM, running on 16-bit OS/2';
  RsVosOS232PM32     = '32-bit PM, running on 32-bit OS/2';
  RsVosNTWindows32   = 'Win32 API, running on Windows/NT';
  RsVosDesignedFor   = 'Designed for ';

  // Strings returned from OSFileTypeToString()

  RsVftUnknown          = 'Unknown';
  RsVftApp              = 'Application';
  RsVftDll              = 'Library';
  RsVftDrv              = 'Driver';
  RsVftFont             = 'Font';
  RsVftVxd              = 'Virtual device';
  RsVftStaticLib        = 'Static-link library';
  RsVft2DrvPRINTER      = 'Printer';
  RsVft2DrvKEYBOARD     = 'Keyboard';
  RsVft2DrvLANGUAGE     = 'Language';
  RsVft2DrvDISPLAY      = 'Display';
  RsVft2DrvMOUSE        = 'Mouse';
  RsVft2DrvNETWORK      = 'Network';
  RsVft2DrvSYSTEM       = 'System';
  RsVft2DrvINSTALLABLE  = 'Installable';
  RsVft2DrvSOUND        = 'Sound';
  RsVft2DrvCOMM         = 'Communications';
  RsVft2FontRASTER      = 'Raster';
  RsVft2FontVECTOR      = 'Vector';
  RsVft2FontTRUETYPE    = 'TrueType';

  // T FileStream

  RsFileStreamCreate         = 'Unable to create temporary file stream';

  // T FileMapping

  RsCreateFileMapping        = 'Failed to create FileMapping';
  RsCreateFileMappingView    = 'Failed to create FileMappingView';
  RsLoadFromStreamSize       = 'Not enough space in View in procedure
LoadFromStream';
  RsFileMappingInvalidHandle = 'Invalid file handle';
  RsViewNeedsMapping         = 'FileMap argument of T FileMappingView constructor
cannot be nil';
  RsFailedToObtainSize       = 'Failed to obtain size of file';

  // GetDriveTypeStr()

  RsUnknownDrive   = 'Unknown drive type';
  RsRemovableDrive = 'Removable Drive';
  RsHardDisk       = 'Hard Disk';
  RsRemoteDrive    = 'Remote Drive';
  RsCDRomDrive     = 'CD-ROM';
  RsRamDisk        = 'RAM-Disk';

  // GetFileAttributeList()

  RsAttrDirectory  = 'Directory';
  RsAttrReadOnly   = 'ReadOnly';
  RsAttrSystemFile = 'SystemFile';
  RsAttrVolumeID   = 'Volume ID';
  RsAttrArchive    = 'Archive';
  RsAttrAnyFile    = 'AnyFile';
  RsAttrHidden     = 'Hidden';

  // GetFileAttributeListEx()
```

```
  RsAttrNormal        = 'Normal';
  RsAttrTemporary     = 'Temporary';
  RsAttrCompressed    = 'Compressed';
  RsAttrOffline       = 'Offline';
  RsAttrEncrypted     = 'Encrypted';
  RsAttrReparsePoint  = 'Reparse Point';
  RsAttrSparseFile    = 'Sparse';

  // T FileMapping.Create

  RsFileMappingOpenFile = 'Unable to open the file';

  // T MappedTextReader

  RsFileIndexOutOfRange = 'Index of out range';

  // FileGetTypeName()

  RsDefaultFileTypeName = ' File';

//  Graphics,  GraphUtils
resourcestring
  RsAssertUnpairedEndUpdate  = 'Unpaired BeginUpdate EndUpdate';
  RsCreateCompatibleDc       = 'Could not create compatible DC';
  RsDestinationBitmapEmpty   = 'Destination bitmap is empty';
  RsDibHandleAllocation      = 'Could not allocate handle for DIB';
  RsMapSizeFmt               = 'Could not set size on class "%s"';
  RsSelectObjectInDc         = 'Could not select object in DC';
  RsSourceBitmapEmpty        = 'Source bitmap is empty';
  RsSourceBitmapInvalid      = 'Source bitmap is invalid';
  RsNoBitmapForRegion        = 'No bitmap for region';
  RsNoDeviceContextForWindow = 'Cannot get device context of the window';
  RsInvalidRegion            = 'Invalid Region defined for RegionInfo';
  RsRegionDataOutOfBound     = 'Out of bound index on RegionData';
  RsRegionCouldNotCreated    = 'Region could not be created';
  RsInvalidHandleForRegion   = 'Invalid handle for region';
  RsInvalidRegionInfo        = 'Invalid RegionInfo';

  RsBitmapExtension          = '.bmp';
  RsJpegExtension            = '.jpg';

// Mapi
resourcestring
  RsMapiError         = 'MAPI Error: (%d) "%s"';
  RsMapiMissingExport = 'Function "%s" is not exported by client';
  RsMapiInvalidIndex  = 'Index is out of range';
  RsMapiMailNoClient  = 'No Simple MAPI client installed, cannot send the
message';

  RsMapiErrUSER_ABORT                = 'User abort';
  RsMapiErrFAILURE                   = 'General MAPI failure';
  RsMapiErrLOGIN_FAILURE             = 'MAPI login failure';
  RsMapiErrDISK_FULL                 = 'Disk full';
  RsMapiErrINSUFFICIENT_MEMORY       = 'Insufficient memory';
  RsMapiErrACCESS_DENIED             = 'Access denied';
  RsMapiErrTOO_MANY_SESSIONS         = 'Too many sessions';
  RsMapiErrTOO_MANY_FILES            = 'Too many files were specified';
  RsMapiErrTOO_MANY_RECIPIENTS       = 'Too many recipients were specified';
  RsMapiErrATTACHMENT_NOT_FOUND      = 'A specified attachment was not found';
  RsMapiErrATTACHMENT_OPEN_FAILURE   = 'Attachment open failure';
  RsMapiErrATTACHMENT_WRITE_FAILURE  = 'Attachment write failure';
  RsMapiErrUNKNOWN_RECIPIENT         = 'Unknown recipient';
  RsMapiErrBAD_RECIPTYPE             = 'Bad recipient type';
  RsMapiErrNO_MESSAGES               = 'No messages';
  RsMapiErrINVALID_MESSAGE           = 'Invalid message';
  RsMapiErrTEXT_TOO_LARGE            = 'Text too large';
  RsMapiErrINVALID_SESSION           = 'Invalid session';
  RsMapiErrTYPE_NOT_SUPPORTED        = 'Type not supported';
  RsMapiErrAMBIGUOUS_RECIPIENT       = 'A recipient was specified ambiguously';
  RsMapiErrMESSAGE_IN_USE            = 'Message in use';
  RsMapiErrNETWORK_FAILURE           = 'Network failure';
```

```
  RsMapiErrINVALID_EDITFIELDS       = 'Invalid edit fields';
  RsMapiErrINVALID_RECIPS           = 'Invalid recipients';
  RsMapiErrNOT_SUPPORTED            = 'Not supported';

  RsMapiMailORIG         = 'From';
  RsMapiMailTO           = 'To';
  RsMapiMailCC           = 'Cc';
  RsMapiMailBCC          = 'Bcc';
  RsMapiMailSubject      = 'Subject';
  RsMapiMailBody         = 'Body';

//  Math
resourcestring
  RsMathDomainError     = 'Domain check failure in  Math';
  RsEmptyArray          = 'Empty array is not allowed as input parameter';
  RsNonPositiveArray    = 'Input array contains non-positive or zero values';
  RsUnexpectedDataType  = 'Unexpected data type';
  RsUnexpectedValue     = 'Unexpected data value';
  RsRangeError          = 'Cannot merge range';
  RsInvalidRational     = 'Invalid rational number';
  RsDivByZero           = 'Division by zero';
  RsRationalDivByZero   = 'Rational division by zero';
  RsNoNaN               = 'NaN expected';
  RsNaNTagError         = 'NaN Tag value %d out of range';
  RsNaNSignal           = 'NaN signaling %d';


//  Midi
resourcestring
  RsOctaveC      = 'C';
  RsOctaveCSharp = 'C#';
  RsOctaveD      = 'D';
  RsOctaveDSharp = 'D#';
  RsOctaveE      = 'E';
  RsOctaveF      = 'F';
  RsOctaveFSharp = 'F#';
  RsOctaveG      = 'G';
  RsOctaveGSharp = 'G#';
  RsOctaveA      = 'A';
  RsOctaveASharp = 'A#';
  RsOctaveB      = 'B';

  RsMidiInUnknownError    = 'Unknown MIDI-In error No. %d';
  RsMidiOutUnknownError   = 'Unknown MIDI-Out error No. %d';
  RsInvalidMidiChannelNum = 'Invalid MIDI channel number (%d)';

//  Miscel
resourcestring
  // CreateProcAsUser
  RsCreateProcOSVersionError          = 'Unable to determine OS version';
  RsCreateProcNTRequiredError         = 'Windows NT required';
  RsCreateProcBuild1057Error          = 'NT version 3.51 build 1057 or later
required';

  RsCreateProcPrivilegeMissing        = 'This account does not have the privilege
"%s" (%s)';
  RsCreateProcLogonUserError          = 'LogonUser failed';
  RsCreateProcAccessDenied            = 'Access denied';
  RsCreateProcLogonFailed             = 'Unable to logon';
  RsCreateProcSetStationSecurityError = 'Cannot set WindowStation "%s" security.';
  RsCreateProcSetDesktopSecurityError = 'Cannot set Desktop "%s" security.';
  RsCreateProcPrivilegesMissing       = 'This account does not have one (or more)
of ' +
     'the following privileges: ' + '"%s"(%s)' + #13 + '"%s"(%s)' + #13;
  RsCreateProcCommandNotFound         = 'Command or filename not found: "%s"';
  RsCreateProcFailed                  = 'CreateProcessAsUser failed';

//  Multimedia
resourcestring

  // Multimedia timer
```

```
  RsMmTimerGetCaps     = 'Error retrieving multimedia timer device capabilities';
  RsMmTimerBeginPeriod = 'The supplied timer period value is out of range';
  RsMmSetEvent         = 'Error setting multimedia event timer';
  RsMmInconsistentId   = 'Multimedia timer callback was called with inconsistent
Id';
  RsMmTimerActive      = 'This operation cannot be performed while the timer is
active';

  // Audio Mixer

  RsMmMixerSource      = 'Source';
  RsMmMixerDestination = 'Destination';
  RsMmMixerUndefined   = 'Undefined';
  RsMmMixerDigital     = 'Digital';
  RsMmMixerLine        = 'Line';
  RsMmMixerMonitor     = 'Monitor';
  RsMmMixerSpeakers    = 'Speakers';
  RsMmMixerHeadphones  = 'Headphones';
  RsMmMixerTelephone   = 'Telephone';
  RsMmMixerWaveIn      = 'Waveform-audio input';
  RsMmMixerVoiceIn     = 'Voice input';
  RsMmMixerMicrophone  = 'Microphone';
  RsMmMixerSynthesizer = 'Synthesizer';
  RsMmMixerCompactDisc = 'Compact disc';
  RsMmMixerPcSpeaker   = 'PC speaker';
  RsMmMixerWaveOut     = 'Waveform-audio output';
  RsMmMixerAuxiliary   = 'Auxiliary audio line';
  RsMmMixerAnalog      = 'Analog';
  RsMmMixerNoDevices   = 'No mixer device found';
  RsMmMixerCtlNotFound = 'Line control (%s, %.8x) not found';

  // E MciError

  RsMmUnknownError     = 'Unknown MCI error No. %d';
  RsMmMciErrorPrefix   = 'MCI-Error: ';

  // CD audio routines

  RsMmNoCdAudio        = 'Cannot open CDAUDIO-Device';
  RsMmCdTrackNo        = 'Track: %.2u';
  RsMMCdTimeFormat     = '%2u:%.2u';
  RsMMTrackAudio       = 'Audio';
  RsMMTrackOther       = 'Other';

//  NTFS
resourcestring
  RsInvalidArgument = '%s: Invalid argument <%s>';
  RsNtfsUnableToDeleteSymbolicLink = 'Unable to delete temporary symbolic link';

//  PeImage
// T PeImage

resourcestring
  RsPeCantOpen                = 'Cannot open file "%s"';
  RsPeNotPE                   = 'This is not a PE format';
  RsPeNotResDir               = 'Not a resource directory';
  RsPeNotAvailableForAttached = 'Feature is not available for attached images';
  RsPeSectionNotFound         = 'Section "%s" not found';

  // PE directory names

  RsPeImg_00 = 'Exports';
  RsPeImg_01 = 'Imports';
  RsPeImg_02 = 'Resources';
  RsPeImg_03 = 'Exceptions';
  RsPeImg_04 = 'Security';
  RsPeImg_05 = 'Base Relocations';
  RsPeImg_06 = 'Debug';
  RsPeImg_07 = 'Description';
  RsPeImg_08 = 'Machine Value';
```

```
RsPeImg_09 = 'TLS';
RsPeImg_10 = 'Load configuration';
RsPeImg_11 = 'Bound Import';
RsPeImg_12 = 'IAT';
RsPeImg_13 = 'Delay load import';
RsPeImg_14 = 'COM run-time';

// NT Header names

RsPeSignature                = 'Signature';
RsPeMachine                  = 'Machine';
RsPeNumberOfSections         = 'Number of Sections';
RsPeTimeDateStamp            = 'Time Date Stamp';
RsPePointerToSymbolTable     = 'Symbols Pointer';
RsPeNumberOfSymbols          = 'Number of Symbols';
RsPeSizeOfOptionalHeader     = 'Size of Optional Header';
RsPeCharacteristics          = 'Characteristics';
RsPeMagic                    = 'Magic';
RsPeLinkerVersion            = 'Linker Version';
RsPeSizeOfCode               = 'Size of Code';
RsPeSizeOfInitializedData    = 'Size of Initialized Data';
RsPeSizeOfUninitializedData  = 'Size of Uninitialized Data';
RsPeAddressOfEntryPoint      = 'Address of Entry Point';
RsPeBaseOfCode               = 'Base of Code';
RsPeBaseOfData               = 'Base of Data';
RsPeImageBase                = 'Image Base';
RsPeSectionAlignment         = 'Section Alignment';
RsPeFileAlignment            = 'File Alignment';
RsPeOperatingSystemVersion   = 'Operating System Version';
RsPeImageVersion             = 'Image Version';
RsPeSubsystemVersion         = 'Subsystem Version';
RsPeWin32VersionValue        = 'Win32 Version';
RsPeSizeOfImage              = 'Size of Image';
RsPeSizeOfHeaders            = 'Size of Headers';
RsPeCheckSum                 = 'CheckSum';
RsPeSubsystem                = 'Subsystem';
RsPeDllCharacteristics       = 'Dll Characteristics';
RsPeSizeOfStackReserve       = 'Size of Stack Reserve';
RsPeSizeOfStackCommit        = 'Size of Stack Commit';
RsPeSizeOfHeapReserve        = 'Size of Heap Reserve';
RsPeSizeOfHeapCommit         = 'Size of Heap Commit';
RsPeLoaderFlags              = 'Loader Flags';
RsPeNumberOfRvaAndSizes      = 'Number of RVA';

// Load config names

RsPeVersion                          = 'Version';
RsPeGlobalFlagsClear                 = 'GlobalFlagsClear';
RsPeGlobalFlagsSet                   = 'GlobalFlagsSet';
RsPeCriticalSectionDefaultTimeout    = 'CriticalSectionDefaultTimeout';
RsPeDeCommitFreeBlockThreshold       = 'DeCommitFreeBlockThreshold';
RsPeDeCommitTotalFreeThreshold       = 'DeCommitTotalFreeThreshold';
RsPeLockPrefixTable                  = 'LockPrefixTable';
RsPeMaximumAllocationSize            = 'MaximumAllocationSize';
RsPeVirtualMemoryThreshold           = 'VirtualMemoryThreshold';
RsPeProcessHeapFlags                 = 'ProcessHeapFlags';
RsPeProcessAffinityMask              = 'ProcessAffinityMask';
RsPeCSDVersion                       = 'CSDVersion';
RsPeReserved                         = 'Reserved';
RsPeEditList                         = 'EditList';

// Machine names

RsPeMACHINE_UNKNOWN = 'Unknown';
RsPeMACHINE_I386    = 'Intel 386';
RsPeMACHINE_R3000   = 'MIPS little-endian R3000';
RsPeMACHINE_R4000   = 'MIPS little-endian R4000';
RsPeMACHINE_R10000  = 'MIPS little-endian R10000';
RsPeMACHINE_ALPHA   = 'Alpha_AXP';
RsPeMACHINE_POWERPC = 'IBM PowerPC Little-Endian';
```

```
// Subsystem names

RsPeSUBSYSTEM_UNKNOWN      = 'Unknown';
RsPeSUBSYSTEM_NATIVE       = 'Native';
RsPeSUBSYSTEM_WINDOWS_GUI  = 'GUI';

RsPeSUBSYSTEM_WINDOWS_CUI  = 'Console';
RsPeSUBSYSTEM_OS2_CUI      = 'OS/2';
RsPeSUBSYSTEM_POSIX_CUI    = 'Posix';
RsPeSUBSYSTEM_RESERVED8    = 'Reserved 8';

// Debug symbol type names

RsPeDEBUG_UNKNOWN        = 'UNKNOWN';
RsPeDEBUG_COFF           = 'COFF';
RsPeDEBUG_CODEVIEW       = 'CODEVIEW';
RsPeDEBUG_FPO            = 'FPO';
RsPeDEBUG_MISC           = 'MISC';
RsPeDEBUG_EXCEPTION      = 'EXCEPTION';
RsPeDEBUG_FIXUP          = 'FIXUP';
RsPeDEBUG_OMAP_TO_SRC    = 'OMAP_TO_SRC';
RsPeDEBUG_OMAP_FROM_SRC  = 'OMAP_FROM_SRC';
RsPeDEBUG_BORLAND        = 'BORLAND';

// T PePackageInfo.PackageModuleTypeToString

RsPePkgExecutable = 'Executable';
RsPePkgPackage    = 'Package';
RsPePkgLibrary    = 'Library';

// T PePackageInfo.PackageOptionsToString

RsPePkgNeverBuild     = 'NeverBuild';
RsPePkgDesignOnly     = 'DesignOnly';
RsPePkgRunOnly        = 'RunOnly';
RsPePkgIgnoreDupUnits = 'IgnoreDupUnits';

// T PePackageInfo.ProducerToString

RsPePkgV3Produced        = 'Delphi 3 or C++ Builder 3';
RsPePkgProducerUndefined = 'Undefined';
RsPePkgBCB4Produced      = 'C++ Builder 4 or later';
RsPePkgDelphi4Produced   = 'Delphi 4 or later';

// T PePackageInfo.UnitInfoFlagsToString

RsPePkgMain     = 'Main';
RsPePkgWeak     = 'Weak';
RsPePkgOrgWeak  = 'OrgWeak';
RsPePkgImplicit = 'Implicit';

//  Print
resourcestring
  RsInvalidPrinter        = 'Invalid printer';
  RsNAStartDocument       = 'Unable to "Start document"';
  RsNASendData            = 'Unable to send data to printer';
  RsNAStartPage           = 'Unable to "Start page"';
  RsNAEndPage             = 'Unable to "End page"';
  RsNAEndDocument         = 'Unable to "End document"';
  RsNATransmission        = 'Not all chars have been sent correctly to printer';
  RsDeviceMode            = 'Error retrieving DeviceMode';
  RsUpdatingPrinter       = 'Error updating printer driver';
  RsIndexOutOfRange       = 'Index out of range setting bin';
  RsRetrievingSource      = 'Error retrieving Bin Source Info';
  RsRetrievingPaperSource = 'Error retrieving Paper Source Info';
  RsIndexOutOfRangePaper  = 'Index out of range setting paper';

  // Paper Styles (PS)
  RsPSLetter      = 'Letter 8 1/2 x 11 in';
  RsPSLetterSmall = 'Letter Small 8 1/2 x 11 in';
  RsPSTabloid     = 'Tabloid 11 x 17 in';
```

```
  RsPSLedger       = 'Ledger 17 x 11 in';
  RsPSLegal        = 'Legal 8 1/2 x 14 in';
  RsPSStatement    = 'Statement 5 1/2 x 8 1/2 in';
  RsPSExecutive    = 'Executive 7 1/2 x 10 in';
  RsPSA3           = 'A3 297 x 420 mm';
  RsPSA4           = 'A4 210 x 297 mm';
  RsPSA4Small      = 'A4 Small 210 x 297 mm';
  RsPSA5           = 'A5 148 x 210 mm';
  RsPSB4           = 'B4 250 x 354';
  RsPSB5           = 'B5 182 x 257 mm';
  RsPSFolio        = 'Folio 8 1/2 x 13 in';
  RsPSQuarto       = 'Quarto 215 x 275 mm';
  RsPS10X14        = '10 x 14 in';
  RsPS11X17        = '11 x 17 in';
  RsPSNote         = 'Note 8 1/2 x 11 in';
  RsPSEnv9         = 'Envelope #9 3 7/8 x 8 7/8 in';
  RsPSEnv10        = 'Envelope #10 4 1/8 x 9 1/2 in';
  RsPSEnv11        = 'Envelope #11 4 1/2 x 10 3/8 in';
  RsPSEnv12        = 'Envelope #12 4 \276 x 11 in';
  RsPSEnv14        = 'Envelope #14 5 x 11 1/2 in';
  RsPSCSheet       = 'C size sheet';
  RsPSDSheet       = 'D size sheet';
  RsPSESheet       = 'E size sheet';
  RsPSUser         = 'User Defined Size';
  RsPSUnknown      = 'Unknown Paper Size';

  RsPrintIniPrinterName    = 'PrinterName';
  RsPrintIniPrinterPort    = 'PrinterPort';
  RsPrintIniOrientation    = 'Orientation';
  RsPrintIniPaperSize      = 'PaperSize';
  RsPrintIniPaperLength    = 'PaperLength';
  RsPrintIniPaperWidth     = 'PaperWidth';
  RsPrintIniScale          = 'Scale';
  RsPrintIniCopies         = 'Copies';
  RsPrintIniDefaultSource  = 'DefaultSource';
  RsPrintIniPrintQuality   = 'PrintQuality';
  RsPrintIniColor          = 'Color';
  RsPrintIniDuplex         = 'Duplex';
  RsPrintIniYResolution    = 'YResolution';
  RsPrintIniTTOption       = 'TTOption';

//  Registry
resourcestring
  RsUnableToOpenKeyRead  = 'Unable to open key "%s" for read';
  RsUnableToOpenKeyWrite = 'Unable to open key "%s" for write';
  RsUnableToAccessValue  = 'Unable to open key "%s" and access value "%s"';

//  RTTI
resourcestring
  RsRTTIValueOutOfRange =   'Value out of range (%s).';
  RsRTTIUnknownIdentifier = 'Unknown identifier ''%s''.';
  RsRTTIInvalidGUIDString = 'Invalid conversion from string to GUID (%s).';
  RsRTTIInvalidBaseType   = 'Invalid base type (%s is of type %s).';

  RsRTTIVar =               'var ';
  RsRTTIConst =             'const ';
  RsRTTIArrayOf =           'array of ';
  RsRTTIOut =               'out ';
  RsRTTIBits =              'bits';
  RsRTTIOrdinal =           'ordinal=';
  RsRTTITrue =              'True';
  RsRTTIFalse =             'False';
  RsRTTITypeError =         '???';
  RsRTTITypeInfoAt =        'Type info: %p';

  RsRTTIPropRead =          'read';
  RsRTTIPropWrite =         'write';
  RsRTTIPropStored =        'stored';

  RsRTTIField =             'field';
  RsRTTIStaticMethod =      'static method';
```

```
  RsRTTIVirtualMethod =      'virtual method';

  RsRTTIIndex =              'index';
  RsRTTIDefault =            'default';

  RsRTTIName =               'Name: ';
  RsRTTIType =               'Type: ';
  RsRTTIFlags =              'Flags: ';
  RsRTTIGUID =               'GUID: ';
  RsRTTITypeKind =           'Type kind: ';

  RsRTTIOrdinalType =        'Ordinal type: ';
  RsRTTIMinValue =           'Min value: ';
  RsRTTIMaxValue =           'Max value: ';
  RsRTTINameList =           'Names: ';
  RsRTTIClassName =          'Class name: ';
  RsRTTIParent =             'Parent: ';
  RsRTTIPropCount =          'Property count: ';
  RsRTTIUnitName =           'Unit name: ';
  RsRTTIBasedOn =            'Based on: ';
  RsRTTIFloatType =          'Float type: ';
  RsRTTIMethodKind =         'Method kind: ';
  RsRTTIParamCount =         'Parameter count: ';
  RsRTTIReturnType =         'Return type: ';
  RsRTTIMaxLen =             'Max length: ';
  RsRTTIElSize =             'Element size: ';
  RsRTTIElType =             'Element type: ';
  RsRTTIElNeedCleanup =      'Elements need clean up: ';
  RsRTTIVarType =            'Variant type: ';

//  Schedule
resourcestring
  RsScheduleInvalidTime      = 'Invalid time specification';
  RsScheduleEndBeforeStart   = 'End time can not be before start time';
  RsScheduleIntervalZero     = 'Interval should be larger than 0';
  RsScheduleNoDaySpecified   = 'At least one day of the week should be specified';
  RsScheduleIndexValueSup    = 'Property IndexValue not supported for current
IndexKind';
  RsScheduleIndexValueZero   = 'IndexValue can not be 0';
  RsScheduleDayNotSupported = 'Property Day not supported for current IndexKind';
  RsScheduleDayInRange       = 'Day values should fall in the range 1 .. 31';
  RsScheduleMonthInRange     = 'Month values should fall in the range 1 .. 12';

//  Strings
resourcestring
  RsInvalidEmptyStringItem  = 'String list passed to StringsToMultiSz cannot
contain empty strings.';
  RsNumericConstantTooLarge = 'Numeric constant too large.';

//  Synch
resourcestring
  RsSynchAttachWin32Handle     = 'Invalid handle to T Win32HandleObject.Attach';
  RsSynchDuplicateWin32Handle = 'Invalid handle to T Win32HandleObject.Duplicate';
  RsSynchInitCriticalSection   = 'Failed to initalize critical section';
  RsSynchAttachDispatcher      = 'Invalid handle to T DispatcherObject.Attach';
  RsSynchCreateEvent           = 'Failed to create event';
  RsSynchOpenEvent             = 'Failed to open event';
  RsSynchCreateWaitableTimer   = 'Failed to create waitable timer';
  RsSynchOpenWaitableTimer     = 'Failed to open waitable timer';
  RsSynchCreateSemaphore       = 'Failed to create semaphore';
  RsSynchOpenSemaphore         = 'Failed to open semaphore';
  RsSynchCreateMutex           = 'Failed to create mutex';
  RsSynchOpenMutex             = 'Failed to open mutex';
  RsMetSectInvalidParameter    = 'An invalid parameter was passed to the
constructor.';
  RsMetSectInitialize          = 'Failed to initialize the metered section.';
  RsMetSectNameEmpty           = 'Name cannot be empty when using the Open
constructor.';

//  SysInfo
resourcestring
```

```
  RsSystemProcess = 'System Process';
  RsSystemIdleProcess = 'System Idle Process';

  RsIntelCacheDescr01 = 'Instruction TLB, 4Kb pages, 4-way set associative, 32
entries';
  RsIntelCacheDescr02 = 'Instruction TLB, 4Mb pages, fully associative, 2
entries';
  RsIntelCacheDescr03 = 'Data TLB, 4Kb pages, 4-way set associative, 64 entries';
  RsIntelCacheDescr04 = 'Data TLB, 4Mb pages, 4-way set associative, 8 entries';
  RsIntelCacheDescr06 = '8KB instruction cache, 4-way set associative, 32 byte
line size';
  RsIntelCacheDescr08 = '16KB instruction cache, 4-way set associative, 32 byte
line size';
  RsIntelCacheDescr0A = '8KB data cache 2-way set associative, 32 byte line size';
  RsIntelCacheDescr0C = '16KB data cache, 4-way set associative, 32 byte line
size';
  RsIntelCacheDescr40 = 'No L2 cache';
  RsIntelCacheDescr41 = 'Unified cache, 32 byte cache line, 4-way set associative,
128Kb';
  RsIntelCacheDescr42 = 'Unified cache, 32 byte cache line, 4-way set associative,
256Kb';
  RsIntelCacheDescr43 = 'Unified cache, 32 byte cache line, 4-way set associative,
512Kb';
  RsIntelCacheDescr44 = 'Unified cache, 32 byte cache line, 4-way set associative,
1Mb';
  RsIntelCacheDescr45 = 'Unified cache, 32 byte cache line, 4-way set associative,
2Mb';

resourcestring
  RsOSVersionWin95 = 'Windows 95';
  RsOSVersionWin95OSR2 = 'Windows 95 OSR2';
  RsOSVersionWin98 = 'Windows 98';
  RsOSVersionWin98SE = 'Windows 98 SE';
  RsOSVersionWinME = 'Windows ME';
  RsOSVersionWinNT3 = 'Windows NT 3.%u';
  RsOSVersionWinNT4 = 'Windows NT 4.%u';
  RsOSVersionWin2000 = 'Windows 2000';
  RsOSVersionWinXP = 'Windows XP';

resourcestring
  RsProductTypeWorkStation = 'Workstation';
  RsProductTypeServer = 'Server';
  RsProductTypeAdvancedServer = 'Advanced Server';
  RsProductTypePersonal = 'Home Edition';
  RsProductTypeProfessional = 'Professional';
  RsProductTypeDatacenterServer = 'Datacenter Server';

//  SysUtils
resourcestring
  RsCannotWriteRefStream = 'Can not write to a read-only memory stream';
  RsStringToBoolean = 'Unable to convert the string "%s" to a boolean';

//  TD32
resourcestring
  RsHasNotTD32Info       = 'File [%s] has not TD32 debug information!';

//  Unicode
resourcestring
  RsUREBaseString = 'Error in regular expression: %s' + #13;
  RsUREUnexpectedEOS       = 'Unexpected end of pattern.';
  RsURECharacterClassOpen  = 'Character class not closed, ''}'' is missing.';
  RsUREUnbalancedGroup     = 'Unbalanced group expression, ''}'' is missing.';
  RsUREInvalidCharProperty = 'A character property is invalid';
  RsUREInvalidRepeatRange  = 'Invalid repetition range.';
  RsURERepeatRangeOpen     = 'Repetition range not closed, ''}'' is missing.';
  RsUREExpressionEmpty     = 'Expression is empty.';

implementation

end.
```

LAMPIRAN G : LISTING UNIT ddkint.pas

File yang berisi Device Driver dan proses untuk menjalankan Driver.

```
interface
uses windows,winsvc;
function CTL_CODE(const DeviceType,Func,Method,Access:Cardinal):cardinal;

const
 FILE_DEVICE_BEEP                = $00000001;
 FILE_DEVICE_CD_ROM              = $00000002;
 FILE_DEVICE_CD_ROM_FILE_SYSTEM = $00000003;
 FILE_DEVICE_CONTROLLER          = $00000004;
 FILE_DEVICE_DATALINK            = $00000005;
 FILE_DEVICE_DFS                 = $00000006;
 FILE_DEVICE_DISK                = $00000007;
 FILE_DEVICE_DISK_FILE_SYSTEM    = $00000008;
 FILE_DEVICE_FILE_SYSTEM         = $00000009;
 FILE_DEVICE_INPORT_PORT         = $0000000a;
 FILE_DEVICE_KEYBOARD            = $0000000b;
 FILE_DEVICE_MAILSLOT            = $0000000c;
 FILE_DEVICE_MIDI_IN             = $0000000d;
 FILE_DEVICE_MIDI_OUT            = $0000000e;
 FILE_DEVICE_MOUSE               = $0000000f;
 FILE_DEVICE_MULTI_UNC_PROVIDER = $00000010;
 FILE_DEVICE_NAMED_PIPE          = $00000011;
 FILE_DEVICE_NETWORK             = $00000012;
 FILE_DEVICE_NETWORK_BROWSER     = $00000013;
 FILE_DEVICE_NETWORK_FILE_SYSTEM= $00000014;
 FILE_DEVICE_NULL                = $00000015;
 FILE_DEVICE_PARALLEL_PORT       = $00000016;
 FILE_DEVICE_PHYSICAL_NETCARD    = $00000017;
 FILE_DEVICE_PRINTER             = $00000018;
 FILE_DEVICE_SCANNER             = $00000019;
 FILE_DEVICE_SERIAL_MOUSE_PORT   = $0000001a;
 FILE_DEVICE_SERIAL_PORT         = $0000001b;
 FILE_DEVICE_SCREEN              = $0000001c;
 FILE_DEVICE_SOUND               = $0000001d;
 FILE_DEVICE_STREAMS             = $0000001e;
 FILE_DEVICE_TAPE                = $0000001f;
 FILE_DEVICE_TAPE_FILE_SYSTEM    = $00000020;
 FILE_DEVICE_TRANSPORT           = $00000021;
 FILE_DEVICE_UNKNOWN             = $00000022;
 FILE_DEVICE_VIDEO               = $00000023;
 FILE_DEVICE_VIRTUAL_DISK        = $00000024;
 FILE_DEVICE_WAVE_IN             = $00000025;
 FILE_DEVICE_WAVE_OUT            = $00000026;
 FILE_DEVICE_8042_PORT           = $00000027;
 FILE_DEVICE_NETWORK_REDIRECTOR = $00000028;
 FILE_DEVICE_BATTERY             = $00000029;
 FILE_DEVICE_BUS_EXTENDER        = $0000002a;
 FILE_DEVICE_MODEM               = $0000002b;
 FILE_DEVICE_VDM                 = $0000002c;
 FILE_DEVICE_MASS_STORAGE        = $0000002d;
 FILE_DEVICE_SMB                 = $0000002e;
 FILE_DEVICE_KS                  = $0000002f;
 FILE_DEVICE_CHANGER             = $00000030;
 FILE_DEVICE_SMARTCARD           = $00000031;
 FILE_DEVICE_ACPI                = $00000032;
 FILE_DEVICE_DVD                 = $00000033;
 FILE_DEVICE_FULLSCREEN_VIDEO    = $00000034;
 FILE_DEVICE_DFS_FILE_SYSTEM     = $00000035;
 FILE_DEVICE_DFS_VOLUME          = $00000036;
 FILE_DEVICE_SERENUM             = $00000037;
 FILE_DEVICE_TERMSRV             = $00000038;
 FILE_DEVICE_KSEC                = $00000039;
```

```
 FILE_DEVICE_KRNLDRVR           = $80ff;

 METHOD_BUFFERED    =             0;
 METHOD_IN_DIRECT   =             1;
 METHOD_OUT_DIRECT  =             2;
 METHOD_NEITHER     =             3;

 FILE_ANY_ACCESS      =           0;
 FILE_SPECIAL_ACCESS  =   (FILE_ANY_ACCESS);
 FILE_READ_ACCESS     =      ( $0001 );    // file & pipe
 FILE_WRITE_ACCESS    =      ( $0002 );    // file & pipe

function driverstart(const name:pchar):integer;
function driverstop(const name:pchar):integer;

// Hanya Untuk Cek Validitas
function driverinstall(const path,name:pchar):integer;
function driverremove(const name:pchar):integer;


// Pesan
function messagestring(const error:integer):string;

implementation

function CTL_CODE(const DeviceType,Func,Method,Access:Cardinal):cardinal;
begin
 Result := DeviceType shl 16 or Access shl 14 or Func shl 2 or Method;
end;


function driverinstall(const path,name:pchar):integer;
var hService: SC_HANDLE;
    hSCMan          : SC_HANDLE;
begin

  Result := 0;

  hSCMan := OpenSCManager(nil, nil, SC_MANAGER_ALL_ACCESS);
  if hSCMan = 0 then  begin
   result := getlasterror;
   exit;
  end;

  hService := 0;

  hService := CreateService(hSCMan, name,name,
              SERVICE_ALL_ACCESS, SERVICE_KERNEL_DRIVER, SERVICE_DEMAND_START,
              SERVICE_ERROR_NORMAL, path,
              nil, nil, nil, nil, nil);
  if (hService = 0) then begin
    result := getlasterror;
    CloseServiceHandle(hSCMan);
    hSCMan := 0;
    exit;
  end
  else
    CloseServiceHandle(hService);
  CloseServiceHandle(hSCMan);
end;

function driverstart(const name:pchar):integer;
var
  hService: SC_HANDLE;
  hSCMan          : SC_HANDLE;
  args:pchar;
begin

  Result := 0;

  hSCMan := OpenSCManager(nil, nil, SC_MANAGER_CONNECT);
```

```
  if hSCMan = 0 then  begin
   result := getlasterror;
   exit;
  end;

  hService := 0;

  // service handle
  hService := OpenService(hSCMan, name, SERVICE_START);
  if hService <> 0 then Begin
     // start the driver
    args := nil;
    if integer(StartService(hService, 0, args )) = 0 then
      result := getlasterror;
    CloseServiceHandle(hService);
  end
  else
    result := getlasterror;
  CloseServiceHandle(hSCMan);
end;

function driverstop(const name:pchar):integer;
Var
  serviceStatus: TServiceStatus;
  hService: SC_HANDLE;
  hSCMan          : SC_HANDLE;
begin

  Result := 0;

  hSCMan := OpenSCManager(nil, nil, SC_MANAGER_CONNECT);
  if hSCMan = 0 then  begin
   result := getlasterror;
   exit;
  end;

  hService := 0;

  hService := OpenService(hSCMan, Name, SERVICE_STOP);
  if hService <> 0 then Begin
     if integer(ControlService(hService, SERVICE_CONTROL_STOP, serviceStatus)) = 0
then
      result := getlasterror;
    CloseServiceHandle(hService);
  end
  else
    result := getlasterror;
  CloseServiceHandle(hSCMan);

end;

function driverremove(const name:pchar):integer;
Var
  serviceStatus: TServiceStatus;
  hService: SC_HANDLE;
  hSCMan          : SC_HANDLE;
begin

  Result := 0;

  hSCMan := OpenSCManager(nil, nil, SC_MANAGER_ALL_ACCESS);
  if hSCMan = 0 then  begin
   result := getlasterror;
   exit;
  end;

  hService := 0;


  hService := OpenService(hSCMan, Name, SERVICE_ALL_ACCESS);
  if hService <> 0 then Begin
```

```
    if integer( DeleteService(hService)) = 0 then
      result := getlasterror;
    CloseServiceHandle(hService);
  end
  else
    result := getlasterror;
  CloseServiceHandle(hSCMan);
end;

function messagestring(const error:integer):string;
var p:pchar;
begin
   getmem(p,200);
   formatmessage(FORMAT_MESSAGE_FROM_SYSTEM,nil,error,0,p,199,nil);
   Result := p;
   freemem(p,200);
end;

end.
```

LAMPIRAN H : LISTING UNIT JCLbase.pas

File ini adalah dasar dari program yang akan dirancang.

```
{$I jcl.inc}

{$WEAKPACKAGEUNIT ON}

interface

uses
  {$IFDEF MSWINDOWS}
  Windows,
  {$ENDIF MSWINDOWS}
  Classes, SysUtils;


const
  JclVersionMajor   = 1;
  JclVersionMinor   = 22;
  JclVersionRelease = 1;
  JclVersionBuild   = 965;
  JclVersion = (JclVersionMajor shl 24) or (JclVersionMinor shl 16) or
               (JclVersionRelease shl 15) or (JclVersionBuild shl 0);


{$IFDEF FPC}

type
  PResStringRec = ^string;

function SysErrorMessage(ErrNo: Integer): string;

{$IFDEF MSWINDOWS}
procedure RaiseLastWin32Error;

procedure QueryPerformanceCounter(var C: Int64);
function QueryPerformanceFrequency(var Frequency: Int64): Boolean;
{$ENDIF MSWINDOWS}

var
  Default8087CW: Word;
{$ENDIF FPC}


type
  EJclError = class (Exception)
  public
    constructor CreateResRec(ResStringRec: PResStringRec);
    constructor CreateResRecFmt(ResStringRec: PResStringRec; const Args: array of
const);
  end;


{$IFDEF MSWINDOWS}

type
  EJclWin32Error = class (EJclError)
  private
    FLastError: DWORD;
    FLastErrorMsg: string;
  public
    constructor Create(const Msg: string);
    constructor CreateFmt(const Msg: string; const Args: array of const);
    constructor CreateRes(Ident: Integer);
    constructor CreateResRec(ResStringRec: PResStringRec);
    property LastError: DWORD read FLastError;
```

```
    property LastErrorMsg: string read FLastErrorMsg;
  end;

{$ENDIF MSWINDOWS}


type
  {$IFDEF MATH_EXTENDED_PRECISION}
  Float = Extended;
  {$ENDIF MATH_EXTENDED_PRECISION}
  {$IFDEF MATH_DOUBLE_PRECISION}
  Float = Double;
  {$ENDIF MATH_DOUBLE_PRECISION}
  {$IFDEF MATH_SINGLE_PRECISION}
  Float = Single;
  {$ENDIF MATH_SINGLE_PRECISION}

  PFloat = ^Float;

{$IFDEF FPC}
type
  LongWord = Cardinal;
  TSysCharSet = set of Char;
{$ENDIF FPC}

type
  PPointer = ^Pointer;


procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);


type
  PLargeInteger = ^TLargeInteger;
  TLargeInteger = record
    case Integer of
    0: (
      LowPart: LongWord;
      HighPart: Longint);
    1: (
      QuadPart: Int64);
  end;

type
  PULargeInteger = ^TULargeInteger;
  TULargeInteger = record
    case Integer of
    0: (
      LowPart: LongWord;
      HighPart: LongWord);
    1: (
      QuadPart: Int64);
  end;


type
  TDynByteArray     = array of Byte;
  TDynShortintArray = array of Shortint;
  TDynSmallintArray = array of Smallint;
  TDynWordArray     = array of Word;
  TDynIntegerArray  = array of Integer;
  TDynLongintArray  = array of Longint;
  TDynCardinalArray = array of Cardinal;
  TDynInt64Array    = array of Int64;
  TDynExtendedArray = array of Extended;
  TDynDoubleArray   = array of Double;
  TDynSingleArray   = array of Single;
  TDynFloatArray    = array of Float;
  TDynPointerArray  = array of Pointer;
  TDynStringArray   = array of string;
```

```
{$IFNDEF COMPILER5_UP}
type
  TObjectList = class (TList)
  private
    FOwnsObjects: Boolean;
    function GetItems(Index: Integer): TObject;
    procedure SetItems(Index: Integer; const Value: TObject);
  public
    procedure Clear; override;
    constructor Create(AOwnsObjects: Boolean = False);
    property Items[Index: Integer]: TObject read GetItems write SetItems; default;
    property OwnsObjects: Boolean read FOwnsObjects write FOwnsObjects;
  end;
{$ENDIF COMPILER5_UP}


{$IFNDEF COMPILER6_UP}
procedure RaiseLastOSError;
{$ENDIF COMPILER6_UP}


{$IFDEF SUPPORTS_INTERFACE}
{$IFNDEF COMPILER6_UP}

type
  IInterface = IUnknown;

{$ENDIF COMPILER6_UP}
{$ENDIF SUPPORTS_INTERFACE}


{$IFDEF COMPILER4}

type
  TStringListCustomSortCompare = function(List: TStringList; Index1, Index2:
Integer): Integer;

procedure StringListCustomSort(StringList: TStringList; SortFunc:
TStringListCustomSortCompare);

{$ENDIF COMPILER4}

implementation

uses
  JclResources;


constructor EJclError.CreateResRec(ResStringRec: PResStringRec);
begin
  {$IFDEF FPC}
  inherited Create(ResStringRec^);
  {$ELSE FPC}
  inherited Create(LoadResString(ResStringRec));
  {$ENDIF FPC}
end;

constructor EJclError.CreateResRecFmt(ResStringRec: PResStringRec; const Args:
array of const);
begin
  {$IFDEF FPC}
  inherited CreateFmt(ResStringRec^, Args);
  {$ELSE FPC}
  inherited CreateFmt(LoadResString(ResStringRec), Args);
  {$ENDIF FPC}
end;


{$IFDEF FPC}
```

```
{$IFDEF MSWINDOWS}

function SysErrorMessage(ErrNo: Integer): string;
var
  Size: Integer;
  Buffer: PChar;
begin
  GetMem(Buffer, 4000);
  Size := FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM or
FORMAT_MESSAGE_ARGUMENT_ARRAY, nil, ErrNo,
    0, Buffer, 4000, nil);
  SetString(Result, Buffer, Size);
end;

procedure RaiseLastWin32Error;
begin
end;

function QueryPerformanceFrequency(var Frequency: Int64): Boolean;
var
  T: TLargeInteger;
begin
  Windows.QueryPerformanceFrequency(@T);
  CardinalsToI64(Frequency, T.LowPart, T.HighPart);
end;

procedure QueryPerformanceCounter(var C: Int64);
var
  T: TLargeInteger;
begin
  Windows.QueryPerformanceCounter(@T);
  CardinalsToI64(C, T.LowPart, T.HighPart);
end;

{$ELSE MSWINDOWS}

function SysErrorMessage(ErrNo: Integer): string;
begin
  Result := Format(RsSysErrorMessageFmt, [ErrNo, ErrNo]);
end;

{$ENDIF MSWINDOWS}
{$ENDIF FPC}


{$IFDEF MSWINDOWS}

constructor EJclWin32Error.Create(const Msg: string);
begin
  FLastError := GetLastError;
  FLastErrorMsg := SysErrorMessage(FLastError);
  inherited CreateFmt(Msg + #13 + RsWin32Prefix, [FLastErrorMsg, FLastError]);
end;

constructor EJclWin32Error.CreateFmt(const Msg: string; const Args: array of
const);
begin
  FLastError := GetLastError;
  FLastErrorMsg := SysErrorMessage(FLastError);
  inherited CreateFmt(Msg + #13 + Format(RsWin32Prefix, [FLastErrorMsg,
FLastError]), Args);

end;

constructor EJclWin32Error.CreateRes(Ident: Integer);
begin
  FLastError := GetLastError;
  FLastErrorMsg := SysErrorMessage(FLastError);
  inherited CreateFmt(LoadStr(Ident) + #13 + RsWin32Prefix, [FLastErrorMsg,
FLastError]);
end;
```

```
constructor EJclWin32Error.CreateResRec(ResStringRec: PResStringRec);
begin
  FLastError := GetLastError;
  FLastErrorMsg := SysErrorMessage(FLastError);
  {$IFDEF FPC}
  inherited CreateFmt(ResStringRec^ + #13 + RsWin32Prefix, [FLastErrorMsg,
FLastError]);
  {$ELSE FPC}
  inherited CreateFmt(LoadResString(ResStringRec) + #13 + RsWin32Prefix,
[FLastErrorMsg, FLastError]);
  {$ENDIF FPC}
end;

{$ENDIF MSWINDOWS}

procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
begin
  LowPart := TULargeInteger(I).LowPart;
  HighPart := TULargeInteger(I).HighPart;
end;

procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
begin
  TULargeInteger(I).LowPart := LowPart;
  TULargeInteger(I).HighPart := HighPart;
end;

{$IFNDEF COMPILER5_UP}

procedure TObjectList.Clear;
var
  I: Integer;
begin
  if OwnsObjects then
    for I := 0 to Count - 1 do
      Items[I].Free;
  inherited;
end;

constructor TObjectList.Create(AOwnsObjects: Boolean);
begin
  inherited Create;
  FOwnsObjects := AOwnsObjects;
end;

function TObjectList.GetItems(Index: Integer): TObject;
begin
  Result := TObject(Get(Index));
end;

procedure TObjectList.SetItems(Index: Integer; const Value: TObject);
begin
  Put(Index, Value);
end;

{$ENDIF COMPILER5_UP}

{$IFNDEF COMPILER6_UP}

procedure RaiseLastOSError;
begin
  RaiseLastWin32Error;
end;

{$ENDIF COMPILER6_UP}

{$IFDEF COMPILER4}

procedure StringListCustomSort(StringList: TStringList; SortFunc:
TStringListCustomSortCompare);
```

```
  procedure QuickSort(L, R: Integer);
  var
    I, J, P: Integer;
  begin
    repeat
      I := L;
      J := R;
      P := (L + R) shr 1;
      repeat
        while SortFunc(StringList, I, P) < 0 do
          Inc(I);
        while SortFunc(StringList, J, P) > 0 do
          Dec(J);
        if I <= J then
        begin
          StringList.Exchange(I, J);
          if P = I then
            P := J
          else
          if P = J then
            P := I;
          Inc(I);
          Dec(J);
        end;
      until I > J;
      if L < J then
        QuickSort(L, J);
      L := I;
    until I >= R;
  end;

begin
  QuickSort(0, StringList.Count - 1);
end;

{$ENDIF COMPILER4}

end.
```

LAMPIRAN I : LISTING UNIT tuningdlg.pas

File Tuning Dialog ini adalah File pendukung yang digunakan untuk menampilkan frekuensi dari nada-nada yang dimainkan.

```
interface
{$INCLUDE JEDI.inc}

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  LMDControl, LMDBaseControl, JclMath, JclMidi,  LMDBaseGraphicControl,
LMDBaseLabel,
  LMDCustomLabel, LMDLabel, ExtCtrls, ComCtrls, StdCtrls, LMDDrawEdge,
  LMDCustomButton, LMDButton, Spin;

type
  Ttuningdialog = class(TForm)
    Bevel1: TBevel;
    Bevel2: TBevel;
    Bevel3: TBevel;
    LMDLabel1: TLMDLabel;
    LMDLabel2: TLMDLabel;
    LMDLabel3: TLMDLabel;
    LMDDrawEdge1: TLMDDrawEdge;
    LMDLabel4: TLMDLabel;
    Midifreq: TEdit;
    Freqhertz: TEdit;
    Notelabel: TLMDLabel;
    OKbtn: TLMDButton;
    MidiKey: TSpinEdit;
    procedure MidifreqChange(Sender: TObject);
    procedure MidifreqExit(Sender: TObject);
    procedure FreqhertzExit(Sender: TObject);
    procedure FreqhertzChange(Sender: TObject);
    procedure MidiKeyChange(Sender: TObject);
    procedure OKbtnClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    FInMIDIFreqChange: Boolean;
    FInFreqHertzChange: Boolean;
    FChangingFrequency: Boolean;
    FChangingMidiFrequency: Boolean;
    FFrequency: Single;
    FMidiFrequency: Single;
    procedure SetFrequency(Value: Single);
    procedure SetMidiFrequency(Value: Single);
  public
    property Frequency: Single read FFrequency write SetFrequency;
    property MidiFrequency: Single read FMidiFrequency write SetMidiFrequency;
  end;

var
  tuningdialog: Ttuningdialog;

implementation

uses utama;


{$R *.DFM}
const
  HalftonesPerOctave = 12;
  MiddleA            = 440.0; // Hertz
  MidiMiddleA        = 69;    // A3 = 440 Hertz
  Digits = 6;
```

```
  MIDIFreqMax = 127.99993896;
  FreqHertzMin = 8.17579892;
  FreqHertzMax = 13289.70346552;

function Hertz(MIDINote: Extended): Extended;
begin
  Hertz := TwoToY((MIDINote - MidiMiddleA) / HalftonesPerOctave) * MiddleA;
end;

function MIDINote(Hertz: Extended): Extended;
begin
  if Hertz < 1.0 then
    MIDINote := Low(Integer)
  else
    MIDINote := LogBase2(Hertz / MiddleA) * HalftonesPerOctave + MidiMiddleA;
end;

procedure TTuningDialog.SetFrequency(Value: Single);
begin
  if FChangingFrequency or (Value = Frequency) then
    Exit;
  FChangingFrequency := True;

  try
    if Value < FreqHertzMin then
      Value := FreqHertzMin
    else
    if Value > FreqHertzMax then
      Value := FreqHertzMax;
    FFrequency := Value;
    if not FInFreqHertzChange then
      FreqHertz.Text := FloatToStrF(Value, ffFixed, 9, Digits);
    MidiFrequency := MIDINote(Value);
  finally
    FChangingFrequency := False;
  end;
end;

procedure TTuningDialog.SetMidiFrequency(Value: Single);
begin
  if FChangingMidiFrequency then
  // or (Value = MidiFrequency) then
    Exit;
  if Value < 0 then
    Value := 0
  else
  if Value > MidiFreqMax then
    Value := MidiFreqMax;
  FChangingMidiFrequency := True;
  try
    FMidiFrequency := Value;
    if not FInMidiFreqChange then
      MIDIFreq.Text := FloatToStrF(Value, ffFixed, 9, Digits);
    Frequency := Hertz(Value);
  finally
    FChangingMidiFrequency := False;
  end;
end;

procedure Ttuningdialog.MidifreqChange(Sender: TObject);
var
  F: Extended;
begin
  if FInFreqHertzChange or (MIDIFreq.Text = '') then
    Exit;
  FInMIDIFreqChange := True;
  try
    {$IFDEF COMPILER6_UP}
    if TryStrToFloat(MidiFreq.Text, F) then
    {$ELSE}
    if TextToFloat(PChar(MidiFreq.Text), F, fvExtended) then
```

```
    {$ENDIF}
      MidiFrequency := F;
  finally
    FInMIDIFreqChange := False;
  end;
end;

procedure Ttuningdialog.MidifreqExit(Sender: TObject);
begin
  MIDIFreq.Text := FloatToStrF(MidiFrequency, ffFixed, 9, Digits);
end;

procedure Ttuningdialog.FreqhertzExit(Sender: TObject);
begin
  FreqHertz.Text := FloatToStrF(Frequency, ffFixed, 9, Digits);
end;

procedure Ttuningdialog.FreqhertzChange(Sender: TObject);
var
  F: Extended;
begin
  if FInMIDIFreqChange or (FreqHertz.Text = '') then
    Exit;
  FInFreqHertzChange := True;
  try

    {$IFDEF COMPILER6_UP}
    if TryStrToFloat(FreqHertz.Text, F) then
    {$ELSE}
    if TextToFloat(PChar(FreqHertz.Text), F, fvExtended) then
    {$ENDIF}
      Frequency := F;
  finally
    FInFreqHertzChange := False;
  end;

end;

procedure Ttuningdialog.MidiKeyChange(Sender: TObject);
begin
  MIDIFrequency := MIDIKey.Value;
  NoteLabel.Caption := MidiNoteToStr(MIDIKey.Value);
end;

procedure Ttuningdialog.OKbtnClick(Sender: TObject);
begin
tuningdialog.Close;
form2.show;
end;

procedure Ttuningdialog.FormCreate(Sender: TObject);
begin
midikey.Value := form2.niltag ;
midikeychange(sender);
end;

end.
```

FAIRCHILD
SEMICONDUCTOR™

August 1986
Revised July 2001

DM7414
Hex Inverter with Schmitt Trigger Input

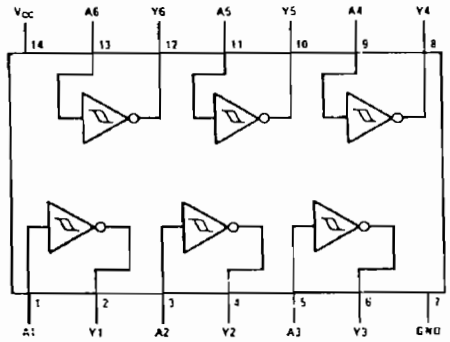
General Description

This device contains six independent gates each of which performs the logic INVERT function. Each input has hysteresis which increases the noise immunity and transforms a slowly changing input signal to a fast changing, jitter free output.

Ordering Code:

Order Number	Package Number	Package Description
DM7414N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Connection Diagram



Function Table

$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	5.5V
Operating Free Air Temperature Range	0 C to +70 C
Storage Temperature Range	−65 C to +150 C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{T+}	Positive-Going Input Threshold Voltage (Note 2)	1.5	1.7	2	V
V_{T-}	Negative-Going Input Threshold Voltage (Note 2)	0.6	0.9	1.1	V
HYS	Input Hysteresis (Note 2)	0.4	0.8		V
I_{OH}	High Level Output Current			−0.8	mA
I_{OL}	Low Level Output Current			16	mA
T_A	Free Air Operating Temperature	0		70	C

Note 2: V_{CC} = 5V

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 3)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −12 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_I = V_{T-} Max	2.4	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_I = V_{T+} Max		0.2	0.4	V
I_{T+}	Input Current at Positive-Going Threshold	V_{CC} = 5V, V_I = V_{T+}		−0.43		mA
I_{T-}	Input Current at Negative-Going Threshold	V_{CC} = 5V, V_I = V_{T-}		0.56		mA
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 5.5V			1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.4V			40	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−1.2	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 4)	−18		−55	mA
I_{CCH}	Supply Current with Outputs HIGH	V_{CC} = Max		22	36	mA
I_{CCL}	Supply Current with Outputs LOW	V_{CC} = Max		39	60	mA

Note 3: All typicals are at V_{CC} = 5V, T_A = 25 C.

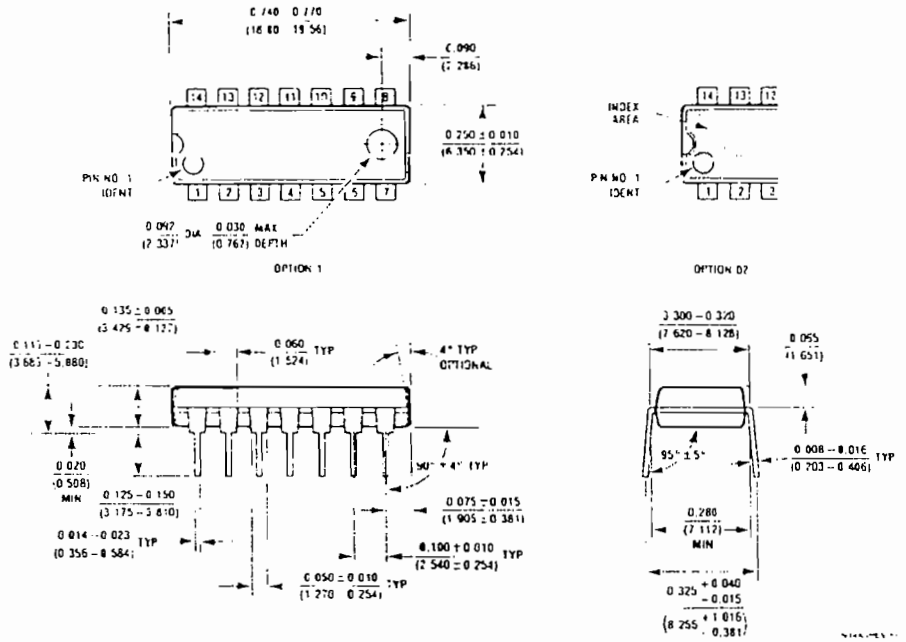
Note 4: Not more than one output should be shorted at a time.

Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Max	Units
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	C_L = 15 pF, R_L = 400Ω		22	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output			22	ns

Physical Dimensions inches (millimeters) unless otherwise noted



14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
Package Number N14A

LAMPIRAN K : DATA SHEET PPI 8255

intel

82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible
- Control Word Read-Back Capability
- Direct Bit Set/Reset Capability
- 2.5 mA DC Drive Capability on all I/O Port Outputs
- Available in 40-Pin DIP and 44-Pin PLCC
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

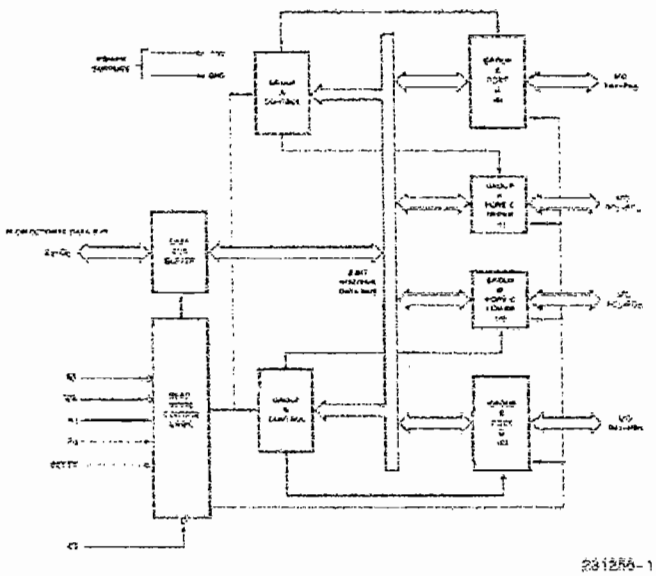


Figure 1. 82C55A Block Diagram

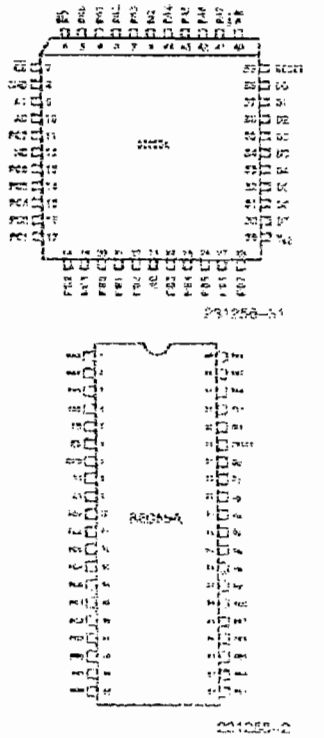


Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

Table 1. Pin Description

Symbol	Pin Number Dip	Pin Number PLCC	Type	Name and Function
PA_{3-0}	1–4	2–5	I/O	**PORT A, PINS 0–3:** Lower nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch
$\overline{RD}$	5	6	I	**READ CONTROL:** This input is low during CPU read operations
$\overline{CS}$	6	7	I	**CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
GND	7	8		**System Ground**
A_{1-0}	8–9	9–10	I	**ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (Read)
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus
					Output Operation (Write)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control
					Disable Function
X	X	X	X	1	Data Bus - 3 - State
X	X	1	1	0	Data Bus - 3 - State

PC_{7-4}	10–13	11,13–15	I/O	**PORT C, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.
PC_{0-3}	14–17	16–19	I/O	**PORT C, PINS 0–3:** Lower nibble of Port C.
PB_{0-7}	18–25	20–22, 24–26	I/O	**PORT B, PINS 0–7:** An 8-bit data output latch/ buffer and an 8-bit data input buffer.
V_{CC}	26	29		**SYSTEM POWER:** + 5V Power Supply.
D_{7-0}	27–34	30–33, 35–38	I/O	**DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus.
RESET	35	39	I	**RESET:** A high on this input clears the control register and all ports are set to the input mode.
$\overline{WR}$	36	40	I	**WRITE CONTROL:** This input is low during CPU write operations.
PA_{7-4}	37–40	41–44	I/O	**PORT A, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch.
NC		1, 12, 23, 34		No Connect

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

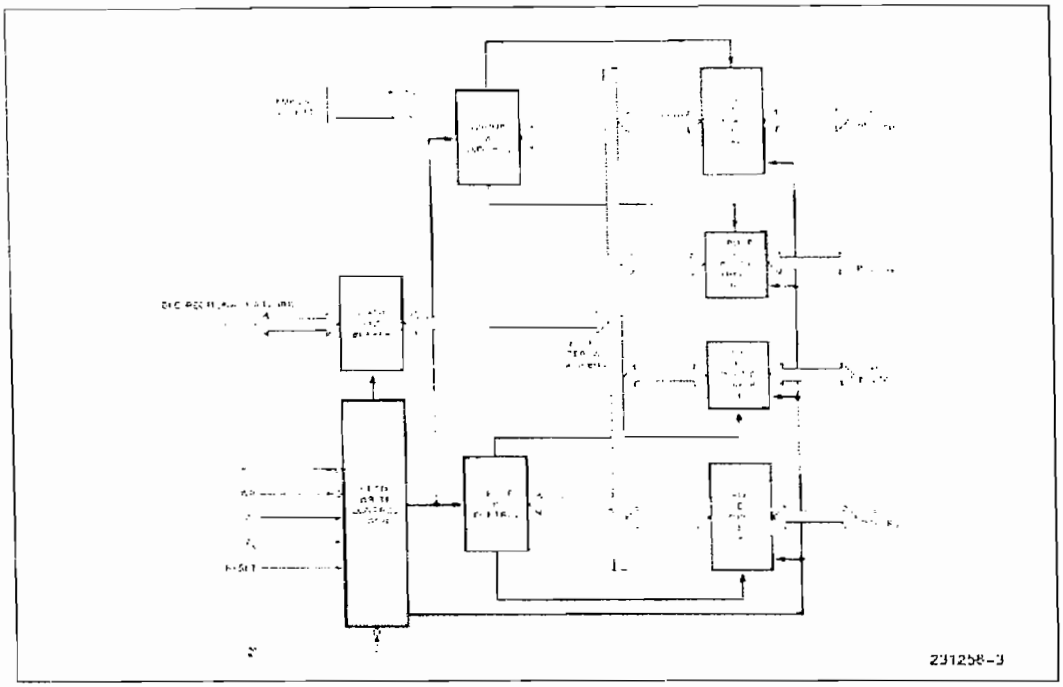
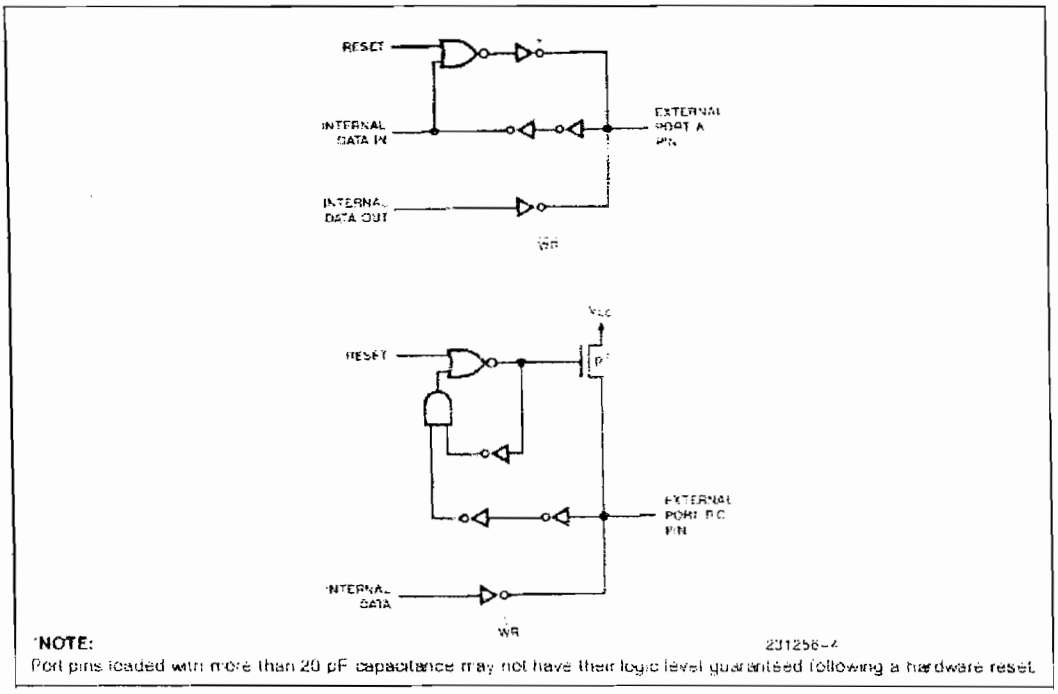


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



*NOTE:
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software.

Mode 0 — Basic Input/output
Mode 1 — Strobed Input/Output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

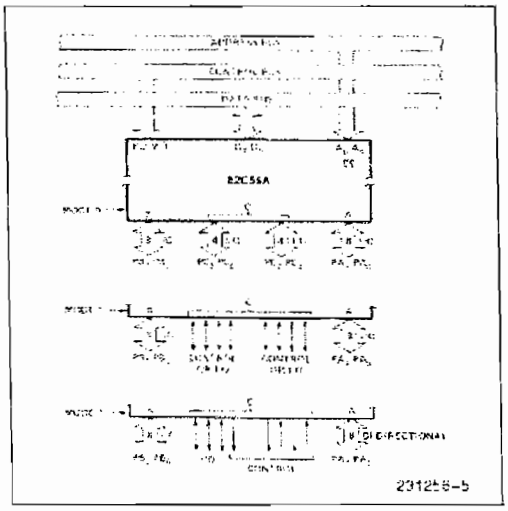


Figure 5. Basic Mode Definitions and Bus Interface

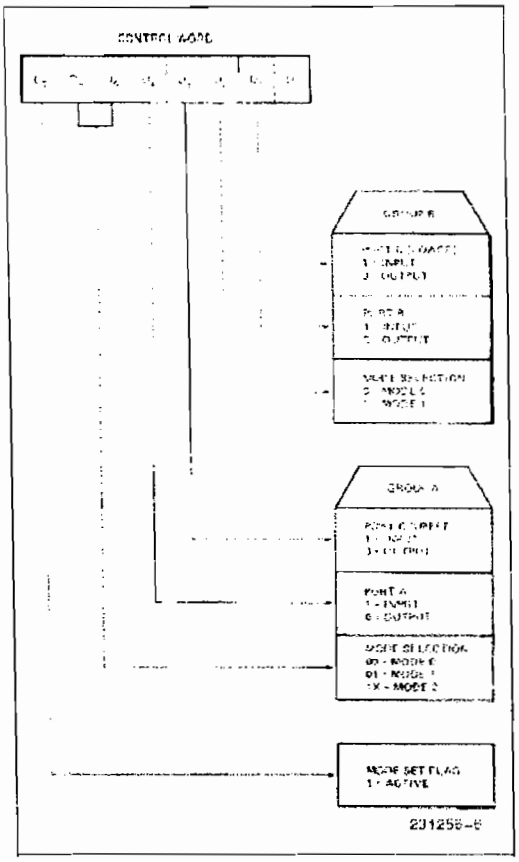


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

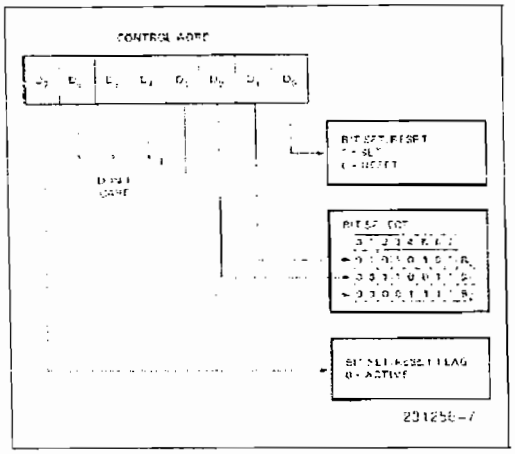


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:
All Mask flip-flops are automatically reset during mode selection and device Reset.

MODE 0 Port Definition

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	‹ ›	
PA_1	IN	OUT	IN	OUT	‹ ›	
PA_2	IN	OUT	IN	OUT	‹ ›	
PA_3	IN	OUT	IN	OUT	‹ ›	
PA_4	IN	OUT	IN	OUT	‹ ›	
PA_5	IN	OUT	IN	OUT	‹ ›	
PA_6	IN	OUT	IN	OUT	‹ ›	
PA_7	IN	OUT	IN	OUT	‹ ›	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the $\overline{ACK}$ and $\overline{STB}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

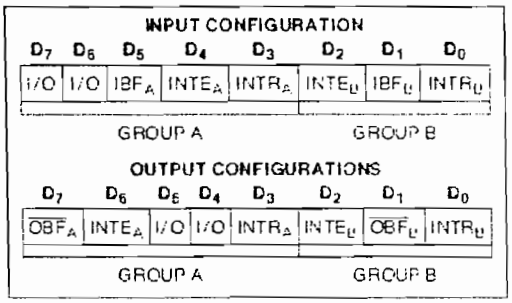


Figure 17a. MODE 1 Status Word Format

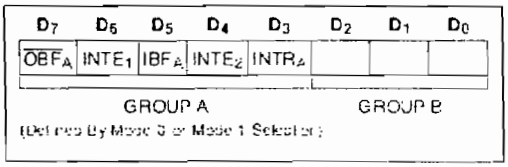


Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	$\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1)
INTE A2	PC4	$\overline{STB}_A$ (Input Mode 1 or Mode 2)
INTE A1	PC6	$\overline{ACK}_A$ (Output Mode 1 or Mode 2

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to + 70°C
Storage Temperature . . . . . 65°C to + 150°C
Supply Voltage . . . . . 0.5 to + 8.0V
Operating Voltage . . . . . + 4V to + 7V
Voltage on any Input . . . . . GND 2V to + 6.5V
Voltage on any Output . . GND 0.5V to V_{CC} + 0.5V
Power Dissipation . . . . . 1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, GND = 0V (T_A = −40°C to +85°C for Extended Temperture)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} − 0.4		V V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{IL}	Input Leakage Current		±1	μA	V_{IN} = V_{CC} to 0V (Note 1)
I_{OFL}	Output Float Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V (Note 2)
I_{DAR}	Darlington Drive Current	−2.5	(Note 4)	mA	Ports A, B, C R_{ext} = 500Ω V_{ext} = 1.7V
I_{PHL}	Port Hold Low Leakage Current	+50	+300	μA	V_{OUT} = 1.0V Port A only
I_{PHH}	Port Hold High Leakage Current	−50	−300	μA	V_{OUT} = 3.0V Ports A, B, C
I_{PHLO}	Port Hold Low Overdrive Current	−350		μA	V_{OUT} = 0.8V
I_{PHHO}	Port Hold High Overdrive Current	+350		μA	V_{OUT} = 3.0V
I_{CC}	V_{CC} Supply Current		10	mA	(Note 3)
I_{CCSB}	V_{CC} Supply Current-Standby		10	μA	V_{CC} = 5.5V V_{IN} = V_{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low $\overline{CS}$ = High Reset = Low Pure Inputs = Low/High

NOTES:
1. Pins A_1, A_0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset.
2. Data Bus: Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

OTHER TIMINGS

Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t_{WB}	$\overline{WR}$ 1 to Output		350	ns	
t_{IR}	Peripheral Data Before $\overline{RD}$	0		ns	
t_{HR}	Peripheral Data After $\overline{RD}$	0		ns	
t_{AK}	$\overline{ACK}$ Pulse Width	200		ns	
t_{ST}	$\overline{STB}$ Pulse Width	100		ns	
t_{PS}	Per. Data Before $\overline{STB}$ High	20		ns	
t_{PH}	Per. Data After $\overline{STB}$ High	50		ns	
t_{AD}	$\overline{ACK}$ 0 to Output		175	ns	
t_{KD}	$\overline{ACK}$ 1 to Output Float	20	250	ns	
t_{WOB}	$\overline{WR}$ 1 to $\overline{OBF}$ 0		150	ns	
t_{AOB}	$\overline{ACK}$ 0 to $\overline{OBF}$ 1		150	ns	
t_{SIB}	$\overline{STB}$ 0 to IBF 1		150	ns	
t_{RIB}	$\overline{RD}$ 1 to IBF 0		150	ns	
t_{RIT}	$\overline{RD}$ 0 to INTR 0		200	ns	
t_{SIT}	$\overline{STB}$ 1 to INTR 1		150	ns	
t_{AIT}	$\overline{ACK}$ 1 to INTR 1		150	ns	
t_{WIT}	$\overline{WR}$ 0 to INTR 0		200	ns	see note 1
t_{RES}	Reset Pulse Width	500		ns	see note 2

NOTE:
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum. The output Ports A, B, or C may glitch low during the reset pulse but all port pins will be held at a logic "one" level after the reset pulse.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pins returned to GND f_c = 1 MHz[5]
$C_{I/O}$	I/O Capacitance		20	pF	

NOTE:
5. Sampled not 100% tested.

A.C. CHARACTERISTICS

T_A = 0° to 70°C, V_{CC} = +5V ±10%, GND = 0V

T_A = -40°C to +85°C for Extended Temperature

BUS PARAMETERS

READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{RD}$ ↓	0		ns	
t_{RA}	Address Hold Time After $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{RD}$ ↓		120	ns	
t_{DF}	$\overline{RD}$ ↑ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{RD}$/$\overline{WR}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{WR}$ ↓	0		ns	
t_{WA}	Address Hold Time After $\overline{WR}$ ↑	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	$\overline{WR}$ Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{WR}$ ↑	100		ns	
t_{WD}	Data Hold Time After $\overline{WR}$ ↑	30		ns	Ports A & B
		30		ns	Port C

LAMPIRAN L : DATA SHEET SOUND CARD

YAMAHA LSI

YMF724F

DS-1

■ OVERVIEW

YMF724F-DS-1 is a high performance audio controller for the PCI Bus. DS-1 consists of two separated functional Blocks. One is the PCI Audio Block and the other is the Legacy Audio Block. PCI Audio Block allows Software Driver to handle maximum of 72 concurrent audio streams with the Bus Master DMA engine. The PCI Audio Engine converts the sampling rate of each audio stream and the streams are mixed without utilizing the CPU or causing system latency. By using the Software Driver from YAMAHA, PCI Audio provides 64-voice XG Wavetable synthesizer with Reverb and Variation. Built-in support DirectSound hardware acceleration, Downloadable Sound (DLS) and DirectMusic acceleration.

Legacy Audio Block supports FM Synthesizer, Sound Blaster Pro, MPU-401 UART mode and Joystick function in order to provide hardware compatibility for numerous PC games written for DOS without any software driver. To achieve legacy DMA compatibility on the PCI, DS-1 supports both PC/PC and Distributed DMA protocols. DS-1 also supports Serialized IRQ for legacy IRQ compatibility.

DS-1 supports the connection to AC'97 which provides high quality DAC, ADC and analog mixing.

In addition, it supports consumer IEC958 Audio Digital Interface (SPDIF) output for high-quality external audio amplification.

■ FEATURES

- PCI 2.1 Compliant
- PC'97/PC'98 specification Compliant
- PCI Bus Power Management Ver. 1.0 Compliant
  - Support D0, D2 and D3 states
- PCI Bus Master for PCI Audio
  - True Full Duplex Playback and Capture with different Sampling Rate
  - Maximum 64-voice XG capable Wavetable Synthesizer including GM compatibility
  - DirectSound Hardware Acceleration
  - DirectMusic Hardware Acceleration
  - Downloadable Sound (DLS) Level-1
- Legacy Audio compatibility
  - FM Synthesizer
  - Hardware Sound Blaster Pro compatibility
  - MPU-401 UART mode MIDI interface
  - Joystick
- Supports PC/PC and Distributed DMA for legacy DMA (8237) emulation
- Supports Serialized IRQ
- Supports YAMAHA AC-3 decoder (YMF727) AC-Link interface to enable AC-3 decoder
- Supports Consumer IEC958 Output (SPDIF) port
- Supports AC'97 Interface (AC-Link)
- Hardware Volume Control
- I²C PROM Interface
- Single Crystal operation (24.576MHz)
- 5V Power supply for I/O, 3.3V Power supply for Internal core logic
- 144-pin LQFP (YMF724F-V)

GENERAL MIDI SOUNDIUS-XG™

XG Sensaura™

YAMAHA CORPORATION

■ Logos

GENERAL MIDI logo is a trademark of Association of Musical Electronics Industry (AMEI), and indicates GM system level 1 Compliant.

XG logo is a trademark of YAMAHA Corporation.

SONDIUS-XG logo is a trademark that Stanford University in the United States and YAMAHA Corporation hold jointly.

Sensaura logo is a trademark of Central Research Laboratories Limited.

1. GM system level 1

   GM system level 1 is a world standard format about MIDI synthesizer which provides voice arrangements and MIDI functions.

2. XG

   XG is a format about MIDI synthesizer that is proposed by YAMAHA, and keeps the upper compatibility of GM system level 1. The good points are the voice arrangements kept extensively, a large number of the voices, modification of the voices, 3 kinds of effects, and so on.

3. SONDIUS-XG

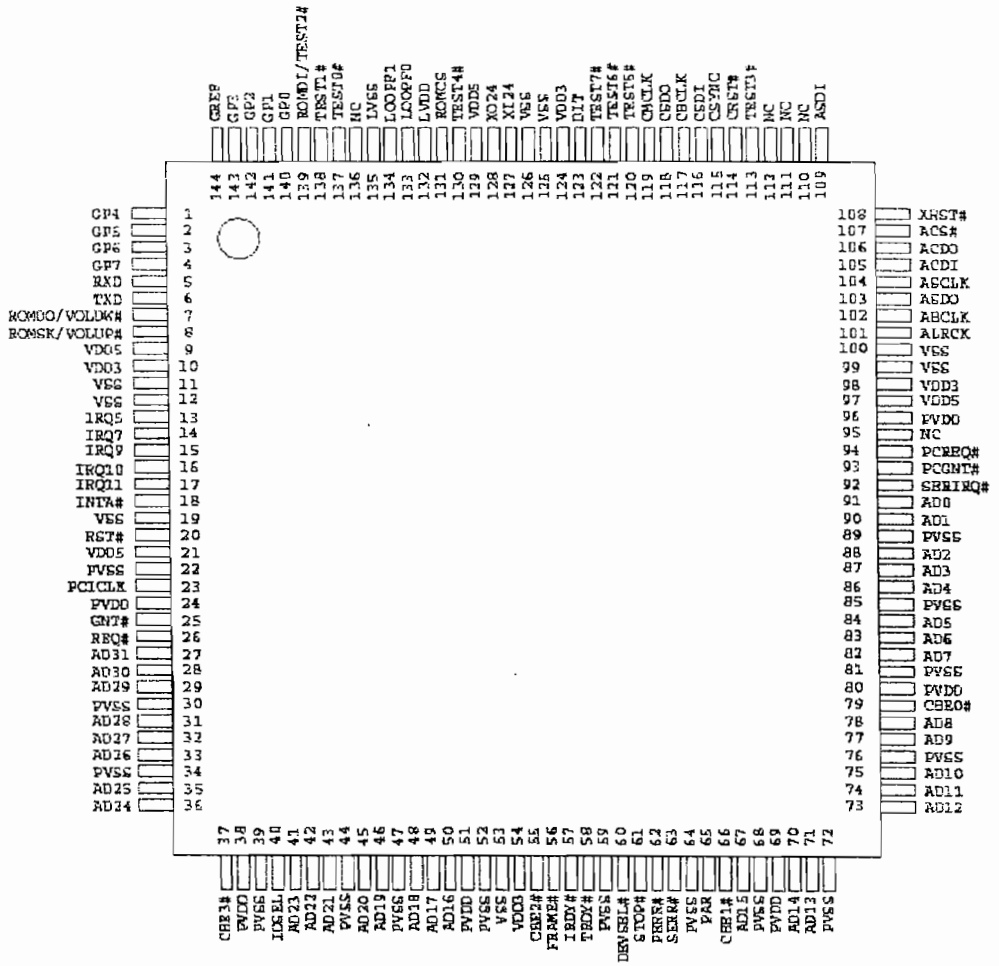
   Products bearing the SONDIUS-XG logo are licensed under patents of Stanford University and YAMAHA Corporation as listed on <http://www.sondius-xg.com>. The SONDIUS-XG produces acoustic sound outputs by running a virtual simulation of the actual acoustic instrument operation. Therefore, it provides much more real-world acoustic sound outputs fundamentally different from the Wavetable sound generator that simply processes the recorded acoustic sound sources only. The SONDIUS-XG adds the technology of virtual acoustic sound to the XG format.

4. Sensaura

   Sensaura is a technology which provides 3D positional audio and moving effect by HRTF (Head Related Transfer Function) with 2 speakers or headphone. This feature makes it possible to enjoy invariable and unchangeable sound feelings in all-positional area covering as wide as 360 degrees.

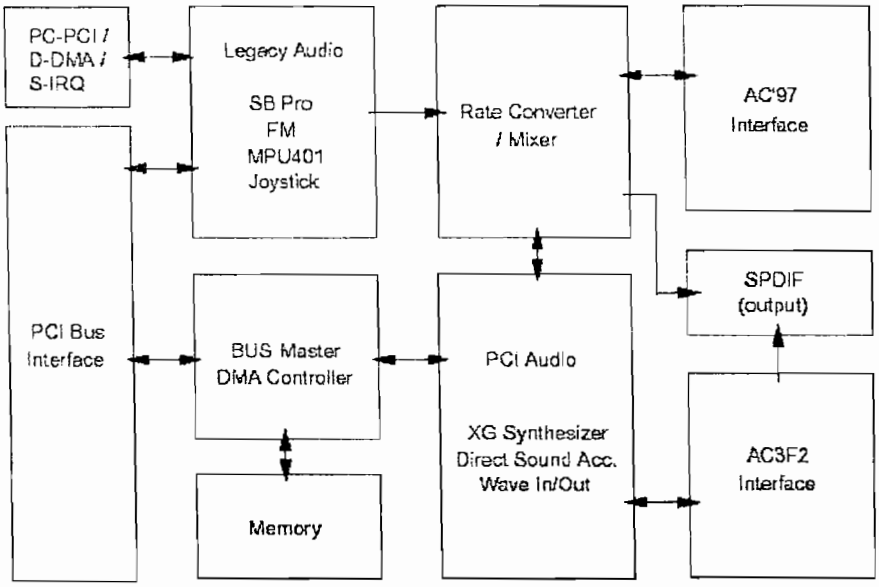
■ PIN CONFIGURATION

YMF724F-V

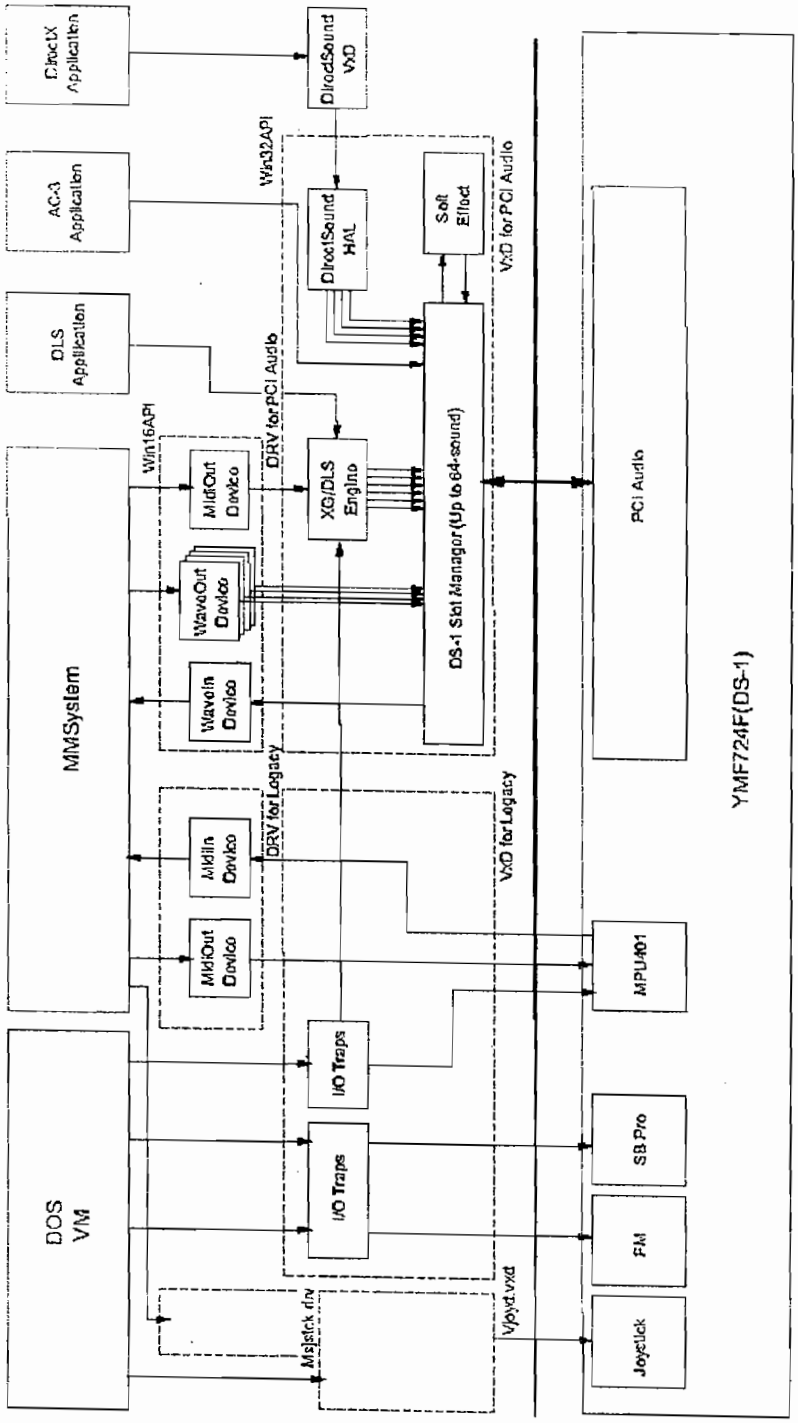


144 Pin LQFP Top View

■ BLOCK DIAGRAM



SYSTEM DIAGRAM



■ FUNCTION OVERVIEW

1. PCI INTERFACE

DS-1 supports the PCI bus interface and complies to PCI revision 2.1.

1-1. PCI Bus Command

DS-1 supports the following PCI Bus commands.

1-1-1. Target Device Mode

C/BE[3:0]#				Command
0	0	0	0	Interrupt Acknowledge (not support)
0	0	0	1	Special Cycle (not support)
0	0	1	0	I/O Read
0	0	1	1	I/O Write
0	1	0	0	reserved
0	1	0	1	reserved
0	1	1	0	Memory Read
0	1	1	1	Memory Write
1	0	0	0	reserved
1	0	0	1	reserved
1	0	1	0	Configuration Read
1	0	1	1	Configuration Write
1	1	0	0	Memory Read Multiple (not support)
1	1	0	1	Dual Address Cycle (not support)
1	1	1	0	Memory Read Line (not support)
1	1	1	1	Memory Write and Invalidate (not support)

DS-1 does not assert DEVSEL# when accessed with commands that are indicated as (not supported) or reserved.

1-1-2. Master Device Mode

C/BE[3:0]#				Command
0	1	1	0	Memory Read
0	1	1	1	Memory Write

When DS-1 becomes a Master Device, it generates only memory write and read cycle commands.

1-2. PCI Configuration Register

In addition to the Configuration Register defined by PCI Revision 2.1, DS-1 provides proprietary PCI Configuration Registers in order to control legacy audio function, such as FM Synthesizer, Sound Blaster Pro, MPU401 and Joystick. These additional registers are configured by BIOS or the configuration software from YAMAHA Corporation.

The following shows the overview of the PCI Configuration Register.

Offset	b[31..24]	b[23..16]	b[15..8]	b[7..0]
00-03h	Device ID		Vendor ID	
04-07h	Status		Command	
08-0Bh	Base Class Code	Sub Class Code	Programming IF	Revision ID
0C-0Fh	Reserved	Header Type	Latency Timer	Reserved
10-13h	PCI Audio Memory Base Address			
14-2Bh	Reserved			
2C-2Fh	Subsystem ID		Subsystem Vendor ID	
30-33h	Reserved			
34-37h	Reserved			Cap Pointer
38-3Bh	Reserved			
3C-3Fh	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line
40-43h	Extended Legacy Audio Control		Legacy Audio Control	
44-47h	Subsystem ID Write		Subsystem Vendor ID Write	
48-4Bh	DS-1 Power Control		DS-1 Control	
4C-4Fh	Reserved		D-DMA Slave Configuration	
50-53h	Power Management Capabilities		Next Item Pointer	Capability ID
54-57h	Reserved		Power Management Control / Status	
58-5Bh	Reserved		ACPI Mode	
5C-FFh	Reserved			

Reserved registers are hardwired to "0". All data written to these registers are discarded. The values read from these registers are all zero.

DS-1 can be accessed by using any bus width, 8-bit, 16-bit or 32-bit.

MIDI Programming

Introduction

There is a separate device file for each installed MIDI interface. The device name contains two decimal digits which specify the interface number. The interface number is shown in the printout produced by `cat /dev/sndstat`. For example the device file for the first installed MIDI port is `/dev/midi00`. Name `/dev/midi` is a symbolic link that points to the default MIDI device file which is usually `/dev/midi00`.

These device files have similar capabilities than the ordinary `/dev/tty` interface. Everything written to the device will be sent to the midi port as soon as possible (there could be some earlier written bytes in the queue which delay the transmit). There are no timing features which make it difficult to use these devices for sequencer like applications. The intended use for this interface is sending and receiving system exclusive messages. For example this is required when making a patch editors and librarians for various MIDI synthesizers.

Reading from the device waits until there are at least one byte in the receive buffer. When the first byte is received, the driver will not wait for additional characters. This means that the read returns usually less bytes than requested. Since the MIDI transfer rate is fairly high

(about 31 kbaud), several bytes will be received before the reading process finally gets activated and is able to continue execution of the read call. On my 486/50 system it can receive up to about 60 bytes at a time. On a slower or heavier loaded system the read could return even more data at once.

There are couple of unnecessary delays in the current implementation, but they seems to be harmless. For example it is possible to route the incoming midi data from one port into another using cat /dev/midi00 > /dev/midi01. There is no annoying delays between a keypress on the keyboard and the sound on the synth connected to the /dev/midi01.

The /dev/midi interface supports select but currently just with Linux.

To use the raw MIDI devices, you will need some knowledge

of the MIDI protocol. The official MIDI 1.0 specification is sold by MIDI Manufacturer's Association (MMA). There are some books containing the most important parts of the protocol. In addition unofficial hacks of various MIDI specifications are available using anonymous ftp from ftp://mitpress.mit.edu/pub/Computer-Music-Journal/Documents/MIDI or read an HTML version here.

Changing parameters

We don't give exact details at this time since the interface will change before final release. Here are just some hints.

There is possibility to set a time-out which the process waits for the first byte. By default it waits infinitely.

Intelligent MIDI adapters

The Roland MPU-401 and some other pro-level MIDI cards have various advanced features not available in most soundcards.

For accessing the MPU-401 in it is intelligent mode, there are two ioctl calls. The first turns on the intelligent mode (or rather it resets the card and turns off the UART mode). The second one sends a command to the card. Since some commands have parameters or return some data, there will be a way to handle these situations. Unlike in some other MPU-401 drivers, OSS will not contain separate ioctl for each of the commands. There will be just one and the command (with parameters) are passed in the argument. The MPU-401 specific features are available only with cards supporting the intelligent mode. Several sound cards have a MPU-401 compatible interface which supports the UART mode only.

We have used the MPU-401 for recording midi bytes and it works fine. The data received from the card contains all the information returned by the card, including the MIDI data, timing bytes, MPU marks and messages. I have not tried output yet but it should be possible to make it work.

The MPU interface is not there so that everyone can program the MPU-401 itself. The /dev/sequencer2 interface will support all the features of MPU-401 in way that is portable. Some other MIDI cards have advanced features like tape sync and the sequencer2 interface makes these features accessible

in device independent way. The MPU interface is there just for making it easier to port MPU specific applications from other environments. (The main reason why it is there is that it makes it easier to study ins and outs of the MPU-401).

Using the raw MPU interface requires deep knowledge about the MPU-401 internals. There are at least two sources for this information. The first is the MPU-401 Technical Reference Manual and the other is the developers library for the Music Quest MQX-32M.

What is MIDI?

MIDI is an acronym for Musical Instrument's Digital Interface (or something like it). MIDI 1.0 Detailed Specification defines both the hardware level interface and the communication protocol used for communication between devices having MIDI interface. It is primarily a data communication specification but also used in many other ways. I don't try to give complete description here. Just the rough idea.

The hardware level MIDI interface is like a RS-323 port but it is not plug compatible with it. MIDI cable has 5 pin DIN connectors at the ends and the transfer rate is nonstandard (31250 baud???). One cable can carry data just to one direction. Bi-directional connection requires two cables. More than two devices can be connected together by chaining the devices.

The MIDI devices communicate by sending messages through the MIDI cable. Every message starts with a status byte and may have one or more additional data bytes. The status byte has 1 in the most significant bit while the data bytes have 0. This means that the data bytes may have just 128 different values and to carry just 7 bits of information.

The first four bits of a status byte (status 0xF0) specify the type of the status and the last 4 bits carry the MIDI channel number. Status bytes 0xF0 to 0xFF are reserved for system messages (the last 4 bits contain the message type).

There are 16 possible channels in the MIDI cable. Each of them can be assigned to physically separate devices or some devices could interpret the messages sent to all channels. Some parameters such as instrument (program) number are assigned by channel so each device listening a particular MIDI

channel will play using the same instrument number. The device has freedom to interpret the instrument number as it wish.

For example when the player hits a key on the keyboard, a NOTE ON message is transmitted to the MIDI cable. It starts with a status byte 0x9X where the X is the channel number. There are two data bytes following the status. The first is the note number which tells which key the player has pressed. The second specifies the velocity of the keypress. The velocity is used to control the volume and some other parameters of the played sound.

It is important to notice that no sound is transferred through the MIDI line. Just instructions how the receiving instrument should control itself.

MIDI file is a file containing MIDI messages and some other data which can be used by the MIDI sequencers and other applications. It is a well defined interchange format which makes it possible to transfer songs between virtually any application supporting the format. In the PC workd these files have an extension .MID.

Unlike some other file formats for storing musical information (.MOD) MIDI files don't contain any instrument data. The instruments are defined just by including some MIDI program change messages into the files. The playing system has complete freedom to assign actual instrument timbres for the program numbers.

Preface

After the MIDI 1.0 standard was finalized in the early 1980's, numerous musical instruments with MIDI jacks appeared upon the market. Musicians started to attach these instruments via their MIDI ports, and quickly discovered that the MIDI 1.0 specification had overlooked some important concerns.

One typical scenario may have been as follows:

A musician attaches his Roland D-10 to his Yamaha DX-7, because he prefers the front panel of the D-10, but prefers the sound of the DX-7, and he wants to use the D-10 to "play" the DX-7. He selects the patch labeled "Piano" on the D-10, and he plays the D-10 keyboard, and on the DX-7 he hears... a trumpet? How did this happen? Well, it happened because MIDI sends a program change message that contains only a patch number -- not the actual name of the patch. So if patch #1 on the DX-7 is a trumpet sound, then that's what he gets on the DX-7, despite the fact

that selecting patch #1 on the D-10 yields a piano sound on the D-10. The MIDI 1.0 specification did not require that particular sounds be assigned to particular patch numbers, so every manufacturer used his own discretion as to "patch mapping".

But the real problem was with MIDI files that the musician made. MIDI files contain only MIDI messages. So, any program change event in a MIDI file refers only to a patch number as well -- not the actual patch name. So, this musician creates a MIDI file using his D-10. He has a piano track, so he puts a program change event at the track's beginning to select patch #1, which happens to be a Piano sound on his D-10. He takes that MIDI file to a friend's house. The friend has a DX-7. They play the MIDI file on that DX-7, and suddenly, the piano part is playing with a trumpet sound. Well, that's because patch #1 on the DX-7 is not a piano -- it's a trumpet sound. To "fix" the MIDI file, now the musician with the DX-7 has to edit the MIDI tracks and change every MIDI Program Change event so that it refers to the correct patch number on his DX-7. This deviance among MIDI sound modules made it very difficult for musicians to create MIDI arrangements that played properly upon various MIDI sound modules.

There were also some other deviances among early MIDI modules that made it more difficult to use them together via MIDI. To address these concerns, Roland proposed an addendum to the MIDI 1.0 specification in the late 1980's. This new addendum was called "General MIDI" (GM). It added some new requirements to the base MIDI 1.0 specification (but does not supplant any parts of the 1.0 specification -- the 1.0 specification is still the base level to which all MIDI devices should adhere). GM has now been adopted as part of the MIDI 2.0 specification.

General MIDI Patches

So to make MIDI Program Change messages of more practical use, Roland found it necessary to adopt a standard "patch bank". In other words, what was needed was to assign specific instrument sounds to specific patch numbers. For example, it was decided that patch number 1 upon all sound modules should be the sound of an *Acoustic Grand Piano*. In this way, no matter what MIDI sound module you use, when you select patch number 1, you always hear some sort of Acoustic Grand Piano sound. A standard was set for 128 patches which must appear in a specific order, and this standard is called *General MIDI* (GM). For example, patch number 25 upon a GM module must be a *Nylon String Guitar*. The chart, GM Patches, shows you the names of all GM Patches, and their respective *Program Change* numbers.

Nowadays, most modules (including the built-in sound modules of computer sound cards) ship with a GM bank (of 128 patches) so that it is easy to play MIDI files upon any MIDI module, without needing to edit all of the Program Change events in the file.

General MIDI Multi-Timbral requirement

Another burgeoning technology in the late 1980's was the multi-timbral module. Typically, there were deviances in the way that various manufacturers implemented this, since the 1.0 specification did not specifically address such devices. For example, some early multi-timbral modules supported only a limited set of the 16 MIDI channels simultaneously, so if you had a MIDI file with tracks upon unsupported MIDI channels, you wouldn't hear those tracks play back. You may not have even realized that those parts weren't being played.

So, one requirement of a GM-compliant module is that it must be fully multi-timbral, meaning that it can play MIDI messages upon all 16 channels simultaneously, with a different GM Patch sounding for each channel.

General MIDI Note Number assignments

There were also deviances in regards to Note Number mapping. For example, some manufacturers mapped middle 'C' to MIDI Note Number 60. Others mapped it to Note Numbers 72 or 48. Some modules even had middle C mapped to various places in different patches, depending upon the instrument. For example, a bass guitar patch may have middle C mapped to the highest C on the keyboard (since the most useful range on a bass guitar is below middle C). A flute patch may have middle C mapped to the lowest C on the keyboard.

The result was that, it became confusing to keep track of which key (ie, MIDI Note Number) played middle C for each patch. Also, when a MIDI track was played back upon certain modules, the part may play back an octave too high or low.

It therefore was decided that all patches must sound an A440 pitch when receiving a MIDI note number of 69. (ie, Note Number 69 plays the A above middle C, and therefore Note Number 60 is middle C).

There were deviances in regards to "drum machines" as well. Most drum machines (and drum units built into multi-timbral modules) play a different drum sound for each MIDI Note Number. But the 1.0 specification never spelled out which drum sounds were assigned to which MIDI note numbers. So, whereas note number 60 may play a snare upon one drum unit, upon another drum unit, it may

play a crash cymbal. Again, this caused trouble with MIDI files, since sometimes a drum part would play back with the wrong drum sounds.

To address this discrepancy, the GM addendum contains a "drum map". This assigns about 48 common drum sounds to 48 specific MIDI Note Numbers. The assignments of drum sounds to MIDI notes is shown in the chart, GM Drum Sounds. Also, it was decided that a GM drum unit should default to using MIDI channel 10 to receive MIDI messages. Therefore, a composer of a GM MIDI file can safely assume that his drum part will play correctly if he uses the GM Drum note assignments and records the drum part upon MIDI channel 10.

General MIDI polyphony

Polyphony is how many notes a module can sound simultaneously. For example, perhaps a module can sound 32 notes simultaneously. Early MIDI modules typically had very limited polyphony. For example, the Prophet 5 could sound only 5 notes simultaneously.

This discrepancy in polyphony among MIDI modules made it difficult for arrangers to create MIDI files that played properly upon various modules. For example, if the arranger created too "busy" an arrangement, it could exceed the polyphony of a particular module, and therefore some of the notes may not be heard.

To address this discrepancy, the GM addendum stipulated that a GM module should be capable of sounding at least 24 notes simultaneously. (Ie, It must have 24 note polyphony). It could exceed 24 note polyphony, but it had to have at least that level of polyphony. In this way, if an arranger ensured that he never had more than 24 notes sounding simultaneously in his MIDI file, all notes of his arrangement would be heard upon any GM module.

Other General MIDI requirements

Finally, the GM addendum attempted to address some other discrepancies by spelling out a few more requirements.

A GM module should respond to velocity (ie, for note messages). This typically controls the VCA level (ie, volume) of each note. but the GM addendum unfortunately did not set a specific function for velocity. Some modules may allow velocity to affect other parameters on some patches.

The pitch wheel bend range should default to +/- 2 semitones. This allows an arranger to use pitch bend messages in his arrangement without worrying whether

a bend that is supposed to be up 2 whole steps will instead jump up 2 octaves upon a certain sound module.

The module also should respond to Channel Pressure (often used to control VCA level or VCO level for vibrato depth). Again, the GM addendum unfortunately did not set a specific function for channel pressure, although typically it defaults to controlling the volume of a note while it is being held.

Finally, a GM module should also respond to the following MIDI controller messages: Modulation (1) (usually hard-wired to control LFO amount, ie, vibrato), Channel Volume (7), Pan (10), Expression (11), Sustain (64), Reset All Controllers (121), and All Notes Off (123). Additionally, the module should respond to these Registered Parameter Numbers: Pitch Wheel Bend Range (0), Fine Tuning (1), and Coarse Tuning (2).

There were also some default settings that a GM module should apply upon power up. Channel Volume should default to 90, with all other controllers and effects off (including pitch wheel offset of 0). Initial tuning should be standard, A440 reference.

General MIDI messages

The GM addendum did specify a couple System Exclusive messages to alter settings that are common to all GM units, but which were not addressed by the 1.0 specification.

One such message is for Master Volume -- not just the volume of a patch upon any one MIDI channel, but the master volume of the module itself.

There is also a System Exclusive message that can be used to turn a module's General MIDI mode on or off. This is useful for modules that also offer more expansive, non-GM playback modes or extra, programmable banks of patches beyond the GM set, but need to allow the musician to switch to GM mode when desired.

Conclusion

GM Standard makes it easy for musicians to put *Program Change* messages in their MIDI (sequencer) song files, confident that those messages will select the correct instruments on all GM sound modules, and the song file would therefore play all of the correct instrumentation automatically. Furthermore, musicians need not worry about parts being played back in the wrong octave. Finally, musicians didn't have to worry that a snare drum part, for example, would be played back on a Cymbal. The GM Standard also spells out other minimum requirements that a

GM module should meet, such as being able to respond to Pitch and Modulation Wheels, and also being able to play 24 notes simultaneously (with dynamic voice allocation between the 16 Parts). All of these standards help to ensure that MIDI Files play back properly upon setups of various equipment.

The GM standard is actually not encompassed in the MIDI specification proper (ie, it's an addendum), and there's no reason why someone can't set up the Patches in his sound module to be entirely different sounds than the GM set. After all, most MIDI sound modules offer such programmability. But, most have a GM option so that musicians can easily play the many MIDI files that expect a GM module.

NOTE: The GM Standard doesn't dictate how a module produces sound. For example, one module could use cheap FM synthesis to simulate the Acoustic Grand Piano patch. Another module could use 24 digital audio waveforms of various notes on a piano, mapped out across the MIDI note range, to create that one Piano patch. Obviously, the 2 patches won't sound exactly alike, but at least they will both be piano patches on the 2 modules. So too, GM doesn't dictate VCA envelopes for the various patches, so for example, the Sax patch upon one module may have a longer release time than the same patch upon another module.

GM Patches

This chart shows the names of all 128 GM Instruments, and the MIDI Program Change numbers which select those Instruments.

The patches are arranged into 16 "families" of instruments, with each family containing 8 instruments. For example, there is a *Reed* family. Among the 8 instruments within the Reed family, you will find Saxophone, Oboe, and Clarinet.

Prog#	Instrument
	PIANO
1	Acoustic Grand
2	Bright Acoustic
3	Electric Grand
4	Honky-Tonk
5	Electric Piano 1
6	Electric Piano 2
7	Harpsichord
8	Clavinet
	ORGAN
17	Drawbar Organ
18	Percussive Organ
19	Rock Organ
20	Church Organ
21	Reed Organ
22	Accoridan
23	Harmonica
24	Tango Accordian
	BASS
33	Acoustic Bass
34	Electric Bass(finger)
35	Electric Bass(pick)
36	Fretless Bass
37	Slap Bass 1
38	Slap Bass 2
39	Synth Bass 1
40	Synth Bass 2
	ENSEMBLE
49	String Ensemble 1
50	String Ensemble 2
51	SynthStrings 1
52	SynthStrings 2
53	Choir Aahs
54	Voice Oohs
55	Synth Voice
56	Orchestra Hit

Prog#	Instrument
	CHROMATIC PERCUSSION
9	Celesta
10	Glockenspiel
11	Music Box
12	Vibraphone
13	Marimba
14	Xylophone
15	Tubular Bells
16	Dulcimer
	GUITAR
25	Nylon String Guitar
26	Steel String Guitar
27	Electric Jazz Guitar
28	Electric Clean Guitar
29	Electric Muted Guitar
30	Overdriven Guitar
31	Distortion Guitar
32	Guitar Harmonics
	SOLO STRINGS
41	Violin
42	Viola
43	Cello
44	Contrabass
45	Tremolo Strings
46	Pizzicato Strings
47	Orchestral Strings
48	Timpani
	BRASS
57	Trumpet
58	Trombone
59	Tuba
60	Muted Trumpet
61	French Horn
62	Brass Section
63	SynthBrass 1
64	SynthBrass 2

REED		PIPE	
65	Soprano Sax	73	Piccolo
66	Alto Sax	74	Flute
67	Tenor Sax	75	Recorder
68	Baritone Sax	76	Pan Flute
69	Oboe	77	Blown Bottle
70	English Horn	78	Skakuhachi
71	Bassoon	79	Whistle
72	Clarinet	80	Ocarina

SYNTH LEAD		SYNTH PAD	
81	Lead 1 (square)	89	Pad 1 (new age)
82	Lead 2 (sawtooth)	90	Pad 2 (warm)
83	Lead 3 (calliope)	91	Pad 3 (polysynth)
84	Lead 4 (chiff)	92	Pad 4 (choir)
85	Lead 5 (charang)	93	Pad 5 (bowed)
86	Lead 6 (voice)	94	Pad 6 (metallic)
87	Lead 7 (fifths)	95	Pad 7 (halo)
88	Lead 8 (bass+lead)	96	Pad 8 (sweep)

SYNTH EFFECTS		ETHNIC	
97	FX 1 (rain)	105	Sitar
98	FX 2 (soundtrack)	106	Banjo
99	FX 3 (crystal)	107	Shamisen
100	FX 4 (atmosphere)	108	Koto
101	FX 5 (brightness)	109	Kalimba
102	FX 6 (goblins)	110	Bagpipe
103	FX 7 (echoes)	111	Fiddle
104	FX 8 (sci-fi)	112	Shanai

PERCUSSIVE		SOUND EFFECTS	
113	Tinkle Bell	121	Guitar Fret Noise
114	Agogo	122	Breath Noise
115	Steel Drums	123	Seashore
116	Woodblock	124	Bird Tweet
117	Taiko Drum	125	Telephone Ring
118	Melodic Tom	126	Helicopter
119	Synth Drum	127	Applause
120	Reverse Cymbal	128	Gunshot

Prog# refers to the MIDI Program Change number that causes this *Patch* to be selected. These decimal numbers are what the user normally sees on his module's display (or in a sequencer's "Event List"), but note that MIDI modules count the first Patch as 0, not 1. So, the value that is sent in the Program Change message would actually be one less. For example, the Patch number for Reverse Cymbal is actually sent as 119 rather than 120. But, when entering that Patch number using sequencer software or your module's control panel, the software or module understands that humans normally start counting from 1, and so would expect that you'd count the Reverse Cymbal as Patch 120. Therefore, the software or module automatically does this subtraction when it generates the MIDI Program Change message.

So, sending a MIDI Program Change with a value of 120 (ie, actually 119) to a Part causes the Reverse Cymbal Patch to be selected for playing that Part's MIDI data.

GM Drum Sounds

This chart shows what drum sounds are assigned to each MIDI note for a GM module (ie, that has a drum part).

MIDI Note #	Drum Sound	MIDI Note #	Drum Sound
35	Acoustic Bass Drum	59	Ride Cymbal 2
36	Bass Drum 1	60	Hi Bongo
37	Side Stick	61	Low Bongo
38	Acoustic Snare	62	Mute Hi Conga
39	Hand Clap	63	Open Hi Conga
40	Electric Snare	64	Low Conga
41	Low Floor Tom	65	High Timbale
42	Closed Hi-Hat	66	Low Timbale
43	High Floor Tom	67	High Agogo
44	Pedal Hi-Hat	68	Low Agogo
45	Low Tom	69	Cabasa
46	Open Hi-Hat	70	Maracas
47	Low-Mid Tom	71	Short Whistle
48	Hi-Mid Tom	72	Long Whistle
49	Crash Cymbal 1	73	Short Guiro
50	High Tom	74	Long Guiro
51	Ride Cymbal 1	75	Claves
52	Chinese Cymbal	76	Hi Wood Block
53	Ride Bell	77	Low Wood Block
54	Tambourine	78	Mute Cuica
55	Splash Cymbal	79	Open Cuica
56	Cowbell	80	Mute Triangle
57	Crash Cymbal 2	81	Open Triangle
58	Vibraslap		

A note-on with note number 42 will trigger a Closed Hi-Hat. This should cut off any Open Hi-Hat or Pedal Hi-Hat sound that may be sustaining. So too, a Pedal Hi-Hat should cut off a sustaining Open Hi-Hat or Closed Hi-Hat. In other words, only one of these three drum sounds can be sounding at any given time.

Similiarly, a Short Whistle should cut off a Long Whistle. A Short Guiro should cut off a Long Guiro. An Mute Triangle should cut off an Open Triangle. A Mute Cuica should cut off an Open Cuica.

Normally, all the above drum sounds have a fixed duration. Regardless of the time between when a Note-On is received and when a matching Note-Off is received, the drum sound always plays for a given duration. For example, assume that a

device has a "Crash Cymbal 1" sound that plays for 4 seconds. If a Note-On for note number 49 is received, that cymbal sound starts playing. If a Note-Off for note number 49 is received only 1 second later, that should not cut off the remaining 3 seconds of the sound. The exceptions may be Long Whistle and Long Guiro, which may use the duration between the Note-On and Note-off to determine how "long" the sound plays.

If a drum is still sounding when another one of its Note-Ons is received, typically, another voice "stacks" another instance of that sound playing.