APPENDIXES

Contents

A Object library

This is a synopsis of the object library. A complete description is in the *ObjectProDSP Library Reference*. Most of this appendix was extracted from the ObjectPro`++` class definitions.

A.1 DSP processing nodes

These nodes have both input and output. They transform their input and write this to the output.

A.1.1 `Add`

Synopsis: `Add` sums two or more input channels.

The parameters for `Add` are:

- `Channels`: number of input channels.
- `ElementSize`: number of multiplexed elements in each channel.
- `Scale`: scale factor for each channel.

A.1.2 `Block`

Synopsis: Converts an input stream to a new blocking and sample size.

The parameters for `Block` are:

- `ElementSize`: output element size.
- `BlockSize`: output block size.
- `OutputArithmetic`: output data: 0 - MachWord, 1 - int32, 2 - float.

A.1.3 `CxFFT`

Synopsis: `CxFFT` computes the complex FFT of a single input channel.

The parameters for `CxFFT` are:

- `LogSize`: log base 2 of FFT size.
- `Overlap`: fractional overlap of successive FFTs.
- `CenterFrequency`: position of center frequency in FFT output bins.
- `InverseFlag`: inverse FFT flag.

A.1.4 `CxFir`

Synopsis: `CxFir` is a complex symmetric (even or odd) fir filter.

The parameters for `CxFir` are:

- `Resample`: filter resampling factor (input rate/output rate).
- `ZeroPad`: zero padding of input.
- `DemodFreq`: demodulation frequency (0 for no demodulation).
- `Odd`: if set the filter length is odd.
- `Coeff`: list of unique coefficients.

A.1.5 `Demod`

Synopsis: `DemodFreq` is a complex modulation/demodulation function.

The parameters for `Demod` are:

- `DataType`: select real or complex data for input or output.
- `DemodFreq`: demodulation frequency (negative values for modulate).

A.1.6 `Demux`

Synopsis: Demultiplexes 1 input channel to `Channels` output channels.

The parameters for `Demux` are:

- `Channels`: number of output channels.
- `InputSampleSize`: number of consecutive words demultiplexed in one input sample.
- `InputElementSize`: element size of input channels.
- `OutputElementSize`: element size of output channels.

A.1.7 `FindStartTail`

Synopsis: discard initial input data within bounds.

The parameters for `FindStartTail` are:

- `LowerBound`: ignore initial input > `LowerBound`.
- `UpperBound`: ignore initial input > `UpperBound`.
- `Flags`: specieal options.
- `Skip`: Samples to skip before processing any data.

A.1.8 `Gain`

Synopsis: Gain provides a linear gain.

The parameter for `Gain` is:

- `Scale`: ratio of input amplitude to output amplitude.

A.1.9 `GainPad`

Synopsis: `GainPad` provides a linear gain.

The parameters for `GainPad` are:

- `Scale`: ratio of input amplitude to output amplitude.
- `ElementSize`: number of multiplexed elements.
- `NullOutputSample`: all samples beyond `NullOutputSample` will be 0.

A.1.10 `Integrate`

Synopsis: `Integrate` sums consecutive input vector.

The parameters for `Integrate` are:

- `IntegrationSize`: Number of samples to sum.
- `OutputStep`: number of input samples for one output.
- `Scale`: output is scaled by this factor.

A.1.11 `Interpolate`

Synopsis: sample rate conversion with linear interpolation.

The parameters for `Interpolate` are:

- `DeltaIn`: input samples for each `DeltaOut` outputs.
- `DeltaOut`: output samples for each `DeltaIn` inputs.

A.1.12 `MaskWord`

Synopsis: applies a mask to a binary data stream.

The parameter for `MaskWord` is:

- `Mask`: mask to be be applied to binary data stream.

A.1.13 `Mux`

Synopsis: Multiplexes `Channels` inputs into 1 output channel.

The parameters for `Mux` are:

- `Channels`: number of input channels.
- `InputSampleSize`: number of consecutive words in one sample.
- `OutputSampleSize`: number of consecutive words in input one sample.

- `MinimumChunk`: minimum number out input samples to process at once.

A.1.14 `PackWord`

Synopsis: packs multiple input words to a single output word.

The parameters for `PackWord` are:

- `InputWordSize`: size in bits to pack from each input.
- `InputsPerOutput`: number or input words to pack into one output.

A.1.15 `Power`

Synopsis: Power computes and scales the power in each sample.

The parameters for `Power` are:

- `Amplitude`: flag select amplitude(1) or power(0).
- `Scale`: scale factor applied before summing powers.

A.1.16 `RealFir`

Synopsis: `RealFir` is a real symmetric (even or odd) fir filter.

The parameters for `RealFir` are:

- `Resample`: filter resampling factor (input rate/output rate).
- `ZeroPad`: zero padding of input.
- `Odd`: if set the filter length is odd.
- `Coeff`: list of unique coefficients.

A.1.17 `RepackStream`

Synopsis: repack bit streams to different physical word sizes.

The parameters for `RepackStream` are:

- `OutputWordSize`: size in bits of output word.
- `InputWordSize`: size in bits of input word.
- `SignedOutput`: option(1) to treat output as signed two's compliment value.

A.1.18 `SampleDelay`

Synopsis: delays the output by a selected number of samples.

The parameters for `SampleDelay` are:

- `Delta`: signal delay in samples.
- `FillValue`: value to output before input data is copied.

A.1.19 `ToInteger`

Synopsis: converts MachWord data stream to integer.

This node has no parameters.

A.1.20 `ToMach`

Synopsis: converts binary data stream to `MachWord`.

The parameter for `ToMach` is:

- `SignedConversion`: conversion of unsigned(0) or signed(1) integer input to `MachWord`.

A.1.21 `Truncate`

Synopsis: Limit the dynamic range and significant bits in a stream.

The parameters for `Truncate` are:

- `Range`: Bits for dynamic range.
- `Accuracy`: Significant bits retained in output.
- `OverflowMode`: For overflows: 0 - saturate, 1 - truncate.
- `Round`: Flag indicates rounding (1) or truncation (0).

A.1.22 `UnpackWord`

Synopsis: unpack a single input word to multiple output words.

The parameters for `UnpackWord` are:

- `OutputWordSize`: size in bits of output word.
- `OutputsPerInput`: number or words to unpack from each input.
- `SignedOutput`: option(1) to treat output as signed two's compliment value.

A.2 Input and signal generation nodes

These nodes have no input. They are either signal generators or read data from a file or other external source.

A.2.1 `ConstantData`

Synopsis: generate a 'MachWord' constant.

The parameter for `ConstantData` is:

- `Value`: constant output level.

A.2.2 `Cos`

Synopsis: Generates the real function `Amplitude` cos(`Phase` + N `Frequency`).

The parameters for `Cos` are:

- `Frequency`: frequency (in radians per sample).
- `Phase`: phase (in radians) of first sample.
- `Amplitude`: absolute maximum amplitude of the continuous cosine function.

A.2.3 `CxCos`

Synopsis: Generates the function `Amplitude` e∧(2 Pi i(`Phase` + N `Frequency`)).

The parameters for `CxCos` are:

- `Frequency`: frequency (in radians per sample).
- `Phase`: phase (in radians) of first sample.
- `Amplitude`: absolute maximum amplitude of the continuous cosine function.

A.2.4 `CxImp`

Synopsis: Generates a periodic impulse or square wave.

The parameters for `CxImp` are:

- `Period`: the impulse is repeated after period samples.
- `Phase`: phase (in radians) of first sample.
- `Amplitude`: magnitude of impulse amplitude.
- `Width`: peak width as a fraction of sample period (0=>impulse).
- `Transition`: sample index where the first transition occurs.

A.2.5 `ImportData`

Synopsis: `ImportData` reads an ascii input file.

The parameters for `ImportData` are:

- `FileName`: name of ascii file to read.
- `Format`: 'C' format to read data values.
- `Fields`: number of data fields on each line.
- `RepeatFlag`: flag to read the file repeatedly.
- `SkipFields`: list of fields to skip (enter single 0 to skip none).
- `SkipColumns`: list of pairs of columns to skip (enter single 0 to skip none.

A.2.6 `InputNode`

Synopsis: `InputNode` reads a disk file written by an `OutputNode`.

The parameters for `InputNode` are:

- `FileName`: name of disk file to read.
- `Flags`: flags for arithmetic type and other option.
- `DeltaOut`: minimum output chunk size.

Following are the member functions of this node.

`DisplayHeader` Synopsis: display the caption and parameters read from the disk. This function returns type `void`.

This function has no parameters.

`IgnoreHeaderCount` Synopsis: ignore the word counts in the data file header. This function returns type `void`.

This function has no parameters.

A.2.7 `InputWord`

Synopsis: `InputWord` reads words in a selected format from a binary file.

The parameters for `InputWord` are:

- `FileName`: binary input file to read.
- `FormatIn`: format: `MachWord`(0), `int8`(1), `int16`(2), `int32`(3), `float`(4), `double`(5).
- `IntegerOut`: output format `MachWord`(0) or `IntegerMachWord`(1).

- `InitialSkip`: bytes to skip at start of file.
- `ElementSize`: number of words per sample.
- `BlockSize`: number of samples per block.

A.2.8 `Normal`

Synopsis: Generate normally distributed noise samples.

The parameters for `Normal` are:

- `Sigma`: standard deviation of normally distributed data.
- `Mean`: mean of normally distributed data.
- `ElementSize`: number of words in one sample.
- `Seed`: random number generator seed.

A.2.9 `Ramp`

Synopsis: generates a linear ramp function.

The parameters for `Ramp` are:

- `Min`: minimum value of ramp sample.
- `Max`: maximum value of ramp sample.
- `Increment`: increment between samples.

A.2.10 `ReadFloat`

Synopsis: `ReadFloat` reads an ascii float input file.

The parameter for `ReadFloat` is:

- `FileName`: ascii file to read.

A.2.11 `ReadInt`

Synopsis: `ReadInt` reads an ascii integer input file.

The parameters for `ReadInt` are:

- `FileName`: ascii file to read.
- `Flags`: flag for hex format input (1) and signed integer output (2).

A.2.12 `UniformNoise`

Synopsis: Generate uniformly distributed noise samples.

The parameters for `UniformNoise` are:

- `Maximum`: largest value of uniformly distributed noise.
- `Minimum`: smallest value of uniformly distributed noise.
- `ElementSize`: number of words in one sample.
- `Seed`: random number generator seed.

A.2.13 `VoiceNode`

Synopsis: `VoiceNode` reads ‘Creative Voice’ format files.

The parameters for `VoiceNode` are:

- `FileName`: input file in ‘Creative Voice’ file format.
- `NoHeader`: file does(0) or does not(1) contain a header.

Following is the member function of this node.

`DisplayHeader` Synopsis: display file header. This function returns type `void`.

This function has no parameters.

A.3 Output and display nodes

These nodes have no output. They display data or write to a file or other external device.

A.3.1 `AsciiFile`

Synopsis: `AsciiFile` writes an ascii file of data sent to it.

The parameters for `AsciiFile` are:

- `FileName`: output file name (defaults to node name).
- `Hex`: option to output data in hex format.
- `NoGroup`: option to not group complex elements on one line.
- `NoHeader`: option to omit data header.

A.3.2 `CompareDisk`

Synopsis: `CompareDisk` compares input to a file written by an `OutputNode`.

The parameters for `CompareDisk` are:

- `FileName`: name of disk file to read.
- `MaxReport`: maximum number of errors to report.
- `Tolerance`: absolute value of minimum difference for an error.
- `ErrorFile`: if set errors will be written to this file.

Following are the member functions of this node.

`DisplayHeader` Synopsis: display the caption and parameters read from the disk. This function returns type `void`.

This function has no parameters.

`IgnoreHeaderCount` Synopsis: ignore the word counts in the data file header. This function returns type `void`.

This function has no parameters.

A.3.3 `EyePlot`

Synopsis: `EyePlot` plots complex signal in eye plot (X versus Y) form.

The parameters for `EyePlot` are:

- `SamplesPerPlot`: the number of samples in one diaplay.
- `Caption`: plot caption.

A.3.4 `HexList`

Synopsis: `HexList` lists a specified number of channels to a display window.

The parameters for `HexList` are:

- `Channels`: number of input channels.
- `Caption`: list caption.

A.3.5 `Listing`

Synopsis: `Listing` lists a specified number of channels to a display window.

The parameters for `Listing` are:

- `Hex`: option(1) to output data in hex format.
- `Caption`: list caption.

A.3.6 `OutputNode`

Synopsis: `OutputNode` writes a specified number of channels to a disk file.

The parameters for `OutputNode` are:

- `FileName`: name of disk file to create.
- `Flags`: flags to overwrite exiting file and other options.
- `Channels`: number of input channels.
- `FileBlockSize`: number of consecutive samples from one channel.
- `Caption`: descriptive caption.
- `DeltaIn`: minimum input chunk size.

A.3.7 `OutputWord`

Synopsis: `OutputWord` writes words in a selected format to a binary file.

The parameters for `OutputWord` are:

- `FileName`: file to create.
- `FormatOut`: format: `MachWord`(0), `int8`(1), `int16`(2), `int32`(3), `float`(4), `double`(5).

A.3.8 `Plot`

Synopsis: `Plot` creates graphs of real, complex and two dimensional data streams.

The parameter for `Plot` is:

- `Caption`: plot caption.

A.3.9 `VoiceStripOut`

Synopsis: `VoiceStripOut` writes a Creative Voice format file with no header.

The parameter for `VoiceStripOut` is:

- `FileName`: file to create.

A.4 Functions common to node groups

These are base classes objects with member functions common to groups of node objects.

A.4.1 `DisplayNodeStr`

`DisplayNodeStr` is a base class for all nodes that have no output channels. Such nodes either display data or write it to an external device. All member functions of this base class are shared by such nodes.

Following are the member functions of this node.

`Raise` Synopsis: raise any window referencing this node. This function returns type `void`.

This function has no parameters.

`DisplayInputTiming` Synopsis: display the timing of the selected input channel. This function returns type `void`.

The parameters of this function are:

- `Channel`: input channel to display timing for.

`Edit` Synopsis: make this node available for editing in a graphical network. This function returns type `void`.

This function has no parameters.

`Unlink` Synopsis: unlink this node. This function returns type `void`.

This function has no parameters.

`LinkIn` Synopsis: select the next input channel to link to. This function returns type `Node&`.

The parameters of this function are:

- `Channel`: input channel to link to.

`NextFreeInput` Synopsis: display next free input link. This function returns type `void`.

This function has no parameters.

A.4.2 `ProcessNodeStr`

This is a base class for all ObjectProDSPprocessing nodes that have both input and output channels. Thus the member functions in this base class are common to all such ObjectProDSPnodes.

Following are the member functions of this node.

`Raise` Synopsis: raise any window referencing this node. This function returns type `void`.

This function has no parameters.

`SetSampleRate` Synopsis: set the sample rate for the network. This function returns type `void`.

The parameters of this function are:

- `Rate`: sample rate for node output.
- `Channel`: output channel to set sample rate for.

`DisplayInputTiming` Synopsis: display the timing of the selected input channel. This function returns type `void`.

The parameters of this function are:

- `Channel`: input channel to display timing for.

`DisplayOutputTiming` Synopsis: display the timing of the selected output channel. This function returns type `void`.

The parameters of this function are:

- `Channel`: output channel to display timing for.

`Edit` Synopsis: make this node available for editing in a graphical network. This function returns type `void`.

This function has no parameters.

`Unlink` Synopsis: unlink this node and connected nodes. This function returns type `void`.

This function has no parameters.

`LinkIn` Synopsis: select the next input channel to link to. This function returns type `Node&`.

The parameters of this function are:

- `Channel`: input channel to link to.

`NextFreeInput` Synopsis: display next free input link. This function returns type `void`.

This function has no parameters.

`NextFreeOutput` Synopsis: display next free output link. This function returns type `void`.

This function has no parameters.

A.4.3 `SignalStr`

`SignalStr` is a base class for all signal generation and data input nodes. The member functions in this class are available in all these nodes.

Following are the member functions of this node.

`Raise` Synopsis: raise any window referencing this node. This function returns type `void`.

This function has no parameters.

`SetSampleRate` Synopsis: set the sample rate for the network. This function returns type `void`.

The parameters of this function are:

- `Rate`: sample rate for node output.
- `Channel`: output channel to set sample rate for.

`DisplayOutputTiming` Synopsis: display the timing of the selected output channel. This function returns type `void`.

The parameters of this function are:

- `Channel`: output channel to display timing for.

`Edit` Synopsis: make this node available for editing in a graphical network. This function returns type `void`.

This function has no parameters.

`Unlink` Synopsis: unlink this node and nodes connected to its output . This function returns type `void`.

This function has no parameters.

`NextFreeOutput` Synopsis: display next free output link. This function returns type `void`.

This function has no parameters.

A.5 Network and system objects

These ObjectProDSP objects are used for controlling buffer allocation, controlling networks of nodes and defining a system of networks.

A.5.1 `CircBufDes`

Synopsis: Circular buffer descriptor.

The parameters for `CircBufDes` are:

- `Size`: buffer size in words.
- `TargetSize`: desired buffer size in target code.
- `TargetSizeGoal`: target buffer size goal (0 minimize size, 1 maximize size).
- `TargetControlGoal`: control goal (0 fixed buffer size, 1 fixed sequence).

- `MaxTargetSize`: maximum size for any single buffer.
- `MinTargetSize`: minimum size for any single buffer.

Following is the member function of this node.

`AssignToEdit` Synopsis: assign this descriptor to the network currently being edited. This function returns type `void`.

This function has no parameters.

A.5.2 `DataFlow`

Synopsis: Data flow based network control and scheduling.

The parameter for `DataFlow` is:

- `TheNet`: Network to control.

Following are the member functions of this node.

`GraphDisplay` Synopsis: display network topology and timing. This function returns type `void`.

The parameters of this function are:

- `Option`: display option (0 - short, 1 - full, 2- timing).

`Execute` Synopsis: execute network generating the specified number of samples. This function returns type `void`.

The parameters of this function are:

- `InputSamples`: input samples to process.

`AssignBuffers` Synopsis: assign buffers to a completely defined network. This function returns type `void`.

The parameters of this function are:

- `Descriptor`: specify buffer characteristics.

`ClearBuffers` Synopsis: remove all buffers from network. This function returns type `void`.

This function has no parameters.

`ClearNetwork` Synopsis: delete all network links. This function returns type `void`.

This function has no parameters.

A.5.3 `Network`

Synopsis: Data flow network objects.

This node has no parameters.

Following are the member functions of this node.

`GraphDisplay` Synopsis: display network topology. This function returns type `void`.

This function has no parameters.

`Execute` Synopsis: execute network generating the specified number of samples. This function returns type `void`.

The parameters of this function are:

- `InputSamples`: input samples to process.

`Raise` Synopsis: raise a window containing this network display. This function returns type `void`.

This function has no parameters.

`ReplaceNode` Synopsis: replace a single node in a network. This function returns type `void`.

The parameters of this function are:

- `ToReplace`: node to replace.
- `Replacement`: node to replace.

`MakeTarget` Synopsis: create target processor code for this network. This function returns type `void`.

The parameters of this function are:

- `Target`: target processor.
- `Create`: create executable flag.
- `Directory`: directory of created files.

`MakeValidate` Synopsis: create a validation test case from this network. This function returns type `void`.

The parameters of this function are:

- `DirName`: directory to place validation files.
- `ExecuteCount`: number of inputs to execute for.
- `ExtraCountCreator`: number of additional inputs to create data.
- `MaxReport`: maximum number of errors to report.
- `Tolerance`: absolute value of minimum difference for an error.
- `errorFile`: if set errors will be written to this file.

`TargetValidate` Synopsis: create a target validation test case from this network. This function returns type `void`.

The parameters of this function are:

- `Target`: target processor.
- `Create`: create executable flag.
- `DirName`: directories to place validation files.
- `ExecuteCount`: number of inputs to execute for.
- `ExtraCountCreator`: number of additional inputs to create data.
- `MaxReport`: maximum number of errors to report.
- `Tolerance`: absolute value of minimum difference for an error.
- `ErrorFile`: if set errors will be written to this file.

`SetTimingExact` Synopsis: set exact or loose timing constraints. This function returns type `void`.

The parameters of this function are:

- `Exact`: timing constraints loose(0) or exact(1).

`ReplaceWithOutput` Synopsis: replaces plot and listing nodes with an `OutputNode`. This function returns type `void`.

This function has no parameters.

`ReplaceWithCompare` Synopsis: replace each `OutputNode` with a 'Compare' node. This function returns type `void`.

The parameters of this function are:

- `MaxReport`: maximum number of errors to report.
- `Tolerance`: absolute value of minimum difference for an error.
- `ErrorFile`: if set errors will be written to this file.

`operator+` Synopsis: the '+' operator adds a thread to a `Network`. This function returns type `Network&`.

The parameters of this function are:

- `TheNode`: node to append to.

`operator>>` Synopsis: operator>> appends a node to a network. This function returns type `Network&`.

The parameters of this function are:

- `TheNode`: node to append to.

`GraphDisplayWindow` Synopsis: display network topology in a specified window size. This function returns type `void`.

The parameters of this function are:

- `Width`: maximum width in pixels of display.
- `Height`: maximum height in pixels of display.

`DisplayNames` Synopsis: display the name of the controller and buffer descriptor. This function returns type `void`.

This function has no parameters.

`SetBufferDescriptor` Synopsis: assign a buffer descriptor to this network. This function returns type `void`.

The parameters of this function are:

- `Descriptor`: buffer characteristics.

`AssociateNode` Synopsis: put this node in this networks display window. This function returns type `void`.

The parameters of this function are:

- `TheNode`: node to associate with this network.

`Link` Synopsis: make next link in network from specified node and channel. This function returns type `Network&`.

The parameters of this function are:

- `TheNode`: processing or signal generator to link to.
- `OutChannel`: output channel of selected node to link to.

`SelfLink` Synopsis: establish a feedback link in a data flow network. This function returns type `Network&`.

The parameters of this function are:

- `NodeOut`: node to link from.
- `NodeIn`: node to link to.
- `ChannelOut`: output channel of output node to link from.
- `ChannelIn`: input channel to link to.

`AssignBuffers` Synopsis: assign buffers to a completely defined network. This function returns type `void`.

The parameters of this function are:

- `Descriptor`: specify buffer characteristics.

`GetBufferDescriptor` Synopsis: get the buffer descriptor associated with this network. This function returns type `BufferDescript&`.

This function has no parameters.

`ClearBuffers` Synopsis: remove all buffers from network. This function returns type `void`.

This function has no parameters.

`GetNetController` Synopsis: get the network controller associated with this network. This function returns type `NetControl&`.

This function has no parameters.

`ClearNetwork` Synopsis: delete all network links. This function returns type `void`.

This function has no parameters.

B Mountain Math Software

Mountain Math Software was founded by Dr. Paul Budnik to create a generic framework for scientific and engineering applications that fully exploits object oriented programming and other recent advances in software technology. Digital Signal Processing was chosen as the first application area because of its importance and because Dr. Budnik has over two decades of experience in DSP research and applications. A generic data flow model developed at the Acoustic Research Center is the basis of the scheduling algorithms in ObjectProDSP.

Working closely with users on real applications was central to the design of this tool. It has been used to solve practical problems and modified as a result.

Dr. Budnik received the B. S. degree in physics and the M. S. and PhD. degrees in Computer Science from the University of Illinois in 1967, 1969 and 1975, respectively.

He founded Mountain Math Software in 1989. In addition to developing ObjectProDSP he has consulted for DSP Semiconductors, Octel Communications, Castelle and Zoran.

From March 1988 through January 1989 he was Director of Tools Systems at Zoran Corporation. He supervised the hardware and software development tools that support Zoran's family of dedicated DSP VLSI components. From June 1986 to March 1988 he was Manager of Software Engineering at Zoran responsible for Zoran's software development tools. From March 1984 to June 1986 he was a consultant and then a DSP Architect for Zoran. During this period he developed the software support tools for Zoran's vector signal processor.

From 1975 through 1984 he was associated with the Acoustic Research Center as a consultant and employee of Systems Control Incorporated and later Systems Development Corporation. He was involved and DSP algorithm research and in the design and implementation of various DSP systems.

During the 1970-71 academic year he was an Acting Assistant Professor at the University of California at Los Angeles.

He holds a patent on Zoran's DFP (Digital Filter Processor). He has presented papers at the Undersea Surveillance Symposium in 1983, 1982, 1981 and 1980.

He is author or coauthor of the following publications.

- "Algorithms and Architecture", Invited paper at the Symposium on High Speed Computer and Algorithm Organization, April 1977. Published in *High Speed Computer and Algorithm Organization*, Academic Press, 1977.
- *Techniques for Parallel Computer Design*, PhD Thesis, University of Illinois, Report #UIUCDCS R-75-763, October, 1975.
- "Measurement of Parallelism in Ordinary Fortran Programs", Computer, January 1974.
- "The Organization and Use of Parallel Memories", IEEE Transactions on Computers, December 1971.
- "TRANQUIL: A Language for an Array Processing Computer", American Federation of Information Processing Societies (AFIPS), Spring Joint Computer Conference Proceedings, Volume 34, 1969.

C GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
>
> c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally

> print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY

OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it

> does.> Copyright (C) 19yy <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19yy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index