VM LABS

Puffin2k

User Manual

April 6, 1998

VM Labs, Inc.
520 San Antonio Road
Mountain View, CA 94040
Tel: (650) 917 8050
Fax: (650) 917 8052

Copyright notice

Contents

1. Introduction

1.1 What is Puffin2k?

VM LABS
Puffin2k
C and Assembly Source-Level Debugger
for the Merlin Media Processor
© 1998/99 by VM Labs Inc.

Puffin2k is a source-level debugger for C and assembly program projects for the VM Labs Merlin Media Processor (MMP). It provides you with necessary functions for fast and efficient debugging your Merlin applications.
Puffin2k features an inuitive and easy to learn graphical user interface (GUI) and is available for Windows 95/98/NT and Linux.
Beside all the common features of a source-level debugger, Puffin2k also includes an emulator, which allows you to debug your program without connecting a Merlin development system.

1.2 Why Puffin2k?

The name "Puffin" exists for historical reasons. Jeff Minter, well known for his fondness for certain furry creatures, had named the Merlin assembler "LLAMA", allegedly an acronym for **L**ow **L**evel **A**ssembler for **M**erlin **A**rchitecture. He then went on to coin the name "ALPacA" for the (yet undeveloped) **A**bstract **L**evel **Pac**king **A**ssembler. David Betz, maintainer both of XLISP and our debugger, thought this trend was getting out of hand and chose the name "Puffin" for the debugger, by way of contrast.
The debugger went through several stages and GUI appoaches, always keeping the name "Puffin". In summer 1998 was the time for a complete redesign of the GUI and step towards C debugging. "Puffin2k" was born - hopefully the "Puffin" for the next millenium. . .

1.3 Why XLISP?

Unlike many other processors and DSP, Merlin was designed from the beginning to serve the "needs" of software engineers. A long time before the first silicon arrived, the instruction set was emulated and tested. The early emulator was written entirely in XLISP, an object oriented dialect of LISP. Today, all time critical parts has been replaced by C functions and we all have a working Merlin chip in hand, yet the emulator remains a useful tool in software development.
The interface to the debugger/emulator is still written in XLISP and XLISP is the built-in scripting language of Puffin2k. All standard functions of the debugger are available through the GUI, and there is no need to

learn or use XLISP at all. Nevertheless, it is a powerful tool and your project may benefit from use of the scripting language. This manual has its own chapter discussing XLISP and its possibilities.

2. Debugging with Puffin2k

2.1 The structure of Puffin2k

We assume that the reader of this manual is familiar with the Merlin Media Processor and especially the Merlin Processor Elements. One basic goal of Puffin2k is to reflect the architecture of the chip in a graphical representation.
Therefore the main windows of Puffin2k are:

MMP window containing all chip-wide functions

MPE windows each containing the context of one MPE

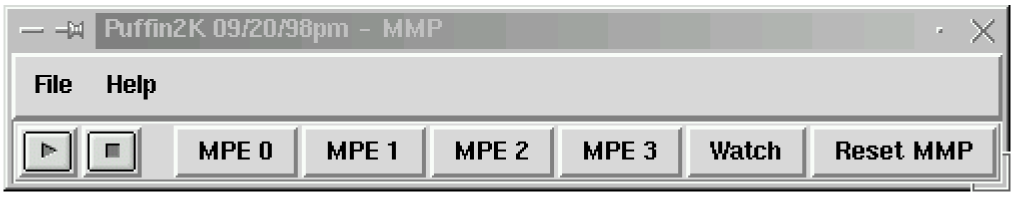
Command window Interface to the XLISP interpreter

Watch window holds all your symbolically-defined registers and memory dumps in assembly mode

Variable window holds all your global symbols in C debugging mode

2.2 MMP window

The MMP window is the main window of Puffin2k. It contains buttons to access the MPE windows, watch window and a menu.



2.2.1 MMP File menu

Load debugger file

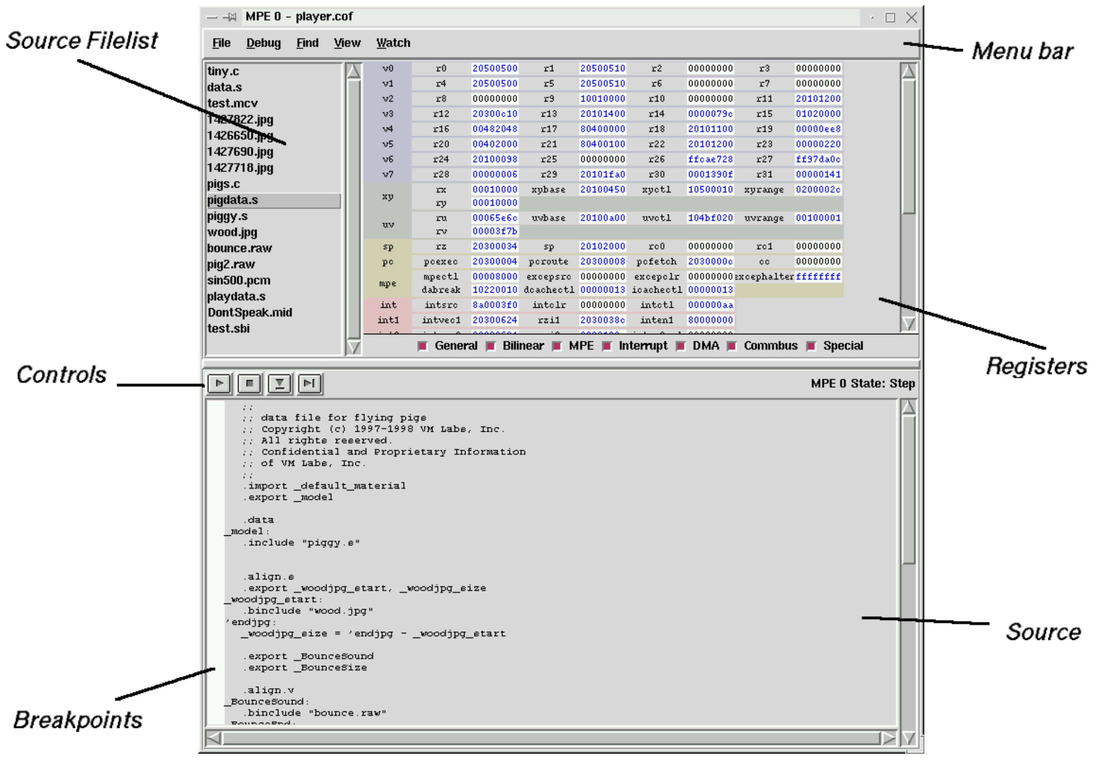
This menu entry allows you to load a debugger file. A debugger file is written in XLISP and can contain all valid XLISP commands in Puffin2k. Debugger files are helpful if you want to setup a specific debug environment, without having to do this by hand every time you start the debugger. Please refer to the scripting chapter for more detailed information about the XLISP scripting language.

Exit

Exits Puffin2k and saves window positions, sizes and the file history list.

2.3 MPE window

Each MPE window contains the context of a processor element:



```
tiny.c
data.s
test.mcv
1427822.jpg
1426650.jpg
1427690.jpg
1427718.jpg
pigs.c
pigdata.s
piggy.s
wood.jpg
bounce.raw
pig2.raw
sin500.pcm
playdata.s
DontSpeak.mid
test.sbi
```

```
 v0     r0     20500500  r1        20500510  r2        00000000  r3         00000000
 v1     r4     20500500  r5        20500510  r6        00000000  r7         00000000
 v2     r8     00000000  r9        10010000  r10       00000000  r11        20101200
 v3     r12    20300c10  r13       20101400  r14       0000079c  r15        01020000
 v4     r16    00482048  r17       80400000  r18       20101100  r19        00000ee8
 v5     r20    00402000  r21       80400100  r22       20101200  r23        00000220
 v6     r24    20100098  r25       00000000  r26       ffcae728  r27        ff97da0c
 v7     r28    00000006  r29       20101fa0  r30       0001390f  r31        00000141
 xy     rx     00010000  xybase    20100450  xyctl     10500010  xyrange    0200002c
        ry     00010000
 uv     ru     00065e6c  uvbase    20100a00  uvctl     104bf020  uvrange    00100001
        rv     00003f7b
 sp     rz     20300034  sp        20102000  rc0       00000000  rc1        00000000
 pc     pcexec 20300004  pcroute   20300008  pcfetch   2030000c  cc         00000000
 mpe    mpectl 00008000  excepsrc  00000000  excepclr  00000000  excephalter ffffffff
       dabreak 10220010  dcachectl 00000013  icachectl 00000013
 int    intsrc 8a0003f0  intclr    00000000  intctl    000000aa
 int1 intvec1  20300624  rzi1      2030038c  inten1    80000000
```

```
   ;;
   ;; data file for flying pigs
   ;; Copyright (c) 1997-1998 VM Labs, Inc.
   ;; All rights reserved.
   ;; Confidential and Proprietary Information
   ;; of VM Labs, Inc.
   ;;
   .import _default_material
   .export _model

   .data
_model:
   .include "piggy.s"


   .align.s
   .export _woodjpg_start, _woodjpg_size
_woodjpg_start:
   .binclude "wood.jpg"
'endjpg:
  _woodjpg_size = 'endjpg - _woodjpg_start

   .export _BounceSound
   .export _BounceSize

   .align.v
_BounceSound:
   .binclude "bounce.raw"
```

2.3.1 MPE File Menu

Load COFF

With this menu entry you can choose a COFF file to be loaded into the MPE. COFF files could be produced by the LLAMA assembler, the gcc C compiler or the vmld linker.

Load MPO

With this menu entry you can choose an MPO file to be loaded in the MPE. MPO files are normally produced by the LLAMA assembler.

Load COFF Symbols

Load the symbol table of a COFF file. This might be useful in the case you want to debug a program that wasn't loaded into a MPE by the debugger (e.g. it was loaded and started by another MPE).

Refresh

This menu entry forces an update of the register and watch windows.

Restart

Resets the MPE and loads the current project again. This can take a few seconds.

File history

The file hstory contains the last four loaded MPO or COFF files.

2.3.2 MPE Debug Menu

This menu contains all debugging functions to control the MPE. Many of the functions are also available in the toolbar.

Run

Starts the MPE at the current program counter.

Stop

Stops the MPE at the current program counter.

Step in

Steps one instruction packet or C source line. This function will enter subroutines/functions.

Step over

Like "Step In", but without entering subroutines or functions.

Disassemble at

Shows you the disassembly starting at the given address or label.

Set/Clear Breakpoint

This entry open a small dialog box where you can specify a valid label, function name or address to set a breakpoint. If there is already a breakpoint at the specified location, it will be cleared.
You also can set breakpoints directly by clicking in the first column of the source (see also "Current Source")

Remove all breakpoints

Deletes all defined breakpoints.

2.3.3 MPE Find menu

Find

In a dialog box you specify a word to search in the current selected source file. You also can specify the search direction and wether the search sould be case sensitive or not.

Find again

Repeats the last search, starting at the curent cursor position.

Clear Highlight

Clears the current selection

Goto Line

Jumps to a given line.

2.3.4 MPE View Menu

If you're debugging a C project, you can switch between the register view and the local variable view. If you're stepping from a C to an assembly function or vice versa, Puffin2k automatically switches to the right view.

View Registers

Shows the MPE registers. Works in assembly and C programs (see "Register Frame").

View Variable

Shows the local variables which are currently in scope (see "C Variable Browser").

View Source/disassembly

The menu entry allows you to switch between source and disassembly view.

2.3.5 MPE Watch Menu

Add watch

By choosing this function you can add an entry to the watch window. It opens a dialog box where you have to specify several parameters of the watch entry:

Symbol name This could be a

- Register name (e.g., r5 or acshift)
- Register equate (e.g., TotalSum)
- Label (e.g., SineTable)

Range Here you specify how many scalars (32bit values) you'd like to see in the watch window. In case of watching a register, the range value specifies how many registers following you like to see. Example: Watching $r0$ with a range of 4 gives you the vector $v0$ ($r0 \ldots r3$)

Format Defines the format in which the date is presented:

Hex	32bit hexadecimal representation
Decimal	decimal representation
Binary	32bit binary representation
ASCII	ASCII representation. Non-printable characters (0-31,128-255) are shown as ".".
Fix	32 bit Fixed Point representation. You have to specify the number of fractional bits in the edit field.

Bitfield Defines a bitfield description used for this date. All bitfield descriptions from the register window (like $intsrc$) are available. It's also possible to define own bitfield description with the $define-bitfield$ XLISP function.

Indirect option Uses the given symbol as a pointer rather than showing it value. One example is watching the stack pointer sp indirect with range of e.g. 16 scalars. In this case you can always see the last 4 $push$ed vectors on the stack!

Local address Interprets the address of the symbol from an MPE local address space view (default)

Look thru cache If the MPE is running in cached mode you can specify this option to look through the cache.

You can activate the "Add watch" function also by double clicking on an symbol in the source window. The word under the mouse pointer will then be copied automatically in the symbol name field. **NOTE:** Puffin2k does not check if it is a valid symbol name at double-click time!

Remove selected

You can select one ore more entries in the watch window by clicking on the symbol name. The pressing the Delete key or choosing this menu entry, all selected watch entries will be deleted.

Add global C varibale

Adds a global C variable to the C variable window. In C-mode you can execute this function by a double-click on a global C variable in the source window.

2.3.6 Source file list

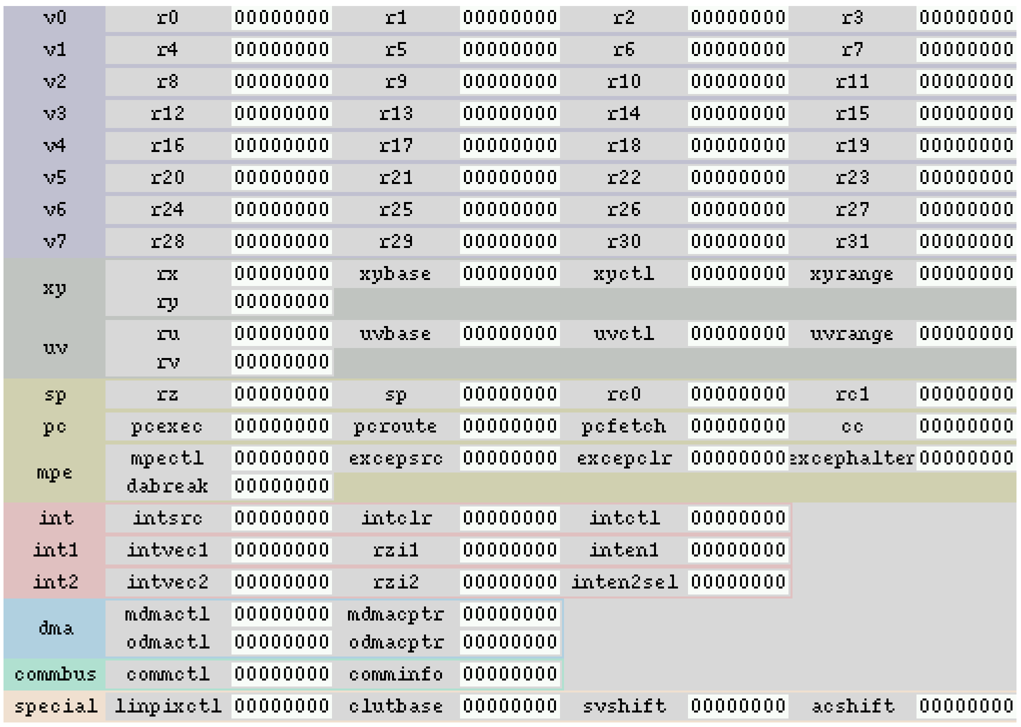
This list in the upper left corner of the MPE window shows you all source files in your loaded project (COFF or MPO). The highlighted file is shown in the source frame below. You can change the currently shown source by simply clicking on the file name in the list.

2.3.7 Current source frame

This part of the MPE window shows you the current selected source in the source file list. Puffin2k automatically tries to show the file and line of the current program counter (pc). The position of the pc is indicated by the highlighted line(s). Because of the existance of instruction packets in assembly language and multiple line C commands, it's possible that more than one line is highlighted.
In the first column of the source frame you can set and remove breakpoints by a single mouse click. You can set breakpoints only on valid lines!
A double click on a word (= one or more character(s) surrounded by whitespace), opens the "Add watch" dialog and adds the word to the symbol name entry (see "MPE Watch Menu").

2.3.8 Register Frame

This part of the MPE window shows all MPE registers. Because there are so many registers, Puffin2k groups them according to their function:



v0	r0	00000000	r1	00000000	r2	00000000	r3	00000000
v1	r4	00000000	r5	00000000	r6	00000000	r7	00000000
v2	r8	00000000	r9	00000000	r10	00000000	r11	00000000
v3	r12	00000000	r13	00000000	r14	00000000	r15	00000000
v4	r16	00000000	r17	00000000	r18	00000000	r19	00000000
v5	r20	00000000	r21	00000000	r22	00000000	r23	00000000
v6	r24	00000000	r25	00000000	r26	00000000	r27	00000000
v7	r28	00000000	r29	00000000	r30	00000000	r31	00000000
xy	rx	00000000	xybase	00000000	xyctl	00000000	xyrange	00000000
	ry	00000000						
uv	ru	00000000	uvbase	00000000	uvctl	00000000	uvrange	00000000
	rv	00000000						
sp	rz	00000000	sp	00000000	rc0	00000000	rc1	00000000
pc	pcexec	00000000	pcroute	00000000	pcfetch	00000000	cc	00000000
mpe	mpectl	00000000	excepsrc	00000000	excepclr	00000000	excephalter	00000000
	dabreak	00000000						
int	intsrc	00000000	intclr	00000000	intctl	00000000		
int1	intvec1	00000000	rzi1	00000000	inten1	00000000		
int2	intvec2	00000000	rzi2	00000000	inten2sel	00000000		
dma	mdmactl	00000000	mdmacptr	00000000				
	odmactl	00000000	odmacptr	00000000				
commbus	commctl	00000000	comminfo	00000000				
special	linpixctl	00000000	clutbase	00000000	svshift	00000000	acshift	00000000

General The 32 genaral purpose registers

Bilinear The bilinear index registers

MPE MPE control registers (like the program counters)

Interrupt Interrupt control registers

DMA DMA control registers

Comm Bus Communication Bus registers

Special Some special registers, which don't fit in one of the above groups.

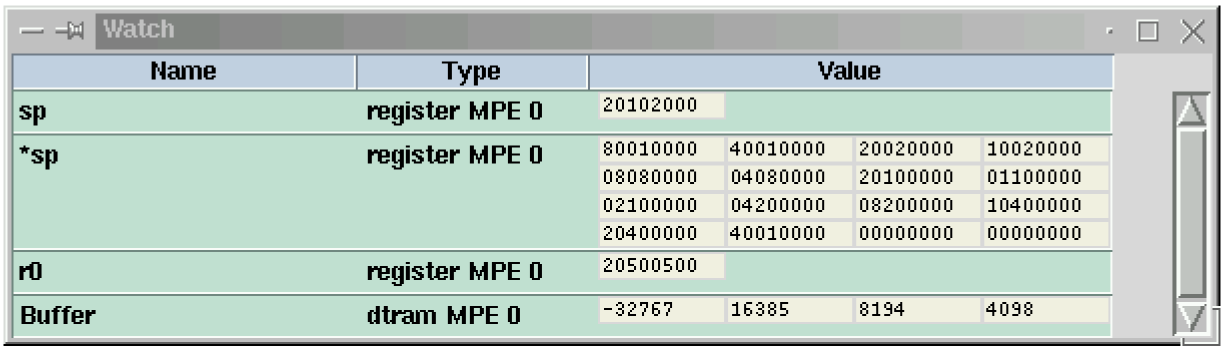
Every group can be turned on and off by clicking on the corresponding checkbox below the registers.
All register values are shown in 32bit hexadecimal. Values that have been changes since the last processor halt are shown in blue color. You can change the value of a register just by overwriting the value and hitting the RETURN key.
If you "park" the mouse pointer over the register name, it will pop-up a yellow register info dialog, which show the value of the register in different representations. Registers which are divided in bitfields, will be expanded to show bitfield names (as specified in the Merlin MMA Manual) and values.

2.3.9 C variable browser

In case of debugging a C project you can switch between the register view and local C variables (see "MPE View Menu"). All local variables in scope are shown in browser style. A small triangle indicates that the variable is a structure (or pointer). By clicking on the triangle, you can open the next level of the structure.

2.4 Watch window

The watch window shows all entries you've added with the "Add watch" menu. Every block represents one watched symbol. Depending on the specified watch range the block can contain more that one line of data.



Name	Type	Value			
sp	register MPE 0	20102000			
*sp	register MPE 0	80010000	40010000	20020000	10020000
		08080000	04080000	20100000	01100000
		02100000	04200000	08200000	10400000
		20400000	40010000	00000000	00000000
r0	register MPE 0	20500500			
Buffer	dtram MPE 0	-32767	16385	8194	4098

A watch entry is divided into three sections:

Symbol name The name of the watched symbol. A preceding "*" indicates (as in C) that the indirect option was selected and that the symbol is used as a pointer.

Symbol type The type of the symbol can be

MPE Register including the MPE number

Type of memory IRAM, DTRAM, SDRAM, SysRAM or ROM

Data The data is shown in the selected format. In hex and decimal mode, four scalars per line are shown. In Fixed Point format, one line contains two values. Only one binary value per line is shown.

As in the register window, values changed since the last processor halt are shown in blue color. All values are editable. You can just overwrite the current value and hit RETURN to confirm. **Note:** You normally specify the new value in the same format as displayed. If you want to use a different format, you have to specifiy it explicitly using one of the following prefixes:

0x	Hex
0b	Binary
0d	Decimal

2.5 Variable window

In case of C debugging you can use the watch function to add global symbols to this window. The representation and functionality is identical with the local variable frame in the MPE window.