PBC - Pulse Blaster Compiler

version: α 0.1 (in development)
Revised: Feb 11, 2002

PBC 0.1 is distributed as freeware. I'm not aware of any bugs that could blow up your power amplifier, but it's understood that you use this software at your own risk.
All other standard freeware disclaimers apply.

Contents

1. Introduction

PBC (Pulse Blaster Compiler) is a GUI around PBCE (Pulse Blaster Compiler Engine). PBCE is a software component (object) that translates (compiles) pulse programs written in a simple, easy to read language (PP="pulse program") into binary programs which can be directly uploaded to *PulseBlaster* (SpinCore Technologies, Inc.). PBC wraps PBCE, and allows user to open/edit/save pulse programs and run them directly if *PulseBlaster* is installed in the PC.
PBCE will be used in NMR/NQR program *SeveNMR* developed at J. Stefan Institute, Ljubljana, Slovenia.
It is easier to develop/debug/test PBCE if it is included in a small program, and PBC is just that.

Basically, if you want to PulseBlaster to produce a train of 2 microseconds separated 1 microsecond long pulses on output bit 0 (pin 13), you don't have calculate and write hex code like:

-
```
00000000030000000000002F
00010000000000600000061
```
-

Instead, write pulse program like:

-
```
Start:
   PULSE [1] 1u
   DELAY 2u
GOTO Start:
```
-

and PBC will compile it to hex code and (optionally) start (trigger) PulseBlaster.

2. Installation

2.1 Windows 95, 98, ME

Unzip PBC.EXE and PBC.INI to any directory of your choice.
Edit PBC.INI and write appropriate PulseBlaster hardware address [PPCARD] section.

2.2 Windows NT, 2000

Install PBC.EXE like for Win95.
Prior to running PBC you have to install also DriverLINX driver to enable I/O port access under Windows NT. A small installation of DriverLINX is available as freeware at:
<http://venezia.cx/~diskdude/software/cbuilder/index.html>.
You have to have administrator privileges to install DriverLINX.

2.3 Linux

There is no Linux version of PBC as yet, although in principle one would need only to recompile Delphi source code under Kylix.

3. PBC program description

3.1 Starting PBC

Upon starting PBC program, PBC.INI initialisation file is loaded. Specify `base`, `frequency`, and `memory` values in the `[PPCARD]` section. Specify pulsemasks for the named pulses in `[PULSEMASKS]` section.

An example of PBC.INI file

```
[PPCARD]
; used only for PULSEBLASTER card
type=PB
; hardware base address
base=$340
; frequency in MHz
frequency=50
; internal (memory=0) or external (memory=1)
memory=0

[PULSEMASKS]
; definitions of pulse masks come here,
; edit this section to suit your connections
; format: <pulsename>=<mask>
; <pulsename> can have up to 10 characters
; <mask> can be either in hex (e.g. $10)
;        or in dec (e.g. 16)
+X=$01
+Y=$02
-X=$04
-Y=$08
TRIGGER=2
GATE=$80
+X_GATE=$81
ALL_PULSES=$FFFFFF
EVEN_PULSES=$555555
ODD_PULSES=$AAAAAA

; Masks for pins on 37 pin J10 connector
PIN_13=$01
PIN_25=$02
PIN_24=$04
PIN_11=$08
PIN_10=$10
PIN_22=$20
PIN_21=$40
PIN_08=$80
PIN_07=$0100
PIN_19=$0200
PIN_18=$0400
PIN_05=$0800
PIN_04=$1000
PIN_16=$2000
PIN_15=$4000
PIN_02=$8000
```

3.2 Using PBC

Neither features nor GUI of PBC are in a stable form as yet, but its usage should be quite self-evident. Basically user has to:

1) Start the program
2) Write pulse program in the “Source” window (alternatively, you can Open pulse program from disc, or paste source code to “Source” window.
3) Click “Parse”. Program will be parsed and eventual parsing errors will be reported. Parsed program will be shown in “Parsed” window.
4) Click “LoadParsed”. Program will be loaded (using the current value of the variables) to *PulseBlaster* card and *PulseBlaster* will be triggered. The value of *PulseBlaster* status bits should be displayed in the status bar and on the right of the screen.
5) Click “Stop” to stop the program

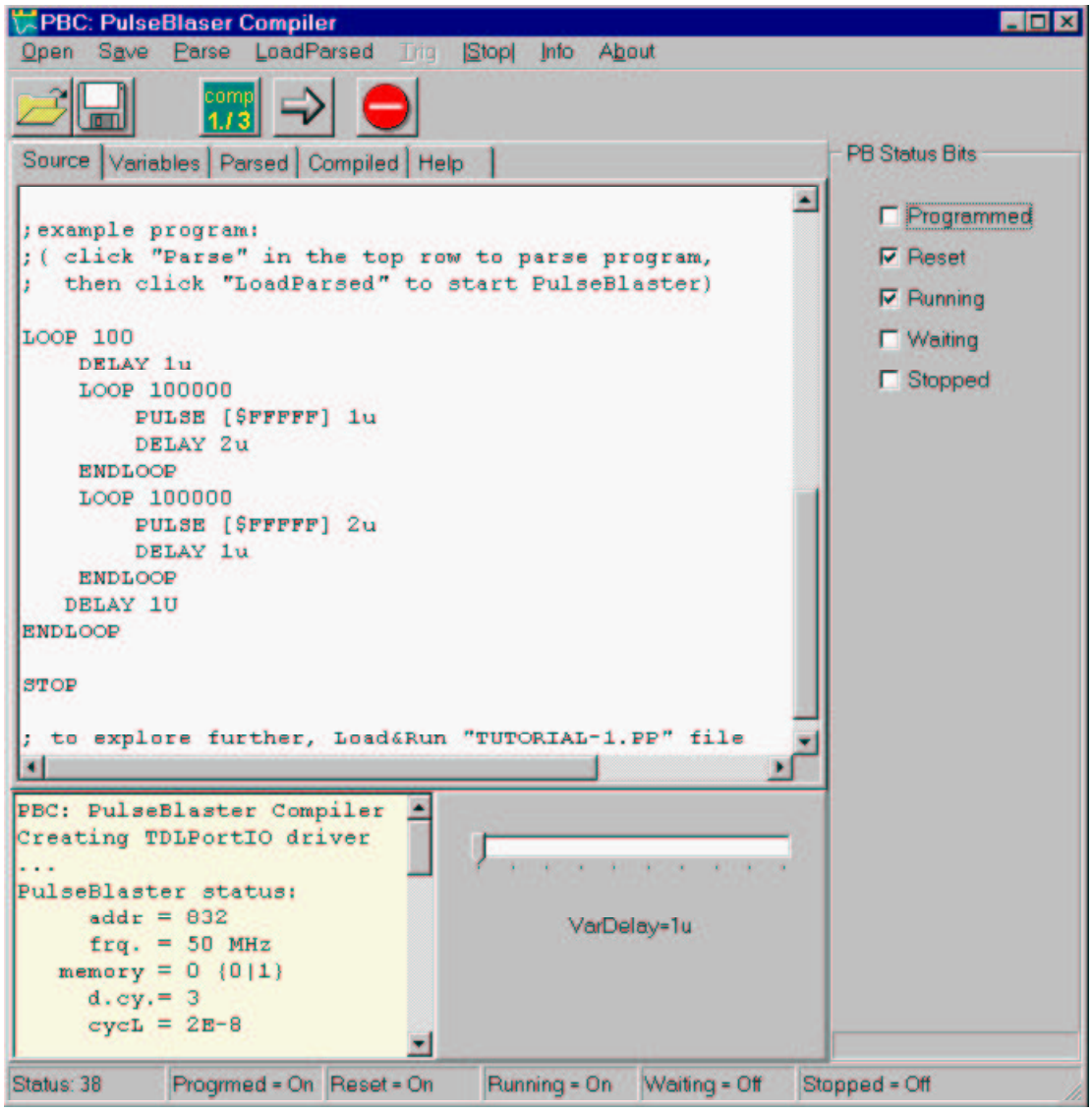

Fig.1.
PBC main screen with “Source” window.

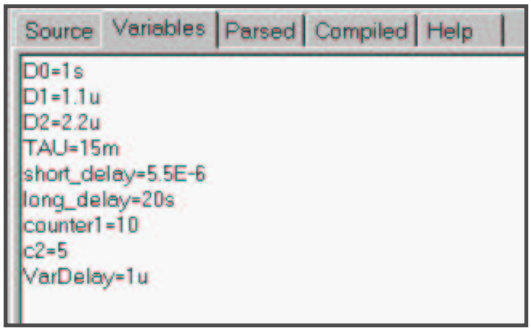

Fig.2
Variables, defined in “Variables” window.

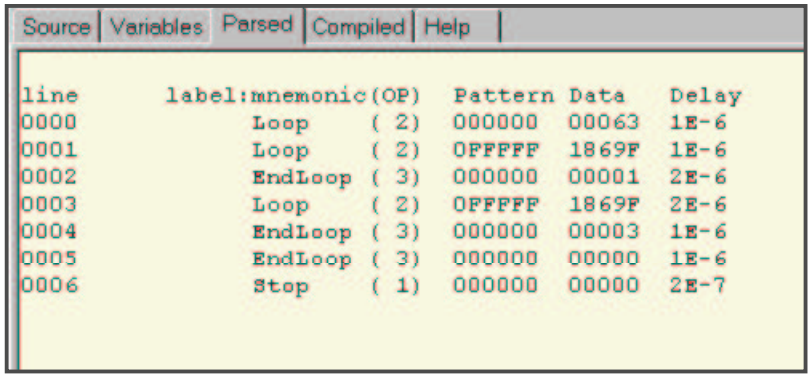

Fig. 3
Parsed program of Fig. 1

4. PP (“PulseProgram”) language

PP programs are ASCII files with .PP extension.

Comments

If the first character of the PP line is semicolon (;), the line is treated as a comment.

Labels

PP program lines can be optionally labelled by a word, terminated by a colon (:). Label can stand on an empty line, or in front of PP command.

Warning:

In some cases (most, in fact), two PP commands are compiled into singe *PulseBlaster* VLIW. Only the first command can be labeled in such a case. Therefore there must be no labels in front of `GOTO`, `GOSUB`, `RETURN` or `ENDLOOP` commands, or immediately after `LOOP` command.

Example:

```
Jump1:      DELAY 10u
            GOSUB Subrut1:
            PULSE [1] 10u
            GOTO Jump1:

Subrut1:
            PULSE [$FF] 100u
            RETURN
STOP
```

4.1 DELAY

Syntax:

`DELAY` *delay*

Parameters:

delay : delay in seconds

DELAY command inserts a "blank" delay of length *delay* with all pulses off.

Parameter *delay* can be:

- constant real number, optionally followed by **"s"** for second (default), **"m"** for millisecond, **"u"** for microsecond, or **"n"** for nanosecond
- variable name (in PBC, variable values are defined in "variables" window.
- general expression, such as `1m+1u, D1-D2 ,` etc.
- in PBC v0.1, only short delays (up to approx. 65 seconds) are supported

Examples:

```
DELAY 10u
DELAY 0.01m
DELAY 0.00001
DELAY 0.00001s
DELAY tau
DELAY D4-1u
```

Remarks:

DELAY and PULSE commands are usually compiled into *PulseBlaster* "Continue" Very-Long Instruction Word (VLIW). If they follow LOOP, or stand in front of ENDLOOP, GOTO, GOSUB or RETURN commands, they will be integrated into *PulseBlaster* "Loop", "End Loop", "Branch", "Jump SR" or "Return SR" VLIWs.

Warning:

Long delays (over 2^{24} *PulseBlaster* clock cycles ~ approx. 1 minute) are not supported in the current version of PBC.

4.2 PULSE

Syntax:

`PULSE [`*pulses*`]` *delay*

Parameters:

- *pulses*: pulse mask (s)
- *delay*: length of pulse in seconds

PULSE command inserts a pulse of length *delay* to the PulseBlaster output lines, defined by [*pulses*]. Parameter *pulses* represents pulse masks. Parameter *pulses* can take any value between 0 and 2^{24}-1. Pulse masks, specified in hexadecimal system, must be preceded by $ character. Several *pulses,* separated by commas, can be specified in PULSE command. In this case, logical OR is applied to *pulses* bitmaps. *pulses* bitmaps can be given in terms of named pulse. Named pulses are not variables, like *delay*s, but are rather constant bitmasks, specified in the PBC.INI file.

Examples:

```
PULSE [3] 1u
PULSE [1,2] 1u
PULSE [+X,GT] 1u
PULSE [$FF] 0.5U
PULSE [255] 500n
```

Remarks:

`PULSE` command is usually compiled into *PulseBlaster* "Continue" VLIW.
" `PULSE [] delay` " is the same as " `DELAY delay` ". See remarks for `DELAY`.

4.3 STOP

STOP command is directly compiled into *PulseBlaster* "Stop" VLIW. No parameters are required (a minimum delay count is inserted by default).

4.4 GOTO *label:*

Unconditionally continue execution of the program on a labelled line:
Example:

```
StartAgain:
      PULSE 10u
      DELAY 10u
      GOTO StartAgain:
```

Remark:
GOTO command and DELAY/PULSE command in front of it are compiled into a single *PulseBlaster* Branch VLIW.

4.5 GOSUB *label:* ……. RETURN

Jump
Example:

```
DELAY 10u
GOSUB Comb:
DELAY 20u
GOSUB Comb:
DELAY 30u
GOSUB Comb:
STOP
;subroutines
Comb:
      PULSE [+X] 5u
      DELAY 5u
      PULSE [+X] 5u
      RETURN
```

Remark:
GOSUB command and DELAY/PULSE command in front of it are compiled into a single *PulseBlaster* "Jump SR" VLIW. RETURN command and DELAY/PULSE command in front of it are compiled into a single *PulseBlaster* "ReturnSR" VLIW.

4.6 LOOP *n* ……. ENDLOOP

Repeat LOOP *n* times.

Remark:

LOOP n command must be followed by DELAY or PULSE command, since they are compiled into a single *PulseBlaster* "Loop" VLIW. ENDLOOP command must be preceded by DELY or PULSE command. Consequently, you need at least 2 "DELAY or PULSE" commands in the "LOOP … ENDLOOP" sandwich.

4.7 WAIT

WAIT command is directly compiled into *PulseBlaster* "Wait" VLIW. No parameters are required (a minimum delay count is inserted by default).

5. Support

No support for PBC is promised, but I will be grateful for any feedback or bugs report sent to tomaz.apih@ijs.si.