A Guide To

Platform Independent Programming

For The Propeller Chip

Very Rough Draft - 2008-06-28

Index

Introduction

For All Propeller Chip Users

- Configuring Source Code for your Hardware
- Creating a Hardware Platform Definition

For Program Developers

- Writing Code for Platform Independence
- Creating a Program Template

Appendices

- Example Program Template
- Example Pindefs_Platform.spin file

Introduction

This guide is intended for all Propeller Users - for program developers to create source code which can be run on any Propeller hardware with the minimum amount of change and for program users to describe how such changes are made.

It is not necessary for developers or program users to understand how the Propeller Platform Independent Programming is achieved as the framework for creating and using Platform Independent programming has been made as simple and easy to use as possible for both developers and users of their programs.

In most cases a user of a program which is Propeller Platform Independent will need to make just a one line change to get that program to compile and run with their platform. A considerable improvement over having to edit _CLKMODE, _XINFREQ and every I/O pin definition for every program downloaded which does not match their particular hardware configuration.

For program developers the changes required are slightly greater but are equally easy to achieve; all that is mainly required is to not use constant definitions for I/O pins but to use variables to define them instead. For most developers this will be a fairly simple change to make and brings the advantage that users of other hardware platforms will be able to use those programs with a minimum of change needed.

Progam developers and program users are all encouraged to adopt the Propeller Platform Independent Programming Framework as this will have benefits for the entire Propeller community, existing Propeller users and those who are new to the Propeller Chip and its programming.

For All Propeller Chip Users

Configuring Source Code for your Hardware

At the top of the main program there will be some lines similar to the following -

```
OBJ
  pindefs : "Pindefs_ProtoBoard"
```

If the name there ("ProtoBoard" in this case) matches the hardware you are using you do not need to change anything; you can compile and download the source code as is.

If the hardware described is not correct, you should alter it to match the hardware you are using, for example, if using the SpinStamp you will need to change this to -

```
OBJ
  pindefs : "Pindefs_SpinStamp"
```

If you are using your own or modified hardware you will need to change the name to indicate the hardware you are using and you will need to create a definition file if one does not already exist. If you call your modified ProtoBoard "Bob's modified ProtoBoard" you would change the definition line to something like -

```
OBJ
  pindefs : "Pindefs_Bob_Modified_ProtoBoard"
```

You will then need to create a *Pindefs_Bob_Modified_Protoboard.spin* file which contains the definition for your hardware. This process is defined in the next section.

Once the name has been changed to match your hardware and any necessary hardware definition file has been created, you can compile and download the program as you normally would.

Creating a Hardware Platform Definition

To use your own or standard but modified hardware you must create a hardware definition file for it. If the hardware platform specified in the main program source code is for example -

```
OBJ
  pindefs : "Pindefs_Bob_Modified_ProtoBoard"
```

You will need to create a *Pindefs_Bob_Modified_Protoboard.spin* file which describes that hardware.

Launch the **EditPindefs.exe** program, click New and enter the name of the hardware description. The "Pindefs_" prefix can either be included or excluded; the program will create the correctly named files regardless. Spaces will be converted to underscores.

Select the clock mode and crystal frequency your system uses.

For each of the 32 Propeller Chip pins (64 for the Propeller Mk II), select what each pin will be used for. If the pin name of a device you are using is not included in the drop-down lists, you will need to click Create New Pin Name.

Complete the pop-up form specifying what the name of the new pin will be and enter a description of its use. The application will check that the name has not already been used and will register its use with a central database via your internet connection. If you do not have an internet connection the name will be accepted but it should be registered as soon as possible to prevent someone else officially registering that name for another use.

The program will keep an up to date copy of all registered pin names and will automatically download the latest list from the central registry via the internet when the registry changes.

When you have assigned all pins and created all new pin name definitions click Ok and your hardware description will be created.

The program will automatically create the required *.spin* files for your hardware definition in the main Propeller Library area and also keep a backup in its own installation directory so definitions are not lost when the Propeller Tool is upgraded or re-installed.

For Program Developers

Writing Code for Platform Independence

The main difference between traditional programming and platform independent programming is that pins used and other platform specific information is not hard-coded into the program but is determined at run time.

For Propeller developers that means not including _CLKMODE, _XINFREQ nor any pin name constants, and using variable names rather than constant names throughout the program when referencing an I/O pin.

Obviously these variables will need to be initialised prior to use and this can be done manually or by generating a Program Template which can be copied into your existing or new Propeller program.

Creating a Program Template

Run the **EditPindefs.exe** program, select the hardware you are using and click Create Program Template.

You will be asked to select whether you are creating a program which will run standalone or is meant to be run under a particular Propeller Operating System.

Select whatever is appropriate and Click Ok. This will create a pop-up window showing the Program Template for the hardware selected and this will automatically be copied to the ClipBoard.

If the hardware definition itself specifies this is a Propeller OS platform then this selection will be automatically made as appropriate and the Program Template will be immediately produced.

This Program Template can then be cut and pasted into an existing program or used as the basis of the new program you are creating.

The Program Template will contain variable definitions for all pins defined by the hardware. Those which are not required by the program you are writing can be deleted or commented out.

Appendices

Example Program Template

A typical program template will look something like this ...

```
OBJ
  pindefs : "Pindefs_ProtoBoard"

VAR
  long txProp           ' Pin for Transmit to PropTool
  long rxProp           ' Pin for Receive from PropTool
  long i2cScl           ' Pin for I2C Clock ( 10K pull-up )
  long i2cSda           ' Pin for I2C Data ( 10K pull-up )

PUB Main
  pindefs.SetPins
  txProp := pindefs.GetPin(pindefs.Packed(String("TX_PROP")))
  rxProp := pindefs.GetPin(pindefs.Packed(String("RX_PROP")))
  i2cScl := pindefs.GetPin(pindefs.Packed(String("I2C_SCL")))
  i2cSda := pindefs.GetPin(pindefs.Packed(String("I2C_SDA")))
```

Note that there are no _CLKMODE, _XINFREQ nor pin definition constants included in the program. All these will be automatically determined at run-time. The clock mode and crystal frequency will be set on the first call of pindefs.GetPin().

Example Pindefs_Platform.spin file

A typical hardware definition file will look something like this ...

```
CON
  _CLKMODE = XTAL1+PLL16x
  _XINFREQ = 5_000_000

CON
  TX_PROP = 31          ' Pin for Transmit to PropTool
  RX_PROP = 30          ' Pin for Receive from PropTool
  I2C_SCL = 29          ' Pin for I2C Clock ( 10K pull-up )
  I2C_SDA = 28          ' Pin for I2C Data ( 10K pull-up )

OBJ
  pindefs : "Pindefs_Library"

PUB SetPins
  pindefs.SetPin(pindefs.Packed(String("CLKMODE")), _CLKMODE )
  pindefs.SetPin(pindefs.Packed(String("XINFREQ")), _XINFREQ )
  pindefs.SetPin(pindefs.Packed(String("TX_PROP")), TX_PROP  )
  pindefs.SetPin(pindefs.Packed(String("RX_PROP")), RX_PROP  )
  pindefs.SetPin(pindefs.Packed(String("I2C_SCL")), I2C_SCL  )
  pindefs.SetPin(pindefs.Packed(String("I2C_SDA")), I2C_SDA  )

PUB GetPin( pinName )
  return pindefs.GetPin( pinName )

PUB GetPinName( pinNumber )
  return pindefs.GetPinName( pinNumber )

PUB Packed( strPtr )
  return pindefs.Packed( strPtr )
```