Enjoy reading the March issue of the HP-39/40g newsletter!

1. Letter from the editor
2. Program of the month
3. Upcoming programs
4. Interview with Joshua King
5. Michaël's cosy programmers corner

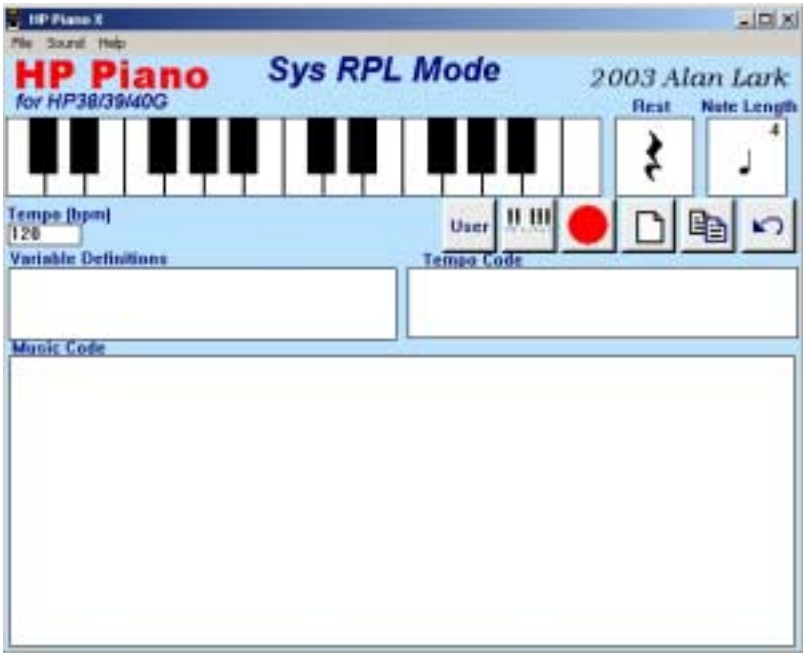
Letter from the editor.

Yeah, we are already March! March is my second most preferred month of the year (first one is July because it's summer, vacation, birthday of this newsletter and my birthday). Why? Simple, because spring is nearing. Well at least here at the northern part of the world. And that makes me happy ☺. You have probably noticed it: again this newsletter is late ☹. I even don't have a good reason for its late arrival. But I apologize for everyone that was waiting on it. Sorry.

In this issue we have an interview with Joshua King (Program of the Month winner from previous month). We have an excellent Program of the Month. There are upcoming programs. And we learn about input forms in Sys-rpl.

Enjoy reading it, the editor, Michaël De Coninck

Program of the month.



The program of this month is '**HP-Piano X**'. Something I'm very glad about! This version features: Sys-RPL support! Real-time recording! Keyboard input! And also some small improvements over the last version. This program is (if you ask me) A MUST if you want to do something with music on your HP-38/39(+)/40g. I love it and use it every time when I make music in my games/aplets (hopefully you see the result of that very soon). When I saw an email with the title 'HP PIANO X IS HERE!!!!' in my inbox, I was so happy, you can't even imagine. Alan you did a very, very good job with this version. E.g.: If you now press on the Note Length key you see the rest and the length of the note, such seen as on a score. There is Sys-rpl support. You can now use the keyboard if you prefer it over the mouse to select notes. And there is real-time support.

Specs:

System: HP-38/39(+)/40g.
Author: Alan Lark
Download where: http://www.hphomeview.com/zipfiles/hppianox.zip

Upcoming programs.

Not really upcoming, but worth mentioning. I've released **blackjack** and **who wants to become a euro millionaire**. You can download it on hpcalc.org or hphomeview.com. For the French readers of this newsletter there is a French version (translated by HP Monsieur) on hp-network.com.

The advantage of being late! **U-lib 0.0** is released!!! Why 0.0? Because its author didn't include all the functions he wanted. It's Open Source so this program is really A MUST for everyone around. Download it on: http://www.hp-network.com/HP40G/Library/U-Lib%200.0%20-%20BETA.zip

Joshua King is working on a development kit for the HP-39(+)/40g (hp-basic).

I wouldn't be me if I wouldn't announce new projects again. Two new secret projects (games) are coming to the HP-39/40g☺.

By the way you may also vote on which game I have to finish first. **SimCity 2004**, **Enslaver** or **Dragonall Z: The Golden Tournament**. To make the choice a little bit easier I've put some screenshots beneath. If you have made your choice write me an email.

Dragonball Z

Enslaver

SimCity 2004

Interview with Joshua King.

Interview	
Michaël	How old are you, and what of education do you have?
Joshua	I'm 19 years old, and I finished high school two years ago. I'm currently studying towards a Software Engineering degree at the University of Western Australia.
Michaël	Where do you live?
Joshua	I live in a suburb of Perth, Western Australia.
Michaël	What calculators do you own?
Joshua	I own both a HP38G and a HP39G.
Michaël	What was the first program you ever wrote?
Joshua	On the calculators, the first (usable) program that I wrote was a random number generator, basically rolling a six-sided dice. There's a version of this still on my web-site.
Michaël	What do you use your calculator for the most?
Joshua	Mostly I wrote programs (the user-level kind) and doing maths work at high school.
Michaël	Why did you start programming calculators and how did you learn it?
Joshua	I had always tinkered with programming on computers, and so when I got my graphics calculator one of the first things I did was start programming it. I learned by reading the somewhat obscure manual (!) and also by reading other's code to learn new things.
Michaël	Are there any programmers you admire?
Joshua	Not really, I haven't actually met many or got involved in the 'cult' side of it all, but in saying so all of the programmers of the HP's are friendly and helpful and comp.sys.hp48 is a great community. Jordi's Ticking Clock is amazing though -- that's my favourite HP program, I only wish it was around when I had more time to use it.
Michaël	Are there any programmes you want to see on your calculator?
Joshua	I know games always go down well, and I'm a Java freak so I have visions of there one day being an implementation of J2ME (the mobile phone version of Java) running on the calculators, but I have nowhere near enough experience to pull that off.
Michaël	Any future projects?
Joshua	At the moment I really just want to finish my development kit project -- it's almost there but it's pretty big, then my focus will return to finishing my degree. After that, who knows?
Michaël	What do you see for the future of HP?
Joshua	The HP community is a really tight and loyal bunch, so they have a good future, but I think HP is losing its grip on the high school market, especially here in Australia, so the high-end calculators (48/49 series) will become the dominant focus for HP. (This is my totally unqualified opinion!)

Michaël's cosy programmers' corner.

Hi this month it's part 1 about input forms. Let's begin:

Everyone who has programmed in HP-Basic knows that input forms needs a lot of arguments. Well then do I have good news for you! In Sys-rpl you need even MORE arguments to make an input form (that's my humour I guess☺). And I'm not talking about one or two more, but quite a lot more…

But don't go running away now, just because of that. You have to see it from the positive side. The Sys-Rpl input form is much more versatile than the HP-Basic one (lists, strings, etc. you can put more text on the screen and have much more control over the input form etc.). Anyway let's begin:

Make an aplet with the name 'input form' and as file name 'inputform' in WSR 1.1b. Scroll down till the beginning of the program and delete the usual lines.
Input forms exist out of two main parts, namely fields and labels. With labels I mean the text like e.g.: "A is =" and with fields I mean the black boxes where you can enter things. In Sys-rpl, unlike HP-Basic, you can have more fields than labels (and vice versa). In this example where going to make a simple input form, where we are going to enter real values.

We are going to start with defining the labels, where they have to be on the screen, and what they have to say. We are going to place "A is =" on the coordinates (13,16). So write `"A is ="` `THIRTEEN SIXTEEN`. Now when the input form will be made the text will be somewhere in the middle of the screen.

Now we are going to make a field where we can enter real numbers and only real numbers. We begin with defining the message handler. What is a message handler? Well that is something which we can manipulate the input form (or the particular field) with. E.g.: You can add a 'A…Z' key, change the soft buttons, and much more. But in this example we don't need a message handler (but in a later lesson we will, so don't worry). So we write `'DROPFALSE`. This indicates that we don't have a message handler. Now we have to define the X-coordinate, the Y-coordinate, the length of our field, and the height of our field. All of these arguments have to be BINTS. As X-coordinate we take 44. So write `BINT44`. For the Y-coordinate we do most of the time Y-coordinate of the label - 2. In this case that is 16 – 2 = 14. So we write: `BINT14`. Of course it's your choice so do whatever you want. Now we have to define the length of the field. Here it's 78 pixels wide. So write: `BINT78`. And now as a last thing we have to tell the compiler what the height is of the black box. Normally that is nine. But of course that's up to you to decide. Anyway we will write: `BINT9`. So far we have:

```
"A is =" THIRTEEN SIXTEEN

'DROPFALSE
BINT44
BINT14
BINT78
BINT9
```

Now we are going to define the type of field we want. We can choose between a field where the user can enter something, the user can enter things or choose out of a list, or the user can only choose out of list with elements. For more information about this I refer to the 'programming in Sys-rpl' document. We are going to make a field where the user can enter something. To do that we simple have to write `ONE`. Or BINT1 (what you prefer, that is actually the same). Now we say to the input form which things he may allow. We only want to allow real numbers so we write `{ ZERO }`. Actually you may also write just `ZERO`. The `{ }` is actually if you want to allow more than one thing. Like BINTs and reals too. But that is not the case here. Now we define how the field has to decompile the numbers we entered. Using Standard mode (`FOUR`) or using the current mode (`TWO`). We use the Standard mode so write `FOUR`. Now we define the help string the field has to display. "Enter a value for A" will do perfectly in this example. So write: `"Enter a value for A."`. Now we have to give ChooseData and the ChooseDecompile. Don't ask me what it is. Because I don't know it. I only know that in 90% (probably even more) of the cases it has to be (for both ChooseData and ChooseDecompile) `MINUSONE`. So write `MINUSONE` and again `MINUSONE`. Or write `MINUSONE DUP`. Now we must say which value the field must have as initial value (when the inputs form is displayed). `%1` is a nice value, so lets take `%1`. After that we have to say which value if del is pressed. Let's take the same value. So `%1`. Or write `DUP`. If you are still with me, your code should look like this:

```
"A is =" THIRTEEN SIXTEEN

'DROPFALSE
BINT44
BINT14
BINT78
BINT9

ONE
{ ZERO }
FOUR
"Enter a value for A."
MINUSONE
MINUSONE
%1
DUP
```

We are almost there so don't give up now! We are in the ending stadium. We only have to a few thing anymore. Now we have to define how many labels and fields there are. We have one label ('A is=') and one field. So write `ONE DUP`. We also have to define a message handler that counts for the whole

input form. But we don't want a message handler so write again: `'DROPFALSE`. Now we only have to give our input form a title. Input form is an appropriate title. So let write that: `"Input Form"`. As last we have to write the command that will take all these arguments and make an input form with it. Write: `DoInputForm`. So we have:

```
"A is =" THIRTEEN SIXTEEN

'DROPFALSE
BINT44
BINT14
BINT78
BINT9

ONE
{ ZERO }
FOUR
"Enter a value for A."
MINUSONE
MINUSONE
%1
DUP

ONE DUP
'DROPFALSE
"Input Form"
DoInputForm
```

Before we see the result let's do still one thing. What if cancel is pressed? Well let's go to the label Entry. So write `NOTcase Entry`. Now compile the program and see what you hard work has brought up. The number you've entered is put on the stack so do whatever you want with it. Store it in a variable or do calculations with it. That was it for this lesson. As a finish I will give the whole source with comments.

```
*********************************************
*                 Labels                    *
*********************************************

"A is =" THIRTEEN SIXTEEN ( "A is =" will be showed at coordinate 13,16 )

*********************************************
*                 Fields                    *
*********************************************

'DROPFALSE              ( no message handler )

* Message handlers are things to control the input forms.
* E.g. you can add A..Z, change soft buttons (those buttons in
* the black buttons) and etc. with those

BINT44                  ( x-coordinate = 44 )
BINT14                  ( y-coordinate = 14 )
BINT78                  ( the lenght of the "black box" )
BINT9                   ( the height of the "black box" normally 9 )

* The next thing allows you to give the user a option to choose
* whether he has only a few options to choose from or he can enter
* anything.

ONE                     ( says it is a DATA FIELD )

* The next thing allows you to control what the users enters.
* here are only real numbers allowed.

{ ZERO }                ( only real numbers are allowed )
FOUR                    ( STD mode be used )

*************************************************************************
* exercice: change FOUR to TWO                                          *
* Than put your calculator to FIX 3 for example                         *
* notice the difference?                                                *
*************************************************************************
"Enter a value for A."                          ( help string )
MINUSONE                                        ( always MINUSONE )
MINUSONE                                        ( always MINUSONE )
%1                                              ( ResetValue )
DUP                     ( InitValue same as ResetValue in this example )

ONE DUP                 ( there is one field and one label )
'DROPFALSE              ( there is no message handler )
*the difference with this and the other message handler is this one is
*for whole the input form.
"Input Form"            ( title from the input form )
DoInputForm       ( make an input form with all the arguments above )
NOTcase                 ( what if cancel was pressed? )
Entry                   ( then return to label Entry )

* The value what the user entered is put on the stack
* lets do the value you entered * 2
* and display the result in a Messagebox

%2 %* DO>STR Ck&DoMsgBox

Entry                   ( then return label Entry )
```