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

Letter from the editor.

Yeah, the HP-39g+ and the HP-48gII are now also available in Belgium! I went this Saturday to a local store and there they were next to the HP-49g+. The price was a little more than expected (29 € more, if you are interested), but then again these are quality machines. I've read on the forum of hp-network.com that the ARM processor of the HP-49g+ resembles a lot like the Samsung. And for that processor there is already a C(++) compiler. When roccoHP (also called Jean-Marc Arsan) has the time he will have a try to convert the thing to work with the HP-49g+. "So what?" you might say. What in the name of the god do I care? Well since the HP-49g+ and HP-39g+ share the same processor. You should be able to program in C for your HP-39g+. Well maybe I'm talking just nonsense here, but anyway if it would be possible. That would be a great asset for the HP-39g+.

In this issue of the HP-39/40g Monthly Newsletter we have a chat with Alan Lark, there is a great upcoming program, we have a super-duper program of the month and we learn more about input forms.

Enjoy reading, Michaël, the editor

Program of the month.

This month ‘the program of the month’ is U-Lib 0.0. In one of the first issues of this monthly newsletter it was rated as most wanted program. And today it’s here…
But not in the form the author really wanted. He couldn’t include all the functions he intended to. So that’s the reason why it’s version 0.0. U-Lib stands for Universal Library and includes important functions that a HP-Basic programmer could find useful. What to think about a fast and easy to use getkey. A hexadecimal ←→ decimal converter and such other routines. There are even a bunch of games included with U-Lib. An important asset from this library is that it is open source. What means you can check how a function works or include your own functions. However there are some disadvantages about U-lib too. There is a bug in the greyscale viewer (when you turn your calc off) and the .wav player doesn’t allow other files than the one included (although with me the whole thing didn’t work).Besides that this library is still worth the download.

Specs:

System: HP-39(+)/40g.
Author: Noda
Download where: http://www.noda.online.fr
Site of the maker: http://www.hp-network.com/HP40G/Library/U-Lib%200.0%20-%20BETA.zip

Upcoming programs.

Not much only one program by me. **Battleship** is coming to your HP-39/40g. Hopefully, somewhere in the near future.

Interview with Alan Lark.

Interview	
Michaël	How old are you, and what of education do you have?
Alan	I'm 18yrs old, and currently studying a first year Science/Engineering double degree at the university of western Australia.
Michaël	Where do you live?
Alan	I live in Perth, Western Australia.
Michaël	What calculators do you own?
Alan	I own a 38g (which I accidentally found in an op-shop for $3.00☺) and a 39g.
Michaël	What was the first program you ever wrote?
Alan	The first program I ever wrote was, believe it or not, a song of mine called looking back. At that stage, I didn't know the frequencies for different notes, so I had to figure them out manually, which took ages. My first complicated program was the original bang game, which was similar to my game bang 2, but without the hills and with sticks for cannons.
Michaël	What do you use your calculator for the most?
Alan	Well, seeing as I'm not allowed to use a graphics as a first year at U.W.A., I haven't been using my calculator much at the moment, but when I do, it's usually for programming, or to check an answer to some horrible maths problem.
Michaël	Why did you start programming calculators and how did you learn it?
Alan	When I was in year 8, I remember seeing the older kids playing arkanoid and diamonds on the bus. When I finally got my 39g in yr 10, I read the programming section straight through. I basically taught myself how to program basic through practice. I recently started trying to learn Sys-Rpl.
Michaël	Are there any programmers you admire?
Alan	I admire anybody who takes the time to write programs and aplets, and allows others to use them and learn from them. In particular, I admire Noda, Lillian Pigallio, Jean-Yves and yourself who have made so many great aplets (especially games) available. I also feel the Jordi Hidalgo and Martin Lang have made great contributions to hp calculators.
Michaël	Are there any programmes you want to see on your calculator?
Alan	I'm sure many of the readers will agree with me when I say that I'd like to see more games available. Some different genres such as fighting games, rpgs and shooters would be great. I would also really like to see a mig aplet or library, and some sampled sound.
Michaël	Any future projects?
Alan	At the moment I'm not sure. I'd like to make some more games, and possibly a pc program designed specifically to create programs for maths routines without a knowledge of hp-basic. Maybe even a karaoke aplet.
Michaël	What do you see for the future of HP?
Alan	As long as people continue to support the calculators, I'm sure that the community will continue to grow. HP's new line of calculators is sure to attract the attention of budding programmers, and the backwards compatibility with the older models should hopefully keep current programmers interested. I don't think that many people realise just what powerful tools the calculators are. The simplicity of hp-basic and the power of Sys-Rpl and machine code, means that the calculators are extremely adaptable, and I believe that it is for this reason that there is such continued interest in the calculators.

Michaël's cosy programmers' corner.

This month is part 2 about input forms:

In this part we will learn how a user can select from a list of items that a programmer chooses for him. We are going to get strings from the user and we are going to use a check field. We have much to do, so let's start right away:

First we are going to start with getting the name of the user. For that we make a data field (like in the previous number) that accepts a string.
We start with defining the label. That means name of label, x-coordinate and y-coordinate. For the name `"Name:"` will do perfectly. X-coordinate will be `ONE` and y-coordinate `TEN`. Now for the field: there isn't a message handler so write: `'DROPFALSE`. X-coordinate of the field will be `BINT22` and the y-coordinate `BINT8`. Length of the field will be `BINT78` and height as normal so `BINT9`. It's a data field so write `ONE`. Now we have to say that we may enter strings so write `MINUSTWO`. Standard mode will be used so write `FOUR`. Finally, the help string: `"Enter your name."` describes perfectly what we want the user to enter so write that. There is no choose data and no choose decomp so write `MINUSONE DUP`. Lastly, what would we want to have as reset and init value? Let's just have an empty string so write `NULL$ DUP`. Your source should like this:

```
"Name:" ONE TEN

'DROPFALSE
BINT22
BINT8
BINT78
BINT9

ONE
MINUSTWO
FOUR
"Enter your name."
MINUSONE DUP
NULL$ DUP
```

Now we are going to make the list field. Scroll up till you see `"Name:" ONE TEN` again. Write under that: `"Choose toy:" ONE BINT24`. Now scroll again down where we were the other time (`NULL$ DUP`). Again for this new field there isn't a message handler, so write `'DROPFALSE`. X-coordinate for this field will be `BINT48` and y-coordinate `BINT22`. Length of this field will be `BINT52` and height as normal so `BINT9`. Data fields are defined as ONE, but this field isn't a data field, it's a list field so now we have to write `TWELVE`. You might wonder how I know all this stuff. Well Sys-Rpl for the HP-39/40g looks a lot like the HP-48g/49g/49g+ (geez ☺),so you can just read some tutorials for these calculators. But anyway back to the subject. Now we have to define the allowed types. With List Fields

that's always MINUSONE. That means there aren't arguments that we can enter. For decompiling we write BINT17. That says we will work with lists. Now we have to define our help string. "Choose the toy you want!" looks appropriate, so let's use that one. Now we have to define our chooseData and ChooseDecompile. You probably think now that it is MINUSONE DUP like usual, but you are thinking wrong. What? Yes, we have to write something different (that is those other 10% I was talking about in the last issue)! We have to define here what the user can select. So let's do that. To do that we must use lists and lists in lists. We make first our 1st list so write {. Now we have to define our first option. That's again a list with two elements in it: the first one is a string which the user will see when he chooses and the second is the element you want to have. After the input form is ended (when the user presses OK) whole the 'your option you have chosen' list is put on the stack. So write { "A car" "a car" }. Note that whole this list will be dropped when the user chose a car. Now as second choice I want a teddy bear. The same string as second element. So write { "A teddy bear" "a teddy bear" }. As a third option I want to see doll as a choice and again the same as second element. So write: { "A doll" "a doll" }. That's all for the options the user can choose so let's end our first list with }. For ChooseDecompile we write: SEVENTEEN. Now for our reset value and initial value we write: { "A car" "a car" } DUP. Your source should look like this:

```
"Name:" ONE TEN
"Choose Toy:" ONE BINT24
```

```
"Enter your name."
MINUSONE DUP
NULL$ DUP

'DROPFALSE
BINT48
BINT22
BINT52
BINT9
TWELVE
MINUSONE
BINT17
"Choose the toy you want!"
{
      { "A car" "a car" }
      { "A teddy bear" "a teddy bear" }
      { "A doll" "a doll" }
}
SEVENTEEN
{ "A car" "A car" }
DUP
```

Now scroll back up to our 'label section' as we want to display "cash?" on coordinates (1,38). Write: "Cash?" ONE BINT38. So the beginning of our source looks like this:

```
* labels

"Name:" ONE TEN
"Choose Toy:" ONE BINT24
"Cash?" ONE BINT38
```

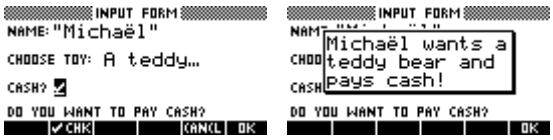
Now scroll back down. Again there isn't a message handler. So write: 'DROPFALSE. Display the check field on (22, 36). Write BINT22 BINT36. Length of a Check field is always SIX. And the height is as normal so NINE. Now we have to define it's a check field. To do that write: BINT32. Again there aren't elements which the user can enter so write MINUSONE. There is also no decompiling so write: DUP. For the help string: "Do you want to pay cash?". No ChooseData and no ChooseDecompile so write: MINUSONE DUP. Now for the reset and initial value we have to write a Boolean value (True or False). We don't want the option to be initially checked. (Incidentally, it's called a checkfield because the Americans call a 'tick' ✓ a check.) So write: FALSE DUP. But we are memory friendly so instead write: FalseFalse. With that last command we save 2.5 bytes! Now we are almost done with our input form. We only have to write how many fields and labels there are. The message handler for the whole input form (=none). The title. The command to make the input form. And the routine for when the user presses cancel. But we all seen that in the previous issue so I won't repeat that here again. Anyway the whole source + some extras

```
* labels

"Name:" ONE TEN
"Choose Toy:" ONE BINT24
"Cash?" ONE BINT38

* fields

'DROPFALSE                                              ( no message
handler )
BINT22
BINT8
BINT78
BINT9

ONE
MINUSTWO
FOUR
"Enter your name."
MINUSONE DUP
NULL$ DUP

'DROPFALSE
BINT48
BINT22
BINT52
BINT9
TWELVE
MINUSONE
BINT17
"Choose the toy you want!"
{
      { "A car" "a car" }
      { "A teddy bear" "a teddy bear" }
      { "A doll" "a doll" }
}
SEVENTEEN
{ "A car" "a car" }
DUP
```

```
'DROPFALSE
BINT24 BINT36
SIX

NINE
BINT32
MINUSONE
DUP
"Do you want to pay cash?"
MINUSONE DUP
FalseFalse

THREE DUP
'DROPFALSE
"Input Form"
DoInputForm
NOTcase
Entry

Cash!

TWO NTHCOMPDROP
" wants "
SWAP&$

&$

Cash@ FALSE EQITE " but doesn't pay cash." " and pays cash!"
&$

Ck&DoMsgBox

Entry
```

As you can see the Boolean value is put first on the stack, then the list and then the string. So the last field is the first on the stack and the first field is indeed the last.

Screenshots:



Next month we will learn about message handlers.