The HP-41 and Alan Turing
Where would we be without them?

Mark Power

This is a fuller version of the presentation given at the HPCC 2009 Mini-Conference and came about through two significant happenings in September 2009: the apology[1] from Gordon Brown, the Prime minister, for the way Alan Turing was treated in the 1950s and our celebration of the launch 30 years ago of the HP-41.

The HP-41

Introduced in 1979, the HP-41 caused quite a bit of debate in the computing press at the time as to whether it was just a fancy programmable calculator or a pocket computer. Its main perceived competitor at the time was the TI-59 over which it scored strongly by having many peripherals, plug-in modules and an alphanumeric LCD screen.

Whilst at school I had been lucky enough to go through several programmable calculators starting with a Commodore (with 25 program steps?), a TI-57, a Casio FX-502P and a Casio FX-702P. In 1983 I went off to study Computer Science and having found home computers of the time too expensive and too bulky, like the BBC model B at £399 plus cost of monitor and disk drive I turned to the flexibility and portability of the HP-41CV. At the time I didn't know that the CX was available, such was the rarity of information about these esoteric devices.

The big advantage of the HP-41 for me was that I could knock up programs for it very quickly and have them on hand to run whenever and wherever I wanted, rather than having to travel and quite often wait to use the ICL 2960 mainframe (first year students weren't allowed to use the PDP-11s).

My HP-41 notebook from the time starts with very simple programs for calculating definite integral's using Simpson's, Mid-point and trapezium methods, solving quadratic equations, performing linear regression, solving equations using Newton-Raphson and Secant methods, and so on.

Then I found a copy of Your Computer Magazine containing the game "Starseed Search" by Frank Wales and I realised that the HP-41 was much more of a general-purpose computer. I was hooked and my next creations ran to Mastermind, Biorhythms, Missile Command, a Text Editor and a Memo Holder. I almost got too carried away, until I started a course on the fundamentals of computer science and computational logic, which amongst other things introduced me to the work of Alan Turing.

[1] http://news.bbc.co.uk/1/hi/technology/8249792.stm

Alan Turing OBE FRS (1912-1954)

Described as one of the father's of modern computer science Alan Turing was responsible for many significant contributions to the world we live in. He was a brilliant mathematician and is most commonly recognised for his cryptanalysis of German naval codes during the Second World War. The importance of this work cannot be underestimated for without it, the outcome could have been very different.

His other works are less well known outside of the computer science and mathematical communities, but his seminal paper "On Computable Numbers, with an Application to the Entscheidungsproblem"[1] described the concept that we now call a Turing Machine: a simple device which using an algorithm can perform any mathematical computation.

Turing also created the first detailed design for a stored program computer and subsequently turned the theory into practice with the Manchester Mark 1. On the more theoretical side he worked on Artificial Intelligence and devised the Turing Test: where a person holds a natural language conversation with a hidden human and a hidden machine and if the person cannot tell the difference between the two, the machine is deemed to have passed.

Turing Machine

A Turing Machine comprises, four simple and easy to define components:

- An infinite tape containing symbols. A simple form of representation on the tape could be by using several "1" symbols to represent larger numbers e.g. 3 could be represented on the tape by the pattern "111" (if only positive integers are required) or "1111" (if you need to be able to handle 0 as well).
- A head, which the tape can be moved past one symbol at a time (either left or right) and which reads and write to the tape.
- An action table. This effectively is the algorithm or program that the Turing Machine will execute.
- A state register, which remembers what state the machine is in during the execution of the algorithm. States might be represented in the form Q1 … Qn.

The Action Table

This is the key to the Turing Machine and is a simple way in which algorithms can be documented and used to drive the computation. Actions are often described in the form of quadruples that comprise:

- For a particular state e.g. when the machine is in state Q1…
- If the tape contains symbol X…

[1] http://www.comlab.ox.ac.uk/activities/ieg/e-library/sources/tp2-ie.pdf

- Perform one of these actions: move the head one symbol to the left (L); move the head one symbol to the right (R), or write a new symbol (any other symbol) and finally
- Put the machine into state Qn

A very simple action table is given below. It assumes that the machine starts in state Q1, the tape contains a number represented as a string of "1" digits followed by a "blank" symbol (in this case "B").

In state	If tape contains	Action	Move to state
Q1	1	R	Q1
Q1	B	L	Q2
Q2	1	M	Q2
Q2	M	R	Q3
Q3	1	R	Q3
Q3	B	R	Q4
Q4	1	R	Q4
Q4	B	1	Q5
Q5	1	L	Q5
Q5	B	L	Q5
Q5	M	1	Q6
Q6	1	L	Q2

Tape

1	1	1	B

State Register

Q1

This algorithm hunts along the tape until it finds a "B" (for Blank). It then backs up one character and replaces the "1" with an "M" (for Marker). It then goes all the way to the right hand end of the tape until it finds the end or another blank and writes a "1". It then backs up all the way to the "M", replaces it with a "1" and moves left one symbol along the tape. If another "1" if found, it is replaced with "M" and the machine goes to the right hand end of the tape and adds another "1" and so on. Alternatively, if there are no more "1" symbols left the algorithm ends.

Following the algorithm through, the tape containing "111B" eventually contains "111B111". In other words, the number is duplicated.

This algorithm might seem somewhat useless, but similar more complex algorithms can be built up. These examples assume the tape represents positive integers only:

- Addition can be built by replacing the "B" in a tape with a "1" and replacing the right-hand most "1" with a "B" e.g. "111B1111" becomes "1111111B".
- Alternatively addition could be performed by copying both numbers to the end of the tape, using a marker to remember where the algorithm has got to, so that "111B111" becomes "111B111B111111".
- Subtraction can be performed by marking off the subtrahend digits from a minuend (literally taking away the digits like a child would with their fingers) so that "1111B11" becomes "11BBBBB".
- Multiplication can be performed by making multiple copies of one of the two numbers using two different markers to traverse the two numbers e.g. "111B11B111111".
- Integer division can be created by repeatedly taking away the divisor from the numerator (and leaving a remainder if you like).

You may want subtraction and division to be able to produce an answer of zero, so it could be preferable to use "1" for 0, "11" for 1, "111" for 2, etc. The programs presented later allow either option, but obviously the algorithms need to be adjusted accordingly.

Turing went on to describe a Universal Turing Machine as one that could run any Turing Machine. This takes us nicely back to the HP-41 as I enjoyed the concept so much that my first program to make use of the Extended Functions Module was a Universal Turing Machine. Because of the ability of EFM to hold and manipulate ASCII text, it made simulating the tape much easier than if I had tried to do it in main memory. Similarly EFM ASCII files are a convenient way of storing and searching through quadruples.

My program is quite simple but horrifically slow, thanks mainly to the speed of EFM operations, however as a first and only attempt at implementing a UTM it does have a certain charm. If you would really like to while away many hours waiting for your HP-41 to come up with an answer, feel free to give it a go. Otherwise I would recommend a quick search on the Internet where there are numerous JavaScript and Java applets that will show you the concept in nice fast graphical ways.

Did they change the world?

Without Alan Turing I believe the world would have been significantly different to the one we enjoy today, not only in terms of the outcome of WWII but the development of modern computing. It's impossible to imagine where computing would have been now if he had survived the laws of his time.

Without the HP-41 my journey into Computer Science would have been very different, my final year project would probably not have been an m-code debugger for one thing, and I'm sure that many others have been inspired to create software to solve their problems on this amazing little machine.

HP41 Universal Turing Machine

Mark Power

This was one of the first programs I wrote for the HP-41CV plus Extended Functions Module in 1984. It was more a learning exercise for me and is not intended as a lesson for the reader in how to write a fast, efficient or particularly user-friendly HP41 program. One of the lessons I learned from this was just how slow and inefficient Extended Functions are at text file handling. If you are interested in UTMs and want to improve the programs, please feel free. As they stand they will run on a bare CV + EFM, or a CX. No synthetics or other modules are required.

CRQ - CReate Quadruples

Use: Number of quadruples in X

Filename of quadruples in ALPHA

XEQ'CRQ'

At the prompt "Q?" enter the quadruples one at a time in the format:

Qx [SPACE] A [SPACE] B [SPACE] Qy and press after each R/S

After the last quadruple press R/S at the "Q?" prompt

CRT – Create Tape

Use: Requires SIZE 027 minimum

Maximum number of characters on the tape in X. Note that unlike Turing's infinite tape, you must specify a maximum size beyond which the tape cannot be expanded.

Filename of tape in ALPHA

XEQ'CRT'

At the prompt "T?" enter the characters on the tape, up to 24 at a time.

After the last tape entry press R/S at the "T?" prompt

XET – eXEcute Turing machine

Use: XEQ'XET'

Enter Tape filename R/S (defaults to ‘TAPE’)

Enter Quadruple filename R/S (no default)

Enter Blank Characters R/S (defaults to ‘B’)

Enter Start position of tape R/S (defaults to 0)

Enter Initial State R/S (defaults to 1 for Q1)

When finished the UTM displays “END”. Use RET or CTI to view results.

RET – REview Tape

Use: Displays the tape used by XET last, along with finishing tape position and final state.

CTI – Convert To Integer

Use: This converts tapes of “1”s into their integer equivalents using tape from last run of XET. Ensure flag 00 is set if you want “1” on the tape to represent zero. Leave flag 00 clear if you want “1” on the tape to represent one.

REQ – REview Quadruples

Use: Quadruple filename in ALPHA

XEQ’REQ’

Displays quadruples from the file in extended memory.

Registers Used

00 Tape filename (alpha)

01 Quadruple filename (alpha)

02 Tape position (numeric)

04 Current state (alpha)

05 Blank character (alpha)

06 Current tape character (alpha)

CRQ uses stack only

REQ uses stack only

CTI uses 00, tape filename from XET

RET uses 00, 02, 04 from XET

CRT uses 00-26 for temporary storage

Flags Used

System flags 25, 28 and 29 are used.

Flag 11 may be set which turns the calculator off immediately on finishing XET.

Quadruple File Format

Each record in the file is made up of four ASCII characters representing:

	Qx	A	B	Qy
ASCII Values	128-255	0-127	0-127	128-255

Tape File Format

Each character is held in a record on its own. Each character is in the range 0-127. If you use '1's and 'B's for the characters on the tape, as described in the Alan Turing article, then CX users can use the inbuilt text editor ED to edit the tape entering a single character on a line.

Programs

Note that REQ is standalone with it's own 'END', all the other programs reside in a single area with a single 'END' at the end of 'LBL E'.

```
LBL'REQ
      CLX  SEEKPTA  FIX 0  CF 29  SF 25
LBL 01  GETREC  FC? 25  GTO 02  ATOX  128  –
      -1  AROT  RDN  ATOX  128  –  ATOX  ATOX
      'Q'  ARCL T  'append space'  X<>Y  XTOA  'append space'
      X<>Y  XTOA  'append space Q'  ARCL Z  PROMPT  GTO 01
LBL 02  FIX 4  SF 29  CLST  END  {line 38}

LBL'CRQ  {line 01}
      5  x  7  +  LASTX  /  INT  CRFLAS  CLX  SEEKPTA  SF 28  CF 29
LBL 01  'Q?'  ASTO X  AON  PROMPT  AOFF  ASTO Y  X=Y?  GTO E
      1  XTOA  ANUM  128  +  XTOA
LBL 02  ATOX  47  X<=Y?  GTO 02
      1  AROT  ATOX  RDN  AROT  ATOX  ANUM  128  +  XTOA
LBL 03  ATOX  1  X≠Y?  GTO 03  APPREC  GTO 01  {line 50}

LBL'CRT  {line 51}
      2  x  7  +  LASTX  /  INT  SF 25  PURFL
      CF 25  CRFLAS  CLX  SEEKPTA
LBL 04  'T?'  ASTO X  AON  PROMPT  AOFF  ASTO Y  X=Y?  GTO E
```

```
        ALENG  STO 25  STO 26
LBL 05  ATOX  STO IND 25  DSE 25  GTO 05
LBL 06  CLA  RCL IND 26  XTOA  APPREC  DSE 26  GTO 06  GTO 04


LBL'XET  {line 90}
        'TAPE?'  AVIEW  'TAPE'  AON  STOP  ASTO 00
        'QUAD?'  PROMPT  ASTO 01
        'BLANK?'  AVIEW  'B'  STOP  ASTO 05  AOFF  CLX
        'POS?'  PROMPT  STO 02  CLA  ARCL 00  SEEKPTA  GETREC
        ASTO 06  1  '1ST STATE?'  PROMPT  CLA  128  +  XTOA  ASTO 04
LBL A  CLX  CLA  ARCL 01  SEEKPTA  CLA  ARCL 04  ARCL 06  POSFL
        X<0?  GTO E  GETREC  ATOX  ATOX  76  ATOX  ASTO 04
        X=Y?  GTO G  82  X=Y?  GTO H  RCL 02  CLA  ARCL 00
        SEEKPTA  GETREC  DELREC  CLA
        RCL Z  XTOA  INSREC  ASTO 06  GTO A  {line 156}
LBL G  RCL 02  1  -  STO 02  CLA  ARCL 00 X<0?  GTO d
        SEEKPTA  GETREC  ASTO 06  GTO A
LBL d  LASTX  ST+ 02  CLX  SEEKPTA
        CLA  ARCL 05  INSREC  ASTO 06  GTO A
LBL H  RCL 02  1  +  STO 02  CLA  ARCL 00  SF 25  SEEKPTA
        FC? 25  GTO c  GETREC  ASTO 06  GTO A
LBL c  CLA  ARCL 05  APPREC  ASTO 06  GTO A  {line 199}

LBL'CTI  {line 200}
        CLA  ARCL 00  CLX  SEEKPTA  SF 25
LBL 10  STO Z  GETREC  FC? 25  GTO 11  ATOX
        49  X≠Y?  GTO 11  RCL Z  1  +  GTO 10
LBL 11  RCL Z  1  FC? 00  CLX  -  STOP  FC? 25  GTO E  CLX  GTO 10

LBL'RET  {line 230}
        CLA  ARCL 00  CLX  SEEKPTA
LBL 08  CLA  12  STO 03
```

```
LBL 07  SF 25 ARCLREC FC? 25 GTO 09  DSE 03  GTO 07  PROMPT  GTO 08
LBL 09  PROMPT  'TAPE @ '  FIX 0  ARCL 02  PROMPT  CLA  ARCL 04
        ATOX 128  -  'STATE '  ARCL X  FIX 4  PROMPT
LBL E  FS? 11  OFF  SF 29  'END'  AVIEW  CLST  TONE 9  END  {line 271}
```

Worked Example – Addition

1) Enter the quadruples for the addition algorithm:

ALPHA ADD ALPHA

5

XEQ 'CRQ'

Q1 1 R Q1 R/S

Q1 B 1 Q2 R/S

Q2 1 R Q2 R/S

Q2 B L Q3 R/S

Q3 1 B Q4 R/S

R/S

2) Review the quadruples to ensure they were entered correctly:

ALPHA ADD ALPHA

XEQ 'REQ'

Press R/S after each quadruple

3) Create the tape with the two numbers to be added:

ALPHA TAPE ALPHA

12

XEQ 'CRT'

111B11 R/S

R/S

4) Run the Turing Machine:

XEQ 'XET'

R/S (defaults to TAPE)

ADD R/S

R/S (defaults to B)

R/S (defaults to tape position zero)

R/S (defaults to first state of 1)

Shows END when finished

5) Review the tape and end state:

XEQ 'RET'

11111BB R/S

Tape @ 5 R/S

State 4 R/S

END

6) Convert the tape to integers:

CF 00 (so that "11111" = 5)

XEQ 'CTI'

5 R/S

0 R/S

0 R/S

END

7) Try another calculation:

ALPHA TAPE ALPHA

PURFL

16

XEQ 'CRT'

11111B1111111 R/S

R/S

XEQ 'XET'

R/S (defaults to TAPE)

ADD R/S (name of quads)

R/S (B for blanks)	XEQ ‘CTI’
R/S (0 for tape starting position)	12 R/S
R/S (1 for 1st state)	0 R/S 0 R/S
END	END