Compiled by Dick Pountain

THE TEXAS TICKLED

Several months ago, a reader and professional TI59 user, Mr Weaver, sent me an article from the German magazine Chip. *In this article, the author reported the discovery of a method of doubling the execution speed of TI59 programs, which, as any patient user will know, is a very worthwhile improvement.*

Unfortunately, the original article contained errors and was written in a rather obscure fashion, as a result of which Mr Weaver could not make it work. My own attempts to decipher the piece were no more successful, but before giving up hope I sent the piece to Norman Horwood, who has previously written on TI affairs for this column (see PCW *3-1, 3-5).*

Mr Horwood's expertise in TI59 programming finally triumphed but only after hours of frustration and the lucky discovery of a vital clue which is omitted from the original piece!

Here is Mr Horwood's account of the method by which programs, written under certain special rules, will run at double speed on the TI58/59 calculations.

The trick depends on fooling the operating system into thinking it is running a module program, when, in fact, it is running your own program. As a result, the processor no longer interrogates the keyboard for user interrupts (eg, R/S, Pause, Trace) between every step executed, as is the case during normal execution.

This results in a 100 percent saving in processing time. The rules for writing programs in this Fast Mode are, however, very strict and rather limit possible program structures.
Dick Pountain

I shall demonstrate the technique by reference to two short programs. One I have published before in *PCW* (May 1980) called 'Auto-Folio' which demonstrates the main modifications required to allow the new system to work, and another, extracted from the *Chip* article and edited and modified to make it work at all, to demonstrate the speed advantage.

The system depends on entering the library module at an address where there is a merged code '31'. This is the Op Code for LRN. (Learn mode, used from the keyboard to enter program instructions). It must have taken our EEC friend many hours of bending over library downloads to find this serendipitous coding, even if he/she knew what he/she was doing. Incidentally, it only seems to work with the Master Library Module (ML 1), at least I have been unable to obtain the fast mode with the Maths Utility Module, even having found a 31 coding in programs 6 and 7 at locations 108 and 454 respectively. The initialising procedure seems to be working but the speed is unchanged.

Apparently, with the ML 1 in place, the pseudo call via an indirect SBR op to the 31 code location, 250, in program number 2 makes the processor think it is in the module when actually processing an external user program. This has the characteristic of ignoring the keyboard during processing, just as when the program enters the module, so that after every operation the pointer no longer tests the R/S, Pause, Trace as in normal processing. This is where the time is saved. It does mean that special steps have to be taken to allow data to be entered during a run, since when a stop instruction is encountered in one of these special programs, leaving data displayed, the attempted addition of different figures merely adds to the displayed figures in decimal form; ie, with the display showing 123, keying in 456 alters the display to 123.456 and not 456, as if an operation had not been completed in the program.

The method of gaining access to the fast mode is via the following key strokes: 000 start point — 2,4,0,STO 00, 2nd, Pgm, 02, SBR, 2nd, Ind 00, EE, INV, EE, R/S. This occupies 14 program steps and is entered in normal learn mode, exiting in the normal manner, LRN. With this program in memory press RST, R/S. The display blanks for a fraction of a second then displays zero. The machine is now in fast mode, and almost anything will either crash or destroy the new mode, reverting to standard. A total reset has taken place and any contents of memory, data or program instructions has been cleared. It is possible to enter the learn mode, but the display has no obvious relevance (1803 52) except that the 52 is the code for 'EE'. Single stepping in this condition produces an incremental increase to the four figure number with the Op Code following the listing INV, EE, R/S. to step No. 1806, any further stepping leaving the learn mode for ever; nothing can be done to feed another program by any method. The Reset key is the only active key, and this reverts the system to standard.

Any operational program now must be read into memory through the card reader. If there has to be a change of partitioning as in the 'Auto Folio' example, this must be part of the initialising process contained in the magnetic card listing. After reading the program card/s, each bank number will be displayed as normal, and it is possible to employ protected cards. From the listings reproduced it can be seen that both the Fast Mode and the Operational programs start at 000. This is for convenience, but it should be remembered that the Fast Mode (FM) program must end with a legal R/S (ie, preceded by either CLR or EE, INV, EE) or the processing will start immediately a card has been read and continue unstoppable until a legal R/S is encountered in the program. Once the required program is safely in memory it is possible to start processing at any location by calling a subroutine and a three digit address. Indirect addressing is allowed, but the register must be pre-loaded with the address location when the cards are loaded into memory. It is *not* possible to manipulate memory data in Fast Mode from the keyboard.

In our examples both programs are started by SBR 000. It is now possible to edit the program by entering Learn without affecting the Fast processing, but if you list the coding and allow the printing to go beyond the program length, this will exit from FM. However, if the listing is stopped by pressing R/S before reaching the end of your program, processing starts at that point and the pointer ends up where the remnant of the program takes it. The system remains in FM as long as the pointer remains in play.

As TI 58/59 users will be aware, in order to produce hard copy of data input, especially where large volumes of results have to be accumulated and analysed from random records, one is forced to use the Trace mode with the printer. This is fine for simple arithmetical entries where each data entry is accompanied on the same line by the operation following (+ — x +). However, when data has to be accumulated into analysed registers the Trace system prints each data value twice and the data register separately, taking three lines for each entry. Auto-Folio prints the data value and the register number together on the same line, and simultaneously sums the amount into that register, so that at the end of a session INV 2nd List produces a full analysis of the data. The partitioning used in the example makes 99 registers available.

The procedure is simple. With the original, standard speed version the value is entered with + or — (+/— key) and 'A' pressed — the display clears — then the desired analysis register number is keyed in (1 to 99) and R/S pressed. After about four seconds the value entered and the data register number are printed on one line. In Fast Mode, the keys A,B,C,D,E, are no longer available so the program has been looped at step 061 to return the pointer to the start position. This allows the R/S key to be used for both entries. The new procedure is therefore — enter the value — press R/S — the display clears — enter the register number as before and press R/S. This time it only takes two seconds for the printout to complete.

I will not dwell on the HIR Op codes here as this aspect has been fully covered before, except to note that while the FM initiation program clears everything out of memory, it leaves the pending operation stack (HIR) and the T register alone.

The conversion from normal subroutine calls which are no longer legal has been obtained in this case at steps 027 to 034 combined with 072 to 075, and 039 to 046 combined with 076 to 079. Here, the GTO instruction has been combined with flag setting, testing and unsetting. This gives reasonable flexibility with safety, and I think, is better than GTO with variable indirect addressing, as long as there is enough program space and you don't run out of flags. I am wary of introducing non-data numerical instructions in the middle of computation, as this usually means that the computed data must be saved somewhere and brought back safely — very risky.

The second example is lifted straight from the *Chip* article, duly edited to make it work. This is a simple iterative loop which calculates factorials, and is a neat method for comparing processing times. The procedure is to load the FM program, run it by RST, R/S, and load the Factorial program card, initialising by pressing SBR 000. To obtain the factorial of any number up to the maximum for the machine (69), enter the number and press R/S. All the calculation is done in the loop 006 to 012, the rest of the instructions format the printout. In normal mode the time for the calculation of 69 is approximately 29 seconds; in Fast Mode the time is literally halved to 14.5 seconds. The only alteration from standard required for this program to operate in FM is the CLR,R/S section at 000. Alternatively, EE, INV, EE,R/S could have been used to leave the result in the display as well as print it. If you change this, don't forget to change the loop address to DSZ, 00, 00, 08. to allow for the two extra instructions. Do not be tempted to use RST instead of GTO, 00, 00. at the end as this will make the system revert to normal after the first run.

Summarising the special rules for using the Fast Mode: almost any key will start processing; calculations cannot be carried out on the keyboard; CLR, CE, EE, INV EE work; RST and CP keys cause exit from FM; RST in a program also causes exit from FM; no SBRs can be called in a program; no Module programs can be called; no functions using firmware (P/R, DMS, E+, etc) can be used; no labels; flag 8 error detection not available; RTN will not work; always start a new program with SBR nnn or SBR Ind NN; always precede R/S in a program with either CLR or EE INV EE or with a pending calculation where the entry of a digit from the keyboard will add to the display and not replace it (with the last method, in order to be able to enter a new value, it is necessary to press CLR or CE first to remove the displayed figure if necessary); all program cards must be recorded in 'Power-up' partition as the FM program re-sets to this.

All in all this fast mode looks as though it can be quite useful, certainly where long loopy routines are required. The preparation of tabulated results with significant amounts of printing will be attractive applications as the speed of printing is also greatly increased, but beware of overcooking the printhead as I have during the experimentation on this method. I am certain that there are more tricks and short cuts yet to be discovered, and I hope *PCW* will find space to print them when someone finds them. Finally, would some bright person please tell me why this works?

N Horwood

Initialise fast mode

```
000  02  2
001  04  4
002  00  0
003  42  STO
004  00  00
005  36  PGM
006  02  02
007  71  SBR
008  40  IND
009  00  00
010  25  CLR
011  91  R/S
```

Auto-folio

```
000  01  1
001  00  0
002  69  OP
003  17  17
004  25  CLR
005  91  R/S
006  82  HIR
007  03  03
008  07  7
009  32  X:T
010  25  CLR
011  91  R/S
012  42  STO
013  00  00
014  55  ÷
015  01  1
016  00  0
017  54  )
018  75  -
019  59  INT
020  82  HIR
021  04  04
022  54  )
023  65  ×
024  01  1
025  00  0
026  54  )
027  86  STF
028  01  01
029  61  GTO
030  00  00
031  64  64
032  22  INV
033  86  STF
034  01  01
035  85  +
036  53  (
037  82  HIR
038  14  14
039  86  STF
040  02  02
041  61  GTO
042  00  00
043  64  64
044  22  INV
045  86  STF
046  02  02
047  54  )
048  65  ×
049  01  1
050  00  0
051  00  0
052  95  =
053  69  OP
054  04  04
055  82  HIR
056  13  13
057  74  SM*
058  00  00
059  69  OP
060  06  06
061  61  GTO
062  00  00
063  04  04
064  22  INV
065  77  GE
066  00  00
067  70  70
068  85  +
069  02  2
070  85  +
071  01  1
072  87  IFF
073  01  01
074  00  00
075  32  32
076  87  IFF
077  02  02
078  00  00
079  44  44
```

Factorial

```
000  25  CLR
001  91  R/S
002  42  STO
003  00  00
004  42  STO
005  01  01
006  43  RCL
007  00  00
008  65  ×
009  97  DSZ
010  00  00
011  00  00
012  06  06
013  42  STO
014  00  00
015  07  7
016  03  3
017  00  0
018  00  0
019  00  0
020  00  0
021  00  0
022  00  0
023  69  OP
024  04  04
025  43  RCL
026  01  01
027  69  OP
028  06  06
029  43  RCL
030  00  00
031  99  PRT
032  61  GTO
033  00  00
034  00  00
```