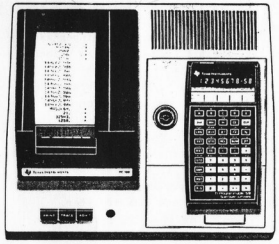
Claes Bahlbles PR ex

PROGRAM BITEN

Komplement till instruktionsboken för TI 58, TI 58C, TI 59



Innehåll

Bilaga till Programbiten 82-1
Pris för icke-medlemmar 15 kr

FÖRORD

Detta komplement till instruktionsboken för TI-58/58C/59 har tillkommit för att nya medlemmar inte skall behöva känna sig främmande för artiklar i Programbiten som förutsätter kännedom om innehållet i tidigare nummer.

När vi skrivit det har vi emellertid i viss mån försökt utforma det så att det skall kunna användas som "uppslagsverk" med hänvisning till olika utförligare artiklar i Programbiten.

Användningen av det här komplementet till instruktionsboken förutsätter givetvis att läsaren känner till innehållet i den ordinarie instruktionsboken.

Stockholm januari 1982

Lars Hedlund
Björn Gustavsson

SYNTETISK PROGRAMMERING

Som bekant lagras en instruktion i programminnet som en tvåsiffrig kod. Tabeller över dessa koder finns i instruktionsboken på s. V-49 och -50. Man finner helt naturligt att funktionerna 2nd, LRN, SST, Ins, BST och Del saknar koder. Om de funnits skulle de ha varit 21 eller 26, 31, 41, 46, 51 resp. 56. Vidare saknas kod 82. Dessa koder, som alltså inte kan matas in i programminnet med en enkel tangenttryckning (och inte ens med flera "kopplade" tangenter) kan emellertid programmeras "syntetiskt" med hjälp av framåt- och bakåtstegning samt borttagning (Del) om man utnyttjar att STO m fl tangenter gör att ett följande tvåsiffrigt tal lagras i ett programsteg. "21" kan t ex programmeras "STO 21 BST BST Del SST". (Vid listning med skrivare skrivs instruktionen 21 ut som "2ND".) - Koden 31 (LRN) i ett program medför att programmet stannar och räknaren går över i programmeringsläge. Beträffande koden 82, se ett följande avsnitt om HIR.

FLAGGOR

TI 58/59 har tio flaggor med nummer 0 - 9 men ej fler. Genom experiment med syntetisk programmering kan man tro sig "upptäcka" fler flaggor men vid närmare kontroll visar sig dessa vara de samma som de tio ursprungliga.

DSZ

I instruktionsboken s. V-63 står att Dsz-funktionen endast opererar på registren 00 - 09. Detta är inte sant, utan alla register utom 40 kan användas. ("Dsz 40" läses nämligen som "Dsz Ind".) Såväl registernumret som den efterföljande hoppadressen måste då emellertid programmeras syntetiskt enligt ovan, eller, i gynnsamma fall, på något enklare sätt. Exempel: "Dsz 29 320" kodas "Dsz CP (=29) 3 2ndCLR (=20). "Dsz 41 021" får t ex programmeras "STO 41 STO 21 BST BST BST (gå till första STO) Dsz (lägges över första STO) SST (gå till andra STO) 0 (läggs över andra STO) SST (för att komma till nästa fria steg)".

DSZ nn BST

En motsvarighet till "Op 30" - "Op 39" (minskning av ett registerinnehåll med en enhet) kan man få med "Dsz nn BST" där både registernumret nn och BST (kod 51) får programmeras syntetiskt. Obs dock att Dsz minskar absolutbeloppet av registerinnehållet ned till noll, ej förbi noll!

"MJUK" OCH "HÅRD" DISPLAY

En display är "mjuk" så länge den kan raderas med CE (dvs under inmatning av siffror) och blir "hård" då man trycker ner en operationstangent (kan då ej raderas med CE).

HIR

AOS (algebraiskt operations system) innebär som bekant att uttryck kan slås in som de är skrivna. För att möjliggöra detta måste räknaren lagra vissa tal tillfälligt. Om t ex 2 + 3 x 4 matas in måste 2 och 3 sparas någonstans i väntan på att beräkningen kan fullföljas. För att lagra dessa s k vilande operationer används 8 stycken specialregister, nämligen HIR-register (Hierarchical Internal Registers). Dessa register betecknas H_1 till H_8 (även HIR_1 till HIR_8).

I exemplet ovan lagras 2 i H_1 och 3 i H_2. Det är faktiskt möjligt att komma åt dessa register direkt från program eller tangentbordet och använda dem som extraregister.

Instruktionen HIR yn används för att komma åt ett HIR-register, där n är numret på HIR-registret och y betecknar den operation som skall utföras enligt följande tabell:

y	Operation
0	lagra i H_n (STO)
1	återkalla H_n (RCL)
(2)	okänt
3	summera till H_n (SUM)
4	multiplicera till H_n (Prd)
5	subtraktion i H_n (INV SUM)
6 (7,8,9)	division i H_n (INV Prd)

Instruktionen HIR har koden 82. Tyvärr finns det ingen tangent med den koden, utan koden måste programmeras syntetiskt.

Exempel: 18 HIR 04 lagrar 18 i H_4. 3 HIR 64 delar innehållet i H_4 med 3. 7 HIR 34 lägger till 7 till H_4. HIR 14 återkallar innehållet i H_4 och ger alltså i detta fall 13.

En viktig begränsning vid användning av HIR är följande: Om tiopotensnotation eller teknisk notation inte är inställt tas absolutbeloppet av x-registrets 10-exponent vid användning av HIR 3n till HIR 9n.

Det betyder att x kommer att multipliceras med en potens av 10 om absolutbeloppet av x är mindre än 1. Exempel: 0.035 HIR 35 resulterar i att $0.035 = 3.5 \cdot 10^{-2}$ ändras till $3.5 \cdot 10^{2} = 350$, som sedan summeras till H_5. Om tiopotensnotation är inställd händer inte detta.

Användning av HIR som extraregister

Räknaren använder HIR-register internt vid beräkning av vissa funktioner, samt som skrivregister. Det är viktigt att kontrollera att ett HIR-register inte används för två saker samtidigt. Så här använder TI-58/58C/59 HIR-registren:

Operation	Använder HIR-register
Op 01	H_5
Op 02	H_6
Op 03	H_7
Op 04	H_8
Op 11	Första 2 tillgängliga HIR
Op 12	H_8 och första 3 tillgängliga HIR
Op 13	Första tillgängliga HIR
Op 14	H_8 och första 3 tillgängliga HIR
Op 15	H_8 och första 3 tillgängliga HIR
P/R	Första tillgängliga HIR samt H_7 och H_8
INV P/R	Första 2 tillgängliga HIR samt H_7 och H_8 (H_7 används inte på TI-58C)
$\Sigma+$	H_7 och H_8
INV $\Sigma+$	H_7 och H_8
D.MS	Första 2 tillgängliga HIR och H_8
INV D.MS	Första 2 tillgängliga HIR och H_8
$\bar{x}$	Första tillgängliga HIR
INV $\bar{x}$	Första 2 tillgängliga HIR och H_8

Observera också att HIR-register används av AOS för att lagra vilande operationer. Detta gör för det mesta H_1 och H_2 oanvändbara som extraregister.

Varken CLR, CMs eller någon annan instruktion nollställer något HIR-register. Ett enstaka HIR-register kan förstås nollställas med 0 HIR On. Om skrivare är ansluten kan alla nollställas med sekvensen CLR D.MS HIR 03 HIR 04 Op 00 (utan skrivare nollställs H_1 till H_4; Op 00 kan då uteslutas).

Användning av HIR som skrivregister

Skrivregistren kan laddas direkt med hjälp av HIR-instruktionen. Som framgick av tabellen över hur HIR-registren används, motsvarar H_5 skrivregister 1, H_6 skrivregister 2 etc.

Op 01 kan alltså ersättas med HIR 05. Men nu visar det sig emellertid att skrivkoderna inte kan skrivas på normalt sätt. Innehållet i ett skrivregister lagras på formen $x.xxxxxxxxxxx \cdot 10^{n}$, där siffran före decimaltecknet är skild från noll. De 3 första siffrorna ignoreras alltid vid utskrift. Skrivkoden är alltid de sista 10 siffrorna.

Om vi t ex utför 131415 HIR 05 för att skriva "ABC", lagras detta som $1.314150000000 \cdot 10^5$ och eftersom de 3 första siffrorna ignoreras blir skrivkoden 41 50 00 00 00 vilket ger utskriften "Ux ". För att få den riktiga utskriften måste vi sätta 3 signifikanta siffror före skrivkoden. Om vi provar 100131415 HIR 05, blir skrivkoden 13 14 15 00 00 och utskriften blir "ABC ". Decimaltecknets placering har ingen betydelse; 10.0131415 hade t ex gått lika bra.

Vid användning av Op 01-04 lagras siffrorna i skrivregistret på en icke-normaliserad form: $0.00xxxxxxxxxx \cdot 10^0$, dvs siffran före decimaltecknet är noll. Vid utskrift ignoreras de 3 första nollorna. (Observera: Det finns ingen möjlighet att åstadkomma en liknande lagring med HIR-instruktionen, utan 3 signifikanta siffror måste inleda skrivkoden.)

Exempel: Om man trycker 131415 Op 01 lagras skrivkoden i H_5 på följande sätt: $0.0000001314150 \cdot 10^0$, där de tre första nollorna ignoreras vid utskrift och ger " ABC".

Vilka är fördelarna med att använda HIR-instruktionen istället för Op 01-04 för att lagra skrivkod?

För det första störs Op-laddning av både fast decimalplacering och tiopotensnotation. HIR-laddning har inga sådana nackdelar.

För det andra kan delar av ett skrivregister ändras om HIR-laddning används. Exempel: Antag att X1, X2, X3 ... skall skrivas ut i högerkanten. Skrivkoden kan laddas med sekvensen 4402 + 1 EE 12 = HIR 08 (ev. CLR för att ta bort tiopotensnotation). Detta ger utskriften X1. För att få X2 exekveras 1 HIR 38; likadant görs för att få X3 etc. (Den första laddningen kan göras kortare om skrivkoden finns lagrad i ett register. Sekvensen blir då RCL nn HIR 08.)

Användning av HIR från tangentbordet

Ibland kan det vara bra att kunna utföra HIR-instruktionen från tangentbordet. Det enklaste sättet är att placera en serie av kod 82 någonstans i programminnet där det finns steg över. Om stegräknaren sedan ställs på ett programsteg där en sådan kod finns, är det bara att trycka SST, för att exekvera HIR-instruktionen. Detta följs sedan av två siffror från tangentbordet, för att mata in operationskoden. Exempel: För att exekvera HIR 11, tryck SST 11.

Artiklar om HIR i Programbiten:

SAMBAND PROGRAMSTEG - DATAREGISTER

På TI-58/58C/59 kan samma minne användas för program och data (självfallet ej samtidigt). Det är därför möjligt att lagra data i ett register och ändra uppdelning, varefter registerinnehållet blivit programkod.

Prova t ex följande: Mata in 9208.821176, tryck STO 59 (STO 29 på TI-58/58C). Ändra uppdelning med 5 Op 17 (2 Op 17). Vid steg 483 (243) finns följande program lagrat:
Lbl A HIR 08 RTN.

R_{59} motsvarar stegen 480-487 på TI-59 och stegen 000-007 på TI-58/58C. I tabellerna nedan visas sambandet mellan programsteg och dataregister.

TI-59							
000-007	R119	352-359	R75	720-727	R29	112-119	R45
008-015	R118	360-367	R74	728-735	R28	120-127	R44
016-023	R117	368-375	R73	736-743	R27	128-135	R43
024-031	R116	376-383	R72	744-751	R26	136-143	R42
032-039	R115	384-391	R71	752-759	R25	144-151	R41
040-047	R114	392-399	R70	760-767	R24	152-159	R40
048-055	R113	400-407	R69	768-775	R23	160-167	R39
056-063	R112	408-415	R68	776-783	R22	168-175	R38
064-071	R111	416-423	R67	784-791	R21	176-183	R37
072-079	R110	424-431	R66	792-799	R20	184-191	R36
080-087	R109	432-439	R65	800-807	R19	192-199	R35
088-095	R108	440-447	R64	808-815	R18	200-207	R34
096-103	R107	448-455	R63	816-823	R17	208-215	R33
104-111	R106	456-463	R62	824-831	R16	216-223	R32
112-119	R105	464-471	R61	832-839	R15	224-231	R31
120-127	R104	472-479	R60	840-847	R14	232-239	R30
128-135	R103	480-487	R59	848-855	R13	240-247	R29
136-143	R102	488-495	R58	856-863	R12	248-255	R28
144-151	R101	496-503	R57	864-871	R11	256-263	R27
152-159	R100	504-511	R56	872-879	R10	264-271	R26
160-167	R99	512-519	R55	880-887	R09	272-279	R25
168-175	R98	520-527	R54	888-895	R08	280-287	R24
176-183	R97	528-535	R53	896-903	R07	288-295	R23
184-191	R96	536-543	R52	904-911	R06	296-303	R22
192-199	R95	544-551	R51	912-919	R05	304-311	R21
200-207	R94	552-559	R50	920-927	R04	312-319	R20
208-215	R93	560-567	R49	928-935	R03	320-327	R19
216-223	R92	568-575	R48	936-943	R02	328-335	R18
224-231	R91	576-583	R47	944-951	R01	336-343	R17
232-239	R90	584-591	R46	952-959	R00	344-351	R16
240-247	R89	592-599	R45			352-359	R15
248-255	R88	600-607	R44	TI-58		360-367	R14
256-263	R87	608-615	R43	000-007	R59	368-375	R13
264-271	R86	616-623	R42	008-015	R58	376-383	R12
272-279	R85	624-631	R41	016-023	R57	384-391	R11
280-287	R84	632-639	R40	024-031	R56	392-399	R10
288-295	R83	640-647	R39	032-039	R55	400-407	R09
296-303	R82	648-655	R38	040-047	R54	408-415	R08
304-311	R81	656-663	R37	048-055	R53	416-423	R07
312-319	R80	664-671	R36	056-063	R52	424-431	R06
320-327	R79	672-679	R35	064-071	R51	432-439	R05
328-335	R78	680-687	R34	072-079	R50	440-447	R04
336-343	R77	688-695	R33	080-087	R49	448-455	R03
344-351	R76	696-703	R32	088-095	R48	456-463	R02
		704-711	R31	096-103	R47	464-471	R01
		712-719	R30	104-111	R46	472-479	R00

För att förstå hur detta samband kan utnyttjas praktiskt, måste vi titta på hur räknaren lagrar tal internt.

Alla tal är lagrade i tiopotensnotation, oavsett hur de visas i displayen, dvs på formen $m \cdot 10^n$, där m är ett 13-siffrigt tal ($1 \leq m < 10$) och n är exponenten. Genom att lagra 2 siffror i varje programsteg går talet in i 8 programsteg. 6 1/2 programsteg används för m, 2 gånger 1/2 programsteg används för exponenten och 1/2 programsteg används för tecknen på m och n. Dessa tecken lagras enligt:

Siffra	Betyder
0	$m \cdot 10^n$
2	$-m \cdot 10^n$
4	$m \cdot 10^{-n}$
6	$-m \cdot 10^{-n}$
8	Overflow (dvs 9.9999999 99 blinkar vid återkallning)

Exempel: Talet $-2471.7176 = -2.4717176 \cdot 10^3$ lagras i R_0 på TI-59 på följande sätt:

959 24
958 71
957 71
956 76
955 00
954 00
953 00
952 32

m är alltså lagrad i steg 953-959. (Det finns ett tänkt decimaltecken mellan tiotals- och entalssiffran i steg 959.) Exponenten (= 03) lagras i entalssiffran av steg 953 och i tiotalssiffran av steg 952. Tecknen är lagrade som en 2:a i entalssiffran av steg 952.

Innehållet bildar samtidigt ett program, nämligen (från steg 956): Lbl SBR SBR CE.

Det är alltså möjligt att låta ett program skapa nya program, som sedan kan användas som subrutiner.

Här skall vi bara titta på en av möjligheterna med den här tekniken. Programmen nedan ordnar ett indirekt hopp till en label vars kod finns i displayen vid anropet.

TI-58/58C		TI-59	
000	76 LBL	000	76 LBL
001	11 A	001	11 A
002	32 X↕T	002	32 X↕T
003	03 3	003	06 6
004	69 OP	004	69 OP
005	17 17	005	17 17
006	32 X↕T	006	32 X↕T
007	85 +	007	85 +
008	93 .	008	93 .
009	06 6	009	06 6
010	01 1	010	01 1
011	95 =	011	95 =
012	42 STO	012	42 STO
013	29 29	013	59 59
014	02 2	014	05 5
015	69 OP	015	69 OP
016	17 17	016	17 17
017	61 GTO	017	61 GTO
018	02 02	018	04 04
019	46 46	019	86 86

Vid körning lagras GTO N (där N är labeln som matades in) vid steg 486 (246 på TI-58/58C).

Artiklar om samband programsteg - dataregister i Programbiten:

FAST MODE

Fast mode är ett tillstånd när program körs med dubbel hastighet. Det upptäcktes i början av 1980 av västtysken Martin Neef. Nu används metoden till alla program som måste köras snabbt.

Fast mode uppnås genom att sekvensen Pgm 02 SBR 239 9 0 placeras i programminnet med början vid steg 005 och exekveras (ML-modulen måste vara isatt). Det resulterar i att hela minnet raderas. Det är då möjligt att mata in ett program direkt från tangentbordet eller läsa in ett magnetkort.

Ett program som skall köras i fast mode kan se ut som på nästa sida. Steg 000-015 är initieringsrutinen som ser likadan ut i de flesta fast mode-program. Därefter kommer det egna programmet, som i det här fallet är en rutin för beräkning av fakultet.

```
000  00  0      012  22  INV     024  00  00
001  00  0      013  58  FIX     025  00  00
002  00  0      014  22  INV     026  20  20
003  76  LBL    015  57  ENG     027  01  1
004  11  A      016  25  CLR     028  95  =
005  36  PGM    017  91  R/S     029  99  PRT
006  02  02     018  42  STD     030  91  R/S
007  71  SBR    019  00  00      031  61  GTD
008  02  02     020  43  RCL     032  00  00
009  39  39     021  00  00      033  18  18
010  09  9      022  65  ×
011  00  0      023  97  DSZ
```

Vi återkommer till programmet, men först en uppräkning av de beräsningar som finns förknippade med fast mode.

1. Vid övergång till fast mode raderas hela programminnet och dataregistren (ej HIR-registren).
2. Uppdelningen ställs alltid om till 479.59 (6 Op 17) på TI-59 (till 239.29 (3 Op 17) på TI-58) vid övergång till fast mode. Det är inte möjligt att gå in i fast mode på TI-58C (se dock slutet av den här beskrivningen).
3. Vid övergång till fast mode ställs den fasta decimalplaceringen om till Fix 0.
4. Program kan matas in eller ändras direkt från tangentbordet. SST, BST, Ins och Del kan användas obehindrat. Kopplade koder får inte matas in, eftersom räknaren då går ur fast mode (exempelvis 72=ST*, 92=RTN).
5. Tal får matas in som vanligt med tangenterna 0-9, +/-, decimalpunkt, CE, CLR, EE.
6. Användning av alla tangenter som efterlämnar en hård display startar körning av programmet i programminnet. Detta kan orsaka att räknaren tas ur fast mode t ex om det inte finns något program. Ett undantag till denna regel är Prt, som kan användas obehindrat.
7. RST använt från tangentbordet när programkörning ej pågår tar bort fast mode. RST använt i ett program tar bort fast mode och kan orsaka en "krasch", efter vilken det bara är att slå av räknaren.
8. R/S kan användas för att starta ett program i fast mode. SBR nnn får användas från tangentbordet för att starta ett program vid steg nnn.
9. Enda sättet att utifrån stoppa ett program som körs i fast mode är att slå av räknaren.
10. Ett R/S i ett program kommer inte att fungera om inte displayen är mjuk eller om inte R/S är omedelbart förbigånget av Prt eller Pause. (Prt går bra även utan skrivare.)
11. RTN får aldrig användas.
12. Inga som helst subrutinanrop kan användas, dvs ej heller funktioner som D.MS, P/R eller statistiska funktioner (som använder den fasta programvaran) eller bibliotekspsrogram.
13. Inga labels får användas. Alla hoppadresser måste vara absoluta.

Låt oss nu titta på programmet ovan. Det matas in på vanligt sätt. Det skall sedan spelas in på magnetkort före provkörning, eftersom programmet raderas vid övergång till fast mode.

Programmet körs genom att A trycks ned. Programminnet raderas samtidigt som räknaren övergår till fast mode. Nu kan magnetkortet läsas in igen. Direkt efter läsningen startar exekvering av programmet. Följande sekvens exekveras då: 9 0 INV Fix INV Eng CLR R/S, och stannar med en nolla i displayen. Nu kan ett heltal matas in och R/S tryckas. Fakultet för talet beräknas.

Följande är viktigt att observera när man gör program i fast mode:

Programstegen 005-011 skall se ut precis som i det här programmet. Stegen 000-004 får innehålla vad som helst, fast vi rekommenderar att steg 003-004 innehåller Lbl A, så att initiering till fast mode kan göras bara genom att trycka A.

Stegen 012-015 får ändras, men observera att de kommer att utföras två gånger. R/S på något av stegen är alltså klart olämpligt. Vidare ställs Fix 0 in automatiskt när man går över till fast mode - om Fix 0 inte är önskvärt bör därför INV Fix stå kvar.

Om ett program skall köras med annan uppdelning än normaluppdelning, måste programmet själv ändra uppdelningen.

Om ett program inte får plats på en kortsida måste den del av programmet som ligger i block 1 läsa in resterande kort. Detta görs lämpligen som i det här programmet:

```
000  00  0      008  02  02      016  02  2
001  00  0      009  39  39      017  91  R/S
002  00  0      010  09  9       018  03  3
003  76  LBL    011  00  0       019  91  R/S
004  11  A      012  22  INV     020  04  4
005  36  PGM    013  58  FIX     021  91  R/S
006  02  02     014  22  INV
007  71  SBR    015  57  ENG
```

Det egentliga programmet skall börja på steg 022. Proceduren för körning är följande: Läs in block 1. Tryck A. Läs in block 1 igen. 2 visas - läs in block 2. 3 visas - läs in block 3. 4 visas - läs in block 4.

Om man har färre än 4 block tas helt enkelt onödiga instruktioner bort.

Till sist: Lägg märke till begränsningarna ovan - särskilt att subrutiner inte får användas.

Nyligen (hösten 1981) har två nya metoder för att komma in i fast mode upptäckts. Dessa kan användas på TI-58C, och programminnet raderas inte vid övergång till fast mode (vilket också gör fast mode praktiskt möjligt på TI-58).

Artiklar om fast mode i Programbiten:

PB 80-3 s 14: Lars Hedlund, Djupdykning i TI-59
PB 80-4 s 38: Björn Gustavsson, Fakultetsprogram i fast mode
PB 81-1 s 11: Björn Gustavsson, Mer om fast mode
PB 81-3 s 29-32: Björn Gustavsson, Mer om fast mode -2

PGM NN R/S

I motsats till vad som sägs i handboken (s IV-52, V-61) kan anropet Pgm nn R/S användas, vilket faktiskt står nämnt i tabellen på s V-62 i handboken! Anropet startar ett modulprogram på det steg som modulens stegräknare råkar peka på. Exempel: Antag att ett modulprogram ser ut så här och har programnumret 06:

```
000  76  LBL      007  20  20
001  15  E        008  72  ST*
002  00  0        009  00  00
003  42  STD      010  92  RTN
004  00  00       011  61  GTD
005  92  RTN      012  00  00
006  69  DP       013  06  06
```

Om anrop först sker med Pgm 06 E kommer modulens stegräknare att peka på steg 006. Efter anrop med Pgm 06 R/S exekveras steg 006-010 och modulens stegräknare pekar sedan på steg 011. Pgm 06 R/S kan nu upprepas så många gånger man önskar.

Allmänt gäller att om R/S kan användas från tangentbordet vid användning av ett visst modulprogram så kan Pgm nn R/S användas i program.

FÖRDRÖJT MODULANROP

Det går att skilja Pgm nn från R/S. Pgm nn kan exekveras i ett program, och när sedan användaren trycker ned R/S från tangentbordet fortsätter programkörningen i modulen! Detta kan användas när man vill kunna påverka ett körande program utifrån utan att stoppa det.

Metoden är följande: Se till att modulens stegräknare pekar på ett lämpligt programavsnitt genom att göra ett normalt subrutinanrop till modulprogrammet. Exekvera sedan Pgm nn BST. Om nu användaren trycker ned R/S kommer programavsnittet att anropas.

Exempel: Följande program ökar R_1 med 1 tills användaren trycker ned R/S. Då visas resultatet.

```
000  92  RTN      009  01  01      018  61  GTD
001  76  LBL      010  71  SBR     019  00  00
002  15  E        011  00  00      020  16  16
003  25  CLR      012  98  98      021  76  LBL
004  42  STD      013  36  PGM     022  11  A
005  00  00       014  01  01      023  43  RCL
006  42  STD      015  51  BST     024  01  01
007  01  01       016  69  DP      025  81  RST
008  36  PGM      017  21  21
```

Så här ser den aktuella delen av Pgm 01 ut (standardmodulen):

```
098  92  RTN
099  76  LBL
100  11  A
101  98  ADV
102  99  PRT
103  62  PG+
104  00  00
105  11  A
106  99  PRT
107  92  RTN
```

När användaren trycker R/S startas modulprogrammet som genast anropar Lbl A i det vanliga programminnet. Där återkallas innehållet i R_1 och RST utföres för att tömma subrutinanropsregistret. Ett problem är emellertid att användaren inte får hålla nere R/S för länge, eftersom körningen då kommer att stoppas när körningen återupptas i det vanliga programminnet.

Lägg märke till följande:

1. Om man efter Pgm nn BST försöker anropa en subrutin i det vanliga programminnet kommer motsvarande rutin i modulen att anropas. Detsamma gäller R/S.
2. RTN tar bort effekten av Pgm nn BST.
3. Det finns som bekant 6 subrutinnivåer. Ett anrop av ett modulprogram som i sin tur anropar det vanliga programminnet tar upp två subrutinnivåer. Dessa kan enbart nollställas med RST.
4. Vid återgång till det vanliga programminnet stoppas exekveringen om R/S är nedtryckt.

Artiklar om fördröjt modulanrop i Programbiten:

PB 81-1 s 24-25: Björn Gustavsson, Att påverka ett program under körning, utan att stoppa programmet
PB 81-1 s 15: Björn Gustavsson, Slumptal
PB 81-4 s 28-31: Anders Persson, Kostnad för telefonsamtal

DEN FASTA PROGRAMVARAN I TI-58/59

I TI-58/59 finns en fast programvara (oberoende av vilken modul som är isatt) som används för att beräkna statistiska funktioner och omvandlingar. Den går att komma åt på följande sätt (för denna formel måste standardmodulen vara isatt): Tryck (blinkningar negligeras, R/S får ej tryckas för snabbt) Op 09 Pgm 02 R/S R/S R/S Op 17 GTO 000 Op 17 R/S P/R LRN.

Om man inte har skrivare kan man enkelstega med SST till steg 487; om man har skrivare trycker man på nytt LRN och därefter List så skrivs programmet ut. Efter steg 487 sker ett hopp till steg 039; likaså avbryts listningen vid steg 503. För att komma till följande programavsnitt får man använda formeln ovan på nytt med "GTO 489" resp "GTO 505". Stegen 000-379 innehåller program för statistiska funktioner och omvandlingar. Steg 380-511 innehåller siffervärden för beräkning av ln x etc (jämför avsnittet om samband mellan programsteg och dataregister; exakt samma regler gäller dock ej här).

Artiklar om fasta programvaran i Programbiten:

PB 80-3 s 14-15: Lars Hedlund, Djupdykning i TI-59
PB 80-4 s 23: Lars Hedlund, Djupdykning i TI-59 (2)

GRAFISK MODE

Enligt instruktionsboken kan figurer (t ex "plottning" av kurvor) endast tecknas med hela skrivtecken. Västtysken Michael Sperber har dock upptäckt att man genom att trycka en formel från tangentbordet kan programmera in en instruktion som avbryter utskriften så att endast skrivtecknens överdel skrivs och utan radframmatning så att nästa rad fortsätter direkt i det föregående tecknet. Genom att välja lämpligt skrivtecken kan man på det här sättet rita figurer med ca tre gånger bättre upplösning i båda riktningarna.

Programmet skall på steg 024-025 ha "SUMInd 80" och därefter 6 st "Nop" eller nollor. Från tangentbordet trycks med standardmodulen isatt i uppdelning 9 eller 10 Op 17 (4 Op 17 på TI-58) "GTO 024 CLR Pgm 19 SBR 045 P/R LRN Ins LRN RST CLR" Blinkningar ignoreras! Denna sekvens medför att en speciell kod inskjutes på steg 024, så att ett "Op 05" på t ex steg 021-022 avbrytes vid exekveringen. Eftersom labels inte får förekomma efter steg 024, måste man ge akt på att direktadresser inom detta område förskjutes ett steg.

Artikel om Grafisk Mode i Programbiten:
PB 81-2 s 10-13: Lars Hedlund, Graphics Mode och Plot 60

Observera att "*" kan skrivas genom hopskrivning av "+" och "x" på maskiner som saknar "*".

PRAKTISKA RAD

"Op 20" - "Op 39"
Använd dessa instruktioner i stället för de längre "1 SUM OO" etc.

Automatisk uppdelning av program
Vi rekommenderar att program som ej köres i normaluppdelning ändå spelas in i normaluppdelning. Erforderlig uppdelning (t ex 7 Op 17) får då sättas in i programmet på lämpligt ställe. Användningen av programmet blir på detta sätt mycket enklare. (Går dock ej med spärrade magnetkort!)

Skrivkod i dataregister
Försök disponera räknarens uppdelning så att skrivkod (gärna i HIR-form) lagras i dataregister vars innehåll spelas in på magnetkort. Programmet blir på detta sätt snabbare än om skrivkoden skulle skapas i programmet.

Uteslutande av högerparentes före "="
Alla högerparenteser före = kan uteslutas, eftersom = stänger alla öppna parenteser.
Exempel: RCL 01 x (RCL 02 + RCL 03 =.

STANDARD FÖR SKRIVNING AV TANGENTSYMBOLER

I Programbiten skrivs tangentsymboler på följande sätt: Alla skrivs så lika symbolen på tangenterna som möjligt, 2nd utelämnas om inte missförstånd kan uppstå. Exempel: 2nd List skrivs "List", 2nd sin skrivs "sin", 2nd CLR skrivs "2nd CLR" eller "2ndCLR".

Undantag är följande tangentsekvenser:

Symbol	Skrivs
INV SBR	RTN
2nd x=t	EQ
2nd x≥t	GE
x⇄t	x:t
2nd P→R	P/R
÷	DIV
2nd |x|	|x| eller ABS (Abs)
STO 2nd Ind	ST* eller STOInd
RCL 2nd Ind	RC* eller RCLInd
SUM 2nd Ind	SM* eller SUMInd
2nd Prd 2nd Ind	PD* eller PrdInd
2nd Exc 2nd Ind	EX* eller ExcInd
GTO 2nd Ind	GO* eller GTOInd
2nd Op 2nd Ind	Op* eller OpInd
2nd Pgm 2nd Ind	PG* eller PgmInd

Övriga indirekta instruktioner, som inte kopplas till ett programsteg, skrivs inte ihop. Exempel: 2nd Dsz 2nd Ind skrivs "Dsz Ind".