

ANWENDER- DOKUMENTATION	Sprachbeschreibung	MOS
3/89	Assembler asf	MUTOS 1700

Programmtechnische
Beschreibung Teil 2

Sprachbeschreibung

Assembler asf

AC A 7100/A 7150

VEB Robotron-Projekt Dresden

Die vorliegende Dokumentation entspricht dem Stand von 3/89.

Die Ausarbeitung erfolgte durch ein Kollektiv der TH Ilmenau im Auftrage des VEB Robotron-Elektronik Dresden Stammbetrieb des VEB Kombinat Robotron.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Herausgeber:

VEB Robotron-Projekt Dresden
Leningrader Str. 9
Dresden 8010

Kurzreferat

Das Programm asf ist ein ganz einfacher Assembler ohne die Fähigkeit, Makros zu verarbeiten. Er erzeugt ein Ausgabe-File, das Verschiebeinformationen und eine vollständige Symboltabelle enthält. Die Ausgabe wird vom Lader ld des Systems MUTOS 1700 akzeptiert, der verwendet wird, um die Ausgabe verschiedener Assemblerläufe zu vereinigen und um Objektprogramme aus Bibliotheken zu holen. Das Format der Ausgabe wurde so gestaltet, daß ein Programm, welches keine Referenzen auf externe Symbole enthält, ohne weitere Bearbeitung ausführbar ist.

Der Assembler asf ist in der Version 1.1 des Betriebssystems MUTOS 1700 verfügbar und unterstützt im Unterschied zum Assembler as den Befehlssatz des Numerikprozessors K 1810 WM 87.

Inhaltsverzeichnis	Seite
1. HINWEISE ZUR BENUTZUNG	5
2. BEMERKUNGEN ZUR SPRACHBESCHREIBUNG AS	6
ANLAGE A BEFEHLSLISTE DES K 1810 WM 87	7
ANLAGE B BEFEHLE FÜR EXTERNE PROZESSOREN DES K 1810 WM 86 (K 1810 WM 87)	9
B.1. FORMAT	9
B.2. BEMERKUNGEN ZUM BEFEHLSSATZ K 1810 WM 87	9
ANLAGE C BEFEHLSSATZ DES K 1810 WM 87	10
C.1. DATA TRANSFER	10
C.2. ARITHMETIK	12
C.3. TRANSCENTENTAL	15
C.4. PROCESSOR CONTROL	15

1. Hinweise zur Benutzung

Der Assembler asf erzeugt unter Nutzung des Befehlssatzes für den Numerikprozessor K 1810 WM 87 ein Zielprogramm (Objektprogramm). Alle Befehle dieses Befehlssatzes beginnen mit dem Buchstaben f (floating instruction) und werden durch einen escape-Befehl kodiert. Ein solcher Befehl wird vom K 1810 WM 86 erkannt und an den Numerikprozessor zur Abarbeitung weitergegeben. In den Anlagen sind der Befehlssatz des K 1810 WM 87 und einige Erläuterungen dazu enthalten.

Beim Aufruf des Assemblers asf können im Unterschied zum Assembler as die Optionen -D und -T nicht verwendet werden. Ansonsten gilt uneingeschränkt die "Programmtechnische Beschreibung Teil 2, Sprachbeschreibung Assembler as" (C 8016-0001-1 M 3030) von 11/87. Deshalb sind in der vorliegenden Dokumentation für den Assembler asf nur die Ergänzungen für den Befehlssatz des Numerikprozessors K 1810 WM 87 enthalten.

2. Bemerkungen zur Sprachbeschreibung as

~~~~~

In der "Programmtechnischen Beschreibung Teil 2, Sprachbeschreibung Assembler as" (C 8016-0001-1 M 3030) von 11/87 sind einige Fehler enthalten. Im folgenden werden die wesentlichsten korrigiert.

- Das Beispiel im Abschnitt 4 auf Seite 12 ist richtig, jedoch muß es im Text darüber heißen, daß um 10 Bytes weitergezählt wird.
- In der 2. Variante des Interrupt-Befehls in der Befehlsliste auf Seite 27 gibt es keine Operanden. In der Spalte für die Anzahl ist statt 1 eine 0 einzutragen, die Spalte für die Operandenbeschreibung bleibt leer.

Anlage A Befehlsliste des K 1810 WM 87

| Befehls-<br>Beschreibung     | Befehls-<br>Mnemonic | Anz. | Operanden-<br>Beschreibung |
|------------------------------|----------------------|------|----------------------------|
| 2 ** X - 1                   | f2xm1                | 0    |                            |
| absolute value               | fabs                 | 0    |                            |
| add real                     | fadd                 | 0    |                            |
| add long-real (64 Bit)       | faddd                | 1    | A+W                        |
| add short-real (32 Bit)      | faddf                | 1    | A+W                        |
| integer add (16 Bit)         | faddi                | 1    | A+W                        |
| long-integer add (32 Bit)    | faddl                | 1    | A+W                        |
| packed decimal load (80 Bit) | fbld                 | 1    | A+W                        |
| packed decimal store and pop | fbstp                | 1    | A+W                        |
| change sign                  | fchs                 | 0    |                            |
| clear exceptions             | fclex                | 0    |                            |
| compare long-real            | fcomd                | 1    | A+W                        |
| compare short-real           | fcomf                | 1    | A+W                        |
| compare integer              | fcomi                | 1    | A+W                        |
| compare long-integer         | fcoml                | 1    | A+W                        |
| compare long-real and pop    | fcompd               | 1    | A+W                        |
| compare short-real and pop   | fcompf               | 1    | A+W                        |
| compare integer and pop      | fcompi               | 1    | A+W                        |
| compare long-integer and pop | fcompl               | 1    | A+W                        |
| compare real and pop twice   | fcompp               | 0    |                            |
| decrement stack pointer      | fdecstp              | 0    |                            |
| disable interrupts           | fdisi                | 0    |                            |
| divide real                  | fdiv                 | 0    |                            |
| divide long-real             | fdivd                | 1    | A+W                        |
| divide short-real            | fdivf                | 1    | A+W                        |
| divide integer               | fdivi                | 1    | A+W                        |
| divide long-integer          | fdivl                | 1    | A+W                        |
| divide real reversed         | fdivr                | 0    |                            |
| divide long-real reversed    | fdivrd               | 1    | A+W                        |
| divide short-real reversed   | fdivrf               | 1    | A+W                        |
| divide integer reversed      | fdivri               | 1    | A+W                        |
| divide long-integer reversed | fdivrl               | 1    | A+W                        |
| enable interrupts            | feni                 | 0    |                            |
| free register                | ffree                | 0    |                            |
| increment stack pointer      | fincstp              | 0    |                            |
| initialize processor         | finit                | 0    |                            |
| load +1.0                    | fld1                 | 0    |                            |
| load control word            | fldcw                | 1    | A+W                        |
| load long-real               | fldd                 | 1    | A+W                        |
| load environment             | fldenv               | 1    | A+W                        |
| load short-real              | fldf                 | 1    | A+W                        |
| load integer                 | fldi                 | 1    | A+W                        |
| load long-integer            | fldl                 | 1    | A+W                        |
| load log2(e)                 | fldl2e               | 0    |                            |
| load log2(10)                | fldl2t               | 0    |                            |
| load log10(2)                | fldlg2               | 0    |                            |
| load loge(2)                 | fldln2               | 0    |                            |
| load constant pi             | fldpi                | 0    |                            |
| load 4-word-integer (64 Bit) | fldq                 | 1    | A+W                        |
| load temp-real (80 bit)      | fldt                 | 1    | A+W                        |

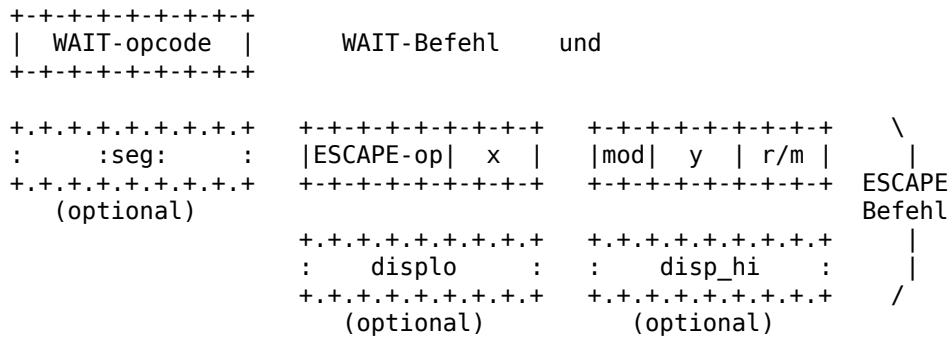
| Befehls-<br>Beschreibung      | Befehls-<br>Mnemonic | Anz. | Operanden-<br>Beschreibung |
|-------------------------------|----------------------|------|----------------------------|
| load +0.0                     | fldz                 | 0    |                            |
| multiply real                 | fmul                 | 0    |                            |
| multiply long-real            | fmuld                | 1    | A+W                        |
| multiply short-real           | fmulf                | 1    | A+W                        |
| multiply integer              | fmulI                | 1    | A+W                        |
| multiply long-integer         | fmulL                | 1    | A+W                        |
| clear exceptions              | fnclex               | 0    |                            |
| disable interrupts            | fndisi               | 0    |                            |
| enable interrupts             | fneni                | 0    |                            |
| initialize processor          | fninit               | 0    |                            |
| no operation                  | fnop                 | 0    |                            |
| save state                    | fnsave               | 1    | A+W                        |
| store control word            | fnstcw               | 1    | A+W                        |
| store environment             | fnstenv              | 1    | A+W                        |
| store status word             | fnstsw               | 1    | A+W                        |
| partial arctangent            | fpatan               | 0    |                            |
| partial remainder             | fprem                | 0    |                            |
| partial tangent               | fptan                | 0    |                            |
| round to integer              | frndint              | 0    |                            |
| restore saved state           | frstor               | 1    | A+W                        |
| save state                    | fsave                | 1    | A+W                        |
| scale                         | fscale               | 0    |                            |
| square root                   | fsqrt                | 0    |                            |
| store control word            | fstcw                | 1    | A+W                        |
| store long-real               | fstd                 | 1    | A+W                        |
| store environment             | fstenv               | 1    | A+W                        |
| store short-real              | fstf                 | 1    | A+W                        |
| store integer                 | fsti                 | 1    | A+W                        |
| store long-integer            | fstl                 | 1    | A+W                        |
| store long-real and pop       | fstopd               | 1    | A+W                        |
| store short-real and pop      | fstpf                | 1    | A+W                        |
| store integer and pop         | fstpi                | 1    | A+W                        |
| store long-integer and pop    | fstpl                | 1    | A+W                        |
| store 4-word-integer, pop     | fstpq                | 1    | A+W                        |
| store temp-real and pop       | fstpt                | 1    | A+W                        |
| store status word             | fstsw                | 1    | A+W                        |
| subtract real                 | fsub                 | 0    |                            |
| subtract long-real            | fsubd                | 1    | A+W                        |
| subtract short-real           | fsubf                | 1    | A+W                        |
| subtract integer              | fsubI                | 1    | A+W                        |
| subtract long-integer         | fsubL                | 1    | A+W                        |
| subtract real reversed        | fsubr                | 0    |                            |
| subtract long-real rev.       | fsubrd               | 1    | A+W                        |
| subtract short-real rev.      | fsubrf               | 1    | A+W                        |
| subtract integer rev.         | fsubri               | 1    | A+W                        |
| subtract long-integer rev.    | fsubrl               | 1    | A+W                        |
| test stack top against 0.0    | ftst                 | 0    |                            |
| CPU-wait while WM 87 is busy  | fwait                | 0    |                            |
| examine stack top             | fxam                 | 0    |                            |
| exchange registers            | fxch                 | 0    |                            |
| extract exponent, significand | fxtract              | 0    |                            |
| Y * log2(X)                   | fyl2x                | 0    |                            |
| Y * log2(X + 1)               | fyl2xp1              | 0    |                            |



Anlage B Befehle für externe Prozessoren des  
K 1810 WM 86 (K 1810 WM 87)

B.1. Format

Der Code, der für den externen Prozessor bereitgestellt wird, besteht aus einem WAIT-Befehl und einem ESCAPE-Befehl.



Die Felder x und y enthalten den Operationscode für den externen Prozessor. Sie werden vom ESCAPE-Befehl nicht ausgewertet.

- x = opcode group
- y = opcode mit group

B.2. Bemerkungen zum Befehlssatz K 1810 WM 87

In ESCAPE-Befehlen auftretende Bezeichnungen haben folgende Bedeutung:

- M = Kennzeichnung der Operandenlänge
- F = Kennzeichnung des Operandentyps

| M | F | Operand      |
|---|---|--------------|
| 0 | 0 | short real   |
| 1 | 0 | long real    |
| 0 | 1 | integer      |
| 1 | 1 | long integer |

- R = 0 - "normaler" Befehl (destination (op) source).
- R = 1 - "umgekehrter" (reversed) Befehl (source (op) destination).

C.1. Data Transfer

F L D = Load to ST(0)

---

Integer/real memory to ST(0)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :M F 1|mod:0 0 0: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Temp-real memory to ST(0)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :0 1 1|mod:1 0 1: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

4-word-integer memory to ST(0)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :1 1 1|mod:1 0 1: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

F B L D = Load packed decimal to ST(0)

---

Memory to ST(0)

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :1 1 1|mod:1 0 0: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

F B S T P = Store packed decimal to memory and pop

---

ST(0) to memory

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :1 1 1|mod:1 1 0: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

F S T = Store ST(0) to memory

---

Integer/real ST(0) to memory

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :M F 1|mod:0 1 0: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

F S T P = Store ST(0) to memory and pop

---

Integer/real ST(0) to memory

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| escape :M F 1|mod:0 1 1: r/m | disp |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Temp-real ST(0) to memory

```
+++++  
| escape :0 1 1|mod:1 1 1: r/m | disp |  
+++++  
4-word-integer ST(0) to memory  
+++++  
| escape :1 1 1|mod:1 1 1: r/m | disp |  
+++++
```

F L D Z = Load +0.0 to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 1 1 0|  
+++++
```

F L D 1 = Load +1.0 to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 0 0 0|  
+++++
```

F L D P I = Load constant pi (+3.1415...) to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 0 1 1|  
+++++
```

F L D L 2 T = Load  $\log_2 10$  to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 0 0 1|  
+++++
```

F L D L 2 E = Load  $\log_2 e$  to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 0 1 0|  
+++++
```

F L D L G 2 = Load  $\log_{10} 2$  to ST(0)

---

```
+++++  
| escape :0 0 1|1 1 1 0 1 1 0 0|  
+++++
```

---

F L D L N 2 = Load  $\log_2$  to ST(0)

---

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 1|1 1 1 0 1 1 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

F X C H = Exchange ST(0) and ST(1)

---

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 1|1 1 0 0 1 0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

## C.2. Arithmetik

~~~~~

F A D D = Addition

Integer/real memory with ST(0)

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :M F 0|mod:0 0 0: r/m |      disp      :
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

ST(1) and ST(0)

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 0|1 1 0 0 0 0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

F S U B = Subtraction and reversed subtraction

Integer/real memory with ST(0)

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :M F 0|mod:1 0 R: r/m |      disp      :
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

ST(1) and ST(0)

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 0|1 1 1 0 R:0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

F M U L = Multiplication

Integer/real memory with ST(0)

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :M F 0|mod:0 0 1: r/m |      disp      :
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

ST(1) and ST(0)

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :0 0 0|1 1 0 0 1 0 0 1|
+---+---+---+---+---+---+---+---+

```

F D I V = Division and reversed division

Integer/real memory with ST(0)

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :M F 0|mod:1 1 R: r/m |      disp  :
+---+---+---+---+---+---+---+---+---+---+

```

ST(1) and ST(0)

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :0 0 0|1 1 1 1 R:0 0 1|
+---+---+---+---+---+---+---+---+---+---+

```

F C O M = Compare with ST(0)

Integer/real memory with ST(0)

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :M F 0|mod:0 1 0: r/m |      disp  :
+---+---+---+---+---+---+---+---+---+---+

```

F C O M P = Compare with ST(0) and pop

Integer/real memory with ST(0)

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :M F 0|mod:0 1 1: r/m |      disp  :
+---+---+---+---+---+---+---+---+---+---+

```

F C O M P P = Compare with ST(0) and pop twice

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :1 0 0|1 1 0 1 1 0 1 0|
+---+---+---+---+---+---+---+---+---+---+

```

F T S T = Test ST(0) against +0.0

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :0 0 1|1 1 1 0 0 1 0 0|
+---+---+---+---+---+---+---+---+---+---+

```

F X A M = Examine ST(0) and report condition code

```

+---+---+---+---+---+---+---+---+---+---+
| escape  :0 0 1|1 1 1 0 0 1 0 1|
+---+---+---+---+---+---+---+---+---+---+

```

F S Q R T = Calculate square root of ST(0)

```

+++++
| escape :0 0 1|1 1 1 1 1 0 1 0|
+++++

```

F S C A L E = Add ST(1) to exponent of ST(0)

```

+++++
| escape :0 0 1|1 1 1 1 1 1 0 1|
+++++

```

F P R E M = Modulo division of ST(0) by ST(1)

```

+++++
| escape :0 0 1|1 1 1 1 1 0 0 0|
+++++

```

F R N D I N T = Round ST(0) to integer

```

+++++
| escape :0 0 1|1 1 1 1 1 1 0 0|
+++++

```

F X T R A C T = Extract exponent and significand from ST(0)

```

+++++
| escape :0 0 1|1 1 1 1 0 1 0 0|
+++++

```

F A B S = Take absolute value of ST(0)

```

+++++
| escape :0 0 1|1 1 1 0 0 0 0 1|
+++++

```

F C H S = Change sign of ST(0)

```

+++++
| escape :0 0 1|1 1 1 0 0 0 0 0|
+++++

```

C.3. Transcendental

F P T A N = Calculate partial tangent

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 1 0 0 1 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F P A T A N = Calculate partial arctangent

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 1 0 0 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F 2 X M 1 = Calculate 2^x-1

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 1 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F Y L 2 X = Calculate function $Y * \log_2 X$

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 1 0 0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F Y L 2 X P 1 = Calculate value of $Y * \log_2(X+1)$

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 1 1 0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

C.4. Processor control

F I N I T / F N I N I T = Initialize processor

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape :0 0 1|1 1 1 0 0 0 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F S T S W / F N S T S W = Store processor status word

```
+++++
| escape :1 0 1|mod:1 1 1: r/m |     disp     :
+++++
```

F L D C W = Load processor control word

```
+++++
| escape :0 0 1|mod:1 0 1: r/m |     disp     :
+++++
```

F S T C W / F N S T C W = Store processor control word

```
+++++
| escape :0 0 1|mod:1 1 1: r/m |     disp     :
+++++
```

F C L E X / F N C L E X = Clear exceptions

```
+++++
| escape :0 1 1|1 1 1 0 0 0 1 0|
+++++
```

F S T E N V / F N S T E N V = Store environment

```
+++++
| escape :0 0 1|mod:1 1 0: r/m |     disp     :
+++++
```

F L D E N V = Load environment

```
+++++
| escape :0 0 1|mod:1 0 0: r/m |     disp     :
+++++
```

F S A V E / F N S A V E = Save processor state to memory

```
+++++
| escape :1 0 1|mod:1 1 0: r/m |     disp     :
+++++
```

F R S T O R = Restore processor state from memory

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :1 0 1|mod:1 0 0: r/m |      disp      :
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F I N C S T P = Increment stack pointer

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 1|1 1 1 1 0 1 1 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F D E C S T P = Decrement stack pointer

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 1|1 1 1 1 0 1 1 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F D I S I / F N D I S I = Disable interrupts

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 1 1|1 1 1 0 0 0 0 1|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F E N I / F N E N I = Enable interrupts

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 1 1|1 1 1 0 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F F R E E = Free stack register ST(0)

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :1 0 1|1 1 0 0 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

F W A I T = Alternate mnemonic for the CPU wait instruction

```
+--+--+--+--+--+--+--+
|1 0 0 1 1 0 1 1|
+--+--+--+--+--+--+--+
```

F N O P = No operation

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| escape  :0 0 1|1 1 0 1 0 0 0 0|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```