

FTKL—Projekt

Flight 68000 Urladerprogramm

Rüf Markus, FN4
1996

```
move.b d0,csrb{a0}      *set baud rate
```

```
*Flight-communication programm
```

```
htllogc
```

```
btst    #tx,srb+duart.l  *wait for tx terminal buffer empty
beq      htllogo
```

```
lea     logo,a0          *load address
bsr      flight
```

```
*header text
flight
```

```
move.l   (a0)+,d0        *output logo
beq      logoc
jsr      txtout
bra      flight
```

```
bra      delay
movea.l  #dbufb,a0
clr.l    d4
clr.l    d5
```

```
*load sformat bytes into dbuff from port b
```

```
load1
jsr      datain          *get character
cmpi.b   #'S',d0
bne      load            *back if first character isn't a 'S'
jsr      datain
cmpi.b   #'6',d0
bhi      loadh           *exit if second character is 7,8,9
```

```
*s_format file
```

```
*see for further information about s_format files booklet
```

```
*first line of s_format files begins with S0
```

```
*(containing file name),--> don't care!
```

```
cmpi.l   #'2',d4
bne      load1
```

```
moveq     #2,d4          *set address count for 'S2'format
moveq     #-3,d5         *preset byte count register
bra .      load2
```

```
load1
```

```
cmpi.l   #'1',d0
bne      load            *back again if not '1'
moveq     #-2,d5         *preset byte count register (size)
moveq     #1,d4          *set address count for 's1' format
```

```
load2
```

```
clr.l     d6             *clear checksummer
bsr      bytein          *get byte count
add.l     d0,d5          *and add to counter
clr.l     d3             *clear address register builder
```

```
load3
```

```
bsr      bytein          *get address byte
lsl.l     #8,d3          *log. shift left 8 spot
add.l     d0,d3          *add to address builder
dbra     d4,load3        *number of address bytes
move.l    d3,dbufb
```

```

loaf
    bsr      bytein      *get data byte
    subq     #1,d5       *decrement byte counter
    tst.b    d5
    beq      loag        *branch if checksum

*write data into addressregister
adl
    move.l   d0,(a0)      *store byte in dbuff
    cmp.l    (a0)+,d0     *stored it? increment a0
    beq      loaf        *branch if it did
    bra      lderr        *error if not

loag
    addq.b   #1,d6        *add 1 to checksummer (d6)
    beq      load        *branch if correct

loah
    bsr      bytein      *finished, get byte count in d0
    moveq     #0,d2
    move.l    d0,d2
    subq.l    #1,d2       *then minus 1 in d2

*print load finished
loaz
    bsr      bytein      *get and ignore remaining
    dbra     d2,loaz      *bytes of line
    move.w    #$4e75,(a0) *load opt.code rts on dbuff
    lea       ldcom,aC

loak
    move.l    (a0)+,d0
    beq      ldc
    jsr      txtout
    bra      loak

    bra      delay
    rts

*points to programm startaddress
gc
    move.l    (a0)+,d0
    beq      gow
    jsr      txtout
    bra      gotxt
    bra      delay
    bra      delay
    movem.l   a6-a0/d7-d0,-(a7) *register retten
    bra      dbuff          *start programm
    movem.l   (a7)+,d0-d7/a0-a6 *recover register
    lea       endl,aC

*programm ende on monitor
ende
    move.l    (a0)+,d0      *out reset mask input
    beq      end0
    jsr      txtout
    bra      ende

```

 *subroutines

```

*get byte i/p {two ascii} in from terminal and
*add to checksum adder (d6)
*exit with byte in d0.b

```

```
bytein
```

```

    bsr      hexin      *get first character
    lsl.b    #4,d0      *shift left one digit
    move.b   d0,d1
    bsr      hexin      *get second character
    andi.b   #$f,d0
    add.b    d1,d0      *combine
    add.b    d0,d6      *add to checksummer
    rts

```

```

*gets character from terminal and converts to hex
*if no hex, 'LOAD ERROR'

```

```
hexin
```

```

    jsr      datain      *get character
    bsr      hexcon      *convert to hex
    bcs      lderr
    rts

```

```
hexcon
```

```

    cmpi.b   #$30,d0      *branch if d0 < '0'
    bmi      nonhex
    cmpi.b   #$39,d0
    ble      hexfin      *branch if d0 <= '9'
    cmpi.b   #$41,d0
    bmi      nonhex      *branch if d0 < 'A'
    cmpi.b   #$46,d0
    bgt      nonhex      *branch if d0 > 'F'
    subq.b   #7,d0

```

```
hexfin
```

```

    andi.b   #$f,d0      *isolate hex digit
    bra      hexout      *and clear 'c' bit

```

```
nonhex
```

```

    ori.b    #1,ccr      *set 'c' bit

```

```
hexout
```

```
    rts
```

```
*loading aborted
```

```
lderr
```

```

    lea      lerror,a0+$2C
    bsr      lerror

```

```
lerror
```

```

    move.l   (a0)+,d0
    beq      lerrc
    jmp      txtout
    bra      lerror

    bra      delay
    rts

```

```
*load data bytes from port b
```

```
datain
```

```

    btst     #rx,srb+duart.l *character in from port b
    beq      datain          *branch if not

```

```

    move.l    trbb+duart.l,d0    *get character, load in d0
    andi.b    #$7f,d0           *and strip parity
    rts                          *return with character

```

```

*load text in transmitter buffer
txtout

```

```

    move.b    d0,trbb+duart.l    *send character to tr_buffer
    rts

```

```

*delay routine / 1sec waitstep
delay

```

```

    move.l    #$40000,d0

```

```

delayl

```

```

    subq.w    #1,d0
    bne       delayl
    rts

```

```

*****
*data

```

```

*
logc    dc.l    'Ü Ü 00Ü00 Ü * FLIGHT 68000 *'
        dc.l    lf,cr
        dc.l    'ÜÜÜÜ Ü Ü *simulation programm*'
        dc.b    lf,cr
        dc.b    'Ü Ü Ü ÜÜÜÜ * by Ruef Markus *'
        dc.l    lf,cr
        dc.l    ' RANKWEIL'
        dc.l    lf,cr
        dc.l    0

```

```

logoc
    rts

```

```

*
larror
    dc.l    '****LOAE ERROR****'
    dc.l    lf,cr
    dc.l    'PLEASE RESET SYSTEM'
    dc.l    lf,cr
    dc.b    0

```

```

lerrc
    rts

```

```

*
ldcom
    dc.l    'LOAD COMPLETED'
    dc.l    0

```

```

ldc
    rts

```

```

*
gotxt
    dc.b    'PROGRAMM IS RUNNING'
    dc.b    lf,cr
    dc.l    0

```

```

gow
    rts

```

```
*
endl      dc.h      'FINISHED'
          dc.h      lf,cr
          dc.b      'PLEASE RESET  SYSTEM!'
          dc.b      lf,cr
          dc.h      0
```

```
end0      rts
```

```
end
```

```
*****
```