

```

1      ;=====
2      ;flyer.s: module for Flyer network card.
3      ;Uses logical file number 9 for data
4      ;and 8 for the command channel.
5      ;Secondary address 9 is used for the
6      ;data channel.
7      ;Requires the following include files:
8      ;  library.s
9      ;  prosys.s
10     ;  c64.s
11     ;=====
12
13     0  program flyer own export
14
15     0  include library
16     7  LIST 0
17     7  LIST
18
19     7  list 0
20     7  list
21
22     7  export data byte datachan=9
23     8  export data byte cmdchan=15
24
25     ;-----
26     ;Close Flyer.
27     ;pass: nothing
28     ;return: nothing
29     ;-----
30     9  export proc closeflyer
31
32     9  begin
33     C  jsr cclose,9
34     16 jsr cclose,8
35     20 end
36
37     ;-----
38     ;Get field from command channel output
39     ;(call readflyer on cmd channel first).
40     ;pass: address of buffer containing full
41     ;      command channel output
42     ;pass: field to retrieve (0-3)
43     ;return: address of desired field
44     ;-----
45     24 export func word getstat
46     1  arg word buffer ;address of buffer
47     3  arg word fld    ;which field
48     5  word curfld
49     7  word w
50     0  own byte statfld[32]
51
52     24 begin
53     24 statfld[0]=0
54     2A curfld=0
55     2E w=0

```

```

256 32 while curfld < 4
257 3B   if curfld=fld
258 43   break
259 46   buffer=buffer+1
260 4D   if buffer@< = ','
261 56   buffer=buffer+1 ;past comma
262 5D   curfld=curfld+1
263 67 while buffer@< <> ',' and buffer@< <> $0d
264 77   statfld[w]=buffer@<
265 7F   buffer=buffer+1
266 86   w=w+1
267 90   statfld[w]=0
268 96   return statfld
269 9A   end
270
271 ;-----
272 ;Open Flyer network card.
273 ;pass: address of URL to open
274 ;return: 0 on success, error code from
275 ;        accumulator otherwise.
276 ;-----
277 9E export func byte openflyer
278 1   arg word url
279 3   byte err
280
281 9B begin
282 9E err=0
283 A1 jsr csetlfs,8,7,15
284 B1 jsr csetnam,0
285 BA jsr copen
286 C1 jsr csetlfs,9,7,9
287 D1 jsr csetnam,lenstr(url),url:<,url:>
288 E7 jsr copen
289 EE if regf and regfc ;carry flag set?
290 F6   err=rega        ;error code in .A
291 FB   closeflyer
292 FE   return err
293 101 end
294
295 ;-----
296 ;Send data to Flyer.
297 ;pass: address of data buffer
298 ;pass: constant for data or command (use
299 ;        constants 'datachan', 'cmdchan')
300 ;return: C64 status byte (0 on success)
301 ;-----
302 105 export func byte sendflyer
303 1   arg word buffer ;array of byte
304 3   arg byte sec
305
306 102 begin
307 105 if sec <> datachan and sec <> cmdchan
308 115   return -1
309 11A m[cstatus]=0
310 121 jsr clisten,7

```

```
311 12B if m[cstatus] > 127
312 137 return m[cstatus]
313 13E jsr csecond,sec or $60
314 14B if m[cstatus] > 127
315 157 jsr cunlsn
316 15E return m[cstatus]
317 165 while buffer@< <> 0
318 16D jsr cciout,buffer@<
319 178 if m[cstatus] <> 0
320 183 return m[cstatus]
321 18A buffer=buffer+1
322 194 jsr cciout,13
323 19E jsr cunlsn
324 1A5 return m[cstatus]
325 1AC end
326
327 ;-----
328 ;Read data from Flyer.
329 ;pass: address of data buffer
330 ;pass: constant for data or command (use
331 ; constants 'datachan', 'cmdchan')
332 ;return: C64 status byte (0 on success)
333 ; or -1 if invalid channel passed
334 ;-----
335 1B0 export func byte readflyer
336 1 arg word buffer
337 3 arg byte sec
338 5 byte first
339 6 byte eof
340 7 byte temp
341
342 1AD begin
343 1B0 if sec <> datachan and sec <> cmdchan
344 1C0 return -1
345 1C5 m[cstatus]=0
346 1CC jsr ctalk,7
347 1D6 if m[cstatus] > 127
348 1E2 return m[cstatus]
349 1E9 jsr ctksa,sec or $60
350 1F6 if m[cstatus] > 127
351 202 jsr cuntlk
352 209 return m[cstatus]
353 210 first=true
354 213 eof=false
355 216 repeat
356 216 jsr cacptr
357 21D temp=rega
358 222 if m[cstatus] <> 0
359 22D if m[cstatus] and $40 ;eof?
360 239 eof=true
361 23C if first
362 241 break
363 247 else
364 247 first=false
365 24D else
```

```
366 24D      temp=m[cstatus]
367 255      jsr cuntlk
368 25C      m[cstatus]=temp
369 264      return m[cstatus]
370 26B      m[buffer]=temp
371 272      buffer=buffer+1
372 279      until eof
373 27E      m[buffer]=0
374 284      jsr cuntlk
375 28B      return 0
376 28D      end
377
378          ;-----
379          ;Send HTTP-TRANSACT command to Flyer.
380          ;return: C64 status byte (0 on success)
381          ;          or -1 if invalid channel passed
382          ;-----
383 291      export func byte transflyer
384 1        byte result
385
386 28E      begin
387 291      result=sendflyer("HTTP-TRANSACT:9",cmdchan)
388 29D      return result
389 2A0      end
390
391          ;=====
392
393 2B4      begin
394 2B8      end
```

FLYER compiled:

394 Source lines

\$02B5 (693) bytes Obj.

\$0 Scalar, \$0020 (32) tot. Vars.

Table usage:

Symbols: 21% Fwd. Refs.: 0%

Strings: 0%