DANSK DATA ELEKTRONIK

ID-7000 UTILITY ROUTINES

for the

ID-7000 MICROPROCESSOR SYSTEM

JUNE 1976

Written by Ole Lading.

TABLE OF CONTENTS:

1.General description.

The ID-7000 utility programs are a set of INTEL 8080 programs, which can be used by user programs written in Assembler language and PL/M language. The set includes I/O-routines for communication with input/output devices and a loader for loading binary tapes in the INTELLEC 8 format into the ID-7000 microprocessor system.

The ID-7000 utility programs are placed in the DEBUG-extension PROM, but they are not a part of the DEBUG-program, and are not used by the DEBUG-program.

---

2. I/O system.

The I/O-system operates with logical and physical devices. The four logical devices are:

CONSOLE	An interactive, character-oriented device, used for both input and output.
READER	A character-oriented, input-only device, which transfers data on command and signals the program when there is no more data (end-of-file).
PUNCH	Acharacter-oriented, output-only device, which accepts a character from the program and records it on the external medium.
LIST	A character-oriented, output-only device which accepts a character from the program and records it on some external medium in human readable form.

The correspondence between the logical devices and the physical devices is made by the I/O status byte (IOSB). This byte is placed in RAM memory in location 0 and can be changed by the user (by means of the DEBUG-program) or by the user program (by means of the utility routines ICHK and ISET).

The I/O status byte has the following format, showing the correspondence between logical and physical devices:

7	6	5	4	3	2	1	0

LIST FIELD:	PUNCH FIELD:	READER FIELD:	CONSOLE FIELD:
00 TTY	00 TTY	00 TTY	00 TTY
01 DUMMY	01 PTP	01 PTR	01 DUMMY
10 DUMMY	10 DUMMY	10 DUMMY	10 BATCH
11 USER L	11 USER P	11 USER R	11 USER C

The physical devices are:

TTY — Teletype or Teletype compatible data terminal. If tty is used as PUNCH-device or READER-device, the unit must be a Teletype 33 ASR. The data terminal must be connected to the microprocessor system via an ID-7004 Asynchronous Data Communication module using the I/O addresses x.A og x.B.

DUMMY — This - non-exixting device - can be used to receive not wanted output (f.ex. assembler list output). Output and input to/from DUMMY results in immediate return to the calling program without any I/O transfer.

PTP — ID-7000 high speed paper tape punch unit connected to the microprocessor system via an ID-7006 reader/punch module using the I/O adresses x.C and x.D.

PTR — ID-7000 high speed paper tape reader unit connected to the microprocessor system via an ID-7006 reader/punch module using the I/O addresses x.C and x.D.

BATCH — When BATCH is specified as CONSOLE device, the console input is taken from the specified READER device and output is sent to the specified LIST device.

USER DEV	These devices are user specified devices. The I/O system contains no drivers for these devices. Drivers must be supplied by the user. When a user supplied device is called, the I/O system executes a JUMP to a specific address as shown below:
USER C	User defined CONSOLE device. Input: JUMP to x.1. OUTPUT:JUMP to x.4.
USER R	User defined READER device. JUMP to x.7.
USER P	User defined PUNCH device JUMP to x.A.
USER L	User defined LIST device JUMP to x.D.

It should be noticed, that locations x.0 - x.F are reserved for I/O status byte and pointers to drivers for user defined devices. These locations should not be used by user programs using the I/O system.

The I/O-system is used by the ID-7000 program evaluation software (TEXT-EDITOR and ASSEMBLER). This permits the user to select the physical units used by the evaluation software. In order to make the I/O subroutines available also for user programs, section 2.1 describes the different I/O subroutines.

2.1 I/O-subroutines. In table 1 is shown the I/O-subroutines and their call addresses. The routines are activated by CALL instructions from the programs to the adresses shown in the table. It should be noticed that I/O-drivers are running without use of interrupts. Anyway the I/O devices may generate interrupts, and the user is responsible to disable the interrupts when activating the I/O routines.

Sections 2.1.1 to 2.1.11 describe the different I/O-subroutines. Appendix 1 contains a source list of the routines.

NAME	FUNCTION	ADDR.(x)	ADDR (d)
CI	CONSOLE input	EE84	61060
RI	READER input	EE95	61077
CO	CONSOLE output	EEAA	61098
PO	PUNCH output	EEBB	61115
LO	LIST output	EECE	61134
CSTS	CONSOLE input status	EEE1	61153
ICHK	Get I/O status byte	EEF7	61175
ISET	Set I/O status byte	EEFB	61179
SSR	Start/Stop READER	EF00	61184
TSET	Set TTY control word	EF09	61193
TBRK	Test TTY BREAK	EEEF	61167

TABLE I : I/O subroutines.

2.1.1 CI - CONSOLE INPUT: This routine returns a character received from the selected CONSOLE input device to the caller in the A register. If the CONSOLE input device is the TTY, the received character is sampled in accordance with the control word previous sent to the interface module (f.ex. by TSET).

The condition flags are affected by the routine

2.1.2 RI - READER INPUT: This routine returns a character received from the selected READER device to the caller in the A register. If no character is read from the device within about 2 sec., the carry flag is set, and the A register is zeroed. Otherwise the carry flag is cleared. If the TTY

is used as READER device, the character is sampled in accordance with the control word previous sent to the interface module (f.ex. by TSET). To get a character from the TTY-READER, this must be running. Start/stop of the TTY-READER is controlled by the I/O-subroutine SSR.

The condition flags are affected by the routine. During execution two locations of the stack are used.

2.1.3 CO - CONSOLE OUTPUT. This routine transmits a character, passed from the calling program in the C register to the selected CONSOLE output device. If the CONSOLE device is the TTY, the character is transmitted in accordance with the contol word previous sent to the interface module (f.ex. by TSET). The A register and the condition flags are affected.

2.1.4 PO - PUNCH OUTPUT: This routine transmits a character from the calling program in the C register, to the selected PUNCH device. If the PUNCH device is the TTY, the character is transmitted in accordance with the control word previous sent to the interface (f.ex. by TSET). The A register and the condition flags are affected.

2.1.5 LO - LIST OUTPUT: This routine transmits a character from the calling program in the C register, to the selected LIST device. If the LIST device is the TTY, the character is transmitted in accordance with the control word previous sent to the module. (F.ex. by TSET). The Aregister and the condition flags are affected.

2.1.6 CSTS - CONSOLE INPUT STATUS: This routine tells the calling program, whether a character is ready from the TTY CONSOLE device or not. If a character is ready, the routine returns a logical 1 (x.FF) in A. Otherwise a logical 0 (x.00) is returned. If CONSOLE devices different from the TTY is specified, no status information is returned. In this case A(1:0) contains the code for the specified CONSOLE device. A(7:2) are all zero. The condition flags are affected by the routine.

2.1.7 ICHK - I/O CHECK. This routine returns the I/O status byte (IOSB) in the A register to the calling program.

2.1.8 ISET - I/O SET. This routine loads the I/O status byte (IOSB) with the value passed from the calling program in the C register.

2.1.9 SSR - START/STOP READER. This routine is used to start and stop the READER device if the TTY READER is selected. Otherwise the routine has no effect.To start the TTY READER, an ASCII X-ON character (x.11) is passed in the C register. To stop the TTY READER, an ASCII X-OFF (x.13) character is passed in the A register. It should be noticed, that the TTY READER is not stopped immediately. A couple of characters are read by the TTY after the SSR program was called.

2.1.10 TSET - TTY CONTROL SET. This routine loads the TTY interface module control register with the value passed in the C register. The A register is affected. The control register of the module defines the format of the received/transmitted character (number of data bits, parity status, number of stopbits etc.). For detailed information, refer to the manual of the ID-7004 Asynchronous communication module. Default value of the control word (as set by the DEBUG program) defines 7 data bits, 1 parity bit (even parity) and 2 stopbits.

2.1.11 TBRK - TTY BREAK TEST. This routine checks the framing error condition of the TTY interface module. This error condition appears if the BREAK key of the TTY has been used. It is reset when a data input from the interface is performed (f.ex by means of CI). If TTY BREAK TEST is true, a logical 1 (x.FF) is returned in the A register. Otherwise a logical 0 (x.00) is returned. The condition flags are affected.

3. INTEL LOADER.

The ID-7000 microprocessor system uses another binary tape format than used in the INTELLEC 8 system to reduce the length of the binary tapes. The ID-7000 DEBUG program includes loaders for this format for different input units (TTY and high speed reader). In order to load binary tapes in the INTELLEC 8 format into RAM memory, the utility program INLD is available.

The program uses the I/O system described in section 2.1 and takes input from the logical device READER. The program uses ten locations in the stack. The stackpointer must be loaded by the operator to some free RAM memory. Beyond this, the program uses no RAM memory.

After mounting of the tape in the READER device and proper loading of the I/O status byte, the program is started from DEBUG in location x.EF61.

The INLD program uses no other logical devices . In error conditions, control is passed to the DEBUG program and a three number statement is written on the TTY:

```
* FFFF
  DATA ADDR
```

ADDR contains the error address (as defined below). The 8 most significant bits in DATA contains the error code. The 8 least significant bits in DATA contains the error data (as defined below).The different error codes and the corresponding error type are listed below:

DATA H: 0	Timeout error. The READER device is not loaded. This error code is also given after loading the complete tape,when tape is leaving the reader. ADDR indicates the location of the last loaded byte. DATA L - 0.
DATA H: 1	Block type error. An illegal block type byte is read. ADDR = 0. DATA L contains the illegal block type.

DATA H: 2 — Memory check error. This error condition indicates, that the memory location just written does not contain the correct information. A possible reason to the error is that no (not protected) RAM memory with the given address is present. ADDR contains the address of the memory location. DATA L contains the correct byte value. (The stored value is displayed by activating the / key).

DATA H: 3 — Check sum error. This error condition indicates, that the modulo 256 sum of all hex digits in the record is not 0. DATA L contains the actual check sum. ADDR contains the address of the last byte in the record.

Appendix 1 contains a list of the INLD program.

```
    0    0000   driv=x.ee84
    0    0000   k=driv
61060    ee84   ttyd=10
61060    ee84   ttys=11
61060    ee84   ptrd=12
61060    ee84   ptrs=13
61060    ee84   ptpd=12
61060    ee84   ptps=13
61060    ee84   iosb=0
61060    ee84   toer=0
61060    ee84   ferr=1
61060    ee84   berr=1
61060    ee84   merr=2
61060    ee84   cerr=3
61060    ee84
61060    ee84
61060    ee84   ;     logical drivers
61060    ee84   ci: lda iosb
61063    ee87        ani 3
61065    ee89        jz ttyi     ;0: ci=tty
61068    ee8c        dcr a
61069    ee8d        rz          ;1: ci=dummy
61070    ee8e        dcr a
61071    ee8f        jz ri      ;2: batch: ci=ri
61074    ee92        jmp 1     ;3: user defined console input
61077    ee95   ri: lda iosb
61080    ee98        ani x.0c
61082    ee9a        jz ttyr     ;0: ri=tty
61085    ee9d        cpi 4
61087    ee9f        jz ptr      ;1: ri=ptr (paper tape reader)
61090    eea2        cpi 8
61092    eea4        rz         ;2: ri=dummy
61093    eea5        nop
61094    eea6        nop
61095    eea7        jmp 7    ;3: ri=user2
61098    eeaa   co: lda iosb
61101    eead        ani 3
61103    eeaf        jz ttyo     ;0: co=tty
61106    eeb2        dcr a
61107    eeb3        rz          ;1: co=dummy
61108    eeb4        dcr a
61109    eeb5        jz lo      ;2: co=lo
61112    eeb8        jmp 4     ;3: co=user defined
61115    eebb   po: lda iosb
61118    eebe        ani x.30
61120    eec0        jz ttyo     ;0: po=tty
61123    eec3        cpi x.10
61125    eec5        jz ptp      ;1: po=ptp (paper tape punch)
61128    eec8        cpi x.20
61130    eeca        rz        ;2: po=dummy
61131    eecb        jmp x.a    ;3: po=user
61134    eece   lo: lda iosb
61137    eed1        ani x.c0
```

```
61139   eed3          jz ttyo     ;0: lo=tty
61142   eed6          cpi x,40
61144   eed8          rz          ;1: lo=dummy (crt)
61145   eed9          cpi x,80
61147   eedb          rz            ;2: lo=dummy
61148   eedc          nop
61149   eedd          nop
61150   eede          jmp x,d     ;3: lo=user2
61153   eee1    csts: lda iosb
61156   eee4          ani 3
61158   eee6          rnz
61159   eee7          in ttys
61161   eee9          ani 16
61163   eeeb          rz
61164   eeec          mvi a,x,ff
61166   eeee          ret
61167   eeef    tbrk: in ttys
61169   eef1          ani 4
61171   eef3          rz          ;if(ttys<>break) return with a=0
61172   eef4          mvi a,x,ff
61174   eef6          ret         ;return with a=ff
61175   eef7    ichk: lda iosb
61178   eefa          ret
61179   eefb    iset: mov a,c
61180   eefc          sta iosb
61183   eeff          ret
61184   ef00    ssr:  lda iosb
61187   ef03          ani x,0c
61189   ef05          rnz         ;if(reader<>tty) return
61190   ef06          jmp ttyo
61193   ef09    tset: mov a,c
61194   ef0a          out ttys
61196   ef0c          ret
61197   ef0d    ;     physical drivers
61197   ef0d    ttyi: in ttys
61199   ef0f          ani 16
61201   ef11          jz ttyi
61204   ef14          in ttyd
61206   ef16          ret
61207   ef17    ttyo: in ttys
61209   ef19          ani 1
61211   ef1b          jz ttyo
61214   ef1e          mov a,c
61215   ef1f          out ttyd
61217   ef21          ret
61218   ef22    ptr:  push b
61219   ef23          mvi b,x,ff
61221   ef25    ptt:  in ptrs
61223   ef27          ani 16
61225   ef29          jnz ptd     ;if(status=ready) goto ptd
61228   ef2c          dcr c
61229   ef2d          jnz ptt
61232   ef30          dcr b
61233   ef31          jnz ptt     ;200 msec time out
61236   ef34          sub a
61237   ef35          stc         ;carry=1
```

```
61238   ef36          pop b
61239   ef37          ret
61240   ef38    ptd:  in ptrd
61242   ef3a          pop b
61243   ef3b          ret
61244   ef3c
61244   ef3c    ttyr: push b
61245   ef3d          mvi b,255
61247   ef3f    ttr:  in ttys
61249   ef41          ani 16
61251   ef43          jnz ttd
61254   ef46          dcr c
61255   ef47          jnz ttr
61258   ef4a          dcr b
61259   ef4b          jnz ttr
61262   ef4e          sub a
61263   ef4f          stc
61264   ef50          pop b
61265   ef51          ret
61266   ef52    ttd:  in ttyd
61268   ef54          pop b
61269   ef55          ret
61270   ef56    ptp:  in ptps
61272   ef58          ani x,80
61274   ef5a          jz ptp
61277   ef5d          mov a,c
61278   ef5e          out ptpd
61280   ef60          ret
61281   ef61
61281   ef61
61281   ef61    ;intel loader
61281   ef61
61281   ef61    inld: mvi a,x,ff
61283   ef63          call tset
61286   ef66          sub a
61287   ef67          mov h,a
61288   ef68          mov l,a
61289   ef69          mov d,a
61290   ef6a          mov e,a
61291   ef6b          mvi c,x,11
61293   ef6d          call ssr
61296   ef70          call ri
61299   ef73    nrec: call getc
61302   ef76          cpi x,3a
61304   ef78          jnz nrec
61307   ef7b          call byte
61310   ef7e          mov e,b
61311   ef7f          call byte
61314   ef82          mov h,b
61315   ef83          call byte
61318   ef86          mov l,b
61319   ef87          call byte
61322   ef8a          mov a,b
61323   ef8b          sub a
61324   ef8c          jz nbyt
61327   ef8f          mov c,b
```

```
61328   ef90          mvi b,berr
61330   ef92          jmp erro
61333   ef95    nbyt:sub a
61334   ef96          cmp e
61335   ef97          jz erec
61338   ef9a          call byte
61341   ef9d          mov m,b
61342   ef9e          mov a,m
61343   ef9f          cmp b
61344   efa0          mov c,b
61345   efa1          mvi b,merr
61347   efa3          jnz erro
61350   efa6          inx h
61351   efa7          dcr e
61352   efa8          jmp nbyt
61355   efab    erec:call byte
61358   efae          sub a
61359   efaf          cmp d
61360   efb0          jz nrec
61363   efb3          mov c,d
61364   efb4          mvi b,cerr
61366   efb6          jmp erro
61369   efb9    ;subroutine getc
61369   efb9    getc:call ri
61372   efbc          jnc nto
61375   efbf          mvi b,toer
61377   efc1          mvi c,0
61379   efc3          jmp erro
61382   efc6    nto: ani x,7f
61384   efc8          ret
61385   efc9
61385   efc9    ;subroutine nipp
61385   efc9    nipp:call getc
61388   efcc          mov c,a
61389   efcd          rlc
61390   efce          rlc
61391   efcf          jc lett
61394   efd2          mov a,c
61395   efd3          ani x,0f
61397   efd5          ret
61398   efd6    lett:mov a,c
61399   efd7          ani x,0f
61401   efd9          adi 9
61403   efdb          ret
61404   efdc
61404   efdc    ;subroutine byte
61404   efdc    byte:call nipp
61407   efdf          rlc
61408   efe0          rlc
61409   efe1          rlc
61410   efe2          rlc
61411   efe3          ani x,f0
61413   efe5          mov b,a
61414   efe6          call nipp
61417   efe9          ora b
61418   efea          mov b,a
```

```
61419   efeb            add d
61420   efec            mov d,a
61421   efed            ret
61422   efee
61422   efee    ;subroutine erro
61422   efee    erro:mvi a,x,13
61424   eff0            call ssr
61427   eff3            mov d,b
61428   eff4            mov e,c
61429   eff5            lxi b,x,ffff
61432   eff8            call x,f02c
61435   effb
```

symbols used

	0	
b	0	
c	1	
d	2	
e	3	
h	4	
k	61435	
l	5	
m	6	
sp	6	
psw	6	
berr	1	0001
byte	61404	efdc
cerr	3	0003
ci	61060	ee84
co	61098	eeaa
csts	61153	eee1
driv	61060	ee84
erec	61355	efab
erro	61422	efee
ferr	1	0001
getc	61369	efb9
ichk	61175	eef7
inld	61281	ef61
iosb	0	0000
iset	61179	eefb
lett	61398	efd6
lo	61134	eece
merr	2	0002
nbyt	61333	ef95
nipp	61385	efc9
nrec	61299	ef73
nto	61382	efc6
po	61115	eebb
ptd	61240	ef38
ptp	61270	ef56
ptpd	12	000c
ptps	13	000d
ptr	61218	ef22
ptrd	12	000c
ptrs	13	000d
ptt	61221	ef25
ri	61077	ee95
ssr	61184	ef00
tbrk	61167	eeef
toer	0	0000
tset	61193	ef09
ttd	61266	ef52
ttr	61247	ef3f
ttyd	10	000a
ttyi	61197	ef0d
ttyo	61207	ef17
ttyr	61244	ef3c
ttys	11	000b

end 67