Errata sheet for the IMS B002

Coding switch

There is an error on page 14 of the User manual; the board boots from link when switch 6 of the coding switch near to the edge connector is set to OFF. Full details of the coding switch settings are as follows:

Switch	Signal	OFF	ON	Standard
1	Link0Special		Standard	ON
2	Link123Special		Standard	ON
3	DisablePLL			OFF
4	LINKSPECIAL			ON
5	ENABLE TEST RAM			ON
6	Boot source	Link	ROM	ON
7	DISABLE INT RAM			ON
8	Parity	See below		ON

Switch 3 must be OFF and switches 5 and 7 must be ON.

When the Parity switch is ON, a parity error is reported on the LEDs, but does not generate an error on UpError; when the switch is OFF a parity error is propagated to UpError.

Memory map

The following memory map may be useful:

7FFF	FFFF	EPROM
7FFE	0000	EPROM
8038	0000	Parity error read
8028	0000	PIO
8020	0000	DUART registers base → 0,4,8,C,10,14,18,1C, 20,24,28,2C,30,34,38,3C
801F	FFFF	Top of 2 Mbytes
800F	FFFF	Top of 1 Mbyte
8000	0800	Bottom of external RAM
8000	07FF	Top of on-chip RAM for T414
8000	0000	Bottom of on-chip RAM

For details of the DUART registers, see the 2681 data sheet.

Transputer

The IMS T414A used on these boards has two bugs in byte input and output, a bug in the timer and a bug in the delayed timer input.

Byte input/output

The first problem may be observed if an attempt is made to input a single byte from a link. The second problem may be observed if the byte position within the word for byte slice input is different from that for the corresponding byte slice output. The problem is avoided if all byte slice communications commence on word boundaries. The following two procedures should be used in preference to the predefined procedures BYTE.SLICE.INPUT and BYTE.SLICE.OUTPUT:

```
PROC bugfix.byte.slice.input (CHAN  c,
       VAR vec[], VALUE start.w, bytes) =
  IF   -- start.w is a word offset
    bytes = 1
      VAR temp[BYTE 2] :
      SEQ
        BYTE.SLICE.INPUT (c, temp, 0, 2)
        vec[BYTE start.w*4] := temp[BYTE 0]
    TRUE
      BYTE.SLICE.INPUT (c, vec,
              start.w*4, bytes):

PROC bugfix.byte.slice.output (CHAN  c,
       VALUE vec[], VALUE start.w, bytes) =
  IF   -- start.w is a word offset
    bytes = 1
      VAR temp[BYTE 2] :
      SEQ
        temp[BYTE 0] := vec[BYTE start.w*4]
        BYTE.SLICE.OUTPUT (c, temp, 0, 2)
    TRUE
      BYTE.SLICE.OUTPUT (c, vec,
              start.w*4, bytes):
```

Timer ticks and timeslice period

Both the low priority timer and the high priority timer of the transputer on this board tick every 1.6 microseconds. The timeslice period is 1024x1.6 microseconds.

The transputer will be corrected as follows:

the high priority timer will tick every 1.0 microseconds;
the low priority timer will tick every 64 microseconds;
the timeslice period will be 5x1024 cycles (not 4096 cycles as stated in the transputer reference manual) of ClockIn (about 1 millisecond).

Note that 15 625x64 = 1 000 000, so that the low priority timer will be able to time seconds exactly.

Note also that a process may run for between one and two timeslice periods.

Delayed timer input

To ensure correct operation of delayed timer input the following occam process

```
TIME ? AFTER absolute.time
```

should be replaced by a call to the following procedure:

```
PROC time.after (VALUE abs.time) =
  VAR delta.time, wait.time :
  VAR now, dummy :
  SEQ
    TIME ? now
    LONGDIFF (dummy, delta.time,
              abs.time, now, 0)
    IF
      (delta.time <= 0) OR (delta.time > 5)
        wait.time := abs.time
      TRUE
        LONGSUM (dummy,wait.time,now,5,0)
    TIME ? AFTER wait.time :
```

Timeout with channel input

An alternative process with a channel input and a time out option, as in the following example,

```
ALT
  c ? datum
    ...  process input
  TIME ? AFTER absolute.time
    ...  process time out
```

can be replaced by the following procedure and its instance:

```
 PROC chan.time.out (CHAN  c,
                     VALUE abs.time,
                     VAR   time.out, datum) =
   VAR delta.time, wait.time :
   VAR now, dummy :
   SEQ
     TIME ? now
     LONGDIFF (dummy, delta.time,
               abs.time, now, 0)
     IF
       (delta.time <= 0) OR (delta.time > 5)
         wait.time := abs.time
       TRUE
         LONGSUM (dummy,wait.time,now,5,0)
     ALT
       c ? datum
         time.out := FALSE
       TIME ? AFTER wait.time
         time.out := TRUE :
VAR time.out :
SEQ
  chan.time.out (c, absolute.time,
                 time.out, datum)
  IF
    time.out
      ...  process time out
    TRUE
      ...  process input
```

Restriction on delayed timer input

There is a further delayed timer input restriction in that only one delayed timer input may be in operation at any one time, and this input must be made by a high priority process.

It is strongly recommended that the above timing procedures are used in only one process, at high priority, in a program.