Kirk Bailey
Logical Systems
742 S.W 4th St
CORVALLIS
OREGON
97333

Dear Kirk,

Please find enclosed some tables listing the effects of various instructions on the C, D, E registers on the transputer.

The problems are as follows:-

The transputer instruction set is specified in the compiler writers guide, and in that spec, many of the instructions do not define the final state of the C register. Therefore a future revision of the transputer, using the same instruction spec, may leave the C register differently, if the INMOS design tools are updated.

The D register is never saved on interrupt. I define interupt for this purpose as a de-schedule of a low priority process for the purpose of running a high priority one...be it comms, event or what. This means that a compiler can only make use of it if either the job runs at high priority, or there are no high priority jobs in the system.

The E register is generally not saved on interrupt, but is saved if the interrupt is caused by the current (low) process executing an in or out instruction on a soft channel where a high priority process is waiting.

Please note that if your compiler is used to create code for ROMs, and has used this information, there is a severe risk that if the CPU is changed under maintenance in ten years time, the target system will not work, due to the then available T414x using the CDE registers differently. It would still be a T414, as no information published by INMOS (datasheet and Compiler writers guide) would be invalid.

Best of luck. This information will be given to compiler writers under strict control. Please treat it as confidential and do not pass it on.

Yours sincerely

Philip 5/Oct/87

Philip G Mattos

ps There is no page 3 !!
Page 2 overflowed so I reprinted only pages 1 and 2

The CReg is affected as follows in the primary instructions :
(T414B, T212A , T800C, M212B)

(AReg means AReg value at start of instruction)
(AReg' means AReg value at end of instruction)

Instruction	
ldl	CReg' := BReg
stl	CReg' := CReg (no change)
ldlp	CReg' := BReg
ldnl	CReg' := CReg (no change)
stnl	CReg' := CReg (no change)
ldnlp	CReg' := CReg (no change)
eqc	CReg' := CReg (no change)
ldc	CReg' := BReg
adc	CReg' := CReg (no change)
j	CReg' := CReg (no change) if timesliced or not
cj	CReg' := CReg (no change)
call	CReg' := CReg (no change)
ajw	CReg' := CReg (no change)
pfix	CReg' := CReg (no change)
nfix	CReg' := CReg (no change)

In the following secondary instructions 'start' is the instruction executed by asserting the Reset input, 'chrdyrun' (ChannelReadyRun) is the micro-interrupt routine which handles requests from the links for a channel alternative or to run a process, and 'timer' it the micro-interrupt routine which is called by the timer alarms being asserted.

The CReg is affected as follows in the secondary instructions :

```
rev             CReg' := CReg (no change)
dup             CReg' := BReg
ret             CReg' := CReg (no change)
ldpi            CReg' := CReg (no change)
gajw            CReg' := CReg (no change)
gcall           CReg' := CReg (no change)
mint            CReg' := BReg
lend            CReg' := incremented loop index, 0 at end of down count
csub0           CReg' := CReg (no change)
ccnt1           CReg' := CReg (no change)
seterr          CReg' := CReg (no change)
testerr         CReg' := BReg
stoperr         CReg' := CReg (no change)
clrhalterr      CReg' := CReg (no change)
sethalterr      CReg' := CReg (no change)
testhalterr     CReg' := BReg
bsub            CReg' := CReg (no change)
wsub            CReg' := CReg (no change)
bcnt            CReg' := CReg (no change)
wcnt            CReg' := BReg
lb              CReg' := CReg (no change)
sb              CReg' := 0
and             CReg' := CReg (no change)
or              CReg' := CReg (no change)
xor             CReg' := CReg (no change)
not             CReg' := CReg (no change)
shl             CReg' := CReg (no change)
shr             CReg' := CReg (no change)
add             CReg' := CReg (no change)
sub             CReg' := CReg (no change)
mul             CReg' := CReg (no change)
fmul            CReg' := less significant word of long result,
                         before rounding to a single word
div             CReg' := remainder from div
rem             CReg' := AReg' (remainder)
gt              CReg' := CReg (no change)
diff            CReg' := CReg (no change)
sum             CReg' := CReg (no change)
prod            CReg' := CReg (no change)
crcword         CReg' := AReg' (CRCResult)
crcbyte         CReg' := AReg' (CRCResult)
bitcnt          CReg' := CReg (no change)
bitrevword      CReg' := CReg (no change)
bitrevnbits     CReg' := CReg (no change)
wsubdb          CReg' := CReg (no change)
xword           CReg' := CReg (no change)
cword           CReg' := CReg (no change)
xdble           CReg' := BReg
csngl           CReg' := CReg (no change)
```

Secondary Instructions /C Register continued

```
ladd            CReg' := CReg (no change)
lsub            CReg' := CReg (no change)
lsum            CReg' := CReg (no change)
ldiff           CReg' := CReg (no change)
lmul            CReg' := BReg'
ldiv            CReg' := BReg'
lshl            CReg' := BReg'
lshr            CReg' := BReg'
norm            CReg' := normalise count
move            CReg' := AddressOfStartOfSourceVector +
                         (NumberOfWordReads * BytesPerWord)
move2dinit      CReg' := CReg (no change)
move2dall       CReg' := AddressOfStartOfFinalSourceArrayRow +
                         (NumberOfWordReadsInRow * BytesPerWord)
move2dnonzero   CReg' := AddressOfStartOfFinalSourceArrayRow +
                         (NumberOfWordReadsInRow * BytesPerWord)
move2dzero      CReg' := AddressOfStartOfFinalSourceArrayRow +
                         (NumberOfWordReadsInRow * BytesPerWord)
startp          CReg' := CReg (no change)
endp            CReg' := CReg (no change) if not returning from high priority OR
                CReg' := (CReg of interrupted process) if restoring low priority
runp            CReg' := CReg (no change)
stopp           CReg' := CReg (no change) if not returning from high priority OR
                CReg' := (CReg of interrupted process) if restoring low priority
ldpri           CReg' := BReg
chrdyrun        CReg' := CReg (no change)
in              CReg' := (as for stopp) if hard channel, or soft channel not
                         ready to output OR
                CReg' := AddressOfStartOfSourceVector +
                         (NumberOfWordReads * BytesPerWord) if soft
                         channel ready to output
out             CReg' := (as for stopp) if hard channel, or soft channel not
                         ready to input OR
                CReg' := AddressOfStartOfSourceVector +
                         (NumberOfWordReads * BytesPerWord) if soft
                         channel ready to input
outword         CReg' := (as for stopp) if hard channel, or soft channel not
                         ready to input OR
                CReg' := AddressOfStartOfSourceVector +
                         (NumberOfWordReads * BytesPerWord) =
                         Wptr + BytesPerWord
                         if soft channel ready to input
outbyte         CReg' := (as for stopp) if hard channel, or soft channel not
                         ready to input OR
                CReg' := AddressOfStartOfSourceVector +
                         (NumberOfWordReads * BytesPerWord) =
                         Wptr + BytesPerWord
                         if soft channel ready to input
resetch         CReg' := CReg (no change)
```

Secondary instructions /C register continued again

```
alt             CReg' := CReg (no change)
altwt           CReg' := CReg (no change) if an enabled process is ready OR
                         (as for stopp) if ALT to wait for an enabled process
altend          CReg' := CReg (no change)
enbs            CReg' := CReg (no change)
diss            CReg' := CReg (no change)
enbs            CReg' := CReg (no change)
enbc            CReg' := CReg (no change)
disc            CReg' := CReg (no change)
timer           CReg' := CReg (no change)
ldtimer         CReg' := BReg
tin             CReg' := CReg (no change) if time to wait has already passed OR
                         (as for stopp) if process has to wait on timer queue
talt            CReg' := CReg (no change)
taltwt          CReg' := (as for altwt) if ALT need not wait on timer OR
                         (as for tin) if ALT must wait on timer
enbt            CReg' := CReg (no change)
dist            CReg' := CReg (no change) if ALT did not wait on timer OR
                         (TimeNotSet token) if process had to be removed
                         from timer queue
unpacksn        CReg' := type of floating point number (0 to 3)
postnormsn      CReg' := (exponent of floating point number)
roundsn         CReg' := Breg (floating point fraction) if  CReg >= #FF OR
                         (CReg >> 9) if  CReg < #FF
ldinf           CReg' := Breg
cflerr          CReg' := CReg (no change)
fseterror       CReg' := CReg (no change)
fclearerror     CReg' := CReg (no change)
fprev           CReg' := CReg (no change)
fpdup           CReg' := CReg (no change)
fpldzerosn      CReg' := CReg (no change)
fpldzerodb      CReg' := CReg (no change)
fabs            CReg' := CReg (no change)
fsqrtfirst      CReg' := CReg (no change)
fsqrtstep       CReg' := CReg (no change)
fsqrtend        CReg' := CReg (no change)
fpchkerror      CReg' := CReg (no change)
fptesterror     CReg' := BReg
frz             CReg' := CReg (no change)
frn             CReg' := CReg (no change)
frp             CReg' := CReg (no change)
frm             CReg' := CReg (no change)
fpgt            CReg' := BReg
fpeq            CReg' := BReg
fpnan           CReg' := BReg
fpnotfinite     CReg' := BReg
fpordered       CReg' := BReg
```

Secondary Instructions /C register continued for the third time

```
fpstnlsn        CReg' := CReg (no change)
fpstnldb        CReg' := CReg (no change)
fpstnli32       CReg' := CReg (no change)
fr32tor64       CReg' := CReg (no change)
fr64tor32       CReg' := CReg (no change)
fdecexpby32     CReg' := CReg (no change)
fincexpby32     CReg' := CReg (no change)
fdecexpby1      CReg' := CReg (no change)
fincexpby1      CReg' := CReg (no change)
fpadd           CReg' := CReg (no change)
fpsub           CReg' := CReg (no change)
fpmul           CReg' := CReg (no change)
fpdiv           CReg' := CReg (no change)
fpremfirst      CReg' := BReg
fpremstep       CReg' := BReg
fpi32tor32      CReg' := CReg (no change)
fpi32tor64      CReg' := CReg (no change)
fpb32tor64      CReg' := CReg (no change)
fprtoi32        CReg' := CReg (no change)
fpint           CReg' := CReg (no change)
fpldnlsn        CReg' := CReg (no change)
fpldnldb        CReg' := CReg (no change)
fpldnlsni       CReg' := CReg (no change)
fpldnldbi       CReg' := CReg (no change)
fpldnladdsn     CReg' := CReg (no change)
fpldnladddb     CReg' := CReg (no change)
fpldnlmulsn     CReg' := CReg (no change)
fpldnlmuldb     CReg' := CReg (no change)
fpentry         CReg' := CReg (no change)
start           CReg' := AddressOfBootInputChannel if boot from link OR
                         no change or power-up value if boot from rom
testpranal      CReg' := BReg
saveh           CReg' := CReg (no change)
savel           CReg' := CReg (no change)
sthf            CReg' := CReg (no change)
sthb            CReg' := CReg (no change)
stlf            CReg' := CReg (no change)
stlb            CReg' := CReg (no change)
sttimer         CReg' := CReg (no change)
teststd         CReg' := CReg (no change)
testste         CReg' := CReg (no change)
teststs         CReg' := CReg (no change)
testldd         CReg' := CReg (no change)
testlde         CReg' := CReg (no change)
testlds         CReg' := CReg (no change)
testhardchan    CReg' := CReg (no change)
```

D,E register access

The testing of DReg and EReg is performed by using instructions which access them from the stack. The DReg is accessed by the following test instructions :

```
testldd :   opcode = #25          byte sequence = #22 #F5
SEQ
  CReg := BReg
  BReg := AReg
  AReg := DReg

teststd :   opcode = #28          byte sequence = #22 #F8
SEQ
  DReg := AReg
  AReg := BReg
  BReg := CReg
```

The EReg is accessed by the following test instructions :

```
testlde :   opcode = #24          byte sequence = #22 #F4
SEQ
  CReg := BReg
  BReg := AReg
  AReg := EReg

testste :   opcode = #27          byte sequence = #22 #F7
SEQ
  EReg := AReg
  AReg := BReg
  BReg := CReg
```

The opcodes are the same on the IMS T800, IMS T414, and IMS T212.

Instructions changing the D register... different on different machines

The following instructions make use of the registers, where 'start' is the instruction executed by asserting the Reset input, 'chrdyrun' (ChannelReadyRun) is the micro-interrupt routine which handles requests from the links for a channel alternative or to run a process, and 'timer' it the micro-interrupt routine which is called by the timer alarms being asserted :

IMS T800 DReg

Instruction	Increments	Purpose
cword	No	temporary
dist	No	"after" test
div	Yes	loop count
enbt	No	"after" test
fmul	Yes	loop count
fpldnldbi	Yes	share ucode
fpldnlsni	Yes	share ucode
ldiv	Yes	loop count
lmul	Yes	loop count
lshl	Yes	loop count
lshr	Yes	loop count
move	Yes	read count
move2dall	Yes	read count
move2dnonzero	Yes	read count
move2dzero	Yes	read count
mul	Yes	loop count
norm	Yes	loop count
prod	No	accumulator
rem	Yes	loop count
sb	No	move logic
start	Yes	link poll
taltwt	No	"after" test
testldd	No	test
teststd	No	test
timer	No	queue address
tin	No	"after" test
wcnt	Yes	shift count

IMS T414 DReg

Instruction	Increments	Purpose
cflerr	No	temporary
cword	No	temporary
dist	No	"after" test
div	Yes	loop count
enbt	No	"after" test
fmul	Yes	loop count
ldiv	Yes	loop count
lmul	Yes	loop count
lshl	Yes	loop count
lshr	Yes	loop count
move	Yes	read count
mul	Yes	loop count
norm	Yes	loop count
postnormsn	Yes	loop count
prod	No	accumulator
rem	Yes	loop count
roundsn	Yes	loop count
sb	No	move logic
start	Yes	link poll
taltwt	No	"after" test
testldd	No	test
teststd	No	test
timer	No	queue address
tin	No	"after" test
unpacksn	Yes	loop count
wcnt	Yes	shift count

IMS T212 DReg

Instruction	Increments	Purpose
cword	No	temporary
dist	No	"after" test
div	Yes	loop count
enbt	No	"after" test
ldiv	Yes	loop count
lmul	Yes	loop count
lshl	Yes	loop count
lshr	Yes	loop count
move	Yes	read count
mul	Yes	loop count
norm	Yes	loop count
prod	No	accumulator
rem	Yes	loop count
sb	No	move logic
start	Yes	link poll
taltwt	No	"after" test
testldd	No	test
teststd	No	test
timer	No	queue address
tin	No	"after" test
wcnt	Yes	shift count

IMS T800 EReg

Instruction	Purpose
chrdyrun	process to run
disc	current WDesc
enbc	current WDesc
gt	alu result
in	process to run
j	current WDesc
lend	current WDesc
move	process to run
move2dall	process to run
move2dnonzero	process to run
move2dzero	process to run
out	process to run
runp	process to run
shl	alu result
start	temporary
startp	process to run
taltwt	process to run
testlde	test
testste	test
timer	process to run
tin	"after" test

IMS T414 EReg

Instruction	Purpose
chrdyrun	process to run
disc	current WDesc
enbc	current WDesc
gt	alu result
in	process to run
j	current WDesc
lend	current WDesc
move	process to run
out	process to run
roundsn	temporary
runp	process to run
shl	alu result
start	temporary
startp	process to run
taltwt	process to run
testlde	test
testste	test
timer	process to run
tin	"after" test

IMS T212 EReg

Instruction	Purpose
chrdyrun	process to run
disc	current WDesc
enbc	current WDesc
gt	alu result
in	process to run
j	current WDesc
lend	current WDesc
move	process to run
out	process to run
runp	process to run
shl	alu result
start	temporary
startp	process to run
taltwt	process to run
testlde	test
testste	test
timer	process to run
tin	"after" test