```
**************************************************
*                                                *
*  INTEL RELEASES 8051 FAMILY O/S TO PUBLIC DOMAIN  *
*                                                *
**************************************************
```

As a service to the many people who have been requesting it, Intel Corporation has placed its 8051 family Real-Time Multitasking Operating System into the public domain.

The Distributed Control Executive, iDCX 51, was introduced in 1984 as a tool for designers of factory automation and robotic control. It was the basis for a distributed control product line using the BITBUS (IEEE 1118) serial bus.

The source code (in 8051 assembler language) for the iDCX 51 is available for anonymous FTP on the Internet server named:

vab02.larc.nasa.gov

under the path name:

tools/dcx51.tar

Documentation for the operating system is available as:

iDCX 51 Distributed Control Executive User's Guide for Rel. 2.0
Order Number 460367-001
Cost (U.S.) $20.00 + postage

from:

Intel Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641 USA
Telephone 1-800-548-4725

```
**************************************************
*                                                *
*   IMPORTANT (NON-) SUPPORT NOTE !!!            *
*                                                *
**************************************************
```

Although Intel has decided to provide this source to the public at no charge, it CAN NOT afford to provide free customer support. Having the source means if you find a bug, you can fix it yourself, and post the fix to the comp.sys.intel newsgroup. Please DO NOT call for Intel customer support on this product!

```
*****************************************************
*                                                   *
*   NOMENCLATURE                                    *
*                                                   *
*****************************************************
```

DCX51 (or iDCX 51)

A Distributed Real-Time Multitasking Executive for the 8051 family of microcontrollers. The current release is R2.1.

DCM44 (or iDCM 44)

A communications task for iDCX 51. When installed as task 0, and run on the 8044 flavor of the 8051 family (the 8044 contains an SDLC unit instead of RS-232C), it provides BITBUS (IEEE 1118) message-passing ability between up to 250 microcontroller nodes. It also provides the RAC (Remote Access and Control) functions for remote message-based operations such as read/write-memory, create/delete DCX task, etc.

You do not *have to* use DCM44 if you only want DCX51, and in fact you *cannot* run DCM44 "AS IS" on processors other than the 8044/8344/8744.

DCS110 (or "BITWARE")

This is a developer's package which contains the files needed to program iDCX 51 tasks *assuming* that you already have an 8044 BEM (the flavor of the BITBUS component with DCX51 + DCM44 already masked in ROM). It includes the DCX51I interface library (to get to ROM entry points) but not the DCX51 library (which actually contains the O/S). This product also contains the file DCM44 which is an ICE-5100 loadable copy of DCM44.

DCS120

Same as DCS110 but no ICE-loadable DCM44 image.

```
*****************************************************
*                                                   *
*   INSTALLATION INSTRUCTIONS                       *
*                                                   *
*****************************************************
```

1) On your UNIX machine, un-compress and un-tar the package:

```
compress -d dcx51.tar.Z
mkdir dcx51
cd dcx51
```

```
   tar xvf ../dcx51.tar

2) Using Microsoft Networks, PC-NFS, or some other
   networking, mount the UNIX drive on your DOS PC
   and copy all the un-tar-ed files to the PC, in
   a manner similar to the following example:

   net use x: \\unixbox\mylogon mypasswd
   mkdir dcx51
   xcopy x:\dcx51 dcx51 /s /e
   net use x: /d

*****************************************************
*                                                   *
*   REGENERATION INSTRUCTIONS                       *
*                                                   *
*****************************************************

1) Make sure you have the ASM-51 product installed, and
   accessible via PATH.  If you don't have this
   expensive Intel product, third-party 8051 assemblers
   might work (but *I* haven't tried them).

2) Rebuild iDCX51 using batch file R2AGEN.BAT:

   cd dcx51\dcx51src.21
   r2agen

   The resultant files are in the dcx51\dcx51src.21\rel21
   subdirectory.  These files should be identical to the
   contents of the DCX51SU product floppy.

3) Rebuild iDCM44 using batch file DCMGEN.BAT:

   cd dcx51\dcm44src.21
   dcmgen

   The resultant files are in the dcx51\dcx51src.21\dcs110
   subdirectory.  These files should be identical to the
   contents of the DCS110SU product floppy.

*****************************************************
*                                                   *
*   ACKNOWLEDGMENTS                                 *
*                                                   *
*****************************************************

   This product was developed primarily by Ron Smith
   and Rick McAlister, both of whom did a great job,
   but want nothing to do with it anymore.

   Placed in the public domain, with permission,
   by Jim Trethewey (flamer@mithril.intel.com).
```

```
rem  **********
rem  R2AGEN.BAT *
rem  **********

;
rem  TITLE:        iDCX 51 Nucleus: System Generation Submit File

rem  RELEASE:      2.0

rem  DESCRIPTION: This module provides all of the commands required
rem               for DCX 51 system generation.

erase rel21
rmdir rel21
mkdir rel21

asm51 r2cr.a51
asm51 r2data.a51
asm51 r2fsm.a51
asm51 r2imlg.a51
asm51 r2init.a51
asm51 r2inlg.a51
asm51 r2msg.a51
asm51 r2rqcr.a51
asm51 r2rqde.a51
asm51 r2rqdi.a51
asm51 r2rqfi.a51
asm51 r2rqse.a51
asm51 r2rqst.a51
asm51 r2rqwt.a51
asm51 r2tmgt.a51
asm51 r2tick.a51
asm51 r2tinv.a51
asm51 r2tlag.a51
asm51 r2vec.a51
asm51 r2sys.a51
asm51 r2utl.a51
asm51 r2plmi.a51
asm51 r2mem.a51
asm51 r2etim.a51

erase dcx51.lib
lib51 create dcx51.lib
lib51 add r2vec.obj, r2data.obj, r2cr.obj, r2init.obj, r2rqcr.obj to dcx51.li
lib51 add r2rqde.obj, r2rqse.obj, r2rqwt.obj, r2utl.obj, r2msg.obj to dcx51.l
lib51 add r2tmgt.obj, r2tinv.obj, r2tlag.obj, r2mem.obj, r2etim.obj to dcx51.
lib51 add r2inlg.obj, r2rqst.obj, r2imlg.obj, r2rqfi.obj, r2fsm.obj to dcx51.
lib51 add r2rqdi.obj, r2tick.obj to dcx51.lib

erase dcx51i.lib
lib51 create dcx51i.lib
lib51 add r2cr.obj, r2sys.obj, r2plmi.obj to dcx51i.lib

copy dcx51.lib rel21
copy dcx51i.lib rel21
copy dcx51a.ext rel21
```

```
copy dcx51a.lit rel21
copy dcx51a.mac rel21
copy dcxb0p.ext rel21
copy dcxb1p.ext rel21
copy dcxb2p.ext rel21
copy dcxb3p.ext rel21
copy dcx51p.lit rel21
copy dcx51a.cfg rel21
copy dcx51a.add rel21
copy samp1.a51 rel21
copy samp2.a51 rel21

rem  End of Product Generation
```

```
;***************
;* SAMP1.A51   *
;***************
;
; iDCX 51 Sample Configuration: w/ 2 Tasks
;

$nolist
$include(dcx51a.ext)
$include(dcx51a.lit)
$include(dcx51a.mac)
$list

;
; Sample Application: Consists of Task 0 and Task 1, which do the following:
;
;  Task 0:                      Loop
;                                 Wait on a timer 1 Overflow interrupt
;                                 Toggle the green LED
;                               End Loop

;  Task 1:                      Loop
;                                 Wait on the next 1-second time interval
;                                 Toggle the red LED
;                               End Loop
;

SAMP1_CODE_SEG     SEGMENT   CODE
                   RSEG      SAMP1_CODE_SEG



ITD0:
%ITD(TASK0, 8, 081H, 000B, 02H, 01000B, ITD1)    ;TASK0    = Beginning PC Addr.
                                                  ;8        = Stack length
                                                  ;081H     = Function ID
                                                  ;000B     = Choose any registe
                                                  ;           bank
                                                  ;02H      = Task priority
                                                  ;01000B   = Assign Timer 1 int
                                                  ;           to this task
                                                  ;ITD1     = Next ITD

ITD1:
%ITD(TASK1, 8, 082H, 000B, 01H, 00000B, 1000H)   ;TASK1    = Beginning PC Addr.
                                                  ;8        = Stack length
                                                  ;082H     = Function ID
                                                  ;000B     = Choose any registe
                                                  ;           bank
                                                  ;01H      = Task priority
                                                  ;00000B   = Assign no interrup
                                                  ;           to this task
                                                  ;1000H    = Terminate ITD chai
                                                  ;           unless another ITD
                                                  ;           exists at 1000H
```

```
;
   The iDCX 51 Configuration Constants are defined here
;

SAMP1_IDATA_SEG  SEGMENT   IDATA
                 RSEG      SAMP1_IDATA_SEG

                 PUBLIC  RQFIRSTITD, RQRAMIDD, RQCLOCKPRIORITY, RQCLOCKTICK
                 PUBLIC  RQSYSBUFSIZE, RQMEMPOOLADR, RQMEMPOOLLEN

RQFIRSTITD       EQU  ITD0                ;ADDRESS OF FIRST TASK DESCRIPTOR
RQRAMIDD         EQU  7FF0H               ;ADDRESS OF OPTIONAL RAM-BASED IDD
RQCLOCKPRIORITY  EQU  5                   ;PRIORITY OF SYSTEM CLOCK
RQCLOCKTICK      EQU  -10000              ;(NEGATIVE OF) CLOCK CYCLES PER TI
RQSYSBUFSIZE     EQU  20                  ;FREE SPACE BUFFER SIZE
RQMEMPOOLADR:    DS   45                  ;SYSTEM MEMORY POOL START ADDR
RQMEMPOOLLEN     EQU  $-RQMEMPOOLADR      ;SYSTEM MEMORY POOL LENGTH

; End of Configuration Constants

; Equates
;
GREEN_LED           EQU    91H
RED_LED             EQU    90H
;

; The code for Tasks 0 and 1 follows
;
                 RSEG      SAMP1_CODE_SEG

TASK0:
  CLR   GREEN_LED                        ;Turn off green LED
  CLR   TR1                              ;Stop Timer 1
  ORL   TMOD,#010H                       ;Set Timer 1 to Mode 1
  SETB  TR1                              ;Start Timer 1
LOOP0:
  MOV   A,#INTERRUPT_EVENT               ;Wait for next interrupt
  MOV   B,#WAIT_FOREVER                  ; indefinitely
  CALL  RQWAIT                           ;
  CPL   GREEN_LED                        ;Toggle green LED
  SJMP  LOOP0                            ;Repeat Loop


TASK1:
  CLR   RED_LED                          ;Turn off red LED
  MOV   A,#100                           ;Set for 1 second intervals
  CALL  RQSETINTERVAL                    ;
LOOP1:
  MOV   A,#INTERVAL_EVENT                ;Wait for next time interval
  MOV   B,#WAIT_FOREVER                  ; indefinitely
  CALL  RQWAIT                           ;
  CPL   RED_LED                          ;Toggle red LED
  SJMP  LOOP1                            ;Repeat Loop

 ND
```

```
;**************
;* R2VEC.A51  *
 **************
;
; COPYRIGHT 1983, 1987 INTEL CORPORATION
;
NAME ROOT

;*****************************************************************************/
;*
;* TITLE:        iDCX 51 Nucleus: Interrupt and System Call Vectoring
;*
;* RELEASE:      2.0
;*
;* DESCRIPTION: This module contains the jump tables for both the hardware
;*               interrupts and the DCX 51 system calls.  Note that a section
;*               of code memory is reserved for future expansion of the
;*               number of interrupts.  This space can be presently overlayed
;*                     by other system code, but caution must be taken not to
;*               overlay the system call jump table following it.
 *
;* UPDATES:
;*
;*****************************************************************************


$nolist
$include(r2sys.inc)
$list

 EXTRN CODE( REQ_INIT,REQCREATETASK,REQDELETETASK,REQGETFUNCTIONIDS)
 EXTRN CODE( REQDISABLEINTERRUPT,REQENABLEINTERRUPT,REQSENDMESSAGE,REQSETINTE
 EXTRN CODE( REQWAIT,REQALLOCATE,REQDEALLOCATE, REQGETMEM, REQRELEASEMEM )
 EXTRN CODE( REQ_EXPORT_LOAD_AND_GO, REQ_INT_LOAD_AND_GO )
 EXTRN CODE( REQ_LOAD_AND_GO, REQ_IMP_LOAD_AND_GO, REQ_MSG_DEQ, REQ_MSG_ENQ )
 EXTRN CODE( REQ_AND_OUT, REQ_OR_IN, REQ_INVOKE_READY_TASK, REQ_TASK_ENQ )
 EXTRN CODE( REQ_SET_TIMER0, REQ_TICK)

DCX51_CODE_SEG   SEGMENT   CODE
                 CSEG      AT 0

                 ORG    000H                        ;ZERO FILL TO 83H
                 DW    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                 DW    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                 DW    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                 DW    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                 DB    0,0,0

                 ORG    000H                         ;RESET
 AJMP   REQ_INIT

 ORG    003H                                        ;EXTERNAL REQUEST 0
 MOV    REQ_TEMP2,#( (0 * 10H) + TEML_INTERRUPT_OMSK )
 MOV    REQ_TEMP1,#REQ_INT_TASKS+0
 SJMP   INT_HANDLER
```

```
                ORG    00BH                        ;INTERNAL TIMER/COUNTER 0
    AJMP   REQ_TICK

                ORG    013H                        ;EXTERNAL REQUEST 1
    MOV    REQ_TEMP2,#( (2 * 10H) + TEML_INTERRUPT_OMSK )
    MOV    REQ_TEMP1,#REQ_INT_TASKS+2
    SJMP   INT_HANDLER

                ORG    01BH                        ;INTERNAL TIMER/COUNTER 1
    MOV    REQ_TEMP2,#( (3 * 10H) + TEML_INTERRUPT_OMSK )
    MOV    REQ_TEMP1,#REQ_INT_TASKS+3
    SJMP   INT_HANDLER

                ORG    023H                        ;INTERNAL SERIAL PORT 1
    MOV    REQ_TEMP2,#( (4 * 10H) + TEML_INTERRUPT_OMSK )
    MOV    REQ_TEMP1,#REQ_INT_TASKS+4
    SJMP   INT_HANDLER

                ORG    02BH                        ;INTERNAL TIMER/COUNTER 2
    MOV    REQ_TEMP2,#( (5 * 10H) + TEML_INTERRUPT_OMSK )
    MOV    REQ_TEMP1,#REQ_INT_TASKS+5

INT_HANDLER:
    %REQ_SAVE_REGS                              ;SAVE RUNNING TASK'S CONTEXT

; Get the task id of the task associated with this interrupt, and
; update the state of that task to reflect the event.

    MOV    DPH,#0                               ;DPTR => REQ_INT_TASKS TABLE
    MOV    DPL,REQ_TEMP1                        ; "
    MOVX   A,@DPTR                              ;A = ID OF INTERRUPT TASK
    MOV    REQ_TEMP1,A                          ;SEE IF THIS TASK IS WAITING ON A
    RL     A                                    ;INTERRUPT EVENT.
    RL     A                                    ; "
    ADD    A,#REQ_EVENTS+2                      ; "
    MOV    DPL,A                                ; "
    MOVX   A,@DPTR                              ; "
    JB     ACC.TEML_INTERRUPT_BMSK,INT_WAITING_ON ;IF SO, BRANCH
    INC    DPL                                  ;IF NOT, UPDATE OCCURRED ENTRY FO
    MOVX   A,@DPTR                              ;TASK WITH INTERRUPT INFO.
    ORL    A,REQ_TEMP2                          ; "
    MOVX   @DPTR,A                              ; "
    AJMP   REQ_INT_LOAD_AND_GO                  ;TASK NOT WAITING ON EVENT

INT_WAITING_ON:
    CLR    A                                    ;CLEAR WAITING ENTRY
    MOVX   @DPTR,A                              ; "
    DEC    DPL                                  ;CLEAR TIMEOUT ENTRY
    MOVX   @DPTR,A                              ; "
    MOV    A,REQ_TEMP1                          ;A = TASK ID
    CJNE   A,RQTASKID,REQ_EXPORT_LOAD_AND_GO    ;WAS TASK PREV. RUNNING?
    MOV    A,REQ_TEMP2                          ;A = RQWAIT RETURN STATUS
    CLR    REQ_FLAGS.IDLE_TASK_RUNNING_BMSK     ;
    MOV    RQPRIORITY,REQ_TEMP6                 ;RESTORE PARTIAL CONTEXT
    DEC    SP                                   ; "
    DEC    SP                                   ; "
```

```
MOV    IE,REQ_TEMP7                    ; "
RETI                                   ;AND PASS CONTROL TO TASK


               CSEG  AT 083H           ;START OF SYSTEM JUMP TABLE
LJMP   REQCREATETASK
LJMP   REQDELETETASK
LJMP   REQGETFUNCTIONIDS
LJMP   REQDISABLEINTERRUPT
LJMP   REQENABLEINTERRUPT
LJMP   REQSENDMESSAGE
LJMP   REQSETINTERVAL
LJMP   REQWAIT
LJMP   REQALLOCATE
LJMP   REQDEALLOCATE
LJMP   REQGETMEM
LJMP   REQRELEASEMEM

END
```