2/27/91

Application Note
Evaluating EDX on the ADS302

This note describes how to get an EDX application example up and running on the ADS302 without access to the source code. The note is aimed at users of the ADS302 wishing to evaluate the EDX kernel. As an example, a multitasking application written in assembly language is given as a basis for the development of more complex applications.

Introduction to EDX

Event Driven Executive (EDX) is a real time multitasking operating system for the Motorola 68000 family of processors. It was written primarily for running the communications software packages, such as X.25 and LAPD, written for the MC68302 IMP. EDX does not support time slicing as found in popular executives such as VRTX etc. Once an EDX task is running, only it can relinquish control and allow another task to run. EDX supports task creation, intertask messaging and allocation of memory partitions. The basic functionality of the EDX makes for an efficient kernel particularly appropriate for running layered protocol software implementations such as X.25. For full details on EDX consult the EDX User's Manual found in the ADS302 documentaion package.

Outline

The example code consists of four basic parts

1. Move the EDX code from EPROM to RAM to demonstrate the portability.
2. Initialize EDX.
3. Create two tasks which will alternatively pass messages to each other.
4. Set the tasks going.

The Code

The code was written in assembler under a cross assembler on the PC, it was then downloaded and run on the ADS302 under 302bug. The code makes use of the bug's trap 15 calls to print messages on the terminal to show progress of the code.

The assembly listing is shown below and should be read in conjunction with the following notes:

A message is output on the terminal to show that the code is running. The Trap 15 facilities of ADS monitor is used for this purpose.

In revision 5 of the ADS EPROMs, the ADS code including the EDX code is located at address $21100c and is approximately 1.5 K bytes long. For demonstration purposes the code

is moved to user RAM starting at address $50000. For other versions of the ADS software, run the help command to find the location of the EDX code.

The two vectors that point to the EDX code and its workspace RAM are loaded in the 68302's exception table at locations $80 and $100 respectively. Address $80 corresponds to the trap 0 vector which is used for all calls to EDX. Address $100 contains a user supplied pointer to the RAM that EDX will use for its internal operation, in the example $40000 has been chosen. The use of these two vectors make for easy integration of EDX into a target environment.

EDX is now initialized by making a trap 0 call. Note that all EDX trap 0 calls have the call type in register d0, the call types are listed in the equates section at the top of the listing. On an initialisation call register a1 is used to point to the configuration table with contains the user defined parameters - max number of tasks, max number of events, max number of partitions, max number of blocks, max stack size and finally the address of a routine called EDX poll. The EDX poll routine is called as control passes between the tasks and is very useful for performing house keeping functions and taking care of interrupts and hardware events which are beyond the scope of EDX. The values chosen for the above parameters effects the amount of RAM required by EDX, which can be calculated by the formulae given in the user's manual.

Two task creation calls are now made to create the tasks task0 and task1. In these calls, a1 is a pointer to the task and d2 contains the task's maximum stack size.

Having created all the tasks that will ever be running in the system we call the multitasking mode, from now on we are in the multitasking mode, the code following this call will never run.

The actual code for task0 and task 1 now follows, these tasks alternately send a message to each other, when waiting for a message the task will halt and the scheduler will than start the other task. Progress of the task is displayed by messages to the terminal.

The code for edxpoll simply puts out a message.

When the example is running, edxpoll can be seen to run between task0 and task1, it also runs on entry to multitasking mode prior to the tasks running. On the ADS board, EDX is configured to provide round robin scheduling which means that there is no concept of task priority. On start up the task with the lowest task id will run first, after which the task that has been waiting the longest time will run next. If desired EDX can be configured at assembly time to implement fixed priorities. In the example, task 0 is seen to run before task 1 because the task id of task 0 is 0.

For the sake of readability the return codes from the EDX calls have not been checked for successful operation. In a real application this would be advisable.

```
04/22/91 05:52:19     EDXNEW    ASM

init    equ     0       16 april 1991
multi   equ     1
tcreate equ     2
tdel    equ     3
pcreate equ     4
send    equ     5
```

```
rec     equ     6
grab    equ     7
gblock  equ     8
rblock  equ     9
rersume equ     10

max_tsk equ     4
max_evn equ     40
max_prt equ     10
max_blk equ     40
ssize   equ     400

tid0    equ     0
tid1    equ     1
tssz    equ     50

cr      equ     $0a
lf      equ     $0d

        org     $30000

start:
        movea.l #$30000,a7
        movea.l #msg2,a0        output alive msg to terminal
        jsr     prt
        move.l  #$21100c,a1     move edx code from rom
        move.l  #$50000,a2                      to ram
        move.l  #512,d1         move 2k bytes to be on safe side
lp1:
        move.l  (a1)+,(a2)+
        dbf     d1,lp1

        move.l  #$50000,$80     trap 0 entry to edx in ram
        move.l  #$40000,$100    ptr to edx workspace ram

        move.l  #init,d0        initialise edx
        move.l  #config,a1
        trap    #0

        move.l  #tcreate,d0     create task 0
        move.l  #tid0,d1
        movea.l #task0,a1
        move.l  #tssz,d2
```

```
        trap    #0

        move.l  #tcreate,d0     create task 1
        move.l  #tid1,d1
        movea.l #task1,a1
        move.l  #tssz,d2
        trap    #0

        move.l  #multi,d0       go multitasking
        trap    #0

        trap #15                return to ads, this code should never run
        dc.w    0

task0:
        movea.l #msg3,a0
        jsr     prt
        jsr     delay
        move.l  #rec,d0
        trap    #0
        move.l  #send,d0
        move.l  #tid1,d1
        move.l  #0,d2           null intertask msg
        trap    #0
        jmp     task0

task1:
        movea.l #msg4,a0
        jsr     prt
        jsr     delay
        move.l  #send,d0
        move.l  #tid0,d1
        move.l  #0,d2           null intertask msg
        trap    #0
        move.l  #rec,d0
        trap    #0
        jmp     task1



edxpoll:
        movea.l #msg1,a0
        jsr     prt
```

```
        rts

prt:
        movem.l d1/a0/a1,-(a7)
        trap    #15
        dc.w    4
        movem.l (a7)+,d1/a0/a1
        rts

delay:
        move.w  #500,d6
del1:
        muls.w  #$0,d7          kill some time
        dbf     d6,del1
        rts

config:
        dc.w    max_tsk
        dc.w    max_evn
        dc.w    max_prt
        dc.w    max_blk
        dc.l    ssize
        dc.l    edxpoll

msg1:

        dc.b    'this is edx poll'
        dc.b    cr
        dc.b    lf
        dc.b    0
        even


msg2:
        dc.b    'edx example running'
        dc.b    cr
        dc.b    lf
        dc.b    0
        even
msg3:
        dc.b    'task 0 running'
        dc.b    cr
        dc.b    lf
```

```
        dc.b    0
        even
msg4:
        dc.b    'task 1 running'
        dc.b    cr
        dc.b    lf
        dc.b    0
        even


        end


PAGE  001  edxnew.asm                    Tue Apr 16 10:15:39
1991


   1 00000000                   init     equ      0            16 april
1991
   2 00000001                   multi    equ      1
   3 00000002                   tcreate  equ      2
   4 00000003                   tdel     equ      3
   5 00000004                   pcreate  equ      4
   6 00000005                   send     equ      5
   7 00000006                   rec      equ      6
   8 00000007                   grab     equ      7
   9 00000008                   gblock   equ      8
  10 00000009                   rblock   equ      9
  11 0000000A                   rersume  equ      10
  12
  13 00000004                   max_tsk  equ      4
  14 00000028                   max_evn  equ      40
  15 0000000A                   max_prt  equ      10
  16 00000028                   max_blk  equ      40
  17 00000190                   ssize    equ      400
  18
  19 00000000                   tid0     equ      0
  20 00000001                   tid1     equ      1
  21 00000032                   tssz     equ      50
  22
  23 0000000A                   cr       equ      $0a
  24 0000000D                   lf       equ      $0d
```

```
  25
  26 00030000                          org     $30000
  27
  28 00030000                  start:
  29 00030000 2E7C00030000                  movea.l  #$30000,a7
  30 00030006 207C0003013E                  movea.l  #msg2,a0
output alive msg to terminal
  31 0003000C 4EB9000300FE                  jsr      prt
  32 00030012 227C0021100C                  move.l   #$21100c,a1
move edx code from rom
  33 00030018 247C00050000                  move.l   #$50000,a2
to ram
  34 0003001E 223C00000200                  move.l   #512,d1
move 2k bytes to be on safe side
  35 00030024                  lp1:
  36 00030024 24D9                      move.l   (a1)+,(a2)+
  37 00030026 51C9FFFC                    dbf      d1,lp1
  38
  39 0003002A 21FC000500000080              move.l
#$50000,$80      trap 0 entry to edx in ram
  40 00030032 21FC000400000100              move.l
#$40000,$100     ptr to edx workspace ram
  41
  42 0003003A 203C00000000                  move.l   #init,d0
initialise edx
  43 00030040 227C0003011A                  move.l   #config,a1
  44 00030046 4E40                      trap     #0
  45
  46 00030048 203C00000002                  move.l   #tcreate,d0
create task 0
  47 0003004E 223C00000000                  move.l   #tid0,d1
  48 00030054 227C00030088                  movea.l  #task0,a1
  49 0003005A 243C00000032                  move.l   #tssz,d2
  50 00030060 4E40                      trap     #0
  51
  52 00030062 203C00000002                  move.l   #tcreate,d0
create task 1
  53 00030068 223C00000001                  move.l   #tid1,d1
  54 0003006E 227C000300BC                  movea.l  #task1,a1
  55 00030074 243C00000032                  move.l   #tssz,d2
  56 0003007A 4E40                      trap     #0
  57
```

```
  58 0003007C 203C00000001                move.l    #multi,d0
go multitasking
  59 00030082 4E40                        trap      #0
  60
  61 00030084 4E4F                        trap      #15             return
to ads, this code should never run
  62 00030086 0000                        dc.w      0
  63

PAGE  002  edxnew.asm                       Tue Apr 16 10:15:39
1991


  64 00030088                     task0:
  65 00030088 207C00030154                movea.l   #msg3,a0
  66 0003008E 4EB9000300FE                jsr       prt
  67 00030094 4EB90003010C                jsr       delay
  68 0003009A 203C00000006                move.l    #rec,d0
  69 000300A0 4E40                        trap      #0
  70 000300A2 203C00000005                move.l    #send,d0
  71 000300A8 223C00000001                move.l    #tid1,d1
  72 000300AE 243C00000000                move.l    #0,d2
null intertask msg
  73 000300B4 4E40                        trap      #0
  74 000300B6 4EF900030088                jmp       task0
  75
  76 000300BC                     task1:
  77 000300BC 207C00030166                movea.l   #msg4,a0
  78 000300C2 4EB9000300FE                jsr       prt
  79 000300C8 4EB90003010C                jsr       delay
  80 000300CE 203C00000005                move.l    #send,d0
  81 000300D4 223C00000000                move.l    #tid0,d1
  82 000300DA 243C00000000                move.l    #0,d2
null intertask msg
  83 000300E0 4E40                        trap      #0
  84 000300E2 203C00000006                move.l    #rec,d0
  85 000300E8 4E40                        trap      #0
  86 000300EA 4EF9000300BC                jmp       task1
  87
  88
  89
  90 000300F0                     edxpoll:
  91 000300F0 207C0003012A                movea.l   #msg1,a0
```

```
  92 000300F6 4EB9000300FE                jsr       prt
  93 000300FC 4E75                      rts
  94
  95 000300FE                   prt:
  96 000300FE 48E740C0                    movem.l   d1/a0/a1,-(a7)
  97 00030102 4E4F                      trap      #15
  98 00030104 0004                      dc.w      4
  99 00030106 4CDF0302                    movem.l   (a7)+,d1/a0/a1
 100 0003010A 4E75                      rts
 101
 102 0003010C                   delay:
 103 0003010C 3C3C01F4                    move.w    #500,d6
 104 00030110                   del1:
 105 00030110 CFFC0000                    muls.w    #$0,d7
kill some time
 106 00030114 51CEFFFA                    dbf       d6,del1
 107 00030118 4E75                      rts
 108
 109 0003011A                   config:
 110 0003011A 0004                      dc.w      max_tsk
 111 0003011C 0028                      dc.w      max_evn
 112 0003011E 000A                      dc.w      max_prt
 113 00030120 0028                      dc.w      max_blk
 114 00030122 00000190                    dc.l      ssize
 115 00030126 000300F0                    dc.l      edxpoll
 116
 117 0003012A                   msg1:
 118
 119 0003012A 746869732069732065647820          dc.b      'this is
edx poll'
     00030136 706F6C6C
 120 0003013A 0A                      dc.b      cr
 121 0003013B 0D                      dc.b      lf
 122 0003013C 00                      dc.b      0
 123 0003013D 00                      even
 124
 125

PAGE  003  edxnew.asm                     Tue Apr 16 10:15:39
1991


 126 0003013E                   msg2:
```

```
  127 0003013E 6564782065786D706C6520          dc.b     'edx
example running'
      0003014A 72756E6E696E67
  128 00030151 0A                              dc.b     cr
  129 00030152 0D                              dc.b     lf
  130 00030153 00                              dc.b     0
  131 00030154                                 even
  132 00030154                         msg3:
  133 00030154 7461736B20302072756E6E69        dc.b     'task 0
running'
      00030160 6E67
  134 00030162 0A                              dc.b     cr
  135 00030163 0D                              dc.b     lf
  136 00030164 00                              dc.b     0
  137 00030165 00                              even
  138 00030166                         msg4:
  139 00030166 7461736B20312072756E6E69        dc.b     'task 1
running'
      00030172 6E67
  140 00030174 0A                              dc.b     cr
  141 00030175 0D                              dc.b     lf
  142 00030176 00                              dc.b     0
  143 00030177 00                              even
  144
  145
  146                                  end
Assembly complete
Total bytes generated: 376
0 errors
0 warnings


 04/22/91 05:53:17     EDXNEW   SRX

S0030000FC
S2140300002E7C00030000207C0003013E4EB9000353
S21403001000FE227C0021100C247C00050000223CFC
S2140300200000020024D951C9FFFC21FC0005000092
S2140300300008021FC0004000001002003C00000000BA
S214030040227C0003011A4E40203C00000002223CA2
S2140300500000000227C00030088243C00000032DD
S2140300604E40203C00000002223C00000001227C9F
```

```
S2140300700003 00BC243C000000324E40203C00003D
S2140300800001 4E404E4F0000207C000301544EB941
S2140300900003 00FE4EB90003010C203C00000006DE
S2140300A04E40 203C00000005223C00000001243C9A
S2140300B0000000004E404EF900030088207C000339
S2140300C001664EB9000300FE4EB90003010C203C46
S2140300D000000005223C00000000243C0000000055
S2140300E04E40203C000000064E404EF9000300BC84
S2140300F0207C0003012A4EB9000300FE4E7548E734
S21403010040C04E4F00044CDF03024E753C3C01F4E6
S21403011 0CFFC000051CEFFFA4E75000400280 00AFB
S2140301200028000001900003 00F0746869732069DA
S2140301307320656478 20706F6C6C0A0D000065642C
S2140301407820657861 6D706C652072756E6E696E69
S2140301506 70A0D007461736B20302072756E6E69CA
S2140301606E670A0D00007461736B20312072756E22
S20C0301706E696E670A0D0000BC
S8030000FC
```