```
                              NOT Conformed by Chip

"Nutson - Sapieha. I remember Chip saying there were more than 40 registers now.
                 ' We will get a full description in due time.

'Just read the preliminary feature list and made this list for my own reference:

'There are 10 memory mapped registers:

INDA/INDB                 0x1F6 - 0x1F7  'Indirect access to COG memory
PINA/PINB/PINC/PIND       0x1F8 - 0x1FB  'Read / write I/O ports
DIRA/DIRB/DIRC/DIRD       0x1FC – 0x1FF  'Set pins to output

'All other registers can be accessed only with specialised instructions

PTRA/PTRB                 'Pointer for hub access
SPA/SPB                   'CLUT (stack) pointer
CNT                       'System time counter
LFSR                      'Random number generator
MACA/MACB                 'Accu for 64 bit MAC operation
CTRA/CTRB                 'Each have FRQ, PHS, SINand COS register
MULLL/MULLH               'etc, registers to acces the multiply, divide, SQRT and CORDIC ooperations
DAC0/DAC1/DAC2/DAC3       'configuration and data for the DAC’s


----------------------------------------------------------------------------------------------------------
JMPTASK

'There are 4 program counters in each cog. They are initialized as follows:

PC0 = $000
PC1 = $001
PC2 = $002
PC3 = $003

'At first, the task register, which is 32 bits (16 two-bit fields), is cleared to 0,
' making all time slots execute task0.
```

```
JMPTASK sets up to all four PC's at once, using a bit field in S and an address in D.

JMPTASK #substart,#%1111 ...would set all PC's to substart
JMPTASK #substart,#%1000 ...would set PC3 to substart
JMPTASK #substart,#%0100 ...would set PC2 to substart
JMPTASK #substart,#%0010 ...would set PC1 to substart
JMPTASK #substart,#%0001 ...would set PC0 to substart

'Until SETTASK is executed (initialized to $00000000), only PC0 is running, making the cog seem normal.

SETTASK #%%3210 ...'would enable all tasks. If no JMPTASK was done,
        PC1..PC3   'would begin execution from $001..$003 (better have some
                JMP's there)

'When you do an immediate SETTASK #, the lower 8 bits of immediate data are replicated four times to fill 32 bits.
' To get more granularity, you could do a register, instead of an immediate,
' and 32 unique bits would be loaded into the task register, which rotates right after each instruction completion,
 with the 2 LSB's determining which task to execute next.



--------------------------------------------------------------------------------
Here's an example's program's by Chip:
1. ----------
        org

        jmp     #task0
        jmp     #task1
        jmp     #task2
        jmp     #task3

task0   settask #%%3210     'turn on all tasks

:loop   notp    #0          'toggle P0
        jmp     #:loop

task1   notp    #1          'toggle P1
        jmp     #task1

task2   notp    #2          'toggle P2
```

```
        jmp     #task2

task3   notp    #3          'toggle P3
        jmp     #task3
```