Home

Kits are available for this project from
Talking Electronics for $10.00 plus postage.
Plus you will need:
6pin to 5pin adapter @ $2.50

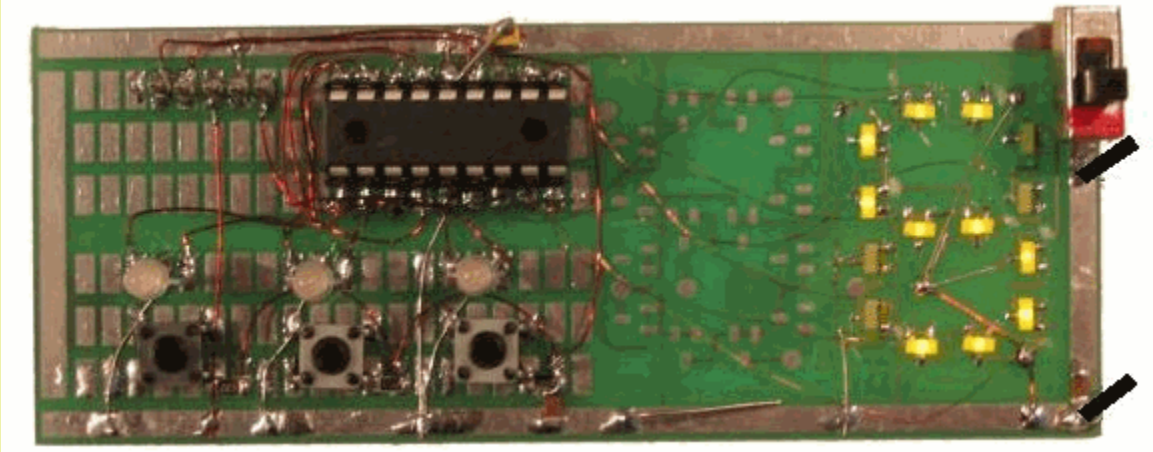
You will also need:
Chip Programmer - PICkit2 from Modtronix
(MPASM and MPLAB come with PICkit2)
- it includes USB lead but not: 6pin to 5pin adapter.

PIC12F629 Data Sheet (.pdf 4,926KB)
Instruction Set for PIC12F629
blank12F629.asm template
PIC12F629.inc

See more projects using micros:
Elektor,EPE,Silicon Chip

Notepad2.zip Notepad2.exe
Library of Sub-routines "Cut and Paste"
Library of routines: **A-E** **E-P** **P-Z**

This is a great game to test your skills.



The complete STROOP GAME

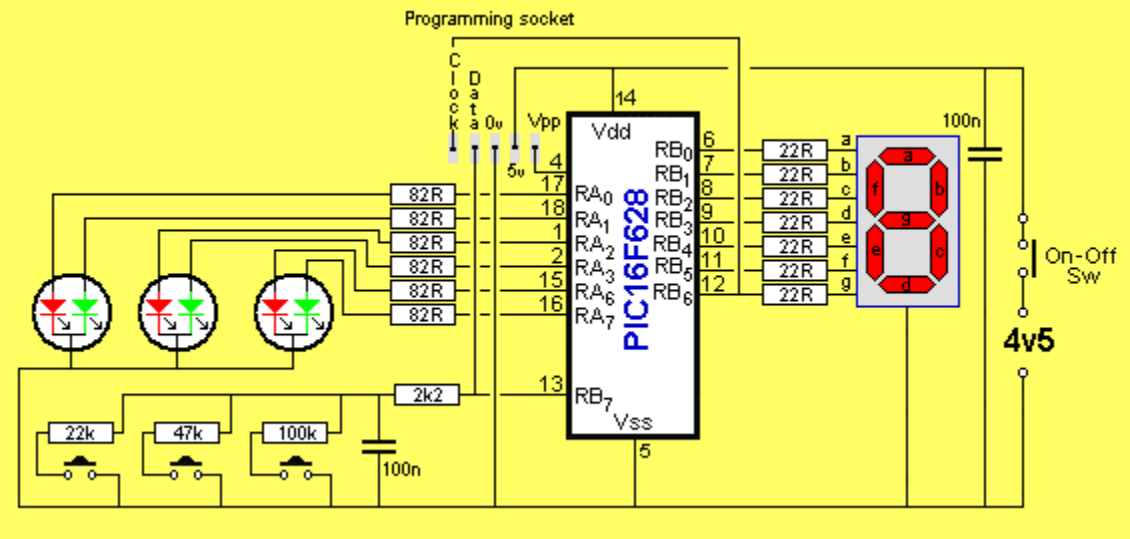
The surface-mount resistors are mounted near the IC socket

This project has been adapted from an experiment by John Ridley Stroop, who published his work in 1935. Basically it is a "trick." It is a trick in that you are required to answer a question at a "second level of thinking."
In our test we have three tri-coloured LEDs and below each is a push-button.
When a LED illuminates, your immediate response is to push the button below the LED.
But this is not the requirement.
The LED will illuminate as one of three colours. Red, Orange or Green.
You are required to push the first button for red, the middle button for orange and the third button for green.
In other words you have to divorce yourself from the urge to push the closest push-button and work on the colour-requirement.
Obviously you will become more-adept at this over a period of time but the most important results will come from the first few attempts.
That's why it will be interesting to have your friends take a test.
The "Stroop effect" has been used to investigate the psychological capacities of a person. In fact it introduces capabilities that have never been investigated before. Although I don't believe in anything to do with psychology, this test is considered to measure selective attention, cognitive flexibility and processing speed. About the only word I understand is "processing speed" and that's how our game works. It runs for 20 seconds and gives a score on the 7-segment display.
You are required to get as many matches as possible in 20 seconds.
The game comes on by displaying the letters "S-t-r-o-o-P" on the 7-segment display and then sits ready for your first try.
The single digit display can actually display up to 99 as it flashes the tens digit first and then the units. It repeats this three times and turns off, ready for the second game. Push any button to start.

The CIRCUIT

The circuit consists of three push-buttons, three tri-coloured LEDs and a 7-segment display made from individual LEDs. All the "timing," outputting and switch-detection is done in the PIC16F628 microcontroller.
The board contains 5 pins for **In-Circuit Programming** so the program can be changed and modified at any time.
The resistor values for the LEDs have been chosen to get the maximum brightness, using the 25mA available from each output.
The 7-segment display is made up of 14 individual LEDs, with two LEDs in series for each

segment. This gives a voltage drop of approx 3.4v and a 22R current-limit resistor is needed.
RB7 is used for the switch inputs and this is also used as the data line when programming. To allow the data to enter the chip while programming, a 2k2 resistor has been added as the 100n upsets the data line if it is connected directly to the programming pin.
The resistor values for the switches have been chosen to separate the timing for each switch and make it easy to recognise in the program.



STROOP GAME CIRCUIT

The 7-segment display actually has 2 yellow LEDs in series for each segment

CONSTRUCTION

You can build the circuit on any type of Proto board or design your own PC board.
Use 3 - AAA cells and not button cells as button cells do not have low enough impedance to keep the voltage high when all the LEDs are illuminated and the chip hic-cups and flashes the display.

The PROGRAM

The program has been kept simple to make it easy to understand. Very few Boolean expressions have been used as they take a lot of understanding and "working out" as to the the outcome of the instruction.
We note that a simpler program was written in "C" and it failed to compile into the 1024 memory locations, so I don't know how the inefficiency of higher-level programming would relate to this project.
In any case, we have used the 35 instructions that come with the chip and this makes fault-finding very easy as you know the fault lies in the code you have generated.
As long as you only introduce a small amount of code at a time, you will be able to gradually get a program up-and-running.

The interesting feature of the program is the overall timing. The micro is counting in the background via timer1 and this consists of two files (registers) capable of counting to 65,536. A prescaler has been added to increase the count to 524,288. This is about half a second.
When the timer overflows, the program-execution is interrupted and the micro goes to location 4 (called the Interrupt location where it finds an instruction to go to a sub-routine called: "isr." At isr, another file is decremented (_20Secs) thirty-nine times and this produces the 20 seconds duration for each game.
(Point to remember: Timer0 does not produce a long delay, so Timer1 has to be used).

The buttons are detected by charging the 100n and waiting 20mS to see if the capacitor has discharged. We know the cap will discharge in less than 8mS if a button is pushed.
The program now knows if a button is pushed or not.
It makes a second pass, if a button is pushed, to work out which button has been pressed.
The first button will discharge the cap in less than 2mS, the second button will discharge the cap in less than 4mS and the third button will discharge the cap in less than 8mS.
The program now performs a 1mS loop, looking for a LOW on the detecting pin.
It will exit with a value of 1-8.
The program now decrements the count file and and if it is zero after one or two decrements, button 1 has been pressed. It continues with decrements until it finds the button.

RANDOM NUMBER

The most difficult thing to produce on a computer is a random number.
You can combine and XOR various files or use a table. but nothing generates a truly random result.
We have used the "waiting time" when a player waits to provide an answer and this generates a new random number, while the program is actually using a previously generated number for the play in progress. That's why the random number has to be generated in a sub-routine called "Create," and this number is passed to the Random Number file for use in the next try.

The program contains a number of very important subroutines that you will be able to "cut and paste" for projects in the future.

MORE

For more details on modifying the program and burning the PIC chip, see Talking Electronics website and click on Elektor,EPE,Silicon Chip in the index.
You can find details of: PICkit-2 and Adapter connected for In-Circuit Programming at this link.

Here is the file you will need for "burning" your chip and/or modifying the program. It comes as .asm, .txt and .hex for using as a file to modify, or to read, or to burn a new chip:
Stroop.asm
Stroop.txt
Stroop.hex
The kit comes with a pre-programmed PIC chip, see parts list below.

```
;*****************************************************************
;Started 18/6/2009
;STROOP GAME - Press button according to the colour of the LED
;
;Port A drives 3 tri-coloured LEDs
;Port B drives 7 segment display and keys
;*****************************************************************

        list P = 16F628;microcontroller
        include         ;registers for F628


        __Config        _cp_off & _lvp_off & _pwrte_on
                & _wdt_off & _intRC_osc_noclkout & _mclre_off

;code protection - off
;low-voltage programming - off
```

```
;power-up timer -  on
;watchdog timer - off
;use internal RC for 4MHz - all pins for in-out


;*****************************************************************
; variables - names and files
;*****************************************************************


               ;Files for F628 start at 20h


temp1          equ 20h ;for delay
temp2          equ 21h ;for delay
count          equ 22h ;counts loops for switch
Random         equ 23h ;random number file
units          equ 24h ;
tens           equ 25h ;
Sw_Flag        equ 26h ;
_20Secs        equ 27h ;file for counting up to 20 seconds
loops          equ 28h ;loops for number display
Produce        equ 29h ;produce random number
temp3          equ 2Ah ;for 500mS delay

;*****************************************************************
;Equates
;*****************************************************************
status         equ     0x03
cmcon          equ     0x1F
rp1            equ     0x06
rp0            equ     0x05
portA          equ     0x05
portB          equ     0x06

z              equ     0x02


;*****************************************************************
;Beginning of program
;*****************************************************************


Start   org     0x00    ;program starts at location 000
        goto    Stroop  ;goto Stroop
        nop
        nop             ;NOPs to get past reset vector address
        org     4
        goto    isr


SetUp   bsf     status,rp0
        movlw   b'00000000'     ;Make RA output
        movwf   05h             ;trisA
        clrf    06h             ;trisB   Make all RB output
        movlw   b'10000000';
```

```
        movwf   OPTION_REG      ; x000 0000 x=1 = weak pull-ups
disabled
        bcf     status,rp0      ;select programming area - bank0
        movlw   b'00000000'     ;6,7=0 disables all interrupts
        movwf   INTCON          ;until we want timing to commence
        clrf    Sw_Flag
        movlw   07h             ;turn comparators off
        movwf   cmcon
        clrf    portA
        clrf    portB
        clrf    units
        clrf    tens
        clrf    Random          ;random will be 1-9
        movlw   .39
        movwf   _20Secs
        goto    Main

;************************************
;* Tables                           *
;************************************

table1  addwf   PCL,F           ;02h,1
        nop                     ;display random LED colour
        retlw   b'00000001'      ; Led A - red
        retlw   b'00000011'      ; Led A - orange
        retlw   b'00000010'      ; Led A - green
        retlw   b'00000100'      ; Led B - red
        retlw   b'00001100'      ; Led B - orange
        retlw   b'00001000'      ; Led B - green
        retlw   b'01000000'      ; Led C - red
        retlw   b'11000000'      ; Led C - orange
        retlw   b'10000000'      ; Led C - green

table2  addwf   PCL,F           ;02h,1  add W to program counter
        retlw   b'00111111'      ; "0"   -|F|E|D|C|B|A
        retlw   b'00000110'      ; "1"   -|-|-|-|C|B|-
        retlw   b'01011011'      ; "2"   G|-|E|D|-|B|A
        retlw   b'01001111'      ; "3"   G|-|-|D|C|B|A
        retlw   b'01100110'      ; "4"   G|F|-|-|C|B|-
        retlw   b'01101101'      ; "5"   G|F|-|D|C|-|A
        retlw   b'01111101'      ; "6"   G|F|E|D|C|-|A
        retlw   b'00000111'      ; "7"   -|-|-|-|C|B|A
        retlw   b'01111111'      ; "8"   G|F|E|D|C|B|A
        retlw   b'01101111'      ; "9"   G|F|-|D|C|B|A


;************************************
;* Sub routines                     *
;************************************

Attract         ;flash all red, orange  green then random LED

        movlw   b'01000101'      ; all red
        movwf   portA
        call    _250mS
        movlw   b'11001111'      ; all orange
```

```
        movwf   portA
        call    _250mS
         movlw  b'10001010'     ; all green
         movwf  portA
        call    _250mS
        clrf    portA
        call    _250mS
        call    _250mS
        retlw   00

                ;produce random number

Create  incf    Produce,f
        movlw   .10             ;put ten into w
        xorwf   Produce,0       ;compare Random file with ten
        btfss   status,2        ;zero flag in status. Set if
Random = ten
        goto    $+3
        clrf    Produce
        incf    Produce,f
        retlw   00


        ;Delays

_1mS    nop
        decfsz  temp1,f
        goto    _1mS
        retlw   00


_10mS   movlw   0Ah
        movwf   temp2
_b      nop
        decfsz  temp1,f
        goto    _b
        decfsz  temp2,f
        goto    _b
        retlw   00

_100mS  movlw   .100
        movwf   temp2
_c      nop
        decfsz  temp1,f
        goto    _c
        decfsz  temp2,f
        goto    _c
        retlw   00

_250mS  movlw   .240
        movwf   temp2
_d      nop
        decfsz  temp1,f
        goto    _d
        decfsz  temp2,f
        goto    _d
```

```
        retlw   00

_500mS  movlw   02
        movwf   temp3
        call    _250mS
        decfsz  temp3,f
        goto    $-2
        retlw   00

_3Sec   movlw   .12
        movwf   temp3
        call    _250mS
        decfsz  temp3,f
        goto    $-2
        retlw   00


;interrupt service routine

isr     nop
        bsf     status,rp0      ;Bank 1
        bsf     PIE1,0          ;,0 1=enables TMR1 interrupt
        bcf     status,rp0      ;bank 0
        bcf     PIR1,0          ;clear TMR1 overflow flag
        bsf     INTCON,7        ;This instruction is needed
HERE!!!
        bsf     INTCON,6        ;1=enable all peripheral
interrupts
        decfsz  _20Secs,f       ;creates 20Sec delay for each
game.
        retfie

        bcf     PIE1,0          ;,0 0=disables TMR1 interrupt
        bcf     INTCON,6        ;0=disable all peripheral
interrupts

        decf    tens,f
        incf    tens,f
        movlw   .10
        subwf   units,f
        btfsc   status,0        ;test carry bit for borrow
        goto    $-4
        movlw   .10
        addwf   units,f

        movlw   03
        movwf   loops

        movf    tens,w
        btfsc   status,z
        goto    $+.18           ;If 0-9, display single digit
        call    table2
        movwf   portB
        call    _500mS
        call    _250mS
        clrf    portB
```

```
        call    _250mS
        movf    units,w
        call    table2
        movwf   portB
        call    _500mS
        call    _250mS
        clrf    portB
        call    _500mS
        call    _500mS
        decfsz  loops,f
        goto    $-.18
        goto    SetUp

        movf    units,w
        call    table2
        movwf   portB
        call    _3Sec
        goto    SetUp


        ; show Stroop

Stroop  bsf     status,rp0
        clrf    06h             ;trisB   Make all RB output
        movlw   b'10000000';
        movwf   OPTION_REG      ; x000 0000 x=1= weak pull-ups
disabled
        bcf     status,rp0      ;select programming area - bank0
        movlw   07h             ;turn comparators off
        movwf   cmcon
        clrf    portA

        movlw   b'01101101'     ; "S"
        movwf   portB
        call    _500mS
        clrf    portB
        call    _250mS
        movlw   b'01111000'     ; "t"
        movwf   portB
        call    _500mS
        clrf    portB
        call    _250mS
        movlw   b'01010000'     ; "r"
        movwf   portB
        call    _500mS
        clrf    portB
        call    _250mS
        movlw   b'01011100'     ; "o"
        movwf   portB
        call    _500mS
        clrf    portB
        call    _250mS
        movlw   b'01011100'     ; "o"
        movwf   portB
        call    _500mS
        clrf    portB
```

```
        call    _250mS
        movlw   b'01110011'     ; "P"
        movwf   portB
        call    _500mS
        clrf    portB
        goto    SetUp


Sw      clrf    Sw_Flag
        bsf     status,rp0
        bcf     06h,7           ;trisB   Make bit 7 output
        bcf     status,rp0
        bsf     portB,7         ;make bit 7 HIGH
        call    _1mS            ;create delay to charge 100n
        bsf     status,rp0
        bsf     06h,7           ;trisB   Make bit 7 input
        bcf     status,rp0
        call    _10mS
        call    _10mS
        btfsc   06h,7           ;if HIGH, button not pushed
        retlw   00
        clrf    count
        bsf     status,rp0
        bcf     06h,7           ;trisB   Make bit 7 output
        bcf     status,rp0
        bsf     portB,7         ;make bit 7 HIGH
        call    _1mS            ;create delay to charge 100n
        bsf     status,rp0
        bsf     06h,7           ;trisB   Make bit 7 input
        bcf     status,rp0
SwA     call    _1mS
        call    _1mS
        incf    count,f
        btfsc   06h,7           ;is input HIGH?
        goto    SwA             ;count exits with 1-8
        bsf     Sw_Flag,0       ;button has been pushed
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,1
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,1
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,2
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,2
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,2
        retlw   00
```

```
        bsf     Sw_Flag,3
        retlw   00


                ;switch released

Sw_Rel  clrf    Sw_Flag
        bsf     status,rp0
        bcf     06h,7           ;trisB   Make bit 7 output
        bcf     status,rp0
        bsf     portB,7         ;make bit 7 HIGH
        call    _1mS            ;create delay to charge 100n
        bsf     status,rp0
        bsf     06h,7           ;trisB   Make bit 7 input
        bcf     status,rp0
        call    _10mS
        call    _10mS
        btfsc   06h,7           ;if HIGH, button not pushed
        retlw   00
        bsf     Sw_Flag,0
        retlw   00


;************************************
;* Main                             *
;************************************

                        ;Stroop comes on "blank" looking for
button-push
Main    call    Create
        call    Sw
        btfss   Sw_Flag,0
        goto    $-3     ;no
                ;button pressed and Random Number generated

                ;Stroop goes into ATTRACT mode then stops on
Random LED

        call    Attract

                        ;Display Random LED colour, waiting for sw
press


;*****************************************************************
;* Start Timer1 to count 20 seconds in the background            *
;*****************************************************************

        bsf     status,rp0      ;Bank 1
        movlw   b'10000000'     ;
        movwf   OPTION_REG      ; x000 0000 x=1= weak pull-ups
disabled
        bcf     status,rp0      ;bank 0

        movlw   b'11000000'     ;b'11000000'
        movwf   INTCON          ;,0  1=RB port change interrupt
```

```
flag
                                ;,1  1=RB0 interrupt occurred
        ;bcf    INTCON,2        ;1=TMR0 overflowed. Clear overflow
flag
        ;bcf    INTCON,3        ;1=enable RB port change interrupt
        ;bcf    INTCON,4        ;1=enable RB external interrupt
        ;bsf    INTCON,5        ;1=enable TMR0 overflow
(interrupt)
        ;bcf    INTCON,6        ;1=enable all peripheral
interrupts
        ;bsf    INTCON,7        ;1=enable all unmasked interrupts

        movlw   b'00110101'     ;b'00110001'
        movwf   T1CON           ;,7  not used
                                ;,6 0=Timer1 is ON
                                ;,5,4  11=8 prescale (max) 01=1:2
                                ;,3 bit ignored
                                ;,2 This MUST BE SET!!!!!!
                                ;,1 0=int clock
                                ;,0 1=enable timer1

        bsf     status,rp0      ;Bank 1 (Must use Bank1)
        bsf     PIE1,0          ;,0 1=enables TMR1 interrupt
        bcf     status,rp0      ;bank 0
        bcf     PIR1,0          ;clear TMR1 overflow flag

        clrf    TMR1L           ;clear the Timer1 low register
        clrf    TMR1H           ;clear the Timer1 high register
                                ;Timer0 is not used
                                ; will go to isr when overflow in
TMR1
                                ;0.52 sec when prescaler=1:8
524,288uS

        bsf     status,rp0      ;Bank 1 (Must use Bank1)
        bsf     PIE1,0          ;,0 1=enables TMR1 interrupt
        bcf     status,rp0      ;bank 0

                        ;game has started with random LED

Main2   movf    Produce,w
        movwf   Random
        call    table1
        movwf   portA           ;show random number
                                ;program gets to here after 1
press

        call    Create
        call    Sw
        btfss   Sw_Flag,0       ;has button been pressed?
        goto    $-3             ;no
                                ;button pressed

        movlw   01
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
```

```
        goto    $+5
        btfss   Sw_Flag,1       ;random=1       Is sw = button1
        goto    release
        incf    units,f
        goto    release

        movlw   02
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,2       ;random=2       Is sw = button2
        goto    release
        incf    units,f
        goto    release

        movlw   03
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,3       ;random=3       Is sw = button3
        goto    release
        incf    units,f
        goto    release

        movlw   04
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,1       ;random=4       Is sw = button1
        goto    release
        incf    units,f
        goto    release

        movlw   05
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,2       ;random=5       Is sw = button2
        goto    release
        incf    units,f
        goto    release

        movlw   06
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,3       ;random=6       Is sw = button3
        goto    release
        incf    units,f
        goto    release

        movlw   07
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,1       ;random=7       Is sw = button1
```

```
        goto    release
        incf    units,f
        goto    release

        movlw   08
        xorwf   Random,0        ;yes
        btfss   status,z        ;test zero bit for compare
        goto    $+5
        btfss   Sw_Flag,2       ;random=8       Is sw = button2
        goto    release
        incf    units,f
        goto    release

        incf    units,f
        goto    release

release clrf    portA
        call    _500mS
        goto    Main2

        End
```

THE GAME

The game is played by switching the project on and seeing which colour is illuminated.
Press the first button if the colour is RED, the second button if the colour is Orange and the third switch if the the colour is GREEN.
The aim is to get as many correct in 20 seconds.
The score appears on the 7-segment display. The display flashes the tens digit and then the units. It then blanks for 2 seconds and repeats the number. It does this 3 times then turns off.

Stroop

Parts List

Cost: au$25.00 plus $7 postage

Kits are available

7 - 22R SM resistor
6 - 82R SM resistor
1 - 2k2 SM resistor
1 - 22k SM resistor
1 - 47k SM resistor
1 - 100k SM resistor

2 - 100n SM capacitors

14 - Orange SM LEDs
3 - tri-coloured LEDs

1 - SPDT mini slide switch
3 - mini tactile push buttons

20cm fine enamelled wire
30cm - very fine solder
1 - 18 pin IC socket
5 - machine pins
1 - PIC16F628 chip (with Stroop routine)
3 - AAA cells (do not use button cells
- they produce false operation)

1 - Prototype PC board

JUST THE MICRO:

Pre-programmed PIC16F628 micro with Stroop routine $15.00 plus $5.00 post

18/8/09