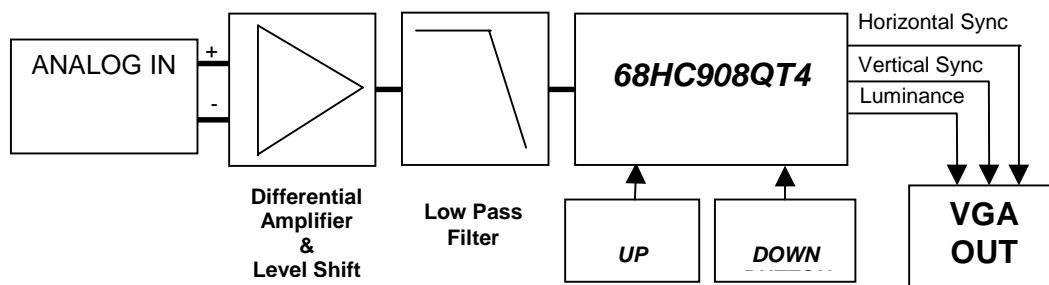


## ENTRY F124 VGA SIGNAL PROBE

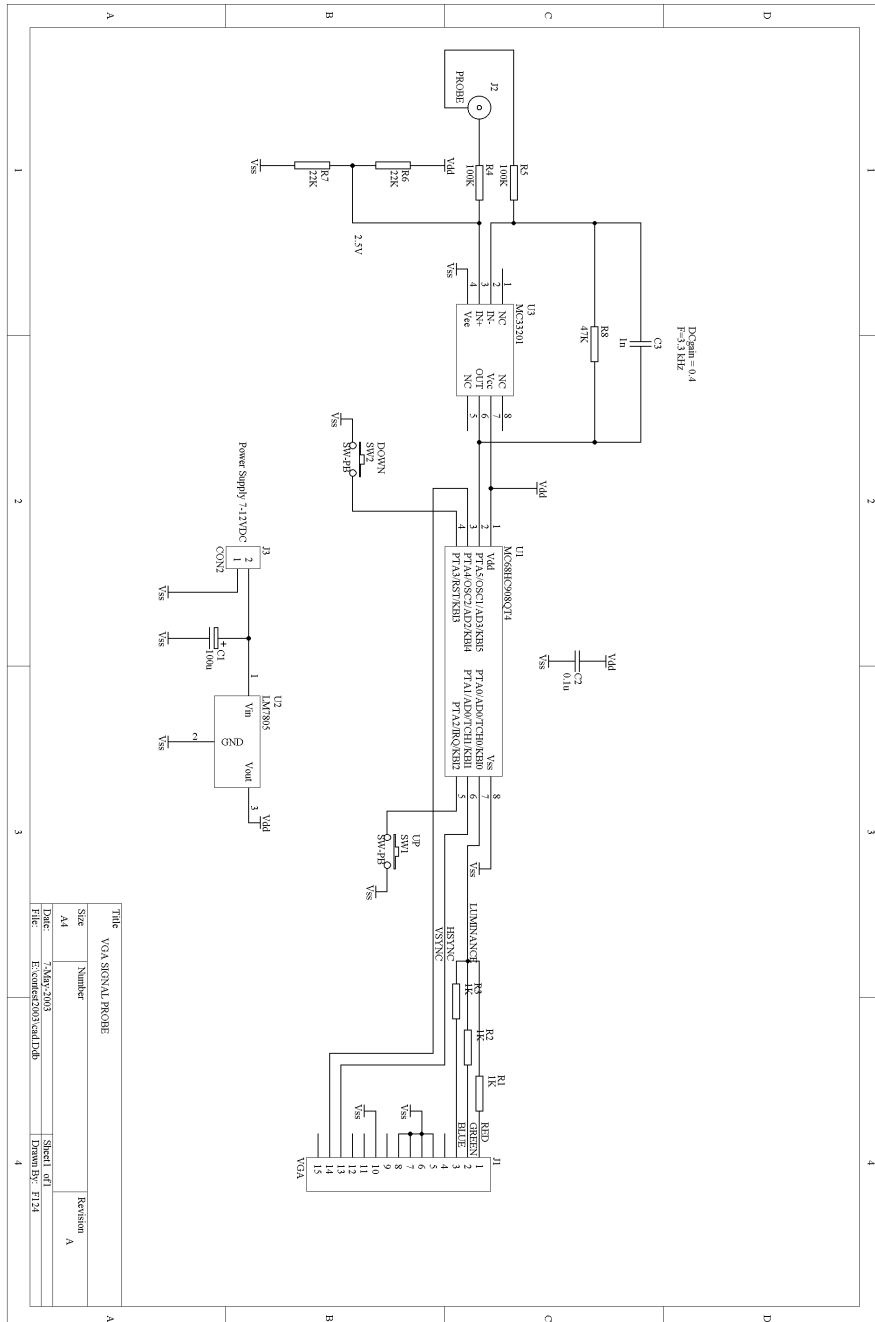
### 1. ABSTRACT

This project utilises a 68HC908QT4 to digitise an analog signal, and generate a corresponding VGA video signal representing the time sequence of the samples. This allows a simple audio-bandwidth CRO to be implemented using two 8-pin ICs. The device includes two buttons for toggling the acquisition mode between free running, and digital storage. In free running mode, the acquisition is synchronised to the horizontal display frequency. In digital storage mode, the buttons also act to adjust the time base, triggering voltage level and edge polarity.

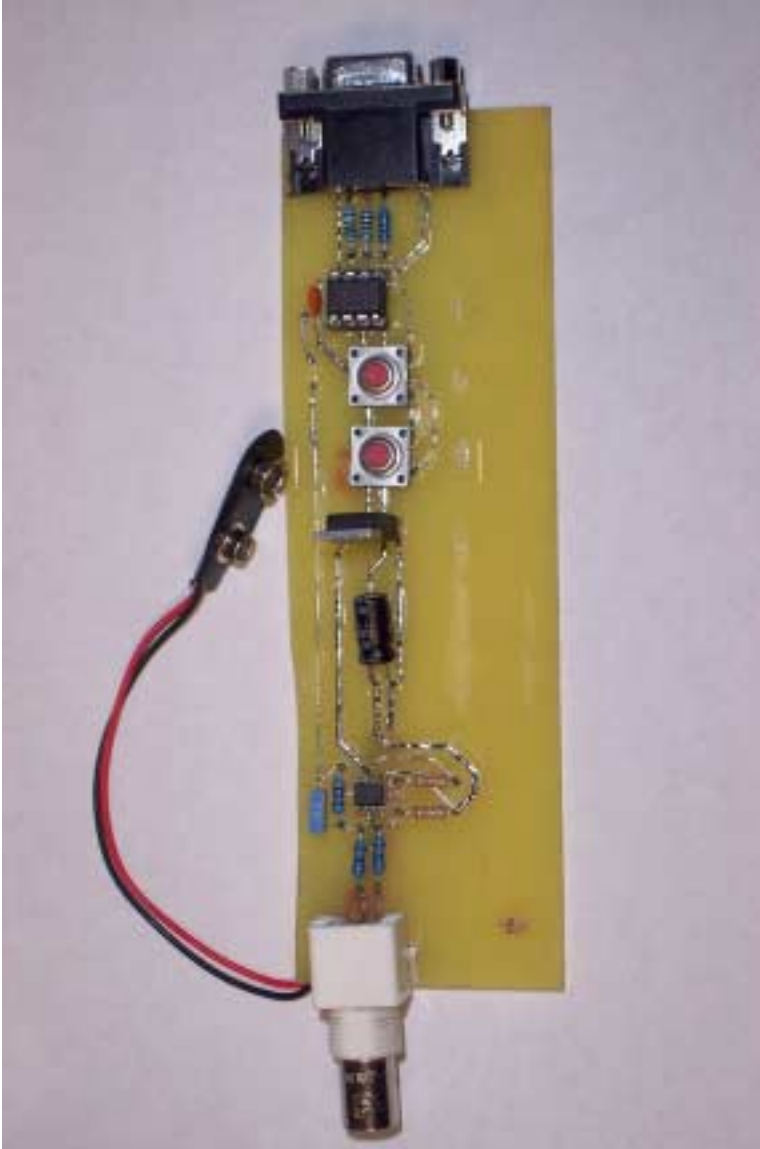
### 2. BLOCK DIAGRAM



### 3. SCHEMATIC



#### 4. PROJECT PICTURE



#### 5. SOURCE LISTING

```
;  
;Motorola/Circuit Cellar  
;  
;Flash Innovation Contest 2003  
;  
;Contest Entry F124  
;  
;VGA Signal Probe  
;
```

```

                org $FFC0

trim_val:  DC.B 105      ;FLASH trim default value

;
;RAM variable definitions
;

BUFSIZ:      equ          350/3  ;size of sample storage buffer
STACKSIZ:    equ          8      ;size of stack

MY_RAM      SECTION SHORT

LUM_TIMER:   ds.b 1           ;luminance timer latch
FLAGS:       ds.b 1           ;general flag bits
TMR_ACC:     ds.b 1           ;subsampling fractional accumulator
SAMPLE_IDX:  ds.b 1           ;sample buffer index
STORAGE:     ds.b BUFSIZ-2     ;sample buffer .. 116 entries
TRIGGER:     ds.b 1           ;trigger sample level
TMR_STEP:    ds.b 1           ;subsampling fractional step (timebase)

MY_ROM      SECTION

;
;QT4 relevant hardware definitions
;
_PTA:        equ          0
_DDRA:       equ          4
_PTAPUE:     equ          0bh
_KBSCR:      equ          1ah
_KBIER:      equ          1bh
_TSC:        equ          20h
_TCNTH:      equ          21h
_TCNTHL:     equ          22h
_TMODH:      equ          23h
_TMODL:      equ          24h
_TSC0:       equ          25h
_TCH0H:      equ          26h
_TCH0L:      equ          27h
_TSC1:       equ          28h
_TCH1H:      equ          29h
_TCH1L:      equ          2ah
_OSCTRIM:    equ          38h
_ADSCR:      equ          3ch
_ADR:        equ          3eh
_ADICLK:     equ          3fh
_COPCTL:     equ          0ffffh

;
;Port A pin allocations
;
LUM_PIN:     equ          0      ;Luminance output (0=black 1=grey)
HSYNC_PIN:   equ          1      ;Horizontal sync output (0=active)
UP_PIN:      equ          2      ;Up pin (0=preserved)
DOWN_PIN:    equ          3      ;Down pin (0=preserved)
VSYNC_PIN:   equ          4      ;Vertical sync output (0=active)

;
;Flag bits
;
RISING:      equ          0      ;set when rising edge triggering
ADJTRIG:     equ          1      ;set when trigger is being adjusted, else timebase
FREERUN:     equ          2      ;set when free run active

;
;VGA Horizontal timing constants
;
HORIZ_WIDTH: equ          101     ;31.5625 us    @ 3.2 MHz
SYNC_WIDTH:  equ          12      ;3.75 us      @ 3.2 MHz

```

```

BACKPORCH_WIDTH:    equ    6                ;1.875 us    @ 3.2 MHz
FRONTPORCH_WIDTH:  equ    3                ;0.9375 us   @ 3.2 MHz
ACTIVE_START:      equ    SYNC_WIDTH+BACKPORCH_WIDTH ;18 image left margin
ACTIVE_END:        equ    HORIZ_WIDTH-FRONTPORCH_WIDTH ;98 image right margin
ACTIVE_WIDTH:      equ    ACTIVE_END-ACTIVE_START    ;80 width of image
RENDER_START:     equ    ACTIVE_START+(ACTIVE_WIDTH-64)/2 ;19 rendering start

                xdef                _Reset

_Reset:
    lda            trim_val            ;load the TRIM value
    sta            _OSCTRIM

    sei
    ldhx           #LUM_TIMER          ;initialise stack pointer to end of RAM
    txs
    jsr            InitIO              ;crank up the I/O ports
    jsr            InitADC             ;crank up the ADC
    jsr            InitTimers          ;crank up the VGA horizontal timers
    cli
    bra            DoFreerun           ;start free running sampler by default

;
;Initialise ADC subsystem
;
InitADC:
    lda            #64                 ;ADC clock= bus clock/4=0.8 MHz @ 3.2 MHz
    sta            _ADICLK
    lda            #32+1+2             ;turn on ADC, continuous conversion on
                                        ;PTA5, no IRQ
    sta            _ADSCR
    rts

;
;Initialise I/O subsystem
;
InitIO:
    lda            #(1<<LUM_PIN)|(1<<HSYNC_PIN)|(1<<VSYNC_PIN)
    ;LUM, HSYNC, VSYNC = output, ADC, DOWN, UP = input
    sta            _DDRA
    lda            #2+16+4+8
    sta            _PTA                ;deassert HSYNC and VSYNC, LUM=0V
    lda            #(1<<UP_PIN)|(1<<DOWN_PIN)
    sta            _PTAPUE             ;enable pullups for input switches

    bset           1,_KBSCR            ;mask keyboard interrupts
    bset           UP_PIN,_KBIER
    bset           DOWN_PIN,_KBIER    ;enable up and down keys
    bset           0,_KBSCR           ;keyboard is edge and level triggered
    bset           2,_KBSCR           ;ack any pending keyboard interrupts

    rts

;
;Initialise VGA timers and system mode
;
InitTimers:
    lda            #16+32
    sta            _TSC                ;stop timer & reset it

    clr            _TMODH
    lda            #HORIZ_WIDTH
    sta            _TMODL             ;timer overflows every horizontal line

    lda            #16+8
    ;unbuffered operation, no IRQ
    ;output compare, LUM clear on match
    sta            _TSC0
    lda            #16+8+4+2
    ;unbuffered operation, no IRQ
    ;output compare, HSYNC set on match, ;toggle
    ;on overflow

```

```

        sta          _TSC1

        clr          _TCH1H
        lda          #SYNC_WIDTH
        sta          _TCH1L          ;HSYNC width
        clr          _TCH0H
        lda          #ACTIVE_START+(ACTIVE_WIDTH)/2
        sta          _TCH0L          ;LUM trigger initially at mid point
        sta          LUM_TIMER

        bclr         7,_TSC0          ;clear LUM output compare flag
        bclr         7,_TSC1          ;clear HSYNC output compare flag

        clr          FLAGS

        bclr         RISING,FLAGS    ;falling edge by default
        bset         FREERUN,FLAGS   ;free running by default

        bclr         LUM_PIN,_PTA    ;LUM off (low)
        bset         HSYNC_PIN,_PTA  ;HSYNC inactive high
        bset         VSYNC_PIN,_PTA  ;VSYNC inactive high

        lda          #128+64
        sta          TRIGGER          ;set trigger level to 50% full scale

        lda          #BUFSIZ-3
        sta          SAMPLE_IDX       ;start sampling

        clr          TMR_ACC          ;reset sample timer accumulator
        lda          #255
        sta          TMR_STEP         ;maximum sample rate

        lda          #64
        sta          _TSC              ;timebase = bus clock, start,
                                        ;overflow IRQ

        rts

;
;Timer overflow IRQ. This occurs at the start of every horizontal line
;

        xdef _TIMER_IRQ

_TIMER_IRQ:
                                                ;
                                                ;9 latency
                                                ;
        lda          LUM_TIMER        ;3
        sta          _TCH0L           ;3 load luminance timer with new value
        bclr         7,_TSC           ;4 ack overflow IRQ
        rti                          ;7

;
;Process keypresses macro
;
;In trigger adjust mode:
;
;UP          increase trigger level, set trigger direction to rising level
;DOWN       decrease trigger level, set trigger direction to falling level
;
;In timebase adjust mode:
;
;UP          increase sampling frequency
;DOWN       decrease sampling frequency
;
;UP+DOWN    toggles between trigger adjust, timebase adjust and freerun
;
;Total execution time < 47 cycles
;
Keypress:    MACRO

```

```

brclr      3, _KBSCR, \@KeypressDone      ;5 check for a keypress flag
bset      2, _KBSCR                        ;4 ack keyboard interrupt

lda        _PTA                            ;3 fetch port state
and        #(1<<UP_PIN)|(1<<DOWN_PIN)      ;2 mask pins
beq        \@Toggle                        ;3 both held = toggle mode
and        #(1<<DOWN_PIN)                  ;2 down pin active?
beq        \@GoUp                          ;3 no...must be up
bra        \@GoDown                        ;3 yes..must be down

\@Toggle:
brclr     FREERUN, FLAGS, \@NotFreerun     ;5
bclr     FREERUN, FLAGS                     ;4
bset     ADJTRIG, FLAGS                     ;4 free run -> adj trigger
bra      \@KeypressDone                    ;3

\@NotFreerun:
brclr     ADJTRIG, FLAGS, \@GoFreerun      ;5
bclr     ADJTRIG, FLAGS                     ;4 adj trig ->
; adj timebase
bra      \@KeypressDone                    ;3

\@GoFreerun:
bset     FREERUN, FLAGS                     ;4 adj timebase ->
; free run
bra      \@KeypressDone                    ;3

\@GoDown:
brset     FREERUN, FLAGS, \@KeypressDone   ;5 only adjust if DSO mode

brset     ADJTRIG, FLAGS, \@DecTrig        ;5
tst      TMR_STEP                          ;3
beq      \@KeypressDone                    ;3
dec      TMR_STEP                          ;4
bra      \@KeypressDone                    ;3

\@DecTrig:
bclr     RISING, FLAGS                      ;4 falling edge triggered
tst     TRIGGER                             ;3
beq     \@KeypressDone                     ;3
dec     TRIGGER                             ;4
bra     \@KeypressDone                     ;3

\@GoUp:
brset     FREERUN, FLAGS, \@KeypressDone   ;5 only adjust if DSO mode

brset     ADJTRIG, FLAGS, \@IncTrig        ;5
inc      TMR_STEP                          ;4
bne     \@KeypressDone                     ;3
dec     TMR_STEP                          ;4
bra     \@KeypressDone                     ;3

\@IncTrig:
bset     RISING, FLAGS                      ;4 rising edge triggered
inc     TRIGGER                             ;4
bne     \@KeypressDone                     ;3
dec     TRIGGER                             ;4
bra     \@KeypressDone                     ;3

\@KeypressDone:
        ENDM

;
;Wait for start of horizontal line, and clear flag
;
HWAIT:  MACRO
\@WtHsync:
brclr     7, _TSC1, \@WtHsync              ;5 wait for HSYNC compare
bclr     7, _TSC1                          ;4 ack it
        ENDM

```

```

;
;Untriggered unstored 31kHz sample + display
;
;Uses 480 line VGA mode
;
DoFreerun:

    sta        _COPCTL        ;watchdog reset
    bset       VSYNC_PIN,_PTA ;deassert VSYNC

    lda        #16+8+4+2      ;unbuffered operation, no IRQ
                                ;output compare, LUM set on match
                                ;clear on overflow

    sta        _TSC0

    ;generate 480 VGA lines

    ldx        #480/3        ;2

LineLoop1:
    lda        _ADR           ;3 read most recent ADC result
    lsra                          ;1
    lsra                          ;1 convert to 6-bit
    add        #RENDER_START    ;2 offset
    sta        LUM_TIMER        ;3 load into LUM timer
    HWAIT
    lda        _ADR           ;3 read most recent ADC result
    lsra                          ;1
    lsra                          ;1 convert to 6-bit
    add        #RENDER_START    ;2 offset
    sta        LUM_TIMER        ;3 load into LUM timer
    HWAIT
    lda        _ADR           ;3 read most recent ADC result
    lsra                          ;1
    lsra                          ;1 convert to 6-bit
    add        #RENDER_START    ;2 offset
    sta        LUM_TIMER        ;3 load into LUM timer
    HWAIT
    dbnzz     LineLoop1        ;3 next line if applicable

    ;deactivate the luminance generator

    lda        #16+8          ;unbuffered operation, no IRQ
                                ;output compare, LUM clear

    sta        _TSC0
    clr        _TSC           ;stop overflow interrupt

    ldx        #11            ;front porch = 0.35 ms = 11 lines

WtFrontPorcha:
    HWAIT
    dbnzz     WtFrontPorcha

    bclr       VSYNC_PIN,_PTA ;assert VSYNC
    ldx        #2

WtVSYNcA:
    HWAIT
    dbnzz     WtVSYNcA        ;vsync = 0.06 ms = 2 lines

    bset       VSYNC_PIN,_PTA ;deassert VSYNC

    Keypress

    ldx        #32            ;back porch = 1.02 ms = 32 lines

WtBackPorcha:
    HWAIT
    dbnzz     WtBackPorcha

```

```

        bclr          7,_TSC          ;ack overflow IRQ

        lda          #64
        sta          _TSC          ;enable overflow interrupt

        brclr       FREERUN,FLAGS,DoDSO ;entering DSO mode?
        jmp         DoFreerun      ;no, next screen

;
;Sample buffer display. One sample per three horizontal lines
;
;Uses 400 line VGA mode
;
DoDSO:
        sta          _COPCTL        ;clear watchdog
        bclr        VSYNC_PIN,_PTA  ;deassert VSYNC

        ldx         #50             ;gap of 50 lines

LineLoop3a:
        HWAIT
        dbnzx       LineLoop3a      ;next line

        lda         #16+8+4+2       ;unbuffered operation, no IRQ
                                         ;output compare, LUM set on match
                                         ;clear on overflow
        sta         _TSC0

        ;generate 350 lines in digital storage mode

        ldx         #BUFSIZ-1

        ;First line of three

LineLoop2:
        lda         STORAGE,x       ;3 read sample from buffer
        lsra        ;1
        lsra        ;1 convert to 6-bit
        add         #RENDER_START    ;2 offset
        sta         LUM_TIMER        ;3 load into LUM timer
        HWAIT

        ;worst case execution below = 33 clocks

        lda         TMR_ACC          ;3
        add         TMR_STEP         ;3
        sta         TMR_ACC          ;3 time to take a sample?
                                         ;(rate = TMR_STEP/256 *
                                         ;scan rate/3)?
        bcc         DoneSampling     ;3 no, skip

        lda         SAMPLE_IDX       ;3 fetch current sample index
        beq         DoneSampling     ;3 at start of buffer..
                                         ; then we are not sampling

        pshx        ;2 save current display index
        tax         ;1 index = sample pointer
        lda         _ADR             ;3 fetch ADC result
        sta         STORAGE,x        ;3 store new sample in buffer
        dec         SAMPLE_IDX       ;3 advance sample pointer

        pulx        ;2 restore current display index

DoneSampling:

        ;Second line of three

        HWAIT

        ;worst case execution below = 35 iclocks

```

```

        lda        SAMPLE_IDX            ;3 at the start of the buffer?
        bne        DoneTrigger           ;3 no, not looking for trigger

        lda        _ADR                  ;3
        sub        TRIGGER                ;3
        beq        DoneTrigger           ;3
        brset     RISING,FLAGS,ChkRising ;5 rising or falling edge?
ChkFalling:
        bcs        DoneTrigger           ;3
        bra        GotTrigger            ;3
ChkRising:
        bcc        DoneTrigger           ;3

        ;fall thru

GotTrigger:

        lda        #BUFSIZ-3             ;2
        sta        SAMPLE_IDX            ;3 start sampling

DoneTrigger:

        ;Third line of three

        HWAIT
        dbnzx     LineLoop2              ;3 next lines set, if applicable

        ;deactivate the luminance generation

        lda        #16+8                 ;unbuffered operation, no IRQ
        sta        _TSC0                 ;output compare, LUM clear
        clr        _TSC                  ;disable overflow interrupt

        ldx        #50                   ;remaining lines

LineLoop3:

        HWAIT
        dbnzx     LineLoop3              ;next lines

        ldx        #13                   ;front porch = 0.41ms = 13 lines
WtFrontPorch:

        HWAIT
        dbnzx     WtFrontPorch

        bset     VSYNC_PIN,_PTA          ;assert VSYNC
        ldx        #2                    ;vsync = 0.06ms = 2 lines
WtVSYNC:
        dbnzx     WtVSYNC
        bclr     VSYNC_PIN,_PTA          ;deassert VSYNC

        Keypress                          ;scan keypad

        ldx        #34                   ;back porch = 1.08ms = 34 lines
WtBackPorch:

        HWAIT
        dbnzx     WtBackPorch

        bclr     7,_TSC                  ;ack overflow IRQ

        lda        #64
        sta        _TSC                  ;enable overflow interrupt

        brset     FREERUN,FLAGS,GoFreerun ;exiting to freerun mode?
        jmp        DoDSO                  ;loop to next DSO screen
GoFreerun:
        jmp        DoFreerun              ;no, flip to freerun

        END

```