Pseudo-Random Noise Generator

Author: *Les Wolff, P.E.*
Goleta, California

INTRODUCTION:

A pseudo-random noise generator is a building block that is useful in many different communication systems. A few of the applications for a pseudo-random noise generator are:

- Data or voice encryption/decryption
- High performance sonar or radar range-finder
- Audio signal source for equalization or other functions

This paper describes a simple and inexpensive implementation of a pseudo-random noise generator, using a single 8 pin IC, the Microchip PIC12C508. The chip has a great deal of additional capability that can be used to incorporate other customized system functions as needed.

APPLICATION OPERATION:

The algorithm for generating the noise signal is quite simple. The process begins by setting up a 15 bit shift register as shown in Figure 1, with the initial value of all 15 bits set as 1's. The register is right shifted, and the lowest bit is transferred to the output register GP0. The lowest and second lowest bits are Exclusive OR'ed, and the result shifted into the highest bit position. The cycle is repeated, with each cycle producing a bit in a pseudo-random bit sequence. The pseudo-random bit sequence described repeats each 32,767 cycles.

The frequency of the pseudo-random noise generator output in this program is approximately 50 kHz, using the internal 4 MHz clock of the PIC12C508. The output frequency can be decreased by adding delays to the program loop, using a external clock, or using the internal clock/counter/prescaler of the PIC12C508. The algorithm, and many useful variations are described in detail in *The Art of Electronics* by Paul Horowitz and Winfield Hill, published by the Cambridge University Press. The repetition rate of the pseudo-random bit sequence can be changed by modifying the register length and the register taps as fully described in *The Art of Electronics*.

FIGURE 1: BLOCK DIAGRAM

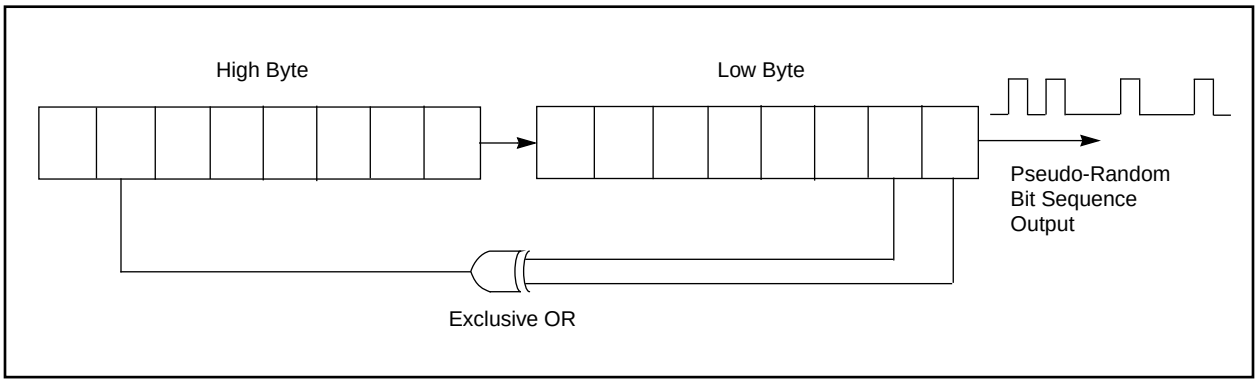
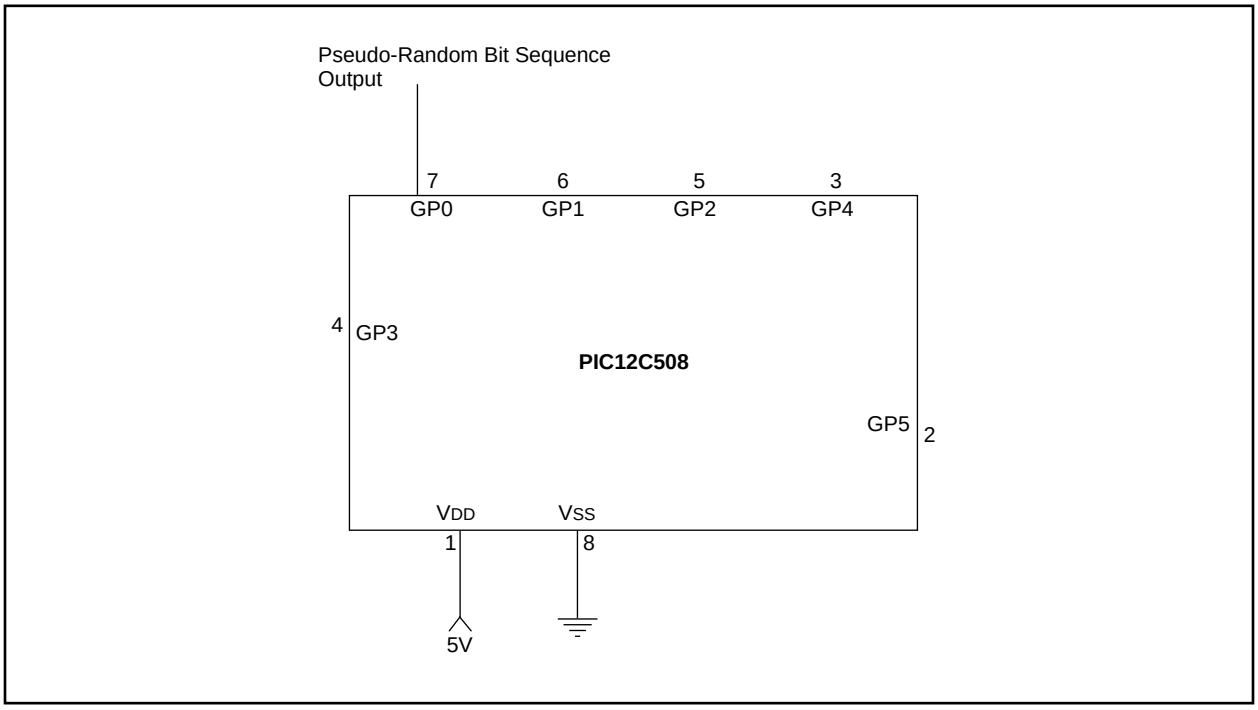


FIGURE 2: SCHEMATIC



APPENDIX A: SOURCE CODE

```
; Title:            Pseudo-Random Noise Generator
; File Name:        PRNG.asm
; Date:             October 25, 1997
; Processor:        PIC12C508
; Clock Freq:       4 MHz
; Description:      Generates a pseudo-random bit sequence
;                   with an output frequency of about 50 kHz.
;
         processor 12c508; define processor
;
         #include “pic12c508.inc”; include standard assembler routines
;
      __CONFIG  _CP_OFF&_WDT_OFF&_IntRC_OSC ;turn code protect off
;                                            turn watch dog timer off
;                                            select internal RC oscillator
; specify boot vector
;
BOOTC508       equ      0x0000        ; the 12C508 boots up at address 0x0000
;
; define memory locations of variables
HIGH_BYTE      equ      007           ; define high byte of shift register
LOW_BYTE       equ      008           ; define low byte of shift register
;
LOAD_OPTION    movlw    0xDF          ; zero bit 5 of option register to
               option                 ; disable timer 0 clock source
;
LOAD_TRIS      movlw    0x08          ; define GPIO 0-2, 4 & 5 as outputs; 3 as input
               tris     GPIO          ;
;
INIT_REGS      movlw    0xff          ;
               movwf    HIGH_BYTE     ; Initialize all bits of registers to 1
               movwf    LOW_BYTE      ;
;
GEN_PRBS       bcf      HIGH_BYTE,7   ; set shift-in value to 0
               rrf      HIGH_BYTE,1   ; shift high byte right
               rrf      LOW_BYTE,1    ; shift low byte right, shift out bit is carry bit in status
               movfw    STATUS        ; load status register
               andlw    0x01          ; isolate carry bit
               movfw    GPIO          ; output new prbs value
GEN_SHIFT_IN   xorwf    LOW_BYTE,0    ; determine value to shift into high bit of register
               andlw    0x01          ; isolate shift-in value, sets zero flag if zero
               btfss    STATUS,Z      ; if shift-in bit is zero, skip the following instruction
               bsf      HIGH_BYTE,6   ; set high bit to 1
               goto     GEN_PRBS

               end
```

NOTES: