

# DIGITAL TRANSMITTER: OPTIMIZATION EXERCISE WITH QPSK ON TI TMS320C54X\*

Douglas L. Jones  
Swaroop Appadwedula  
Matthew Berry  
Mark Haun  
Jake Janovetz  
Michael Kramer  
Dima Moussa  
Daniel Sachs  
Brian Wade

This work is produced by The Connexions Project and licensed under the  
Creative Commons Attribution License <sup>†</sup>

## Abstract

You will implement and optimize for execution time a quadrature phase-shift keying (QPSK) digital transmitter and pseudo-noise (PN) sequence generator.

## 1 Implementation

You will implement the complete system shown in Digital Transmitter: Introduction to Quadrature Phase-Shift Keying<sup>1</sup>. Then you will optimize the system for execution time. The optimization process will probably be much easier if you plan for optimization before you begin any programming.

### 1.1 PN generator

Once you have planned your program strategy, implement the PN generator from Digital Transmitter: Introduction to Quadrature Phase-Shift Keying<sup>2</sup> and verify that it is working. You may wish to refer to the

---

\*Version 2.8: Feb 25, 2004 12:01 pm US/Central

<sup>†</sup><http://creativecommons.org/licenses/by/1.0>

<sup>1</sup>"Digital Transmitter: Introduction to Quadrature Phase-Shift Keying", Figure 1: QPSK Transmitter  
<<http://cnx.org/content/m10042/latest/#fig1>>

<sup>2</sup>"Digital Transmitter: Introduction to Quadrature Phase-Shift Keying", Figure 2: Pseudo-Noise Generator  
<<http://cnx.org/content/m10042/latest/#fig2>>

description of assembly instructions for logical operations in *Section 2-2* of the *Mnemonic Instruction Set*[?] reference. Initialize the shift register to one.

## 1.2 Transmitter

For your transmitter implementation, use the signal constellation shown in Digital Transmitter: Introduction to Quadrature Phase-Shift Keying, a digital carrier frequency  $\omega_c$  of  $\frac{\pi}{2}$  and a digital symbol period of  $T_{\text{syms}} = 16$  samples.

Viewing the transmitted signal on the oscilloscope may help you determine whether your code works properly, but you should check it more carefully by setting breakpoints in Code Composer and using the **Memory** option from the **View** menu to view the contents of memory. A **vector signal analyzer (VSA)** provides another method of testing, which is described in Vector Signal Analyzer: Testing a QPSK Transmitter<sup>3</sup>.

## 1.3 Grading

One objective of this exercise is to teach optimization and efficient code techniques. For this reason, your performance will be judged primarily on the total execution time of your system. Note that by execution time we mean cycle count, not the number of instructions in your program. Remember that several of the TI TMS320C54xx instructions take more than one cycle. The multicycle instructions are primarily the multi-word instructions, including instructions that take immediates, like `stm`, and instructions using direct addressing of memory (such as `ld *(temp),A`). Branch and repeat statements also require several cycles to execute. The *Mnemonic Instruction Set*[?] reference will tell you how many cycles required for each instruction; make sure you look at the cycle count for the syntax you are using. It is also possible to use the debugger to determine the number of cycles used by your code.

You will be graded based on the number of cycles used between the return from one `WAITDATA` call and the arrival at the next `WAITDATA` call. If the number of cycles between one `WAITDATA` and the next is variable, the maximum possible number of cycles will be used. You must use the `core.asm`<sup>4</sup> file as provided; this file may not be modified. You explicitly may not change the number of samples read and written by each `WAITDATA` call! We reserve the right to test your code by substituting the test vector `core` file<sup>5</sup>.

---

<sup>3</sup>"Vector Signal Analyzer: Testing a QPSK Transmitter on Hewlett Packard 89440A"  
<<http://cnx.org/content/m10667/latest/>>

<sup>4</sup><http://cnx.rice.edu/author/workgroups/90/m10017/core.asm>

<sup>5</sup><http://cnx.rice.edu/author/workgroups/90/m10017/vectcore.asm>