Freescale Semiconductor
Application Note

Document Number: AN3752
Rev. 0, 08/2008

XGATE Library: Quadrature Decoder Driver

Decoding Quadrature Signals with the XGATE Coprocessor

by: Steven McLaughlin
Microcontroller Solutions Group, East Kilbride, Scotland

This application note explains how to use the XGATE coprocessor to decode quadrature signals — two square wave signals that are displaced by 90° from one another — which provide positional information needed by the application.

The software described in this application note is available as a ZIP file (AN3752SW), which can be downloaded from www.freescale.com.

Contents

1 Introduction

The field of motor applications is just one example where continuous feedback on velocity and position is required. A quadrature decoder has the ability to provide such information by measuring the many pulses — an ideal task for XGATE. The XGATE coprocessor is a 16-bit RISC core with an instruction set optimized for data manipulation and interrupt loading, thereby reducing CPU loading.

© Freescale Semiconductor, Inc., 2008. All rights reserved.

2 Quadrature Signals and XGATE Significance

Quadrature signals are characterized as two square waves, one of which is displaced by 90° as shown in Figure 1.

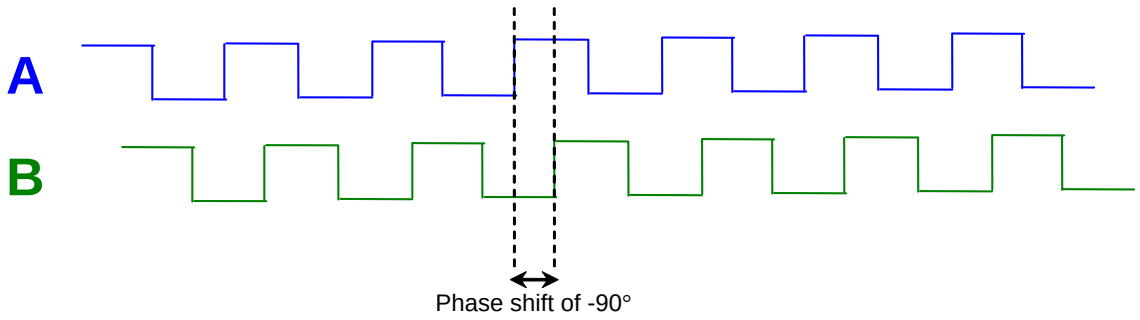

Figure 1. Quadrature Signals, phase shift of -90° between signals A and B

These signals are generated by various methods. One common method employs a spinning disc (with transparent and non-transparent areas to allow or block light transmission between a light source and sensor — Figure 2), attached to a motor shaft, which spins through and breaks the light at a fixed period, thus generating square wave signals of known frequency with a 90° phase shift — such as those shown in Figure 1.

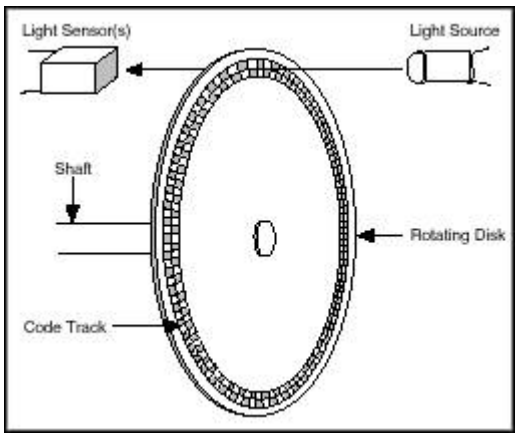

Figure 2. An example of generating a quadrature signal via a spinning encoder and light sensor

The significance of the phase difference is that it provides positional information about the application. For example, in a motor application, if waveform A is leading waveform B, then we can say that the motor is rotating in a particular direction; on the other hand, if A is lagging B then the motor is deemed to be rotating in the opposite direction.

2.1 Where is XGATE used?

The majority of real-time embedded applications require many interrupt driven processes to service simple functions at high execution rates. The quadrature signals can generate interrupts that can easily be

processed by the main CPU. However, by utilizing the interrupt handling capabilities of XGATE, the main CPU can be freed for other tasks.

Figure 3 illustrates the reading of quadrature signals through edge detection by the timer module and using software executed by XGATE to monitor interrupts on specific edges.

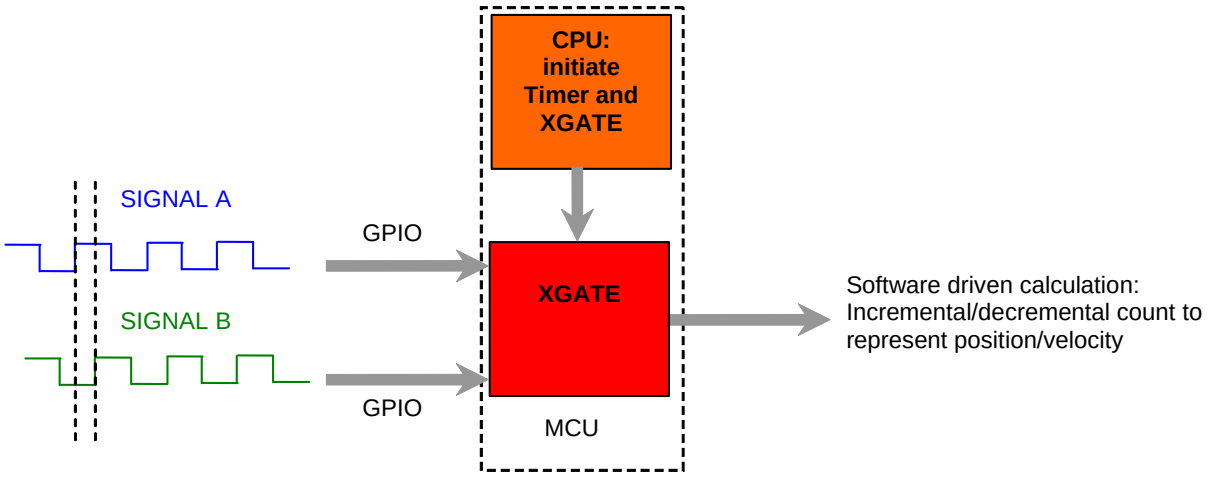

Figure 3. Diagrammatic representation of XGATE detecting quadrature signals on GPIO

The XGATE is programmed to interrupt on the edge of one of the signals (software will dictate whether this is a rising edge or a falling edge) and, through a simple software routine, will enable a count that will represent the position of an application.

By way of a motor example, there are two options, clockwise or counter-clockwise rotation, the direction of rotation being determined by the following software process and illustrated in Figure 4:

1. XGATE is programmed to interrupt on the rising edge of signal A.
2. On detection of a rising edge, an XGATE interrupt service routine (XSR) is entered, and a short software routine reads whether signal B is high or low. The rotational direction, clockwise or counter-clockwise, would also be associated with the high or low states of signal B. This should be determined by the software.

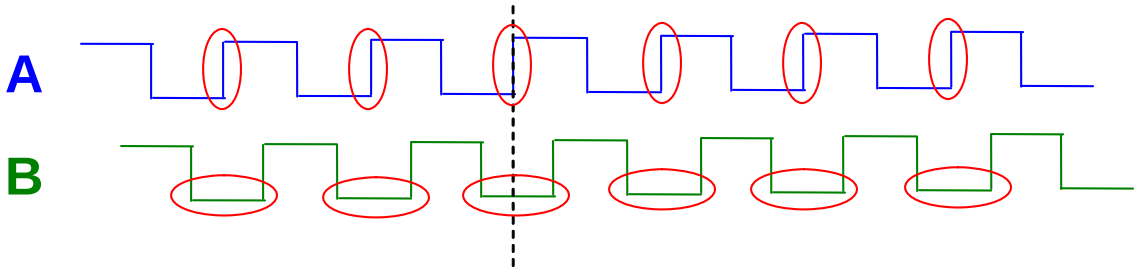

Figure 4. Example of rising edge detection on one signal (A) and XGATE routine programmed to recognize the level of the second signal (B)

3 XGATE

XGATE is a 16-bit RISC (reduced instruction set computer) core that has access to the same peripherals as the CPU, for example, RAM, flash, and registers. To take advantage of the higher bus speeds at which it can operate, XGATE is usually executed from RAM — XGATE can run up to twice as fast as the main CPU (RAM has shorter read/write times than flash). It is advisable to enable RAM protection after downloading XGATE code (normally from flash to RAM) to prevent overwriting of XGATE code or RAM variables from either the CPU or XGATE itself. The XGATE interrupt vector table must also be initialized appropriately.

As this application note has been written to outline the decoding of quadrature signals using XGATE, further information on the internal operation of XGATE can be found in the following Freescale Semiconductor publications.

- *"Introducing the XGATE Module to Consumer and Industrial Application Developers" (AN3224)*
- *"How to Configure and Use XGATE on S12X Devices" (AN2685)*
- *"XGATE Library: Using the Freescale XGATE Software Library" (AN3145*
- Freescale web site – http://www.freescale.com (keyword: XGATE Solutions)

4 Program Initialization

The basic steps required to run the software are as follows.

1. Set up the required Port T I/O as inputs (enhanced capture timer (ECT)/timer (TIM) module). The two signals from a quadrature encoder (as described in Figure 1 for example) should be input to the microcontroller here. The demonstration software uses Port T7 and T6.
2. Monitor XGATE interrupt being entered on I/O port (set as outputs). In the case of the demonstration software, Port A0 and Port A1 indicate leading and lagging signals. Variables 'decrement' and 'increment' can also be used in the debugging environment to complement the output signals.

The decoding can be performed either by the main CPU or XGATE core. When using XGATE, software interrupt 0 is called to initialize the XGATE driver. A *"#define"* is used within the program to enable XGATE; this called by *'#ifdef INITALISE_BY_XGATE'* shown in Figure 5.

```
#ifdef INITIALISE_BY_XGATE
 XGATE.xgswt.word = XGSWT0M | XGSWT0;
#else
    Signal_Capture_Init();
#endif
```

Figure 5. Compiler setting to initialize by XGATE and call within the Main program

5 Performance Measurements and Limitations

A brief analysis of this software shows that the XGATE edge detection thread takes approximately 30 cycles to complete. Moreover, it should also be remembered that, when the RISC core is triggered by an interrupt request, it first executes a vector fetch sequence, which performs three bus accesses.

1. A vector fetch to fetch the initial content of the program counter
2. A vector fetch to fetch initial content of the data segment pointer (R1)
3. A program word fetch to load the initial op-code

When any XGATE thread is complete, an ‘RTS’ instruction (2 cycles) represents termination. The three initial bus accesses plus the concluding RTS adds 5 cycles, but this does not account for any access to RAM that is required. The software provided takes approximately 40 cycles to complete, which accounts for entry to and exit from the thread, as well as RAM accesses.

Some analysis carried out on the code has found that this software will not function properly for encoders producing square wave quadrature signals above 1MHz.

Some measurements in Table 1 show the average load of the XGATE core when running the edge detection interrupt thread. These measurements were taken using the procedures set out in AN3253 (XGATE Library: Load Measurement, Measuring the XGATE Coprocessor Load). As described in AN3253, there will be some latency and extra processing times which have not been accounted for. These values in should be used as a guide only, and are not recommended as a basis for a design.

Table 1. XGATE Loading at Different Frequencies of Quadrature Input Signals

Input Frequency (kHz)	Average Load (%)
100	3.75
200	7.5
300	11.2
400	15.0
500	18.7
600	22.5

6 Future Refinements

The current code emulates an incremental encoder, allowing good judgement to be made on the position (and velocity) of the encoder. However, it would be possible to expand the code to allow the function of an absolute encoder, that is, one that is capable of determining the absolute position of an object. The only disadvantage associated with this is the additional hardware required to allow absolute encoding, which would add to system cost. Moreover, it would be wise to adjust the code to count the pulses, not the edges as it does currently.

The following adjustments would be simple to implement with the current demonstration software. It is possible to perform a calculation to determine the number of revolutions; the number of generated pulses divided by the number of pulses forming one rotation would give the number of revolutions (which is commented out in the demonstration code in the XGATE routine). It is important to remember that XGATE

is not optimized for mathematical operations; therefore, if the number of revolutions calculation was programmed on XGATE, this would have an impact upon the XGATE load. It would be possible to undertake this calculation on the CPU if load and timings could not be afforded on XGATE.

For motor applications, it is possible to adjust the code to add a third pulse, to occur when one revolution has occurred. This single pulse could be used for precise determination of a reference position. This would be relatively inexpensive to implement, as only a software update would be required, and it would be possible to monitor a signal related to a reference position on a GPIO of the microcontroller.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3752
Rev. 0
08/2008

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org

© Freescale Semiconductor, Inc. 2008. All rights reserved.

freescale™
semiconductor