Rotary Pulse Generators and other Lab 3 Considerations

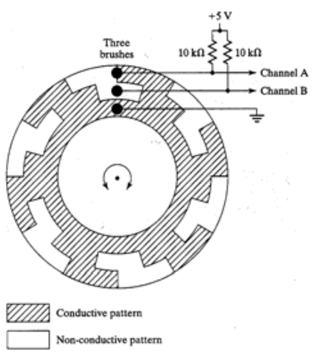
55:036

Embedded Systems and Systems Software

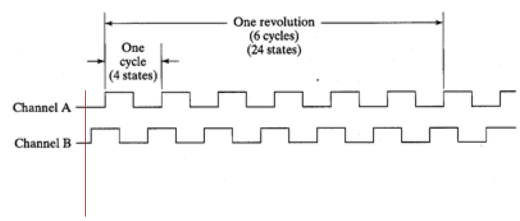
Rotary Encoders (Pulse Generators)

Specifications	3315	ECW	EPS	EAW
Product Photo				
Type	Incremental	Incremental	Incremental	Absolute
Package Diameter	9mm	22mm	22mm x 27mm	22mm
Sealable	IP67	IP40	IP40	IP40
Detents	No	Yes	Yes	Yes
Switch	No	No	Yes	No
Shaft	Plastic	Plastic	Plastic	Plastic
Bushing	Plastic/Metal	Plastic	Plastic	Plastic
Terminal Configuration	5 Pin Axial or Radial	3 Pin Axial or Radial	5 Pin Axial or Radial	10 Pin Axial or Radial
Resolution Range	6, 16	6, 9, 12, 24, 36	6, 9, 12, 24, 36	128
Rotational Speed	120 rpm max.	120 rpm max.	120 rpm max.	120 rpm max.
Contact Rating	TTL compatible	10 mA @ 10 VDC	10 mA @ 10 VDC	10 mA @ 10 VDC
Rotational Life	100,000 cycles @ 6 ppr / 25.00 cycles @ 16 ppr	200,000 shaft revolutions	200,000 shaft revolutions	50,000 shaft revolutions
Material Declaration Sheet	material data	ECW1J: material data		EAW CEV: material data EAW FR4: material data

Rotary Encoder Internals

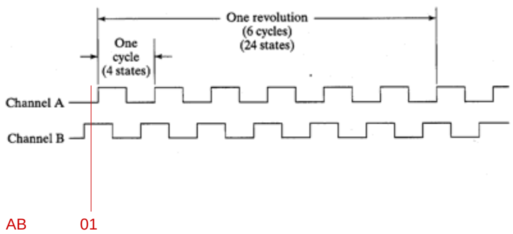


Gray-Code Output

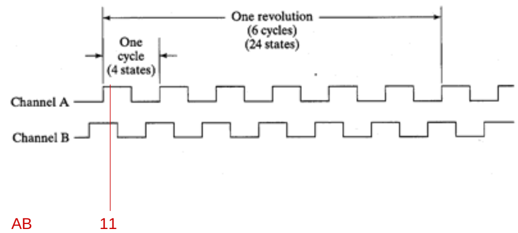


AB 00

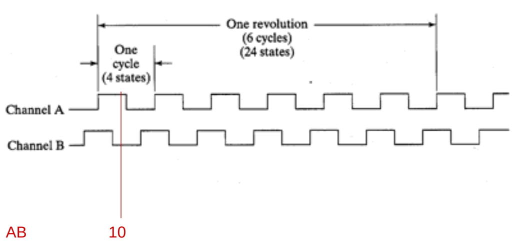
Gray-Code Output



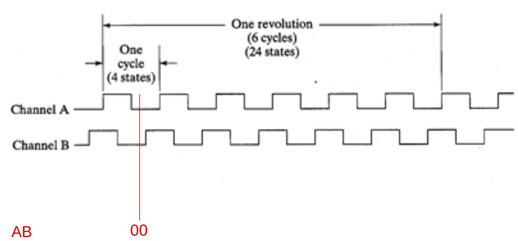
Gray-Code Output



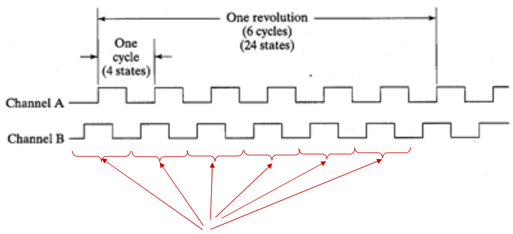
Gray-Code Output



Gray-Code Output



Gray-Code Output



Gray code sequence 00 => 01 => 11 => 10
repeats six times per revolution

Rotary Encoders—Some Considerations

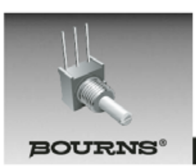
Features
- Miniature package for design flexibility
- Long operating life
- Conductive plastic element
- Bushing or PC board mount
- Quadrature output

BOURNS®
3315 - 9 mm Square Sealed Incremental Encoder

Electrical Characteristics

Output	2-bit gray code, Channel A leads Channel B electrically turning clockwise (CW)
Closed Circuit Resistance	5 ohms maximum
Contact Rating	TTL compatible loads
Insulation Resistance (500 VDC)	1,000 megohms minimum
Dielectric Withstanding Voltage	
Sea Level	900 VAC minimum
Electrical Travel	Continuous
Contact Bounce	5 milliseconds maximum
RPM (Operating)	120 maximum

Rotary Encoders—Some Considerations



Features
- Miniature package for design flexibility
- Long operating life
- Conductive plastic element
- Bushing or PC board mount
- Quadrature output

BOURNS®
3315 - 9 mm Square Sealed Incremental Encoder

Electrical Characteristics

Output	2-bit gray code, Channel A leads Channel B electrically turning clockwise (CW)
Closed Circuit Resistance	5 ohms maximum
Contact Rating	TTL compatible loads
Insulation Resistance (500 VDC)	1,000 megohms minimum
Dielectric Withstanding Voltage	
Sea Level	900 VAC minimum
Electrical Travel	Continuous
Contact Bounce	5 milliseconds maximum
RPM (Operating)	120 maximum

Contacts may “bounce” during transitions. This can cause erroneous readings-- e.g. a 00 => 01 transition may be read as: 00 => 01 => 00 => 01.

Rotary Encoders—Some Considerations

Features
- Miniature package for design flexibility
- Long operating life
- Conductive plastic element
- Bushing or PC board mount
- Quadrature output

BOURNS®
3315 - 9 mm Square Sealed Incremental Encoder

Electrical Characteristics

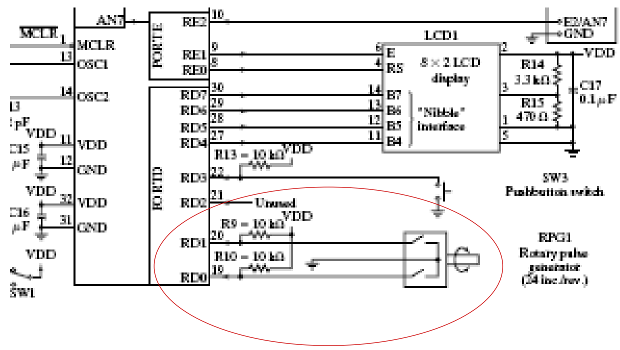
Output	2-bit gray code, Channel A leads Channel B electrically turning clockwise (CW)
Closed Circuit Resistance	5 ohms maximum
Contact Rating	TTL compatible loads
Insulation Resistance (500 VDC)	1,000 megohms minimum
Dielectric Withstanding Voltage	
Sea Level	900 VAC minimum
Electrical Travel	Continuous
Contact Bounce	5 milliseconds maximum
RPM (Operating)	120 maximum

Device has a maximum rotational speed of 120 RPM (2 revs/sec)
This corresponds to a max. count rate of 48 counts/sec or a minimum count period of 20.8 msec./count

Reading the Encoder Output

- An encoder sampling rate of 100 Hz (10 msec. per sample) will:
  - Ensure that no transition is missed
  - Mitigate contact bounce problems (since the device will never be sampled more than once during a bounce interval.
- Hence your lab 3 program should sample the rotary encoder at approximately a constant 100 Hz rate.

RPG Connection on QwikFlash




```
;;;;;;; RPG subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine deciphers RPG changes into values of DELRPG of 0, +1, or -1.
; DELRPG = +1 for CW change, 0 for no change, and -1 for CCW change.

RPG
    clrf  DELRPG          ;Clear for "no change" return value
    movf  PORTD,W          ;Copy PORTD into W
    movwf  TEMP            ; and TEMP
    xorwf  OLDPORTD,W       ;Any change?
    andlw  B'00000011'      ;If not, set Z flag
    IF_  .NZ.              ;If the two bits have changed then...
      rrcf  OLDPORTD,W      ;Form what a CCW change would produce
      IF_  .C.             ;Make new bit 1 = complement of old bit 0
        bcf  WREG,1
      ELSE_
        bsf  WREG,1
      ENDIF_
      xorwf  TEMP,W         ;Did the RPG actually change to this output?
      andlw  B'00000011'
      IF_  .Z.             ;If so, then change  DELRPG to -1 for CCW
        decf  DELRPG,F
      ELSE_                ;Otherwise, change DELRPG to  +1 for CW
        incf  DELRPG,F
      ENDIF_
    ENDIF_
    movff  TEMP,OLDPORTD    ;Save PORTD as OLDPORTD for ten ms from now
    return
```


```
;;;;;;; RPG subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine deciphers RPG changes into values of DELRPG of 0, +1, or -1.
; DELRPG = +1 for CW change, 0 for no change, and -1 for CCW change.

RPG
    clrf  DELRPG          ;Clear for "no change" return value
    movf  PORTD,W          ;Copy PORTD into W
    movwf  TEMP            ; and TEMP
    xorwf  OLDPORTD,W       ;Any change?
    andlw  B'00000011'      ;If not, set Z flag
    IF_  .NZ.              ;If the two bits have changed then...
      rrcf  OLDPORTD,W      ;Form what a CCW change would produce
      IF_  .C.             ;Make new bit 1 = complement of old bit 0
        bcf  WREG,1
      ELSE_
        bsf  WREG,1
      ENDIF_
      xorwf  TEMP,W         ;Did the RPG actually change to this output?
      andlw  B'00000011'
      IF_  .Z.             ;If so, then change  DELRPG to -1 for CCW
        decf  DELRPG,F
      ELSE_                ;Otherwise, change DELRPG to  +1 for CW
        incf  DELRPG,F
      ENDIF_
    ENDIF_
    movff  TEMP,OLDPORTD    ;Save PORTD as OLDPORTD for ten ms from now
    return
```


test to see if RPG output has changed since last time. OLDPORTD holds the previous reading.

```
;;;;;;; RPG subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine decyphers RPG changes into values of DELRPG of 0, +1, or -1.
; DELRPG = +1 for CW change, 0 for no change, and -1 for CCW change.

RPG
        clrf  DELRPG            ;Clear for "no change" return value
        movf  PORTD,W           ;Copy PORTD into W
        movwf  TEMP             ; and TEMP
        xorwf  OLDPORTD,W       ;Any change?
        andlw  B'00000011'      ;If not, set Z flag
        IF_  .NZ.               ;If the two bits have changed then...
          rrcf  OLDPORTD,W      ;Form what a CCW change would produce
          IF_  .C.              ;Make new bit 1 = complement of old bit 0
            bcf  WREG,1
          ELSE_
            bsf  WREG,1
          ENDIF_
          xorwf  TEMP,W         ;Did the RPG actually change to this output?
          andlw  B'00000011'
          IF_  .Z.              ;If so, then change  DELRPG to -1 for CCW
            decf  DELRPG,F
          ELSE_                 ;Otherwise, change DELRPG to  +1 for CW
            incf  DELRPG,F
          ENDIF_
        ENDIF_
        movff  TEMP,OLDPORTD    ;Save PORTD as OLDPORTD for ten ms from now
        return
```

If so, figure out if the rotation was clockwise or counter-clockwise and set DELRPG accordingly

Detecting the Direction of Rotation

Clockwise (positive) rotation pattern:

00 -> 01 -> 11-> 10 -> 00 …

Counter-clockwise (negative) rotation pattern:

00 -> 10 -> 11-> 01 -> 00 …

Detecting the Direction of Rotation

Clockwise (positive) rotation pattern:

≠ ≠ ≠ ≠

00 -> 01 -> 11-> 10 -> 00 …

= = = =

Counter-clockwise (negative) rotation pattern:

= = = =

00 -> 10 -> 11-> 01 -> 00 …

≠ ≠ ≠ ≠

```
;;;;;;; RPG subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine decyphers RPG changes into values of DELRPG of 0, +1, or -1.
; DELRPG = +1 for CW change, 0 for no change, and -1 for CCW change.

RPG
        clrf  DELRPG            ;Clear for "no change" return value
        movf  PORTD,W           ;Copy PORTD into W
        movwf  TEMP             ; and TEMP
        xorwf  OLDPORTD,W       ;Any change?
        andlw  B'00000011'      ;If not, set Z flag
        IF_  .NZ.               ;If the two bits have changed then...
          rrcf  OLDPORTD,W      ;Form what a CCW change would produce
          IF_  .C.              ;Make new bit 1 = complement of old bit 0
            bcf  WREG,1
          ELSE_
            bsf  WREG,1
          ENDIF_
          xorwf  TEMP,W         ;Did the RPG actually change to this output?
          andlw  B'00000011'
          IF_  .Z.              ;If so, then change  DELRPG to -1 for CCW
            decf  DELRPG,F
          ELSE_                 ;Otherwise, change DELRPG to  +1 for CW
            incf  DELRPG,F
          ENDIF_
        ENDIF_
        movff  TEMP,OLDPORTD    ;Save PORTD as OLDPORTD for ten ms from now
        return
```

Save this RPG reading for the next call

Rate-Sensitivity

- Consider Lab 3 requirements:
  - Suppose that the RPG is used to adjust the duty cycle of a square wave over the range 0.1% to 99.9% in 0.1% increments.
  - If:

  one RPG count ⇔ 0.1% duty cycle adjustment how many revolutions of the RPC will be required to adjust across the entire duty cycle range?

Rate-Sensitivity

- Consider Lab 3 requirements:
  - Suppose that the RPG is used to adjust the duty cycle of a square wave over the range 0.1% to 99.9% in 0.1% increments.
  - If:

  one RPG count ⇔ 0.1% duty cycle adjustment how many revolutions of the RPG will be required to adjust across the entire duty cycle range?

  1000/24 = 41.67 revolutions

Rate-Sensitivity

- If

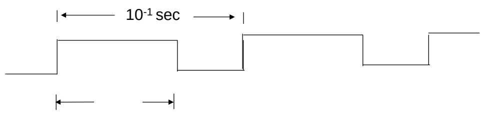
  one RPG count ⇔0.1% duty cycle adjustment how many revolutions of the RPG will be required to adjust across the entire duty cycle range?

  1000/24 = 41.67 revolutions
- Can reduce this by using a rate-sensitive approach
  - Use larger increment/decrement amounts when RPG is being turned rapidly
  - The text presents a rate-sensitive RPG subroutine

The Rate-Sensitive RPG Approach

- Set a Threshold T
- If changes in RPG output occur more than T*10 msec. apart, use “slow change”
  - e.g. increment/decrement DELRPG by 1
- if changes in RPG output are closer than T*10msec apart, use “fast change”
  - e.g. increment/decrement DELRPG by 2
- You can use this algorithm (Figure 8-4 in the text) as a starting point but you may need to adapt it.

Adjusting the Square Wave Duty Cycle



DC($10^{-1}/10^{2}$) sec
= DC(10^{-3}) sec

Where $0.1 < DC < 99.9$

So, Your Program Must:

- Read the RPG at 10 msec intervals
- Adjust the current waveform duty cycle up or down according to detected RPG rotation
  - increment or decrement the duty cycle percentage
  - determine the required timer value(s) to time waveform on-time and off-time for this duty cycle.
  - reload the timer(s) to begin timing this new duty cycle

Lab 3: Some Concluding Comments

- You may need two (or more) timers
  - one to time the 10 msec. RPG read interval
  - one (or more) to time the duty cycle
- You cannot build this program around a 10 msec. main loop like many of the examples in the book
  - Need to time duty cycle intervals much smaller than 10 msec.
- Think carefully about the structure and design of your program before you start to write code