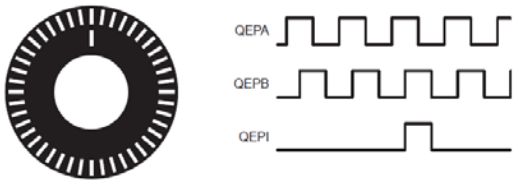
03-2 Rotary Encoders - eQEP

Measuring Rotation

Rotary Encoders

- Rotary encoder measure rotation by using two switches that open and close 90° out of phase with each other
- They have two inputs, **A** and **B**

Figure 15-130. Optical Encoder Disk



Reading the Encoder

- We could hook **A** and **B** to GPIO pins and read them ourselves
- The Bone has hardware for reading encoders called the Enhanced Quadrature Encoder Pulse (eQEP) Module
- See section 15.4 of the TRM

eQEP

- Let's use eQEP 2
- Derek Molloy's table show they appear on pins P8_11 and 12

Beaglebone Black P8 Header

Head_pin	$PINS	ADDR/OFFSET	GPIO NO.	Name	Mode7	Mode6	Mode5	Mode4	Mode3	Mode2
P8_01				DGND						
P8_02				DGND						
P8_03	6	0x818/018	38	GPIO1_6	gpio1[6]					
P8_04	7	0x81c/01c	39	GPIO1_7	gpio1[7]					
P8_05	2	0x808/008	34	GPIO1_2	gpio1[2]					
P8_06	3	0x80c/00c	35	GPIO1_3	gpio1[3]					
P8_07	36	0x890/090	66	TIMER4	gpio2[2]					timer4
P8_08	37	0x894/094	67	TIMER7	gpio2[3]					timer7
P8_09	39	0x89c/09c	69	TIMER5	gpio2[5]					timer5
P8_10	38	0x898/098	68	TIMER6	gpio2[4]					timer6
P8_11	13	0x834/034	45	GPIO1_13	gpio1[13]			eQEP2B_in	mmc2_dat1	mmc1_dat5
P8_12	12	0x830/030	44	GPIO1_12	gpio1[12]			EQEP2A_IN	MMC2_DAT0	MMC1_DAT4
P8_13	9	0x824/024	23	EHRPWM2B	gpio0[23]			ehrpwm2B	mmc2_dat5	mmc1_dat1
P8_14	10	0x828/028	26	GPIO0_26	gpio0[26]			ehrpwm2_tripzone_in	mmc2_dat6	mmc1_dat2
P8_15	15	0x83c/03c	47	GPIO1_15	gpio1[15]			eQEP2_strobe	mmc2_dat3	mmc1_dat7
P8_16	14	0x838/038	46	GPIO1_14	gpio1[14]			eQEP2_index	mmc2_dat2	mmc1_dat6
P8_17	11	0x82c/02c	27	GPIO0_27	gpio0[27]			ehrpwm0_synco	mmc2_dat7	mmc1_dat3
P8_18	35	0x88c/08c	65	GPIO2_1	gpio2[1]	mcasp0_fsr			mmc2_clk	gpmc_wait1
P8_19	8	0x820/020	22	EHRPWM2A	gpio0[22]			ehrpwm2A	mmc2_dat4	mmc1_dat0
P8_20	33	0x884/084	63	GPIO1_31	gpio1[31]					mmc1_cmd
P8_21	32	0x880/080	62	GPIO1_30	gpio1[30]					mmc1_clk

Configuring the Encoder

```
/ {
    compatible = "ti,beaglebone", "ti,beaglebone-black";

    /* identification */
    part-number = "bone_eqep2";
    version     = "00A0";

    fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            pinctrl_eqep2: pinctrl_eqep2_pins {
                pinctrl-single,pins = <

                    0x038 0x24  /* P8_16 = GPIO2_12 = EQEP2_index,  MODE4 */
                    0x03C 0x24  /* P8_15 = GPIO2_13 = EQEP2_strobe, MODE4 */
                    0x030 0x34  /* P8_12 = GPIO2_10 = EQEP2A_in,    MODE4 */
                    0x034 0x34  /* P8_11 = GPIO2_11 = EQEP2B_in,    MODE4 */
                >;
            };
        };
    };
```

Hardware address

Value

From: https://groups.google.com/forum/#!searchin/beagleboard/eQep/beagleboard/Orp3tFcNgCc/mYacP_GkCQQJ

eQEP

Beaglebone Black P8 Header

Head_pin	$PINS	ADDR/OFFSET	GPIO NO.	Name	Mode7	Mode6	Mode5	Mode4	Mode3	Mode2
P8_01				DGND						
P8_02				DGND						
P8_03	6	0x818/018	38	GPIO1_6	gpio1[6]					
P8_04	7	0x81c/01c	39	GPIO1_7	gpio1[7]					
P8_05	2	0x808/008	34	GPIO1_2	gpio1[2]					
P8_06	3	0x80c/00c	35	GPIO1_3	gpio1[3]					
P8_07	36	0x890/090	66	TIMER4	gpio2[2]					timer4
P8_08	37	0x894/094	67	TIMER7	gpio2[3]					timer7
P8_09	39	0x89c/09c	69	TIMER5	gpio2[5]					timer5
P8_10	38	0x898/098	68	TIMER6	gpio2[4]					timer6
P8_11	13	0x834/034	45	GPIO1_13	gpio1[13]			eQEP2B_in	mmc2_dat1	mmc1_dat5
P8_12	12	0x830/030	44	GPIO1_12	gpio1[12]			EQEP2A_IN	MMC2_DAT0	MMC1_DAT4
P8_13	9	0x824/024	23	EHRPWM2B	gpio0[23]			ehrpwm2B	mmc2_dat5	mmc1_dat1
P8_14	10	0x828/028	26	GPIO0_26	gpio0[26]			ehrpwm2_tripzone_in	mmc2_dat6	mmc1_dat2
P8_15	15	0x83c/03c	47	GPIO1_15	gpio1[15]			eQEP2_strobe	mmc2_dat3	mmc1_dat7
P8_16	14	0x838/038	46	GPIO1_14	gpio1[14]			eQEP2_index	mmc2_dat2	mmc1_dat6
P8_17	11	0x82c/02c	27	GPIO0_27	gpio0[27]			ehrpwm0_synco	mmc2_dat7	mmc1_dat3
P8_18	35	0x88c/08c	65	GPIO2_1	gpio2[1]	mcasp0_fsr			mmc2_clk	gpmc_wait1
P8_19	8	0x820/020	22	EHRPWM2A	gpio0[22]			ehrpwm2A	mmc2_dat4	mmc1_dat0
P8_20	33	0x884/084	63	GPIO1_31	gpio1[31]					mmc1_cmd
P8_21	32	0x880/080	62	GPIO1_30	gpio1[30]					mmc1_clk

GPIO Settings				
Bit 6	Bit 5	Bit 4	Bit 3	Bit 2,1,0
Slew Control	Receiver Active	Pullup/Pulldown	nable Pullup/Pulldow	Mux Mode
0 Fast	0 Disable	0 Pulldown select	0 Enabled	000 Mode 0 to
1 Slow	1 Enable	1 Pullup select	1 Disabled	111 Mode 7

Compile .dts file

```
bone$ dtc -O dtb -o bone_eqep2b-00A0.dtbo -b 0 -@
bone_eqep2b.dts
bone$ cp bone_eqep2b-00A0.dtbo /lib/firmware
bone$ echo bone_eqep2b > /sys/devices/bone_capemgr.*/slots
```

Read eQEP

```
bone$ cd /sys/devices/ocp.*/48304000.epwmss
bone$ ls -F
48304100.ecap/  48304200.ehrpwm/  modalias  subsystem@
48304180.eqep/  driver@           power/    uevent
bone$ cd 48304180.eqep/
bone$ ls -F
driver@  enabled  modalias  mode  period
position  power/  subsystem@  uevent
bone$ cat position
-50
```

Read from JavaScript

```
#!/usr/bin/env node
// This uses the eQEP hardware to read a rotary encoder
// echo bone_eqep2b > $SLOTS

var b = require('bonescript'),
    fs = require('fs');

var eQEP0 = "/sys/devices/ocp.3/48300000.epwmss/48300180.eqep/",
    eQEP1 = "/sys/devices/ocp.3/48302000.epwmss/48302180.eqep/",
    eQEP2 = "/sys/devices/ocp.3/48304000.epwmss/48304180.eqep/",
    eQEP = eQEP2;

var oldData,        // pervious data read
    period = 100;   // in ms
```

Read from JavaScript (2)

```
   period = 100;    // in ms
// Set the eEQP period, convert to ns.
fs.writeFile(eQEP+'period', period*1000000,
   function(err) {
      if (err) throw err;
      console.log('Period updated to ' +
                           period*1000000);
})

// Enable
fs.writeFile(eQEP+'enabled', 1,
    function(err) {
      if (err) throw err;
      console.log('Enabled');
})
```

Read from JavaScript (3)

```
setInterval(readEncoder, period); // Check state every 100 ms

function readEncoder(x) {
    fs.readFile(eQEP + 'position',
       {encoding: 'utf8'},
       printValue);
}
function printValue(err, data) {
    if (err) throw err;
    if (oldData !== data) {
       console.log('position: ‘ + data + ' speed:‘
              +(oldData-data));
           oldData = data;
       }
}
```