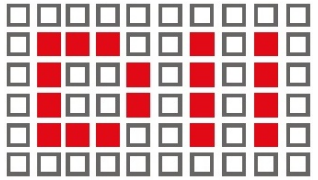
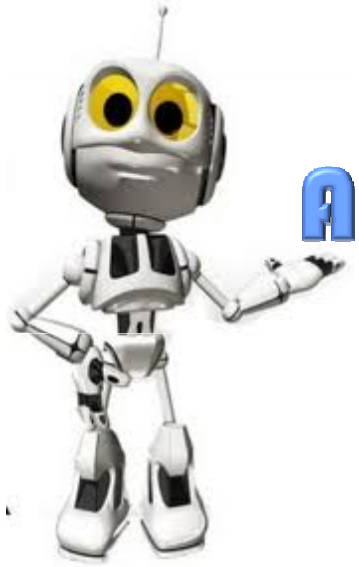
Dipartimento di Ingegneria dell'Informazione

Università Politecnica delle Marche





Automation laboratory

μ-CONTROLLER RENESAS RX63N

Optic rotary Encoder

YRDKRX63N – Encoder

Index of content

- Introduction
- Structure declaration
- Constant declaration
- Initializing function
- Interrupt
- Read Function

YRDKRX63N – Encoder

Introduction

In order to acquire data, MTU and TPU modules are used in phase counting mode.

They take as input the two phase signal from the encoder, throwing interrupts for position, speed and overflow evaluation at read calls.

Note that in the related code, some lines are encapsulated in IF0 ENDIF statement, because they were used in debug process.

Refers to hardware manual:

- MTU page 752 cap. 23
- Phase counting mode 4 page 824
- TPU page 931 cap. 25

YRDKRX63N – Encoder

Structure declaration

The used structure will contain the encoder state variables, that will be updated on every read call, via the function Query_Enc.

Its declaration is contained in the header file.

```
52 /* Main structure that contains the measured data of position and speed */
53 struct encoder_data {
54         long int position;                                      /* absolute position in the number of
   ▒▒▒▒▒▒▒▒pulses (with sign) starting from the initial position taken as equal to 0 */
55         int      speed;                                        /* speed in number of pulses per ms */
56         int      oscillazione;                                 /* indicates whether the encoder (the
   ▒▒▒▒▒▒▒▒tree) is stopped */
57         float position_in_degree;                              /* position with sign of degrees from the
   ▒▒▒▒▒▒▒▒initial position assumed equal to 0 */
58         float position_in_pigreek_radians_units;/* position in pi greek units with  sign */
59         float speed_in_degree_per_sec;                  /* speed in degrees per second with sign */
60         float speed_in_rad_per_sec;                     /* speed in radians per second with sign */
61 };
```

YRDKRX63N – Encoder

Constant declaration

This part is contained in the first part of the code file.

The first tree constants are based on the characteristics of the motor and encoder in use.

The last two define the maximum register counter.

```
30 #define ENCODER_PPR            500 /* Pulse per revolution  of the Encoder */
31
32 #define REDUCTION_GAIN       66 /* reduction ratio reverse total (approximate);


37 #define PHASE_COUNT_MODE         4 /* phase counting mode x4 normally */


46 /* Set position cycle */
47 #define CH0_TGRC_CYCLE   (unsigned short)(48000-1)
48                                 /* MTU2 CH0 TGRC compare match cycle(1ms)       */
49                                 /* 1ms / 21ns(@48MHz) = 48000                          */
50 /* Set speed cycle */
51 #define CH0_TGRA_CYCLE   (unsigned short)(40000-1)
52                                 /* MTU2 CH0 TGRC compare match cycle(0.84ms)    */
53                                 /* 0.84ms / 21ns(@48MHz) = 40000
```

YRDKRX63N – Encoder

Initializing Function

The first function called in the main, it's aim is to:

- Initialize the structure containing the data;
- Start the MTU peripheral;
- Set interrupts priority and enable them;
- Set logic port register;
- Start timer.

YRDKRX63N – Encoder

Initializing Function

In the first lines a dedicated function initialize the structure, the MTU and the ports.

```
119         Init_Encoder_1_vars();
120
121 #ifdef PLATFORM_BOARD_RDKRX63N
122 SYSTEM.PRCR.WORD = 0xA50B; /* Protect off */
123 #endif
124
125         /* ==== Setting of MTU2  ==== */
126         mtu2_init();


253     /* ==== Setting of PFC(Pin Function Controller) ==== */
254     pfc_init();
255
256     /* ==== MTU2 timer start register(TSTR) ==== */
257     MTU.TSTR.BYTE |= 0x03;      /* TCNT_0,1 performs count operation */
```

YRDKRX63N – Encoder

Initializing Function

The MTU register are declared here, and the chapter is in the hardware manual at page 757.

```
317     /* ---- Timer control register_1(TCR_1) ---- */
318
319             /* 7-5:CCLR[2:0]=b'001: TCNT cleared by TGRA input capture */
320             /* 4-3:CKEG[1:0]=b'00 : This setting is ignored
321                                     in phase counting mode */
322             /* 2-0:TPSC[2:0]=b'000: This setting is ignored
323                                     in phase counting mode */
324     MTU1.TCR.BYTE = 0x04; /* TCNT clearing disabled
325                                                 -External clock MTCLKA
326                                                 */
327
```

RENESAS YRDKRX63N – Encoder

Initializing Function

The MTU register are declared here, and the chapter is in the hardware manual at page 760 and follow.

```
335     /* ---- Timer mode register_1 (TMDR_1) ---- */
336     MTU1.TMDR.BYTE = 0x04;
337             /* 7-4:       =b'0000: reserve        */
338             /* 3-0:MD[3:0]=b'0100: Phase counting mode 1    */
339
340     /* ---- Timer I/O control register_1 (TIOR_1) ---- */
341     MTU1.TIOR.BYTE = 0xcc;
342             /* 7-4 IOB[3:0]=b'1100 :Input capture at generation of
343                                     TGRC_0 compare match */
344             /* 3-0 IOA[3:0]=b'1100 :Input capture at generation of
345                                     TGRA_0 compare match */
346
347     /* ---- Timer interrupt enable register_1 (TIER_1) ---- */
348     MTU1.TIER.BYTE = 0x33;
349             /* 7  : TTGE=b'0 : A/D Converter Start Request Disabled */
350             /* 6  :     =b'0 : reserve        */
351             /* 5  :TCIEU=b'1 : Underflow Interrupt Enable   */
352             /* 4  :TCIEV=b'1 : Overflow Interrupt Enable    */
353             /* 3-2:     =b'00: reserve        */
354             /* 1  :TGIEB=b'1 : TGRB Interrupt Enable        */
355             /* 0  :TGIEA=b'1 : TGRA Interrupt Enable        */
```

RENESAS YRDKRX63N – Encoder

Initializing Function

The port function is specified as usual.

```
427 void pfc_init(void)
428 {
429     /* porta P24, pin 25 conn. JN2 */
430         MPC.P24PFS.BYTE  = 0x02; /* 1 defines P24 to be MTCLKA, with no IRQ */
431
432     PORT2.PDR.BIT.B4 = 0;    /* Set P24 as input */
433
434     PORT2.PMR.BIT.B4 = 1;    /* Set P24 as peripheral function bit */
435
436     /* porta PA6, pin 6 conn. JN2 */
437     MPC.PA6PFS.BYTE  = 0x02; /* 1 defines PA6 to be MTCLKB, with no IRQ */
438
439     PORTA.PDR.BIT.B6 = 0;    /* Set PA6 as input */
440
441     PORTA.PMR.BIT.B6 = 1;    /* Set PA6 as peripheral function bit */
442 }
```

RENESAS YRDKRX63N – Encoder

Interrupt - Position

This interrupt simply get the register value and overflow count.

```
632 #pragma interrupt MTU1_TGIA1_isr(vect = VECT_MTU1_TGIA1, enable)
633 static void MTU1_TGIA1_isr(void)
634 {

646     TGRA1_data = MTU1.TGRA;    /* Read TGRA capture register */
647
648     Contatore_di_overflow = Under_over_flow_cnt;
649
```

Then set itself off, and flag the acquisition.

```
662     posizione_acquisita = 1;
663
664     /* Disable interrupt TGIA1 */
665         /* ---- Timer interrupt enable register_1 (TIER_1) ---- */
666         MTU1.TIER.BYTE &= 0xFE;
667                         /* 7  : TTGE=b'0 : A/D Converter Start Request Disabled */
668                         /* 6  :     =b'0 : reserve        */
669                         /* 5  :TCIEU=b'1 : Underflow Interrupt Enable   */
670                         /* 4  :TCIEV=b'1 : Overflow Interrupt Enable    */
671                         /* 3-2:     =b'00: reserve        */
672                         /* 1  :TGIEB=b'1 : TGRB Interrupt Enable        */
673                         /* 0  :TGIEA=b'0 : TGRA Interrupt Disable       */
674
```

YRDKRX63N – Encoder

Interrupt - Speed

This interrupt get data to evaluate speed, taking two position sample and overflow count.

```
700 #pragma interrupt MTU1_TGIB1_isr(vect = VECT_MTU1_TGIB1, enable)
701 static void MTU1_TGIB1_isr(void)
702 {

729         switch(state)
730         {
731          case 0:
732                  TGRB1_data_old = MTU1.TGRB;            /* Read TGRB capture register */
733                  state = 1;
734                  Read_overflow_old = (int)Under_over_flow_cnt;
735                  speed_sample = 0;
736                  break;
737          case 1:
738                  speed_sample = 1;
739                  state = 0;
740                  TGRB1_data_new = MTU1.TGRB;            /* Read TGRB capture register */
741
742                  Read_overflow = (int)Under_over_flow_cnt;
743
```

Then set itself off, as before, the acquisition flag is in the cases.

RENESAS YRDKRX63N – Encoder

Interrupt - Overflow

This interrupt update the overflow counter in both direction.

```
809 #pragma interrupt MTU1_TCIV_TCUV_isr(vect = VECT_ICU_GROUPE1, enable)
810 void MTU1_TCIV_TCUV_isr(void)
811 {
```

The direction is specified by these register, which cause if increment or decrement the overflow counter. In page 864 of hardware manual is described interrupt priority of MTU.

```
817                 if(ICU.GRP[GRP_MTU1_TCIV1].BIT.IS_MTU1_TCIV1)
818                 {
819                         /* ---- Clearing of TCIV Overflow Flag ---- */
820                         if(!ICU.GCR[GCR_MTU1_TCIV1].BIT.CLR_MTU1_TCIV1)
821                         {
822                                 ICU.GCR[GCR_MTU1_TCIV1].BIT.CLR_MTU1_TCIV1 = 1;
823                                 dummy = ICU.GCR[GCR_MTU1_TCIV1].BIT.CLR_MTU1_TCIV1;
824                         }
825                         Under_over_flow_cnt++;
826                 }
```

RENESAS YRDKRX63N – Encoder

Read Function

```
455 int Query_Enc_1(void)
456 {
457         // Variable used by the state machine
458  static unsigned char state = INIT_STATE_ENC;
459
460  // variable returned by the function
461  int ret_val = 0;
462
463  switch(state)
464  {
465   case INIT_STATE_ENC: // initial state
466            posizione_acquisita = 0; //reset of
467            speed_sample = 0;            //the variables
468
469            /* Enable interrupt TGIA1 and TGIB1 */
470            /* ---- Timer interrupt enable register_1 (TIER_1) ---- */
471            MTU1.TIER.BYTE = 0x33;
472                     /* 7  : TTGE=b'0 : A/D Converter Start Request Disabled */
473                     /* 6  :     =b'0 : reserve        */
474                     /* 5  :TCIEU=b'1 : Underflow Interrupt Enable   */
475                     /* 4  :TCIEV=b'1 : Overflow Interrupt Enable    */
476                     /* 3-2:     =b'00: reserve        */
477                     /* 1  :TGIEB=b'1 : TGRB Interrupt Enable        */
478                     /* 0  :TGIEA=b'1 : TGRA Interrupt Enable        */
479            state = READ_STATE_ENC; // change the state of the variable
480            ret_val = 0;
481           break;
482   case READ_STATE_ENC:
```

YRDKRX63N – Encoder

Read Function

The function is divided in two state:

- the first call switch the interruptors on;
- the second one get the red data from the registers and convert them in different measurement unit, after that, they are saved in the encoder structure.

The values can be retrieved in main function directly from the structure because it's declared as global.

YRDKRX63N – Encoder