**ENCODER 1**

/\* Implementazione degli interrupt per la gestione della lettura dell'encoder 1 \*/

**#pragma** interrupt MTU1\_TGIA1\_isr(vect = VECT\_MTU1\_TGIA1, enable)

**static** **void** **MTU1\_TGIA1\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**if**(ICU.IR[VECT\_MTU1\_TGIA1].BIT.IR)

{

ICU.IR[VECT\_MTU1\_TGIA1].BIT.IR = 0;

dummy = ICU.IR[VECT\_MTU1\_TGIA1].BIT.IR;

}

par\_encoder\_1.TGRA1\_data = MTU1.TGRA; /\* Read TGRA capture register \*/

par\_encoder\_1.Contatore\_di\_overflow = par\_encoder\_1.Under\_over\_flow\_cnt;

par\_encoder\_1.posizione\_acquisita = 1;

/\* Disable interrupt TGIA1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

MTU1.TIER.BYTE &= 0xFE;

/\* 7 : TTGE=b'0 : A/D Converter Start Request Disabled \*/

/\* 6 : =b'0 : reserve \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'1 : TGRB Interrupt Enable \*/

/\* 0 :TGIEA=b'0 : TGRA Interrupt Disable \*/

}

//INTERRUPT USATO PER IL CALCOLO DELLA VELOCITA' (TGRB MTU1)

**#pragma** interrupt MTU1\_TGIB1\_isr(vect = VECT\_MTU1\_TGIB1, enable)

**static** **void** **MTU1\_TGIB1\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**static** **unsigned** **char** state = 0;

**if**(ICU.IR[VECT\_MTU1\_TGIB1].BIT.IR)

{

ICU.IR[VECT\_MTU1\_TGIB1].BIT.IR = 0;

dummy = ICU.IR[VECT\_MTU1\_TGIB1].BIT.IR;

}

**switch**(state)

{

**case** 0:

par\_encoder\_1.TGRB1\_data\_old = MTU1.TGRB; /\* Read TGRB capture register \*/

state = 1;

par\_encoder\_1.Read\_overflow\_old = (**int**)par\_encoder\_1.Under\_over\_flow\_cnt;

par\_encoder\_1.speed\_sample = 0;

**break**;

**case** 1:

par\_encoder\_1.speed\_sample = 1;

state = 0;

par\_encoder\_1.TGRB1\_data\_new = MTU1.TGRB; /\* Read TGRB capture register \*/

par\_encoder\_1.Read\_overflow = (**int**)par\_encoder\_1.Under\_over\_flow\_cnt;

/\* Disable interrupt TGIB1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

MTU1.TIER.BYTE &= 0xFD;

/\* 7 : \*/

/\* 6 : \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'0 : TGRB Interrupt Disable \*/

/\* 0 :TGIEA=b' \*/

**break**;

**default**:

state = 0;

par\_encoder\_1.speed\_sample = 0;

}

}

//INTERRUPT USATO PER IL CONTEGGIO DEGLI OVERFLOW-UNDERFLOW DI TCNT1 CON SEGNO

**#pragma** interrupt MTU1\_TCIV\_TCUV\_isr(vect = VECT\_ICU\_GROUPE1, enable)

**void** **MTU1\_TCIV\_TCUV\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**do** {

**if**(ICU.GRP[*GRP\_MTU1\_TCIV1*].BIT.IS\_MTU1\_TCIV1)

{

/\* ---- Clearing of TCIV Overflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_MTU1\_TCIV1*].BIT.CLR\_MTU1\_TCIV1)

{

ICU.GCR[*GCR\_MTU1\_TCIV1*].BIT.CLR\_MTU1\_TCIV1 = 1;

dummy = ICU.GCR[*GCR\_MTU1\_TCIV1*].BIT.CLR\_MTU1\_TCIV1;

}

par\_encoder\_1.Under\_over\_flow\_cnt++;

}

**if**(ICU.GRP[*GRP\_MTU1\_TCIU1*].BIT.IS\_MTU1\_TCIU1)

{

/\* ---- Clearing of TCIU Underflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_MTU1\_TCIU1*].BIT.CLR\_MTU1\_TCIU1)

{

ICU.GCR[*GCR\_MTU1\_TCIU1*].BIT.CLR\_MTU1\_TCIU1 = 1;

dummy = ICU.GCR[*GCR\_MTU1\_TCIU1*].BIT.CLR\_MTU1\_TCIU1;

}

par\_encoder\_1.Under\_over\_flow\_cnt--;

}

}**while**(ICU.IR[VECT\_ICU\_GROUPE1].BIT.IR == 1);

}

/\*

\* QUESTA FUNZIONE DI INTERRUPT PER ORA NON LA USO:

\* SERVE PER DARE LA VELOCITA' ISTANTANEA DELL'ALBERO;

\* PER LA VELOCITA' PRELEVO INVECE LA CATTURA SU TGRB della MTU1

\*/

**#pragma** interrupt MTU0\_TGIB0\_isr(vect = VECT\_MTU0\_TGIB0, enable)

**static** **void** **MTU0\_TGIB0\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**signed** **long** data;

/\* ---- Clearing of TGIB interrupt flag ---- \*/

**if**(ICU.IR[VECT\_MTU0\_TGIB0].BIT.IR)

{

ICU.IR[VECT\_MTU0\_TGIB0].BIT.IR = 0;

dummy = ICU.IR[VECT\_MTU0\_TGIB0].BIT.IR;

}

data = MTU0.TGRB - MTU0.TGRD;

/\* Read TGRB capture register \*/

/\* Read TGRD buffer register \*/

**if** ( data < 0){

par\_encoder\_1.TGRD0\_B0\_data\_diff = data + (CH0\_TGRC\_CYCLE+1); /\* set data \*/

}**else**{

par\_encoder\_1.TGRD0\_B0\_data\_diff = data; /\* set data \*/

}

}

**ENCODER 2**

/\* Implementazione degli interrupt per la gestione della lettura dell'encoder 2 \*/

**#pragma** interrupt TPU1\_TGIA1\_isr(vect = VECT\_TPU1\_TGI1A, enable)

**static** **void** **TPU1\_TGIA1\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**if**(TPU1.TSR.BIT.TGFA)

{

TPU1.TSR.BIT.TGFA = 0;

dummy = TPU1.TSR.BIT.TGFA;

}

**if**(ICU.IR[VECT\_TPU1\_TGI1A].BIT.IR)

{

ICU.IR[VECT\_TPU1\_TGI1A].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU1\_TGI1A].BIT.IR;

}

par\_encoder\_2.TGRA1\_data = TPU1.TGRA; /\* Read TGRA capture register \*/

par\_encoder\_2.Contatore\_di\_overflow = par\_encoder\_2.Under\_over\_flow\_cnt;

par\_encoder\_2.posizione\_acquisita = 1;

/\* Disable interrupt TGIA1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

TPU1.TIER.BYTE &= 0xFE;

/\* 7 : TTGE=b'0 : A/D Converter Start Request Disabled \*/

/\* 6 : =b'0 : reserve \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'1 : TGRB Interrupt Enable \*/

/\* 0 :TGIEA=b'0 : TGRA Interrupt Disable \*/

}

**#pragma** interrupt TPU1\_TGIB1\_isr(vect = VECT\_TPU1\_TGI1B, enable)

**static** **void** **TPU1\_TGIB1\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**static** **unsigned** **char** state = 0;

**if**(TPU1.TSR.BIT.TGFB)

{

TPU1.TSR.BIT.TGFB = 0;

dummy = TPU1.TSR.BIT.TGFB;

}

**if**(ICU.IR[VECT\_TPU1\_TGI1B].BIT.IR)

{

ICU.IR[VECT\_TPU1\_TGI1B].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU1\_TGI1B].BIT.IR;

}

**switch**(state)

{

**case** 0:

par\_encoder\_2.TGRB1\_data\_old = TPU1.TGRB; /\* Read TGRB capture register \*/

state = 1;

par\_encoder\_2.Read\_overflow\_old = (**int**)par\_encoder\_2.Under\_over\_flow\_cnt;

par\_encoder\_2.speed\_sample = 0;

**break**;

**case** 1:

par\_encoder\_2.speed\_sample = 1;

state = 0;

par\_encoder\_2.TGRB1\_data\_new = TPU1.TGRB; /\* Read TGRB capture register \*/

par\_encoder\_2.Read\_overflow = (**int**)par\_encoder\_2.Under\_over\_flow\_cnt;

/\* Disable interrupt TGIB1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

TPU1.TIER.BYTE &= 0xFD;

/\* 7 : \*/

/\* 6 : \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'0 : TGRB Interrupt Disable \*/

/\* 0 :TGIEA=b' \*/

**break**;

**default**:

state = 0;

par\_encoder\_2.speed\_sample = 0;

}

}

**#pragma** interrupt TPU1\_TCIV\_TCUV\_isr(vect = VECT\_ICU\_GROUPE3, enable)

**void** **TPU1\_TCIV\_TCUV\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**do** {

//ICU.GRP[ GRP ## x ].BIT.IS ## x

//GRP\_TPU0\_TCI0V=1,GRP\_TPU1\_TCI1V=1,GRP\_TPU1\_TCI1U=1,

**if**(ICU.GRP[*GRP\_TPU1\_TCI1V*].BIT.IS\_TPU1\_TCI1V)

{

/\* ---- Clearing of TCIV Overflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_TPU1\_TCI1V*].BIT.CLR\_TPU1\_TCI1V)

{

ICU.GCR[*GCR\_TPU1\_TCI1V*].BIT.CLR\_TPU1\_TCI1V = 1;

dummy = ICU.GCR[*GCR\_TPU1\_TCI1V*].BIT.CLR\_TPU1\_TCI1V;

}

**if**(TPU1.TSR.BIT.TCFV)

{

TPU1.TSR.BIT.TCFV = 0;

dummy = TPU1.TSR.BIT.TCFV;

}

par\_encoder\_2.Under\_over\_flow\_cnt++;

}

**if**(ICU.GRP[*GRP\_TPU1\_TCI1U*].BIT.IS\_TPU1\_TCI1U)

{

/\* ---- Clearing of TCIU Underflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_TPU1\_TCI1U*].BIT.CLR\_TPU1\_TCI1U)

{

ICU.GCR[*GCR\_TPU1\_TCI1U*].BIT.CLR\_TPU1\_TCI1U = 1;

dummy = ICU.GCR[*GCR\_TPU1\_TCI1U*].BIT.CLR\_TPU1\_TCI1U;

}

**if**(TPU1.TSR.BIT.TCFU)

{

TPU1.TSR.BIT.TCFU = 0;

dummy = TPU1.TSR.BIT.TCFU;

}

par\_encoder\_2.Under\_over\_flow\_cnt--;

}

}**while**(ICU.IR[VECT\_ICU\_GROUPE3].BIT.IR == 1);

}

/\*

\* QUESTA FUNZIONE DI INTERRUPT PER ORA NON LA USO:

\* SERVE PER DARE LA VELOCITA' ISTANTANEA DELL'ALBERO;

\* PER LA VELOCITA' PRELEVO INVECE LA CATTURA SU TGRB della TPU1

\* \*/

**#pragma** interrupt TPU0\_TGIB0\_isr(vect = VECT\_TPU0\_TGI0B, enable)

**static** **void** **TPU0\_TGIB0\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**signed** **long** data;

**if**(TPU0.TSR.BIT.TGFB)

{

TPU0.TSR.BIT.TGFB = 0;

dummy = TPU0.TSR.BIT.TGFB;

}

/\* ---- Clearing of TGIB interrupt flag ---- \*/

**if**(ICU.IR[VECT\_TPU0\_TGI0B].BIT.IR)

{

ICU.IR[VECT\_TPU0\_TGI0B].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU0\_TGI0B].BIT.IR;

}

data = TPU0.TGRB - TPU0.TGRD;

/\* Read TGRB capture register \*/

/\* Read TGRD buffer register \*/

**if** ( data < 0){

par\_encoder\_2.TGRD0\_B0\_data\_diff = data + (CH0\_TGRC\_CYCLE+1); /\* set data \*/

}**else**{

par\_encoder\_2.TGRD0\_B0\_data\_diff = data; /\* set data \*/

}

}

/\* End of File \*/

**ENCODER 3**

/\* Implementazione degli interrupt per la gestione della lettura dell'encoder 3 \*/

**#pragma** interrupt TPU4\_TGIA4\_isr(vect = VECT\_TPU4\_TGI4A, enable)

**static** **void** **TPU4\_TGIA4\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**if**(TPU4.TSR.BIT.TGFA)

{

TPU4.TSR.BIT.TGFA = 0;

dummy = TPU4.TSR.BIT.TGFA;

}

**if**(ICU.IR[VECT\_TPU4\_TGI4A].BIT.IR)

{

ICU.IR[VECT\_TPU4\_TGI4A].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU4\_TGI4A].BIT.IR;

}

par\_encoder\_3.TGRA1\_data = TPU4.TGRA; /\* Read TGRA capture register \*/

par\_encoder\_3.Contatore\_di\_overflow = par\_encoder\_3.Under\_over\_flow\_cnt;

par\_encoder\_3.posizione\_acquisita = 1;

/\* Disable interrupt TGIA1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

TPU4.TIER.BYTE &= 0xFE;

/\* 7 : TTGE=b'0 : A/D Converter Start Request Disabled \*/

/\* 6 : =b'0 : reserve \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'1 : TGRB Interrupt Enable \*/

/\* 0 :TGIEA=b'0 : TGRA Interrupt Disable \*/

}

/\* INTERRUPT PER IL CALCOLO DELLA VELOCITA' \*/

**#pragma** interrupt TPU4\_TGIB4\_isr(vect = VECT\_TPU4\_TGI4B, enable)

**static** **void** **TPU4\_TGIB4\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**static** **unsigned** **char** state = 0;

**if**(TPU4.TSR.BIT.TGFB)

{

TPU4.TSR.BIT.TGFB = 0;

dummy = TPU4.TSR.BIT.TGFB;

}

**if**(ICU.IR[VECT\_TPU4\_TGI4B].BIT.IR)

{

ICU.IR[VECT\_TPU4\_TGI4B].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU4\_TGI4B].BIT.IR;

}

**switch**(state)

{

**case** 0:

par\_encoder\_3.TGRB1\_data\_old = TPU4.TGRB; /\* Read TGRB capture register \*/

state = 1;

par\_encoder\_3.Read\_overflow\_old = (**int**)par\_encoder\_3.Under\_over\_flow\_cnt;

par\_encoder\_3.speed\_sample = 0;

**break**;

**case** 1:

par\_encoder\_3.speed\_sample = 1;

state = 0;

par\_encoder\_3.TGRB1\_data\_new = TPU4.TGRB; /\* Read TGRB capture register \*/

par\_encoder\_3.Read\_overflow = (**int**)par\_encoder\_3.Under\_over\_flow\_cnt;

/\* Disable interrupt TGIB1 \*/

/\* ---- Timer interrupt enable register\_1 (TIER\_1) ---- \*/

TPU4.TIER.BYTE &= 0xFD;

/\* 7 : \*/

/\* 6 : \*/

/\* 5 :TCIEU=b'1 : Underflow Interrupt Enable \*/

/\* 4 :TCIEV=b'1 : Overflow Interrupt Enable \*/

/\* 3-2: =b'00: reserve \*/

/\* 1 :TGIEB=b'0 : TGRB Interrupt Disable \*/

/\* 0 :TGIEA=b' \*/

**break**;

**default**:

state = 0;

par\_encoder\_3.speed\_sample = 0;

}

}

**#pragma** interrupt TPU4\_TCIV\_TCUV\_isr(vect = VECT\_ICU\_GROUPE4, enable)

**void** **TPU4\_TCIV\_TCUV\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**do** {

//ICU.GRP[ GRP ## x ].BIT.IS ## x

//GRP\_TPU0\_TCI0V=1,GRP\_TPU1\_TCI1V=1,GRP\_TPU1\_TCI1U=1,

**if**(ICU.GRP[*GRP\_TPU4\_TCI4V*].BIT.IS\_TPU4\_TCI4V)

{

/\* ---- Clearing of TCIV Overflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_TPU4\_TCI4V*].BIT.CLR\_TPU4\_TCI4V)

{

ICU.GCR[*GCR\_TPU4\_TCI4V*].BIT.CLR\_TPU4\_TCI4V = 1;

dummy = ICU.GCR[*GCR\_TPU4\_TCI4V*].BIT.CLR\_TPU4\_TCI4V;

}

**if**(TPU4.TSR.BIT.TCFV)

{

TPU4.TSR.BIT.TCFV = 0;

dummy = TPU4.TSR.BIT.TCFV;

}

par\_encoder\_3.Under\_over\_flow\_cnt++;

}

**if**(ICU.GRP[*GRP\_TPU4\_TCI4U*].BIT.IS\_TPU4\_TCI4U)

{

/\* ---- Clearing of TCIU Underflow Flag ---- \*/

**if**(!ICU.GCR[*GCR\_TPU4\_TCI4U*].BIT.CLR\_TPU4\_TCI4U)

{

ICU.GCR[*GCR\_TPU4\_TCI4U*].BIT.CLR\_TPU4\_TCI4U = 1;

dummy = ICU.GCR[*GCR\_TPU4\_TCI4U*].BIT.CLR\_TPU4\_TCI4U;

}

**if**(TPU4.TSR.BIT.TCFU)

{

TPU4.TSR.BIT.TCFU = 0;

dummy = TPU4.TSR.BIT.TCFU;

}

par\_encoder\_3.Under\_over\_flow\_cnt--;

}

}**while**(ICU.IR[VECT\_ICU\_GROUPE4].BIT.IR == 1);

}

/\*

\* QUESTA FUNZIONE DI INTERRUPT PER ORA NON LA USO:

\* SERVE PER DARE LA VELOCITA' ISTANTANEA DELL'ALBERO;

\* PER LA VELOCITA' PRELEVO INVECE LA CATTURA SU TGRB della TPU1

\* \*/

**#pragma** interrupt TPU3\_TGIB3\_isr(vect = VECT\_TPU3\_TGI3B, enable)

**static** **void** **TPU3\_TGIB3\_isr**(**void**)

{

**volatile** **unsigned** **char** dummy;

**signed** **long** data;

**if**(TPU3.TSR.BIT.TGFB)

{

TPU3.TSR.BIT.TGFB = 0;

dummy = TPU3.TSR.BIT.TGFB;

}

/\* ---- Clearing of TGIB interrupt flag ---- \*/

**if**(ICU.IR[VECT\_TPU3\_TGI3B].BIT.IR)

{

ICU.IR[VECT\_TPU3\_TGI3B].BIT.IR = 0;

dummy = ICU.IR[VECT\_TPU3\_TGI3B].BIT.IR;

}

data = TPU3.TGRB - TPU3.TGRD;

/\* Read TGRB capture register \*/

/\* Read TGRD buffer register \*/

**if** ( data < 0){

par\_encoder\_3.TGRD0\_B0\_data\_diff = data + (CH0\_TGRC\_CYCLE+1); /\* set data \*/

}**else**{

par\_encoder\_3.TGRD0\_B0\_data\_diff = data; /\* set data \*/

}

}