Thermidor
Technologies

Germinimal – Keyboard Controller

Firmware

INDICE

1. INTRODUZIONE

Il controllo del sintetizzatore è realizzato dalla scheda KBD-CONTR. Il circuito realizza un convertitore MIDI-CV, l' implementazione è basata su un microcontrollore PIC16F76.
Il microcontrollore provvede a convertire i messaggi MIDI in tensione, mediante l' utilizzo dei due generatori PWM (uno per la tensione di controllo e uno per il Key-Velocity).
La conversione in tensione continua è realizzata tramite due filtri passa basso a quattro poli.
Il firmware, inoltre gestisce gli eventi dei due generatori di inviluppi, presenti su altra scheda.

2. SPECIFICHE

Di seguito le specifiche del firmware:

1) Ingresso MIDI standard.
2) Conversione in tensione con risoluzione 10 bit.
3) Frequenza di clock 20MHz.
4) Non viene riconosciuto il canale MIDI.
5) I messaggi di nota riconosciuti sono compresi tra C1 (36) e C5 (84).
6) Pitch-Bend di ± 1 tono.
7) Vengono considerati solo i 6 bit più alti del valore MSB, relativo al messaggio MIDI di Pitch-Bend.
8) La funzione di Key Velocity, al momento è lineare, ci si aspetta che la variazione logaritmica della scala sia generata dalla tastiera.
9) Implementazione del Running Status.
10) Nessuna interferenza con i messaggi di System Common.
11) Gestione dei due generatori di inviluppi.
12) Sviluppo del firmware in Pascal.

3. IL PROTOCOLLO MIDI

Al fine di rendere più semplice la spiegazione del firmware, viene fornita una breve spiegazione del protocollo, almeno per i punti salienti.

3.1. Struttura del protocollo MIDI

Da un punto di vista strettamente hardware, il MIDI non è altro che una trasmissione seriale a 31,25kBaud, il segnale è trasmesso come loop di corrente a 5mA ed è isolato galvanicamente.
Ogni singolo Byte è trasmesso con un bit di start e un bit di stop.

Visto come software, il protocollo è diviso in cinque categorie principali:

1) CHANNEL VOICE MESSAGES
2) CHANNEL MODE MESSAGES
3) SYSTEM COMMON MESSAGES
4) REAL TIME MESSAGES
5) SYSTEM EXCLUSIVE

Tutti i messaggi sono suddivisi in pacchetti di 1, 2 o 3 byte, nel quale il primo identifica il comando (o Status es: NOTE-ON, PITCH-BEND, PROGRAM-CHANGE), mentre i successivi il valore da usare.

3.1.1. CHANNEL VOICE MESSAGES

Questo gruppo comprende i comandi che riguardano l' esecuzione come ad esempio i messaggi di NOTA-ON, NOTA-OFF, PROGRAM-CHANGE, CONTROL-CHANGE. La tabella di seguito elenca i tipi di messaggi disponibili:

CHANNEL VOICE MESSAGES

STATUS	DATA BYTES	DESCRIPTION
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off
	vvvvvvv:	note off velocity
1001nnnn	0kkkkkkk 0vvvvvvv	Note On
	vvvvvvv > 0:	velocity,
	vvvvvvv = 0:	note off
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (After-Touch)
	vvvvvvv:	pressure value
1011nnnn	0ccccccc 0vvvvvvv	Control Change
	ccccccc:	control # (0-121)
	vvvvvvv:	control value
	ccccccc = 122 thru 127:	Reserved
1100nnnn	0ppppppp	Program Change,
	ppppppp:	program number (0-127)
1101nnnn	0vvvvvvv	Channel Pressure (After-Touch)
	vvvvvvv:	pressure value
1110nnnn	0vvvvvvv	Pitch Bend Change LSB
	0vvvvvvv	Pitch Bend Change MSB

NOTES:

nnnn: Voice Channel # (1-16, coded)
kkkkkkk: note # (0 - 127)
kkkkkkk = 60: Middle C of keyboard
vvvvvvv: key velocity

3.1.2. CHANNEL MODE MESSAGES

Questo gruppo riguarda il modo di funzionamento del dispositivo, tipo MONO o POLY:

CHANNEL MODE MESSAGES

STATUS	DATA BYTES	DESCRIPTION
1011nnnn	0ccccccc 0vvvvvvv	Mode Messages
-	ccccccc = 122:	Local Control
	vvvvvvv = 0,	Local Control Off
	vvvvvvv = 127,	Local Control On
-	ccccccc = 123:	All Notes Off
	vvvvvvv = 0	
-	ccccccc = 124:	Omni Mode Off (All Notes Off)
	vvvvvvv = 0	
-	ccccccc = 125:	Omni Mode On (All Notes Off)
	vvvvvvv = 0	
-	ccccccc = 126:	Mono Mode On (Poly Mode Off)
		(All Notes Off)
	vvvvvvv = M,	where M is the number of channels.
	vvvvvvv = 0,	the number of channels equals
		the number of voices in the receiver.
-	ccccccc = 127:	Poly Mode On (Mono Mode Off)
	vvvvvvv = 0	(All Notes Off)

3.1.3. SYSTEM COMMON MESSAGES

In questo gruppo sono contenuti i messaggi tipo il TUNE REQUEST, SONG POINTER

SYSTEM COMMON MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11110001		Undefined
11110010	0lllllll 0hhhhhhh	Song Position Pointer
		lllllll: (Least significant)
		hhhhhhh: (Most significant)
11110011	0sssssss	Song Select
		sssssss: Song #
11110100		Undefined
11110101		Undefined
11110110	none	Tune Request
11110111	none	EOX: "End of System Exclusive" flag

3.1.4. SYSTEM REAL TIME MESSAGES

Questo gruppo contiene i messaggi per sincronizzare il sistema utilizzato, che può essere composto da vari dispositivi: sintetizzatori, batterie elettroniche, sequencer.

REAL TIME MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11111000		Timing Clock
11111001		Undefined
11111010		Start
11111011		Continue
11111100		Stop
11111101		Undefined
11111110		Active Sensing
11111111		System Reset

3.1.5. SYSTEM EXCLUSIVE

Questo gruppo fornisce la possibilità ai costruttori di inviare messaggi personalizzati, in genere viene usato per il trasferimento di dati, ma non solo.

REAL TIME MESSAGES

STATUS	DATA BYTES	DESCRIPTION
11110000		Bulk dump etc.
	0iiiiiii	iiiiiii: identification
	.	
	(0*******)	
	.	Any number of bytes may be sent here,
	.	for any purpose, as long as they all
	(0*******)	have a zero in the most significant bit.
	.	
	11110111	EOX: "End of System Exclusive"

3.2. Running Status

Per i messaggi relativi ai gruppi VOICE e MODE, è previsto il funzionamento in Running Status.
Come si può notare questo modo di funzionamento coinvolge i messaggi riguardante l' esecuzione vera e propria, in pratica i messaggi di NOTA, PITCH-BEND, PROGRAM-CHANGE, AFTERTOUCH e i CONTROL-CHANGE in genere.
Attraverso l' utilizzo di questa modalità è possibile risparmiare la trasmissione di un Byte per ogni pacchetto inviato, e di velocizzare quindi il sistema.

Il funzionamento è piuttosto semplice:

- Il trasmettitore invia il primo Byte (Status)
- Il ricevitore lo riconosce e si predispone a ricevere gli altri messaggi riguardanti quello Status
- Il trasmettitore invia i Byte successivi (ad esempio Valore della nota e Velocità)
- Il ricevitore decodifica i messaggi e comanda il dispositivo, al termine rimane in quello Status
- Il trasmettitore, se deve inviare altri messaggi riguardanti quello Status, omette il Byte di Status e invia solo i codici successivi
- Il ricevitore riconosce i messaggi successivi come appartenenti allo Status corrente e li interpreta
- Se il trasmettitore deve cambiare tipo di messaggio (ad esempio passare da NOTE-ON a PITCH-BEND), invia il nuovo Byte di Status
- Il ricevitore riconosce il nuovo Status, esce da quello precedente e si predispone a ricevere i messaggi inerenti il nuovo
- La transazione prosegue fino all' invio di un nuovo Status

Questo modo di comunicare, per messaggi molto corposi di dati come il PITCH-BEND o l' EXPRESSION, consente di ridurre di un terzo il numero di Byte coinvolti, aumentando la velocità del sistema.

Rimangono esclusi da questa modalità gli altri messaggi, che mantengono però una priorità maggiore. Ad esempio un codice di sincronismo si può intercalare in un messaggio di Running Status, il ricevitore lo deve eseguire con priorità o ignorarlo (a seconda del dispositivo), e ritornare allo Status corrente.

Nella figura di seguito è illustrata la trasmissione di un messaggio di NOTE-ON:

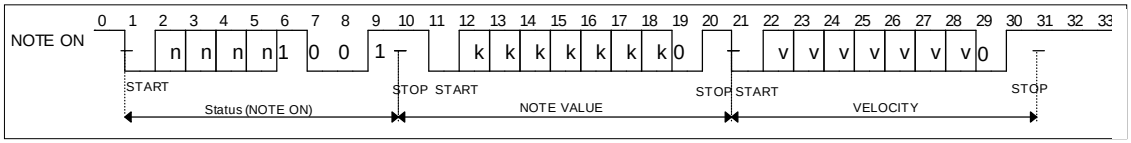


Figura 1

Si possono vedere in sequenza transitare lo Status, il Key code e il Key Velocity.

La stessa procedura utilizzando il Running Status per due messaggi di NOTE-ON, è illustrata nella figura di seguito:

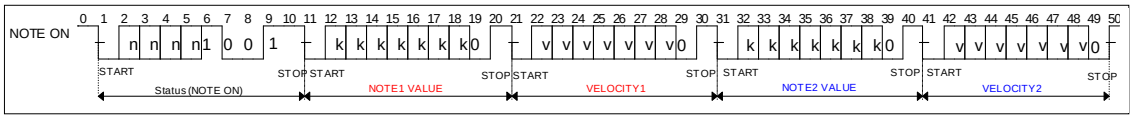


Figura 2

Come si nota, il secondo messaggio non utilizza più il Byte di Status, in quanto il codice inviato appartiene allo stesso Status del precedente.

Nella figura successiva è possibile vedere come un messaggio di REAL-TIME, possa inserirsi all' interno di un messaggio di Running Status:

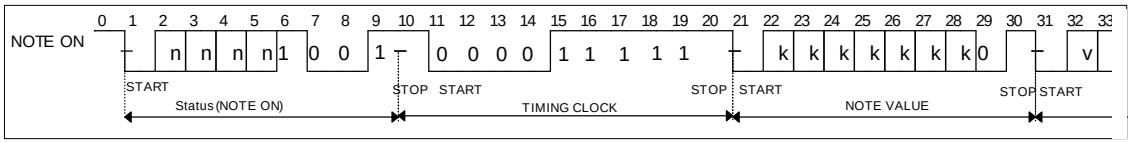


Figura 3

Nel caso esposto, un messaggio di TIMING-CLOCK viene inserito nel pacchetto di NOTE-ON, il sistema ricevente ha due possibilità:

1) Processa l' evento e poi ritorna al Running Status corrente.
2) Lo ignora.

Il tipo di scelta dipende ovviamente dal dispositivo, una batteria elettronica lo processerà ed un sintetizzatore che sta utilizzando sequenze pure, un sintetizzatore che lavora come Lead lo ignorerà.

4. IL FIRMWARE

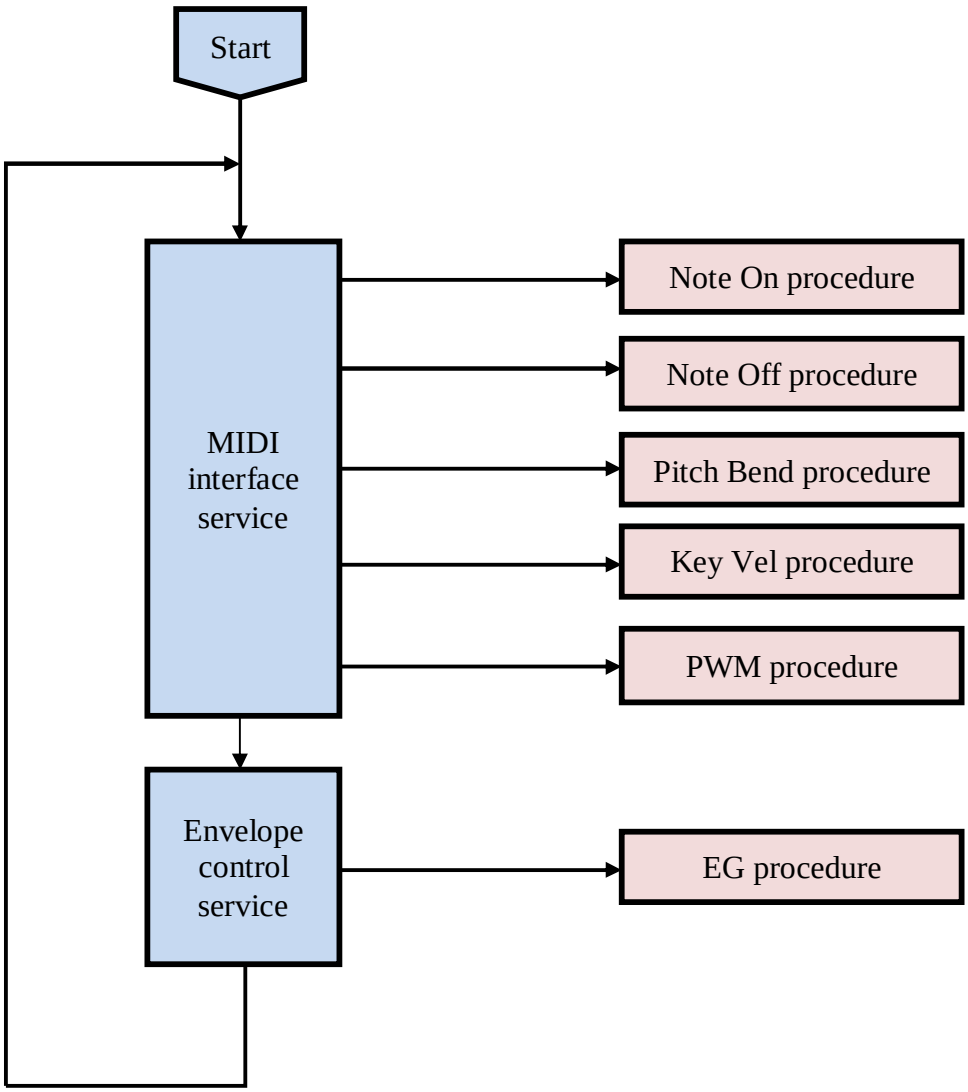
Il convertitore MIDI-CV è stato sviluppato con MikroPascal demo della MikroElektronika, la versione dimostrativa compila solo fino a 2kByte di codice e **non può essere usata per scopi commerciali.**

4.1. Struttura del programma

Il programma è costituito da un corpo principale (Main) che svolge due funzioni:

1) Interfaccia verso l' ingresso MIDI.
2) Gestione dei generatori di inviluppo.

E da una serie di procedure di comando.
La struttura può essere semplificata come nella figura di seguito:



4.2. Principio di funzionamento

Per generare la tensione di controllo (CV), viene utilizzato un generatore PWM, al quale viene fatto variare il duty-cycle.
Un successivo filtraggio del segnale, restituirà il valore medio dell' onda quadra, variandone il duty-cycle da 0% al 100% è possibile ottenere una tensione continua da 0 a 5Vdc.
La risoluzione del generatore PWM è di 10bit (4096 valori).
Partendo dal valore intermedio del convertitore (2048), si avrà che il raddoppio di tensione si raggiungerà a 4096, ora il valore massimo che si può inviare è 4095 ma questo non è un problema in quanto il salto di ottava lo si ottiene per divisione della tensione generata.
Per velocizzare il sistema, i valori da inviare al PWM, sono all' interno di una tabella (look-up table), partendo dalla prima nota (il DO), e sono spaziati tra di loro in intervalli di 1/32 di semitono.
Questo consente di ottenere una variazione, determinata dal comando di Pitch-Bend, fino a 1 tono, in intervalli appunto di 1/32 di tono.

Quindi:

$$DO = 2048$$

Incremento:

$$\frac{1}{32} = 0{,}03125$$

Lo step successivo:

$$2048 \cdot 2^{\frac{0{,}03125}{12}} = 2051{,}7 \; approssimato \; 2052$$

Così di seguito fino a riempire tutta la tabella che sarà costituita da:

$$12 \; note \cdot 32 \; step = 384 \; valori$$

Che andranno dal DO al SI più quasi un semitono. Occorre ora fare in modo che il valore della nota da processare più o meno il Pitch-Bend, formi un valore che punti alla tabella contenente il valore da inviare al PWM.

4.2.1. Costruzione del valore del puntatore

La figura di seguito mostra uno schema semplificato di come viene generata la tensione di uscita:

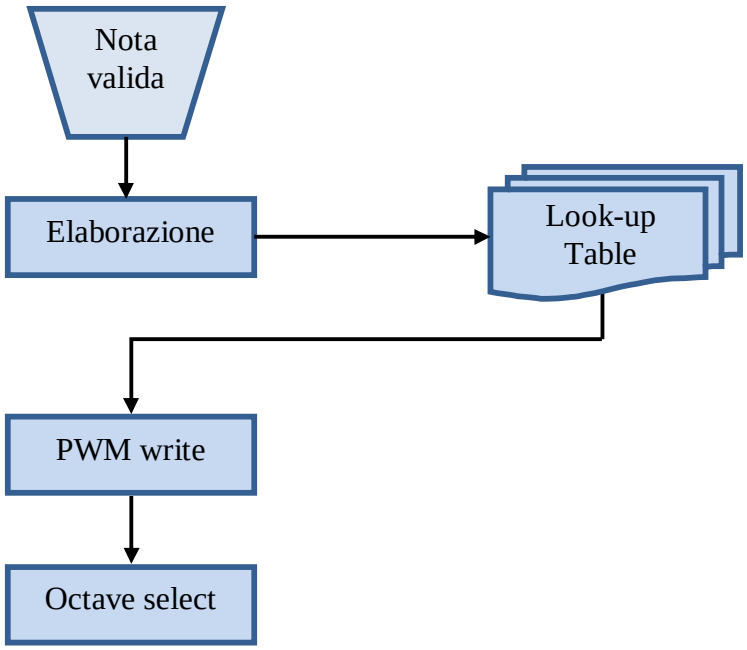


Figura 5

Nel dettaglio tutta l' operazione viene eseguita chiamando la Procedure "write_note", la quale a sua volta chiama le Procedure "assemble_cv" e "find_note", a questo punto è in grado di impostare il PWM con il valore corretto.

4.2.1.1. Procedure assemble_cv

Questa Procedure esegue le seguenti istruzioni:

1. riceve il dato in forma binaria compreso tra 36 e 84, nella forma:

$$0nnnnnnn$$

2. il dato da byte viene trasformato in word nel formato:

$$0000nnnnn00000$$

3. a questo valore viene sommato (o sottratto) il valore corrente di Pitch-Bend (scalato a 6 bit):

$$\begin{matrix} 000000nnnnn00000 \\ 0000000000ppppppp \end{matrix} \pm$$

4.2.1.2. Procedure find_octave

A questa Procedure viene passato il dato calcolato in precedenza nella forma:

$$000000nnnnnppppp$$

Viene isolato il valore delle nota:

$$Nota = 000nnnn$$

E il valore finale viene ricavato:

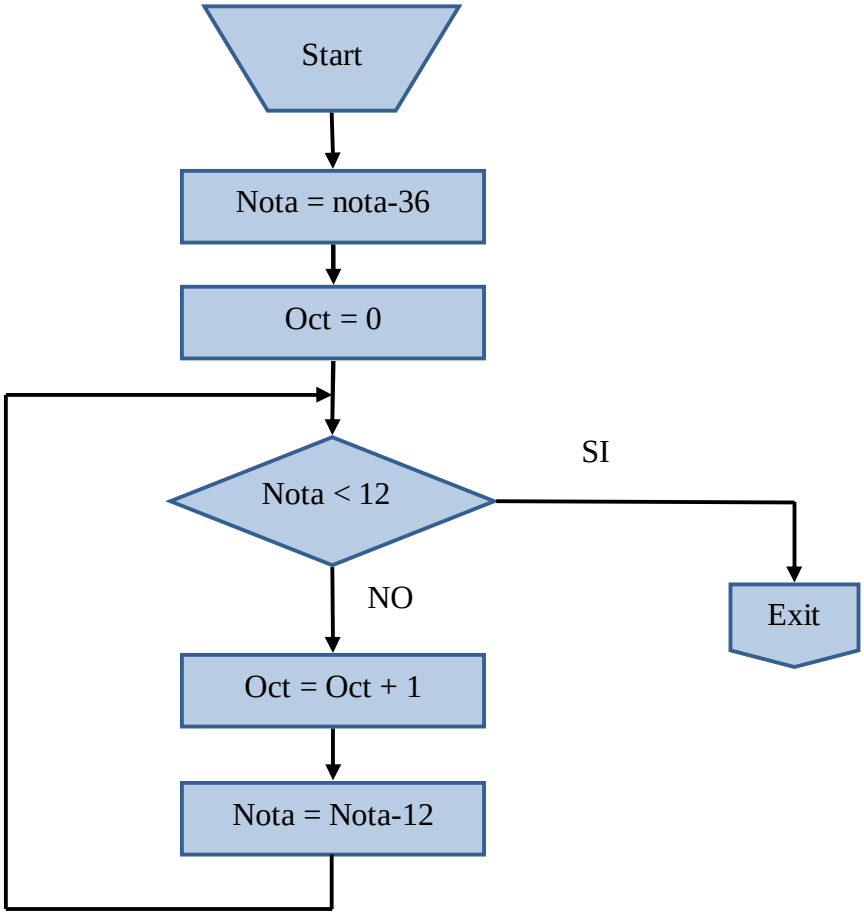


Figura 6

A questo punto viene aggiornata la variabile table_pointer con il valore di nota calcolato più il pitch-bend, nel formato:

$$0000000NNNNppppp$$

Dove NNNN è un valore tra 0 e 11.

4.2.1.3. Procedure write_note

E’ stata messa per ultima ma in realtà è la prima ed è quella che chiama le due precedenti.
A questo punto la Procedure si trova con tutti i dati giusti:

- La variabile oct contiene il valore della ottava da inviare al selettore.
- La variabile table_pointer contiene l’ indirizzo a cui trovare il valore da scrivere nel PWM.

4.2.2. Key Velocity

La gestione del Key Velocity è affidata alla Procedure write_vel.
Viene chiamata sia in caso di note-on che di note-off.
Questo perché la Procedure tiene conto del numero di note premute:

- Ad ogni key velocity > 0 viene incrementato il numero di tasti premuto.
- Ad ogni key velocity = 0 o ad ogni note-off, viene decrementato il numero di tasti premuto.
- Quando il numero di tasti premuto è uguale a 0, viene attivata l’ uscita RELEASE (il GATE).
- Se il numero di tasti premuto è >=1 viene disattivata l’ uscita RELEASE (il GATE).

Attualmente il valore attuale di Key Velocity viene inviato al PWM corrispondente come è, questo significa che la risposta è lineare o, per meglio dire occorre che il dispositivo di controllo abbia la possibilità di inviare una Key Velocity con andamento esponenziale.

4.2.3. Il Main

Il Main è un loop infinito che si può schematizzare come illustrato nella figura di seguito:

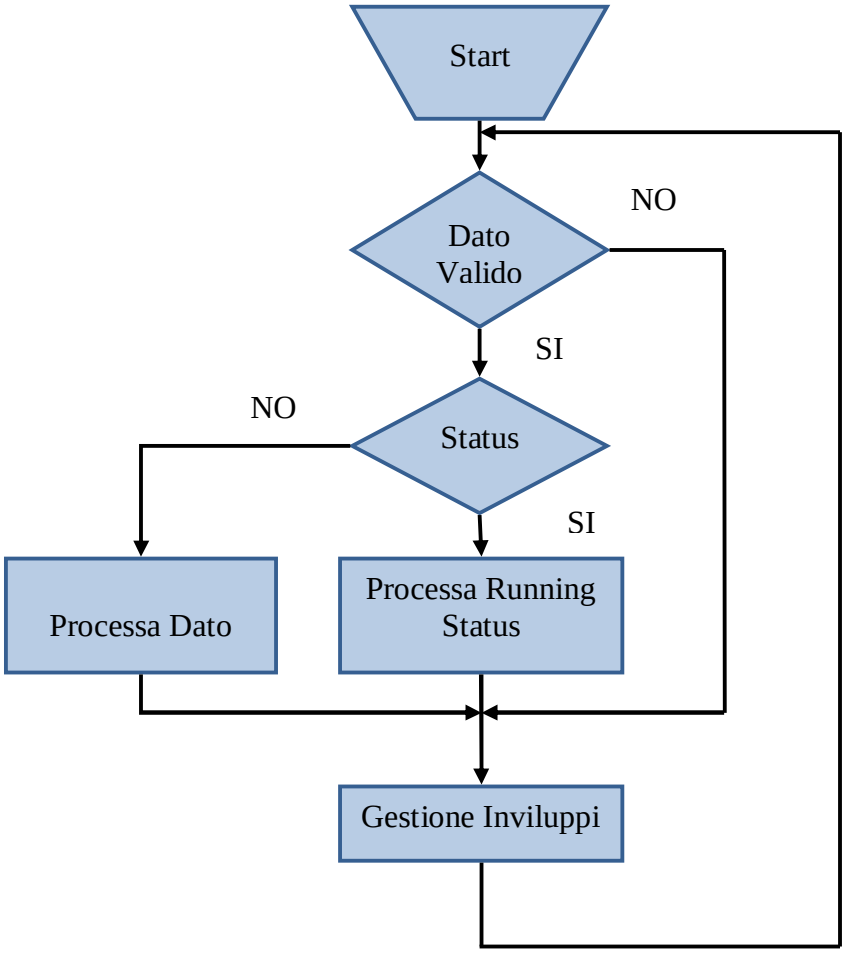


Figura 7

Occorre precisare che alla ricezione di un byte di Status, viene mascherato il canale MIDI, questo perché ormai con tutti i dispositivi di moltiplicazione di porte MIDI a basso costo, non vale più la pena di stare a identificare il canale.
Inoltre il dato valido di nota deve essere compreso tra C1 (midi 36) e C5 (midi 84), tutti gli altri valori sono ignorati.
Per completezza occorre aggiungere che i dati nella look-up table non vanno da 2048 a 4095, ma partono da 32768, questo per facilitare il calcolo del valore finale da inviare al PWM.

4.3. Listato

Di seguito il listato del Firmware, suddiviso nei blocchi principali.

4.3.1. Le dichiarazioni iniziali, la look-up table, e i settaggi delle variabili

```
var
     note_count : byte; absolute $20;
     get_midi : byte;  absolute $21;
     note_on_flag : byte; absolute $22;
     run_stat : byte; absolute $23;
     overrange : byte; absolute $24;
     current_note : byte; absolute $25;
     current_pitch : byte; absolute $26;
     pwm_cv : word; absolute $27;
     pwm_kvel : byte; absolute $29;
     temp_word : word; absolute $2A;
     temp_calc : byte; absolute $2C;
     table_pointer : word; absolute $2D;
     octave : byte; absolute $2F;
     save_last_note : byte; absolute $30;
     attack1 : byte;
     attack2 : byte;

//Tabella dei valori da inserire nel PWM
const table_note : array[0..383] of word = (
32768,32827,32887,32946,33005,33065,33125,33185,33245,33305,33365,33425,33486,33546,33607,33667,33728
,33789,33850,33911,33973,34034,34095,34157,34219,34281,34343,34405,34467,34529,34591,34654,
34716,34779,34842,34905,34968,35031,35095,35158,35221,35285,35349,35413,35477,35541,35605,35669,35734
,35798,35863,35928,35993,36058,36123,36188,36254,36319,36385,36450,36516,36582,36648,36715,
36781,36847,36914,36981,37047,37114,37181,37249,37316,37383,37451,37518,37586,37654,37722,37790,37859
,37927,37996,38064,38133,38202,38271,38340,38409,38479,38548,38618,38688,38757,38828,38898,
38968,39038,39109,39180,39250,39321,39392,39463,39535,39606,39678,39749,39821,39893,39965,40037,40110
,40182,40255,40328,40400,40473,40547,40620,40693,40767,40840,40914,40988,41062,41136,41211,
41285,41360,41434,41509,41584,41659,41735,41810,41886,41961,42037,42113,42189,42265,42342,42418,42495
,42572,42649,42726,42803,42880,42958,43035,43113,43191,43269,43347,43425,43504,43582,43661,
43740,43819,43898,43978,44057,44137,44216,44296,44376,44456,44537,44617,44698,44779,44859,44941,45022
,45103,45185,45266,45348,45430,45512,45594,45677,45759,45842,45925,46008,46091,46174,46257,
46341,46425,46509,46593,46677,46761,46846,46930,47015,47100,47185,47270,47356,47441,47527,47613,47699
,47785,47871,47958,48044,48131,48218,48305,48393,48480,48568,48655,48743,48831,48920,49008,
49097,49185,49274,49363,49452,49542,49631,49721,49811,49901,49991,50081,50172,50262,50353,50444,50535
,50626,50718,50810,50901,50993,51085,51178,51270,51363,51456,51549,51642,51735,51829,51922,
52016,52110,52204,52298,52393,52488,52582,52677,52773,52868,52963,53059,53155,53251,53347,53444,53540
,53637,53734,53831,53928,54026,54123,54221,54319,54417,54515,54614,54713,54811,54910,55010,
55109,55209,55308,55408,55508,55609,55709,55810,55911,56012,56113,56214,56316,56417,56519,56622,56724
,56826,56929,57032,57135,57238,57341,57445,57549,57653,57757,57861,57966,58071,58176,58281,
58386,58491,58597,58703,58809,58915,59022,59128,59235,59342,59449,59557,59664,59772,59880,59988,60097
,60205,60314,60423,60532,60642,60751,60861,60971,61081,61191,61302,61413,61524,61635,61746,
61858,61970,62081,62194,62306,62419,62531,62644,62757,62871,62984,63098,63212,63326,63441,63555,63670
,63785,63901,64016,64132,64248,64364,64480,64596,64713,64830,64947,65065,65182,65300,65418
);org 0x1000;


//Definizione dei pin di I/O
var  EG1_IN : sbit at PORTA.B3;
     EG2_IN : sbit at PORTA.B4;
     OCT_1 : sbit at PORTB.B0;
     OCT_2 : sbit at PORTB.B1;
     OCT_3 : sbit at PORTB.B2;
     EG1 : sbit at PORTB.B3;
     EG2 : sbit at PORTB.B4;
     RELEASE : sbit at PORTB.B5;
```

4.3.2. Le procedure

```
//*****************************************************************************
//Procedures e Functions
//*****************************************************************************
procedure assemble_cv;
begin
     asm
        movf          _get_midi
        movwf         $2B
        movlw         0
        movwf         $2A
     end;
     temp_word := temp_word shr 3; //Originale: 0nnnnnnn -> 0000nnnn|nnn00000
     temp_calc := current_pitch;  //Originale: 0sdddddd
     if temp_calc.B6 = 1 then     //verifica del segno
        begin
             temp_calc := temp_calc and $3F;      //pitch positivo
             temp_word := temp_word + temp_calc;
        end
     else
         begin
              temp_calc := 64 - temp_calc;     //pitch negativo
              temp_word := temp_word - temp_calc;
         end;
     //Composizione del dato temp_word : 0000nnnn|nnnppppp
     //Clamping il valore della nota deve essere tra 36 e 84
     if temp_word < 1152 then temp_word := 1152;
     if temp_word > 2688 then temp_word := 2688;
end;
//*****************************************************************************
procedure find_octave;
begin
      temp_word := temp_word shl 3;
      current_note := hi(temp_word);
      current_note := current_note - 36;
      octave := 0;
      while current_note > 11 do
        begin
             current_note := current_note - 12;
             inc (octave);

        end;
      asm
        movf      _current_note
        movwf     $2B
      end;
      table_pointer := temp_word shr 3;
end;
//*****************************************************************************
procedure write_note;
begin
     assemble_cv;
     find_octave;
     CCPR2L := hi(table_note[table_pointer]);
     temp_calc := lo(table_note[table_pointer]);
     CCP2CON := CCP2CON and %11001111;
     if temp_calc.B7 = 1 then CCP2CON.B5 := 1;
     if temp_calc.B6 = 1 then CCP2CON.B4 := 1;
     PORTB := PORTB and %11111000;
     PORTB := PORTB or octave;
end;

//*****************************************************************************
procedure write_vel;
begin
        temp_calc := get_midi shl 1;
        if get_midi = 0 then
         begin
              Dec(note_count);
              if note_count = 0 then RELEASE := 1;
          end
        else
            begin
            Inc(note_count);
            CCPR1L := temp_calc;
            RELEASE := 0;
            end;
end;
//*****************************************************************************
procedure write_pb;
begin
     current_pitch := get_midi;
```

```
end;
//*****************************************************************************
procedure envelope;
begin
     if EG1_IN = 1 then attack1 := 1;
     if EG2_IN = 1 then attack2 := 1;
     if attack1 = 1 then EG1 := 1;
     if attack2 =1 then EG2 :=1;
end;
//*****************************************************************************
```

4.3.3. Main i settaggi delle variabili

```
//*****************************************************************************
//Main Program
//*****************************************************************************
begin                       //Main
     run_stat := 0;
     note_count := 0;       //Azzera il contatore di note attive
     note_on_flag := 0;     //Azzera il flag di nota ricevuta
 //Posizione dei bit
 //PORTA:
 //          *RA3  EG1_IN input la tensione di inviluppo ha raggiunto il amssimo attack
 //          *RA4  EG2_IN input la tensione di inviluppo ha raggiunto il amssimo attack
 //          *RA5  Extenal Gate input per gate esterno
 //PORTB:
 //          *RB0  OCT_1        out  selettore ottava LSB
 //          *RB1  OCT_2        out  selettore ottava
 //          *RB2  OCT_3        out  selettore ottava MSB
 //          *RB3  EG1          out  comando attack/decay envelope 1
 //          *RB4  EG2          out  comando attack/decay envelope 2
 //          *RB5  RELEASE      out  comando release
 //PORTC:
 //          *RC1  PWM_CV       out  uscita PWM CV
 //          *RC2  PWM_KVEL     out  uscita PWM KEY VELOCITY
 //          *RC7  MIDI_IN      input ingresso MIDI

     trisa := $FF;
     trisb := 0;
     trisc := %11111001;
     current_note := 60;
     save_last_note := current_note;
     current_pitch := 64;
     OCT_1 := 0;
     OCT_2 := 0;
     OCT_3 := 0;
     EG1 := 0;
     EG2 := 0;
     RELEASE := 1;
     note_count := 0;
     attack1 := 0;
     attack2 := 0;

//Impostazione Timer2 per il PWM
     T2CON.B2 := 1;
     PR2 := $FF;          //f=19,5kHz
//Impostazioni PWM
     CCP1CON := $0F;
     CCP2CON := $0F;
     CCPR1L := $80;
     CCPR2L := $80;


     uart1_init(31250);    //Inizializza la linea midi a 31250 baud
```

4.3.4. Main

```
//Main Loop
     while true do
     begin
          if uart1_data_ready() = 1 then //Verifica se è arrivato un dato
             begin                       //E' arrivato un dato valido
             get_midi := uart1_read();   //Legge il dato midi
             if get_midi < $80 then      //Running Status
                    begin
                        case run_stat of
                        $90: begin
                             // Verifica che la nota in arrivo sia compresa tra i valori min-max
                                if get_midi > 84 then overrange := 1
                                else overrange := 0;
                                if get_midi < 36 then overrange := 1
                                else overrange := 0;
                             // Se è nel range allora procede
                                if overrange = 0 then
                                   begin
                                        save_last_note := get_midi;
                                        write_note;
                                   end;
                                run_stat := $91;
                             end;
                        $91: begin
                                  if overrange = 0 then write_vel;
                                  run_stat := $90;
                             end;
                        $E0: run_stat := $E1;
                        $E1: begin
                             write_pb;
                             get_midi := save_last_note;
                             write_note;
                             run_stat := $E0;
                             end;
                        $80: run_stat := $81;
                        $81: begin
                             get_midi := 0;
                             write_vel;
                             run_stat := $80;
                             end;
                        $C0: begin
                             run_stat := $C1;
                             end;
                        $C1: begin
                             run_stat := $C0;
                             end;
                        end;
                    end
               else
                    begin
                    if get_midi < $F8 then get_midi := get_midi and $F0;
                       case get_midi of
                       $90:  run_stat := $90;            //Note ON/OFF
                       $E0:  run_stat := $E0;            //pitch Bend
                       $F8:  asm
                                nop
                             end;                        //Sync
                       $FE:  asm
                                nop
                             end;                        //Active sensing
                       $FA:  asm
                                nop
                             end;
                       $FC:  asm
                                nop
                             end;                       //Disabilita il Sync
                       $80:  run_stat := $80;           //Note OFF
                       $C0:  run_stat := $C0;           //Non usato
                       end;
                    end;
             end;
          if RELEASE = 0 then envelope
          else begin
                    attack1 := 0;
                    attack2 := 0;
                    EG1 := 0;
                    EG2 := 0;
               end;
     end;
end.
```