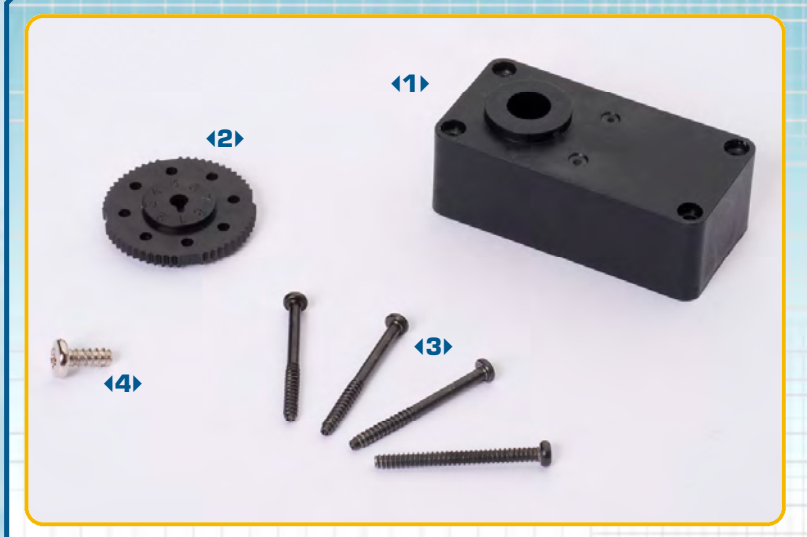
ANALIZZIAMO IL PROGRAMMA 'TEMPLATE'

Nelle prossime pagine trovi la prima parte dell'esempio Template.bas, insieme a una descrizione dettagliata delle nuove istruzioni RoboBasic.

Dopo aver presentato le sequenze motorie che RoboZak può compiere grazie al programma **Template.bas**, possiamo analizzare il codice di questo importante esempio **RoboBasic**. A pagina 8, 9 e 10 trovi la prima parte del codice, che comprende la struttura centrale di controllo. Per ragioni di spazio è stato omesso il codice delle routine di movimento (il prossimo **CD-Rom** conterrà l'esempio completo). Questo programma ti permette di controllare il robot attraverso tre modalità di funzionamento: con il telecomando a infrarossi **Remocon**, con il software **RoboRemocon** (il telecomando 'virtuale'), oppure tramite un altro programma **RoboBasic** (mediante una modalità di programmazione chiamata '**multiprogrammazione**'). Per controllare il robot utilizzando il telecomando Remocon è necessario per prima cosa impostare l'**ID** di controllo (vedremo questa operazione nel prossimo fascicolo). Il software RoboRemocon, come già visto più volte, ti consente di sfruttare il personal computer per simulare il telecomando **Remocon** (ovviamente il robot deve essere collegato al PC tramite il cavo seriale). La multiprogrammazione, infine, ti permette di caricare un nuovo programma sulla scheda **MR-C3024** e invocare le sequenze di movimento del programma **Template.bas** per mezzo dell'istruzione **ACTION**. A pagina 10, 11 e 12 trovi una spiegazione dettagliata delle nuove istruzioni **RoboBasic** e alcune importanti informazioni sulla multiprogrammazione.



COMPONENTI

- **‹1› base superiore per servo tipo B**
- **‹2› squadretta circolare di tipo 3 per servo**
- **‹3› 4 viti tipo T-2 da 2x21 mm (nere)**
- **‹4› vite tipo T-2 da 2,6x6 mm**

CODICE ROBOBASIC: TEMPLATE.BAS›››

In queste due pagine è mostrato il codice sorgente del programma Template.bas. Per ragioni di spazio è riportata solo la prima parte del codice, contente la struttura di controllo dell'esempio. La restante parte, ossia il codice delle routine di movimento, è stata omessa (sul sesto CD-Rom troverai l'esempio completo e potrai caricarlo sul robot). A pagina 10, 11 e 12 trovi una descrizione dettagliata circa il funzionamento delle nuove istruzioni utilizzate. Le prime due righe di codice fanno sì che quest'ultimo venga caricato nella parte alta della memoria della scheda MR-C3024. La prima variabile, RR, viene utilizzata sia per memorizzare il codice numerico inviato dal telecomando virtuale RoboRemocon, sia per il passaggio di parametri tra il programma Template.bas e un eventuale programma presente nella parte bassa della memoria (nel caso sia stata effettuata una multiprogrammazione della scheda, come spiegato a pagina 10). Inizialmente viene controllato il valore della variabile RR: se il suo valore è compreso tra '51' e '82' (ossia se il programma Template.bas è stato avviato per mezzo di un comando ACTION da parte di un programma caricato nella parte bassa della memoria), viene invocata la routine action_proc. In caso contrario l'esecuzione prosegue e si entra nella routine MAIN. Se RR è pari a '0' (ossia se non è stato inviato alcun segnale alla scheda da parte del software RoboRemocon) viene invocata la routine MAIN1, che si occupa della gestione del telecomando Remocon. Altrimenti (RR diverso da '0') viene invocata una routine di movimento in base al valore inviato dal software RoboRemocon.

```
'====================================
' Template.bas
'====================================

GOTO AUTO
FILL 255,10000

DIM RR AS BYTE
DIM A AS BYTE
DIM A16 AS BYTE
DIM A26 AS BYTE

CONST ID = 0 ' 1:0, 2:32, 3:64, 4:96

IF RR > 50 AND RR < 83
   THEN GOTO action_proc

RR = 0
PTP SETON
PTP ALLON

DIR G6A,1,0,0,1,0,0
DIR G6B,1,1,1,1,1,1
DIR G6C,0,0,0,0,0,0
DIR G6D,0,1,1,0,1,0

TEMPO 230
MUSIC "CDE"
GETMOTORSET G6A,1,1,1,1,1,0
GETMOTORSET G6B,1,1,1,0,0,0
GETMOTORSET G6C,1,1,1,0,0,0
GETMOTORSET G6D,1,1,1,1,1,0
'== motor power on
SPEED 5
MOTOR G24
GOSUB standard_pose

MAIN:
IF RR = 0 THEN GOTO MAIN1
ON RR GOTO
  MAIN, K1, K2, K3, K4, K5, K6, K7,
  K8, K9, K10, K11, K12, K13, K14,
  K15, K16, K17, K18, K19, K20, K21,
  K22, K23, K24, K25, K26, K27, K28,
  K29, K30, K31, K32
GOTO main_exit

MAIN1:
A = REMOCON(1)
A = A - ID
ON A GOTO
  MAIN, K1, K2, K3, K4, K5, K6, K7,
  K8, K9, K10, K11, K12, K13, K14,
  K15, K16, K17, K18, K19, K20, K21,
  K22, K23, K24, K25, K26, K27, K28,
  K29, K30, K31, K32
GOTO MAIN

action_proc:
A = RR - 50
ON A GOTO
  MAIN, K1, K2, K3, K4, K5, K6, K7,
  K8, K9, K10, K11, K12, K13, K14,
  K15, K16, K17, K18, K19, K20, K21,
  K22, K23, K24, K25, K26, K27, K28,
  K29, K30, K31, K32
RETURN

main_exit:
I'F RR > 50 THEN RETURN
RR = 0
GOTO MAIN
```

```
k1:
    GOSUB bow_pose
    GOSUB standard_pose
    GOTO main_exit

k2:
    GOSUB hands_up
    DELAY 500
    GOSUB standard_pose
    GOTO main_exit

k3:
    GOSUB sit_down_pose
    DELAY 1000
    GOSUB standard_pose
    GOTO main_exit

k4:
    GOSUB sit_hands_up
    DELAY 1000
    GOSUB standard_pose
    GOTO main_exit

k5:
    GOSUB foot_up
    GOSUB standard_pose
    GOTO main_exit

k6:
    GOSUB body_move
    GOSUB standard_pose
    GOTO main_exit

k7:
    GOSUB wing_move
    GOSUB standard_pose
    GOTO main_exit

k8:
    GOSUB right_shoot
    GOSUB standard_pose
    DELAY 500
    GOSUB left_shoot
    GOSUB standard_pose
    DELAY 500
    GOTO main_exit

k9:
    SPEED 8
    GOSUB handstanding
    DELAY 1000
    SPEED 6
    GOSUB standard_pose
    GOTO main_exit

k10:
    GOSUB fast_walk
    GOSUB standard_pose
    GOTO main_exit

k11:
    GOSUB forward_walk
    GOSUB standard_pose
    GOTO main_exit

k12:
    GOSUB backward_walk
    GOSUB standard_pose
    GOTO main_exit

k13:
    SPEED 8
    GOSUB right_shift
    SPEED 6
    GOSUB standard_pose
    GOTO main_exit

k14:
    SPEED 8
    GOSUB left_shift
    SPEED 6
    GOSUB standard_pose
    GOTO main_exit

k15:
    GOSUB left_attack
    GOSUB standard_pose
    GOTO main_exit

k16:
    GOSUB sit_down_pose16
    GOTO main_exit

k17:
    GOSUB left_forward
    GOSUB standard_pose
    GOTO main_exit

k18:
    TEMPO 230
    MUSIC "C"
    GOTO main_exit

k19:
    GOTO main_exit

k20:
    GOSUB right_attack
    GOSUB standard_pose
    GOTO main_exit

k21:
    GOSUB forward_tumbling
    GOSUB standard_pose
    GOTO main_exit
```

```
k22:
    GOSUB left_turn
    GOSUB standard_pose
    GOTO main_exit

k23:
    TEMPO 230
    MUSIC "D"
    GOTO main_exit

k24:
    GOSUB right_turn
    GOSUB standard_pose
    GOTO main_exit

k25:
    GOTO main_exit

k26:
    GOSUB sit_down_pose26
    GOTO main_exit

k27:
    GOSUB right_forward
    GOSUB standard_pose
    GOTO main_exit
```

```
k28:
    GOSUB left_tumbling
    SPEED 10
    GOSUB standard_pose
    GOTO main_exit

k29:
    GOSUB forward_punch
    SPEED 10
    GOSUB standard_pose
    GOTO main_exit

k30:
    GOSUB righ_tumbling
    SPEED 10
    GOSUB standard_pose
    GOTO main_exit

k31:
    GOSUB back_tumbling
    SPEED 10
    GOSUB standard_pose
    GOTO main_exit

k32:
    TEMPO 230
    MUSIC "E"
    GOTO main_exit
```

ISTRUZIONI ROBOBASIC

MULTIPROGRAMMAZIONE

Prima di dedicarci all'analisi delle nuove istruzioni introdotte con il codice del programma Template.bas, è necessario parlare di un'importante caratteristica della scheda di controllo MR-C3024, ossia la possibilità di caricare due programmi distinti sulla memoria di quest'ultima. La memoria della scheda, dove vengono caricati i programmi RoboBasic, è assimilabile a una sequenza di 'celle' all'interno delle quali vengono inserite le istruzioni e le variabili (opportunamente tradotte in linguaggio macchina). La parte iniziale della memoria, dove risiede la prima porzione di celle, viene chiamata 'memoria bassa' (più precisamente si tratta dei primi 10.000 byte). La parte finale, invece, viene chiamata 'memoria alta'. Se sulla scheda sono presenti due programmi, quando essa viene accesa il suo software di controllo fa sì che venga avviato il programma residente nella parte bassa della memoria. Spetta a quest'ultimo, utilizzando una speciale istruzione RoboBasic (ACTION), passare il controllo al programma caricato nella parte alta. Per sfruttare questa caratteristica della scheda di controllo è necessario che i due programmi caricati soddisfino alcuni importanti requisiti. Per prima cosa i due programmi devono essere caricati sulla scheda rispettando un ordine preciso, iniziando da quello che vogliamo far risiedere nella parte alta della memoria. Quest'ultimo deve obbligatoriamente iniziare con una precisa coppia di istruzioni: 'GOTO AUTO' e 'FILL 255, 10000'. Tali comandi, come vedremo a breve, sono indispensabili per far sì che il programma non venga

memorizzato nella parte iniziale nella memoria (parte bassa) bensì più avanti (nella parte alta). Senza queste istruzioni il programma sarebbe caricato nella parte bassa e il successivo caricamento di un secondo programma provocherebbe una cancellazione (in gergo tecnico chiamata 'sovrascrittura') del primo programma. Dopo aver caricato il primo programma bisogna caricare sulla scheda di controllo il secondo, verificando però che la sua lunghezza non superi i 10.000 byte (pena la sovrascrittura del primo programma). Il programma Template.bas soddisfa tutti i requisiti necessari per essere caricato nella parte alta della memoria. Dopo avere caricato questo programma, quindi, sarà possibile inserire nella scheda un nuovo programma in grado di richiamare le sequenze motorie presenti in Template.bas.

GOTO AUTO
FILL 255, 10000

Questo blocco di istruzioni fa sì che il codice del programma non venga caricato nella parte iniziale della memoria presente sulla scheda di controllo, bensì dopo il decimillesimo byte. Per fare questo i primi 10.000 byte della memoria vengono riempiti con il valore '255' (questo corrisponde a riempire ogni singolo bit con il valore '1'). Quando si carica un secondo programma, esso viene posizionato nei primi 10.000 byte: in questo modo il primo programma non viene sovrascritto e sulla scheda di controllo possono convivere i due singoli programmi (a patto che il secondo programma, una volta compilato, non occupi più di 10.000 byte). IMPORTANTE: le due istruzioni devono essere le prime istruzioni del codice (escludendo i commenti).

ACTION X

Quando sono caricati due diversi programmi sulla scheda di controllo, all'accensione di quest'ultima viene avviato il programma memorizzato nella parte bassa della memoria. L'istruzione ACTION permette a quest'ultimo di interrompersi e avviare il programma residente nella parte alta della memoria. Il parametro X permette di passare un valore numerico al secondo programma: tale valore viene memorizzato nella prima variabile dichiarata all'interno del programma residente nella parte alta (nel caso del programma Template.bas, la prima variabile dichiarata è RR). In realtà, il programma residente nella parte alta della memoria riceve il valore passato aumentato di '50'. Per fare in modo che il controllo ritorni al programma precedente, è sufficiente utilizzare l'istruzione RETURN.

Esempio:
Supponiamo di avere caricato sulla scheda di controllo i due semplici programmi seguenti, caricando per primo 'Programma1.bas' e per secondo 'Programma2.bas'.

```
' Programma1.bas
GOTO AUTO
FILL 255, 10000
DIM A AS BYTE
DIM B AS BYTE
B = A - 50
IF B = 1 THEN MUSIC "ABC"
ELSEIF B = 2 THEN MUSIC "DEF"
ENDIF
RETURN
```

```
'Programma2.bas
DIM A AS BYTE
ACTION 1
ACTION 2
END
```

All'accensione della scheda viene avviato il secondo programma. L'istruzione 'ACTION 1' avvia l'esecuzione del primo programma, passandogli il valore '51' (ossia '1' aumentato di '50', come previsto dal comando ACTION). Quest'ultimo emette la sequenza musicale 'ABC' e ripassa il controllo al primo programma. L'istruzione 'ACTION 2' avvia nuovamente l'esecuzione del primo programma, questa volta passandogli il valore '52' ('2' + '50'). Dopo l'esecuzione della sequenza musicale 'DEF' il controllo passa nuovamente al secondo programma, che termina la sua esecuzione per mezzo dell'istruzione END.

CONST X = K

Permette di definire una costante, ossia un valore numerico fisso che non cambia durante l'esecuzione del programma. Come per le variabili, le costanti vengono identificate da una sigla alfanumerica. A differenza di queste ultime, però, il loro valore è costante e non può cambiare durante l'esecuzione del programma. Il parametro X definisce la costante, il parametro K definisce il valore numerico (può essere un BYTE o un INTEGER).

Esempio:

```
CONST ID = 0
```

Definisce la costante ID, associandole il valore '0'.

A = REMOCON(1)

Questa istruzione legge il valore inviato dal telecomando Remocon al ricevitore a infrarossi (che va collegato alla porta AD7 della scheda di controllo). Il parametro A è una variabile in cui viene memorizzato il valore numerico associato al pulsante premuto (da '1' a '32'). Se non viene premuto nessun pulsante nella variabile viene memorizzato il valore '0'. La scheda MR-C3024 è in grado di gestire quattro diversi canali per comunicare con il telecomando. Questo significa che è possibile comandare quattro diversi RoboZak in contemporanea, con quattro differenti Remocon, senza interferenze. La procedura per impostare l'ID del telecomando sarà presentata nel prossimo fascicolo. Quando viene impostato l'ID numero '1', nella variabile A viene memorizzato il valore preciso del tasto premuto. Quando invece è impostato un ID pari a '2', '3' o '4', la variabile A contiene il valore del tasto aumentato rispettivamente di '32', '64' e '96'. Quando si utilizza il programma Template.bas con un ID diverso da '1', è necessario modificare la riga di codice 'CONST ID = 0', inserendo il valore '32', '64' o '96' a seconda dell'ID impostato.

Esempio:

```
A = REMOCON(1)
```

Memorizza nella variabile A il valore numerico associato al tasto premuto su Remocon.

RIEPILOGO COMPONENTI

In questo elenco trovi tutte le tipologie di pezzi che ti sono state fornite a partire dal primo fascicolo: puoi consultarlo quando devi affrontare le fasi di montaggio, in modo da avere un riferimento immediato per i componenti che dovrai utilizzare e per quelli che hai a disposizione.

- armatura del dorso
- armatura del torace
- base inferiore per servo A
- base inferiore per servo B
- base inferiore per servo C
- base superiore per servo A
- base superiore per servo B
- base superiore per servo C
- bullone da 3x4 mm
- caricabatterie
- cavo di prolunga per pacco batterie
- cavo seriale
- circuito con LED

- coperchio vano batterie
- copertura in plastica del piede sinistro e destro
- cuscinetto a sfera
- distanziatore da 3x5 mm
- elementi plastici della mano
- fascetta di fissaggio dei cavi
- fascetta di metallo
- fascetta in plastica per il raggruppamento dei cavi
- guaina in plastica proteggi cavo
- intelaiatura metallica del dorso
- intelaiatura metallica del piede
- intelaiatura metallica superiore
- intelaiatura metallica del polso
- intelaiatura metallica del torace
- motore elettrico cavo 200 mm (6N200 - Servo C)
- motore elettrico cavo 300 mm (4N300 - Servo A)
- motore elettrico cavo 400 mm (5N400 - Servo B)
- nastro biadesivo
- pacco batterie ricaricabili
- parte anteriore della testa
- parte posteriore della testa
- perno da 1,6x14 mm
- perno da 1,6x9 mm
- protezione per scheda MR-C3024
- ricevitore IR
- rondella da 6x2,2x0,5 mm
- rondella da 7,6x2,8x0,5 mm
- ruota dentata di tipo 1
- ruota dentata di tipo 2
- ruota dentata di tipo 3
- ruota dentata di tipo 4
- scheda MR-C3024
- scheda PC Servo Control
- sensore di contatto
- sensore di distanza
- sensore di luce
- sensore di suono
- sostegno per potenziometro
- squadrette circolari per servo (tipo 1, 2, 3, 4)
- squadretta circolare per il fissaggio della testa
- squadretta metallica a I
- squadrette metalliche a U (16 fori e 22 fori)
- squadretta metallica ad H
- squadretta metallica spalle (interna ed esterna)
- telecomando Remocon
- tubetto di grasso
- visiera
- vite di tipo M da 2,6x4 mm
- vite di tipo M da 2x4 mm
- vite di tipo M da 3x4 mm
- vite di tipo T-2 da 2,6x6 mm
- vite di tipo T-2 da 2x12 mm
- vite di tipo T-2 da 2x18 mm
- vite di tipo T-2 da 2x21 mm (nera)
- vite di tipo T-2 da 2x26 mm (nera)
- vite di tipo T-2 da 2x4 mm
- vite di tipo T-2 da 2x5 mm
- vite di tipo T-2 da 2x8 mm