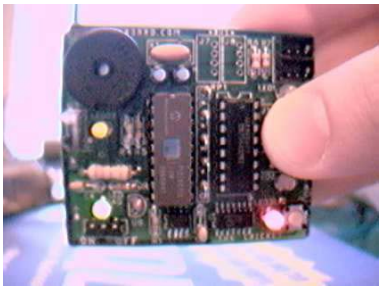
คู่มือคำสั่ง Cricket Logo

แปลโดย อานันท์ สีห์พิทักษ์เกียรติ
ท่านสามารถ download ต้นฉบับของเอกสารนี้ได้จาก
http://www.media.mit.edu/~arnans/cricket
แปลจากต้นฉบับภาษาอังกฤษโดย Fred Martin
http://handyboard.com/cricket/cricket-logo.html

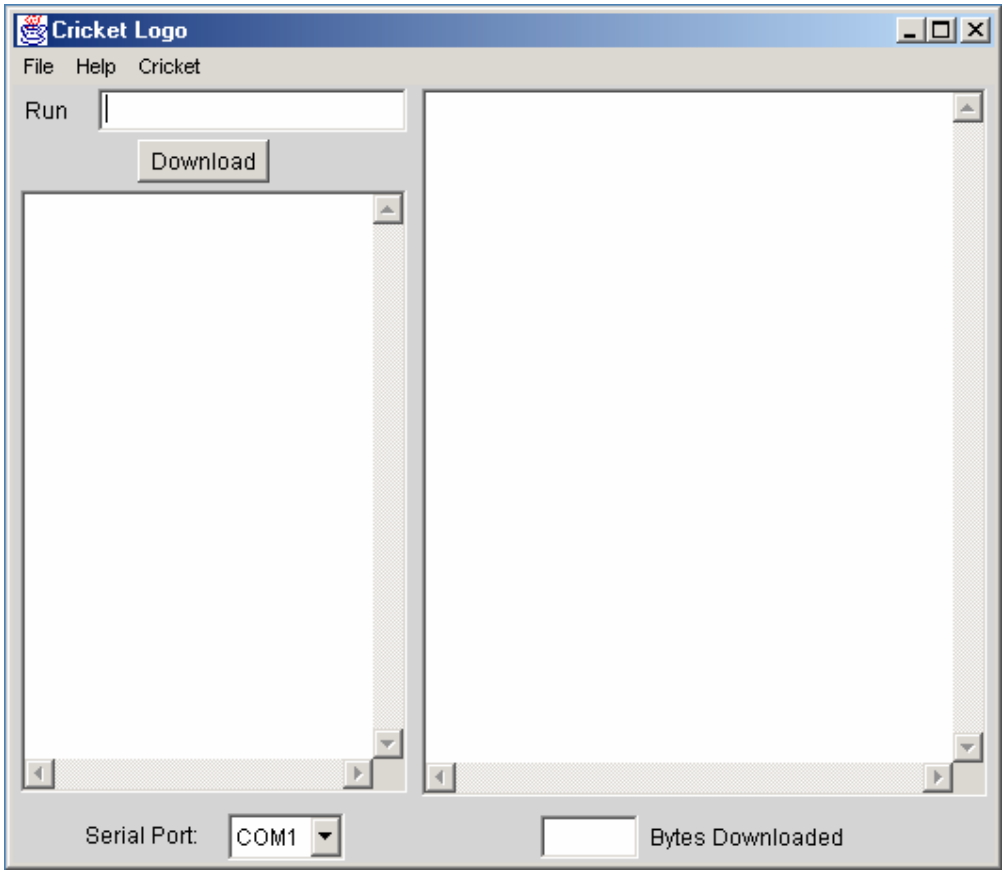


ภาพรวมของ Cricket Logo

Cricket Logo มีคุณสมบัติและความสามารถดังต่อไปนี้:

- สามารถสร้าง procedure ที่มีการส่งผ่านและรับค่ากลับได้;
- สามารถสร้างตัวแปร Global ได้ และสามารถสร้างตัวแปร local โดยผ่านการรับค่าของ procedure ;
- มีคำสั่งโครงสร้าง เช่น if, repeat, และ loop;
- ใช้ระบบตัวเลข 16 บิท (ในการ บวก, ลบ, คูณ, หาร, หารเอาเศษ, การเปรียบเทียบ, การกระทำกับบิท (bitwise perations), และ การสร้างค่าสุ่ม) ;
- มีคำสั่งเพื่อใช้งาน sensor และ motor;
- มีคำสั่งเกี่ยวกับเวลา และการสร้างเสียงหลายระดับ ;
- มีคำสั่งในการ บันทึก และ เรียกคืน ข้อมูล;
- มีคำสั่งเกี่ยวกับการติดต่อสื่อสาร.

หน้าจอหลักของ Cricket Logo



ภาพข้างบนนี้แสดงให้เห็นหน้าจอของ Cricket Logo

บริเวณด้านซ้ายของจอคือ ศูนย์คำสั่ง (command center) คำสั่งที่ป้อนเข้าไปในบริเวณนี้จะถูกส่งไปยัง Cricket และเรียกใช้งานในทันที ในภาพแสดงคำสั่ง beep ซึ่งเป็นคำสั่งที่จะทำให้ Cricket ส่งเสียงออกมา

บริเวณทางด้านขวามีไว้เพื่อสร้าง procedure ซึ่ง cricket logo กำหนดไว้ว่าจะต้องเริ่มต้นด้วยคำว่า to และจบด้วยคำว่า end ตัวอย่างต่อไปนี้แสดงให้เห็น procedure ชื่อ demo ซึ่งจะเปิดมอร์เตอร์ A เป็นเวลาหนึ่งวินาทีแล้วส่งเสียง:

```
to demo
  a, onfor 10
  beep
end
```

เมื่อกดปุ่ม "Download" โปรแกรมจะส่ง procedure ทุก procedure ที่สร้างไว้ไปยัง cricket

กล่องข้อความด้านบนซ้ายของจอใช้กำหนดว่าหลังจากที่ download โปรแกรมแล้วต้องการให้ cricket เริ่มทำงานจาก procedure ใดเมื่อกดปุ่ม Run/Stop บนตัว cricket

มอร์เตอร์

Cricket มีมอร์เตอร์สองตัวชื่อว่า A และ B มอร์เตอร์แต่ละตัวจะมีไฟสถานะอยู่ สีของไฟนี้จะบ่งบอกว่ามอร์เตอร์แต่ละตัวอยู่ในสถานะใด

การใช้งานมอร์เตอร์นั้นจะเริ่มต้นด้วยการเลือกมอร์เตอร์ (ใช้คำสั่ง a, b, หรือ ab,) แล้วบอกมันว่าต้องการให้ทำอะไร (เช่น, on (เปิด), off(ปิด), rd (กลับทิศ) ฯลฯ)

a,	เลือกสั่งงานมอร์เตอร์ A
b,	เลือกสั่งงานมอร์เตอร์ B
ab,	เลือกสั่งงานมอร์เตอร์ AB
on	เปิดมอร์เตอร์ที่เลือกไว้
off	ปิดมอร์เตอร์ที่เลือกไว้
onfor *ระยะเวลา*	เปิดมอร์เตอร์ไว้เป็นเวลาหนึ่ง, "ระยะเวลา" เป็นตัวกำหนดว่ามอร์เตอร์จะถูกเปิดไว้เป็นเวลานานเท่าใด หน่วยของเวลาคือหนึ่งในสิบของวินาที ยกตัวอย่างเช่น onfor 10 จะเปิดมอร์เตอร์ไว้เป็นเวลาหนึ่งวินาที
thisway	กำหนดทิศทางการหมุนของมอร์เตอร์ให้เป็น "ทางนี้" (ซึ่งจะเป็นทางไหนนั้นขึ้นอยู่กับการต่อมอร์เตอร์) ไฟสถานะเมื่อมอร์เตอรหมุน "ทางนี้" จะเป็นสีเขียว
thatway	กำหนดทิศทางการหมุนของมอร์เตอร์ให้เป็น "ทางนั้น" (ซึ่งจะเป็นทางตรงกันข้ามกับ "ทางนี้") ไฟสถานะเมื่อมอร์เตอรหมุน "ทางนั้น" จะเป็นสีแดง
rd	กลับทิศการหมุน ไม่ว่ามอร์เตอร์จะหมุนทางใด คำสั่งนี้จะกลับทิศการหมุนนั้นให้เป็นทางตรงข้าม
setpower *ระดับ*	ตั้งค่ากำลังของมอร์เตอร์: ค่า "ระดับ" มีช่วงอยู่ระหว่าง 0 (ไม่กำลังเลย มอร์เตอร์จะหยุด) ไปจนถึง 8 (เต็มกำลัง) ค่าระดับกำลังปกติของมอร์เตอร์คือ 4

เวลาและเสียง

คำสั่งเกี่ยวกับเวลาและเสียงใช้เพื่อสั่ง cricket ให้ทำงานเป็นระยะเวลาหนึ่ง

เวลา

ด้วยคำสั่ง wait เราสามารถบอกให้ cricket "อยู่เฉยๆ" ให้เวลาผ่านไประยะหนึ่ง ในขณะที่ cricket อยู่เฉยๆ นี้ มอร์เตอร์จะถูกเปิดทิ้งไว้ก็ได้ เช่น

ab, on wait 20 off

จะเปิดมอร์เตอร์ A และ B ไว้เป็นเวลา 2 วินาที คำสั่งนี้ให้ผลเหมือนกันกับ:

ab, onfor 20

นอกจากนี้ในตัว cricket ยังมีนาฬิกาที่เดินอยู่ตลอดเวลาแม้ในขณะที่ cricket กำลังทำงานอย่างอื่นอยู่ มีคำสั่งสองคำสั่งที่ใช้งานนาฬิกานี้คือ resett ซึ่งจะทำให้นาฬิการเริ่มนับจากศูนย์ใหม่ และ timer ซึ่งจะแจ้งค่านาฬิกาปัจจุบัน (เป็นค่าตัวเลขค่าหนึ่ง)

ตารางต่อไปนี้สรุปการใช้งานคำสั่ง wait, timer, และ resett

wait *ระยะเวลา*	หยุดอยู่เฉยๆ เป็นเวลาเท่ากับค่า "ระยะเวลา" ที่กำหนด หน่วยของเวลาคือ หนึ่งในสิบวินาที ตัวอย่างเช่น wait 10 จะทำให้ cricket อยู่เฉยๆ เป็นเวลาหนึ่งวินาที.
timer	แจ้งค่านาฬิกา หน่วยของเวลาจะนับทุกๆ 4 ms (1ms คือหนึ่งในพันวินาที) ดังนั้นถ้า timer = 250, แสดงว่านาฬิกาเริ่มนับมาเป็นเวลาหนึ่งวินาที
resett	บอกให้นาฬิกาเริ่มนับจากศูนย์ใหม่

เสียง

cricket มีลำโพงในตัวซึ่งสามารถใช้สร้างเสียงระดับต่างๆ ได้ คำสั่งเกี่ยวกับเสียงมีอยู่สองคำสั่งด้วยกันคือ beep ซึ่งจะส่งเสียงร้องออกมาเป็นเวลาสั้นๆ อีกคำสั่งหนึ่งคือ note ซึ่งสามารถกำหนดระดับและความยาวของเสียงได้

beep	ส่งเสียงร้องสั้นๆ
note *ระดับ ระยะเวลา*	สร้างเสียงตามค่า "ระดับ" และเล่นเสียงนั้นเป็นเวลาเท่ากับ "ระยะเวลา" เมื่อเพิ่มค่า "ระดับ" จะได้เสียงที่ต่ำลง หน่วยของ ระยะเวลา คือหนึ่งในสิบวินาที ตารางต่อไปนี้แสดงให้เห็นความสัมพันธ์ของค่า "ระดับ" และโน๊ตดนตรี เริ่มจากเสียง C กลางไปยัง C สูง

ระดับ	119	110	110	105	100	100	94	89	84	84	79	74	74	70	66	66	62	59
โน๊ตดนตรี	c	c#	db	d	d#	eb	e	f	f#	gb	g	g#	ab	a	a#	bb	b	c2

ตัวอย่างเช่น note 119 5 จะส่งเสียงโน๊ต C กลางเป็นเวลาครึ่งวินาที

ข้อสังเกต: หน่วยของเวลาที่ใช้กับคำสั่งในกลุ่มนี้มีอยู่สองหน่วยด้วยกันคือ หน่วยที่ใช้กับคำสั่ง wait, onfor, และ note จะมีค่าเท่ากับ 0.1 วินาที หน่วยที่สองคือหน่วยที่ใช้กับคำสั่ง timerจะมีค่าเท่ากับ 0.004 วินาที

เซ็นเซอร์

Cricket มีเซ็นเซอร์อยู่สองพอร์ทด้วยกัน ชื่อว่า A และ B; มีอุปกรณ์หลายชนิดที่สามารถใช้งานกับเซ็นเซอร์นี้ได้ เช่น

- อุปกรณ์ที่มีสถานะเปิด/ปิด เช่นปุ่ม หรือ สวิตช์ชนิดต่างๆ
- เซ็นเซอร์ที่เปลี่ยนแปลงค่าความต้านทานของตัวเอง เช่น เซ็นเซอร์แสง (light-sensitive photocells) หรือ เซ็นเซอร์อุณหภูมิ (temperature-sensitive thermistors)
- อุปกรณ์อิเลคทรอนิคส์ใดๆ ที่ผลิตความต่างศักดิ์ (voltage) ระหว่าง 0 ถึง 5 โวลต์

คำสั่งที่ใช้อ่านค่าเซ็นเซอร์มีอยู่สองแบบด้วยกัน แบบแรกเรียกว่า switch ซึ่งจะส่งค่าจริงหรือเท็จกลับมาเท่านั้น (ใช้กับเซ็นเซอร์ที่มีสองสถานะคือเปิดหรือปิด) แบบที่สองคือ sensor ซึ่งจะส่งค่าตัวเลขระหว่าง 0 ถึง 255 ออกมา โดยค่าที่ได้จะขึ้นอยู่กับเซ็นเซอร์ที่ใช้

switcha	ถ้าสวิทช์ที่ต่ออยู่กับเซ็นเซอร์ A ถูกกดอยู่ คำสั่งนี้จะรายงานค่า "จริง" ออกมา ถ้าไม่เช่นนั้นก็จะรายงานค่า "เท็จ"
switchb	ถ้าสวิทช์ที่ต่ออยู่กับเซ็นเซอร์ B ถูกกดอยู่ คำสั่งนี้จะรายงานค่า "จริง" ออกมา ถ้าไม่เช่นนั้นก็จะรายงานค่า "เท็จ"
sensora	รายงานค่าเซ็นเซอร์ A โดยค่านี้จะอยู่ระหว่าง 0 ถึง 255
sensorb	รายงานค่าเซ็นเซอร์ Bโดยค่านี้จะอยู่ระหว่าง 0 ถึง 255

คำสั่งโครงสร้าง

Cricket Logo มีคำสั่งโครงสร้างชุดเล็กๆ ที่มีประโยชน์มาก คำสั่งเหล่านี้ประกอบไปด้วยคำสั่งที่ใช้ในการ วนรอบ (loop), ทดสอบเงื่อนไข (conditional), รอแบบมีเงื่อนไข (busy-wait), และคำสั่งจบการทำงานของโปรแกรม

ภาพรวม

คำสั่งโครงสร้างที่มีใน Cricket logo ถูกสรุปไว้ในตารางต่อไปนี้

repeat *ครั้ง* [*คำสั่ง*]	วนทำ "คำสั่ง" เป็นจำนวนครั้งเท่ากับ "ครั้ง" ค่า "ครั้ง" อาจเป็นค่าคงที่,ค่าจากการคำนวน, หรือตัวแปรก็ได้
loop [*คำสั่ง*]	วนทำ "คำสั่ง" ไปแบบไม่มีที่สิ้นสุด
if *เงื่อนไข* [*คำสั่ง*]	ถ้า "เงื่อนไข" เป็นจริง จะทำ "คำสั่ง" เงื่อนไขที่มีค่าเป็น 0 ถือว่ามีค่าเป็น เท็จ ค่าอื่นที่ไม่ใช่ 0 จะถือว่ามีค่าเป็น จริง
ifelse *condition* [*คำสั่ง-1*] [*คำสั่ง-2*]	ถ้า "เงื่อนไข" เป็นจริง จะทำ "คำสั่ง-1" ถ้าไม่เช่นนั้น จะทำ "คำสั่ง-2"

waituntil [*เงื่อนไข*]	โปรแกรมจะรอและไม่ทำคำสั่งถัดไปจนกระทั่ง "เงื่อนไข" เป็นจริง โปรดสังเกตว่าเงื่อนไขจะต้องอยู่ในวงเล็บเหลี่ยม ไม่เหมือนกับคำสั่ง if และ ifelse ที่เงื่อนไขไม่ต้องอยู่ในวงเล็บใดๆ
stop	หยุดการทำงานของ procedure ปัจจุบัน และ กลับไปทำคำสั่งถัดไปใน procedure แม่ (procedure ที่เรียกใช้งาน procedure ปัจจุบัน)
output *ค่า*	หยุดการทำงานของ procedure ปัจจุบัน และ ส่ง "ค่า" กลับไปยัง procedure แม่

ตัวอย่าง

procedure ต่อไปนี้จะทำให้มอร์เตอร์ A หมุนกลับไปกลับมา 10 ครั้ง

```
to flippy
  repeat 10 [a, onfor 10 rd]
end
```

สองตัวอย่างต่อไปนี้แสดงให้เห็นวิธีที่จะทำให้มอร์เตอร์ A หมุนกลับไปกลับมาเรื่อยๆ ไม่มีวันสิ้นสุด

```
to flippy-forever-1
  loop [a, onfor 10 rd]
end

to flippy-forever-2
  a, onfor 10 rd
  flippy-forever-2
end
```

วิธีที่สอง (วิธีที่เหลีกเลี่ยงการใช้คำสั่ง loop) เป็นวิธีที่มีประสิทธิภาพมากกว่า เพราะ compiler มองมันเป็น tail recursion และแปลงมันเป็นคำสั่ง goto

procedure ต่อไปนี้ทำการเปิดมอร์เตอร์ A แล้วรอจดกว่าสวิทช์ B ถูกกด แล้วจึงปิดมอร์เตอร์

```
to on-wait-off
  a, on
  waituntil [switchb]
  off
end
```

procedure ต่อไปนี้จะทำการอ่านค่าสวิทช์ B อยู่เรื่อยๆ ถ้าสวิทช์ถูกกดมอร์เตอร์ A จะหมุนไป "ทางนี้" แต่ถ้าสวิทช์ไม่ถูกกดมอร์เตอร์จะหมุนไป "ทางนั้น"

```
to switch-controls-direction
  a, on
  loop [
    ifelse switchb [thisway][thatway]
  ]
end
```

ระบบตัวเลข

cricket ใช้ระบบตัวเลขขนาด 16 บิท ซึ่งหมายความว่าค่าตัวเลขที่สามารถใช้งานได้จะอยู่ระหว่าง -32768 ถึง +32767.

การใช้งานเครื่องหมายทางเลขคณิตจะต้องมีการเว้นวรรคทั้งสองด้านเสมอ นั่นคือการเขียน 3+4 เป็นรูปแบบที่ผิด รูปแบบที่ถูกคือ 3 + 4 (มีวรรคก่อนหน้าและหลังเครื่องหมายบวก).

cricket logo ไม่ได้ใช้ระบบลำดับความสำคัญของเครื่องหมายคณิตศาสตร์ที่เป็นมาตรฐานทั่วไป แต่จะถือเอาตามลำดับการเขียนเรียงจากซ้ายไปขวา ดังนั้น

3 + 4 * 5

จะมีค่าเท่ากับ 35 เพราะ cricket logo จะทำ 3 + 4 แล้วคูณผลลัพท์ด้วย 5. (ซึ่งต่างจากมาตรฐานการประมวลผลในภาษาคอมพิวเตอร์ทั่วไป ซึ่งจะถือว่า * สำคัญกว่า + ดังนั้นผลลัพท์ที่จะได้จะเป็น 4*5 แล้วบวกด้วย 3)

วงเล็บเป็นวิธีการที่ใช้ในการกำหนดลำดับก่อนหนังให้กับการคำนวน เช่น

(3 + (4 * 5))

ค่าที่ได้คือ 23.

ตารางต่อไปนี้แสดงเครื่องหมายทางคณิตศาสตร์ทั้งหมดที่มีใน cricket logo

+	บวก (แบบ infix)
-	ลบ (แบบ infix)
*	คูณ (แบบ infix)
/	หาร (แบบ infix)
%	หารเอาเศษ (เช่น 5 % 3 จะเท่ากับ 2)
and	ตรรกะ "และ" ใช้ทั้งกับการหาทางทางตรรกศาสตร์ (จริงหรือเท็จ) และ bitwise operation
or	ตรรกะ "หรือ"
not	ตรรกะ "ไม่"
random	ใช้สุ่มค่าตัวเลข ค่าที่ได้จะอยู่ระหว่าง -32768 ถึง +32768. ถ้าต้องการลดช่วงของค่าลง ให้ใช้การหารเอาเศษ (%) เช่น (random % 100) จะได้ค่าสุ่มตั้งแต่ 0 ถึง 99

Procedures และการรับ-ส่งค่า (input-output)

คำจำกัดความ

การสร้าง procedure จะเริ่มด้วยคำสั่ง to ตามด้วยชื่อ procedure ตามด้วยชุดคำสั่งที่เป็นใจความของ procedure แล้วจบด้วยคำสั่ง end ตัวอย่างต่อไปนี้ เป็นการสร้าง procedure ชื่อว่า flash ซึ่งทำการเปิดและปิดมอร์เตอร์ A สิบครั้ง

```
to flash
  repeat 10 [a, onfor 5 wait 5]
end
```

การรับค่า (Inputs)

เราสามารถกำหนด Procedures ให้ทำการรับค่าได้ ซึ่งค่าดังกล่าวจะกลายเป็นตัวแปรของ procedure นั้นๆ (local variable) การกำหนดการรับค่าจะทำโดยการใช้เครื่องหมาย colon (:) ตัวอย่างต่อไปนี้เป็นการสร้าง procedure ชื่อ flash ซึ่งมีการรับค่าหนึ่งค่า (ชื่อว่า n) ค่าที่รับเข้ามานี้ถูกใช้ในการกำหนดจำนวนครั้งการวนรอบของคำสั่ง repeat

```
to flash :n
  repeat :n [a, onfor 5 wait 5]
end
```

เมื่อเรียกใช้ procedure นี้จะต้องตามด้วยค่าตัวเลขหนึ่งค่าเสมอ เช่น flash 5, flash 10, flash 20, ฯลฯ

procedure สามารถรับค่ากี่ค่าก็ได้ cricket logo ไม่ได้จำกัดไว้ แต่ในทางปฏิบัติบริมาณหน่วยความจำที่เหลืออยู่ของ cricket จะเป็นตัวจำกัด

การส่งค่า (Outputs)

Procedure สามารถส่งค่ากลับได้โดยใช้คำสั่ง output เมื่อ procedure เรียกใช้คำสั่งดังกล่าวแล้ว มันจะจบการทำงานทันที ตัวอย่างต่อไปนี้แสดง procedure ชื่อ detect ซึ่งจะส่งค่า 0, 1, หรือ 2 ขึ้นอยู่กับค่าของเซ็นเซอร์ A

```
global [temp]

to detect
  settemp sensora
  if temp < 30 [output 1]
  if temp < 50 [output 2]
  output 3
end
```

ในตัวอย่างนี้มีการสร้างตัวแปร global ชื่อ temp ซึ่งถูกใช้ในการเก็บค่าของเซ็นเซอร์ A ถ้าค่าเซ็นเซอร์นี้น้อยกว่า 30 procedure จะส่งค่า 1 แต่ถ้าค่าเซ็นเซอร์มากกว่า 30 คำสั่งถัดไปจะทำงาน ซึ่งจะทดสอบว่าถ้าค่าดังกล่าวน้อยกว่า 50 procedure จะส่งค่า 2 ท้ายที่สุดถ้าค่าเซ็นเซอร์ไม่น้อยกว่า 50 procedure จะส่งค่า 3

ข้อควรระวัง ถ้าตัดสินใจใช้คำสั่ง output แล้ว จะต้องตรวจสอบให้แน่ใจเสมอว่า procedure นั้นจะมีการส่งค่าไม่ว่าในกรณีใดๆ (นั่นคือ procedure จะส่งค่าบ้างไม่ส่งค่าบ้างไม่ได้) การทำงานของ cricket จะล้มเหลวทันทีถ้า procedure นั้นจบการทำงานโดยไม่มีการส่งค่า

ตัวแปร Global

การสร้างตัวแปร Global จะทำโดยใช้คำสั่ง global [*รายชื่อตัวแปร*] ณ จุดเริ่มต้นของโปรแกรม (บรรทัดที่ 1) เช่น

global [cats dogs]

จะสร้างตัวแปรสองตัวชื่อ cats และ dogs นอกจากนั้นแล้ว cricket logo จะเพิ่มคำสั่ง setcats กับ setdogs เข้าไปในระบบ เพื่อใช้ในการกำหนดค่าให้กับตัวแปรนั้นๆ เช่น

setcats 3

จะกำหนดค่า 3 เข้าไปในตัวแปร cats

setcats cats + 1

จะเพิ่มค่าตัวแปร catsขึ้น 1

ตัวแปร Global จะถูกจัดเก็บในหน่วยความจำ RAM ซึ่งจะต้องมีไฟเลี้ยงอยู่เสมอ ดังนั้นค่าตัวแปรจะสูญหายถ้าปิด cricket ในกรณีที่ต้องการให้ข้อมูลคงอยู่แม้ไม่มีไฟเลี้ยง ให้ใช้คำสั่งการ "บันทึก" และ "เรียกคืน" ค่า หรือเก็บข้อมูลใน global array หน่วยความจำเหล่านี้ไม่จำเป็นต้องมีไฟเลี้ยง

Global Arrays

คำสั่งที่ใช้สร้าง Global arrays คือคำสั่ง array ซึ่งจะต้องอยู่ที่จุดเริ่มต้นของโปรแกรม เช่น

array [clicks 50 clacks 25]

จะสร้าง array สองตัว ชื่อว่า clicks และ clacks ซึ่งสามารถเก็บค่าได้ 50 และ 25 ค่าตามลำดับ

คำสั่งที่ใช้ในการอ่านค่าใน array คือคำสั่ง aget ส่วนการบันทึกค่าลงใน array จะใช้คำสั่ง aset

aget ชื่อ ดรรชนี	อ่านค่าจากตำแหน่ง "ดรรชนี" ใน array ชื่อที่กำหนด
aset ชื่อ ดรรชนี ค่า	บันทึก "ค่า" ลงไปในตำแหน่ง "ดรรชนี" ใน array ชื่อ ที่กำหนด

ยกตัวอย่างเช่น aset clicks 31 1000 บันทึกค่า 1000 ลงไปในตำแหน่งที่ 31 ของ array ชื่อ clicks และ send aget clicks 31 จำทำการส่งค่าจากตำแหน่งที่ 31 ของ array ชื่อ clicks ออกไปทางช่องสัญญาณ infrared

ค่าที่บันทึกลงใน array จะไม่สูญหายแม้เมื่อ cricket ไม่มีไฟเลี้ยง (เช่น ถูกปิดหรือถ่านหมด)

cricket logo ไม่มีการตรวจสอบขอบเขตการบันทึกของมูลของ array ข้อมูลที่บันทึกเกินขอบเขต (เช่นถ้าบันทึกในตำแหน่งที่ 51 ของ array ชื่อ clicks ข้างต้น) อาจสร้างความเสียหายให้กับข้อมูลหรือโปรแกรมได้

การบันทึกและเรียกคืนข้อมูล (Data Recording and Playback)

cricket logo มี global array ขนาด 1000 ช่อง อยู่หนึ่งตัวซึ่งสามารถใช้งานกับคำสั่งต่อไปนี้ได้

setdp ตำแหน่ง	ตั้งค่าตัวชี้ตำแหน่ง
record ค่า	บันทึก "ค่า" ลงไปในตำแหน่งปัจจุบัน และเลื่อนตัวชี้ให้ไปอยู่ในตำแหน่งถัดไป
recall	เรียกคืนค่าในตำแหน่งปัจจุบัน และเลื่อนตัวชี้ให้ไปอยู่ในตำแหน่งถัดไป

ตัวอย่าง procedure ชื่อ take-data ต่อไปนี้จะบันทึกค่าเซ็นเซอร์ A ทุกๆ หนึ่งวินาที

```
to take-data
  setdp 0
  repeat 1000 [record sensora wait 10]
end
```

prcedure ชื่อ send-data ต่อไปนี้จะเรียกคืนค่าข้อมูลและส่งออกไปยังช่องสัญญาณ infrared ทุกๆ 0.5 วินาที

```
to send-data
  setdp 0
  repeat 1000 [send recall wait 5]
end
```

ข้อมูลที่ส่งออกไปยังช่องสัญญาณ infrared ข้างต้น จะไปปรากฏอยู่บนหน้าจอของ Cricket logo บนคอมพิวเตอร์ ท่านสามารถใช้โปรแกรม Logo Graph (เป็นโปรแกรมแยก

ต่างหากจาก cricket logo) ในการ ดึงข้อมูล, สร้างกร๊าฟ และ วิเคราะห์ข้อมูลจาก cricket ได้

หมายเหตุ cricket logo ไม่มีการตรวจสอบว่าการบันทึกข้อมูลเกินขอบเขตหรือไม่ (มากกว่า 1000) ซึ่งอาจส่งผลให้เกิดความผิดพลาดกับข้อมูลหรือโปรแกรมส่วนอื่น

Recursion

Cricket สนับสนุนการทำ recursion ซึ่งหมายถึง procedures ที่มีตัวมันเองเป็นส่วนหนึ่งของชุดคำสั่งใน procedure นั้นๆ

ยกตัวอย่างเช่น เมื่อพิจารณาความหมายทางคณิตศาสตร์ของฟังก์ชัน Factorial ค่า factorial n (หรือ n!) จะมีค่าเท่ากับ n คูณด้วย ค่า factorial ของ (n-1) โดยค่า factorial ของ 1 คือ 1 ค่าจำกัดความนี้สามารถเขียนเป็น procedure ใน cricket logo ได้ดังต่อไปนี้

```
to fact :n
  if :n = 1 [output 1]
  output n * fact :n - 1
end
```

ถ้าจะทดสอบ ให้ download procedure นี้ แล้วลองพิมพ์คำสั่งต่อไปนี้ในศูนย์คำสั่ง (command center)

```
repeat fact 3 [beep wait 1]
```

cricket ควรจะส่งเสียง 6 ครั้ง (เพราะค่า factorial ของ 3 คือ 6)

เนื่องจากหน่วยความจำที่มีในการจัดการกับ recursion บนตัว cricket มีน้อย การใช้งาน recursion จะถูกจำกัดไว้ที่ประมาณ 6 ระดับ ถ้าหน่วยความจำนี้ล้น cricket จะหยุดทำงานและส่งเสียงห้าครั้ง

Tail recursion เป็น recursion ชนิดพิเศษ โดยหมายถึง procedure ที่เรียกตัวมันเองเป็นคำสั่งสุดท้าย cricket logo จะทำการแปลง recursion แบบนี้ให้ให้เป็นการวนรอบ (โดยใช้คำสั่ง goto) ซึ่งจะกำจัดปัญหาเรื่องหน่วยความจำที่กล่าวไปข้างต้น

ตัวอย่างต่อไปนี้แสดงการใช้งาน tail recursion ในรูปแบบที่ถูกต้อง และจะส่งผลให้เกิดการทำงานวนรอบไม่มีที่สิ้นสุด

```
to beep-forever
  beep
  wait 1
  beep-forever
end
```

Multi-Tasking

Multi-Tasking คือการทำงานหลายๆ คำสั่งไปพร้อมๆ กัน cricket สนับสนุนการทำงานแบบนี้ (แม้จะค่อนข้างจำกัดก็ตาม) ตัว cricket จะมี background task ที่คอยตรวจสอบเงื่อนไขที่ผู้ใช้กำหนด เมื่อเงื่อนไขนั้นเป็นจริง cricket จะหยุด procedure ใดๆ ที่ทำงานอยู่ และเปลี่ยนไปทำการประมวลผลคำสั่งพิเศษที่ผู้ใช้กำหนด หลังจากนั้น cricket จึงจะกลับเข้าสู่การทำงานปกติ จนกว่าเงื่อนไขจะเป็นจริงอีกครั้ง

คำสั่ง when เป็นตัวที่ใช้กำหนดเงื่อนไขที่ cricket จะคอยตรวจสอบ และเป็นตัวกำหนดด้วยว่า cricket จะทำอะไรเมื่อเงื่อนไขเป็นจริง ยกตัวอย่างเช่น

when [switcha] [beep] loop [a, onfor 5 rd]

(คำสั่งนี้สามารถพิมพ์เข้าไปในศูนย์คำสั่งเพื่อทำการทดสอบได้)

ในตัวอย่างนี้ cricket จะหมุนมอร์เตอร์ A กลับไปกลับมาทุกๆ 0.5 วินาที แต่ในขณะเดียวกัน background task จะคอยตรวจค่าของ switcha เมื่อใดก็ตามที่ค่านี้เป็นจริง (เช่น เมื่อปุ่มที่ต่ออยู่กับเซ็นเซอร์ A ถูกกด) cricket จะส่งเสียง beep นี่เป็นการแสดงให้เห็นว่า cricket ทำงานสองอย่างไปพร้อมๆ กันได้

ข้อควรสังเกตเกี่ยวกับตัวอย่างข้างต้น:

- คำสั่ง when จะต้องมาก่อนคำสั่ง loop ถ้าไม่เช่นนั้นโปรแกรมจะติดอยู่ที่นั่นและไม่มีวันไปถึงคำสั่ง when
- คำสั่ง when เพียงกำหนดเงื่อนไขให้กับ background task เท่านั้น เมื่อกำหนดแล้วมันจะข้ามไปทำงานคำสั่งถัดไปทันที (ในตัวอย่างนี้คือคำสั่ง loop) คำสั่ง when ไม่จำเป็นต้องอยู่ใน loop และเรียกใช้งานเพียงครั้งเดียวก็เพียงพอแล้ว
- คำสั่งที่ควบคู่กับเงื่อนไขใน when จะทำงานหนึ่งรอบสำหรับแต่ละครั้งที่เงื่อนไขเป็นจริง ในตัวอย่างข้างต้นเมื่อปุ่มถูกกด cricket จะส่งเสียง beep เพียงครั้งเดียว และจะไม่ส่งเสียงอีกจนกว่าเงื่อนไขจะเปลี่ยนสถานะจากเท็จเป็นจริงอีกครั้ง ดังนั้น cricket จะส่งเสียงอีกก็ต่อเมื่อปล่อยและกดปุ่มอีกครั้งเท่านั้น (cricket จะส่งเสียงเพียงครั้งเดียวถ้ากดคาไว้)
- โปรดสังเกตว่าทั้งเงื่อนไขและชุดทำสั่งของ when จะต้องอยู่ในวงเล็บเหลี่ยม
- cricket จะตรวจสอบเงื่อนไขได้หนึ่งเงื่อนไขเท่านั้น ถ้ามีการเรียกใช้คำสั่ง when มากกว่าหนึ่งครั้ง เงื่อนไขของคำสั่ง when ล่าสุดจะลบทับตัวก่อนหน้าเสมอ
- ถ้าต้องการยกเลิกการทำงานของคำสั่ง when ให้ใช้คำสั่ง whenoff.

การสื่อสารข้อมูลทางช่องสัญญาณ Infrared

ภาพรวม

cricket สามารถส่งและรับข้อมูลหากันผ่านทางช่องสัญญาณ infrared ได้โดยใช้คำสั่ง send และ ir ตามลำดับ คำสั่ง ir จะรายงานค่าที่ได้รับล่าสุด นอกจากนี้ยังมีคำสั่ง newir? ซึ่งจะรายงานค่าจริงถ้า cricket ได้รับรับมูลตัวใหม่เข้ามาแต่ยังไม่ได้ถูกนำไปใช้

ลองพิจารณาตัวอย่างต่อไปนี้ procedure ชื่อ sender จะทำการส่งข้อมูลไปยัง cricket ตัวที่สอง, โดยค่าที่ส่งจะเป็นค่าสุ่มระหว่าง 0 ถึง 2

```
to sender
  send random % 3
  beep
  wait 30
  sender
end
```

คำสั่ง random % 3 จะสร้างค่า 0, 1, หรือ 2 ซึ่งเป็นผลจากการใช้เครื่องหมาย "หารเอาเศษ" ค่าที่ได้จะถูกส่งโดยคำสั่ง send หลังจากนั้น ก็จะส่งเสียง beeps และรอ 3 วินาทีก่อนที่จะวนสร้างและส่งค่าออกไปอีกครั้ง

ใน cricket ตัวที่สอง procedure ชื่อ doit จะทำการรับค่าที่ cricket ตั้งแรกส่งออกมา แล้วจะเปิดมอร์เตอร์ A, มอร์เตอร์ B หรือทั้งคู่ ขึ้นอยู่กับค่าที่มันได้รับ

```
to doit
  waituntil [newir?]
  if ir = 0 [a, onfor 10]
  if ir = 1 [b, onfor 10]
  if ir = 2 [ab, onfor 10]
  doit
end
```

หมายเหตุ

cricket ใช้ค่า 128 ถึง 134 สำหรับการทำงานระดับต่ำระหว่าง cricket ดังนั้นพยายามอย่าส่งค่าตัวเลขเหล่านี้ ค่าเหล่านี้อาจทำให้หน่วยความจำของ cricket บางตัว (ตัวที่เปิดอยู่แต่ไม่ได้ทำงานอะไร) ถูกเขียนทับและส่งผลเสียหายได้

ปุ่มบนตัว cricket

เมื่อกดปุ่มในขณะที่ cricket ไม่ได้ทำงานอะไร จะทำให้มันเริ่มต้นประมวลผล procedure ที่ระบุไว้ใน remote-start บรรทัดที่หนึ่งในหน้าจอของ cricket logo

ถ้ากดปุ่มในขณะที่ cricket กำลังทำงานจะส่งผลให้มันหยุดการทำงานปัจจุบันทันที

Last modified: Fri Nov 23, 2001 by Arnan (Roger) Sipitakiat