Guida non ufficiale ai comandi per gli script di AstroArt

Premessa

Da quando utilizzo la camera CCD per le mie osservazioni astronomiche il mio software di riferimento per la ripresa e l'elaborazione delle immagini è sempre stato AstroArt. Naturalmente nel corso degli anni mi sono trovato a maneggiare anche altri software di questo tipo, alcuni sicuramente molto validi, e sebbene la mia limitata esperienza con essi non mi permetta di sbilanciarmi in valutazioni comparative, cosa che in ogni caso ben mi guarderei dal fare, ritengo per me insuperabile l'immediatezza con la quale l'utente di AstroArt riesce ad utilizzare praticamente fin da subito questo software. Un interfaccia utente molto amicale, unita al rigore scientifico degli algoritmi usati ha fatto di AstroArt uno dei software di punta in ambito astronomico amatoriale. L'avvento poi con la versione 3.0 e successive dei comandi di script ha ulteriormente esteso le sue potenzialità, tanto che risulta possibile (se provvisti dell'hardware necessario) effettuare procedure automatizzate di puntamento e ripresa che agevolano moltissimo lo svolgimento delle sessioni osservative. Chiunque abbia avuto un esperienza anche minima di programmazione sa quanto sia difficile e tedioso stilare, ma soprattutto tenere aggiornato il manuale utente. Di conseguenza accade che la manualistica disponibile non sia sempre aggiornata rispetto all'evoluzione del software, ma anche che alcune funzionalità non sempre vengano spiegate in maniera sufficientemente estesa per l'utilizzatore medio.

Questa guida cerca di illustrare alcuni comandi utilizzabili negli script di AstroArt che, nel manuale utente sono scarsamente o magari per nulla documentati. Come recita il titolo questa è una guida non ufficiale e sebbene in alcune sue parti prenda a modello il manuale utente originale essa non si propone in sostituzione ad esso, quanto piuttosto come una informale integrazione per quanto riguarda la parte dei comandi di scripting di questo magnifico software che è AstroArt.

I comandi descritti sono presenti nella versione 4.0 di AstroArt, GUI 3.8 e successive. Ogni eventuale inesattezza ed errore che venisse ad evidenziarsi sulla presente guida sono da imputare esclusivamente al sottoscritto e ne chiedo preventivamente scusa.

Gli Scripts

Uno script consiste in una lista di comandi eseguiti in sequenza. Attraverso gli scripts è possibile come abbiamo già detto automatizzare molte procedure osservative quali riprese per ricerca automatica di asteroidi e supernovae, riprese fotometriche, imaging in tri-o quadricromia etc. Tutto ciò è possibile perché attraverso AstroArt è possibile comandare sia la CCD che la ruota porta filtri ed il telescopio (se questo è a puntamento assistito).
Il linguaggio di script di AstroArt, denominato anche "ABasic" è una sorta di dialetto BASIC con una sintassi molto simile a quella di molti tipi di BASIC, (GWBasic™,QuickBasic™, Visual Basic™, VBScript™, etc.)

Esempio di script (tratto dal manuale di AstroArt e testabile con il simulatore CCD presente in AstroArt)

```
Camera.Start(10)
Camera.Wait
Image.Save("C:\sample.fit")
```

La prima riga di comando fa partire un esposizione di 10 secondi, la seconda riga comanda di attendere il termine dell'esposizione prima di proseguire con lo script. Infine la terza riga di comando salva l'immagine appena acquisita nel disco c:\ denominandola "sample.fit".

Un altro esempio di script sempre tratto dal manuale di AstroArt: la ripresa di 50 immagini per la ricerca di supernovae.

```
for i = 1 to 50
ra = Telescope.List.Ra(i)
de = Telescope.List.Dec(i)
name$ = Telescope.List.Name$(i)
Telescope.Goto(ra,de)
Telescope.Wait
Camera.Start(60)
Camera.Wait
Image.Rename(name$ + ".fit")
Image.save(name$ + ".fit")
next i
```

In questo semplice script le coordinate e il nome delle galassie vengono direttamente recuperati dalla lista preventivamente caricate nella Telescope Window. Per ciascuna galassia lo script provvede a puntare il telescopio verso di essa, far partire l'esposizione e salvare l'immagine acquisita.

Variabili e funzioni

L'ABasic supporta due tipi di variabili, variabili numeriche e variabili stringa. Una variabile numerica contine un numero, mentre una variabile stringa contiene una stringa alfanumerica.

Variabili numeriche

Esse contengono un numero. Tale numero è internamente rappresentato da un valore a virgola mobile con doppia precisione (64 bit, 15 digits).

Variabili stringa

Una variabile stringa contiene testo alfanumerico. Questo testo potrebbe essere rappresentato da una sola o più righe di testo. La dimensione massima di una variabile stringa è di 64 MByte.

Esempio:

```
a$ = "Hello"
b$ = a$ + "World"
```

La variabile b$ ora è composta dalla stringa "HelloWorld"

Un singolo carattere di una stringa può essere letto usando le parentesi quadre, in tal modo a$[1] restituisce come risultato "H" e a$[2] restituisce "e" e cosi via.
Se il numero indice indicato nelle parentesi quadre eccede la lunghezza della stringa si riparte dall'inizio, pertanto a$[6] restituisce ancora "H".

Una singola riga di una stringa multi linea può essere letta usando parentesi graffe

Esempio:

Se la variabile a$ contiene il seguente testo distribuito su tre linee

```
"Questo testo
è distribuito
su tre linee"
```

In tal caso a${2} restituisce la parte di stringa distribuita sulla seconda linea "è distribuito".
La funzione count(a$) restituisce su quante linee è contenuta una stringa multi linea.

Parole riservate.

Queste parole fanno parte del linguaggio di ABasic, pertanto normalmente esse non devono venire utilizzate come argomento nelle variabili stringa. Esse sono:

IF	MOD	WHILE	CLS
THEN	REM	ENDWHILE	
ELSE	FOR	GOTO	
ENDIF	NEXT	GOSUB	
OR	STEP	PRINT	
AND	BREAK	INPUT	
NOT	CONTINUE	END	

Proviamo ora a descriverle in maniera più dettagliata.

Funzioni cicliche: FOR, NEXT, STEP, BREAK, CONTINUE, WHILE, ENDWHILE.

L'ABasic supporta due tipi di istruzioni cicliche: l'istruzione for-next e l'istruzione while-endwhile.

La sintassi completa per l'istruzione ciclica FOR-NEXT è la seguente:

FOR <variabile> = <espressione> TO <espressione> [STEP <costante numerica>]
...
...
NEXT <variabile>

L'istruzione for-next da avvio ad una reiterazione, ovvero ad una ripetizione per un certo numero di volte delle istruzioni contenute tra I due comandi: For (inizio ciclo) e Next (fine ciclo). Un esempio semplice di ciclo for-next è il seguente dove vengono scritti a schermo i numeri da 1 a 10, "a" è la variabile di controllo.

```
for a = 1 to 10
print a
next a
```

L'istruzione BREAK consente di uscire da un ciclo, nel seguente esempio il ciclo viene interrotto qundo il valore della variabile di controllo 'a' diventa più grande di 5.

```
for a = 1 to 10
print a
if a>5 then break
next a
```

L'istruzione CONTINUE si usa all'interno di un ciclo FOR-NEXT e agisce in maniera analoga all' istruzione NEXT, di conseguenza da inizio immediatamente ad una nuova iterazione. Ad esempio:

```
for a = 1 to 10
print a
if a > 5 then continue
print "Test"
next a
```

Infine l'istruzione STEP si usa all'inzio di un ciclo FOR-NEXT per determinare la progressione della variabile di controllo. Ad esempio:

```
for a = 1 to 10 step 2
print a
next a
```

in questo modo la variabile di controllo ‘a’ salta tutti I numeri pari da 1 a 10. Alla funzione STEP possiamo anche fornire valori negativi, ad esempio:

```
for a = 10 to 1 step -1
print a
next a
```

Invece l’istruzione WHILE-ENDWHILE valuta la condizione all’inizio del ciclo. Se la condizione è falsa allora il ciclo si interrompe e l’esecuzione continua dopo l’istruzione ENDWHILE.

Ad esempio:

```
a = 1
while a <= 10
print a
a = a+1
endwhile
print “ciclo finito!”
```

Dato che il commando WHILE valuta la condizone all’inizio del ciclo le istruzioni contnute all’interno del ciclo potrebbero anche non venire mai eseguite.
Le istruzioni BREAK e CONTINUE possono essere usate in un ciclo WHILE-ENDWHILE in maniera del tutto simile a quanto già visto per il ciclo FOR-NEXT.

Funzioni condizionali: IF, THEN, ELSE, ENDIF, OR, AND, NOT

L’istruzione IF-THEN-ENDIF valuta un espressione logica e determina il flusso del programma secondo il risultato di tale espressione.

Alcuni esempi di espressioni logiche:

```
a > 5 and b$ = "astro"
a >= 3 or not (b = 5)
```

Gli operatori logici e matematici utilizzati nelle espressioni logiche hanno una loro scala di precedenze quando devono venire scritti nel listato di istruzioni, di seguito si da la scala di precedenza di tali operatori andando dalla precedenza più alta verso la precedenza più bassa.

Precedenza degli operatori:

Alta precedenza ( ), < , > , <= , >= , <> , = , NOT, AND, OR Bassa precedenza

Sintassi estesa del comando IF-THEN-ENDIF

IF <espressione logica> THEN
...
...
[ELSE]
...
...
ENDIF

Esempio di condizione IF-THEN-ENDIF

```
for i = 10 to -10 step -1
if i>0 then
print "valori positivi"
endif
if i=0 then print "zero"
if i<0 then
print "valori negativi"
endif
```

Sintassi compatta del comando IF-THEN

IF <espressione logica> THEN <istruzione> [ELSE]< istruzione>

Esempio di comando compatto IF-THEN

```
for a = 1 to 10
if a <= 5 then print "-" else print "+"
next a
```

Altre funzioni

Funzione	Dettagli	Esempi
REM	Qualunque scrittura preceduta dal comando REM viene ignorata durante l'esecuzione del programma. Questa funzione permette l'inserimento di annotazioni all'interno della lista comandi. Anche'inserimento di un apice "'" funziona in maniera identica a REM	**REM annotazioni** **'annotazioni** **Output:**
GOTO n	Salta con l'esecuzione del programma alla linea **n** ignorando tutto ciò che si trova tra il comando **GOTO n** e la linea **n**.	**for i = 1 to 10** **if i = 5 then goto 10** **print i** **next i** **10 print "sono saltato alla linea 10"** **Output:** **1** **2** **3** **4** **sono saltato alla linea 10**

GOSUB n **RETURN**	Salta con l'esecuzione del programma alla linea **n** ignorando tutto ciò che si trova tra il comando **GOSUB n** e la linea **n**, ma una volta incontrato il comando **RETURN** ritorna alla linea immediatamente sottostante il comando **GOSUB n**.	**for i = 1 to 10** **if i = 5 then gosub 10** **print i** **next i** **END** **10 print "sono saltato alla linea 10"** **print "ma ritorno subito da dove sono partito"** **RETURN** **Output:** **1** **2** **3** **4** **sono saltato alla linea 10** **ma ritorno subito da dove sono partito** **5** **6** **7** **8** **9** **10**
PRINT "s"	Scrive a video un testo "**s**", una variabile numerica **n** o una stringa alfanumerica **s$**. Testo e variabili possono anche venire concatenate sulla medesima riga di comando.	**a=4** **b$="Versione"** **c$="eseguito con"** **print "script AstroArt "** **print c$+" Astroart "+b$;a** **Output:** **script AstroArt** **eseguito con Astroart Versione 4**
INPUT	Permette l'inserimento da parte dell'utente di variabili numeriche o variabili stringa.	**input "inserisci un numero: ",n** **input "inserisci una stringa: ",s$** **print n** **print s$** **Output:** **il programma mostra in successione due finestre in cui inserire i dati. Dati che compariranno scritti nella finestra di output.**
END	Termina l'esecuzione di uno script.	**print "il programma terminerà alla riga 3"** **print "riga 1"** **print "riga 2"** **print "riga 3"** **END** **print "riga 4"** **Output:** **riga 1** **riga 2** **riga 3**
CLS	Pulisce la finestra di Output.	**for i = 1 to 10** **print "abcdefghilmo"** **next i** **message ("premi 'OK' per pulire la finestra di output")** **CLS** **Output:**

Funzioni numeriche.

Funzione	Dettagli	Esempi
pi()	Funzione a valore fisso che restituisce il valore del pi greco	**Print pi()** **Output:** 3.141592654
sin(n)	Calcola il seno di un angolo **n** espresso in radianti. Se **n** esprime l'angolo in gradi anziché in radianti utilizzare le seguenti procedure: **sin(n*pi()/180)** oppure **sin(degtorad(n))**	**Print sin(90)** **Output:** 0.8939966636
cos(n)	Calcola il coseno di un angolo **n** espresso in radianti. Se **n** esprime l'angolo in gradi anziché in radianti utilizzare le seguenti procedure: **cos(n*pi()/180)** oppure **cos(degtorad(n))**	**Print Cos(90)** **Output:** -0.4480736161
tan(n)	Calcola la tangente di un angolo **n** espresso in radianti. Se **n** esprime l'angolo in gradi anziché in radianti utilizzare le seguenti procedure: **tan(n*pi()/180)** oppure **tan(degtorad(n))**	**Print tan(50)** **Output:** -0.271900612
exp(n)	Calcola il valore del numero di Nepero elevato alla **n**, ovvero e^n	**Print exp(10)** **Output:** 22026.46579
ln(n)	Calcola il valore del logaritmo in base e (logaritmo naturale) di **n**	**Print ln(10)** **Output:** 22026.46579
log10(n)	Calcola il valore del logaritmo in base 10 di **n**	**Print log10(100)** **Output:** 2
log2(n)	Calcola il valore del logaritmo in base 2 di **n**	**Print log2(50)** **Output:** 5.64385619
sqr(n)	Calcola il valore della radice quadrata di **n**	**Print sqr(16)** **Output:** 4
abs(n)	Calcola il valore assoluto (detto anche modulo)di un numero **n**	**Print abs(15)** **Print abs(-15)** **Output:** 15 15

rnd(n)	Restituisce un numero randomizzato tra **0** ed **n**	**For i = 1 to 5** **Print rnd(10)** **Next i** **Output:** 0.9364372841 6.289201556 2.800253921 8.77184656 1.612342733
sgn(n)	Restituisce il segno di un numero **n** secondo lo schema: **sgn(n) = -1 se n < 0,** **sgn(n) = 0 se n = 0,** **sgn(n) = 1 se n > 0.**	**Print sgn(-12.345)** **Print sgn(0)** **Print sgn(12.345)** **Output:** -1 0 1
fix(n)	Restituisce la parte intera di un numero **n**	**Print fix(8.771845)** **Output:** 8
int(n)	Restituisce la parte intera di un numero **n**	**Print int(8.771845)** **Output:** 8
round(n[,n1])	Arrotonda un numero **n** alla **n1** esima cifra decimale. N.B. Se la cifra viene omessa l'arrotondamento avviene per zero cifre decimali.	**Print round(8.771845,3)** **Print round(8.771845)** **Output:** 8.772 9
frac(n)	Restituisce la parte frazionaria di un numero **n**	**Print frac(10.45678)** **Output:** 0.45678
asin(n)	Calcola l'arcoseno in radianti di un numero **n**. Funzione valida nell'intervallo (1,-1)Per valori esterni a quest'intervallo viene restituito il valore nullo "NAN". Per convertire da radianti a gradi: **asin(n)*180/pi()** oppure **radtodeg(asin(n))**	**Print asin(1),"radianti"** **Print asin(n)*180/pi(),"Gradi"** **Output:** 1.570796327 radianti 90 Gradi
acos(n)	Calcola l'arcocoseno in radianti di un numero **n**. Funzione valida nell'intervallo (1,-1)Per valori esterni a quest'intervallo viene restituito il valore nullo "NAN". Per convertire da radianti a gradi: **acos(n)*180/pi()** oppure **radtodeg(acos(n))**	**Print acos(1),"radianti"** **Print acos(n)*180/pi(),"Gradi"** **Output:** 0 radianti 0 Gradi
atan(n)	Calcola l'arcotangente in radianti di un numero **n**. Per convertire da radianti a gradi: **atan(n)*180/pi()** oppure **radtodeg(atan(n))**	**Print atan(1),"radianti"** **Print atan(1)*180/pi(),"Gradi"** **Output:** 0.7853981634 radianti 45 Gradi

atan2(nx,ny)	Calcola l'arcotangente2 in radianti di un punto avente coordinate (**nx, ny**). Per convertire da radianti a gradi: **atan2(nx,ny)*180/pi()** oppure **radtodeg(atan2(nx,ny))**	**print atan2(40,50)** **Output:** **0.6747409422**
sinh(n)	Calcola il seno iperbolico di un numero **n** espresso in radianti Per convertire da radianti a gradi: **sinh(n*pi()/180)** oppure **sinh(degtorad(n))**	**print sinh(10)** **Output:** **11013.23287**
cosh(n)	Calcola il coseno iperbolico di un numero **n** espresso in radianti. Per convertire da radianti a gradi: **cosh(n*pi()/180)** oppure **cosh(degtorad(n))**	**print cosh(10)** **Output:** **11013.23292**
tanh(n)	Calcola la tangente iperbolica di un numero **n** espresso in radianti. Per convertire da radianti a gradi: **tanh(n*pi()/180)** oppure **tanh(degtorad(n))**	**print tanh(10)** **Output:** **0.9999999959**
asinh(n)	Calcola l'arcoseno iperbolico espresso in radianti di un numero **n.** Per convertire da radianti a gradi: **asinh(n)*180/pi()** oppure **radtodeg(asinh(n))**	**print asinh(100)** **Output:** **5.298342366**
acosh(n)	Calcola l'arcocoseno iperbolico espresso in radianti di un numero **n**. Per convertire da radianti a gradi: **acosh(n)*180/pi()** oppure **radtodeg(acosh(n))**	**print acosh(10)** **Output:** **2.993222846**
atanh(n)	Calcola l'arcotangente iperbolica espressa in radianti di un numero **n**. Funzione valida nell'intervallo (1,-1)Per valori esterni a quest'intervallo viene restituito valore infinito "INF". Per convertire da radianti a gradi: **atanh(n)*180/pi()** oppure **radtodeg(atanh(n))**	**print atanh(0.5)** **Output:** **0.5493061443**
degtorad(n)	Converte un numero **n** da gradi a radianti	**n=57.29577951** **print "il valore in gradi pari a: ";n** **print "equivale a radianti: "; degtorad(n)** **Output:** **il valore in gradi pari a: 57.29577951** **equivale a radianti: 0.9999999999**

radtodeg(n)	Converte un numero **n** da radianti a gradi.	**n=1** **print "il valore in radianti pari a: ";n** **print "equivale a gradi: ";** **radtodeg(n)** **Output:** **il valore in radianti pari a: 1** **equivale a gradi: 57.29577951**
modulo(n1,n2)	Calcola l'espressione radq((n1^2)+(n2^2)).	**print modulo(1,5)** **Output:** **5.099019514**
len(s)	Restituisce il numero di caratteri di una stringa (conteggia anche gli spazi tra le parole).	**a=len("viva AstroArt!")** **print "la scritta contiene ";a;" caratteri"** **Output:** **la scritta contiene 14 caratteri**
val(s)	Converte una stringa contenente caratteri numerici nel corrispondente numero.	**a$="1"** **print val(a$)+2** **Output:** **3**
asc(s)	Restituisce il codice ANSI del primo carattere a sinistra della stringa.	**print asc("AstroArt")** **Output:** **65**
pause(n)	Mette in pausa l'esecuzione del programma per un numero **n** di secondi.	**pause(30)** **Output:**
n1 mod n2	Restituisce il resto della divisione **n1/n2**	**print 14 mod 4** **Output:** **2**
count(s)	Restituisce il numero di righe contenuto in una stringa multi linea **s**.	a$="ciao"+crlf$()+"ciao" print a$ print crlf$() print "il numero di righe nella variabile è: ";count(a$) **Output:** **ciao** **ciao** **il numero di righe nella variabile è: 2**
counter()	Teoricamente dovrebbe restituire il tempo in millisecondi trascorso dall'avvio di Windows, tuttavia l'ABasic non sembra riconoscere questa funzione.	

Funzioni Stringa.

Funzione	Dettagli	Esempi
ucase$(s)	Converte in maiuscoli tutti i caratteri di una stringa **s**	**print ucase$(“astroart”)** **Output:** **ASTROART**
lcase$(s)	Converte in minuscoli tutti i caratteri di una stringa **s**	**print lcase$(“ASTROART”)** **Output:** **astroart**
ltrim$(s)	Toglie gli spazi vuoti alla sinistra di una stringa.	**print "senza ltrim$: "+" astroart"** **print "con ltrim$: "+ltrim$(" astroart")** **Output:** **senza ltrim$: astroart** **con ltrim$: astroart**
rtrim$(s)	Toglie gli spazi vuoti alla destra di una stringa.	**print "astroart "+" senza rtrim$:"** **print rtrim$("astroart ")+" con rtrim$:"** **Output:** **astroart senza rtrim$:** **astroart con rtrim$:**
chr$(n)	Restituisce il carattere corrispondente al numero **n** di codice ASCII.	**print chr$(64)** **Output:** **@**
str$(n)	Converte un numero **n** da valore numerico a stringa di caratteri.	**a=234** **a$=str$(a)** **print "a = ";a;" un numero"** **print "a$ = "+a$+" una stringa"** **Output:** **a = 234 è un numero** **a$ = 234 è una stringa**
mid$(s,n1,n2)	Restituisce una sottostringa della stringa **s** che viene tagliata a sinistra partendo dal carattere numero n1 e risulta lunga numero **n2** caratteri	**print mid$("abcdefghilmnopqrstuvz",2,5)** **Output:** **bcdef**
hex$(n)	Converte un numero **n** decimale nella stringa che ne rappresenta il valore in formato esadecimale.	**print hex$(1000)** **Output:** **3E8**
left$(s,n)	Restituisce una sottostringa della stringa **s** che viene tagliata a sinistra partendo dal primo carattere e risulta lunga numero **n** caratteri	**print left$("AstroArt",5)** **Output:** **Astro**
right$(s,n)	Restituisce una sottostringa della stringa **s** che viene tagliata a	**print right$("AstroArt",3)** **Output:**

	destra partendo dall'ultimo carattere e risulta lunga numero **n** caratteri.	Art
ltab$(s,n)	Sposta una stringa a destra di **n**-len(**s**) caratteri rispetto alla stringa **s**. Agisce solo se: **n**-len(**s**)>0	**a$="Astro"** **print "la parola'";a$;"' lunga ";len(a$);" lettere"** **for n=0 to 10** **print ltab$(a$,n)+str$(n)** **next n** **Output:** **la parola' Astro ' lunga 5 lettere** **Astro0** **Astro1** **Astro2** **Astro3** **Astro4** **Astro5** **Astro 6** **Astro 7** **Astro 8** **Astro 9** **Astro 10**
rtab$(s,n)	Sposta la stringa s a destra di **n**-len(**s**) caratteri rispetto all'inizio riga. Agisce solo se: **n**-len(**s**)>0	**a$="Astro"** **print "la parola'";a$;"' lunga ";len(a$);" lettere"** **for n=0 to 10** **print rtab$(a$,n)+str$(n)** **next n** **Output:** **la parola' Astro ' lunga 5 lettere** **Astro0** **Astro1** **Astro2** **Astro3** **Astro4** **Astro5** **Astro6** **Astro7** **Astro8** **Astro9** **Astro10**
format$(n,s)	Sostituisce i caratteri 0 (zero) presenti nella stringa **s** con il valore numerico **n**. La sostituzione avviene da destra verso sinistra. Se il numero di 0 presenti in **s** è inferiore al numero di cifre di **n** le rimanenti cifre verranno visualizzate alla sinistra dell'ultimo 0. Se invece il numero di 0 è superiore al numero di cifre di **n** gli 0 a sinistra dell'ultima cifra di **n** verranno visualizzati come zero.	**print date$(); " la data di oggi"** **aaaammgg$=left$(date$(),4)+mid$(date$(),6,2)+right$(date$(),2)** **print format$(20101006,"Ovvero: anno 0000 mese 00 giorno 00")** **Output:** **2010 10 06 la data di oggi** **Ovvero: anno 2010 mese 10 giorno 06**
time$()	Ritorna una stringa con il valore attuale dell'ora nel formato hh mm ss	**print time$()** **Output:** **09 06 36**

date$()	Ritorna una stringa con il valore attuale della data nel formato aaaa mm gg	**print date$()** **Output:** **2010 10 06**
crlf$()	Funzione equivalente al ritorno a capo, inserisce una riga vuota nella finestra di output.	**a$="AstroArt"** **b$="astronomical software:"** **print a$+" "+b$+" Così compare tutto su una linea e può essere scomodo da leggere..)"** **print crlf$()** **print a$+crlf$()+b$+crlf$()+"Così invece le scritte"+crlf$()+"sono distribuite su più linee "+crlf$()+"(più comodo da leggere no?!..)"** **Output:** **AstroArt astronomical software: Così compare tutto su una linea e può essere scomodo da leggere..)** **AstroArt** **astronomical software:** **Così invece le scritte** **sono distribuite su più linee** **(più comodo da leggere no?!..)**
opentext$(s)	Apre un file di testo di nome **s**. N.B. Nella stringa **s** deve comparire anche l'estensione del file ed eventualmente il path	**file$=opentext$("C:\WINDOWS\system32\rsvpcnts.h")** **print file$** **Output:** **/*++** **Copyright (c) 1996 Microsoft Corporation** **#define RSVPOBJ 0** **#define RSVP_INTERFACES 2** **#define RSVP_NET_SOCKETS 4** **#define RSVP_TIMERS 6** **#define API_SESSIONS 8** **#define API_CLIENTS 10** **Etc etc etc…….**
savetext$(s1,s2)	Scrive la stringa **s1** su un file di testo **s2** N.B. Nella stringa **s2** deve comparire anche l'eventuale path nonché l'eventuale estensione del file. Attenzione: questo comando sovrascrive un eventuale altro file col medesimo nome presente nella stessa cartella.	**print savetext$("AstroArt, the best software for astronomical imaging","c:\AA.txt")** **print "Nella directory c:\ dovrebbe esser comparso" +crlf$()+"un file di testo denominato 'AA.txt' " +crlf$()+"contenente al suo interno la frase:" +crlf$()+"'AstroArt, the best software for" +crlf$()+"astronomical imaging'"** **Output:** **Nella directory c:\ dovrebbe esser comparso** **un file di testo denominato 'AA.txt'** **contenente al suo interno la frase:** **'AstroArt, the best software for**

		astronomical imaging'
copytext$(s)	Copia la stringa **s** negli appunti	**print copytext$("AstroArt")** **a$=pastetext$()** **print a$** **Output:** **AstroArt**
pastetext$()	Incolla il contenuto degli appunti sullo schermo di output o in una variabile	**print copytext$("AstroArt")** **a$=pastetext$()** **print a$** **Output:** **AstroArt**

finddir$(s,s)		**input "Percorso directory ",path$**
		input "nome directory da trovare",dir$
		a$=finddir$(path$,dir$)
		print "sto cercando la directory: "+a$
		al$=lcase$(a$)
		dirl$=lcase$(dir$)
		if al$=dirl$ then
		print "directory trovata"
		else
		print "directory NON trovata"
		endif
		
		Output
		directory trovata (se essa esiste nella posizione specificata)
		
		oppure:
		
		directory NON trovata (se essa non esiste nel pc nella posizione specificata)

findfile$(s1,s2)	Ricerca un file **s2** in un percorso **s1**.	**pathltp$="c:\"**
		b$="AA.txt"
		c$= pathltp$+b$
		print savetext$("AstroArt, the best software for astronomical imaging",c$)
		input "nome file? ",obj_name$
		findf$=findfile$(pathltp$,obj_name$+".txt")
		if findf$=(obj_name$+".txt") then
		print "File TROVATO!"
		endif
		if findf$<>(obj_name$+".txt") then
		print "File NON TROVATO!"
		endif
		
		Output:
		File TROVATO! (se nell'imput si è digitato 'AA')
		File NON TROVATO! (se nell'imput si è digitato una qualsiasi stringa diversa da 'AA'scritta in maiuscolo)

message(s)	Mostra un messaggio a video contenente la stringa **s**.	**Message("Hello Milky way")** **Output:** 
ra$(n)	Converte il valore di ascensione retta espresso in decimali dal numero **n** in una stringa indicante il valore di ascensione retta espresso in hh mm ss.s	**alpha=05.345678** **print "Ascensione Retta: "+ra$(alpha)** **Output:** **Ascensione Retta: 05 20 44.4**
dec$(n)	Converte il valore di declinazione espresso in decimali dal numero **n** in una stringa indicante il valore di declinazione espresso in +/-gg pp ss.s	**delta=-12.345678** **print "Declinazione: "+dec$(delta)** **Output:** **Declinazione: -12 20 44**
createdir(s)	Crea una directory con nome e posizione specificata dalla stringa **s**. Se non viene specificato un drive e/o un percorso la directory verrà create dentro la directory corrente.	**createdir("c:\immagini")** **Output:**

Funzioni per CCD , Ruota portafiltri e Telescopio.

Funzione	Dettagli	Esempi
Camera.Start(n[,n1])	Fa partire un esposizione di **n** secondi. il numero opzionale **n1** settato con valore zero viene consente la ripresa di un dark frame di durata **n** secondi.	**Camera.Start(60,0)**
Camera.Wait	Attende la fine dell'esposizone della CCD	
Camera.Exposing	Ritorna il valore "1" se l'esposizione è in corso, altrimenti da valore "0"	
Camera.Binning(n)	Setta il valore di binning **n** a cui far operare la camera. Nel pannello CCD alla pagina denominata "Setting" si trova una lista con i valori di binning utilizzabili per la camera installata.	**Camera.Binning(2)**

Camera.SelectDarkFrame **Camera.EnableDarkFrame(n)**	Selezione l'immagine corrente come immagine di dark ed automaticamente abilita la correzione per le seguenti immagini. Abilita o disabilita la correzione per il dark frame. **n** = 1 correzione abilitata **n** = 0 correzione disabilitata	**Camera.SelectDarkFrame()** **Camera.EnableDarkFrame(0)**
Camera.Stop	Ferma l'esposizione corrente	
Guider.Stop	Ferma la sessione di autoguida corrente	
Guider.Close	Chiude la finestra di autoguida.	
Guider.Select(n)	Seleziona quale camera CCD debba venir usata per l'autoguida: 1= CCD principale, 2= CCD guida, 3= camera secondaria.	**Guider.Select(2)**
Guider.MoveReference([dx, dy])	Cambia le coordinate **x** e **y** della stella di riferimento per effettuare la dittere guide. Se le coordinate **x** e **y** non vengono specificate viene effettuato uno spostamento pesudo randomizzato.	**Guider.MoveReference()** **GuiderMoveReference(-0.3, 0.7)**
Camera.Connect([driver]) **Camera.Disconnect**	Connette il driver CCD ad AstroArt. Disconnette il driver CCD da AstroArt	**Camera.Connect("Simulator")**
Camera.StartAutoguide([x, y])	Avvia una sessione di autoguida, usando la stella guida avente le coordinate di lastra x e y. Se tali coordinate non vengono impostate viene automaticamente acquisita un immagine campione e scelta la miglior stella presente.	**Camera.StartAutoguide()** **x = Image.GetPointX()** **y = Image.GetPointY()** **Camera.StartAutoguide(x,y)**
Camera.StopAutoguide()	Ferma la procedura di autoguida.	

Camera.Autofocus([x,y])	Avvia una sessione di autofocus (richiede che sia installato il plugin 'autofocus' reperibile sul sito di AstroArt). usando la stella guida avente le coordinate di lastra x e y. Se tali coordinate non vengono impostate viene automaticamente scelta la miglior stella presente sull'immagine corrente.	**Camera.Autofocus()** **x = Image.GetPointX()** **y = Image.GetPointY()** **Camera.Autofocus(x,y)**
Focuser.GotoRelative(n)	Muove il focheggiatore avanti o indietro di una quantità specifica **n**.	**Focuser.GotoRelative(-50)**
Focuser.GotoAbsolute(n)	Muove il focheggiatore ad una data coordinata **n** (solo per focheggiatori digitali).	**Focuser.GotoAbsolute(1000)**
Telescope.Goto(ra,dec)	Muove il telescopio alle coordinate equatoriali **ra** e **dec** espresse in formato decimale. (non in hh mm ss e +/-gg pp ss).	**Telescope.Goto(23.45, 44.12)**
Telescope.Wait	Attende che il telescopio abbia completato il puntamento.	
Telescope.Stop	Ferma il movimento di goto del telescopio (non il tracking)	
Telescope.Ra **Telescope.Dec**	Restituisce le coordinate della posizione corrente del telescopio	**x = Telescope.Ra** **y = Telescope.Dec**
Telescope.Pulse(dir$ [,time])	Muove il telescopio di <**time**> secondi verso una direzione **<dir$> ("N","S", "E","W")**. Se <**time**> è negative la direzione viene invertita. Se invece <**time**> viene omesso il telescopio si muoverà finche non riceve un comando **Telescope.stop.**	**Telescope.Pulse("N", 0.5)**
Telescope.Speed(n)	Setta la velocità per i movimenti effettuati da pulsante secondo il seguente schema: (1=guide, 2=center, 3=find, 4=slew)	**Telescope.Speed(4)**
Telescope.List.Open(file$)	Apre un file di testo file$ contenente un elenco di oggetti e coordinate. Vedi capitolo 6.1 del manuale utente.	**Telescope.List.Open(" c:\data\galaxies.txt")**

Telescope.List.Count	Restituisce il numero di oggetto che sono presenti nella object list della finestra del telescopio.	**n = Telescope.List.Count**
Telescope.List.Clear	Cancella la lista di oggetti presenti nella object list della finestra del telescopio.	
Telescope.List.Ra(n) **Telescope.List.Dec(n)**	Restituisce le coordinate dell'oggetto n°**n** presente nella object list della finestra del telescopio.	**x = Telescope.List.Ra(25)**
Telescope.List.Name$(n)	Restituisce il nome dell'oggetto n°**n** presente nella object list della finestra del telescopio.	**a$ =Telescope.List.Name$(42)**
Telescope.Send(s)	Invia una stringa alfanumerica **s** al telescopio attraverso la porta seriale.	**Telescope.Send("#Hc#")**
Wheel.Filters	Restituisce il numero di filtri presenti nella ruota porta filtri.	**n = Wheel.Filters**
Wheel.Goto(n) **Wheel.Goto(s)**	Posiziona la ruota portafiltri su un dato filtro espresso dal suo numero **n** o dalla sua denominazione **s**.	**Wheel.Goto(4)** **Wheel.Goto("R")**
Image.Save(filename$)	Salva l'immagine corrente con il path e nomefile specificati da **filename$**. Se il path viene omesso l'immagine viene salvata nella directory attiva.	**Image.Save("C:\images\ saturn.fit")**
Image.Rename(name$)	Rinomina l'immagine corrente.	**Image.Rename("jupiter.fit")**
Image.Open(filename$)	Apre un immagine dal disco con il path e nomefile specificati da **filename$**. Se il path viene omesso l'immagine viene salvata nella directory attiva.	**Image.Open("C:\moon.tif")**
Image.GetKey$(key$) **Image.GetKey(key$)**	Legge i valori stringa del parametro **key$** dall'header fit. Legge i valori numerici del parametro **key$** dall'header fit.	**a =Image.GetKey("NAXIS")** **a=Image.GetKey("EXPOSURE")**
Image.SetKey(key$,value)	Scrive e se già presente sovrascrive un parametro **key$** dell'header con il valore **value**.	**Image.SetKey("COMMENT","Bad seeing")** **Image.SetKey("JD",34234)**
Image.FlipH	Ribalta orizzontalmente l'immagine corrente. Questa funzione necessita di AstroArt 4.0 + Service Pack 1.	
Image.FlipV	Ribalta verticalmente l'immagine corrente. Questa funzione necessita di AstroArt 4.0 + Service Pack 1.	

Image.Resize(x,y)	Ridimensiona un immagine alle dimensioni orizzontali **x** e verticali **y**. Questa funzione necessita di AstroArt 4.0 + Service Pack 1.	**Image.Resize(320,240)**
Image.BlinkAlign	Allinea l'immagine corrente con la successiva presente sul desktop di Astroart ed esegue il blink. Questa funzione necessita di AstroArt 4.0 + Service Pack 1.	**Image.BlinkAlign**
Image.Close	Chiude l'immagine corrente.	
Image.GetPointX() **Image.GetPointY()**	Restituisce le coordinate del punto, della stella o del rettangolo selezionato sull'immagine corrente.	**x = Image.GetPointX()**
Image.DSS(ra,dec,name$)	Crea una nuova imagine utilizzando l'atlante digitale 'Digital Sky Survey'. Tale immagine sarà centrata sulle coordinate **ra** e **dec** e sarà denominata **name$**. L'utilizzo di questa funzione richiede il plugin DSS.	**Image.DSS(12.034,45.213,"asteroid.fit")**
Output.Save(filename$)	Salva il contenuto del pannello di output sul disco con il path e nomefile specificati da **filename$**. Se il path viene omesso il file viene salvato nella directory attiva.	**Output.Save("C:\Log.txt")**
Output.Copy	Copia il contenuto del pannello di controllo sulla clipboard.	
System.Execute(filename$)	Esegue un nuovo programma con il path e nome file eseguibile specificati da **filename$**.	**System.Execute("C:\Windows\Notepad.exe myfile.txt")**
System.Broadcast(message$, wparam,lparam)	Spedisce un messaggio Windows a tutte le finestre. Questa funzione può essere utilizzata per controllare altri programmi. La funzione è equivalente a: **h = RegisterWindowMessage(message$) SendNotifyMessage(HWND_BROADCAST,h,wparam,lparam).**	

Funzioni non documentate.

Funzione	Dettagli	Esempi
system.shutdown	Chiude AstroArt, ed arresta il computer. Funzione irreversibile da usare con cautela.	