MSXcompilerV4 versione 2025.1 (download)

Creato con Cerberus X

Special thanks to Zeda Thomas for the float32 routines!
Special thanks to Grauw for information about assembly language http://map.grauw.nl
Special thanks to Augusto Ruiz for the WYZPlayer and the WYZTracker to compose music
Special thanks to WYZ (WYZplayer ASM Code and music modules)

Funziona su Windows, Mac OS e Linux. Scrivi il tuo codice in basic (molto simile al basic MSX, ma con alcune differenze) e il compilatore creerà un file assembly (linguaggio macchina) in versione .ROM o .DSK, da provare su qualunque emulatore o su un vero MSX.

Per testare il tuo programma puoi utilizzare qualsiasi emulatore a tua scelta. È necessario prima specificare il percorso relativo all'emulatore nelle impostazioni (su Mac OS è necessario impostare un percorso aggiuntivo perché il file eseguibile si trova all'interno della cartella dell'app dell'emulatore).

Come funziona

Il compilatore legge il codice BASIC e lo traduce in linguaggio macchina, quindi lo traduce in codice binario. È possibile salvare e vedere il file ASM se lo si desidera, selezionando l'opzione relativa nella finestra Impostazioni.

Se il codice binario risultante dalla compilazione è maggiore di 16kb, è necessario dividere il progetto in più di un file, ciascuno con un massimo di 16kb di codice binario, fino a un massimo di 128 file. È possibile effettuare chiamate tra i file con le istruzioni CALLPAGE (funziona come Gosub) e JUMPPAGE (funziona come GOTO)

esempio: CALLPAGE 1,100 --> fa una chiamata alla pagina 1, alla riga 100

esempio: JUMPPAGE 2,start --> fa un salto alla pagina 2, all'etichetta "start"

è possibile scegliere il TARGET della compilazione (CARTUCCIA o DISCO)

A seconda del TARGET (cartuccia o disco), verrà creato un file .ROM di un massimo di 2MB o un file .DSK contenente fino a un massimo di circa 100 files binari di 16kb ciascuno!

E’ inoltre possibile scegliere in che modo verrà gestito il progetto in memoria, cioè se creare un progetto in MEMORY MAPPER oppure in SLOTS

- MEMORY MAPPER: è possibile sfruttare tutta la memoria RAM, tranne le prime 2 pagine da 16kb, la prima è dedicata alle variabili di sistema (0C000h-0FFFFh) e quindi non va modificata, mentre la seconda è usata per memorizzare le variabili del programma (8000h-0BFFFh). Le altre pagine verranno usate per memorizzare il codice macchina del programma, questo nel caso in cui andremo a creare un progetto in versione .DSK.

- SLOTS: il programma verrà memorizzato nelle pagine 1 e 2 della memoria RAM, o ROM nel caso abbiamo scelto come TARGET la cartuccia. Quindi si troverà negli indirizzi (4000h-0BFFFh), per un massimo di 32KB. Le variabili verranno memorizzate a partire dall’indirizzo 0D170h in poi, tenendo presente che lo STACK è inizializzato a 0DB00H.

Organizzazione della memoria

• Memoria contenente il codice binario (4000H-7FFFH) con mapper di memoria per modificare la pagina
• Memoria contenente le variabili (8000H-BFFFH)
• Memoria riservata al compilatore (C000H-D16FH)
• Memoria aggiuntiva disponibile da D170H in poi. (Si noti che il puntatore dello stack è inizializzato su DB00H)
• Stack (DB00H) ma è possibile modificarlo nella finestra Impostazioni

Impostare il target

Il TARGET è il prodotto della compilazione. Dopo aver premuto il tasto F5, il progetto viene compilato e verrà creato un singolo file, che può essere un file disco, o un file cartuccia. Il file avrà quindi un'estensione .dsk oppure .rom.

DISCO

Il disco è un file con estensione .DSK e con lo stesso nome del progetto da cui inizia la compilazione. Ad esempio, se viene compilato il file “esempio.msxproj”, il file del prodotto sarà “esempio.dsk”

Nelle impostazioni del disco è possibile:

- Scegliere se includere o meno il file "autoexec.bas"
- Selezionare il nome dei file contenuti nel disco (max 8 caratteri), esempio "nameapp"
- Selezionare se creare un disco a faccia singola o a faccia doppia
- Selezionare la modalità di creazione del disco:
- Modalità 1: disco normale con tutti i file

• Modalità 2: include solo il file "autoexec.bas" e il file "nameapp.ldr", tutti gli altri file dell'applicazione non saranno visibili.

CARTUCCIA

La cartuccia è un file con estensione .ROM e con lo stesso nome del progetto da cui inizia la compilazione. Ad esempio, se viene compilato il file “esempio.msxproj”, il file prodotto sarà “esempio.rom”
La dimensione del file viene creata automaticamente, a seconda delle dimensioni del progetto, cioè da quante "pagine" è composto.

Impostare l’Emulatore

È possibile selezionare l'emulatore desiderato per testare l'applicazione.

Istruzioni di precompilazione

Le istruzioni di precompilazione sono particolari istruzioni che hanno effetto solo durante la fase di compilazione del progetto. Iniziano tutte con il simbolo #, seguito dal nome dell’istruzione.

Istruzione CheckSystem

Con l’istruzione CheckSystem è possibile ricavare alcune caratteristiche della macchina su cui sta girando l’applicazione. Vengono rilevate durante la fase di esecuzione, andando a leggere le seguenti variabili:

CHECKMODEL -> 1 = MSX, 2 = MSX2, 3 = MSX2+, 4 = MSX turbo R
CHECKVDP -> 1 = TMS99xx, 2 = V9938, 3 = V9958
CHECKRAM -> amount of RAM in KBytes
CHECKVRAM -> amount of VRAM in KBytes

Creare diverse versioni del progetto

Tramite alcune istruzioni di precompilazione o con l’istruzione CheckSystem è possibile creare diverse versioni del vostro progetto, a seconda del tipo di MSX o di VDP di destinazione.
La differenza tra istruzioni di precompilazione e CheckSystem è questa:
le istruzioni di precompilazione hanno effetto durante la fase di compilazione, mentre CheckSystem va gestita durante la fase di Runtime.

Ecco alcuni esempi:

Tramite le istruzioni #If… #Else… #EndIf è possibile dare indicazioni al compilatore su quale codice vogliamo effettivamente compilare e quale no:

Nel seguente esempio, se impostiamo come TARGET la cartuccia, allora verrà compilata solo la prima istruzione “print”, se invece impostiamo il TARGET su disco, allora verrà compilata la seconda istruzione “print”:

```
#If TARGET="ROM" Then
        Print “This is a ROM version”
#Else
        Print “This is a DISK version”
#EndIf
```

In questo esempio, simile al precedente, invece scegliamo quali righe compilare a seconda del VDP che abbiamo scelto nella finestra Impostazioni:

```
#If VDPMOD="TMS99xx" Then
        Screen 2
#Else
        Screen 4
#EndIf
```

Tenere ben presente che il vdp di cui stiamo parlando (gestito tramite la variabile VDPMOD) non è il modello di VDP presente sulla macchina, ma quello che abbiamo scelto nelle impostazioni.

Se volessimo fare la stessa cosa, ma andando a leggere il modello di VDP della macchina su cui sta girando la nostra applicazione, allora useremo prima l’istruzione CheckSystem e poi leggeremo la variabile CHECKVDP:

```
CheckSystem

If CHECKVDP=1 Then
        Screen 2
Else
        Screen 4
EndIf
```

Mentre nel primo caso (VDPMOD) viene compilata solo la riga specificata, in quest’ultimo (CHECKVDP) invece vengono compilate entrambe le righe (Screen 2 e Screen 4), ma in fase di RUNTIME verrà eseguita solo quella interessata.

Compilazione da riga di comando

È possibile compilare il progetto direttamente da riga di comando del terminale o tramite file .bat (windows).

L'esempio più semplice consiste nello specificare solo il percorso e il nome del file di progetto (.msxproj) o del file principale (nel caso il progetto sia composto da più files) tramite la stringa -FILEPROJ

MSXcompilerV4_v2025_1 -FILEPROJ="path/nameproject.msxproj"

In questo modo verrà creato un file .ROM, con le impostazioni di default (stack impostato su 0DB00h.
Nel caso volessimo creare la versione disco allora dovremo aggiungere il comando -TARGET="disk". Al termine della compilazione i file di progetto su disco si chiameranno PROJECT, il disco creato sarà in doppia faccia e i files verranno preservati nel caso esista già il file .DSK, inoltre verrà incluso il file "AUTOEXEC.BAS")

Nel prossimo esempio invece creeremo solo la versione .ROM (cartuccia) e la proveremo direttamente su emulatore una volta compilato il progetto, quindi dovremo anche specificare il percorso e il nome dell'emulatore:

MSXcompilerV4_v2025_1 -FILEPROJ="path/nameproject.msxproj" -TARGET="ROM" -EMULPATH="emulatorpath/emulator.exe"

Di seguito la lista delle possibili impostazioni da settare nella linea di comando:

-FILEPROJ	"percorso/nomeprogetto.msxproj" del main project
-TARGET	"ROM" oppure "DISK" (o "DSK")
-EMULPATH	"percorsoemulatore/nomeemulatore.exe"
-EMULCOMM	impostazioni per la riga di comando dell'emulatore
-STACK	Indirizzo dello stack pointer (in decimale o esadecimale con $)
-DISKNAME	nome dei files di progetto su disco (max 8 caratteri)
-DISKPRES	"YES" oppure "1" (preserva i file del disco, tranne quelli di progetto, quando viene creato il progetto su disco) "NO" oppure "0" (crea un nuovo disco)
-DISKSIDE	"1" oppure "2" (indica di quante facce verrà creato il disco)
-DISKMODE	"NORMAL" oppure "0" (disco normale) "SECTORS" oppure "1" (il disco contiene solo i files di caricamento)
-AUTOEXEC	"YES" oppure "1" (include il file "autoexec.bas") "NO" oppure "0" (non include il file autoexec)
-PROJMODE	"MAPPER" oppure "0" (progetto in memory mapper) "SLOTS" oppure "1" (progetto 32KB)
-ROMDISK	"YES" oppure "1" (supporto per disk drive quando viene creato il file .ROM) "NO" oppure "0" (nessun supporto per il disco)
-MSXMOD	"MSX1", "MSX2", "MSX2+", "MSXTR"
-VDPMOD	"TMS99XX", "V9938", "V9958"
-CUSTOM	"stringa a piacere" (usato con i comandi preprocessore)

Il resto delle informazioni su MSXcompilerV4 versione 2025, verrà inclusa in questo manuale prossimamente.