GMC-4

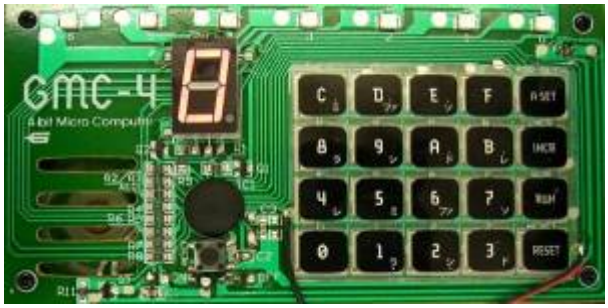
Der **Gakken GMC-4** ist ein kompletter vom PC unabhängig zu nutzender kleiner Lerncomputer Er basiert auf dem Science Fair Microcomputer Trainer von Radio Shack aus den 80ern.

Dezember 2014 gab es das Franzis Lernpaket „Spielecomputer selbst programmieren“ mit dem GMC-4 und einem 96seitigen Handbuch von F. Kainka für nur 20€, so dass ich mir auch einen solchen kleinen Computer zugelegt habe.



Systembeschreibung

Bild, kurze Beschreibung, https://en.wikipedia.org/wiki/GMC-4



Der GMC-4 ist ein 4-Bit-Microcomputer.

Der GMC-4 verfügt über einen einfachen Befehlssatz, der an Assembler angelehnt ist, teilweise aber auch höhere Funktionen enthält, hinter denen sich komplexe Routinen verstecken.

Es gibt vier 4-Bit-Register A, B, Y und Z und einen zusätzlichen zweiten Registersatz A‘, B‘, Y‘ und Z‘. Jedes dieser Register kann eine Zahl zwischen 0 und 15 enthalten. Die meisten Befehle verwenden das Register A.

Der Einplatinenrechner besitzt eine Hexadezimal-Tastatur, 4 Funktionstasten, 7 LEDs und eine einstellige 7-Segmentanzeige. Außerdem gibt es einen kleinen Lautsprecher zur Tonausgabe.

Downloads

- Reasembler gmc4_reass2.pl
  gmc4_reass2.zip
- alle Beispielprogramme aus dem originalen Handbuch liegen demc Simulatorpaket GMC4Sim-1_38.zip bei (http://dansan.air-nifty.com/blog/gmc4-simulator.html).
- Anleitung (engl., GMC-4-Manual)
  vol24_microcomputer_gmc-4.pdf

technische Daten

Merkmal	Beschreibung
CPU	EM61001
ROM	-
RAM	128 Byte a 4 Bit
Takt	4 MHz
Anzeige	7 Bit LED, 1 stellig Siebensegment-Anzeige
Tastatur	20 Tasten Hexadezimaltastatur
Peripherie	-
Software	-

CPU und Takt spielen keine Rolle, im ROM-Programm des Prozessors wird ein spezieller 4-Bit-Prozessor simuliert (s.u.). Nur für diesen 4-Bit-Prozessor können Programme geschrieben werden, der zugrunde liegende Prozessor ist nicht ansprechbar. Das Konzept ist ähnlich der CHIP-8-Sprache.

Im Science Fair Microcomputer Trainer wird ein spezieller Microcontroller der SerieTMS1000-Controller genutzt, konkret ein TMS1312. Dieser wurde mit 400 kHz betrieben.

Literatur

1. http://www.polylith.com/~brendan/ClassicComputers/Tandy/image/MCT_28_260.pdf (von http://www.polylith.com/~brendan/ClassicComputers/Tandy/uCptrTrainManual1.html) Das originale Handbuch zum Microcomputer Trainer. Das Handbuch ist komplett für den GMC-4 nutzbar!
2. http://otonanokagaku.net/magazine/vol24/pdf/vol24manual.pdf Das Handbuch zum FX-165. Das ist i.W. eine japanische Übersetzung des obigen Handbuchs. Die 100 Beispielprogramme, die Kapitelnummern u.a. stimmen überein!
3. http://fkainka.de/tag/gmc-4/

Bedienung

0	
1	ラ
2	シ
3	ド

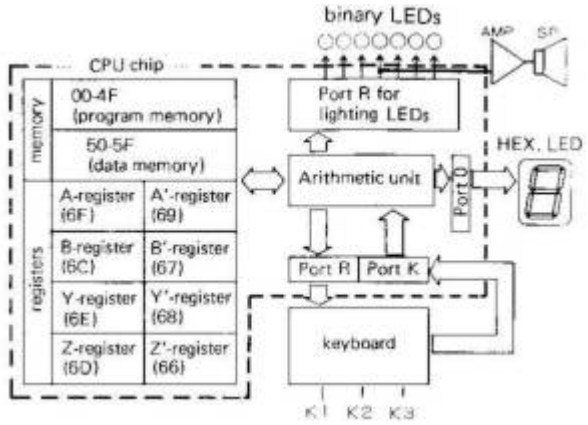
4	レ
5	ミ
6	ファ
7	ソ
8	ラ
9	シ
A	ド
B	レ
C	ミ
D	ファ
E	ソ
F	

Sonstiges

Prozessor

Es wird ein spezieller 4-Bit-Prozessor simuliert.

- 4 Register A, B, Y, Z
  A ist der Akku, Y ein Indexregister zum Speicherzugriff. B und Z sind Wechselregister.
- 3 Ports P (LEDs), O (Siebensegmentanzeige), K (Tastatur)
- 80 Byte (a 4 Bit) Programm-Speicher (RAM-Bereich 00h..4Fh)
- 16 Byte Datenspeicher (RAM-Bereich 50h..5Fh)
- 1 Flag. Der Befehl JUMP ist ein bedingter Sprung und wird nur ausgeführt, wenn das Flag 1 ist
- 4 zusätzliche Wechselregister A', B', X', Z'



Der Befehlssatz

Der simulierte 4-Bit-Prozessor versteht 16 Maschinenbefehle.
Die Befehle sind 1..3 Byte (a 4 Bit) lang.

n = 0..F
M = RAM-Adr (50+Y) (indexierter Zugriff)

Code	Befehl	Wirkung	Flag	Kommentar
0	KA (Key to A)	A := K	0, 1	Tastendruck nach A übernehmen, Flag = 0 bei Tastendruck, sonst Flag = 1
1	AO (A to Output)	O := A	1	A an Siebensegmentanzeige ausgeben
2	CH (Change)	A ⇔ B Y ⇔ Z	1	Tausche A und B sowie Y und Z
3	CY (Change A and Y)	A ⇔ Y	1	Austausch der Inhalte von A und Y
4	AM (A to Memory)	M := A	1	A in RAM-Adr (50+Y) kopieren
5	MA (Memory to A)	A := M	1	RAM-Adr (50+Y) in A zurücklesen
6	M+	A := M+A	0, 1	RAM-Adr (50+Y) zu A addieren, bei Überlauf: Flag = 1
7	M-	A := M-A	0, 1	RAM-Adr (50+Y) von A subtrahieren, bei Überlauf: Flag = 1
8 n	TIA n (Transfer Immediate to A)	A := n	1	Konstante in A laden
9 n	AIA n (Add Immediate to A)	A := A+n	0, 1	Konstante zu A addieren, bei Überlauf: Flag = 1
A n	TIY n (Transfer Immediate to Y)	Y := n	1	Konstante in Y laden
B n	AIY n (Add Immediate to Y)	Y := Y+n	0, 1	Konstante zu Y addieren, bei Überlauf: Flag = 1
C n	CIA n (Compare Immediate with A)	A == n ?	0, 1	A mit Konstante vergleichen, bei Übereinstimmung: Flag =0
D n	CIY n (Compare Immediate with Y)	Y == n ?	0, 1	Y mit Konstante vergleichen, bei Übereinstimmung: Flag =0
E n	CAL n (Call)	—	—	Unterprogrammaufrufe, Erweiterte Befehle
F n n	JUMP n n	Adr := nn	1	Direkter Sprung zur Adresse nn (High, Low) wenn Flag = 1

Unterprogramme:

Code	Befehl	Flag	Kommentar
E 0	CAL RSTO (Reset Port O)	1	Siebensegmentanzeige löschen
E 1	CAL SETR (Set Port R)	1	Einzelne LED einschalten. In Y wird die Nummer der LED (0-6) übergeben.
E 2	CAL RSTR (Reset Port R)	1	Einzelne LED ausschalten. In Y wird die Nummer der LED (0-6) übergeben.
E 4	CAL CMPL (Complement)	1	Komplement des A-Registers (aus F wird 0)
E 5	CAL CHNG	1	Inhalte der Register A,B,Y,Z mit A',B',Y',Z' tauschen
E 6	CAL SIFT	0, 1	A-Register bitweise nach recht schieben. Flag wird 1 wenn das rechte Bit 0 war.
E 7	CAL ENDS	1	Ende-Sound
E 8	CAL ERRS	1	Error-Sound
E 9	CAL SHTS (Short Sound)	1	Kurzer Ton
E A	CAL LONS	1	Langer Ton
E B	CAL SUND	1	Note spielen, die im A-Register übergeben wird (1 ... E)
E C	CAL TIMR	1	(A + 1) * 0,1 Sekunden warten
E D	CAL DSPR (Display on Port R)	1	Ausgabe der RAM-Adressen 5F (high) und 5E (low) an die LEDs

Reassembler

Ich habe einen kleinen kommentierenden Reassembler gmc4_reass2.pl in Perl geschrieben. Aus dem Hexcode

```
E D 8 0 E C A F 5 E 6 F 1 A 4 A E 5 E 6 9 8 4 F 2 1 4 A E 5 E 6 4 0 F 0 0 A
F 5 9 8 4 F 0 0
```

macht er folgendes (Beispiel binary_light_show.asm):

```
00: E D   LBL_00: CAL DSPR     ; Ausgabe der RAM-Adressen M[F] (high) und
M[E] (low) an die LEDs
02: 8 0           TIA 0        ; Konstante 0 in A laden
04: E C           CAL TIMR     ; (A + 1) * 0,1 Sekunden warten
06: A F           TIY F        ; Konstante 15 in Y laden
08: 5             MA           ; RAM-Adr M[Y] in A zurücklesen
09: E 6           CAL SIFT     ; A-Register bitweise nach rechts schieben
0B: F 1A          JUMP LBL_1A  ; wenn das rechte Bit 0 war Sprung zur
Adresse LBL_1A
0E: 4             AM           ; A in RAM-Adr M[Y] kopieren
0F: A E           TIY E        ; Konstante 14 in Y laden
11: 5             MA           ; RAM-Adr M[Y] in A zurücklesen
12: E 6           CAL SIFT     ; A-Register bitweise nach rechts schieben
14: 9 8           AIA 8        ; 8 zu A addieren
16: 4             AM           ; A in RAM-Adr M[Y] kopieren
17: F 21          JUMP LBL_21  ; Sprung zur Adresse LBL_21
1A: 4     LBL_1A: AM           ; A in RAM-Adr M[Y] kopieren
1B: A E           TIY E        ; Konstante 14 in Y laden
1D: 5             MA           ; RAM-Adr M[Y] in A zurücklesen
1E: E 6           CAL SIFT     ; A-Register bitweise nach rechts schieben
20: 4             AM           ; A in RAM-Adr M[Y] kopieren
21: 0     LBL_21: KA           ; Tastendruck nach A übernehmen
22: F 00          JUMP LBL_00  ; wenn kein Tastendruck Sprung zur Adresse
LBL_00
25: A F           TIY F        ; Konstante 15 in Y laden
27: 5             MA           ; RAM-Adr M[Y] in A zurücklesen
28: 9 8           AIA 8        ; 8 zu A addieren
2A: 4             AM           ; A in RAM-Adr M[Y] kopieren
2B: F 00          JUMP LBL_00  ; Sprung zur Adresse LBL_00
```

From:
https://hc-ddr.hucki.net/wiki/ - **Homecomputer DDR**

Permanent link:
https://hc-ddr.hucki.net/wiki/doku.php/homecomputer/gmc-4?rev=1533644949

Last update: **2018/08/07 12:29**