VIERTE DIMENSION

2023/2

The Forth Magazine

for science and technology, for commercial EDP,
for ICE–technology, for the passionate hobbyist

In this Issue:

AW60 in Production Device

MC9S08AW Flasher

Large Literals for the RP2040 Assembler

Astroimaging in Forth: Image File Format and Camera SDK

Minutes of the General Meeting of the Forth–Gesellschaft e.V. on May 7, 2023

Volume 39 — **Issue 2023/2** — 4,– €

Organ of the German Forth–Gesellschaft e.V.

Services and Products

ServoNaut

Fahrtregler - Lichtanlagen - Soundmodule - Modellfunk

tematik GmbH
Technische
Informatik

Feldstraße 143
D–22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
`http://www.tematik.de`

Seit 2001 entwickeln und vertreiben wir unter dem Markennamen „Servonaut“ Baugruppen für den Funktionsmodellbau wie Fahrtregler, Lichtanlagen, Soundmodule und Funkmodule. Unsere Module werden vorwiegend in LKW–Modellen im Maßstab 1:14 bzw. 1:16 eingesetzt, aber auch in Baumaschinen wie Baggern, Radladern etc. Wir entwickeln mit eigenen Werkzeugen in Forth für die Freescale–Prozessoren 68HC08, S08, Coldfire sowie Atmel AVR.

Forth Trainings

Do you want to learn the Forth programming language or get trained in the new Forth developments? Do you have products based on Forth and would like to train employees in the maintenance and further development of these products?

We offer training in legacy Forth systems (FIG–Forth, Forth83), ANSI–Forth and the latest Forth 200x standards. Our trainers have over 20 years of experience with Forth programming on embedded systems (ARM, MSP430, Atmel AVR, M68K, 6502, Z80 and many more) and on PC systems (Linux, BSD, macOS and Windows).

Carsten Strotmann carsten@strotmann.de
`https://forth-schulung.de`

RetroForth
Linux · Windows · Native
Generic · L4Ka::Pistachio · Dex4u
Public Domain
`http://www.retroforth.org`
`http://retro.tunes.org`

Diese Anzeige wird gesponsort von:
EDV–Beratung Schmiedl, Am Bräuweiher 4,
93499 Zandt

CORNU

Cornu GmbH
Ingenieurdienstleistungen
Elektrotechnik

Weitlstraße 140
80995 München
`sales@cornu.de`
`www.cornu.de`

Unser Themenschwerpunkt ist automotive SW unter AutoSAR. In Forth bieten wir u. a. Lösungen zur Verarbeitung großer Datenmengen, Modultests und modellgetriebene SW, z. B. auf Basis eCore/EMF.

KIMA Echtzeitsysteme GmbH

Güstener Straße 72 — 52428 Jülich
Tel.: 02463/9967–0 — Fax: 02463/9967–99
`www.kimaE.de` — info@kimaE.de

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug– und Sondermaschinen, Fuzzy Logic.

FORTecH Software GmbH

Tannenweg 22 m — D–18059 Rostock
`https://www.fortech.de/`

Wir entwickeln seit fast 20 Jahren kundenspezifische Software für industrielle Anwendungen. In dieser Zeit entstanden in Zusammenarbeit mit Kunden und Partnern Lösungen für verschiedenste Branchen, vor allem für die chemische Industrie, die Automobilindustrie und die Medizintechnik.

Ingenieurbüro
Klaus Kohl–Schöpe

Tel.: **(0 82 66)–36 09 862**
Prof.–Hamp–Str. 5
D–87745 Eppishausen

FORTH–Software (volksFORTH, KKFORTH und viele PD–Versionen). FORTH–Hardware (z. B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs– und Messtechnik.

Microcontroller Pool
Forth–Gesellschaft e. V.

We provide high quality evaluation boards, also FPGA, including Forth systems: Cypress, RISC–V, TI, GA144, MicroCore, SeaForth, MiniMuck, Zilog, Motorola, 68HC11, ATMEL, Hitachi, Renesas, Lego …
`https://wiki.forth-ev.de/doku.php/mcv:mcv2`

Letters to the Editor and News 5

AW60 in Production Device 9
Rafael Deliano

MC9S08AW Flasher 12
Rafael Deliano

Large Literals for the RP2040 Assembler 15
Willem Ouwerkerk

Astroimaging in Forth: Image File Format and Camera SDK 18
Andrew Read

Minutes of the General Meeting of the Forth–Gesellschaft e.V. on May 7, 2023 22
Anton Ertl

Cover picture: Geminids meteor shower AUTHOR: mk
Source: fetched from the internet, modified

Imprint

Title of the Magazine

Vierte Dimension

Publisher

Forth–Gesellschaft e. V.
P.O. Box 1030
48481 Neuenkirchen
Germany
Email: `Secretary@forth-ev.de`
`Direktorium@forth-ev.de`
Bank account: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Editorial & Layout

Bernd Paysan, Ulrich Hoffmann
Email: `4d@forth-ev.de`

Ad Management

Office of the publisher

Editorial Deadline

January, April, July, October each in the third week

Publication Frequency

1 issue per calendar quarter

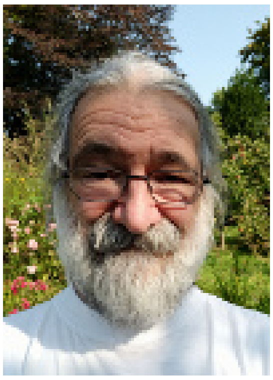
Unit Price

4,00 € plus postage and packing

Manuscripts and Rights

All manuscripts sent in will be considered. Letters to the editor may be reproduced without consultation. For articles marked with the name of the author, the editorial team only accepts responsibility under press law. The articles published in this magazine are protected by copyright.
Translation, reproduction, as well as storage on any media, in whole or in part, is only permitted with exact reference to the source. The submitted contributions must be free from claims of third parties. Published programs go — as far as nothing else is noted — into the public domain.
No liability can be accepted for text, circuit diagrams or assembly sketches which lead to the non–functioning or possible damage of components. All publications are made without consideration of possible patent protection. Trade names are used without any warranty of free use.

Dear readers,



the Geminids are one of the strongest meteor shower of the year with a maximum in the middle of December. Its radiant, the point from which the shooting stars seem to come, lies just north of the bright star Castor in the constellation Gemini (Latin gemini), which gave the Geminids their name.

Our Forth magazine now also has a twin. Wolfgang Strauss has translated it completely into English. There are many hopes attached to this. We are accommodating our international readership. And we hope that the articles will be translated better than the various automats are able to do — they will then have enough to do to bring this from English into all the other languages of this world.

Let's stay with the stars. Andrew Read provides further insight into astroimaging. This time it's about controlling the camera, which is not at his home, but further away, where the sky is starry and without light pollution by the many lights of civilization. If you are ever there, let us know.

Willem Ouwerkerk gives insight into the other direction, into the very small that can only be seen with a microscope, the registers in the RP2040 chip. Using these efficiently for noForth T is his goal. Yes, right: "T" — noForth conquers the M0+ core, and comes along with a forthy Thumb2 assembler.

Whether it's the smallest or the biggest in space, there's always something to control. Rafael Deliano has two articles in this issue. He lets us participate in the development, production and quality control of such components, forth–style. Prototyping and small series production work particularly well with them.

Even if at the last minute: the announcement of the Forth summer meeting still made it onto the back page of the magazine. Have you all already registered? That's great. Then I look forward to seeing you again soon at the Linux Hotel.

All yours Michael

You do not have to manually type the source codes in the VD. You can easily download them from the web page of the association.
`http://fossil.forth-ev.de/vd-2023-02`

The Forth–Gesellschaft e. V. is represented by its board of directors:

Ulrich Hoffmann   Contact: `Direktorium@Forth-ev.de`
Bernd Paysan
Gerald Wodni

Charlatans, now digital too

AI is on everyone's lips, a huge hype. Inspired by Ulli Hoffmann and his experiences I could not resist and tried ChatGPT myself. I quote:

> *Forth's Future Outlook:*
>
> **user:** Forth programming language. Future prospects?
>
> **assistant:**
>
> Forth is a programming language that has been around since the 1970s and is used especially in embedded system development and control engineering. There is still an active community of Forth developers and users who continue to develop and maintain the language.
>
> However, Forth has become less important in recent years as other programming languages such as Python, JavaScript, Java and C++ have become more popular. This is partly because these languages are better adapted to modern development environments, libraries and frameworks, and have a larger community of developers and users.
>
> However, this does not mean that Forth has no future. There are still specialized applications where Forth has advantages over other languages due to its efficiency and flexibility. Also, there is still an active community of Forth developers who continue to improve and maintain the language.
>
> Overall, I would say that Forth will not be widely used in the foreseeable future, but will continue to play an important role in specialized applications.

Super. I couldn't have formulated it better myself, and in perfect English too. That inspires confidence in this technology.

Since I always spend a lot of time looking for suitable components for my products, I'll ask the AI right away:

> *SMT MOSFET with Vgs*
>
> **user:** Find a small SMT N–channel MOSFET with an Vgs of at least 30 V
>
> **assistant:**
>
> One example of an SMT (Surface Mount Technology) N–channel MOSFET with a Vgs of at least 30 V is the 2N7002LT1G MOSFET manufactured by ON Semiconductor.
>
> Some of its specifications are:
> – Drain–Source Voltage (Vdss): 60 V
> – Gate–Source Voltage (Vgs): ±20 V
> – Continuous Drain Current (Id): 330 mA
> – Power Dissipation (Pd): 200 mW — Package: SOT–23
>
> Please note that there may be other MOSFETs available with similar specifications, and it is important to choose a MOSFET that is appropriate for your specific application.
>
> **user:** Wrong. The Vgs should be greater than ±30 V.
>
> **assistant:** I apologize for the mistake in my previous answer. To find an SMT N–channel MOSFET with a Vgs of at least ±30 V, you can consider the following options:
>
> 1. Vishay Siliconix SiB121EDJ–T1–GE3 MOSFET: This MOSFET has a Vgs of ±40 V and a package size of SOT–23. Some of its key specifications are:
> – Drain–Source Voltage (Vdss): 60 V
> – Continuous Drain Current (Id): 4 A
> – Rds(on): 48 mOhms
>
> 2. Diodes Incorporated DMT6008LPS MOSFET: This MOSFET has a Vgs of ±30 V and a package size of PowerDI 5x6. Some of its key specifications are:
> – Drain–Source Voltage (Vdss): 60 V
> – Continuous Drain Current (Id): 3.8 A
> – Rds(on): 25 mOhms
>
> Please note that there may be other MOSFETs available with similar specifications, and it is important to choose a MOSFET that is appropriate for your specific application.
>
> **user:** any smaller parts
>
> **assistant:** If you are looking for an SMT N–channel MOSFET with a Vgs of at least ±30 V in a smaller package, you can consider the following options:
>
> 1. ON Semiconductor NTLUD3317PZTBG MOSFET: This MOSFET has a Vgs of ±30 V and a package size of SOT–563. Some of its key specifications are:
> – Drain–Source Voltage (Vdss): 20 V
> – Continuous Drain Current (Id): 0.4 A
> – Rds(on): 700 mOhms
>
> 2. Diodes Incorporated DMN61D0LQ MOSFET: This MOSFET has a Vgs of ±30 V and a package size of SOT–23–3L. Some of its key specifications are:
> – Drain–Source Voltage (Vdss): 60 V
> – Continuous Drain Current (Id): 0.3 A
> – Rds(on): 2.8 Ohms
>
> Please note that these MOSFETs have lower maximum drain–source voltage ratings and current ratings than the MOSFETs I mentioned in my previous answer. It is important to choose a MOSFET that is appropriate for your specific application and meets the necessary requirements.

Yes, that's awesome! What a saving in working time!

However — not really … A 2N7002LT1G actually exists. But if you try to buy the other types mentioned, you will get a surprise: they do not exist. The transistors are freely invented. Whereby "invent" is possibly the wrong word. Experts tend to speak of “hallucinating”[1] in connection with an AI. And these digital hallucinations, unlike in this case, are often difficult to spot, because everything seems totally plausible, even the type designations correspond to the usual pattern of the respective alleged manufacturer.

In this context this is amazing, I was speechless at first. In other contexts, however, this could be very dangerous.

> “Charlatans blend truth with lies to create a truth substitute.”

That’s the best definition I’ve found on the net. And perfectly applicable to what the AI is here, a *digital charlatan*.

Brave New World. Jörg Völker

Write a “Hello World” Programm in Forth

> Sure! Here’s an example of a “Hello World” program written in Forth:
>
> `." Hello World!" CR`
>
> Explanation: `."` is a Forth word that prints a string to the output. “Hello World!” is the string that we want to print. `CR` is a Forth word that outputs a carriage return, which moves the cursor to a new line. When you run this Forth program, it will print “Hello World!” to the output with a new line, typically on the terminal or console where you are running the Forth interpreter or compiler.
>
> Note: The syntax and functionality of Forth can vary depending on the implementation and system you are using. This example assumes a standard Forth environment with a period (`.`) as the word separator and a carriage return (`CR`) as the newline character. Please refer to the documentation or user guide of your specific Forth system for more information on how to write and execute Forth programs.

That’s what ChatGPT answered me the other day. How it comes to the conclusion that the word separator is a period, is beyond me, as it is the space character in reality.[2] Be that as it may. The program is correct. The explanation as well. As an expert, one simply irons out the mistakes and has written a newspaper article in no time.

Note: If you don’t believe that this came from ChatGPT: I saved the screenshot. But because it was written in a light font on a dark background, which would have been difficult to print in our magazine, the text is available here as a quote. It’s good that we always indicate the author of a contribution, and sources. That’s how we want to continue to do it in the future. mk

New Kid on the Block: noForth T

In May, Willem Ouwerkerk sent an interesting article for our magazine — you’ve probably already discovered it in the table of contents. It came out that noForth has conquered another frequently used MCU. He wrote:

> „It will be available soon on the noForth web–site! In the meantime, the pre–release version is available when people send me an email. Then I will send them the ZIP of the pre–release version to test.
>
> This pre–release contains the tools, an assembler and disassembler, in addition a PIO assembler and disassembler, configuration files, commacode, etc.“

May the noForth Model T[3] become as successful as the Model T once was for Ford, the “Tin Lizzie”. mk

`https://home.hccnet.nl/anij/nof/noforth.html`

VCFe — Vintage Computer Festival Europa

For the 22nd time the VCF was held on the weekend of April 29 — May 01, 2023 in our beautiful city of Munich. On `https://www.vcfe.org/D/` you can read about it:

> “VCF is not only a regular event in California’s Silicon Valley, but has also become a fixed point in European schedules.
>
> So let’s go back to the Good Ol’ Days, when hackers weren’t security consultants, bytes weren’t megabytes, and Little Green Men were just Little Green Men!”

Every year, the VCFe is dedicated to a main topic. This time it was: “Silicon Mountains — Computers along the Alps.” From *ERMETH*, *Smacky* and *Lilith* in the west to *Mailüfterl* and *Mupid* in the east and important manufacturers of the early days, like *Siemens* in the north and *Olivetti* in the south, the journey went, and to the basic concepts from *ALGOL* to *Stack*.

Rafael Deliano took a look around and picked out something for our magazine.

Lilith

He wrote to me:

---

[1] Hallucination (from Latin *alucinatio* “reverie”) is a perception for which there is no demonstrable external stimulus basis

[2] In psychopathology, confabulation or confabulating (as in Latin confabulatio, “narrative”, from fabula, “fable”, ”story”, “fairy tale”) refers to the production of objectively false memories, e.g., when someone tries to retrieve more information from his or her memory than is actually stored (so–called provoked confabulations)

[3] ARM, **T**humb–2

[4] The Xerox Alto was developed at the Xerox PARC Research Center in 1973 and was the first workstation with a graphical user interface

> "A vague reference to Forth would have been this old–timer: *Lilith*, the European Alto[4]. The Lilith was developed as a high–performance stand–alone computer with high–resolution graphics and mouse operation starting in 1980 at ETH Zurich. All of the software is written in the high–level language Modula–2. Hardware and software were developed by a team around Niklaus Wirth. The exhibition showed an original Lilith as well as the emulator *Emulith*, with which one could try out the original operating software — the main topic of JOS DREESEN.
>
> The hardware was a "2900" bit–slice computer, which was a stack computer. However, the intermediate language was M–code."
>
> `http://www.bitsavers.org/pdf/eth/lilith/Lilith_hardware_manual.pdf`

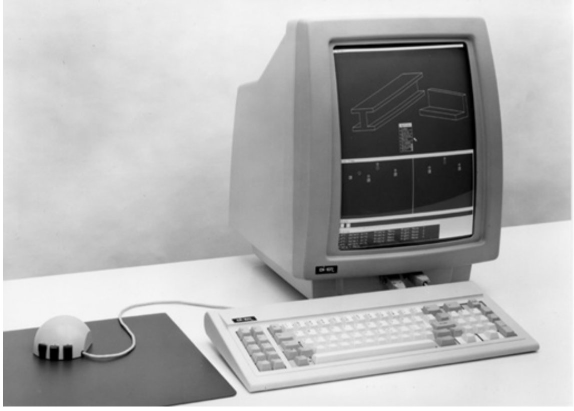


Figure 1: Lilith

Z79Forth

> "Interesting then also a *Z79Forth*. Unfortunately, the exhibitor FRANCOIS LAAGEL did not show up or I overlooked him. Therefore only a quote: 'Z79Forth is a journey into retro–computing using a Hitachi 63C09, but takes advantage of modern technology where appropriate (CMOS, USB and CompactFlash). Wire wrapping is particularly adapted to digital hardware experimentation and provides the developer with a great deal of flexibility.' "
>
> `https://www.elektormagazine.de/labs/72832/francois-laagel`

Maybe we can learn more about it from Laagel. mk

Erratum

FRANCOIS LAAGEL emailed us in May this year:

> "I noticed an overview of Forth systems in VD 4/2022 and would like to contribute a few fixes to the entry labelled as 'Zforth.'
>
> Name: Z79Forth
> Author: Francois Laagel
> Standard: 79 or ANS94"

Dear readers, please add this correction to your list. And please let us know if you find any other errors in the list. mk

Dragon Moving

SWAPPY, our favorite dragon, loves to look over his keeper's[5] shoulder and learn new things about Forth and other things. However, there always comes a time when he thinks he has seen everything there is to see at his current location. I noticed lately that he was getting a bit restless, obviously he started to itch in his claws and wings again.

And so, as is his annual habit, he called his dragon council together on the Saturday evening of the Forth meeting to consult with them about his next destination. At the end of the deliberations, Swappy decided, to the approving murmur of his council, that he would spend the next year with WILLEM OUWERKERK in Arnhem. He had heard many good things about Willem, had seen him make public appearances in matters Forth, had heard of his dancing robot arms, and of course he is curious about noForth.

Figure 2: Swappy in his travel box, ready for departure

So it was up to me to bring out his wonderful and almost new travel cave, used only once before, and assist Swappy to take a seat in it. For a short while, neither of us could remember how around Swappy had lain in his cradle, so that the seat belt secured well, and the seat gave good support. But then we noticed that Swappy had left clearly visible impressions on the seat with his back ridge, and with the memory then coming back to us, everything was then clear again. Swappy took his seat, and I put his harness on. Another wink of the right eye of the right head, hood pulled over, treasure chest stowed, cave locked, and off — or so I thought. But then Swappy found that his lair rattled a bit too much when I moved the travel box; he didn't want to go like that. So — what wouldn't you do for a dragon — open the cave again, get paper, take out the treasure chest, open the treasure

---

[5] For the past 12 months, that's been me

chest, stuff some paper in it, close the treasure chest, stow the treasure chest, pack paper around it, close the cave, put the key under the doormat, close the package, get the destination coordinates ready, and off to the next post office. Swappy has similar travel preferences to Iwan's, if Terry Pratchett's "Thief of Time" is to be believed.

I am curious to hear what Swappy will report about his new domicile and sphere of activity.

PS: I could have spared Swappy and myself the search for the correct seating position if I had read my own report in 4d2022–02 earlier. :-) Philip Zembrod

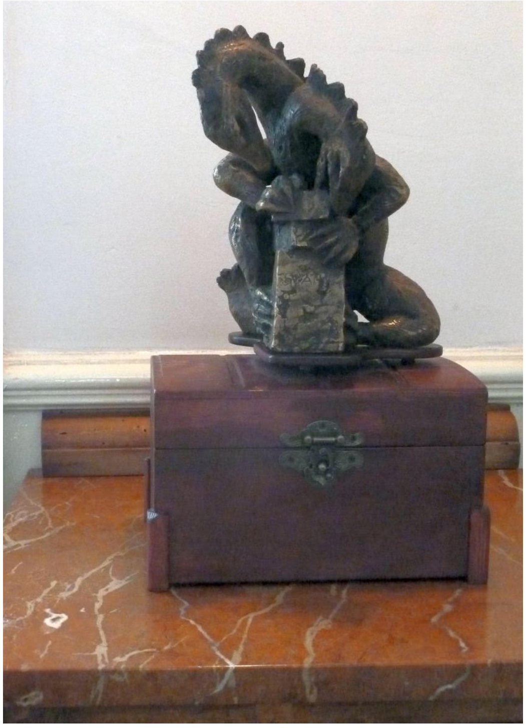


Figure 3: Swappy at Willem's home

Dragon Arrival

Yes, the swap dragon arrived in very healthy condition. Despite the somewhat long journey for him (he had to spend a few days in a depot). Beautifully wrapped and on top a nice filled treasure chest. I am very happy that he gets to live here for a year. He also got a very nice spot as shown in the picture (figure 3). Willem Ouwerkerk

VD Team is Looking for International Support

At the membership meeting in 2023, we decided to publish an international edition of our club magazine "Vierte Dimension" in English in addition to the German edition. We expect this to stimulate Forth activities and expand the networking of Forthers around the globe.

The project is designed as a test for a period of 4 issues. After that it will be decided if the project will be continued. The small German team will not be able to handle the additional workload alone in the long run.

Therefore, we are on the lookout for contributors for various areas of the project:

- English native speakers (proof readers)
- Translator German → English
- Translator English → German
- English authors
- German authors
- Makers who program hardware projects in Forth

Not sure if you fit into one of these categories? We should talk to each other. An interesting cooperation will be found.

The magazine "Vierte Dimension" is a hobby project and will be created on a voluntary basis. So do not worry: we don't expect full professionals with several years of work experience. Enthusiasm for the programming language Forth and solid knowledge of English or/and German are quite sufficient.

Please contact the Vierte Dimension team at `vd@forth-ev.de` if you are interested. Questions and feedback are also always welcome.

Wolfgang Strauß

Continued on page 11

AW60 in Production Device

Rafael Deliano

Due to the continued poor supply situation for controllers, the MC9S08AW60 had to be used once again for production. It is already obsolete, but as a former automotive standard type it is better available than what the manufacturers are promoting.

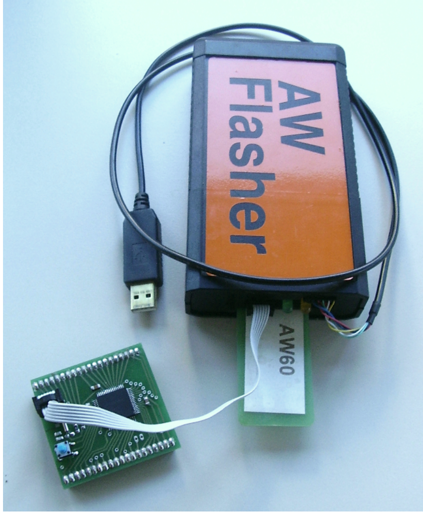


Figure 1: Flasher and AW60 SBC

Because of the manageable quantities, the existing programmer should be used unchanged for this purpose (figure 1). A printed circuit board with the AW60 normally contains the 6–pin header (figures 2, 3) defined by Motorola. This BGD[1] interface is required for flashing the controller.

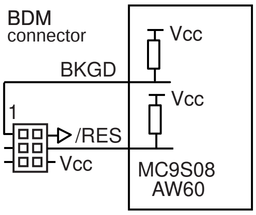


Figure 2: BDM schematic

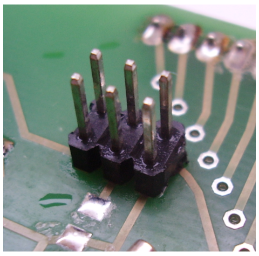


Figure 3: BDM as 6–pin header on SBC

To save on material costs, and also because of the overall height, a cut–off 2.54 mm edge connector was used instead of a pin header (figure 4); a common alternative would be spring–loaded test pins. There are also a variety of clips from China via `ebay.com` (figure 5) with different pin counts and pitches.

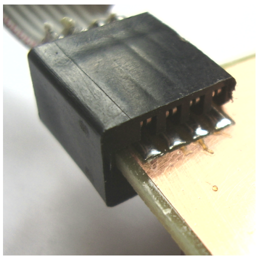


Figure 4: 8–pin edge connector

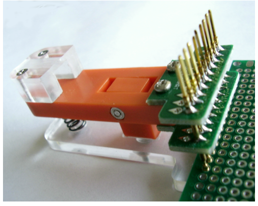


Figure 5: 10–pin pogo clip from China

BDM Extended

More lines make additional functions available. RXD/TXD of the UART are the bare minimum. Via the command line of FORTH you can test the PCB comfortably. The pin `/EN` stops the disturbing external watchdog. Instead of a crystal oscillator, another pin feeds a 32 kHz clock. From this the 4.19 MHz bus clock was generated on the AW60 SBC. Normally the nanoFORTH firmware initializes the controller in this mode, but not in this case. Instead, the internal RC clock is to be used to save components.

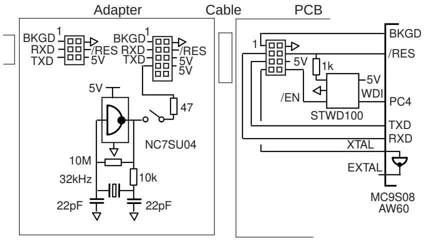


Figure 6: Schematic with adapter board

Trimming the RC Clock

Due to the 32 kHz oscillator on the adapter board in the cable (figure 6), the command line is available via UART after flashing FORTH. By using a terminal emulator a program can now be uploaded, which is then compiled into RAM — the advantage of the Von Neumann CPU. The routine `RXD#` is a counter which determines the width of a low pulse in the ASCII character U (`55h`); here it happened to be the value `33h`. Indeed, the resolution is low, because it is a slow controller — the disadvantage of the Von Neumann CPU.

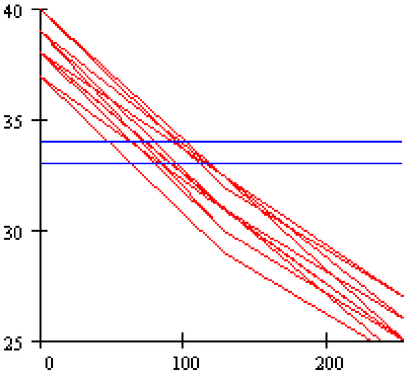


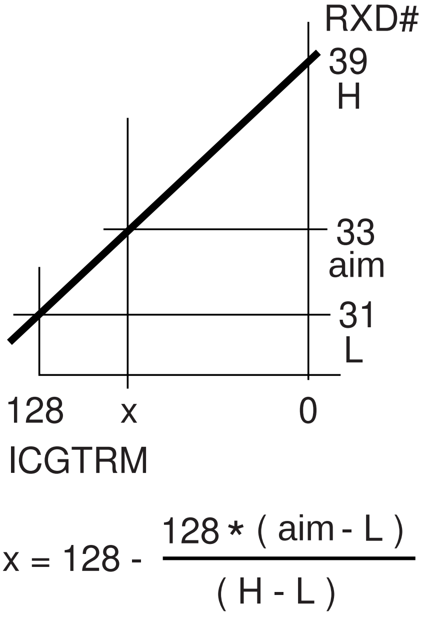
Figure 7: Scattering of the RC clock

The trim factor `ICGTRIM` of the RC clock is adjustable between `00`...`FF`. The characteristic curve is somewhat sagging (figure 7). Here the target value is therefore not calculated from the straight line from `00` to `FF`, but from `00` to `80h` (figure 8). The target

---

[1] BackGround Debug

value at `70h` is stored in the EEPROM. In a second step nanoFORTH is patched so that the AW60 starts with the RC oscillator after reset.



$$x = 128 - \frac{128 * (\text{aim} - L)}{(H - L)}$$

Figure 8: Calculation of the correction

FTDI

Nowadays, the RS232 interface, which is common for terminal programs, has become rare on PCs. The replacement by USB[2] in form of a FTDI[3] cable has the advantage that now the 5 V supply from the PC becomes possible. The disadvantage for 5 V controllers is that RXD/TXD are rated at 3.3 V. Since a level shifter is necessary for 5 V controllers, it has proven its worth to use an ADUM1301, which takes care of the galvanic isolation (figure 9). By doing so, one is no longer connected to the protective conductor via the PC and has thus eliminated a source of interference that makes analog measurements difficult. The usual disadvantage of FTDI cables is that the cheap ones don't work. This especially affects the XON/XOFF handshake as required by the nanoFORTH. The expensive 5 EUR Deek Robot (figure 10) from China have proven themselves, though. However, the connector pinout is not uniform. The in–house standard used here (figure 11) is only one of many possible ones. After all, the contacts on the Dupont sockets can be easily swapped. The connector is not reverse polarity protected. Usual solution: cut off an unused pin and plug its socket.

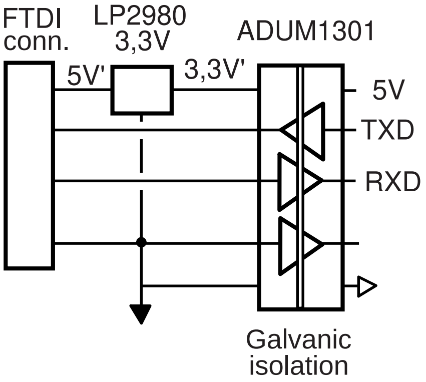


Figure 9: Level shifter

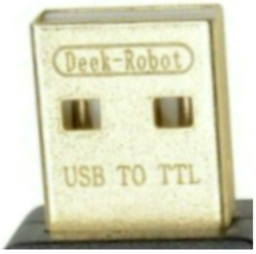


Figure 10: Deek Robot plug on FTDI cable

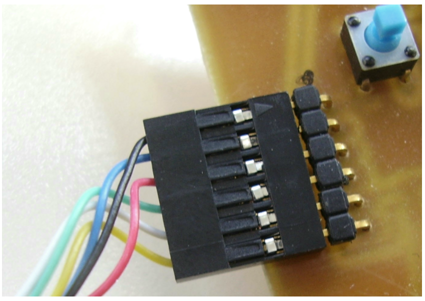


Figure 11: FTDI Dupont connection

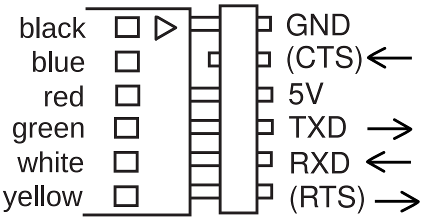


Figure 12: FTDI connection scheme

Listing 1

```
<| \ calc TRIM and store in EEPROM

FFFF 812E E!   \ erase EEPROM

FCODE >C !     \ compile to RAM

:CODE RXD#          \ (  -- UC1 )
     N       CLR,
1 $:
1 $  PE 1    BBS,   \ wait RXD low
3 $:
2 $  PE 1    BBS,   \ 5cyc
     N       INC,   \ 1
3 $          BNE,   \ 3
     N FF #. MOV,
2 $:         DEX,
     N       LDA,
         0,X STA,
             DEX,
         0,X CLR,
             RTS,
CODE;

:CODE CLK=RC    \ switch clock to RC
2 $:
48 B% 00101000 #. MOV,  \ ICGC1
49 B% 01010010 #. MOV,  \ ICGC2
1 $: 1 $    4A 3  BBC,
            48    LDA,
   B% 00101000 #. CMP,
2 $               BNE,
                  RTS,
CODE;

:CODE CLK=32kHz \ switch clock to 32k
2 $:
48 B% 00111000 #. MOV,  \ ICGC1
49 B% 00000000 #. MOV,  \ ICGC2
1 $: 1 $     4A 3 BBC,
            48    LDA,
            38 #. CMP,
2 $               BNE,
                  RTS,
CODE;
```

---

[2] Universal Serial Bus

[3] Future Technology Devices International

```

: (R)  \ ( UC1 -- UC2 )
CR DUP CH. 4E C!
." press U " D% 100 MSEC
CLK=RC D% 100 MSEC RXD# TERMINAL?
DROP CLK=32kHZ D% 1000 MSEC ;

\ Z = D% 31
\ TRIM = 128 - [ 128 *
\ [ D% 33 - @80 ] / [ @0 - @80 ]]

: RUN    \ ( -- UC1 )
D% 128 80 (R) DUP ND.
D% 33 OVER - D% 128 U*
ROT 00 (R) DUP ND. SWAP - U/ -
DUP NH.
812E F! \ store in EEPROM
COLD ;  \ exit via Reset

\ RUN
\ press U     33  \ soll 32kHz Ref
\ 00 press U  40  \ ist mit RC=00
\ 80 press U  32  \ ist mit RC=80

|>
```

Listing 2

```
<| \ patch firmware

2852 SYS-EE E!

FFFF 80E0 E! \ clear EEPROM
FFFF 80E2 E!
FFFF 80E4 E!
FFFF 80E6 E!

80E0 >C !  \ compile to EEPROM
[CODE
812F LDA, 4E STA,  \ copy TRIM
F7AA         JMP,
CODE]

\ redirect Reset-Vector in EEPROM
CCCC SYS-EE 7C + E! \ JMP,-Opcode
80E0 SYS-EE 7D + E!

COLD  \ restart via Reset
|>
```

Continuation of “Letters to the Editor” from page 8

Impressions from the Meeting of the German “Forth Gesellschaft”

On the first weekend in May, the time had finally come. After the meeting had to be postponed from April to May for organizational reasons, just over a dozen Forthers met via screen, camera and microphone to spend a few pleasant hours together on the subject of Forth. The meetings of the Forth Gesellschaft are always open to the public, so non–members can also participate.

Friday: Chatting and Warming Up

Traditionally, there is an informal meeting on the first day, i.e. a get–together without a special program. So it was this time. These are opportunities for casual conversation and to share information about the latest projects. For me, this is one of the highlights of the conferences.

Saturday: Presentations

Saturday was the main day. I couldn’t get out of my state of astonishment. Very interesting lectures were offered. The presentations were “broadcast” live on the Twitch streaming platform. Unfortunately, they can only be viewed there for a limited period of time; after that, they are automatically deleted by the operator. It’s a good thing that Gerald “saves” them beforehand and uploads them to YouTube. So you can watch them even years later. The link to the channel is `https://www.youtube.com/@4ther-tv/videos`

On Saturday evening we had two curiosities: the common dinner and a little later the Dragon Council. It’s funny to eat sitting in front of a camera while watching the others eating and having good conversations. Who would come up with such an idea? The Dragon Council is also such an idea. By the way, this is the only part of the meeting that is not open to the public. This is where our club mascot Swappy chooses a new place to stay for a year with a worthy Forther.

Sunday: Members Meeting

Not only the conference, but also the members’ meeting is open to the public. If you think that our meeting is such a boring, obligatory event with a lot of “club formalities”, you are in for a disappointment. What the protocol of the meeting (printed in this magazine) unfortunately does not provide are the details of the individual items. The reports on the status of existing projects and ideas for new projects together with the subsequent “brainstorming” should be emphasized.

An important point of the general meeting was the presentation of the Dragon Award to our Dutch friend Willem Ouwerkerk. Philip Zembrod held the laudation. Van harte gefeliciteerd, Willem!

It was an enjoyable event.

Wolfgang Strauß

MC9S08AW Flasher

Rafael Deliano

The commercial offer of programming devices is poor for many controllers. That's where DIY is the better solution. An unprotected printed circuit board is sufficient to function, but packaging in a robust housing is usually required for use in production.

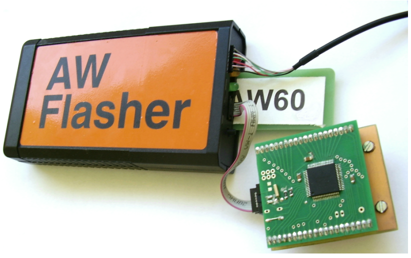


Figure 1: Flasher with needle adapter for SBC

Setup

The FTDI cable to the PC supplies the 5 V for operation (figure 5). After power–up the GP32 SBC runs with this voltage. But it can reduce its supply via pin PD5 to 3.3 V in case the target PCB needs this voltage — some of the targets are not 5 V tolerant.

To protect both PCBs, the short–circuit current is limited, in this case by means of a resistor R1=4.7 Ohm. The value of this resistor must be adjusted to the current gain of the transistor used (figure 8). The resulting output voltage can be read at the A/D converter of pin PB2. Thus the error detection — mostly twisted plugging of the 6–pin cable — is possible from the program. The port pins for the BGD connector are buffered with ICs of type 74HCxx (figure 4). Thus they are more robust against ESD[1], have a better signal quality due to the Schmitt triggers of the receivers and a higher driver performance of the transmitters. All transmitters are disabled after reset to protect the target PCB.

Debug UART

The RXD/TXD at the BGD connector are mainly used to adjust the RC clock in the AW60 or PA60 controller[2], because for the BGD the bus clock of the GP32 must fit exactly. But with this "odd frequency" the UART does not work. Therefore its baud rate is fed to pin PA6. This allows the half–duplex operated UART to synchronize itself in software.

Memory

Two alternatives have been implemented. The EEPROMs (figure 7) are located on a pluggable memory card. Since the GP32 has no I^2C interface, they are controlled via port pins — not ideal for speed.

5 V FRAMs (figure 6) have a capacity of only 32 kByte, so two ICs are needed. In addition, they are only available in SO8 packages and not in DIP8. Therefore they were mounted on adapters. Since they are controlled via the SPI of the controller, very fast reading and writing is possible. However, it turned out later that the download via a terminal program in Win 11 is slow, so that the FRAM variant was not used after all.

Mechanics

The BOS752 enclosure from Bopla[3] belongs to a series (table 1) that also includes variants for displays. However, a display was unnecessary this time, as was the 9 V battery compartment.

Nr.	Type of enclosure
BOS750	Panoramic display screen
BOS751	Display screen
BOS752	Membrane keypad area

Table 1: Bopla enclosures

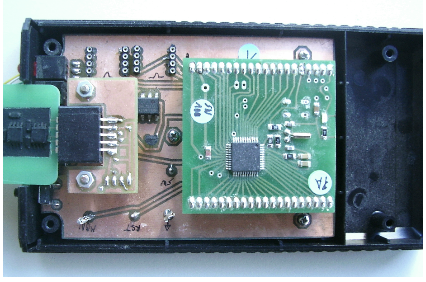


Figure 2: Main board in the upper shell

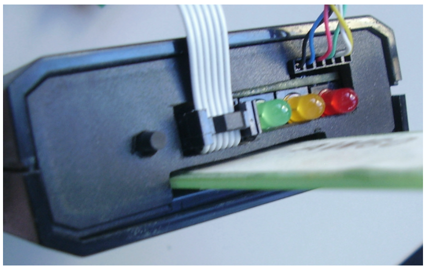


Figure 3: Front panel

One advantage of compatible enclosure families is that you can realize devices for different requirements faster, since new layouts can be derived from old ones. This type of enclosure is not cheap, but it is very spacious. And ABS is more pleasant to work with than cheap plastics. Since no membrane keyboard is used here, the recess on the top can be used for labeling (figure 1). This is useful because different firmware is installed for AW60 and the newer PA60, depending on the controller.

---

[1] ElectroStatic Discharge

[2] PA is the newer generation of Freescale/NXP controllers; it is "incredibly cheap" but currently not readily available. The smaller packages are also less attractive for hand soldering. For hobby and often for industry as well, newer is not always better. (RD)

[3] Bopla Enclosure Systems GmbH, located in Bünde, Germany

[4] Brother ScanNCut SDX1200 cutting plotter

The vinyl film was cut by a SDX1200[4] and transferred with transfer foil.

All controls are mounted on the front panel. The supplied ABS plate from Bopla was used. Print the outline on self–adhesive vinyl film, stick it on, drill and saw the contour with the fretsaw. Until everything fits correctly, however, a rework with the file is still due. Then you have a presentable prototype. For small series, however, more convenient alternatives are necessary. Since a standard 1.5 mm PCB will fit, it would be possible to have it made with the other PCBs and stick foil on it. If one has a milling machine, it is more obvious to machine ABS or colored GRP plates.[5] A blue 5 watt laser that can cut plastic is probably ideal.

GP32 SBC

Why is the antique GP32 from 2001 used to program the oldtimer AW60 from 2006? The BGD protocol (figure 9) of the AW60 has a timing which can be reproduced even in assembler only if host and target work with the same clock. The target, however, uses an RC oscillator whose frequency deviates. The frequency of the AW is about 4 MHz, the frequency of the newer PA from 2012 is about 8 MHz. So it is necessary to use a controller whose clock can be precisely adjusted via PLL[6]. This is only the case with the GP32.[7] Complicated setup, but many setting options. Especially the 10–bit frequency divider N with high resolution (figure 10) is useful. For 2.45. . . 8 MHz one works with the prescaler P with the setting P=/1, for up to 16 MHz with P=/2. Above 10 MHz, however, one is nominally already outside the specified speed of the GP32.

SYNC

The SYNC sequence is provided in the BGD protocol for synchronization (figure 9). The host first sends an over–wide pulse and then receives back from the target a pulse with exactly 128 etu[8]. In software, one needs a subroutine (listing 1) that scans the port pin after a programmable delay. A search routine can then determine exactly the rising edge. For the required high temporal resolution, a list of 256 NOP opcodes, the beginning of which is skipped by a programmable JMP opcode, is suitable.

Needle Adapter

The question remains where the pin mentioned at the beginning is and why it exists at all. The 6–pin header costs money, causes problems in some applications with the height and is not swap–proof. Therefore it is no longer used on the SBCs, but instead a small needle adapter with only one needle for the BKGD pin is used (figure 12). The other signals are already present on the 40–pin header.

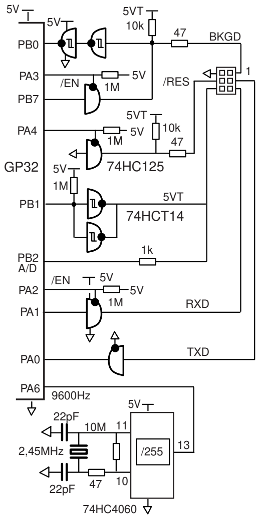


Figure 4: BGD interface

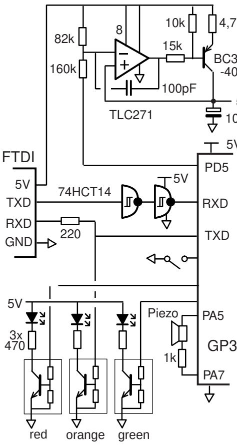


Figure 5: Power supply, FTDI, controls

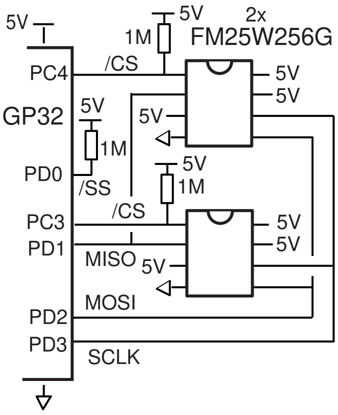


Figure 6: FRAM

---

[5] Acrylonitrile butadiene styrene copolymers (abbreviated ABS) are thermoplastic terpolymers available as sheets from 1 mm thick. GRP stands for “Glass fiber Reinforced Plastic.

[6] Phase Locked Loop

[7] This was the first generation with PLL

[8] elementary timing units

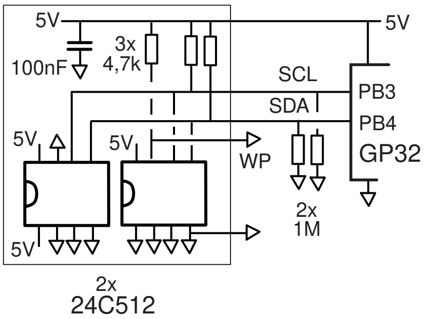


Figure 7: EEPROM on chip card

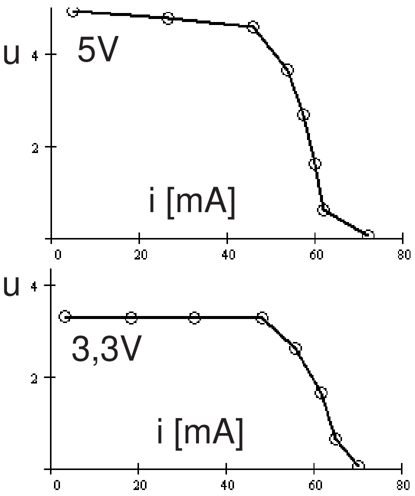


Figure 8: Current limitation

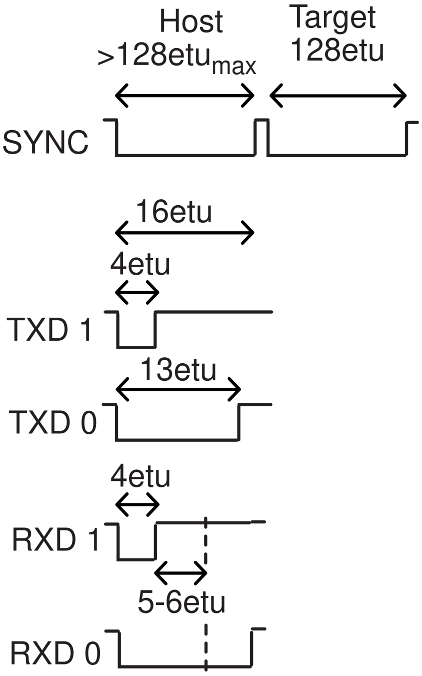


Figure 9: Data transmission

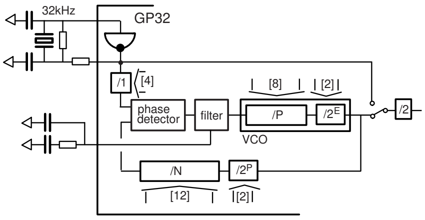


Figure 10: Clock unit of the GP32

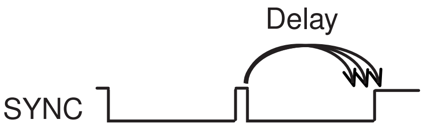


Figure 11: SYNC sampler

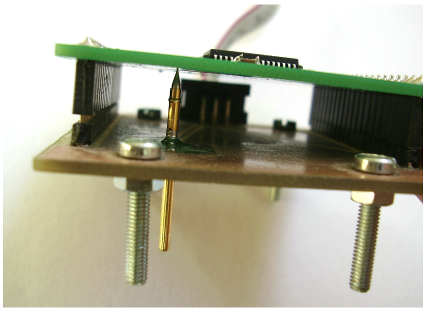


Figure 12: Needle adapter for the BKGD pin

Listing 1: SYNC Sampler

```
:CODE SYNC’    \ ( UC1 -- flag ) UC1  = delay
                \         flag = level at BKGD-pin
      1 ,X          LDA,
XSAVE               SHX,
                    TAX,   \ stack to X-register
                 H. CLR,
      BKGDo         MBC,   \ pulse low
      FF         #. LDA,
1 $: 1 $         A. DBN,   \ 3
      BKGDo         MBS,   \ 4cyc
      BKGDo-en      MBS,   \ 4cyc
      NOP, NOP, NOP, NOP,  \ 4cyc
>C @  3 +        ,X JMP,   \ 6cyc
\ Insert 256 * 1cyc
\ i.e. 256 NOP Opcodes
\ Note: In 8MHz-Target  128 * 1cyc will do.
NOP, NOP, NOP, NOP, NOP, NOP, NOP, NOP,
...
NOP, NOP, NOP, NOP, NOP, NOP, NOP, NOP,


      BKGD-PORT     LDA,   \ 3cyc  sample pin
2 $:  2 $  BKGDi    BBC,   \ wait for rising edge
      BKGDo         MBS,
      BKGDo-en      MBC,
              01 #. AND,   \ pin to stack
XSAVE               LHX,
      1 ,X          STA,
                    RTS,
CODE;
```

Large Literals for the RP2040[1] Assembler

Willem Ouwerkerk

Only literals from 3 to 8 bits can be used directly in the assembler for the RP2040. So for 16–bit and 32–bit numbers we need a different method. The classic way with ARM are "literal pools".

What is a Literal Pool?

> "Perhaps the most common type of literal pool are those used by the LDR Rd,=const pseudo–instruction in ARM assembly language and similar instructions in IBM System/360 assembly language, which are compiled to a LOAD with a PC–relative addressing mode and the constant stored in the literal pool." `https://en.wikipedia.org/wiki/Literal_pool`

In Forth we use literals to push integers onto the data stack as quickly as possible. The routine for this must therefore be in machine code, but which one is the best, short and quick?

In combination with the LDR opcode in the M0+ instruction set, there is a label mechanism that enables literal pooling:

```
LDR Rn, =ape
```

or this:

```
LDR Rn, =0x12345678
```

However, this would mean that a fairly large assembler would have to be loaded in (no)Forth. Because the literals could only be resolved after the assembled code was complete. A difficult task for a single–pass assembler! So we tried forth–like alternatives.

Let's Use It!

For *noForth T*[2] we can use a register that is already used by `NEXT`.

`NEXT` goes like this:

```
: NEXT, ( -- )
  ip { w } ldm,
   \ Read CFA to W & increase IP
  w { hop } ldm,
   \ Read code address to HOP & increase W
  pc hop mov, ;
   \ Move code address to PC
```

As you can see, the W register is loaded with the address of the code field. After reading the contents to the HOP register, the W register is increased to right behind the code field (CFA). See figure at the end of the article.

Single and Multiple Inline One–Pass Pooling

We can take advantage of this, here is an example:

```
code COLD
  10000047 ,
  code>
    hop w ) ldr,
    hop bx,
  end-code
```

In this case the W register points to the data that is compiled right after the CFA of cold. The word `CODE>` is a directive that redirects the CFA to where the real code starts. Because the W register points directly to the data, we can simply read it to any register we want. In this case the HOP register. Finally `HOP BX,` jumps directly to `10000047` which is the address of the secondary boot code to restart noForth T.

We can also use this to read a whole range of data into registers, for example like this:

```
code SET-GPIO ( -- ) \ GPIO 2 to 29 are inputs
 D0000020 , \ = HOP
 40014014 , \ = DAY
 code>
   w { hop day } ldm,
   \ HOP = D0000020, DAY = 40014014
```

Which in one 16–bit opcode reads one or more numbers, to as many registers, and it's fast, too! However, this is of no use in subroutines or interrupts.

The next example gives another way to store and read numbers.

`pool,` and `apool,` are two noForth T assembler directives that allow the use of large literals in the middle of any assembler code. They are used as follows:

```
code ANYWORD ( -- )
 tos sp -) str, \ Save TOS
 apool, \ Start new aligned pool
   87654321 , \ Literal 0
   10000100 , \ Literal 1
   true ,     \ Literal 2
 then,
 tos sp -) str, \ Save TOS
 w { tos hop day } ldm,
  \ Read all three literals in one go
   \ TOS = lit0, HOP = lit1, DAY = lit2
```

---

[1] **R**aspberry **P**i MCU; **2** Cores, Typ M**0**+, **4***64 KB (+ 8 KB for a total of 264 KB RAM), further 4 KB for USB and a 16–KB XIP cache, **0** Flash on chip.

[2] T stands for Thumb, and of course the T of Ford T. It is a noForth for the Thumb2 instruction set. And it runs on all Pico boards because they all use the RP2040.

```
 ...
 next, end-code
```

The literals are compiled between the words `apool>` and `then,` but how does that work?

`apool,` saves the program counter PC in the W register. Then it jumps over the *inline data pool* using `ahead,` and `then,` .

The directives `pool,` and `apool,` are:

```
: POOL, ( -- )
  w pc mov, ahead, ;
   \ Start not aligned literal pool
: APOOL, ( -- )
  align, pool, ;
   \ Start aligned literal pool
```

So we have a set of tools to use fast 16–bit and 32–bit literals in the Forth assembler without having to use complex mechanisms.

Another Simple Literal Pooling Method

There is a third mechanism that is only useable for single literals and it's a bit less efficient. It looks a bit like the traditional ARM notation. I call it *simple literal pooling*:

```
code SOMEWORD ( -- n )
 tos sp -) str,
  \ Save TOS
 tos pc ) ldr, 12345678 ##
  \ Load 12345678 to TOS ...
```

It assembles to:

```
code SOMEWORD ( -- n )
 sp sp 4 # subs, \ Save TOS
 tos sp 0 x) str,
 tos pc 0 x) ldr, \ Load inline number to TOS
 ahead,
   12345678 , \ Inline number
 then,
 ...
```

Finally, a few more examples from the noForth T target file.

Real Examples

The first word resets most built–in RP2040 devices.

```
code RESET-ALL      ( -- )
    4000C000 ,  4000E000 ,  01FFCDBF ,  4000F000 ,  003C7FFE ,
code>   \ DAY = 4000C000, SUN = 4000E000, MOON = 1FFCDBF
    w  { day sun moon } ldm,
    moon  sun ) str,        \ Set most built in devices in reset state
    w  { hop moon } ldm,    \ HOP = 4000F000, MOON = 3C7FFE
    moon  hop ) str,        \ Restart most built in devices
    begin,
        w  day 8 x) ldr,    \ Read reset done register to W
        w w mvns,
        w moon ands,        \ Ready when all tested bits are zero?
    =? until,
    next,
end-code
```

The `key` primitive, it reads one key to TOS using UART0:

```
code KEY)    ( -- c )
    40034000 ,  \ UART0 base address
code>
    sun  w ) ldr,
    tos  sp -) str,
    begin,
        day  sun 18 x) ldr, \ Read flags (UARTFR)
        moon 10 # movs,     \ Isolate KEY? flag
        day moon ands,
    =? until,               \ Flag set?
    tos  sun ) ldrb,        \ Read character
    next,
end-code
```

Calculate the depth of the (no)Forth stack used:

```
code DEPTH  ( -- n )     \ ok
    ramadr: SP0 ,        \ Pointer to bottom of datastack
code>
    day  w ) ldr,        \ 2 - Read SP0-pointer to DAY
    day day ) ldr,       \ 2 - Read stack base to DAY
    day day sp subs,     \ 1 - Calc. used stack space
    tos  sp -) str,      \ 3 - Save TOS
    tos day mov,         \ 1 - Used space to TOS
    tos tos 2 # asrs,    \ 1 - Divide by 4
    next,                \ 6
```

Links

You can find the complete noForth T assembler in the wiki:
`https://wiki.forth-ev.de/doku.php/pfw:assemblers_rp2040-assembler`

Cortex–M0+ Technical Reference Manual r0p1
`https://developer.arm.com/documentation/ddi0484/c/CHDCICDF`

ARM Compiler Toolchain Assembler Reference Version 5.03
`https://developer.arm.com/documentation/dui0489/i/arm-and-thumb-instructions/ldr--immediate-offset-`

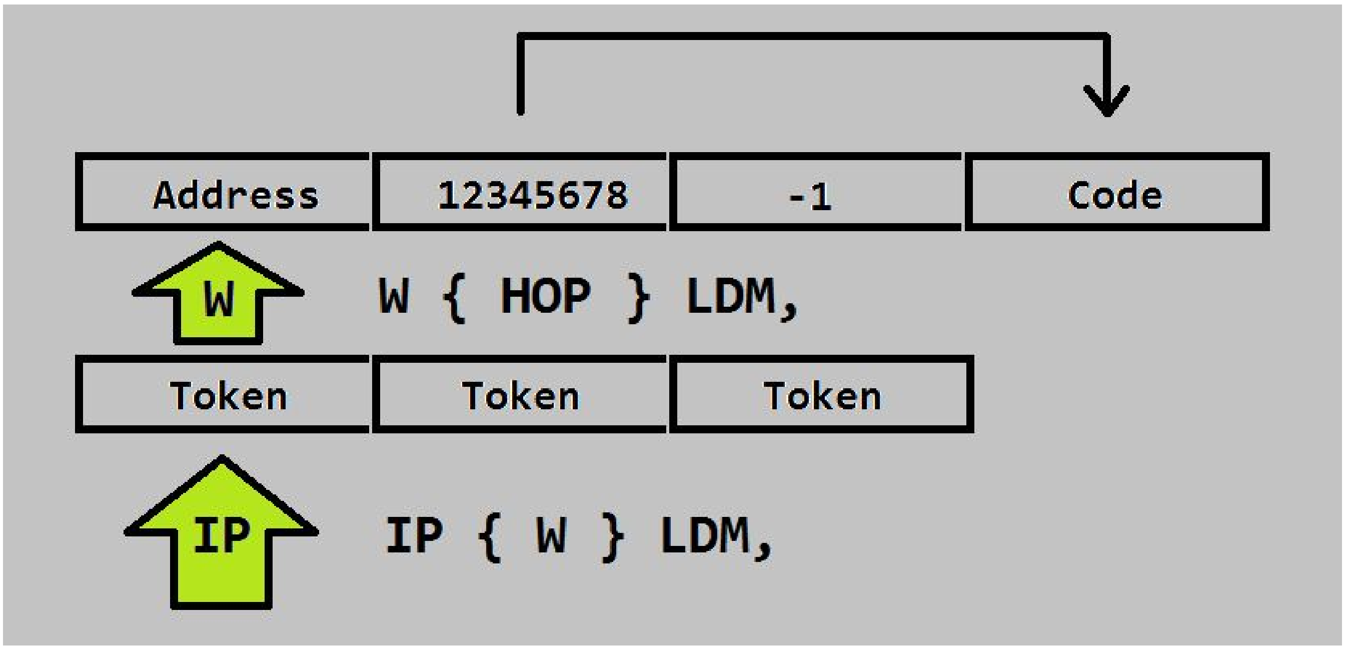


Figure 1: Fast long literal in noForth T

Astroimaging in Forth: Image File Format and Camera SDK

Andrew Read

Last time we spoke about astroimaging in general and specifically the control of the telescope mount over TCP/IP. Today I would like to share with you my progress in controlling a specialized camera with the manufacturer's SDK and saving images in an industry standard file format.

FITS is the longstanding astroimage file format. Its drawback is that 16–bit integers (most cameras capture pixel luminescence values as 16–bit integers) are stored in big–endian format. Little–endian CPUs must byte swap every word of the image buffer with each load or save. To illustrate the computational burden: the camera I am using, an ASI6200 (figures 1 and 2), has 62 million pixels, i.e. 62 million words in the image buffer! A newer standard for astroimages, XISF, which permits either endian format, but which is little endian by default and backwardly compatible with FITS metadata, is becoming more popular as a result.

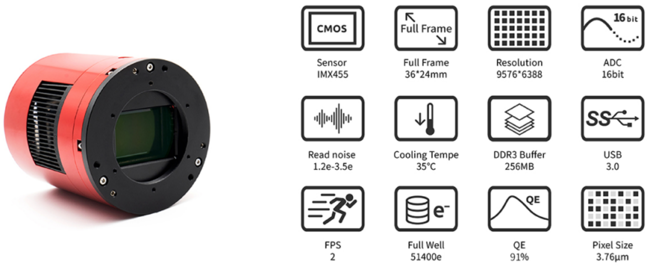


Figure 1: ASI6200, picture and features

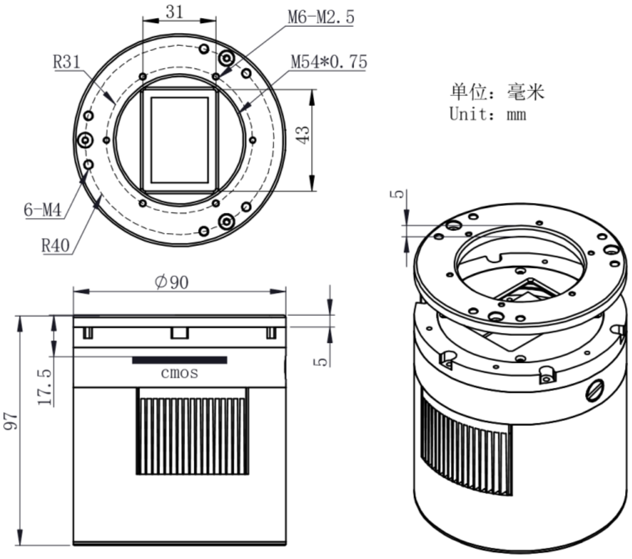


Figure 2: ASI6200, some technical info

At first glance of the XISF standard I wasn't optimistic. The file format is very flexible: XML is used to hold metadata, image data can be chained across several files, there are optional checksums, data may be encrypted. How would XISF be approached in C? First we would look for (or take some time to write by ourselves) an XML library. Next, study the XISF standard documentation and build "business logic" to deal with all the XISF file structural possiblities. Obtain a variety of XISF files and test our code reading and writing them. Then I recalled a conversation with Ulli Hoffmann — "in C you build a framework to deal with the general case, in Forth you build exactly and only what you need". Let's proceed that way …

First step: create an XISF image file with my camera using the manfacturer's software and open it in a hex editor (I use UltraEdit). Compare the file with the XISF standard and delete everything that is optional according to the standard and unnecessary to me. Save the abbreviated file, and confirm that it opens successfully in PixInsight (software for processing XISF images). Extract the XML part of the file and inspect the XML tags (UltraEdit again). Decide which parts of the XML are effectively just "constant text" and which parts should be written uniquely for each image. Finally, develop a lexicon of Forth words (`XISF.f`) to create this "theoretical minimum" XISF file: "signature" followed by "header" followed by image data. Here is how we proceed to initialize a buffer in the format of an XISF file.

```
XISF_BUFFER BUFFER: XISFBuffer
XISFBuffer XISF.StartHeader
    XISF.StartXML
        XISF.StartImage
```

XISF files include "FITS keyword" metadata carrying information about the image in string format within XML tags. The most common FITS keywords cover, for example, the target coordinates (RA DEC), date and time, exposure duration, equipment, etc. We use a Forth defining word to associate a FITS keyword with a variable, in this case an integer representing the focus position.

```
variable fpos 2000 fpos !

s" FOCUSPOS" fpos
 XISF.MAKE-FITSKEY-INT
 XISF.FITSfocuspos
```

Writing the focus position to the header is done by `XISF.FITSfocuspos`. Other words from the lexicon complete the header. The structure of the Forth code mirrors the structure of the header — the result is simple and logical.

```
XISFBuffer XISF.StartHeader
    XISF.StartXML
        XISF.StartImage
            XISF.FITSfocuspos
            \ write other FITS keywords
        XISF.FinishImage
    XISF.FinishXML
XISF.FinishHeader
```

Before the final step of writing the buffer to a binary file the image data is obtained by calling the camera SDK with a pointer to the appropriate section of the buffer.

ZWO, the manufacturer of ASI imaging cameras supplies an SDK comprising a DLL, a C–language header file, and documentation. I had anticipated that it would be rather difficult to use all this from Forth but in fact it was very straightforward. Firstly, to confirm that the SDK “worked” and could connect to a camera I used Microsoft Visual Studio 2022 and prepared a simple C–language console application to call the functions in the DLL. VS makes this straightforward, but all of the templates are in C++ rather than C and need to be adapted — mainly with the delete key.

VFX Forth interfaces with DLLs through library and Extern: words, best understood by reviewing the listing (`ASI_SDK.f`). A few notes: (1) the DLL is most simply placed in the directory of the VFX executable. (2) the VFX developer console has a tool to extract all of the functions from DLLs and copy them to the clipboard to save time. (3) it is actually very easy and quick to write the Extern: function declarations by hand, referencing the C header file; an automated tool is probably not worth the trouble. (4) explicit whitespacing in the declarations is essential, even between the closing parenthesis and the semicolon.

ASI SDK functions generally take pointers to variables and structures and return an IOR–type error code. I prepared a Forth word to decode those error codes (referencing the SDK), and a simple succeed–or–abort error handler for exploring the SDK functionality at the interpreter. C–language structures are easily mimicked with the Forth `BEGIN-STRUCTURE` and releated words. I made two interesting observations:

1. the C compiler may pad structures for alignment but this must be compensated for manually in Forth.
2. C’s `sizeof()` does not always give the correct answer. A C double was reported as 8 bytes but compiled as 12 bytes (a long double).

I think you are already aware of these C–language data size quirks. In Forth it was not hard to explore and resolve the issues with `DUMP` but I guess in C my code would have just failed and I’d be turning to a debugger to set breakpoints and single–step. `BEGIN-STRUCTURE` also inspired a thought to mirror C enums with a word `BEGIN-ENUM`. This is certainly not original but it’s interesting how Forth encourages just–in–time reinventions!

I can now operate the ASI camera and save the images in XISF standard format using Forth. Some open–source repositories do the same thing in C++ or Python with, literally, thousands of lines of code. In Forth “first–level” functionality required roughly 250 lines and a couple of afternoons. The next step is to develop a camera control language at a level of abstraction about the ASI SDK, reflecting how I might like to interact with the telescope and camera during a live imaging session.

The code is presently in private repositories on GitHub but will become open source. Please contact me if you would like access and I welcome correspondence about the project in general. `andrew81244@outlook.com`

Links

XISF format:
`https://pixinsight.com/doc/docs/XISF-1.0-spec/XISF-1.0-spec.html`

ASI SDK:
`https://www.zwoastro.com/downloads/developers`

Listing

```
\ XISF.f - code for writing in the XSIF image file format

4032 constant XISFHeaderMaxLen               \ review and adjust
640 480 2 * * constant XISFDataMaxLen        \ camera image size in bytes

BEGIN-STRUCTURE XISF_BUFFER
\ a 'theoretical minimum' XISF file
   8 +FIELD XISF_SIGNATURE
   4 +FIELD XISF_HEADER_LEN
   4 +FIELD XISF_RESERVED
   XISFHeaderMaxLen  +FIELD XISF_HEADER      \ header with trailing zeros
   XISFDataMaxLen    +FIELD XISF_DATA        \ data with trailing zeros
END-STRUCTURE

variable XISFBufferPointer
variable XISFHeaderPointer

```

```
: XISF.StartHeader ( XISFbuff -- )
   dup XISFBufferPointer !
   XISF_Header XISFHeaderPointer !
;

: XISF.HeaderLength ( -- n )
   XISFHeaderPointer @ XISFBufferPointer @ XISF_Header -
;

: XISF.WriteToHeader ( addr n -- )
   dup >R
   XISFHeaderPointer @ swap cmove
   R> XISFHeaderPointer +!
;

: XISF.WriteIntToHeader ( x -- )
\ write an interger in string format - for XML tags
   <# dup SIGN 0 ( x-as-double) #S #> ( caddr u)
   XISF.WriteToHeader
;

: XISF.StartXML ( -- )
\ XISF specification requires this
   s\" <?xml version=\"1.0\" encoding=\"UTF-8\"?>" XISF.WriteToHeader
   s\" <xisf version=\"1.0\">"      XISF.WriteToHeader
;

: XISF.StartImage
   s\" <Image geometry=\""          XISF.WriteToHeader
   \ ... write XISF defined parameters in XML (e.g. the image size)
;

: XISF.MAKE-FITSKEY-INT ( caddr u addr <name> -- )
\ defining word for a FITS keyword having integer value
\ e.g. variable-name S" keyword" XISF.MAKE-FITSKEY-INT <name>
   CREATE
      , $,
   DOES> ( -- , write a FITS keyword with a value in XML)
      dup >R @ @                    ( value)
      R> cell+ count                ( value caddr u)
      s\" <FITSKeyword name=\""     XISF.WriteToHeader
      ( value caddr u)              XISF.WriteToHeader
      s\" \" value=\""              XISF.WriteToHeader
      ( value)                      XISF.WriteIntToHeader
      s\" \" />"                    XISF.WriteToHeader
;

: XISF.WriteFile ( caddr n --)
\ write the buffer to a binary file
   \ ... create-file ... write-file ... close-file
;

\ a test run
XISF_BUFFER BUFFER: XISFBuffer
variable fpos 2000 fpos !
s" FOCUSPOS" fpos XISF.MAKE-FITSKEY-INT XISF.FITSfocuspos

XISFBuffer XISF.StartHeader
   XISF.StartXML
      XISF.StartImage
         XISF.FITSfocuspos
         \ ... write other FITS keywords
      XISF.FinishImage
   XISF.FinishXML
XISF.FinishHeader
\ ... call the ASI SDK with a pointer to the buffer and expose an image
s" C:\test\MadeInForth.XISF" XISF.WriteFile

\ \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\ ASI_SDK.f - code for interfacing to the ASI SDK

LIBRARY: ASICamera2.dll    \ place the dll in directory of the VFX executable

Extern: int "C" ASIGetCameraProperty( int * ASICameraInfo , int CameraIndex ) ;
Extern: int "C" ASIGetNumOfConnectedCameras( void ) ;
\ ... around 20 further Extern: function declarations in this SDK
```

```
: ASI.Error ( n -- caddr u)
\ convert and ASI error code to text
   CASE
    0 OF s" SUCCESS" ENDOF
    1 OF s" INVALID_INDEX" ENDOF
    2 OF s" INVALID_ID" ENDOF
   \ ... other codes
   s" OTHER_ERROR" rot ( care of case selector)
   ENDCASE
;

: ASI.?abort ( n --)
\ do-or-die error handler, for interactive testing
   dup ASI.Error type CR
   IF abort THEN
;

BEGIN-STRUCTURE ASI_CAMERA_INFO     \ 240 bytes
64 +FIELD ASI_CAMERA_NAME
 4 +FIELD ASI_CAMERA_ID
 4 +FIELD ASI_MAX_HEIGHT
 4 +FIELD ASI_MAX_WIDTH
12 +FIELD ASI_PIXEL_SIZE            \ long double
   \ ... other fields
END-STRUCTURE

\ define an ENUM data-structure with similar syntax to a STRUCTURE
: BEGIN-ENUM 0 ;
: END-ENUM drop ;
: +ENUM dup 1+ swap CONSTANT ;

BEGIN-ENUM ( ASI_IMG_TYPE )
   +ENUM ASI_IMG_RAW8
   +ENUM ASI_IMG_RGB24
   \ ...
END-ENUM
-1 CONSTANT ASI_IMG_END

ASI_CAMERA_INFO   BUFFER: ASICameraInfo
VARIABLE CameraID

: ASI.ReviewCameras
\ get and print all the properties of all the connected cameras
   ASIGetNumOfConnectedCameras ( -- n)
   ?dup
   IF
      \ loop over each connected camera
      0 do
         ." Camera index " i . CR
         ASICameraInfo i ASIGetCameraProperty ( buffer index --) ASI.?abort
         ASICameraInfo ASI_CAMERA_NAME ." ASI_CAMERA_NAME " zcount type CR
         ASICameraInfo ASI_CAMERA_ID ." ASI_CAMERA_ID " l@ dup CameraID ! u. CR
         ASICameraInfo ASI_MAX_HEIGHT ." ASI_MAX_HEIGHT "  l@ . CR
         \ ... other properties
         CR
      loop
   ELSE
      ." No ASI cameras connected" CR
   THEN
;

\ a test run
." ASICamera2.dll load address " ASICamera2.dll u. CR
.BadExterns CR \ list any unrecognized extern functions
ASI.ReviewCameras
```

Minutes of the General Meeting of the Forth–Gesellschaft e.V. on May 7, 2023

Anton Ertl

1. **Welcome of the Participants and Verification of the Quorum**
   14 participants, thus given.

2. **Election of the Secretary**
   M. Anton Ertl

3. **Election of the Chairman of the Meeting**
   Klaus Kohl–Schöpe

4. **Additions to the Agenda**
   The points raised here were all dealt with under "Miscellaneous".
   Then: award of the Dragon Prize to Willem Ouwerkerk.

5. **Report of the Board of Directors**
   – Membership development and cash report (Carsten Strotmann); 2 resignations, 3 admissions. The cash report was handed in. Details are available on request from the board.
   – The cash auditor (Wolfgang Strauß) was able to comprehend the cash book.
   – Around the Forth magazine (Ulrich Hoffmann).
   – Internet presence: activities on YouTube/Twitch/Website (Gerald Wodni).
   – External representation and projects (Bernd Paysan).

6. **Approval of the Actions of the Board of Directors**
   Anton Ertl requests the discharge of the Board of Directors. Vote on `strawpoll.de`: 13Yes:0No:1Abstention. The Board of Directors was thus discharged.

7. **Election of the Board of Directors**
   The existing Board of Directors is available for a further period. There are no further candidates. Vote on `strawpoll.de`: "Do you want to keep the existing directorate?" 14Yes:0No:0Abstention. The new, old directors accept the election.

8. **Projects**
   – Ulrich Hoffmann reports on "Project Forthworks".
   – Philip Zembrod reports on Volksforth.
   – Wolfgang Strauß reports on "Project Feuerstein".
   – Carsten Strotmann reports on the book project.
   – Anton Ertl reports on "Rethinking Forth".

9. **Miscellaneous**
   – Carsten Strotmann: Increase of the membership fee? Result of the discussion: No.
   – At the moment the VD costs more than is foreseen for the special purpose operation! Result of the discussion: This is acceptable for us.
   – Opt–out for the paper version without reduction of the membership fee? Result of the discussion: We will do it.
   – Bernd Paysan reports about the planned shutdown of the Strato server. Mailman porting. Discussion about switching off the mailing lists.
   – Wolfgang Strauß: English VD. Wolfgang makes once the next three issues also in English. The German issue is unaffected and will be published as usual.
   – Ulrich Hoffmann: Cooperation with the Facebook group Forth2020.
   – Carsten Strotmann suggests to post selected articles as a whole on the website, linking to the VD. Gerald Wodni proposes to post this under a VD logo; accepted.
   – Ulrich Hoffmann: Dagstuhl seminar for concatenative languages; to note.
   – Carsten Strotmann: Summer meeting from June 30 to July 2 at Linuxhotel decided.
   – Carsten Strotmann: Mastodon — new server needed for Forth society.
   – Carsten Strotmann: `cloud.forth-ev.de` currently paid by Carsten; Forth society takes over.
   – Carsten Strotmann: Virgil Dupras (Tumbleforth, CollapseOS). Advertise for him or similar; accepted.
   – Martin Bitter: entry of the Forth company in `handelsregister.de`; for your information.

Addresses and Contact Persons

Regional Forth Groups

Please ask the organisers if the meetings will take place. This may vary depending on the pandemic situation.

Mannheim **Thomas Prinz**
Tel.: (0 62 71) – 28 30$_p$
Ewald Rieger
Tel.: (0 62 39) – 92 01 85$_p$
Treffen: jeden 1. Dienstag im Monat
Vereinslokal Segelverein Mannheim e.V. Flugplatz Mannheim–Neuostheim

München **Bernd Paysan**
Tel.: (0 89) – 41 15 46 53
`bernd@net2o.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00 auf `http://public.senfcall.de/forth-muenchen`, Passwort `over+swap`.

Hamburg **Ulrich Hoffmann**
Tel.: (04103) – 80 48 41
`uho@forth-ev.de`
Treffen alle 1–2 Monate in loser Folge
Termine unter: `http://forth-ev.de`

Ruhrgebiet **Carsten Strotmann**
`ruhrpott-forth@strotmann.de`
Treffen alle 1–2 Monate im Unperfekthaus Essen
`http://unperfekthaus.de`.
Termine unter: `https://www.meetup.com/Essen-Forth-Meetup/`

Services of the Forth–Gesellschaft

Nextcloud `https://cloud.forth-ev.de`

GitHub `https://github.com/forth-ev`

Twitch `https://www.twitch.tv/4ther`

μP–Controller–Pool **Carsten Strotmann**
`microcontrollerverleih@forth-ev.de`
`mcv@forth-ev.de`

Special Fields

Forth hardware in VHDL microcore (uCore)	**Klaus Schleisiek** Tel.: **(0 58 46) – 98 04 00 8**$_p$ `kschleisiek@freenet.de`
AI, Object Oriented Forth, Safety–critical systems	**Ulrich Hoffmann** Tel.: **(0 41 03) – 80 48 41** `uho@forth-ev.de`
Forth distribution volksFORTH ultraFORTH RTX / FG / Super8 KK–FORTH	Ingenieurbüro **Klaus Kohl–Schöpe** Tel.: **(0 82 66)–36 09 862**$_p$

Events

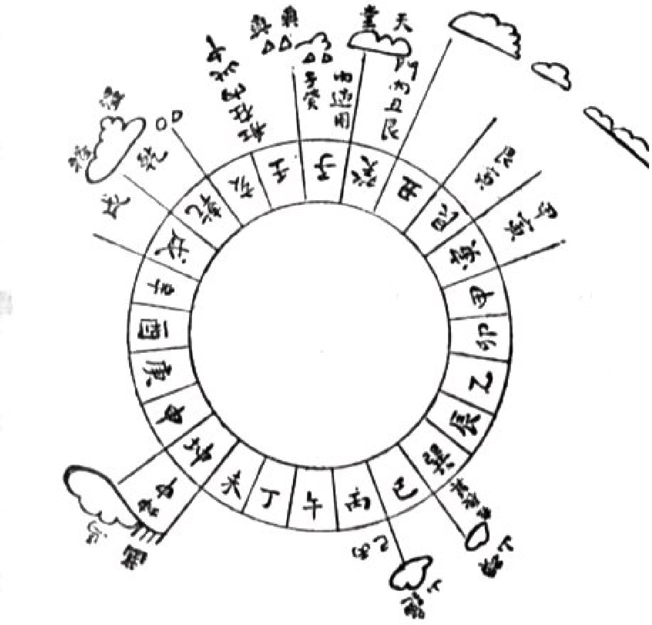
Thursdays from 20:00
Forth–Chat net2o `forth@bernd` using the key `keysearch kQusJ`, complete key:
`kQusJzA;7*?t=uy@X}1GWr!+0qqp_Cn176t4(dQ*`

Every 1st Monday of the month from 20:30
Forth Evening
Video meeting (not only) for beginners.
Info and participation link: Email to `wost@ewost.de`

Every 2nd Saturday of the month
ZOOM meeting of the Forth2020 Facebook group
Participation info: `www.forth2020.org`

Forth Summer Meeting 2023, June 30 to July 02
For details, please see backcover of this magazine.

You can find details on the dates at `http://forth-ev.de`



Would you like to start a local Forth group in your area, or just initiate regular meetings? Or can you imagine providing assistance to Forthers seeking advice on Forth (or other topics)? Would you like to make contacts that go beyond the VD and the annual membership meeting? Just write to the VD — or call — or send us an email!

Notes on the abbreviations after the telephone numbers:
Q = Answering machine
p = Private, outside typical working hours
g = Relating to business
The office addresses of the Forth–Gesellschaft e.V. and of the VD can be found in the imprint of the magazine.

Invitation to the

Forth Summer Meeting 2023 from June 30 to July 02
at the Linux Hotel in Essen–Horst, Germany

The meeting will take place at Villa Vogelsang, Antonienallee 1, 45279 Essen–Horst. The building is located in a park with a view of the Ruhr river — an oasis in the middle of the Ruhr area.

How to get there

Essen can be reached by train, Essen–Horst by S–Bahn; by car via the A40 (Essen–Kray or Essen–Frillendorf). There are at least three airports to choose from: Düsseldorf, Dortmund, Cologne. But they are not “around the corner”. And yes, you can also get there on foot or by bike — is proven to be feasible!

Registration

`https://sommer.theforth.net`

Program (might still change, but in principle relaxed, and always with Forth in it. :)

Friday

morning	Early bird, excursion
from 13:00	Arrival, lobby
18:00	Dinner together
20:30	Workshops

Saturday

09:00	Breakfast
10:00	Meetings and conferences
12:30	Lunch
14:00	Meeting, talking shop
18:00	Dinner together
20:30	Workshops

Sunday

09:00	Breakfast
10:00	Meetings and conferences
12:30	Farewell, departure