# Three Forths Make a Hole

2003 Sep 25
Howerd Oakford howerd@inventio.co.uk
Stephen Pelc stephen@mpeltd.demon.co.uk

## Abstract

There are many theories about the best way to program a computer – the ACE project showed just what can be done with Forth, especially in a situation which was constantly changing.

## Background

Even now, 10 to 20 unexploded WW2 bombs ("air dropped munitions") are found and made safe every year in the UK. After the recent Balkan conflicts, it was estimated that the failure rate (did not explode) of these bombs was about 10-20%, depending on the supplier, leading to an increasing need for disposal equipment that could be easily and rapidly set up. Some of the technology developed for this project is also applicable to landmine clearance. In some countries, especially in Africa, there are still tens of millions of uncleared landmines.

## The Project

The Abrasive Cutting Equipment ( ACE ) was developed as a Custom Off The Shelf (COTS) project. The idea was to buy some easily available pieces, tie them together with a minimum amount of hardware and software glue, and create a prototype. The design requirement was for a complete system that could be used by Armed Forces personnel to cut holes in unexploded bombs. There must be no electrical circuits within 25m of the target, and the cutting head must be controlled from up to 500m away. The cutting must be done using a very fine jet of water and abrasive, powered by a 500 Bar water pump, and must be controlled in four axes (x, y, z and rotate) to within +/- 0.2mm. Since the "safe distance" for a 1000 pound bomb is 1500m, safety is a key element in the design.

## The People

Carl Moorhouse – Chief Designer at Richmond
Gerry Ede – Chief Mechanical Engineer at Richmond
John Winslow – Chief Engineer at Disarmco
Stephen Pelc – MD of Microprocessor Engineering
Howerd Oakford – MD of Inventio Software
And many others, too numerous to mention.

These people made ACE work. We actually cut holes in a 1000 pound bomb (albeit one filled with concrete). We all made mistakes in the development process – but these were wiped out by the chunk of 15mm steel that we removed so delicately from the side of the bomb. This paper is about how human fallibility can be accepted as part of the design process, and how it can be overcome with the help of Forth.

## The Timescale

Sometime in July 2002, Stephen worked out a timeline with the other companies involved. This required MPE to have at least a model but functioning set of mechanics by September 1st 2002, but then MPE was asked to do a GUI interface to the system on a PC, as well as supply and program the embedded ARM systems for the motion control and power management systems.

In early September 2002, Stephen asked Howerd "would you be interested in two weeks work programming a user interface and communications protocol on a PC in MPE's VFX Forth?" Howerd had already ported most of his MSDOS PPP program to VFX Forth in preparation for adding it to MPE's PowerNet TCP/IP stack, so we could use the HDLC transport layer from that, and the user interface sounded simple. So Howerd said "yes", and thus began one of the most challenging projects of our careers.

The demonstration was scheduled for November 11th 2002, and was to perform a series of cuts on various pieces of metal, including a real, but defused, 1000 pound bomb. The tests were to be performed at the Ministry of Defence's Shoeburyness firing range. Fortunately for us the firing range was closed down for a week while an unexploded shell was cleared, so our demo was moved back a week to November 18th. So, about eight weeks to get the PC program to talk to two ARM based systems, one controlling the cutting head air valves, the other the diesel engine, air and water pumps….

## The system

### *Mechanics*

The mechanics are driven by air motors controlling a water jet. These are linked by air, water, and optical cables to the service module.

### *Service Module (SM)*

An ARM system containing an MPE ARM Development Kit mounted talking to motor control boards based around Xilinx FPGAs. The SM also talks to the Power and Command modules below over RS485 links. Programmed using MPE's VFX Forth 6 cross compiler.

### *Power Module (PM)*

Another MPE ARM system that looked after the diesel generator and air and water pumps. Programmed using MPE's VFX Forth 6 cross compiler.

### *Command module (CM)*

An industrial PC with a touch screen and an RS485 interface. Programmed using MPE's VFX Forth for Windows.

## The Problems

When we arrived on site, nothing had been tested, nothing worked except the basic PCs and the ARM systems.

The original design that Howerd had been given used a conventional keyboard and mouse – he arrived on-site to find that there was now a touch screen. This meant that he had to re-write the screen layout, to add an on-screen numeric keypad. This was just the start…

The Sharp ARM processor had an undocumented feature that required the data cache to be disabled. This cost us maybe five days, trying to debug interrupt code which was perfectly correct but could never have worked.

The custom made FPGA boards to interface to the pneumatic control valves and position sensors had the classic two-byte read of a 16 bit counter which got a wrong reading about once every 30 minutes (but of course only when the motors were actually running). At this point Howerd stopped programming the GUI, and started as the hardware consultant with a crash course in VHDL for FPGAs. We added a begin…while…repeat loop to guarantee two consecutive readings the same and made a mental note to test it, but never did. Having spent many days debugging ground loops, and other noise related crashes, Howerd finally found that he had got the sense of the test wrong, so that the code waited for the counter to change, then hung until the watchdog kicked in if one of the readings was wrong. Oops.

At this point, Stephen could start actually testing the interface to the air motors that made everything move. Over a period of two weeks, we wrote three completely different sets of motor drivers. The real problem was that the cutting water leaves the jet at supersonic speed and contains an abrasive compound. The result is that the whole environment is covered in wet abrasive powder, including the leadscrews for the motion control system. Consequently smooth motion is difficult to achieve. The situation is made worse because most bombs are several feet underground, and the cutting equipment normally operates in a ditch cut by hand around and below one end or side of the bomb. Rubber boots and thick jackets were normal wear while testing. Fortunately, the one piece of equipment that never failed was the MPE coffee machine.

The RS485 serial lines used the mains supply earth as their ground reference. Since everything was opto-isolated, serial communications was only possible because one of the ARM systems had a short between the 0V of its 24V power supply (which happened to be connected to mains earth through the chassis) and the +5V of its 5V power supply. Everything worked fine when both ends of the cable were plugged into the same mains socket, but failed when installed 500m apart. If you unplugged the power cable between the PC and ARM units *before* unplugging the data cable, the opto-isolators were reverse biased and blew. This only cost one day, but boy, what a day! There was no schematic, just a bad photocopy of a pencil-drawn sketch of the sort of thing the circuit should do. The actual wiring had very little to do with the sketch, which anyway had some fatal design flaws. We changed the opto-couplers (thankfully in sockets) and soldered in some

1N4001's to stop them blowing again if the cables were changed in the wrong order.

On the night before the system was due to be delivered to the MOD firing range we noticed that the velocity control algorithm would not function at low speeds. It was fine on 10mm plate, but trying to cut 20mm required about a quarter of the speed, and the movement became jerky. As a temporary fix, we defined two words, Z and ZZ to make the cutting head move left and right so that you could cut in both directions.

The user interface lacked a "reverse" button. It had "forward" to cut, and "rewind" to return to the beginning, but it soon became clear that you needed to stop and go back over a section of the cut if, for example, the abrasive ran out. We could program in Forth to do these things, but this was not something you could expect someone to do whilst actually cutting into a live bomb!

The Service and Power Modules' control systems allowed you to turn the abrasive and water on or off, but it was important not to turn the water off until all of the abrasive has been flushed out of the jet, otherwise it would "set" and require dismantling to clean it. The original user interface tried to protect the equipment from the user by holding the water on for 30 seconds before the water pump could be turned off. All it did, however was force people to use the emergency stop button to turn the unit off (rather than wait 30 seconds – you may not have 30 seconds…). This stalled the diesel engine and tended to snap the drive belts to the air and water pumps.

It was also possible to set up a combination of states that forced abrasive into the water valve, and forced water back into the bottle of abrasive destroying them both. Remember that this is 500 Bar (7500 PSI) pressure with water jet speeds approaching the speed of sound. Apparently walking into the water jet causes immediate amputation – not an environment where you want to wait for a timeout. After some changes to the safety task, John repaired the abrasive feed system, carefully calibrating the high-pressure flow valves to maximise cutting speed and minimise the amount of abrasive used.

The position of each axis was determined by two pairs of fibre-optic cables detecting a quadrature signal from a vane on each motor. The 16 optical fibres were gathered together in a single cable, with 16 way optical connector so that the cutting head could be disconnected from its control box. Unfortunately, the manufacturers of this cable had "used the wrong glue" to fix the fibres into the connector, which made the fibres go slightly opaque. Also there was not enough slack on the x and y axis fibres on the cutting head so that when the head was moved to its extreme position the fibres were bent beyond their recommended curvature and snapped. Carl took on the job of getting the fibre optics to work, and work they did when glued, sliced and spliced correctly.

The last bug was wonderfully subtle – and another classic. The pneumatic control valves were powered by a 4 channel driver chip, each channel could handle 500mA, so a 350mA valve was fine – until three or four were on at once. The direction valves were only on when going down, left, away from the bomb or anti-clockwise. When cutting a rectangle starting from the top left and going right it was fine, going down was fine, but if the cutting head needed to move away from the bomb as it moved left, at least three direction valves were on, the driver chip overheated and turned all its outputs off. The feedback loop panicked and went at full speed, the cutting head moved to the right instead of left and charged into the bomb. A small heatsink fixed to each driver chip was all it needed. Note that this failure only occurred after about 45 minutes of cutting, and only when the weather was warm, or the previous cut had ended in an anti-clockwise direction.

## The Solutions

The problems listed above are not that unusual, but what made the ACE project different was the fact that they all came together, mob handed. The deadline was (almost) immovable, as it involved two people from each of the Army, Navy, RAF, MOD and Health & Safety Executive, and the hiring of a firing range. What also made the ACE project different was working together with people who were prepared to do what it took to get it to work. And of course Forth :

The three Forths mentioned in the title were MPE's VFX Forth for Windows, VFX Forth for the ARM and Forth, Inc.'s MSDOS polyForth. The latter ran Howerd's PPP.com program configured to display the ACE protocol packets.

VFX Forth for Windows provides a tight coupling to the MSDN documentation. Use the MSDN CD or go to the Microsoft website – find the Windows function you want, copy and paste the prototype into your Forth source, a couple of tweaks and you're done. You can also import resource scripts in the same way. The PC side took maybe four weeks in total – but it did have to be re-written twice.

The VFX ARM Forth compiles to optimised native code. This means that you don't have to write any assembler to achieve speed. It also means that the comms protocol file is *exactly* the same for Windows and ARM – no #ifs. On the application side, we didn't write a line of assembler, even for the motion control timer interrupts.

Both VFX Forths have the DocGen utility so that you document your code as you go (a good habit to get into), and you can then create a glossy glossary and user manual in a couple of minutes. Source and documentation are always in sync.

The service module and mechanics arrived about mid-way through the development process, then promptly disappeared to have cases re-sprayed. When it was around Stephen had priority, as he had to get the motor control side working. So almost all of the protocol development was done using three PCs. Because Forth is a high-level language the transfer to the real hardware was trivial. Provided **key** and **emit** worked, the protocol worked.

The comms protocol was specified as having to be readable by a dumb terminal. To do this Howerd changed the HDLC code gleaned from PPP to “escape” everything except one CR-LF pair. This made each packet appear on a new line, with the message in ASCII text, then some garbage checksum characters etc. We reserved two of the HDLC channels for text to be interpreted by a remote Forth interpreter, and text to be sent in reply. If you only use one channel the two ends chatter endlessly, the one replying to the other’s “?”.

What this created was extremely useful: from any of the three modules you can connect either a dumb terminal or a slightly intelligent one that stripped/added the HDLC wrapper, and then you can run the Forth interpreter on any of the remote machines.

The ARM systems have three serial ports, and any packet arriving on one of the two RS485 ports was echoed on the other. This meant that you could connect the three modules together using any of their serial ports and they could all talk to each other. Using this technique we were able to program the “Z” and “ZZ” words mentioned above (from the relative comfort of the “bomb shelter”) on the Service Module 500m away in an altogether less friendly place – on a windswept firing range in November, 25 m from a 500 Bar water jet and an unexploded bomb. We also used the technique to try out various times between turning off the water valve and reducing the diesel engine speed to idle. This meant that Gerry did not have to walk the 500m through sleet and snow to restart the diesel engine when it stalled.

And as for the hardware – running tests from an interpreter as fast as you can type them, compilation, downloading and Flash programming through the Ethernet port in seconds – nothing got in the way.

After the first demo, Howerd spent another three weeks, re-designing the user interface, and trying out a PID loop for the velocity control (which didn’t work) then a bang-bang loop (which did). We used a “one line digital oscilloscope” program to display the position of the cutting head :

```
: scope    begin  cr  @pos 64 mod  spaces .″ *″  key? until ;
```

This repeatedly displays a ”*”in a column corresponding to the value returned by @pos . You can see immediately the effect of the feedback loop. This showed that the simple bang-bang control loop was much better than the analog control in the original design. This allowed about £1500 worth of variable flow pneumatic values to be removed from the system, halving the total power consumption.

Two months later, another demo had been arranged to complete the set of tests from the first one. This time it was at Richmond Engineering’s site – this meant that the bomb we used contained 1000lb of concrete instead of TNT. The ACE equipment cut all sizes and shapes of hole, including chopping straight through the nose-cone – 42mm of steel followed by about 80mm of concrete. The user interface was friendly – you could stop, start, reverse and control the water and abrasive freely, with an indicator showing you where the abrasive was in the system. The comms protocol and all hardware was rock solid. In short ACE was working!

## Russian Front Programming – Howerd’s view

Nobody plans for a five month project to be completed in four, but sometimes the unexpected happens, the pressure builds, the hours get longer and the caffeine stronger.

ACE would not have happened without Forth, for three reasons :

1. The software on the ACE project was a relatively minor part of the work – in any other programming environment I reckon that the software alone would have taken a year. Perhaps I am exaggerating - maybe a team of four could have it completed it in 6 months.

2. A large part of the work was debugging hardware. OK, it would have been nice if that phase had been completed before we started, but given the urgency of the project there had been no time. Forth allows interactive testing of hardware. This was crucial to getting the system going at all.

3. No one had ever built ACE before, therefore no one could have written a complete specification for it without actually cutting holes. For example, when the water jet comes within about 0.5mm of a previously cut hole, such as the start point or corner, the jet tends to take the easy route and bends around the remaining web of metal rather than cutting it. To make sure that this tiny piece of metal gets removed, the jet is made to stop and “wobble” by +/-1.0mm for 5 seconds, then gradually speed up again. It took maybe one hour to add this functionality to the code and test interactively the best amount of wobble, pause time and rate of acceleration. Failure to get this detail right means that someone has to leave the (relative) comfort of their bunker, walk 500m to the bomb and attack it with a crowbar. Any volunteers? When we started the project we didn’t know that this was necessary, but when we found out that it was, it was easy to add, calibrate and test it, and explanatory comments in the code appear in the manual automatically. This is the right direction of information flow: experience -> code -> documentation -> specification, repeated as required.

## Russian Front Programming – Stephen’s view

When this job started, I knew it was going to be tight.

Most of what caused 100 hour weeks and delays with this project was lack of mechanical module testing because of the tight timescale and the constantly changing requirements to overcome unforeseen problems. When the joint services team came to review the project, the BDOs (bomb disposal officers) made suggestions and told us how they actually wanted to use the equipment. This information generated significant changes. The conversation with the naval people about doing this underwater was memorable.

Once you have cut a hole in a bomb, you have to remove the explosive. This is done with the same water jet used to cut the steel case, but with no abrasive. The jet is moved around the hole to “wash” away the explosive. Originally the specification was for circular holes at the nose of the bomb or rectangular holes along the axis. In reality, the BDO defines a set of points – a polyline. A quick search revealed a set of graphics algorithms for line crossing, and these were translated from Java and C (floating point) into Forth (integer) and tested in a few hours. The specification for the motion during washing suddenly changed to a random walk and so on. The net is a lifesaver for a tired and desperate programmer.

What went right was the almost uncanny ability in an interactive environment to change major system software structure very quickly. At 2am one morning, we decided that the current air/water interlock just wasn’t going to work, so the “safety” task was added to look after this, and the cutting task and washout tasks were cut down. This was another example of how factoring Forth words saves the job because code reuse is so easy.

Having code and documentation in the same file is wonderful. Looking at some of the code a few months later, I realise that having a tool such as DocGen, which automatically generates the documentation encourages the habit of documenting code, which significantly reduces interruptions from your co-workers.

## Summary

When writing in Forth you test each word interactively, getting to the bedrock of the requirements, re-writing code and creating a language in which to describe the system. Anyone who has programmed in Forth knows that this is the way to write programs that work, quickly.

The ACE project demonstrates this in a way that deserves attention from anyone confined to “mainstream” programming practices.

Bomb disposal people are special and have our admiration. They are also completely mad.