The development of a microprocessor-based system on a limited budget

CHRISTOPHER R. BROWN
University of Sheffield, Sheffield S10 2TN, England

This paper describes the design and implementation of a microprocessor-based front-end processor built to increase the throughput of a NOVA minicomputer used for stimulus preparation tasks. The problems encountered during the various stages of the project are discussed, and the various hardware and software development aids are described. The paper concludes with an assessment of the relative merits of in-house development of microprocessor-based systems and the purchase of ready-made equipment.

Microcomputers may be bought as complete "switch on and go" systems, such as the Commodore PET, or may be configured by plugging the desired combination of processor, memory, and interfaces into a prewired motherboard (e.g., Cromemco Z2). Both approaches offer cheap computing power without necessarily requiring any electronics expertise. The availability of microprocessor chips, however, raises the possibility of building "raw" computing power into almost any electronic equipment, by treating microprocessors as circuit elements and designing at the chip level. This approach, unlike the first two, does require electronics expertise, and yet it demands skills and development facilities in excess of those required for conventional hard-wired logic design.

This paper attempts to illustrate this point and to demonstrate the practicality of this approach with limited workshop facilities, by describing the development of a fairly large microprocessor-based system within our department.

THE APPLICATION

A Data General NOVA minicomputer is used extensively within the Department of Psychology at Sheffield University for the production of pictures used as stimuli in experimental work on stereopsis and feature recognition in human vision (Frisby & Mayhew, 1978; Mayhew & Frisby, 1978). Usually, each stimulus consists of a pair of 50-mm-sq pictures, set side by side, suitable for stereoscopic viewing. Each square comprises an array of 128 by 128 picture elements (pixels), each of which is assigned a "gray level" in the range 0 to 63. Many of the pictures are generated numerically within the NOVA, but some are derived by processing actual pictorial scenes. For these we use a simple picture-scanning

device, using a modified X-Y plotter interfaced to the NOVA, to bring the original image (from a photograph) into the computer (Brown, 1977). This device is rather slow, requiring about 5 min to digitize a single picture.

In order to produce "hard-copy" output of the pictures in a form suitable for use in a binocular tachistoscope, we use a Muirhead facsimile reproducer (Type K550-B/1). This machine produces high-definition (40 lines/cm) halftone pictures on a chemically treated paper using an electrolytic reaction, which avoids the need for a dark room or subsequent chemical development. The picture is built up using a slow (1 line/sec) mechanical raster scan. The machine is designed for the reception of pictures transmitted as an amplitude-modulated carrier over a telephone line, but originally a simple interface was constructed to allow it to be driven directly by the NOVA. This machine is also slow; about 5 min of computer time is required to output a stereo pair of pictures.

As the use of these facilities grew, the amount of NOVA time required for picture output (and, to a lesser extent, picture scanning) became an increasing problem, and we looked for a solution. We decided to construct a new digital interface between the NOVA and the picture reproducer, including enough random-access memory (RAM) to store a complete pair of pictures and enough "intelligence" to control the reproducer and output picture data to it entirely independently of the NOVA. The NOVA would then only be required to load the picture data into the interface's memory, and then be free for other tasks while the reproducer was running. This proposal looked even more attractive when we decided that the easiest way of putting the required "intelligence" into the interface was to use a microprocessor. This not only simplified the hardware design (as compared with hard-wired logic), it also transformed what would have been a very special-purpose interface into a much more general-purpose front-end processor (FEP). The NOVA is able to load not only picture data into the FEP memory but also the controlling software

This project was supported in part by SRC Grant 50894. I also wish to thank Mr. A. C. Daly, who contributed to the design, and Mr. R. Batty, who did most of the construction.

that programs it for a specific application. The work load on the NOVA was further reduced by connecting the picture scanner via the FEP. Later, graphics facilities were added to display stereogram pictures on a CRT. The final configuration is shown in Figure 1.

THE HARDWARE

Figure 2 shows the architecture of the FEP in more detail. It is based on a Motorola M6800 microprocessor (MPU). Data are transferred to and from the NOVA via 26 signal lines (12 data lines plus one strobe line in each direction). These connect to a standard parallel digital interface (Data General Type 4066) at the NOVA end and to a pair of M6820 peripheral interface adaptor (PIA) chips at the FEP end. In each direction, 8 of the 12 lines carry actual data, and the other 4 carry command and status information. Data is transferred a byte (8 bits) at a time; each transfer is initiated by a strobe pulse from the NOVA, which generates a nonmaskable interrupt in the MPU. The FEP itself cannot initiate a data transfer, but it can generate a program interrupt in the NOVA, via its own output strobe line. The interfaces to the scanner and reproducer consist mainly of analogue-to-digital and digital-to-analogue converters, connected to the MPU via additional PIAs. A Motorola MC6840 counter/timer chip is also used (with appropriate software) to control the timing of the programs that operate the scanner and reproducer.

The main memory of the FEP consists of 34K bytes of RAM, of which 32K bytes are used to store picture data (two 128 by 128 arrays) and 2K bytes are available for the control program. This memory uses 1K-bit

Figure 1. Block diagram of the environment in which the FEP operates.

Figure 2. Internal architecture of the FEP.

static RAM chips (Texas TMS4033) and accounts both physically and financially for over half of the entire FEP. A small (300 bytes) monitor program, permanently resident in an erasable, programmable read-only memory (EPROM), handles communication with the NOVA. This provides for transfer of data to and from memory and starts the control program at a specified address; it also provides some simple diagnostic facilities.

An unusual feature of the design is the way in which the main memory is shared between the MPU and the graphics controller. Two memory switch modules each provide tristate buffering of the data, address, and control signals to the memory. In normal operation, the graphics memory switch is "off" and the MPU memory switch is "on," allowing the memory to be loaded from the NOVA. During a graphics refresh cycle, the graphics memory switch is "on" and the MPU memory switch is "off," thus disconnecting the memory from the MPU and allowing it to be read directly by the graphics controller. Some such form of direct memory access is essential, because the data transfer rate required to ensure a flicker-free refreshed display (1 byte every 2 microsec) is far too high to be handled through a PIA.

IMPLEMENTATION

The entire system was developed from scratch and assembled from individual integrated circuits and components. The only items bought ready-made were the power supplies. The time from conception to completion of the project was 8 months; the total effort required was about 1 man-year. This may be broken down into four stages: design (40%), construction (40%), testing

(5%), and program development (15%). The lessons learned during this process are described along with the hardware and software development aids required at each stage.

Design Phase

Designing with microprocessors draws the concepts of hardware and software so close together that it is highly desirable for both aspects of the design to be handled by the same person. This person need not necessarily be skilled in writing large, well-structured programs, but he needs to understand the capabilities of the software. At an early stage of design, he must decide which functions are best handled by special-purpose hardware and which can best be done in software (hardware/software trade-off). During the testing phase it is also very helpful if the designer can write his own diagnostic programs as and when he needs them. (Such programs are often only a dozen or so instructions long.) However, the prime requirement is the ability to design digital electronics; consequently, most designers tend to be people with a hardware background who have acquired the necessary software skills.

Learning to program in assembly language can be a daunting prospect for an electronics technician but in practice has proved successful in our department, particularly if motivated in the context of a specific, real application. Most of our knowledge was gained from the appropriate programming and applications manuals (Motorola, Note 1, Note 2); the applications manual is particularly good. The relatively straightforward architecture and instruction set of the 6800 also helped, although the various addressing modes proved confusing to some people and needed additional explanation. Previous programming experience, even in a high-level language such as BASIC, was found helpful in establishing concepts such as loops, conditional branching, flow charts, and arrays. The final major learning aid was a small development system (American Microsystems Inc., Type EVK300). This gave users the opportunity of keying in and running their own programs. One of the cheaper evaluation kits, such as the Motorola MEK6800D2 (for the 6800 MPU) or the MOS Technology KIM-1 (for the 6502 MPU), would probably be adequate for this purpose also. They have the advantage of not requiring a Teletype or visual display terminal, but they are too limited for serious program development or hardware debugging work.

In design, we found it best to start at the "edges" of the system and work inward (i.e., consider the input/output requirements first). How many digital inputs and outputs are needed? What analogue signals are to be handled? What data transfer rates are expected? Will they require direct memory access techniques? Is a hardware timer needed? How will interrupts be used? Also, it is best to keep the hardware design as general as possible, by handling all timing, sequencing, and control functions in software wherever possible. This not only simplifies

the hardware, it also achieves maximum flexibility through software changes. Microprocessor computing power is so inexpensive it can quite legitimately be squandered in performing very mundane control functions, but it takes time to get used to this idea.

Construction Phase

Construction of the hardware is no different from the construction of any other digital system. Indeed, the use of a microprocessor reduces the number of components, and the bus architecture considerably simplifies the back-panel wiring. No particular difficulties were encountered during construction—there was just a lot of it. (The system contains 420 integrated circuits and 1,700 wirewrap connections, roughly 80% of them for the 34K-byte memory.)

Three printed circuit boards were designed for use in the system: an MPU board, a memory switch board (of which two are used), and a 2K-byte RAM board (of which 17 are used). All three were designed as general-purpose boards with a view to being useful in future projects. The MPU board in particular was a valuable design project. It carries the MPU, clock and reset circuits, two PIAs, 1K bytes of EPROM (using two AMI S6834 chips), and 256 bytes of RAM (using two AM 9111BPC chips), and costs about \$110. The interfaces to the scanner and reproducer, and the graphics controller, were built on three wirewrap circuit boards using an inverted type of wirewrap socket (Robinson Nugent AS series, for example, AS-163-U3-T) in which the wirewrap pins protrude on the same side of the board as the components. This enables us to maintain a board-to-board spacing of .6 in.

Testing Phase

It was not until we began to test the completed hardware, and tried to run programs, that the radical differences between a microprocessor-based system and a conventional hard-wired digital system really became apparent. Hard-wired logic can usually be tested a section at a time, for example, by injecting a signal at one point in a circuit and observing the result a little further downstream on an oscilloscope. In contrast, microprocessor systems have an uncomfortable all-or-nothing characteristic to their operation. The MPU, EPROM, RAM, and PIAs all have to be working before a program will run. The program itself must be free of major errors. Thus, any prototype system, when first tested, is very likely to contain faults that will prevent it from working altogether. Furthermore, it is, as it stands, extremely difficult to debug. The only signals that can be monitored conveniently with an oscilloscope, and for which it is reasonably easy to predict what to expect, are the input/output lines, but these give little indication of what is happening inside the system. The internal operation depends on the combined effect of a large number of signals, that is, the address, data, and control busses, and piecing together what is happening

by observing one or two of these signals at a time is no easy matter. The use of a logic analyzer helps considerably, but logic analyzers are expensive. Even so, unless the system happens to be a fully fledged micro-computer complete with a Teletype interface and keyboard monitor program, there is still no effective way of communicating with it. The user is unable to examine or modify memory locations or to test his program and hardware in a systematic manner. The system may represent an excellent solution to the problem for which it was designed, but (even if the hardware is working) it provides a very poor environment for program development.

We avoided these problems by making use of the facilities provided by the EVK300 development system and connecting it directly to the FEP using a technique called *in-circuit emulation*. This is illustrated in Figure 3. The principle is simple. The MPU chip is removed from the system under test (usually referred to as the target system in this context), and all the signals from the MPU in the development system are connected across to the vacant MPU socket in the target system via an electronic "umbilical cord." The development system's MPU then emulates the operation of the MPU in the target system. We were then able to use the Teletype and keyboard monitor program of the development system to test the target system's RAM, to check the contents of the EPROMS, and to communicate directly with the PIAs. We could also write simple diagnostic routines that ran in the development system's RAM and performed more thorough, real-time tests of the target system's hardware. Finally, we were able to debug the monitor program for the FEP (i.e., the program that handles communication between the FEP and the NOVA) by running it in the development system's RAM before committing it to an EPROM. (At this stage the FEP was connected to both the NOVA and the development system). This was particularly valuable, because programs in EPROM cannot be patched or have break-points set in them, and it takes time to generate a new version by erasing and reprogramming the EPROM.

There are more sophisticated ways of performing in-circuit emulation; for example, it is possible to

emulate the target system's RAM as well as the MPU. Purpose-built emulators are commercially available (e.g., Tektronix 8001), but they are expensive. Our crude techniques required only a small development system, but they were very effective. There are, however, some important points to be watched:

(1) Those devices in the target system requiring clock signals (e.g., the PIAs) must run synchronously with the development system. This was achieved by synchronizing the target system's clock generator to the development system's clock.

(2) It is essential to avoid address assignment conflicts between the development and target systems. That is, address space occupied by devices in the target system must not overlap with address space used by the development system. In our case, this required temporary changes to the address decoding in the FEP, and meant that we were not able to test all of the RAM simultaneously. In view of the common practice, particularly in small systems, of not decoding all the address lines, this point is worth bearing in mind at the design stage.

(3) It is important to guard against overloading the signals from the development system's MPU, and we recommend that the signals be buffered. (Bus buffering is not usually provided on the cheaper evaluation kits.) The "umbilical cord" should be kept short and adequately screened. We used ribbon cables 50 cm long, with alternate cores grounded.

(4) A small evaluation kit is inadequate for in-circuit emulation, and we recommend the purchase of a slightly more expensive development system. The system should include a buffered bus, a Teletype/VDU interface with a keyboard monitor program, a few hundred bytes of RAM, a convenient way of loading programs (we load off paper tape through a Teletype reader), and facilities for programming EPROMS. The EVK300 meets all these requirements and currently costs \$720.

One final, very simple item of test equipment we built was a "byte tester." This merely contains a row of eight LEDs with drivers and latches to monitor an output byte, plus eight switches with pull-up resistors to set up an input byte. This was useful in testing input and output lines from PIAs.

We tested the FEP very systematically, building it up a section at a time and testing each section thoroughly before adding the next. This gives the best chance of picking up faults at an early stage which otherwise might later start to interact with the rest of the system and become very obscure. Testing of the entire system took about 1 week.

Program Development Facilities

The facilities we use for preparation of 6800 assembly-language programs are shown in Figure 4.

Source programs are initially prepared on the NOVA, where a powerful disk operating system and editor are available. They are then assembled on the NOVA using a cross-assembler that produces a printed listing file and an

Figure 3. Set-up for in-circuit emulation.

Figure 4. Program development facilities.

object file. The cross-assembler (Motorola M68SAM) is available from Motorola for about \$720, but it is also fairly widely available within educational establishments. Documentation is provided in the *M6800 Microprocessor Programming Manual* (Motorola, Note 1). It is written in FORTRAN and we had little difficulty in transferring it onto the NOVA. There must, of course, be some way of transferring the resultant object code from the host computer to the system on which the program is to run. In our case, object files destined to run on the FEP are directly down-loaded into it from the NOVA. Object files destined to run on the EVK300, or to be programmed into an EPROM, are punched onto paper tape and loaded into the EVK300 through the Teletype tape reader. We were fortunate in having a spare ASR33 Teletype available for use with the EVK300. The only additional hardware we needed to buy was the ultra-violet EPROM eraser (\$160 from Anderman and Company Ltd.).

The M6800 instruction set is comparatively easy to hand-code, and we occasionally resorted to this for short test programs to avoid using the NOVA. Hand-coding is also adequate (even desirable) while learning about the instruction set but presents a major obstacle to the development of programs longer than, say, 50 instructions. The use of a proper editor and assembler was therefore an essential development aid for the project. Facilities for running a cross-assembler represent an inexpensive solution. Users lacking such facilities would need to purchase a more sophisticated development system with cassette or floppy disk storage, supporting its own editor and assembler (e.g., Motorola Exorciser).

To aid the development of applications programs for the FEP, a debugging program called FEPBUG was written to run on the NOVA. FEPBUG allows users to communicate with the FEP via the NOVA's Teletype, to load an object file from disk into FEP memory, to examine and modify memory locations, to set break-points, and to start program execution at a specified address. In fact, the combination of NOVA and FEP constitutes a powerful microprocessor development

system in its own right. However, we try to resist the temptation to use it as such, because the purpose of the FEP is to reduce the workload on the NOVA, not increase it.

DOCUMENTATION

Good documentation is extremely important for the future maintenance and modification of any equipment, particularly something as complex as the FEP, and often takes a surprisingly large amount of time. We have found the following guidelines produce adequate documentation for minimum effort.

(1) Assume that the reader will have the same general level of knowledge as you have, but no knowledge of the specific project.

(2) Hardware documentation should include circuit diagrams, component layouts, and wiring lists, plus block diagrams and descriptive text where required to clarify the operation.

(3) Data sheets for any unusual devices used should be included.

(4) Source programs should be commented heavily and in a consistent style. Each subroutine and main section should have a descriptive header paragraph. If this is done, the program listing file (supplemented by an occasional flow diagram) will provide adequate software documentation.

(5) MPU-based projects require additional documentation covering the hardware/software interface. Documentation should include a list of address assignments for all memory and peripheral registers and tables showing the function of each data, control, and status bit in each peripheral register.

CHOICE OF APPROACH

Having completed the project, we inevitably asked ourselves whether our decision to design and build the system from scratch was really a sensible approach or whether it would have been better to configure a commercially available system for the purpose. An obvious basis for comparison is cost. The components for our own system cost \$2,400, and admittedly a comparable commercial system would not be much more expensive. This is quite astonishing when one considers that a commercial system comes already built and probably includes a fair amount of utility software. For a true cost comparison, we should take into account the cost of a man-year of development time. On this basis, it is clear that, if a user's needs are met by a standard, ready-made commercially available micro-computer, he would be well advised to buy one. However, we still consider the project a success. Our approach has several advantages: (1) The system is exactly tailored to our needs. In particular, it might have been difficult to obtain the required shared-memory architecture from a commercial system, and we would in any case have had

to build our own interfaces for the scanner, reproducer, and graphics CRT. (2) We have an intimate working knowledge of the entire system and, consequently, have no fears about repairing, modifying, or extending it. (3) We learned a lot about microprocessor design, which we would not have done by buying a ready-made system. (4) The three printed circuit boards we designed for the system will be useful in future applications. (Three new projects using the MPU card are already well advanced.)

Finally, we should stress that we consider the FEP to be an atypical design project, in that it is a large system and rather resembles a full-blown microcomputer. We expect that the majority of our future microprocessor projects will be small dedicated systems, purpose-built for specific applications and not appearing as "programmable" devices to the end user. For these, a cost comparison is no longer valid, because no commercial equivalent is usually available. The choice now is between a microprocessor-based solution and a conventional hard-wired logic approach. We believe that over a very wide range of psychological applications, the ability to design with MPUs as circuit elements will offer better performance, lower component cost, and (having gained the initial experience) a shorter development time.

CONCLUSION

The development of microprocessor-based systems is certainly within the scope of a small electronics laboratory and, with intelligent use of existing resources, need not require heavy capital investment. However, in-house construction of a complete general-purpose microcomputer is not an economically sensible alternative to the purchase of a ready-made system.

REFERENCE NOTES

1. Motorola, Inc. *M6800 microprocessor programming manual*, 1975.
2. Motorola, Inc. *M6800 microprocessor applications manual*, 1975.

REFERENCES

- BROWN, C. R. A simple on-line device for scanning photographs. *Perception*, 1977, 6, 479-481.
- FRISBY, J. P., & MAYHEW, J. E. W. Contrast sensitivity function for stereopsis. *Perception*, 1978, 7, 423-429.
- MAYHEW, J. E. W., & FRISBY, J. P. Contrast summation effects and stereopsis. *Perception*, 1978, 7, 537-550.

(Received for publication March 7, 1979;
revision accepted June 19, 1979.)