ijeee

27/2

volume twenty-seven/number two
April 1990

INTERNATIONAL JOURNAL of ELECTRICAL ENGINEERING EDUCATION

MANCHESTER UNIVERSITY PRESS for the
University of Manchester Institute of Science and Technology

Int. J. Elect. Enging Educ., Vol. 27, pp. 132–140. Manchester U.P., 1990. Printed in Great Britain

A MODULAR ARCHITECTURE FOR AN INTRODUCTORY STUDY OF 8-BIT MICROPROCESSORS

JOSÉ MANUEL MARTINS FERREIRA and RAUL FERNANDO DE ALMEIDA MOREIRA VIDAL
Department of Electrical and Computer Engineering, College of Engineering, University of Porto, Portugal

INTRODUCTION

There are several 8-bit microprocessor kits easily available to support an introductory course in this area. Although having been used in many 'hands-on' circumstances, ranging from short term knowledge updating seminars for electronic technicians, up to university-level courses in electronic engineering, these kits are based on architectures that reflect their distinct development environments. In fact, just a few years difference in these product development dates is enough to make them have little resemblance to each other, essentially due to the very fast rate of appearance of new components, as well as the continuous decrease in their price/performance ratio (just recall that many higher capacity EPROMs are now cheaper than their lower capacity counterparts, or that high performance liquid crystal displays—LCD's—are becoming common due to their low prices).

These architectural differences (either in hardware, or in software) in the available 8-bit microprocessor kits, as well as their distinct user interfaces, do not help towards achieving a practical feel for the different capabilities of the respective CPU's, which should be one of the aims of an introductory course. It thus usually happens that due to time restrictions, a particular CPU is chosen, the others being only slightly inferior.

This has for a consequence that each individual's preference for a particular microprocessor is most often the result of circumstances that have little or nothing to do with a conclusion drawn from an equal-basis comparison of the various components available.

On the other hand, and taking a second thought on this subject, it is easy to conclude that the most commonly used 8-bit microprocessors (essentially those from Zilog, Intel, and Motorola) are all based on the same internal architecture, differing only on the way it is implemented in each particular case. In fact, this generic architecture can easily be described on the basis of

This paper was first submitted in August, 1988.

five types of blocks, as fo

- a set of blocks to all
  and control);
- a block used to temp
  monly called the 'ins
- directly connected wi
  the instruction to exe
  sequence of elementa
  monly called the 'ins
- a block charged with
  corresponding to the
  (commonly called 'A
- finally, a set of regist
  generic (like registers
  defined tasks (like st
  status register).

The reader is invited to c
the various microprocess
in order to keep this intr

This similarity between
microprocessors leads th
an introductory course b
then present the various r
provided that a convenier
based on the same minim
These two aspects are rel
characteristics, and its im
associated with the famili
exist.

The purpose of this wo
architecture, in terms of i
with the aim of providing
course on 8-bit microproc

The applications field f
activities, from short term
expansibility features wou
situations.

Finally, a few comment
a short list of possible pro
presented.

DESCRIPTION OF A M

The following sections wil
istics of the proposed arch

. 1990. Printed in Great Britain

)DUCTORY STUDY

L FERNANDO

'ollege of Engineering,

ilable to support an
:n used in many 'hands-
:e updating seminars for
n electronic engineering,
distinct development
ese product development
to e h other, essentially
nents, as well as the
(just recall that many
lower capacity counter-
s—LCD's—are becom-

or in software) in the
stinct user interfaces, do
erent capabilities of the
an introductory course.
a particular CPU is

reference for a particular
ces that have little or
l-basis comparison of the

n this subject, it is easy to
ocessors (essentially
on the same internal
ted . each particular
scribed on the basis of

five types of blocks, as follows:

- a set of blocks to allow the interface with the external buses (address, data and control);
- a block used to temporarily store the instruction to be executed (commonly called the 'instruction register');
- directly connected with this last block, another one that decodes which is the instruction to execute, and that is responsible for controlling the sequence of elementary operations corresponding to its execution (commonly called the 'instruction decode and control');
- a block charged with the execution of arithmetic and logic operations corresponding to the set of instructions supported by the microprocessor (commonly called 'ALU');
- finally, a set of registers intended to be places of temporary storage, some generic (like registers, accumulators, etc.), and some others devoted to predefined tasks (like storing the program counter, the stack pointer, and the status register).

The reader is invited to consult the respective data sheets for a comparison of the various microprocessors' block diagrams, which are not reproduced here in order to keep this introduction short.

This similarity between the internal architectures of the most common 8-bit microprocessors leads thus to the conclusion that it would make sense to start an introductory course by describing this generic internal architecture, and then present the various microprocessors on an equal-importance basis, provided that a convenient set of kits were available. These kits should be all based on the same minimal architecture, and have the same user interface. These two aspects are related respectively to their hardware and software characteristics, and its importance comes from the fact that the time overhead associated with the familiarization with each particular kit would thus not exist.

The purpose of this work is precisely to describe a minimal modular architecture, in terms of its hardware and software components, developed with the aim of providing a common approach to be used on a first level course on 8-bit microprocessors.

The applications field for the proposed architecture covers a wide range of activities, from short term seminars up to more advanced studies, where its expansibility features would be used to develop larger systems for specific situations.

Finally, a few comments on some actual implementations will be made, and a short list of possible projects based on these implementations will be presented.

DESCRIPTION OF A MINIMAL MODULAR ARCHITECTURE

The following sections will now describe the hardware and software characteristics of the proposed architecture. Both descriptions will take place on a basis

intended to be detailed enough to allow an easy understanding of its implementation, but also sufficiently generic as to make it independent of any particular microprocessor.

Hardward aspects

The hardware architecture can be seen in Fig. 1, and has six main components: the central processing unit; the memory (ram and eprom); the parallel input/output; the input channel (a keyboard, which is the user-system link); the output channel (an LCD, which is the system-user link); and the decoder block.

The CPU is obviously the chosen microprocessor.

The memory block is composed of two sockets—one for static RAM (volatile, or not), the other for EPROM—and a few jumpers to allow for the accommodation of different capacity devices, within a given range. This block remains unaltered whatever the microprocessor chosen.

The parallel I/O block is implemented using the corresponding microprocessor peripheral, in each case providing 16 parallel I/O lines. The input channel is a 24 key matrix keyboard (6 columns, 4 rows) which allows the user to communicate with the system. Ten of the sixteen I/O lines are used to interface the keyboard (6 output lines to rotate a zero through the columns, and four input lines to read the rows). This block remains unaltered whatever the microprocessor chosen.

The decoder is a combinational logic block which takes its inputs from the address and control buses, and provides the select signals for the various components, and eventually some modified control signals, when required. It is advantageously implemented with a bipolar PROM, which allows the use of the same single component, whatever the microprocessor chosen. Some particular implementations may also require the use of a sequential logic complement to allow for timing compatabilities.

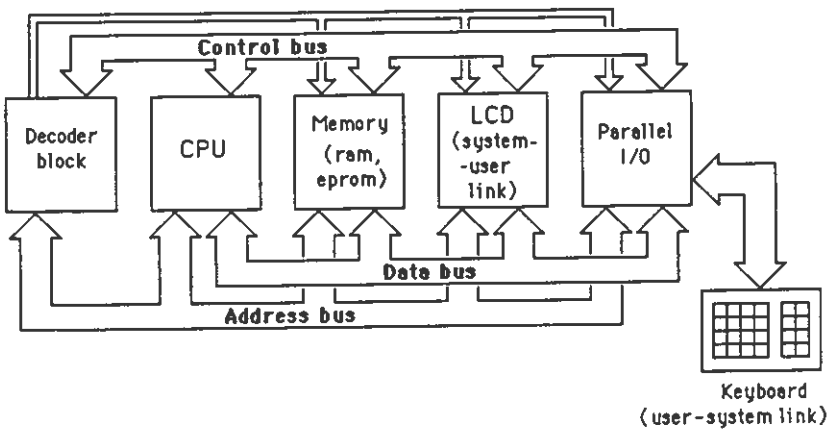


FIG. 1

It is at this point worthwhil
expansibility. In fact, more ad
of further possibilities, such as
this in mind, the decoder bloc
signals for the corresponding s
the real time clock component

A final word goes for the la
tations, preferably with a deta
could be available, when the f
the keyboard.

Software aspects

Coming now to the software a
system monitor requires essen

- a system initialization rou
condition is forced, and w
functions;
- an examine/substitute me
or alter the contents of an
previously indicated;
- a routine which allows th
EPROM, or previously in
instruction execution sequ
- finally, a warmstart routir
invoked by some type of b
rupt, etc.), and its executi
microprocessor registers,
grams.

The definition of these four
it enables the specification of t
between both parts (user, syst
(construction of messages fron

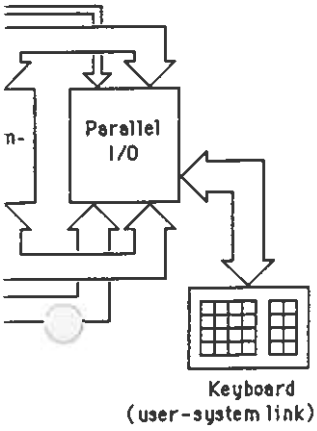
0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

understanding of its imple-
e it independent of any

and has six main compo-
am and eprom); the parallel
ich is the user-system link);
-user link); and the decoder

sor.
—one for static RAM
few jumpers to allow for the
hin a given range. This block
hosen.
ie corresponding micro-
arallel I/O lines. The input
4 rows) which allows the user
en I/O lines are used to
zero through the columns,
remains unaltered whatever

ich takes its inputs from the
t signals for the various
rol signals, when required. It
ROM, which allows the use of
rocessor chosen. Some partic-
a sequential logic comple-



It is at this point worthwhile to add a few comments concerning system expansibility. In fact, more advanced applications could require the existence of further possibilities, such as an RS232C channel, or a real time clock. With this in mind, the decoder block should thus already provide the selection signals for the corresponding serial communications peripheral, and also for the real time clock component.

A final word goes for the layout, which sould be the same for all implementations, preferably with a detachable keyboard, so that all 16 parallel I/O lines could be available, when the foreseen application does not require the use of the keyboard.

Software aspects

Coming now to the software architecture, it is easy to conclude that any system monitor requires essentially four main routines, as follows:

- a system initialization routine, running on power-up, or when the reset condition is forced, and which programs all components to their specific functions;
- an examine/substitute memory routine, which allows the user to examine or alter the contents of any memory position, according to an address previously indicated;
- a routine which allows the user to run a program, either resident in EPROM, or previously included in RAM. This routine transfers the instruction execution sequence to an address previously indicated;
- finally, a warmstart routine may prove very useful to the user. It should be invoked by some type of breakpoint instruction (restart, software interrupt, etc.), and its execution should send to the display the contents of the microprocessor registers, thus helping the user to debug his own programs.

The definition of these four main routines is in fact an important step, since it enables the specification of the sequence of messages to be exchanged between both parts (user, system), i.e., it defines which keys are necessary (construction of messages from the user to the system), and which messages

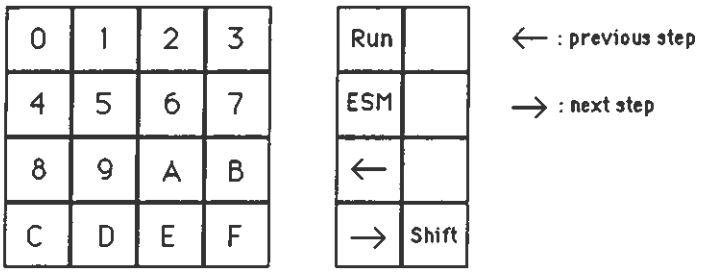


FIG. 2

should be displayed (construction of messages from the system to the user).

The following attributes are then allocated to the keyboard, as shown in Fig. 2. Note that:

- the 'next step' key (→) has a somewhat dual function, since it is used either to validate an address (when an address specification is required, both in the run routine, or in the examine/substitute memory routine), or to increment the actual address (in the examine/substitute memory routine, but after an address has been validated). It is thus designated neither as 'enter', nor as 'next position', but as a more generic 'next step', since it really allows the user to move ahead from the actual state.
- a few keys are left free. Also, the inclusion of a shift key (to be software implemented) will duplicate the number of attributes, in case any user application requires a larger number of possibilities.

Referring now to the messages to be sent from the system to the user (displayed messages), it is easy to conclude that two out of the four main routines will have a characteristic display that is independent of the microprocessor chosen. These are the cases of the run and examine/substitute memory routines, to which the messages shown in Fig. 3 were assigned. The system initialization routine could also have the same characteristic display, but it is perhaps more convenient that the log in message includes the reference of the microprocessor used in each particular implementation.

Finally, and since the warmstart routine is intended to show the contents of the microprocessor registers, it has a characteristic display that is obviously dependent upon each microprocessor.

The complete set of all necessary routines can then be enumerated as follows:

- a system initialization routine (**SYSINI**), already described;
- an examine/substitute memory routine (**ESM**), already described;
- a routine to allow program execution (**RUN**), already described;
- a warmstart routine (**WSTART**), also already described;
- a routine that keeps dealing with the keyboard (**RDKEYB**) until it detects a key closure, and then sends back the corresponding code;

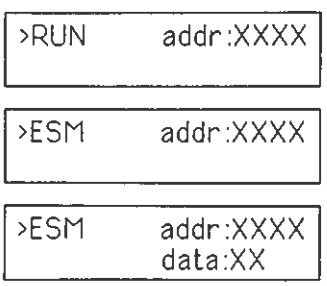


FIG. 3

- a routine to rotate a buffe
  digits, and which is invok
  for data or address specifi
- a routine to send the cont
  again, this can be either a
- a routine which sends dat
  commands) to the display
- finally, a routine that stor
  the four main routines.

These nine routines are sho
dependencies, as well as their
cates the routines called by ea
level specification of the set of
for the ESM case:

ESM clear display
display the ESM r
clear the address st

	SYSINI	ESM	RUN	WSTART
SYSINI		§	§	
ESM		§	§	
RUN		§		
WSTART		§	§	
RDKEYB				
ROTBUF				
DSPBUF				
LCDOUT				
MSGLIB				

·om the system to the user).
the keyboard, as shown in

l function, since it is used
ess specification is required,
ubstitute memory routine), or
ine/substitute memory rou-
l). It is thus designated neither
ore generic 'next step', since it
he actual state.
f a shift key (to be software
ttributes, in case any user
sibilities.

n the system to the user
two out of the four main
s independent of the micro-
and examine/substitute
in Fig. 3 were assigned. The
same characteristic display,
n message includes the
rticular implementation.
ended to show the contents of
tic display that is obviously

then be enumerated as

eady described;
M), already described;
), already described;
ly described;
ard (**RDKEYB**) until it detects
esponding code;

un routine)

xamine/substitute
emory routine, before
dress validation)

xamine/substitute
emory routine, after
dress validation)

- a routine to rotate a buffer in memory (**ROTBUF**), of either two or four digits, and which is invoked each time a new digit is entered, respectively for data or address specification;
- a routine to send the contents of a buffer to the display (**DSPBUF**). Once again, this can be either a two (data) or four (address) digits buffer;
- a routine which sends data codes (for characters) or control codes (for commands) to the display (**LCDOUT**);
- finally, a routine that stores all the messages (**MSGLIB**) corresponding to the four main routines.

These nine routines are shown in Table 1 which gives their functional dependencies, as well as their input/output parameters. Each '§' signal indicates the routines called by each horizontal entry. Each routine has a high level specification of the set of tasks to be executed, such as is illustrated below for the ESM case:

ESM clear display
display the ESM routine message
clear the address storage buffer

TABLE 1

	SYSINI	ESM	RUN	WSTART	RDKEYB	ROTBUF	DSPBUF	MSGLIB	LCDOUT	Input Parameters	Output Parameters
SYSINI		§	§		§			§	§	None	None
ESM		§	§		§	§	§	§	§	None	None
RUN		§			§	§	§	§	§	None	None
WSTART		§	§		§		§	§	§	Register contents (in the stack), SP	None
RDKEYB										None	Code of the pressed key (in the accumulator)
ROTBUF										2 byte buffer in memory	None
DSPBUF								§		2 byte buffer in memory	None
LCDOUT										Data, or Control code, in the acc.	None
MSGLIB								§		None	None

addr display actual contents of the address buffer
Loop1 read keyboard
if key code is for RUN then jump to RUN routine
if key code is for Next Step then jump to data
if key code is for an hexadecimal digit then (rotate the address buffer and insert this new digit and jump to addr)
else jump to Loop1
data display the actual address and its content
Loop2 read keyboard
if key code is for ESM then jump to ESM routine
if key code is for RUN then jump to RUN routine
if key code is for Next Step then (increment address buffer and jump to data)
if key code is for Previous Step then (decrement address buffer and jump to data)
if key code is for an hexadecimal digit then (rotate actual memory position and insert this new digit and jump to data)
else jump to Loop2

All nine routines are written as individual modules of source code, and then assembled and linked together, generating a single module of object code, which is then transferred to an EPROM. This modular approach has the advantage that it results in a very clear software architecture, absolutely independent of the microprocessor chosen. The linker output lists the physical addresses allocated to each routine, which are then also available to the user for his own applications. It is also worthwhile to mention that this approach greatly enhances software expansibility, since each new module (for example, to deal with a serial channel, or with a real time clock, both resultant from eventual hardware expansions) can just be linked together with the existing routines, thus ending in an homogeneously expanded system.

SOME COMMENTS ON ACTUAL IMPLEMENTATIONS

The first implementation of this architecture took place around an MC6802 CPU (Motorola), which was chosen due to its fairly transparent instruction set, thus considered an adequate choice for first-time practitioners in the area.

A Z80 implementation briefly followed as a result of students' work, showing that it is very easy to adapt the described architecture to any common 8-bit microprocessor.

SOME SUGGESTED PROJECTS

Since the decoding block already provides some select signals to ease system expansion, it is now of interest to list some suggested projects intended for more experienced users.

A first suggestion corresponds to the implementation of an RS232C serial link, in order to allow interconnection to a large range of external equipment.

As previously described,
processor peripheral is al
this task consists essentia
prove to be a very interes
will allow the user to dev
any PC-compatible (edit
to the kit for immediate

Another interesting pr
allow for time-dependent
mented included the dec
time clock (an upgrade o
internal architecture, als
including crystal, and th
to the Z80 is also fairly s

A varied range of othe
interfaces for analog sign
or not), tape interface fo

FINAL COMMENTS

Actual results obtained t
conclusions, as follows:

- the proposed archite
both from its hardwa
sufficiently high level
between various 8-bi
- it has provided a me
various microproces
cerns our original ai
comparative analysis

Anyone interested in t
well as the source code li
tions, may contact either

REFERENCES

[1] Hall, Douglas V., Micr
0-07-025571-7 (1980).
[2] Wiatrowski, Claude A.,
Systems, McGraw Hill,
[3] Ciarcia, Steve, Build Y
[4] Rafiquzzaman, Moham
Harper & Row, ISBN
[5] Hayes, John P., Digital
0-07-027367-7 (1985).
[6] Programmable Hardwa

›uffer

UN routine
o data
hen (rotate the address buffer
ddr)

:nt

;M routine
UN routine
nent address buffer and jump to

:crement address buffer and

hen (rotate actual memory
ımp to data)

›dules of source code, and then
gle module of object code,
noc r approach has the
: architecture, absolutely
linker output lists the physical
hen also available to the user
o mention that this approach
ıch new module (for example,
: clock, both resultant from
:d together with the existing
anded system.

ıENTATIONS
›k place around an MC6802
airly transparent instruction
-time practitioners in the area.
:sult of students' work,
ed architecture to any

: se signals to ease system
:ested projects intended for

:ntation of an RS232C serial
: range of external equipment.

As previously described, the chip select signal for the corresponding microprocessor peripheral is also available as an output of the decoding PROM, so this task consists essentially in developing the necessary software. This may prove to be a very interesting project, since the connection to a PC serial port will allow the user to develop his applications using readily available tools for any PC-compatible (editor, assember, linker) and then to download the code to the kit for immediate execution.

Another interesting project involves the inclusion of a real time clock to allow for time-dependent execution of actions. The actual versions implemented included the decode signals for the Dallas Semiconductor DS1287 real time clock (an upgrade of the MC6818), which besides a very attractive internal architecture, also includes a lithium battery, complete time base including crystal, and the possibility of Intel or Motorola timings (connection to the Z80 is also fairly simple).

A varied range of other projects could also be recommended, such as interfaces for analog signals acquisition, inclusion of relay outputs (solid state, or not), tape interface for acquired data storage, etc.

FINAL COMMENTS

Actual results obtained through usage of these kits highlight two main conclusions, as follows:

- the proposed architecture leads to a fairly straighforward implementation, both from its hardware or software viewpoints, while remaining a sufficiently high level specification, such as to allow easy conversion between various 8-bit microprocessors;
- it has provided a means to support students' enthusiasm in dealing with various microprocessors, thus bringing a rewarding result in what concerns our original aim of making it possible to present a practical comparative analysis of the most common of such components.

Anyone interested in the detailed hardware diagrams and PCB routings, as well as the source code listings and linker data, for the existing implementations, may contact either of the authors at the above address.

REFERENCES

[1] Hall, Douglas V., Microprocessor and Digital Systems, McGraw Hill, ISBN 0-07-025571-7 (1980).

[2] Wiatrowski, Claude A., and House, Charles H., Logic Circuits and Microcomputer Systems, McGraw Hill, ISBN 0-07-070090-7 (1980).

[3] Ciarcia, Steve, Build Your Own Z80 Computer, Byte Books, ISBN 0- 07-010962-1 (1981).

[4] Rafiquzzaman, Mohamed, Microprocessors and Microcomputer Development Systems, Harper & Row, ISBN 0-06-045312-5 (1984).

[5] Hayes, John P., Digital System Design and Microprocessors, McGraw Hill, ISBN 0-07-027367-7 (1985).

[6] Programmable Hardware, Byte, (January 1987).

ABSTRACTS–ENGLISH, FRENCH, GERMAN, SPANISH

A modular architecture for an introductory study of 8-bit microprocessors
The different architectures of various microprocessor kits makes it difficult to give equal treatment to the most representative of such components. However, considering the similar internal architecture of the most common 8-bit microprocessors, it would make sense to have them presented on an equal–importance basis, if a corresponding set of identical (hardware and software) kits were available. The aim of this work is to describe such a minimal architecture.

Une architecture modulaire pour l'introduction à l'étude des microprocesseurs de 8 bits
Les différentes architectures des kits introductoires disponibles sur le marché rendent difficile une approche balancé aux différents microprocesseurs. Pourtant, comme l'architecture interne des microprocesseurs à 8 bits plus répandus est pareille, il semblerait raisonable de les présenter au même niveau d'importance, si on disposait d'un ensemble de kits identiques (même architecture de matériel et de logiciel). Dans de travail, on décrit une architecture minimale qui poursuit ce but.

Eine Bausteinkonfiguration für ein einführendes Studium von 8-bit-Mikroprozessoren
Die unterschiedlichen Konfigurationen verschiedener Mikroprozessorausrüstungen erschweren gleichförmige Behandlung der typischsten solcher Bauteile. Jedoch in Anbetracht des ähnlichen Innenaufbaus der üblichsten 8-bit-Mikroprozessoren würde es sinnvoll sein, sie auf der Basis gleicher Wichtigkeit zu behandeln, falls ein entsprechender Satz identischer Hardware- und Softwareausrüstungen vorhanden wäre. Das Ziel dieser Arbeit ist, eine solche minimale Konfiguration zu beschreiben.

Una arquitectura modular para un estudio introductorio a los microprocesadores de 8 bits
Las diferentes arquitecturas de los distintos conjuntos básicos de microprocesadores hace dificil dar el mismo tratamiento a los más representativos de estos. Sin embargo, considerando la similar arquitectura interna de los microprocesadores más usuales de 8 bits, podria tener sentido que fueran presentados en bases equivalentes, si estuviera disponible el correspondiente conjunto de equipos idénticos (hardware y software). El fin de este trabajo es describirlo con la minima arquitectura.

BOOK REVIEW

Computing for Engineers—a problem solving approach to programming in Pascal: C. GRANT (Prentice Hall, 1989, 364 pp., £15.95)

The high level language Pascal has been widely adopted in universities as a suitable language to teach undergraduate engineers the rudiments of structured programming. *Computing for Engineers* adopts this approach to provide an introduction to efficient programme design and development for problem solving.

As is the nature of the subject, the book itself is well structured and easily read. Part I is devoted to programming fundamentals, hardware, software and data representation mechanisms. Part II describes the practical application of structured programming, modular programming and top down programming to engineering case studies. Pascal data structures are treated in the final third part of the book. The first appendix provides a reference to standard Pascal whereas the second enunciates the mathematics of linear difference equations. Two further appendices are dedicated to programme and examples indices. Exercises are set at the end of each chapter.

The strength of the text lies in the use of case studies to demonstrate the practical engineering applications of Pascal. Consequently, the book is suitable as an undergraduate teaching book and as a reference source for standard solutions to engineering problems. The text does suffer slightly in its lack of consideration for other high level languages and the use of numerical library routines in problem solving.

ANDREW A. P. GIBSON, *Department of Electrical Engineering and Electronics, UMIST*

AN UNDERGRADUATE S
DESIGN WITH INDUSTRI

D. R. CAMPBELL
Department of Electrical and E
nology, Scotland

1 INTRODUCTION

The Honours and Ordinary d
Electronic Engineering at Pai
sandwich' structure. The indu
able for the potential professi
students and employers. On e
choose a set of options, each
both final year courses are als
assignment.

The two-term, final year of
constraint on the volume and
assimilated by the student. A
to provide some practical exp
tory programme is allocated t
time for project work. A comb
simulation[1], and case studies
spite of the short time availab
taking advantage of the know
and academic environments.

One means of ensuring that
the spirit of Engineering Appl
with an industrial collaborato
ideas, about the development
to investigate. This may be du
in a particular area. The idea
successful that development e
or corporate decisions may si
particular product line.

In all these cases companie
product development idea by

This paper was first submitted in Ma