HEX

5000, OFFSETBASE D!

CR

100 0 DO HEX

I 6 * 1 + LCA 100 * 0 < IF

I 6 * 0 + LCA DUP F = IF ELSE

DUP E = IF I 6 * 1 + LCA 7F - 28 * DECIMAL

." 0.00" ABS ." V " HEX

I 6 * 4 + LCA 100 * I 6 * 5 + LCA + 32 /MOD DECIMAL ." ." 2 * .

." SECONDS " HEX

I 6 * 2 + LCA 100 * I 6 * 3 + LCA + 9 /MOD DECIMAL ." ."

HEX C * DECIMAL .

." % OF CYCLE"

CR ELSE

DUP D = IF I 6 * 1 + LCA 7F - 28 * DECIMAL

." 0.00" ABS ." V " HEX

I 6 * 4 + LCA 100 * I 6 * 5 + LCA + 32 /MOD DECIMAL ." ." 2 * .

." SECONDS " HEX

I 6 * 2 + LCA 100 * I 6 * 3 + LCA + 9 /MOD DECIMAL ." ."

HEX C * DECIMAL .

." % OF CYCLE"

CR ELSE

DUP B = IF I 6 * 1 + LCA 7F - F * DECIMAL

." 0." ABS ." V " HEX

I 6 * 4 + LCA 100 * I 6 * 5 + LCA + 32 /MOD DECIMAL ." ." 2 * .

." SECONDS " HEX

I 6 * 2 + LCA 100 * I 6 * 3 + LCA + 9 /MOD DECIMAL ." ."

HEX C * DECIMAL .

." % OF CYCLE"

CR ELSE
DUP 7 = IF 1 6 * 1 + LC9 7F = 4 /MOD DECIMAL

." 8 * ABS ." V " HEX
I 6 * 4 + LC9 100 * I 6 * 5 + LC9 + 32 /MOD DECIMAL ." ." 2 *
." SECONDS " HEX
I 6 * 2 + LC9 100 * I 6 * 3 + LC9 + 9 /MOD DECIMAL ." ." HEX C * DECIMAL .
." % OF CYCLE"
CR ELSE
THEN THEN THEN THEN THEN DROP
ELSE
I 6 * 0 + LC9 DUP F = IF ELSE
    DUP E = IF I 6 * 1 + LC9 7F = 22 * DECIMAL

." -0.00" ABS ." V " HEX
I 6 * 4 + LC9 100 * I 6 * 5 + LC9 + 32 /MOD DECIMAL ." ." 2 *
." SECONDS " HEX
I 6 * 2 + LC9 100 * I 6 * 3 + LC9 + 9 /MOD DECIMAL ." ." HEX C * DECIMAL .
." % OF CYCLE"
CR ELSE

DUP D = IF I 6 * 1 + LC9 7F = 28 * DECIMAL

." -0.00" ABS ." V " HEX
I 6 * 4 + LC9 100 * I 6 * 5 + LC9 + 32 /MOD DECIMAL ." ." 2 *
." SECONDS " HEX
I 6 * 2 + LC9 100 * I 6 * 3 + LC9 + 9 /MOD DECIMAL ." ." HEX C * DECIMAL .
." % OF CYCLE"
CR ELSE
DUP 6 = IF 16 * 1 + LCA 7F - 9 * DECIMAL

" -0.0" AB3 " V " HEX
16 * 4 + LCA 100 * 16 * 5 + LCA + 32 /MOD DECIMAL .. " 2 * .
" SECONDS " HEX
16 * 2 + LCA 100 * 16 * 3 + LCA + 9 /MOD DECIMAL .. " ."
HEX C * DECIMAL .
" % OF CYCLE" CR ELSE

DUP 7 = IF 16 * 1 + LCA 7F - 5 /MOD DECIMAL .

" ." 8 * ABS ." V " HEX
16 * 4 + LCA 100 * 16 * 5 + LCA + 32 /MOD DECIMAL .. " 2 * .
" SECONDS " HEX
16 * 2 + LCA 100 * 16 * 3 + LCA + 9 /MOD DECIMAL .. " ."
HEX C * DECIMAL .
" % OF CYCLE" CR ELSE

THEN THEN THEN THEN THEN DROP
THEN
LOOP ;

Table B3 Forth data manipulation program
APPENDIX F

The MC68000 Microprocessor System

The MC68000 microprocessor is a design from the UWTec division of the University of the Witwatersrand. It has been adapted so in order to accept data from the digital partial discharge detector and process this data.

The following are special qualities of the system:
- Stand-alone capability for development and maintenance
- Board physical size
- Low power dissipation
- Minimal printed circuit board construction
- Easily maintainable and expandable

The hardware and software of the kernel are considered in outline in the following sections.

1. Hardware Configuration

A printed circuit board of 120mm x 120mm size. The board contains:

1.1 microcomputer
1.2 interface of DIP DRAM (128K x 8)
1.3 timer circuits (8253)
1.4 host valve generator and watchdog circuit
1.5 RS232 interface board and counter

One serial port is sufficient to drive an ASCII interface.

2. Firmware on the Board

The firmware is pre-programmed with the following:

- Serial port driver and receiver software
- 8-bit timer
- 8-bit timer with 16-year timer driver
- Two time-out functions
- UPD interface
- Flp-flop language interpreter and compiler
The UWTec MC68000 Microprocessor

A single board microprocessor-based kernel microcomputer has been developed at UWTec. This board is intended as a fundamental building block for a number of research applications. The board contains all the necessary software and hardware to operate as a self-standing, maintainable unit. Various hardware and software enhancements are available or are under development that will tailor the basic design to specific tasks. A host computer, such as an IBM PC can be used to assist software development. The kernel communicates with the host via a serial 3-wire link. Programs for the host to handle this communication have been written in Pascal and BASIC. The kernel processor is programmable in Fig-FORTH. In order to facilitate a role in realtime industrial control research, the board contains a watchdog timer and a realtime clock.

The following are special features of the kernel:

- Standalone capability for development and maintenance.
- Small physical size.
- Low power dissipation.
- Single printed circuit board construction.
- Easily extensible in both hardware and software.

The hardware and software of the kernel are considered in outline in the following sections.

1. Hardware Configuration

A printed circuit board of 104mm x 220mm is used. This board contains:

1.1 A 16 bit CPU (MC68000).
1.2 16 kilobytes of EPROM (2 x 2764).
1.3 16 kilobytes of RAM (2 x 6264).
1.4 Reset pulse generator and watchdog timer.
1.5 Dual asynchronous serial port and counter (MC68681 DUART).

One serial port is buffered to drive an RS232c interface.

2. Firmware in the board

The EPROMs are preprogrammed with the following:

2.1 Serial port driver and receiver software.
2.2 Realtime clock and watchdog timer driver, operating from a 5ms DUART interrupt.
2.3 Fig-FORTH language interpreter and compiler.
3. User Enhancements to the kernel

Additional hardware and software may be added to the kernel to cater for specific applications. Areas currently under consideration are listed below:

3.1 Development of inputs and outputs (I/O).
3.2 Data logging.
3.3 Analogue control.
3.4 Local area networking.
3.5 Software programming systems.

All these areas will eventually form component layers of a Distributed Computer Control Systems node as shown below.

![Figure 1: DCCS Processor Node](image)

The present status of these various areas is outlined as follows:

3.1 Input/output.

A design for a low-cost analogue Input/output system has been developed. This allows for one 8-bit resolution analogue output in the range of 0 to 5 volts and six 8-bit analogue inputs. Each input can derive its signal from either a 0 to 5 volt source, or a differential signal from a passive resistance bridge, such as a strain gauge. Analogue to digital conversion is done using a software generated successive approximation algorithm. Conversion time is 300 microseconds per channel.
A second development has been to incorporate six 8-bit high speed analogue to digital converters (A/D) with the CPU kernel. Each A/D has its own instrumentation amplifier to enable low voltage differential signals to be measured. All six A/D converters operate in parallel at 15μS per conversion.

3.2 Data logging.

An original, memory efficient, data logging system has been developed to store up to 12 kilobytes of logged data from 6 analogue input channels per board. Maximum logging rate is 200 samples per second. Higher speeds are also being considered.

3.3 Analogue control.

A proportional-Integral-Derivative (PID) controller has been developed that can handle any number of control loops within the processor. Other types of controllers are under consideration.

3.4 Local Area Networking.

The addition of a Local Area Network processor, the Western Digital WD2840, has been carried out. Software and hardware development to provide LAN capabilities to the kernel is well advanced.

3.5 Software programming systems.

The FORTH system software is intended mainly as a maintenance and development tool for the kernel. Projects to enhance the software structure are underway and include the development of multi-tasking capabilities within the kernel and a multi-processor operating system.

4. Host Interfacing

The kernel is intended to operate in conjunction with a host computer. Since it contains no disc drives or equivalent bulk storage medium, it therefore must download its code from the other computer. The method of using an IBM PC as the host computer with a serial link between the host and the processor kernel has proved successful. Editing of the FORTH code is done on the host computer using any of the popular editing programs. During execution of these programs by the kernel, the host acts as an intelligent terminal and user interface.
Firmware Description

This section provides a description of the software features available on the UMTeC kernel microprocessor. It is intended for those who wish to make use of the FORTH programming system. Modification to the kernel firmware will require more detailed information. The following aspects of the kernel will be discussed:

1. FORTH environment
2. Realtime clock and internal timers
3. Serial interface
4. Reserved memory
5. Additional FORTH functions

1. FORTH Environment

The well known fig-FORTH programming language has been implemented in ROM form within the kernel microprocessor. All the standard FORTH words are present with the exception of disc related commands. The details of this language are not described here, but a summary of FORTH words can be found in the appendix.

2. Realtime clock and built in timers

The memory map (Figure 2) shows the location of the clock and timer registers. The clock is implemented as an interrupt-driven, 32bit counter. The interrupts occur at 5ms intervals. This interval is programmed on initialisation, but may be altered by setting new values in the DUART CTUR and CTLR registers. The interrupt vector is also set in the DUART, and may be altered with care.

3. Serial Interface

Only the A side of the DUART is used by the kernel. This has been configured as an asynchronous serial port with the following characteristics:
- 2400 baud,
- 7 data bits,
- 2 stop bits,
- no parity.

Routines to transmit and receive characters via this port are included in the FORTH language. The receive routine passes all characters to the FORTH except the following three:

- ^C (hex 03); this causes a warm reset equivalent to a hardware reset,
- ^S (hex 05); this inhibits the transmit routine for about 2 seconds,
- ^O (hex 06); this restarts the transmit routine immediately.
The only have an effect when the FORTH is accessing the DUART.
The second serial port is unconfigured and is available for user applications.

The initial DUART values are shown in the following table:

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR</td>
<td>F0H</td>
<td>TIMER MODE</td>
</tr>
<tr>
<td>CRA</td>
<td>10H</td>
<td>DISABLE UARTS</td>
</tr>
<tr>
<td>MRA1</td>
<td>16H</td>
<td>SET MRA1</td>
</tr>
<tr>
<td>MRA2</td>
<td>0FH</td>
<td>SET MRB1</td>
</tr>
<tr>
<td>CSRA</td>
<td>88H</td>
<td>2400 BAUD</td>
</tr>
<tr>
<td>CRA</td>
<td>05H</td>
<td>ENABLE UART A</td>
</tr>
<tr>
<td>CTUR</td>
<td>02H</td>
<td>SET 5mS INTERRUPT</td>
</tr>
<tr>
<td>CTRL</td>
<td>2FH</td>
<td></td>
</tr>
<tr>
<td>IVR</td>
<td>40H</td>
<td>SET INT. VECTOR</td>
</tr>
<tr>
<td>IMR</td>
<td>08H</td>
<td>ENABLE TIMER INT.</td>
</tr>
</tbody>
</table>

4. Reserved memory

The usage of the system RAM and ROM is shown in Figure 3. A limited amount of RAM is allocated to the FORTH system, and a large block is available for user applications. Timer 2 (addr. 7FF2H) is available for user applications. The value in this memory location will decrement to zero from any preset value in 5ms ticks. Once zero is reached, decrementing will stop.
FIGURE 3: System memory usage

- Real-time clock
- Real-time clock
- Timer X
- Timer 1
- Timer 2

Reserved

<table>
<thead>
<tr>
<th>hours</th>
<th>mins</th>
<th>tenths</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000H</td>
<td>7FF0H</td>
<td>7FF1H</td>
</tr>
<tr>
<td>4400H</td>
<td>7FF2H</td>
<td>7FF3H</td>
</tr>
<tr>
<td>4300H</td>
<td>7FF4H</td>
<td>7FF5H</td>
</tr>
<tr>
<td>4200H</td>
<td>7FF6H</td>
<td>7FF7H</td>
</tr>
</tbody>
</table>

Main program register store

Interrupt program register store

User RAM area
(12 kilobytes)

FORTH RAM

Return stack

Computational stack

FORTH user variables

Scratch pad

FORTH ROM

System interrupt vectors

5000H

4400H

4300H

4200H

4010H

4000H

0400H

0000H
5. Additions to FORTH

A number of routines have been added to the FORTH language to provide additional features to the kernel. These are:

**OFFSETBASE**
This is a double word variable used in long address accesses and stores in the 68000.

**LC1, LC0**
These are equivalent to CI and CO, but use the value in OFFSETBASE as an address offset.

**CIT**
This sets up the backspace character for the CIT terminal.

**DT**

This returns the 32 bit realtime value to the stack as a double word.

**.CLOCK**
This routine prints out the time from a double word on the stack in hours, mins, secs and tenths format. The values for hours, mins, secs and tenths are also stored in their respective memory locations.

**CLOCK**
This routine prints out the time using DT and .CLOCK.

**SCLK**
This routine converts the 3 separate time components to a double word on the stack.

**RSCLK**
This routine sets the realtime clock.

**RUNMC**
This routine causes execution of machine code starting at address 'addr'.
